STATISTICAL TOOLS FOR EFFICIENT CONFIRMATION OF DIAGNOSIS IN PATIENTS WITH SUSPECTED PRIMARY CENTRAL NERVOUS SYSTEM VASCULITIS

John Brooks

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
Master of Science Epidemiology Specialization in Biostatistics

School of Epidemiology and Public Health
Faculty of Medicine
University of Ottawa

Abstract

The management of missing data is a major concern in classification model generation in all fields but poses a particular challenge in situations where there is only a small quantity of sparse data available. In the field of medicine, this is not an uncommon problem. While widely subscribed methodologies like logistic regression can, with minor modifications and potentially much labor, provide reasonable insights from the larger and less sparse datasets that are anticipated when analyzing diagnosis of common conditions, there are a multitude of rare conditions of interest. Primary angiitis of the central nervous system (PACNS) is a rare but devastating entity that given its range of presenting symptoms can be suspected in a variety of circumstances. It unfortunately continues to be a diagnosis that is hard to make. Aside from some general frameworks, there isn't a rigorously defined diagnostic approach as is the case in other more common neuroinflammatory conditions like multiple sclerosis. Instead, clinicians currently rely on experience and clinical judgement to guide the reasonable exclusion of potential inciting entities and mimickers.[i] In effect this results in a smaller quantity of heterogenous that may not optimally suited for more traditional classification methodology (e.g., logistic regression) without substantial contemplation and justification of appropriate data cleaning / preprocessing. It is therefore challenging to make and analyze systematic approaches that could direct clinicians in a way that standardizes patient care.

In this thesis, a machine learning approach was presented to derive quantitatively justified insights into the factors that are most important to consider during the diagnostic process to identify conditions like PACNS. Modern categorization techniques (i.e., random forest and support vector machines) were used to generate diagnostic models identifying cases of PACNS from which key elements of diagnostic importance could be identified. A novel variant of a random forest (RF) approach was also demonstrated as a means of managing missing data in a small sample, a significant problem encountered when exploring data on rare conditions without clear diagnostic frameworks. A reduced need to hypothesize the reasons for missingness when generating and applying the novel variant was discussed. The application of such tools to diagnostic model generation of PACNS and other rare and / or emerging diseases and provide objective feedback was explored. This primarily centered around a structured assessment on how to prioritize testing to rapidly rule out conditions that require alternative management and could be used to support future guidelines to optimize the care of these patients.

The material presented herein had three components. The first centered around the example of PACNS. It described, in detail, an example of a relevant medical condition and explores why the data is both rare and sparse. Furthermore, the reasons for the sparsity are heterogeneous or non-monotonic (i.e., not conducive to modelling with a singular model). This component concludes with a search for candidate variables to diagnose the condition by means of scoping review for subsequent comparative demonstration of the novel variant of random forest construction that was proposed. The second component discussed machine learning model development and simulates data with varying degrees and patterns of missingness to demonstrate how the models could be applied to data with properties like what would be expected of PACNS related data. Finally, described techniques were applied to separate a subset of patients with suspected PACNS from those with diagnosed PACNS using institutional data and proposes future study to expand upon and ultimately verify these insights. Further development of the novel random forest approach is also discussed.

Contents

I. Overview of Thesis

A. Goal

The central goal of this thesis was to justify the development of a classification tool to address a clinical question regarding a rare entity (i.e., PACNS) in a way that could facilitate future guideline development. It first demonstrated that there is a purpose for such a tool by presenting a question of practical significance (i.e., developing a best approach to PACNS diagnosis) that is tied to data that is rare, sparse and what is argued not well suited to traditional approaches of managing missing data, particularly with regards to discerning the importance of various characterizing features. It was then shown how the new tool addressed several missing data concerns compared to contemporary methods using simulated, stock (i.e., the wine data set) and real-world data (i.e., the PACNS data set). Potential modeling of PACNS diagnosis and the relative importance of relevant features was then explored as an illustrative example.

B. Components

a. The Motivating Question

"In patients with suspected central nervous system (CNS) vasculitis, how does one best predict ultimate diagnosis of PACNS?" The thesis and its cardinal clinical components (a scoping review of the literature to identify candidate variables and an institutional case review to inform generation of a model to predict the diagnosis of PACNS) considered, as a starting point, how one investigated patients with the suspicion that they may have a rare disease such as PACNS. The clinical review illustrated the complexity of a data set that would arise from such a process and set the stage for subsequent comparative analysis regarding the results of applying several candidate classification techniques. The analysis focused on model accuracy and capacity of reporting to users the importance of the features drawn on by the proposed models.

b. Methodology Development

The thesis outlines approaches taken to address missing data, a principal problem posed by the PACNS dataset, that can be applied during data pre-processing / cleaning where data cleaning is defined as the compensatory process of addressing data deficiencies like missingness or error. A discussion of classification techniques ensued, including principally SVM and random forest, that are used to form a diagnostic model for a condition like PACNS. A novel random forest design was then proposed and the properties of this model relative to established models was discussed. All applied classification approaches models and their relative susceptibilities to various forms of missingness was addressed in subsequent discussion.

c. Relative Model Performance

Relative classification model performance was compared with a focus on two properties namely classification accuracy and the appraisal of relative feature importance to the models.

Random forest and variant random forest, support vector machines, and logistic regression were applied to purely simulated, established (i.e., wine data set) and a real world (i.e., PACNS data set) data sets to compare model accuracy in a context where a significant proportion of the data was missing (i.e., as would be expected for the diagnosis of PACNS). Model accuracy was primarily compared head-to-head using the area under the receiver operator curves (AUROC).

Concepts in the evaluation of relative feature importance were discussed and the models were also compared with respect to estimation of feature importance using Gini importance as well as permutation importance. In the simulated data set, missing data with varying properties to simulate missingness completely at random, missingness at random and missingness not at random was introduced and analysis of the change in relative feature importance was made. This was repeated for the wine data set. With regards to feature importance, measures were first taken on simulated data where relative feature importance could be objectively appreciated. Both simulated and wine data sets had feature importance measured before and after the addition of missing data to demonstrate the bias introduced on the measures of feature importance in the presence of varying types of missingness.

Within the clinical population of interest (i.e., those with possible PACNS), the approach taken to diagnosis / specific diagnostic testing that was performed to partition patients with primary angiitis of the central nervous system from other diagnoses was explored. A slate of features / factors / tests / indicators previously described as candidates to identify a patient afflicted with PACNS versus not were used to generate a predictive model and relative feature importance was explored. A process of subsequent expansion and validation was discussed.

II. Structure

This thesis demonstrated the use of machine learning, including a novel adaptation of a random forest model, to facilitate the generation of a rigorously justified diagnostic process / guideline in PACNS. The presented thesis has 5 foci / chapters within the aforementioned sections including: (1) a scoping review of the literature pertaining to diagnosis of PACNS / excluding alternative diagnoses where PACNS is potentially afflicting a patient; (2) a review of an approach to missing data with a focus on the context of random forest use; (3) multiple candidate model discussion that could be applied to the diagnosis of PACNS and the development of new technique that addresses many of the needs for classification in that data set, (4) comparative analysis of these derived models (i.e. relative diagnostic accuracy when comparing predictions made from a random forest model versus a novel random forest versus a support vector machine algorithm); (5) record extraction from the Ottawa Hospital data warehouse where records were queried and reviewed to extract testing and diagnostic conclusions regarding patients evaluated for suspected PACNS. The thesis concludes with a summary a proposal of a larger study in which a useful model was constructed and validated.

A principal component of the thesis was evaluating the insights obtained when applying various classification approaches to the data. The overall flow of data processing was summarized in Figure 1. Three data sets were analyzed (i.e., simulated data, wine data set, and the PACNS dataset) and four models to each applied including a novel random forest, a regular random forest (i.e., standard / stock random forests where the standard random forest was a novel random forest run on fully imputed data which was equivalent to applying the scikit built in “stock” random forest), SVM and logistic regression. In all, 12 models in total were created to compare the performance and insights of the models across the 3 data sets. Comparisons between the results of logistic regression, SVM, standard RF, and novel RF were discussed. In some simulations a stock versus standard random forest model was specified which reflects that either the preprogramed packed distributed with scikit learn was used or the manual coding used to

operate the novel random forest design run to emulate the preprogramed package (i.e., the novel code run with imputed values). This was merely to demonstrate the validity of programming.

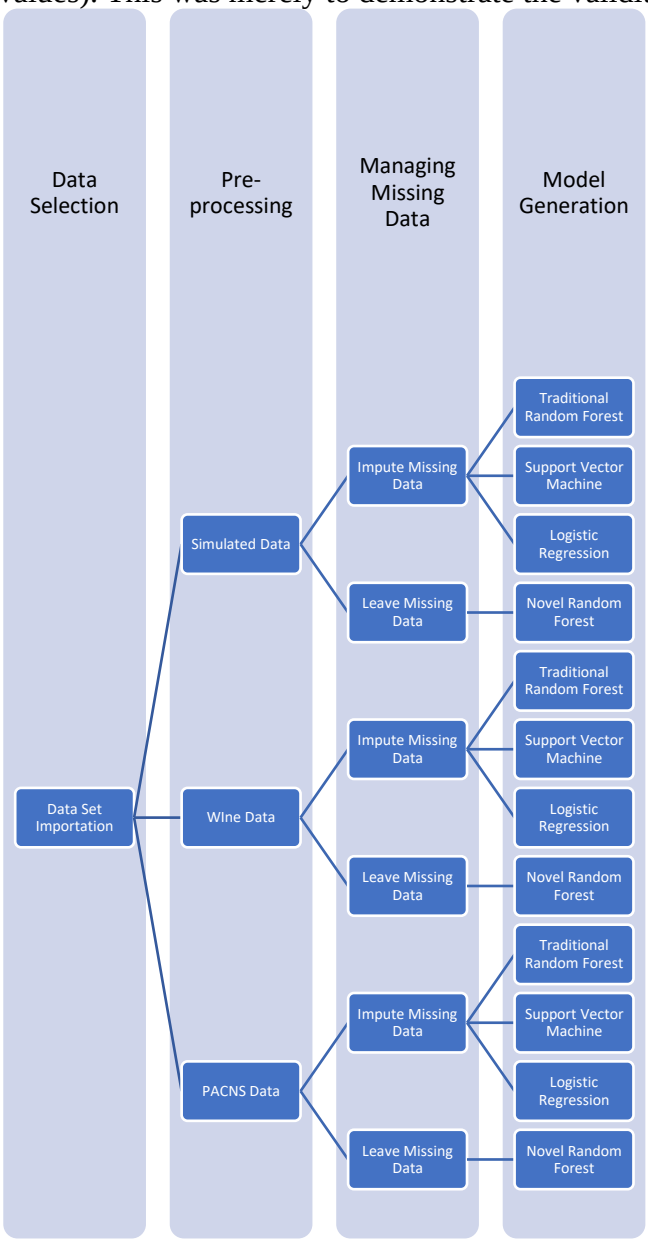


Figure 1. Summary of the modelling for comparing statistical approaches.

III. Introduction

PACNS, being a clinical entity that has evolved substantially since its original description in the 1950's, has seen significant changes to the understanding of the condition and diagnostic practices applied when suspected (e.g., available testing, imaging techniques, associated / exclusionary pathological hallmarks, etc.).[ii],[iii],[iv] This in turn has made reaching a consensus regarding the diagnostic approach challenging. At its core, PACNS is a diagnosis that represents inflammation of blood vessels within the brain and spine (i.e., Vasculitis / Angiitis of the CNS) without evidence of significant inflammation in blood vessels outside of the brain and absent

must be any clear impetus from a systemic condition causing blood vessel inflammation (e.g., Granulomatosis with Polyangiitis, Eosinophilic Granulomatosis with Polyangiitis, Behcet disease, etc.).[v],[vi] Afflicted patients will either have sudden changes to their function like numbness, weakness and/or vision change from a stroke, changes to their thinking / ability to process information and interact with others (i.e., cognitive change), and/or epileptic seizure.[vii] Fortunately, the condition is rare with an estimated incidence of 2.4 cases per million person-years and this peeks at age 50 with males more often to be afflicted than females.[viii] Unfortunately, there are many conditions that can cause any one or a constellation of the cardinal symptoms and so recognizing PACNS is not straightforward.[ix] This problem is highly relevant as coming to a diagnosis to facilitate effective treatment is paramount for good prognosis and the good prognosis in those with mimicking conditions.[x]

More practically, diagnosis of PACNS is one that is in large part exclusionary (i.e., one must exclude that a systemic vasculitis / angiitis or infection for example is the source), and awareness of similar alternative diagnoses and available tests have changed over time.[xi] Resultantly, the collective data that has been drawn on to comment on when to pursue and then make the diagnosis over the years is heterogeneous and for some candidate variables that aid in case identification values are sparse (e.g., pathology stains for amyloid, spinal testing for protein like tau and beta amyloid, and the use of susceptibility weighted imaging have become available to diagnose another entity, amyloid beta related angiitis, separate from PACNS since its original description).[xii],[xiii] This is not the only reason for such sparsity. Without a dominant, stable consensus on diagnostic approach, data generated on these diagnostic conundrums is shaped substantially by the preferences of individual investigators pursuing an individual patient's case.[xiv],[xv],[xvi] Fundamentally, each clinician has their own body of experience directing him / her / them and may even have restrictions on the testing that can be ordered including the conclusions that can be drawn from that testing. Obviously, the testing a doctor at the Mayo clinic has access to differs substantially from what one at a community hospital, or even the Ottawa Hospital can perform and that has the potential to greatly impact diagnostic practice (e.g., with respect to PACNS the availability of black blood protocol imaging including the expertise required for its interpretation differs by location of care).[xvii],[xviii] The complexity of the resultantly heterogeneous data and inferences resulting from that data has led largely to a reliance on expert experience and opinion to provide some standardization.

Tools exist to present more objective support for expertise generated heuristics such as through a modified Delphi approach, however the result still relies on expert opinion without a necessarily clearly laid out foundational knowledge that can be readily critiqued over time.[xix] While traditional statistical techniques like logistic regression can formalize inferences directly from clinical data at scale in an easily digestible way, conditions such as primary CNS vasculitis (i.e., PACNS) are rare and issues around data sparsity and collinearity leave such techniques at a disadvantage.[xx, xxi, xxii, xxiii] Discarding or imputing much of the data where only a smaller amount of data exists at the outset can impair model convergence, generalizability and robustness.[xxiv],[xxv] Neural networks on the other hand can seamlessly wrangle data and produce accurate results from the most unruly of sources although large samples are typically needed to effectively "train" such models.[xxvi] They also unfortunately achieve their insights with an opacity that thwarts the reciprocal learning of clinicians needed to then develop guidelines for other clinicians.[xxvii] Machine learning, with techniques like support vector machines (SVMs) or

random forests (RFs), presents an opportunity to provide feedback to clinicians on the consistency of their conclusions while maintaining a decipherable institutional legacy that can be used to train future practitioners.

As a primer, the technical aspects / descriptions of the techniques that are later applied are presented in the following subsections.

A. Techniques

1. Random Forest

Random forests are a classification technique that relies on combining the results of many decision trees. The critical element is that there is an aggregation of the results of more basic and varied subunits and there are several variations that exist with regards to the construction of the subunits and how to aggregate them. For reference, the pseudocode for two popular incarnations of random forests are presented and then the latter approach (i.e., the approach employed by the python / scikit statistics package used in subsequent analysis during the thesis) discussed in more detail in the ensuing subsections.

A random forest is constructed as follows where the variability is centered about the nodes of trees that compose the forest (Random Forest Nodal Pseudocode):[xxviii]

1. Model Development
    1. Input
        1. Matrix of n cases with m features (i.e., $n \times m$ matrix)
    2. Construct a Tree
        1. Randomly select k features from the m available features
        2. Compute for all k features the best threshold to split the data presented to a given node
        3. Use the threshold that creates the greatest drop in the selected metric of purity (e.g., entropy, Gini importance) to split the data into daughter nodes.
        4. Repeat steps 1 to 3 on daughter nodes until a model parameter has been exceeded. Such parameters could include:
            1. The number of cases / samples that each daughter node must possess (i.e., the minimum number of leaf samples)
            2. The minimum number of cases available at a node to consider a split
            3. The minimum change / decrease in impurity required
            4. The maximum tree depth (i.e., the maximum number of chained nodes).
    3. Build a Forest
        1. Repeat steps 1 to 4 contained within "Construct a Tree" t times to create t trees
2. Prediction (Hard Voting)
    1. Apply sample data each of the n trees to create n predictions
    2. Assign a class to the sample of interest that reflects the class predicted by the majority of the n trees.

This is a popular approach, however this approach differs from that is set forth in scikit learn where the variability in the trees is created by portioning the data into feature subspaces (Stock Random Forest Pseudocode):[xxix, xxx]

1. Model Development / Training
    1. Input
        1. Matrix of n cases with m features (i.e., $n \times m$ matrix)
    2. Construct a Tree (i.e., construct the ith tree within an ensemble of t trees)
        1. Randomly select l features from the m available features
        2. Compute for all l features the best threshold to split the data presented to a given node
        3. Use the threshold that creates the greatest drop in the selected metric of purity (e.g., entropy, Gini importance) to split the data into daughter nodes.
        4. Repeat steps 2 to 4 on daughter nodes until a model parameter has been exceeded as described in the random forest nodal pseudocode.
    3. Build a Forest
        1. Repeat steps 1 to 5 contained within "Construct a Tree" t times to create t trees
2. Importance Computation
    1. Compute m summations over all the nodes within all the trees of the purity changes associated with each of the m features and divide by the total number of trees.
3. Prediction
    1. Apply sample data each of the n trees to create n predictions
    2. Assign a class to the sample of interest that reflects the class predicted by the majority of the t trees with valid votes.

Note that in the context of random forest regression, which was discussed in section V, the votes are not for a class but instead for a variable value that is being regressed. It is as if there were a continuum of classes with a numeric value that could be predicted by any one tree within the random forest. In that case some aggregate prediction (e.g., simple average of the results of all trees within the forest) is the result of the regression.[xxxi]

The process of creating a random forest model starts with generation of a "learner" or a single exhaustive / nearly exhaustive decision tree using a randomly selected / bootstrapped sample from a supplied dataset like the 2,600+ data points (collections of patient data) extracted from TOH databased patients that have first ANCA tests with one month of CSF sampling (see Figure 2 for an example of two "learners" or decision trees) or a complete case subset (i.e. a sub-set of the data where no cases are missing the data of interest). The available descriptive variables (i.e., features) for those selected samples are then used to split up these sample members to maximize order across each new division (i.e. reduced [disorder] entropy or increased enthalpy [order]). Because the tree is very detailed it can often cycle through many variables to create perfect splits or near perfect splits every time given the data used to construct it (reduced bias) but new data (e.g. "out-of-bag" data which is discussed later) presented to the decision tree has a necessarily higher probability of being misclassified (higher variance) particularly when it is of a type not

considered in the original construction (e.g. consider if a case of delirium was posed to Figure 2a). The solution principally applied by random forests to address this issue is bagging (i.e., bootstrap aggregating) or to "bag" these decision trees together in some way that on average, when presented with new data, comes more reliably to the correct conclusion (i.e. bagging can be used to reduce variance without altering bias).[xxxii] Often this is achieved by having hundreds or thousands of decision trees, each constructed with training sets (i.e., randomly drawn case sets from the n original cases) that review the data and render their own prediction on how to classify the data which is then subsequently amalgamated across trees. In the simplest of cases, the "hard" majority (see Figure 3) represents the prevailing classification. By contrast "soft" voting allows trees within the model to indicate probabilities of certain classifications that are then tallied (with or without differential weights for individual classifiers) for a final determination.[xxxiii]

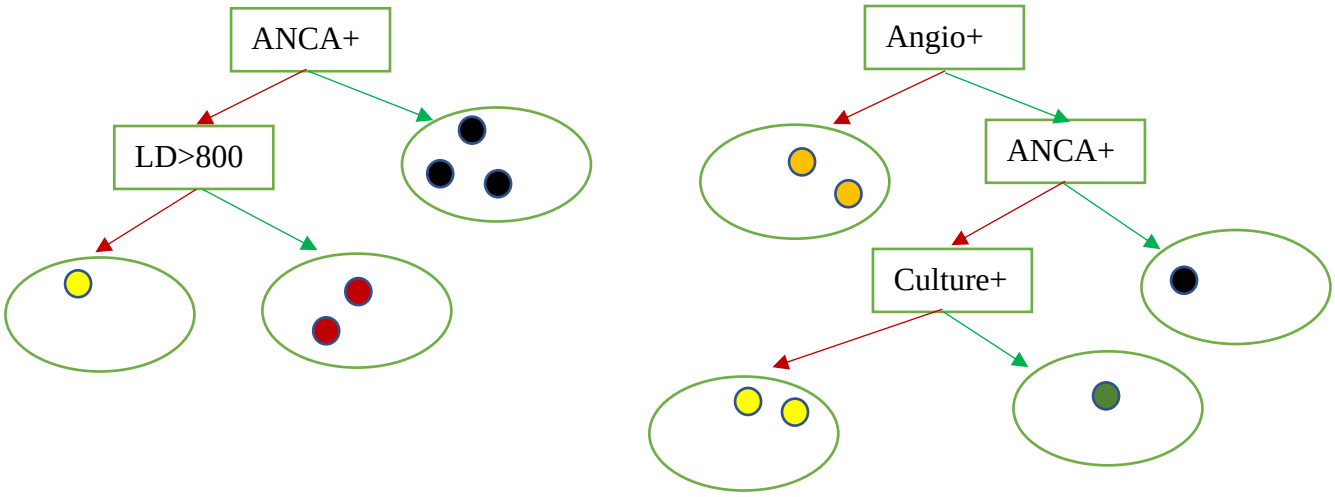


Figure 2. Decision trees. These are examples of decision trees that split PACNS (yellow balls), lymphoma (red balls), granulomatous polyangiitis (black balls), delirium (orange balls), and infection (green balls). Decision nodes are the green rectangles, leaf nodes are the green ovals, and root nodes are not shown but would contain all the items for sorting (i.e. balls) prior to them going through the sorting process. Red arrows indicate the decision criteria (e.g. that the LD be greater than 800) was not satisfied and the green arrows indicate that it was. ANCA: Anti-neutrophil cytoplasmic antibody, Angio: Angiography, PACNS: Primary angiitis of the central nervous system.

Performance of the trees is determined in various ways. One is to consider an "out of bag error" which looks at, for every decision tree, how well it classifies the sub-set data that was available but was not drawn from during bootstrapping to create the trees. Another way is to take the "test" data, which may have been previously separated from the training data as discussed and use that to develop estimates of sensitivity, specificity, and accuracy.

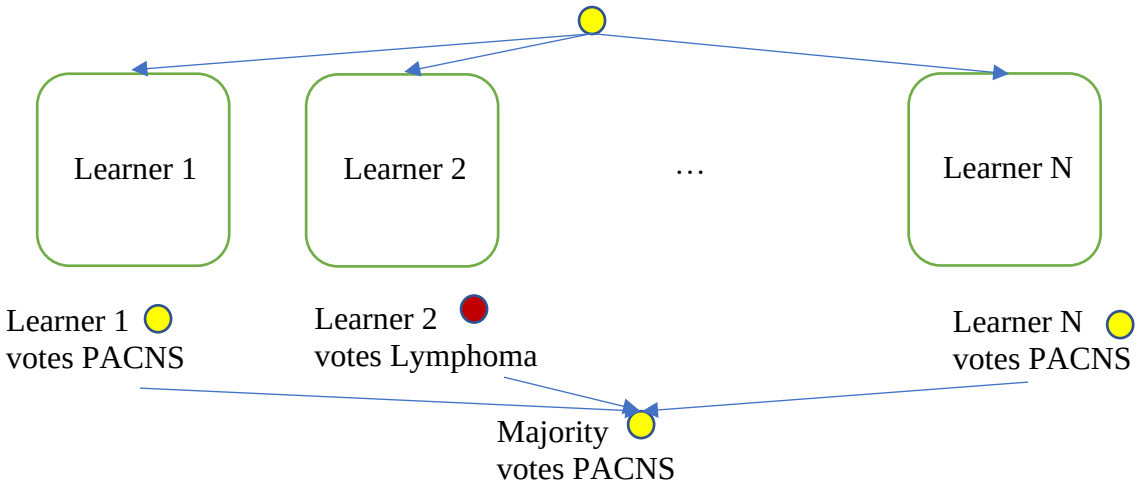


Figure 3. Voting scheme for learners

The important feature of random forest leveraging the combination of many composite models with low bias (i.e., the trees) to reduce variance in many cases without impacting bias is a clear benefit of the technique. Unfortunately input data characteristics may introduce a significant amount of bias of their own as is seen in some forms of missing data.

2. Novel Variant

In the presence of missing data, it is useful to consider techniques that limit the losses of available input data that could increase model error / bias (Note missing data will be discussed in more detail in subsequent sections) while not overstating one's knowledge of the missing data which could introduce another form of bias. The impetus for creating the novel variant random forest (summarized here with a more detailed description in section VI) was inspired by a practical consideration that is best viewed by how one might hope a diagnostician (e.g., doctor) might respond to being presented a case for diagnosis by a known colleague where all the F relevant features relevant to the diagnostician as it pertains to a particular diagnosis can be tabulated. Now consider that the colleague presenting the case missed a single value of importance (see Figure 4).

Feature 1	Feature 2	…	Feature F
?	84	…	30

Figure 4. Feature table for a case presented from a colleague to a diagnostician with a single missing variable of relevance.

If the diagnostician is clever, and they have faith in their colleague's diagnostic abilities, they may consider if the tendency to miss a certain value indicates their colleague's own belief there exists a particular diagnosis. This may then sway the diagnostician's final prediction in addition to the various feature values. However, what if the diagnostician is told it was from one colleague with certain tendencies when in fact it came from another with dissimilar tendencies, perhaps one the diagnostician did not know at all and/or was at a different stage in their investigations? The vote may well then be unduly biased.

If the diagnostician were then blinded as to which colleague was presenting the data (and they were told not to try to deduce the colleague presenting the case by the missing data pattern), there could be several other ways the diagnostician responds to that missingness. They could fill this with an average value or a value similar to a feature value that cases with similar other feature values have presented with before. This may seem quite reasonable but what if a large number of variables are missing (e.g., in the extreme case, all but one value is missing – see Figure 5)?

Feature 1	Feature 2	…	Feature F
5	?	…	?

Figure 5. Feature table for a case presented from a colleague to a diagnostician with multiple missing variables of relevance

If a similar approach were used the analysis of the case will heavily gravitate toward some mean value, either of the diagnostician body of experience as a whole or to a large subset of their experience. It might then be of value to draw upon their experience of similar situations and not use that to fill in the blanks and then make a prediction but to make the prediction directly (i.e., the novel random forest approach). It might also be of value to use some combination of these approaches (i.e., note that the approach for various features and cases could vary depending on the perceptions of the validity of various insight or imputation techniques) although further discussion will assume that one has decided to apply one approach or the other to the data.

In the training of traditional random forests, all data entries must be filled either by dismissing cases with any incomplete data or by imputing the missing data. If there is a high degree of missingness in a dataset, then complete case analysis may leave little data for use in the model. In some cases, where the number of features is vast and even modest amounts of missingness present per variable there may be no data left to use at all. Bias may also be introduced when the missingness of data is dependent on classification (e.g., if one is trying to identify smokers from their answers on questionnaires then a concern of judgmental perceptions may lead to omissions of predictive answers to certain questions).[xxxiv, xxxv] Imputation may address this although an appropriate model for the imputation would need to be selected. If multiple imputation were used for example, it would suppose that the data is homogenous enough that the available data and patterns of missingness could accurately specify a plausible distribution to direct the filling of the missing data.[xxxvi] As an alternative to complete case analysis and imputation, consider the following structure of a proposed novel random forest (Novel Random Forest Pseudocode) where data must be complete for each individual tree generation with select variable at training time and prediction only draws on those trees presented with complete data:

1. Model Development / Training
    1. Input
        1. Matrix of n cases with m features (i.e., $n \times m$ matrix)
    2. Construct a Tree (i.e., construct the ith tree within an ensemble of t trees)
        1. Randomly select l features from the m available features
        2. With a training dataset of size n_T, take only those n_{ci} cases that possess no missing values across the l selected features for training of the ith tree

3. If the cases represent only a single class or there are fewer than 3 cases, then register a "lock-out" by adding a value of 1 to the appropriate rows and columns indicating the features / feature combinations present during the "lock-out" (e.g., if the prospective tree contained features one, two, and three then a 1 would be added to row/columns of lock-out matrix $lock_{out}$ one/one, two/two, three/three, one/two, two/one, one/three, three/one, two/three, three/two) and return to step 1.
4. Compute for all l features the best threshold to split the data presented to a given node
5. Use the threshold that creates the greatest change in the selected metric of purity (e.g., entropy, Gini importance) to split the data into daughter nodes.
6. Repeat steps 4 to 5 on daughter nodes until a model parameter has been exceeded as described in the random forest pseudocode.
7. Register a "non-lock-out" by adding a value of 1 to the appropriate rows and columns indicating the features / feature combinations present during the "non-lock-out" / successful tree creation (e.g., if the prospective tree contained features one, two, and three then a 1 would be added to row/columns of non-lock-out matrix $lock_{non}$ one/one, two/two, three/three, one/two, two/one, one/three, three/one, two/three, three/two)

3. Build a Forest
    1. Repeat steps 1 to 7 contained within "Construct a Tree" t times to create t trees

2. Importance Computation
    1. Compute m summations over all the nodes within all the trees of the purity changes associated with each of the m features and divide by the total number of trees.
    2. Multiply each of the m summations (i.e., the m feature importance measures) by the trace of $lock_{non}$ and divide by the associated diagonal term (i.e., for the i^{th} feature importance divide by the term in $lock_{non}$ at row/column i/i)
3. Prediction
    1. Apply sample data each of the n trees to create n predictions
    2. In those trees that were not able to provide a valid vote because they relied on features that were unavailable, discount their votes
    3. Assign a class to the sample of interest that reflects the class predicted by the majority of the t trees with valid votes.
    4. In the event of a tie or in the event of total abstention, the default class (e.g., non-PACNS) was submitted as the result of the voting procedure.

The major novelty of the novel random forest variant that was proposed is that it allows for vote abstention. Trees are trained using complete data given a set of randomly selected features and when voting takes place using multiple trees, the trees that would otherwise draw on data that is missing in the case at hand abstain from voting. For illustrative purposes, consider the following training example with incomplete data where features 1, 3 and 4 are selected:

$$\begin{pmatrix} \_ & 0.24 & \_ & 0.18 & C_1 \\ 0.36 & 0.53 & 0.97 & 0.78 & C_1 \\ 0.79 & 0.76 & 0.34 & 0.57 & C_1 \\ \_ & 0.44 & 0.84 & 0.55 & C_1 \\ 0.32 & 0.51 & 0.34 & 1 & C_2 \\ 0.33 & 0.02 & \_ & 0.62 & C_2 \\ 0.54 & 0.40 & \_ & 0.92 & C_2 \\ 0.45 & 0.03 & \_ & 0.73 & C_2 \\ 0.62 & \_ & \_ & 0.19 & C_2 \end{pmatrix} \rightarrow \begin{pmatrix} 0.36 & 0.97 & 0.78 & C_1 \\ 0.79 & 0.34 & 0.57 & C_1 \\ 0.32 & 0.34 & 1 & C_2 \end{pmatrix} \rightarrow \begin{pmatrix} 0.32 & 0.34 & 1 & C_2 \\ 0.32 & 0.34 & 1 & C_2 \\ 0.32 & 0.34 & 1 & C_2 \\ 0.36 & 0.97 & 0.78 & C_1 \\ 0.36 & 0.97 & 0.78 & C_1 \\ 0.36 & 0.97 & 0.78 & C_1 \\ 0.32 & 0.34 & 1 & C_2 \\ 0.36 & 0.97 & 0.78 & C_1 \\ 0.32 & 0.34 & 1 & C_2 \end{pmatrix}$$

The usable data in this case has been reduced to a 3 by 3 matrix with an additional classifier column (i.e., from 9 complete cases the data to be used has been reduced to 3). From that a bootstrapped sample regenerated a 9-case matrix where one of the complete cases was not selected (out of bag). A decision tree could be generated from that bootstrapped matrix and form a single tree in the ensemble (i.e., the random forest). Now when time comes to characterize a case, consider an ensemble of 5 trees where they were built as follows: tree 1 – features 1, 3, and 4; tree 2 – features 2 and 4; tree 3 – features 2, 3, and 4; tree 4 – features 1, and 2; tree 5 – feature 3 alone. If we were to a have a case (0.40, 0.32, _, 0.31 | ?) then, with feature 3 missing, only trees 2 and 4 would vote and if 50% (at least one vote) implicated the class of interest, then the result would be classifying the result as the class of interest.

There were several advantages to this scheme. First, the learners were constructed using data they have, and thus imputation is not necessary (although still technically possible) to provide more data for modelling than would be available in strict complete case analysis. This is potentially useful when considering diagnostics for patients where the tests applied may vary greatly from practitioner to practitioner although techniques may still be available to successfully navigate random or non-random missingness. While imputation with auxiliary variables might address this to some degree, data later fed into the algorithm for a prediction may have missing entries for reasons totally different than the data used to create the model (e.g., all cases used in the model are retrospective and all testing deemed appropriate collected and resulted / testing complete where some clinicians may in practice try the model that is at an earlier stage of investigation). In effect, the novel method creates dynamic vote weighting whereby abstention from trees that were trained with a feature value absent in the subsequently presented data are censored and there is a seamless shunt / heavier weight placed on the votes of those trees that possessed the available variables. Second, like random forests in general, the scheme is more accurate than implementing a single decision tree which is also limited in terms of the patients that can be sorted by it if relevant data is missing. Third, insights can be gained into which variables / features can create the greatest clarity (i.e., Gini Impurity reduction, a concept that is reviewed in detail in section III. B. 2. a.) when they are applied (i.e., lead to the least impurity / most reduced misclassification in leaf nodes). This variable importance measure in the context of missing variable is also more straightforward than some incarnation of previously proposed random forests (i.e., the nodal technique) as importance metrics could be recomputed by summing values over the remaining trees when those trees that rely on missing values are rejected. Finally, again like random forests in general, predictions can be made for more than just binary outcomes if desired (i.e., instead of a restriction to PACNS versus non-PACNS one can classify infectious causes of vasculitis and look at performance).

There are some drawbacks, however. Unlike a single decision tree where one can print and disseminate an easy to interpret algorithm, a novel random forest, and random forests in general, are composites of many of these trees and so only a summary of the component parts is readily available to describe the composite trees. Therefore, when an algorithm provides a potential classification, it is not a decision that can be easily explained / justified to the user of the model (e.g., if a PACNS diagnosis were made, the insights into how that classification was arrived at cannot be directly extracted from the model).

Of note, the novel variant of the random forest does suffer from an eventuality which does not occur in the case of the standard random forest design. While, so long as there were enough cases with different classes to ascertain a suitable division criterion, it would be highly unusual to select a basket of features whose values were uninformative for a division (i.e., no variability in the feature values) and furthermore it would not be possible for a random selection of such features at a perspective node to be made recurrently without the dataset being wholly uninformative. Eventually the random forest algorithm would invite an informative feature to assist in the division at a node. In the novel variant of the random forest, it was possible for a grouping of l features to be selected such that the values of the features are uninformative or there were not enough cases with a different class to warrant generation of a tree (e.g., all remaining cases had the same class and thus a null tree or a tree with no nodes was formed). In this event, a lock-out event was logged with the variables selected during the lock-out and another attempt was made to create a non-null ith tree. There is an important consequence of this, however. Instead of obtaining an even distribution of trees that had the opportunity to draw from a certain feature, those features that often we included in locked-out trees (typically with higher quantities of missing data) were less available to participate in the formation of trees in the final model. Thus, when Gini importance is calculated, fewer trees commented on the importance of sparser features and summing over all trees, other features had more estimates to add together. To compensate for this, a correction factor was introduced to Gini importance measures which was the inverse proportion of variable representation in non-locked out trees. This in effect adjusted Gini importance to estimate what the average Gini importance is in those trees that possessed a given feature.

Related to the concept of lock-out is one of class-skewing (not to be confused with a later concept explored referred to proportioning). In the case of class-skewing, it is not the number of opportunities a feature had to be included in a tree and subsequently accrue importance, but it is a shift in the proportion of classes available to divide the sample that might bias an importance estimate. This is related to a criticism of complete case analysis where bias is introduced into a model when the elimination of cases is related to classification outcomes (i.e., where case elimination alters the proportion of classification outcomes in the training set).[xxxvii] Consider the case where missingness were value related and that feature's value was associated with classification. Complete data sets featuring a variable would be disadvantaged by the asymmetry in classification of interest versus not (i.e., using the cases where the feature value of interest was present would lead to a different proportion of target classes available than if there were no association of missingness and the feature value). In the case of such features, the maximum Gini impurity and thus the maximum amount by which Gini impurity could drop (i.e., the maximum importance) vary depending on symmetry. If a data set has a 50 / 50 split in classification, then

the maximal change in importance is 0.5 (2 x 0.5 x 0.5). If there is a 10 / 90 split, then the maximal change is 0.18 (0.1 x 0.9). Thus, by considering a tree where the blend of classifications has shifted from 50/50 to 10/90, the maximal possible Gini importance of a variable has been reduced by 0.32. This can be addressed by normalizing the importance estimate by tree before summing them for the composite importance estimate for the random forest. This in effect adjusted Gini importance to estimate what the average proportionate Gini drop was across the trees in the novel random forest. While applying the class-skewing proportion correction adjusts feature importance estimates to values comparable to traditional random forests, this correction is not applied to the ultimate implementation of random forests.

The implications of both lock-out and class-skewing corrections and the justification for their application / non-application were reviewed in the simulation section of this thesis (i.e., sections VI. D. 2. e. v. and vi.).

3. Support Vector Machine (SVM)

The Support Vector Machine (SVM) algorithm is an alternative classification technique that also adheres to a supervised machine learning paradigm. Unlike the random forest approach, it cannot directly partition data into multiple different potential classes (e.g., PACNS, infectious vasculitis, etc.) and focuses solely on a binary divide at any one step (e.g. final diagnosis of primary angiitis of the central nervous system versus a diagnosis other than primary angiitis of the CNS) although techniques to combine classifiers that separate one class (e.g. diagnosis) from another or one class from all others repeatedly to create multiple classes exist.[xxxviii] In principle the algorithm attempts to place binary dividers to bifurcate the data as accurately as possible in many cases after the application of a kernel (i.e., data transformation).[xxxix] As simple, linear solutions may not always exist (i.e., a linear kernel is not always ideal for mapping / transforming the data), the technique also relies on conceptualizing the data in a multidimensional space or transformed space (i.e., application of a non-linear kernel) and then, within this transformed space, attempts to find a "hyperplane" or divider that leaves members of one class / category on one side opposite the members of the alternative class(es).[xl] To optimize the division between groups the algorithm focuses on those feature groupings / data points (e.g. all the clinical variables associated with a patient) that lie closest to the margin of separation (i.e. the "hyperplane") to generate a potential boundary.

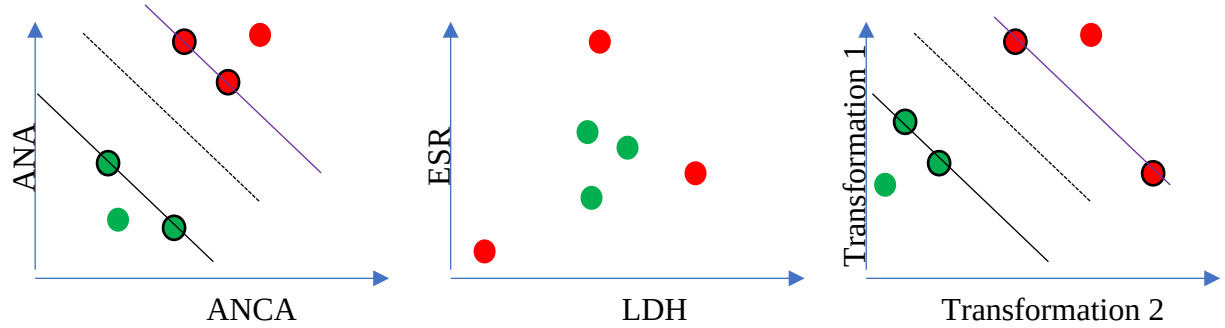


Figure 6. 1a (Left) Shows a simple linear hyperplane separation where the combinations of the ANA and ANCA titers represented by the green balls have a support vector (black solid line connecting only those balls with a black outline) that is parallel to the support vector of the red balls (purple line) allowing a trivial linear hyperplane to be drawn (black dashed line). 1b (middle) does not have a trivial linear divide that is apparent and thus the axes are transformed / features (i.e. ESR and LDH) are combined by a function discovered by SVM to create an obvious solution such as is shown in 1c.

In Figure 6 we showed ideal (1a) and potentially overfitted (1c) scenarios in the sense that the raw data or the transformations (i.e. data after processing with a non-linear kernel function) provided marginal points that reside nearly perfectly along obvious parallel support vectors (i.e. the black and purple solid bounding lines). In practice, some of the points may be separated from these ideal lines/planes by some error term (ε') resulting an irregular "distance" from the derived hyper-plane (ε_i'). This then requires some plan (i.e. cost function) to optimally balance a desire for less convoluted transformations to generate the "ideal" but over fitted picture (i.e. a desire for "flatness") while minimizing the error with which the hyperplane follows the support vectors and lead to potential misclassification.

Formally, with regards to linear SVMs, a hyperplane could in general be defined as some weighted composition of provided data points x_i (where $i = 1 \ldots n$ and n is the number of cases / objects to be classified) contained within the vector x.[xli]

$$w^T x + b = 0 \quad (1)$$

Where w^T is the transpose of a weighting vector and b is a constant term. To find such a plane that maximizes the margin one could find:

$$\max_{\gamma,w,b} \gamma \quad (2)$$

$$t_i(w^T x_i + b) \geq \gamma \forall i \quad (3)$$

$$w^T w = 1 \quad (4)$$

Where t_i is the class of x_i and γ is a dummy variable that satisfies the above equations. To solve this problem, we can reframe the optimization in the form of its Lagrangian.

$$L = -\gamma - \sum_i \alpha_i (t_i(w^T x_i + b) - \gamma) - \beta(w^T w - 1) \quad (5)$$

Where L is the Legrangian, and α_i (note $\alpha_i \geq 0$) and β are the associated multipliers. The following solutions to ensure optimization can then subsequently be formulated:

$$\sum_{i=1}^{n} \alpha_i = 1 \quad (6)$$

$$\sum_{i=1}^{n} t_i \alpha_i = 0 \quad (7)$$

$$w = \sum_{i=1}^{n} t_i \alpha_i x_i = 0 \quad (8)$$

From this system of equations, a solution can be drawn to find the optimal hyperplane.

B. General Concepts

1. Bias-Variance Decomposition

When generating models / estimators, error reduction is critical along with the interpretability of the result. Pursuing increased statistical efficiency (i.e., reduced variance) and bias reduction are key components of error reduction. An important concept that was revisited throughout the thesis is the bias-variance relationship that is often explored via bias-variance decomposition. Bias and Variance (Var) are defined as:[xlii]

$$Bias(\hat{\theta}) = E[\hat{\theta}] - \theta \tag{9}$$

$$Var(\hat{\theta}) = E\left[\left(E[\hat{\theta}] - \hat{\theta}\right)^2\right] = E[\hat{\theta}^2] - E^2[\hat{\theta}] \tag{10}$$

Where $Bias(\hat{\theta})$ represents the bias of the estimator $\hat{\theta}$ and $E[\hat{\theta}]$ represents the expectation value (i.e., expected value) of $\hat{\theta}$ (i.e., the prediction of the class / classification of the case of interest) and θ is the true class. Error is then often derived from the concept of mean squared error (MSE) defined as:

$$MSE(\hat{\theta}) = E\left[\left(\hat{\theta} - \theta\right)^2\right] \tag{11}$$

$$\begin{aligned}
MSE(\hat{\theta}) &= E[\hat{\theta}^2 + \theta^2 - 2\hat{\theta}\theta] \\
MSE(\hat{\theta}) &= E[\hat{\theta}^2] + E[\theta^2] + E[2\hat{\theta}\theta] \\
MSE(\hat{\theta}) &= E[\hat{\theta}^2] + \theta^2 - 2\theta E[\hat{\theta}] \\
MSE(\hat{\theta}) &= E[\hat{\theta}^2] - E^2[\hat{\theta}] + E^2[\hat{\theta}] + \theta^2 - 2\theta E[\hat{\theta}] \\
MSE(\hat{\theta}) &= Var(\hat{\theta}) + \left(E[\hat{\theta}] - \theta\right)^2 \\
MSE(\hat{\theta}) &= Var(\hat{\theta}) + Bias(\hat{\theta})^2
\end{aligned} \tag{12}$$

A more general form of MSE recognizes an irreducible error or noise:

$$MSE(\hat{\theta}) = Var(\hat{\theta}) + Bias(\hat{\theta})^2 + \text{Noise}(\hat{\theta}) \tag{13}$$

With each prediction technique or model, a difference balance of variance and bias is struck. In so far as machine learning approaches such as random forests are concerned, the increased complexity of a model gives rise to greater variability with some asymptotic level of bias.

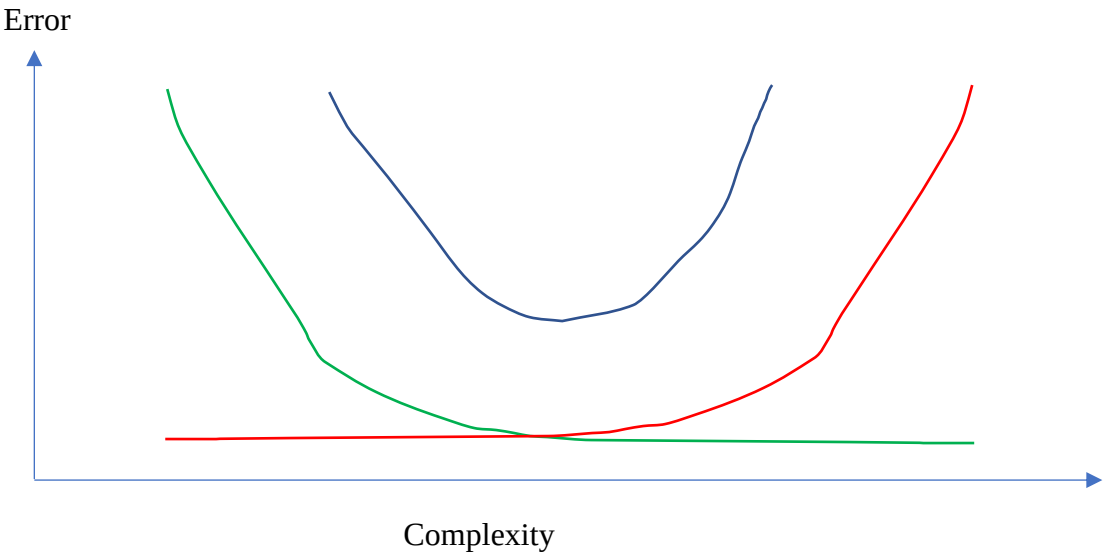


Figure 7. Error curves showing bias-variance tradeoffs with overall error in blue, bias in green, and variance in red. The x-axis represents model complexity, and the y-axis represents the amount of error.

2. Variable Importance

a. Gini Importance

The relative importance of variables / features is a prominent theme that was explored in this thesis. First, for random forest model variants, variables can be characterized by their Gini importance. Gini importance is computed first by computing a sample's impurity (where: C represents that total number of classes that can be selected (e.g., two: either PACNS or not PACNS), f_i represents the frequency of the ith class (e.g., 0.4 are PACNS and 0.6 are not), ι represents the total impurity):[xliii]

$$\iota = \sum_{i=1}^{C} f_i(1 - f_i) \tag{14}$$

In a two-class scenario, as we will discuss extensively, this simplifies as follows:

$$\iota = \sum_{i=1}^{2} f_i(1 - f_i) = f_1(1 - f_1) + f_2(1 - f_2) = f_1 f_2 + f_2 f_1 = 2 f_1 f_2 \tag{15}$$

The importance of a node can then be computed by the decrease in impurity (increase in purity) created by the separation rule introduced. Assuming a binary separation this would take the form of:

$$Im_j = wt_j \iota_j - \left(wt_{Lj}\iota_{Lj} + wt_{Rj}\iota_{Rj}\right) \tag{16}$$

Where Im_j represents nodal importance of node j, wt_j represents the weighted number of entities (e.g., patients) sent to / contained at node j, ι_j represents the impurity contained within an undivided sample at node j, wt_{Lj} represents the weighted number of entities sent to the left branch per the criteria at node j, ι_{Lj} represents the impurity contained within the left branch sample at

node j, wt_{Rj} represents the weighted number of entities sent to the right branch per the criteria at node j, ι_{Rj} represents the impurity contained within the right branch sample at node j.

For any given tree, a feature importance can be discerned by adding all the nodal importance values up that use the feature of interest to divide the data (i.e., add up all j nodes such that the criteria at the nodes to be included in the sum depend on feature i) and normalizing:

$$F_i = \frac{\sum_{j:node\ splits\ using\ feature\ i} Im_j}{\sum_{k\epsilon\ all\ nodes} I_k} \quad (17)$$

Where F_i represents importance of feature importance across the random forest.

Further explicit normalization may be required depending on the nature of the nodal criteria (e.g., nodes may separate based on a linear combination of two features and thus a nodal importance may be distributed to more than one feature):

$$N_i = \frac{F_i}{\sum_{j\epsilon\ all\ features} F_j} \quad (18)$$

Where N_i represents the normalized importance of feature i, Note: f_i converges to N_i where each node is only featured once in the importance sums of various features. To compute the overall importance of a feature throughout the random forest one takes the average normalized importance:

$$R_i = \frac{\sum_{j\epsilon\ all\ trees} N_{ij}}{n_T} \quad (19)$$

Where R_i represents the importance of feature i computed across the data set and n_T represents the total number of trees contained within the random forest.

Gini importance has a reported drawback of over emphasizing those features that are continuous or categorical features with high cardinality (i.e., many levels where features can divide up a very small node / are highly suggestive of very small groups) and this may unfortunately compromise generalizability.[xliv] An extreme example of this would be that one is able to use a medical record number or name to divide off cases very successfully one at a time but provides no real insight into data trends. There is a potential way to address this issue by recognizing it as an artefact of the overfitting.

When data is divided, it must occur for each node at a single value (i.e., every node has a cut point / value). Consider that if one is dividing at multiple values of a certain feature or invoking a feature at several nodes in a decision tree across a random forest reflects truly non-linear behavior (e.g., the function of the variable with respect to classification is non-linear or the relationship of the variable with respect to classification is legitimately altered by blues of certain other variables) then this would be appropriately selected / included in the model and the accrual of importance for those divisions would also be justified. If the tree is taking advantage of

random variability in features then this would represent overfitting (i.e., if a feature value is truly an element of real numbers [i.e., $\epsilon\Re$] existing within a continuum of whole and fractional values then the probability two values will be precisely the same and hence inseparable is vanishingly small and that could be used as a form a sort of unique identification resulting in overfitting). Because random forests are non-linear techniques, appropriate threshold setting can reflect a dramatic or subtle shift and shifts could be increased or decreased without interfering with the appropriateness of the division (see Figure 8).

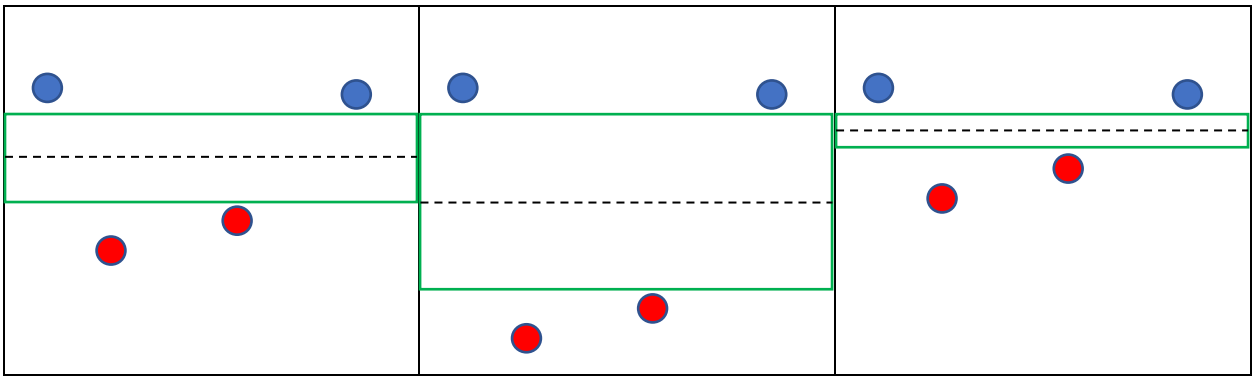


Figure 8. Where blue dots represent one classification and red dots another, the dashed black line represents an ideal threshold to separate the two set of objects (left diagram) and the green box represents the gap in feature values that could be arbitrarily expanded (center diagram) or contracted (right diagram) without a change in the assessed improvement in purity following the split.

Introducing some random noise is not of an amplitude that interferes with identifying the relative ordering of the data (which was subsequently referred to as pseudo-randomness in this thesis), and by extension the ideal threshold, does not add or detract from an ability for a variable to be used to create separations about the ideal threshold. It does however afford features with entries containing the same values (i.e., categorical features with any degree of cardinality or continuous variables with ties) the same potential that continuous variables have for overfitting regardless of the amplitude of that added variability (see Figure 9) as explored in Figure 8.

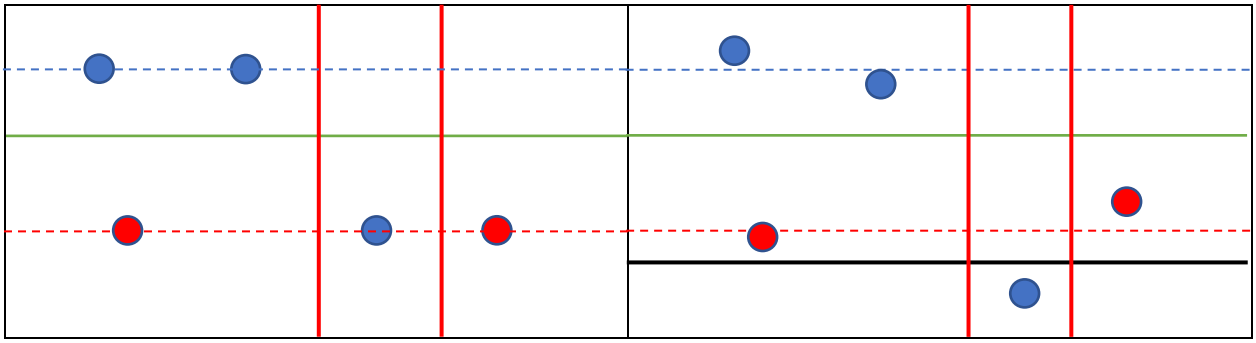


Figure 9. Where blue dots represent one classification and red dots another, consider the points arranged such that the feature whose value is represented by the x-axis has a value that varies randomly with respect to the classification but after separating by the bilevel feature (whose value is represented by the y-axis) at the green line indicated threshold it provides some avenue for impurity reduction by placing thresholds at the red lines (right diagram); by adding variability to the feature on the y-axis another division can be made by the black line indicated threshold and thus one could assign exactly the same amount of additional impurity reduction to either feature.

Appreciating the bias of Gini importance toward continuous variables versus variables with high cardinality versus lower cardinality as a known weakness of Gini importance-based feature

importance assessment, pseudo-randomness was used in this thesis to address this bias of cardinality in random forest models.[xlv]

To reiterate on the topic of proportioning*class-skewing, with regards to the behaviour of importance measure assessment of the novel random forest variant, it is of value to note that while each tree in a standard random forest will have roughly an equal number of n_T cases presented to it (around 63 – 67% of the global number of cases in the training dataset n) the forests with a novel random forest design will only have a fraction of that number separated during the training process (i.e., only that fraction of cases with all of the l features present where l is the number of features selected to generate the ith tree of the random forest ensemble).[xlvi, xlvii] Where Gini importance is formulated by summing the importance measures obtained for various features across trees, one could consider weighting the sum by the number of n_{Ci} cases presented to an individual tree. The impacts of this were reviewed in simulations that were presented later in the thesis. Such a correction was not made in the final simulations / construction of the novel random forest technique with the view that the novel approach specifically sought to create a special case of the random forest technique where only a subset of cases with present information of a series of l features would be available (and the trees would be confined to considering a subset of those l features at any one node). The impact of this assumption was noted as an aside within the simulation series for the novel random forest.

b. Permutation Importance

Aside from Gini based importance assessments, variable importance can also be explored using a permutation approach.[xlviii] If there is a significant volume of variables to be assessed, hierarchical clustering can be used to visualize the correlation between variables and a cut point selected by inspection. Candidate variables can be selected from each cluster and a model generated using these distilled variables. Accuracy of the model can then be determined using the test set and then the change in performance (e.g., ROC, accuracy) after a variable is shuffled can be recorded and compared. Variables of greatest significance are recognized as those that, when shuffled, result in the greatest change / decrease in model performance.

3. Receiver Operator Curves (ROC) / Area Under the Curve (AUC) Analysis

ROC and AUC analysis are frequently used to compare the performance of statistical approaches. For the ensemble technique, true positive (sensitivity) and false positive rates (1 – specificity) are augmented by altering the threshold proportion of votes required to indicate that a case class of interest is predicted. For example, if the feature data pertaining to a case of interest was fed into the trained algorithm and resulted in a percentage of votes (e.g., 56%) in favor of being a classified correctly as PACNS (i.e., the class of interest) then requiring an even larger percentage of votes (e.g. 70%) could precipitate more false negatives but could also reduce non-PACNS cases from being misclassified (reduced false positive rate). A similar rationale can be applied to the SVM approach for comparison.

Note that ROC and AUC are convenient metrics of performance when one's purpose is not to find the ideal model for detecting a class like PACNS versus non-PACNS per se but to appreciate if a method could generate models with the best possible performance. Although accuracy is an oft preferred metric in demonstrating the relative utility of models, AUC has the advantage of avoiding "guttering." Consider that accuracy sets a threshold of 50% and if this is

an ideal set point then accuracy will be a faithful representation of best possible model performance. If it is not, it might be that in the event ROC approximates and ideal step function (i.e., AUC is 1), increasing the threshold reduces the false positive rate without substantially decreasing the true positive rate or decreasing the threshold increases the true positive rate without substantially increasing the false positive rate (See Figure 10).

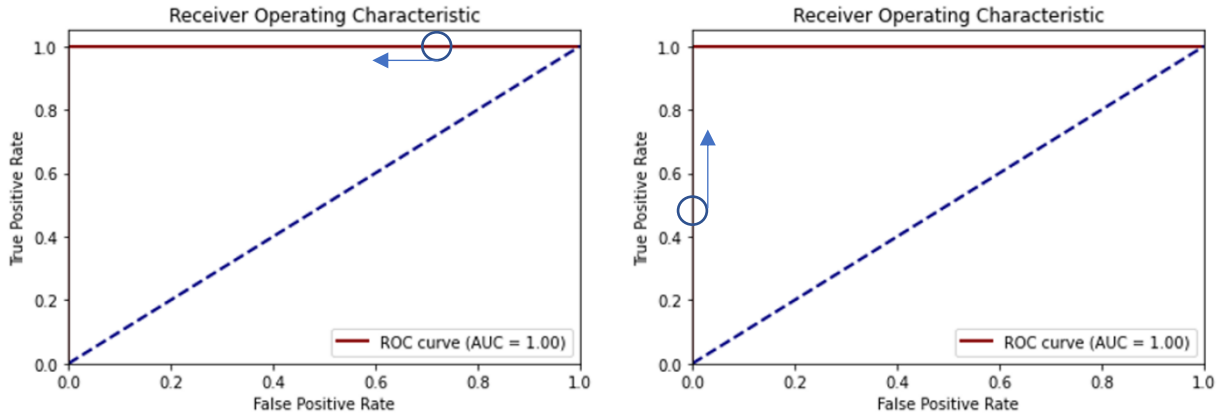


Figure 10. Note that with a perfect ROC curve, when selecting a threshold for determining if a case is positive (i.e., a case of interest identified), if there are excess false positives than optimal (note that as false positive rate = 1 – true negative rate, this is equivalent to saying to few true negatives are reported), increasing the threshold reduces the false positive rate (note: "raising the bar" / increasing a threshold to declare a test positive maintains or reduces the likelihood of finding any case to be positive and thus any case falsely positive per that test) without decreasing the true positive rate which is represented by moving the blue indicator leftward on the ROC curve (left diagram); if using some threshold results in fewer than optimal true positives (note that as true positive rate = 1 – false negative rate, this is equivalent to saying excess false negatives are reported), decreasing the threshold increases the true positive rate without increasing the false positive rate which is represented by moving the blue indicator upward on the ROC curve (right diagram).

C. Primer on Managing Missing Data

Since their inception, modern classification methods including variations on random forests and model boosting (e.g., boosted trees) have been maturing to provide increasingly more robust and accurate judgments than those of simpler classification tools (e.g., singular decision trees). To address missing data, imputation and data pre-processing choices have been popular approaches to fill in data gaps using inferences from available data. Alternative approaches have focused on adapting the classification methods to accept data with missing entries.[xlix] Often these techniques that can adapt during the training phase of the algorithm and are built into the construction of composite predictors.

1. Standard Approach to Missing Data: Adapting to Missingness by Pre-processing

Missing data is a key issue for both the SVM and the random forest techniques to contend with.[l] As in more classical approaches such as logistic regression, varying degrees of bias are introduced into the model depending on how the missing data is managed and whether the data is missing randomly (e.g. a test for ANA was not recorded) or non-randomly (e.g. a clear overriding test was positive such as a CSF culture for streptococcus, which justifiably precluded further testing in the pursuit of a PACNS diagnosis) as previously discussed. A simple approach would be to use only complete cases but in diseases like PACNS that are diagnosed based on the exclusion of other entities, this could be catastrophic. Decisive testing performed early in the diagnostic process would be dismissed in model generation as all those cases for which the variable was not suggestive of PACNS (i.e., the testing confirmed another disease process)

would be removed. As in the example of CNS infection with cryptococcus, no further tests exploring the diagnosis of PACNS would be required once a high cryptococcal antigen titer in the blood / CSF was found. The presence of that test / titer effectively confirmed a CNS cryptococcal infection, not PACNS, and those cases would therefore have incomplete data as they would not require further testing. Thus, only cases where those key early variables could be consistent with PACNS would be included for analysis and potentially decisive variables could seem irrelevant.

Additionally, diagnostic testing for rare clinical entities like PACNS often relies more on the investigating clinician's experience and less on a protocolized approach. Testing is therefore expected to vary by practitioner, potentially systematically, creating data sparsity that would result in dismissal of a substantial quantity of cases. Fortunately, SVM would be expected to be more robust to this issue, than logistic regression for example, as it doesn't necessarily use all the data within a given dataset.[li] It uses data from only those cases on the margin of classification (i.e., those who are more difficult to classify and thus may be expected to have more complete testing) to determine how best to classify individuals.

Complete case analysis is not the only method of dealing with missing data. Imputation is a common approach for filling missing data gaps. Missing entries can either be filled with a single value (i.e., a Strawman approach as it related to random forests), typically the mean or median value seen for the feature of interest across the sample (e.g. a missing ANA being replaced with a titer level of 1:125.6 or 0.0080) or drawn from a distribution or algorithm.[lii] That distribution/algorithm can either be created/computed from the entire sample (e.g. Multivariate imputation by chained equations [MICE]) or only those with similar data determined by the complete data available (k-Nearest Neighbors [kNN]).

For reference MICE is executed as follows (MICE Pseudocode):[liii]

1. Initial Imputation
    1. For every missing value, replace missing feature values with a simple imputation such as the mean value for that feature
2. Update imputation
    1. Rotate through all features one feature at a time to update the imputations
        1. Return missing values associated with a feature of interests to missing
        2. Create a model to predict the value of the feature of interest using the observed feature values of interest and the data completed with a simple imputation (i.e., create a predictive model using complete cases analysis after the missing values have been restored to the feature of interest).
        3. Replace missing values with the predictions made using the predictive model
    2. Continue the process of rotating through and updating predictions for feature values one feature at a time.

For reference kNN imputation of missing values can vary in its execution as but is commonly applied as follows:[liv]

1. For every missing value pertaining to a particular feature (i.e., feature of interest) for a particular case (i.e., case of interest):
    1. Compute the distance between the case of interest with the missing value and all other available cases using differences in feature values between all other features other than the feature of interest.
        1. For all other non-missing features of the case of interest compute sum over the differences between its feature values and the feature values of all other cases. The cumulative distance between a case of interest a and a donor candidate case $b\ is$:

$$dist_{a-b} = \frac{n_{available}}{n_{complete}} \sum_{i=1}^{n_{complete}} d_{sep}(x_{ai}, x_{bi}) \quad (20)$$

where $n_{complete}$ represents the number of features with non-missing values in a, $n_{available}$ represents the number of instances where both the features of a and b are non-missing, d_{sep} is the distance function (e.g., for Euclidean distance the distance function would be the square of the difference in feature values), and x_{ai} and x_{bi} are the ith feature values for a and b respectively. Note that b must have a non-missing feature value of interest.

        2. Find a number k of closest / nearest neighboring cases (i.e., those where $dist_{a-b}$ is smallest).
        3. Replace the missing value of the feature of interest for a case of interest with some prediction based on the data from neighboring cases or some combination of the values for the feature of interest in the k nearest neighboring cases (e.g., the mean value).
2. Repeat the procedure until all missing values are resolved

Imputation may also use a random forest itself to identify missing data values prior to subsequent processing. A technique known as proximity random forest runs cases through a random forest and identifies situations where cases with missing data fall into the same terminal nodes as other cases with the data of interest intact. The missing feature then becomes a weighted sum of the feature values of all the other complete cases that fell into the same nodes with computation iterated until subsequent computations result in a sufficiently stable value estimation.[lv] A missForest technique applies random forest regression to fill missing data using all available alternative data and then repeats computations until changes between regression estimates are adequately small. Imputation may also be dynamic where imputation may occur just before a split is to occur (i.e., at a node) allowing more similar data to be drawn on.[lvi]

Another approach still is to adapt an ensemble technique and avoid preprocessing (i.e., the novel technique that was proposed in this thesis). Each component model (e.g., tree) could be constructed within a randomly selected feature space in such a way that it sidesteps missingness by relying on partly complete data. In other words, each model or "weak learner" relies on different variables like ANA and ANCA versus another with ESR and LDH for example and use

only the bootstrapped sample from the complete cases with all the selected variables. This is discussed in more detail later as a novel approach to random forest analysis.

2. Adapting to Missingness by Adjusting Algorithm Components

The foundation of classification algorithms is the application of mechanisms, such as decision trees or rules (e.g., equations or thresholds), to separate cases out into broader classes by one or several feature values. In decision trees, when specific candidate criteria are being considered at a node for optimal partitioning, cases with missing data could be: 1) dropped if they do not possess the criteria of interest (Complete cases), 2) randomly divided between branches, 3) systematically assigned to one branch (e.g., if ANA presence is demonstrated then the data is divided into one branch and if it is not or the value is unknown then it is assigned to the other branch), 4) preprocessed by a node prior to the proposed decision node that divides data where a value is missing versus not (i.e., the so called BEST versus trifurcation approach where a case reaching a node can ultimately be separated into three separate branches where one branch represents cases where data is missing, a second represents cases with the feature value on one side of the proposed threshold, and a third with the feature value on the other side of the proposed threshold), 5) probabilistically assigned to one branch versus the other, 6) proportions of individual cases are sent down each decision path (e.g., Quinlan's C4.5 algorithm where cases are "divided" in a proportion equal to how the data with known values were split at the node of interest).[lvii], [lviii]

3. Adapting to Missingness by Adjusting the Ensemble Approach

The novel approach reviewed here finds a subset of random forest ensemble possibilities that "side-step" missingness. As noted, when generating a random forest, the complete(d) data set is used to create several short decision trees. The variables to be used for any one of these trees are first selected at random and then cases are drawn with replacement to train the model. This is done a pre-selected number of times. Those cases that are randomly never used but were formally part of the training set are considered "out of bag." Case selection and omission could be more deterministic, however. Once a set of variables is selected, only the cases with complete data with respect to those variables could be included for the bootstrapping process. In the case of the Novel random forest variant, such selectivity is applied, and sub-setting could lead to critically low numbers of cases for separation. Of note, if the variables selected lead to the generation of a set of 3 or fewer cases or if there are not representatives from both target classes (i.e., PACNS and non-PACNS), then no learner is generated, and another set of specified variables is drawn. Note that a set of 3 cases in this situation in considered a critical number for a population as 2 cases would be too few to characterize / probe for a non-linear variable / classification environment.

4. Missingness at it pertains to SVM

SVM possess an inherent robustness to missing data as it is a margin classifier.[lix] Nevertheless techniques such as complete case analysis and imputation can still be applied. In this thesis, a single value imputation method was applied for comparability across algorithms and to simplify the discussion regarding the susceptibility of various algorithms to missing data patterns.

5. Exploring Patterns of Missingness

Although missingness can be accounted for in many ways, the pattern of missingness can impact the applicability and success of adaptive techniques. Consider a situation whereby in a sample of F features, the E_{ij} values are random. Only one feature contains some insight into the ultimate classification although that insight is perfect and linear.

$$\left(\begin{array}{ccc|c|c} E_{11} & \cdots & E_{1(F-1)} & 1 & 1 \\ & & & 2 & 2 \\ & & & 3 & 3 \\ & & & 4 & 4 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ & & & n-3 & n-3 \\ & & & n-2 & n-2 \\ & & & n-1 & n-1 \\ E_{n1} & \ldots & E_{n(F-1)} & n & n \end{array}\right)$$

More concretely, consider a grouping of nine samples where every value 5 and above in the last of F features predicts a C_H classification (where C_H represents the class where there is a higher underlying value of the perfectly predictive feature) and every value below 5 predicts a C_L classification (where C_L represents the class where there is a lower underlying value of the perfectly predictive feature).

$$\left(\begin{array}{ccc|c|c} E_{11} & \cdots & E_{1(F-1)} & 1 & 1 \\ E_{21} & & E_{2(F-1)} & 2 & 2 \\ E_{31} & & E_{3(F-1)} & 3 & 3 \\ E_{41} & & E_{4(F-1)} & 4 & 4 \\ E_{51} & \ddots & E_{5(F-1)} & 5 & 5 \\ E_{61} & & E_{6(F-1)} & 6 & 6 \\ E_{71} & & E_{7(F-1)} & 7 & 7 \\ E_{81} & & E_{8(F-1)} & 8 & 8 \\ E_{91} & \ldots & E_{9(F-1)} & 9 & 9 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c|c} E_{11} & \cdots & E_{1(F-1)} & 1 & C_L \\ E_{21} & & E_{2(F-1)} & 2 & C_L \\ E_{31} & & E_{3(F-1)} & 3 & C_L \\ E_{41} & & E_{4(F-1)} & 4 & C_L \\ E_{51} & \ddots & E_{5(F-1)} & 5 & C_H \\ E_{61} & & E_{6(F-1)} & 6 & C_H \\ E_{71} & & E_{7(F-1)} & 7 & C_H \\ E_{81} & & E_{8(F-1)} & 8 & C_H \\ E_{91} & \ldots & E_{9(F-1)} & 9 & C_H \end{array}\right)$$

Now consider the situation where a missing value of the last column itself carries an 80% likelihood that the true value was 5 or above with increasing probability for increasing underlying value.

$$\left(\begin{array}{cccc|c} E_{11} & \cdots & E_{1(F-1)} & 1 & C_L \\ E_{21} & & E_{2(F-1)} & 2 & C_L \\ E_{31} & & E_{3(F-1)} & 3 & C_L \\ E_{41} & & E_{4(F-1)} & 4 & C_L \\ E_{51} & \ddots & E_{5(F-1)} & 5 & C_H \\ E_{61} & & E_{6(F-1)} & 6 & C_H \\ E_{71} & & E_{7(F-1)} & 7 & C_H \\ E_{81} & & E_{8(F-1)} & 8 & C_H \\ E_{91} & \cdots & E_{9(F-1)} & 9 & C_H \end{array}\right) \rightarrow \left(\begin{array}{cccc|c} E_{11} & \cdots & E_{1(F-1)} & 1 & C_L \\ E_{21} & & E_{2(F-1)} & 2 & C_L \\ E_{31} & & E_{3(F-1)} & - & C_L \\ E_{41} & & E_{4(F-1)} & 4 & C_L \\ E_{51} & \ddots & E_{5(F-1)} & 5 & C_H \\ E_{61} & & E_{6(F-1)} & - & C_H \\ E_{71} & & E_{7(F-1)} & - & C_H \\ E_{81} & & E_{8(F-1)} & - & C_H \\ E_{91} & \cdots & E_{9(F-1)} & - & C_H \end{array}\right)$$

Now consider that the missing values are filled with the mean feature value from the observed set. In this case the split that most reduces Gini impurity (and in this case entropy) would be to divide the group such that those cases with the last factor having values up to 2 are in one group and 3 or more would be in another group.

$$\left(\begin{array}{cccc|c} E_{11} & \cdots & E_{1(F-1)} & 1 & C_L \\ E_{21} & & E_{2(F-1)} & 2 & C_L \\ E_{31} & & E_{3(F-1)} & - & C_L \\ E_{41} & & E_{4(F-1)} & 4 & C_L \\ E_{51} & \ddots & E_{5(F-1)} & 5 & C_H \\ E_{61} & & E_{6(F-1)} & - & C_H \\ E_{71} & & E_{7(F-1)} & - & C_H \\ E_{81} & & E_{8(F-1)} & - & C_H \\ E_{91} & \cdots & E_{9(F-1)} & - & C_H \end{array}\right) \xrightarrow{m} \left(\begin{array}{cccc|c} E_{11} & \cdots & E_{1(F-1)} & 1 & C_L \\ E_{21} & & E_{2(F-1)} & 2 & C_L \\ E_{31} & & E_{3(F-1)} & 3 & C_L \\ E_{41} & & E_{4(F-1)} & 4 & C_L \\ E_{51} & \ddots & E_{5(F-1)} & 5 & C_H \\ E_{61} & & E_{6(F-1)} & 3 & C_H \\ E_{71} & & E_{7(F-1)} & 3 & C_H \\ E_{81} & & E_{8(F-1)} & 3 & C_H \\ E_{91} & \cdots & E_{9(F-1)} & 3 & C_H \end{array}\right) \xrightarrow{2|3} \begin{array}{l} \left(\begin{array}{cccc|c} E_{11} & \cdots & E_{1(F-1)} & 1 & C_L \\ E_{21} & \cdots & E_{2(F-1)} & 2 & C_L \end{array}\right) \\ \left(\begin{array}{cccc|c} E_{31} & \cdots & E_{3(F-1)} & 3 & C_L \\ E_{41} & & E_{4(F-1)} & 4 & C_L \\ E_{51} & & E_{5(F-1)} & 5 & C_H \\ E_{61} & \ddots & E_{6(F-1)} & 3 & C_H \\ E_{71} & & E_{7(F-1)} & 3 & C_H \\ E_{81} & & E_{8(F-1)} & 3 & C_H \\ E_{91} & \cdots & E_{9(F-1)} & 3 & C_H \end{array}\right) \end{array}$$

$$\iota_{Before} = \sum_{i=1}^{C} f_i(1 - f_i) = 2 \times \left(\frac{4}{9}\right) \times \left(\frac{5}{9}\right) = 0.49 \quad (21)$$

$$\iota_{After} = \sum_{sb=1}^{n_b} w_{sb} \sum_{i=1}^{C} f_{si}(1 - f_{si}) = 2 \times \left(\frac{2}{9}\right) \times \left(\frac{2}{2}\right) \times \left(\frac{0}{2}\right) + 2 \times \left(\frac{7}{9}\right) \times \left(\frac{2}{7}\right) \times \left(\frac{5}{7}\right) = 0.32 \quad (22)$$

Where w_s is the relative number of cases assigned to branch sb and n_b is the number of branches. Alternatively, one could utilize the BEST algorithm or allow for a trifurcation of the data which could lead to an even greater drop in the impurity. One should note that while the BEST algorithm and trifurcation algorithms are closely related and do present some additional potential to identify "ideal" splits and therefor maximize order / purity, there are important conceptual differences. BEST specifically uses a gating variable (i.e., the presence or absence of data) to detect when to engage a split but does automatically pursue a value-based split thereafter as seen in Figure 11.

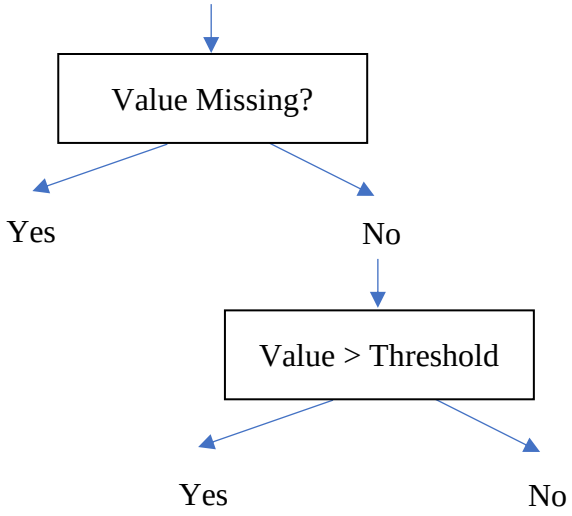


Figure 11. BEST splitting node

An alternative way to view this approach is that it essentially introduces a series of dummy variables that detect missing data but links their call necessarily and immediately to a value-based decision for the underlying data (i.e., a node is introduced by the purity change associated with recognizing a missing value and a value-based split also then takes place).

A trifurcation (node pictured in Figure 12) is just as it sounds, a three-way split in one step. Unfortunately, the first time a variable is called it can be insinuated into a situation where another division would be more appropriate or dismissed when it would be a better choice by automatically adding the weighting created by the impurity of data where the variable is not known.

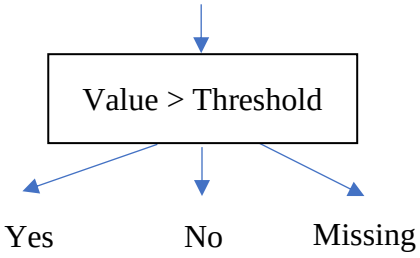


Figure 12. Trifurcation node

It is useful for the sake of discussing our proposed algorithm, to view both BEST and trifurcation as creating a situation where, for every tree generated for the random forest calling on at least one variable with missing data to mediate a division, there are at two overlapping tress created in a fashion similar to surrogate value divisions (i.e., one tree that handles the data when it is present and the other that handles the data where it is not present while the other abstains). Consider the following tree in Figure 13 where unknown values exist for features 1 and 2.

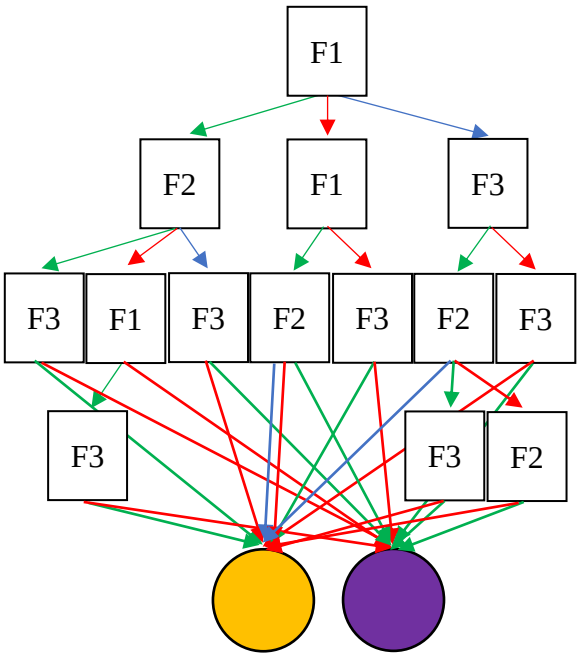


Figure 13.Decision tree with 3 features and decision nodes with thresholds relating to those features indicated by black squares with the feature numbers indicated on the inside (i.e., F1 for feature 1, F2 for feature 2 and F3 for feature 3). Branches that indicate the threshold was exceeded are indicated in green and not exceeded are in red. Where the data was unavailable the arrow is indicated in blue.

If one now considers the scenarios whereby data could be 1: complete or missing in three different ways (i.e., 2: the data for feature 1 is missing but the data for feature 2 is present, 3: the data for feature 2 is missing but the data for feature 1 is present, and 4: the data for both features 1 and 2 is missing) one would see 4 sperate trees emerge:

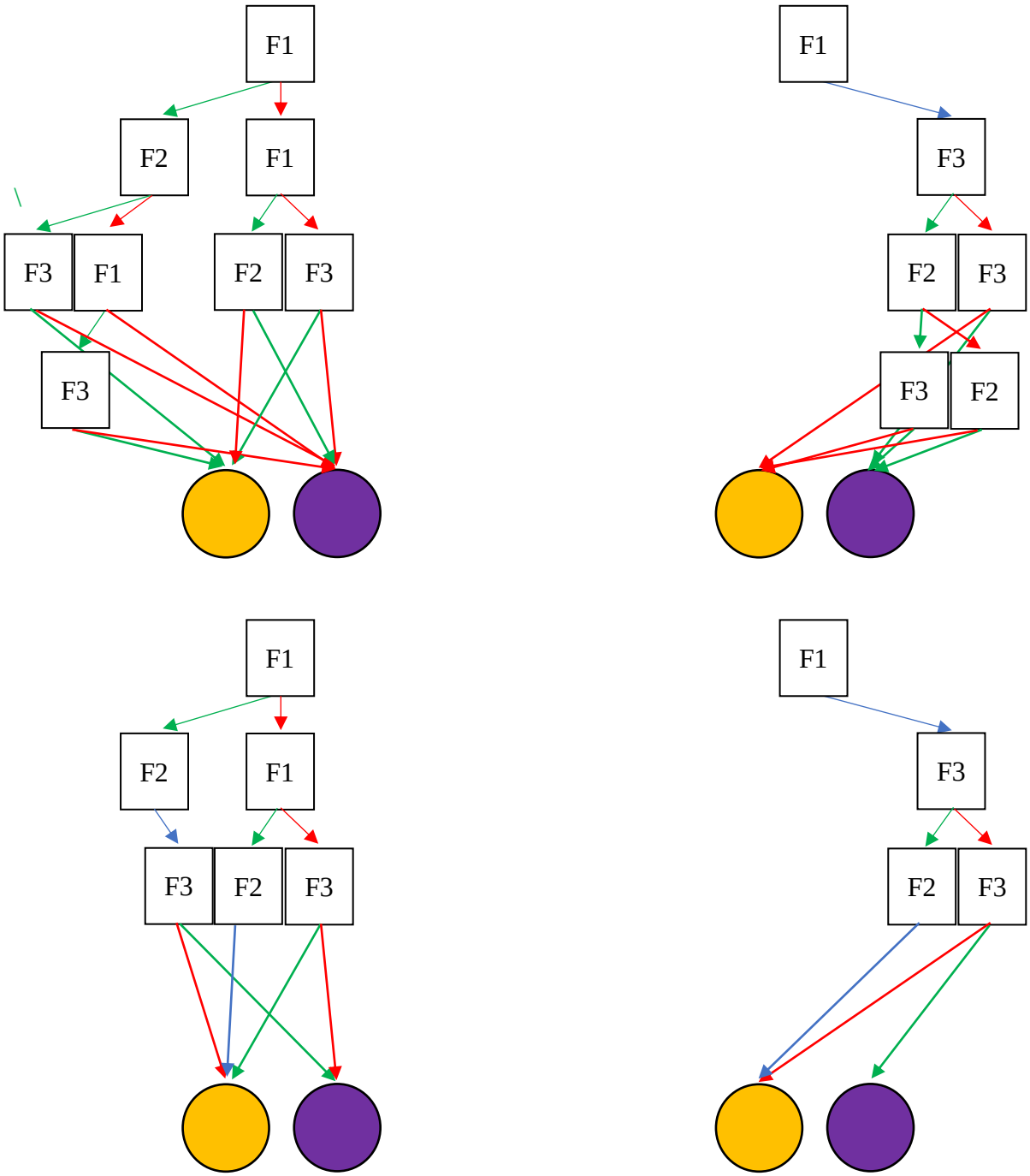


Figure 14. Daughter trees that are embedded as the result of allowing for gating. A) Top Left - F1 and F2 data present, B) Top Right – F1 absent but F2 preset, C) Bottom Left – F1 present but F2 absent, D) Bottom Right – F1 and F2 absent

Note that once data for a feature has been found to be present, future BEST nodes using the feature no longer have missing data to deal with and trifurcations effectively become bifurcations. Similarly, if the feature was found to be missing it would not be called on again for its discriminative potential downstream in the algorithm (i.e., it has no discriminative potential as it has been established that the data does not exist).

Returning to the example, the BEST algorithm would split as follows with impurity computed for gating (i.e., whether the value of importance was present or not):

$$\begin{pmatrix} E_{11} & \dots & E_{1(F-1)} & 1 & C_L \\ E_{21} & & E_{2(F-1)} & 2 & C_L \\ E_{31} & & E_{3(F-1)} & - & C_L \\ E_{41} & & E_{4(F-1)} & 4 & C_L \\ E_{51} & \ddots & E_{5(F-1)} & 5 & C_H \\ E_{61} & & E_{6(F-1)} & - & C_H \\ E_{71} & & E_{7(F-1)} & - & C_H \\ E_{81} & & E_{8(F-1)} & - & C_H \\ E_{91} & \dots & E_{9(F-1)} & - & C_H \end{pmatrix} \xrightarrow{x} \begin{pmatrix} E_{11} & \dots & E_{1(F-1)} & 1 & C_L \\ E_{21} & \ddots & E_{2(F-1)} & 2 & C_L \\ E_{41} & & E_{4(F-1)} & 4 & C_L \\ E_{51} & \dots & E_{5(F-1)} & 5 & C_H \end{pmatrix} \begin{pmatrix} E_{31} & \dots & E_{3(F-1)} & - & C_L \\ E_{61} & & E_{6(F-1)} & - & C_H \\ E_{71} & \ddots & E_{7(F-1)} & - & C_H \\ E_{81} & & E_{8(F-1)} & - & C_H \\ E_{91} & \dots & E_{9(F-1)} & - & C_H \end{pmatrix}$$

$$\xrightarrow{4|5} \begin{pmatrix} E_{11} & \dots & E_{1(F-1)} & 1 & C_L \\ E_{21} & \ddots & E_{2(F-1)} & 2 & C_L \\ E_{41} & \dots & E_{4(F-1)} & 4 & C_L \end{pmatrix} (E_{51} \quad \cdots \quad E_{5(F-1)} \quad 5 | C_H) \begin{pmatrix} E_{31} & \dots & E_{3(F-1)} & - & C_L \\ E_{61} & & E_{6(F-1)} & - & C_H \\ E_{71} & \ddots & E_{7(F-1)} & - & C_H \\ E_{81} & & E_{8(F-1)} & - & C_H \\ E_{91} & \dots & E_{9(F-1)} & - & C_H \end{pmatrix}$$

$$\iota_{After\ Gate} = \sum_{s=0}^{m} w_{sb} \sum_{i=1}^{C} f_{sbi}(1 - f_{sbi}) \qquad (23)$$

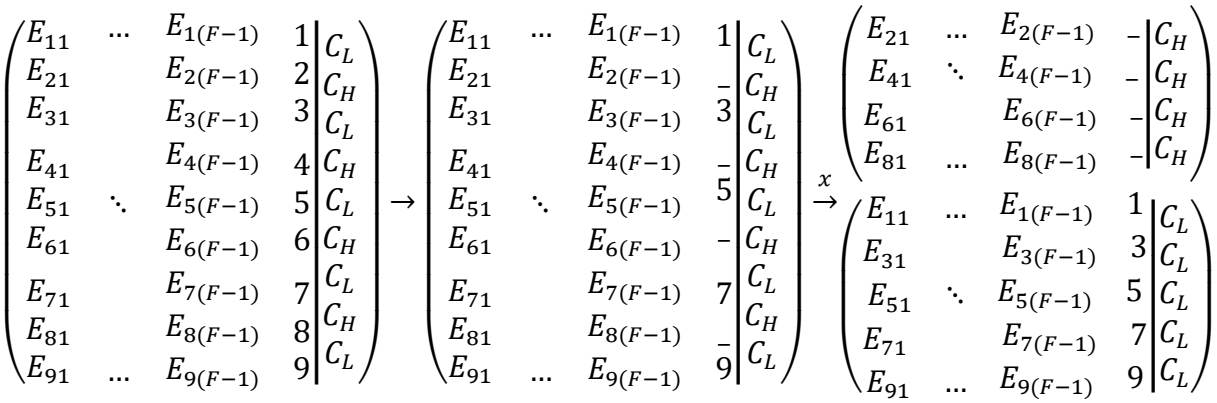
$$= 2 \times \left(\frac{4}{9}\right) \times \left(\frac{3}{4}\right) \times \left(\frac{1}{4}\right) + 2 \times \left(\frac{5}{9}\right) \times \left(\frac{1}{5}\right) \times \left(\frac{4}{5}\right) = 0.34$$

For trifurcation one would compute the impurity change with all the impurity shift combined:

$$\begin{pmatrix} E_{11} & \dots & E_{1(F-1)} & 1 & C_L \\ E_{21} & & E_{2(F-1)} & 2 & C_L \\ E_{31} & & E_{3(F-1)} & - & C_L \\ E_{41} & & E_{4(F-1)} & 4 & C_L \\ E_{51} & \ddots & E_{5(F-1)} & 5 & C_H \\ E_{61} & & E_{6(F-1)} & - & C_H \\ E_{71} & & E_{7(F-1)} & - & C_H \\ E_{81} & & E_{8(F-1)} & - & C_H \\ E_{91} & \dots & E_{9(F-1)} & - & C_H \end{pmatrix} \xrightarrow{4|5} \begin{pmatrix} E_{11} & \dots & E_{1(F-1)} & 1 & C_L \\ E_{11} & \ddots & E_{2(F-1)} & 2 & C_L \\ E_{11} & \dots & E_{4(F-1)} & 4 & C_L \end{pmatrix} (E_{11} \quad \cdots \quad E_{5(F-1)} \quad 5 | C_H) \begin{pmatrix} E_{11} & \dots & E_{3(F-1)} & - & C_L \\ E_{11} & & E_{6(F-1)} & - & C_H \\ E_{11} & \ddots & E_{7(F-1)} & - & C_H \\ E_{11} & & E_{8(F-1)} & - & C_H \\ E_{11} & \dots & E_{9(F-1)} & - & C_H \end{pmatrix}$$

$$\iota_{After} = \sum_{sb=0}^{m} w_{sb} \sum_{i=1}^{C} f_i(1 - f_i) = 0 + 0 + 2 \times \left(\frac{5}{9}\right) \times \left(\frac{1}{5}\right) \times \left(\frac{4}{5}\right) = 0.18 \qquad (24)$$

Here, the missingness of the data represented mostly the value of missing data which in turn directly dictated the classification. When one considers that the missingness of data may present different information than the values of the present data it might be that missingness itself of greatest value and thus forcing its attachment to a value-based split may lead to inefficiencies in characterization. Consider the following:



One can clearly see that the value of the feature does not matter here when the missing data has been accounted for. Perhaps, and more than likely, this is because the data from some other feature presence / value obviated the classification or the first step in a diagnostic algorithm resulted in an alternative diagnosis and therefore downstream testing was not necessary (i.e., the result is clear before all stages of a serial diagnostic algorithm are completed). Forcing a further split of the data in this case is clearly unnecessary although admittedly non harmful as there is no change to impurity. One could imagine however that if one of the missing feature values was randomly present (i.e., the presences / missingness was not dependent on the same circumstances that allowed the missing data to predict the classification outcome) that a threshold as random as the random component relating to the features missingness would be forced into the algorithm. For example, if the 7th case's final feature value were present then the threshold would favor separating out values 7 and up and if the 3rd case's final feature value were present then the threshold would favor separating out cases with values 3 and down. This in turn could lead to misclassification of a substantial portion of test cases depending on the tie breaking rule (i.e., if you have a leaf that is split 50/50 then which class is assigned). Overall, the predictive value of the missingness of the data in this example implies there are other features somewhere in the greater model or that testers were otherwise aware of (i.e., outside the model) and it is important to appreciate the inclusion of this missingness reassigns some or in this case all of the importance from other relevant factors.

Another concern arises that for any given data set it is unclear that the cases with missing data represent the same circumstances / context of data. While one could argue that, if the difference in context was material there would be an early impetus to split by the missingness of the feature, this issue evokes consideration of data context sensitive approaches. Lazy classifiers are an example whereby data is stored, and a new model is generated for a new test case (e.g., a new patient coming into the clinic) by using the multiple older cases but limiting the used training data to only those prior cases that had similar data present. As random forests randomly select subsets of features upon which to base the component decision trees, it is alternatively possible to simply incorporate this practice consciously into the forest itself (i.e., only using the cases that have complete data for the requested features as outlined for the proposed novel approach). One could also add the dummy variables indicating the presence of a factor and ensure they are called on prior to the value-based separation by a factor and not compound in one step as set of two steps of which one may be less ideal. In the former scenario there is no technical violation of the training of a random forest procedure per se. It is possible that in the generation of a random

forest that when data is selected to create a composite tree, one could by chance select only those cases with complete data given the presented features.

Although accuracy of the algorithms is important, obtaining some intuition about the data / factors is also necessary to translate useful insights to the users. In the situation where one is trying to appreciate which features are most important to direct guidelines these insights may be even more important than the models themselves. These presented algorithms (BEST and trifurcation) can create some complexities in interpreting the importance of all variables as alluded to previously. By wrapping in the presence or absence of data into the division of the same feature you ignore the possibility that the missingness may actually be dictated by another variable / feature within or outside of the model (i.e., recognized as true randomness from the models perspective the in the latter case). If the missingness is dictated by the value of one or other variables, then a part of the importance of the dictating variable(s) is to some degree stolen by this separation. When a variable with missing data is called to be part of creating a single decision tree in a random forest, to instead generate 2^n trees where n is the number of variables with missing data (i.e., generate trees relying of permutations of the variables with missing variables while relying on a core of variables with complete data) and use only the complete cases for the given set of variables may be a more interpretable approach. Another alternative to balance accuracy and preserve importance assessments at the expense of processing time is to generate the random forest with a greater number of trees and variable-wise complete data (i.e., any one tree with a certain compliment of variables). One might be tempted to liken this to allowing all available data from a selection to be considered during tree construction, but when any one node is being evaluated, only use data available (i.e., the surrogate data approach). The problem with this technique is that a fair amount of information would be dismissed mid procedure. All steps prior to the dismissal of that data would also be assuming that the data is immaterial when in fact such techniques have been demonstrated to be inaccurate where the assumption of data missing at random is violated.

A. Summary

PACNS is a rare condition and as such one must maximize the insights obtained from a small quantity of data. As there is variability in how the condition might be investigated the features for any one case might vary significantly and large proportions of missing data that is missing for heterogenous reasons would be expected of the dataset for PACNS diagnosis. Managing that missing data can be handled by complete cases analysis, imputation, or by the altering the structure of an algorithm itself. Once a model is devised extracting insights that can be succinctly transmitted to users is benefitted from some appreciation on the relative importance of the features used to reach diagnostic conclusions is useful however some consideration of how the missing data may impact the downstream analysis is important for an accurate assessment. In the context of PACNS diagnosis it is therefore ideal to employ a model, such as the novel random forest, that is suited to a small dataset with a complex pattern of missingness which can further provide feedback to clinicians on what real world data can be collected to make patient care decisions.

IV. Chapter 1: Informal Scoping Review to Candidate Features in PACNS Classification

B. Introduction

A variety of tests have been applied to elucidate the diagnosis of PACNS. To reflect on which tests are likely to be the most helpful, systematic collection of a comprehensive list of these variables was conducted. The purpose of this review was to identify variables that could assist in the classification of PACNS versus non-PACNS cases for later use in generating diagnostic models for PACNS.

C. Methods

1. Operational Definitions

For the purposes of the review, "Suspected PACNS" was defined as those patients where PACNS or a subcategory of PACNS (i.e. Granulomatous angiitis of the CNS or GANS) was articulated by the author as a potential diagnostic consideration or condition of concern in the abstract or body of the article. A diagnostic process in this case was defined specifically as the application of diagnostic testing (e.g. imaging, blood counts, erythrocyte sedimentation rate [ESR], etc.) as opposed to taking direction from clinical variables (e.g. signs, symptoms, etc.). Note was made that as PACNS is generally considered to be a diagnosis of exclusion, these criteria in large part serve to incrementally separate out / remove from consideration mimickers to ultimately posit the likely diagnosis of PACNS. For example, evidence of histoplasma, Aspergillus, and/or Blastomycosis in the blood constitute an infectious serology group suggesting endemic fungal infection if positive and not PACNS whereas the absence of demonstrable fungal serologies, negative ANA titers and ANCA titers among other factors would suggest a diagnosis of PACNS.

2. Context

All articles providing opinions or patient data pertaining to the diagnostic process confirming (or refuting) PACNS in those afflicted with a "suspected PACNS" were included. Epidemiological, prognostic or treatment focused articles were not the focus of this analysis and were excluded if they do not provide information on diagnostic testing.

3. Searches

Reference databases (i.e. MEDLINE, and EMBASE) was queried. The search strategy identified articles 1) where some suspicion of PACNS was raised or mention of PACNS was made and 2) review the diagnostic process taken in pursuit of that diagnosis: [(Primary angiitis adj3 (cns or central nervous system)).tw,kf. **OR** (Granulomatous Angiitis or Granulomatous Arteritis).tw,kf. **OR** (primary and (cns or central nervous system) and (angiitis or vasculitis)).tw,kf. **OR** "Vasculitis, Central Nervous System"/ and primary.tw.] **AND** diagnos*.mp. Filters were applied to remove non-english and non-adult focused articles. Reports were included if they pertained to reviews and or discussions of guidelines. The review was confined to those articles where English translation was available. No restrictions were made with regards to the types of reports.

4. Inclusions / Exclusions

The target sample was those articles that discuss testing used to explore a potential diagnosis of PACNS.

Specifically, studies were included if: 1) the content of the article related to "suspected" cases of PACNS as previously defined in the operational definitions, 2) commentary within the article did discuss measures to rule in/out the diagnosis of PACNS when it was suspected

Studies were excluded if: 1) Information was not provided regarding the evaluation of subjects for possible PACNS; 2) The subjects presented, or the process discussed did not apply to those who were 18 years of age or older.

5. Data extraction (selection and coding)

a. Study selection

Eligibility criteria were applied by a single reviewer.

b. Data extraction

Data was extracted by a single reviewer. Information on testing, imaging or tissue sampling in patients suspected of having PACNS were gathered. A sectioned table was used to organize the information resulting from the review. The first table section reported on the testing found to be suggestive of PACNS and report on specific thresholds, where available, that differentiate PACNS from alternative diagnoses. A second table section catalogued the relevant imaging features of PACNS and those of other entities reported when PACNS was suspected. This table does not aim to create a comprehensive list of all imaging features of each clinical entity but, by focusing the literature search to when PACNS was considered in the differential diagnosis, it highlights features noted in a context where classifying the entity apart from PACNS was of significance. A third table section catalogued biopsy.

c. Risk of bias (quality) assessment

As the purpose of the review was an exploratory one to find candidate variables, no bias assessment was performed.

d. Strategy for data synthesis

The information on relevant testing was compiled and the context of the testing, in addition to the aforementioned summary tables, was narratively reviewed (i.e. questions addressed: what is the specific purpose of the testing item, it what situations would it be required, and what are the implications of the possible results).

D. Results

1. Process

Abstracts were selected if they discussed both: 1 - PACNS; 2- a process or approach for the diagnosis of PACNS. On full text review, articles were included for review if they presented an approach to the diagnosis of PACNS in adults and had an available English translation.

Of the 406 articles identified by the modified search strategy for this preliminary review, 75 were selected for full text having demonstrated a primary focus on an approach to PACNS diagnosis. Twenty-seven articles were excluded as full text was not available in 7, English translations were not available for 10, and pediatric cases were the focus in another 10 (See Figure 15).

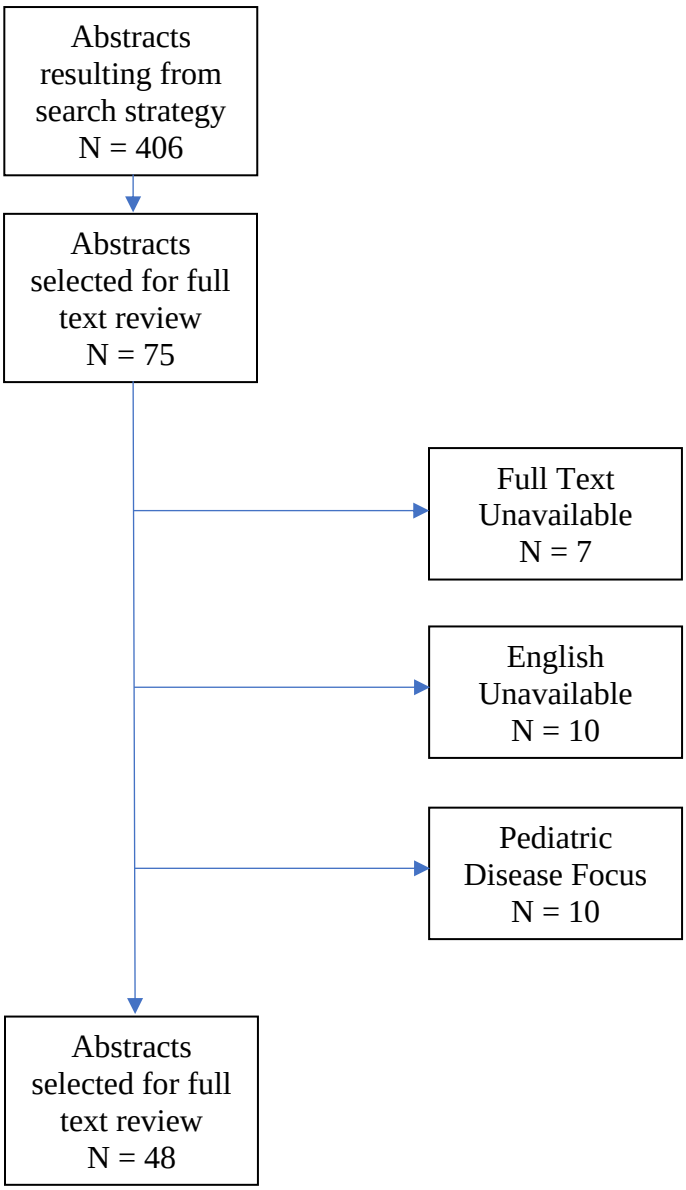


Figure 15. Scoping review for candidate variables inclusions / exclusions.

2. Content

a. Blood work

Vasculitis attributed to an infectious source (for which serologic testing is available) has been reported secondary to VZV, HIV, Lyme / Borrelia, Toxoplasmosis, Tuberculosis, Syphilis, Mycoplasma, Bartonella, Rickettsia, Aspergillus, mucormycosis, candidiasis, cysticercosis, coccidiomycosis, blastomycosis, histoplasmosis, hepatitis C, CMV and Pavrovirus B19.[lx] Listeria, Menigococcus and pneumococcus are also relevant considerations for which blood

culture may be obtained.[lxi] Other non-infectious, non-inflammatory vasculopathies can be investigated with targeted including blood smear with sickling(sickle cell anemia), ANA, dsDNA, C3/4 (lupus - favored to be an inflammatory source of non-inflammatory vasculopathy in the majority of cases), and ADAMTS13 (Thrombotic Thrombocytopenic Purpura).[lxii] Multiple stroke may mimic a diagnosis of vasculitis and be associated with anti-phospholipid antibodies (e.g., anti-cardiolipin, anti-Beta-2-glycoprotein, and lupus anticoagulant). Inflammatory systemic vasculitidies with cerebral involvement may be probed with extractable nuclear antigens such as SSA and SSB (Sjogren Syndrome), ESR (suggestive of giant cell arteritis when over 50 in the appropriate context), and ANCA/MPO/PR3 (ANCA associated vasculitis).

b. CSF analysis

i. Cellular analysis

A mild to moderate pleocytosis is considered consistent with the diagnosis of PACNS. A range of cut-off's in cellularity has been presented from 100 cells / micro-liter to 250 cells / micro-liter.[lxiii] Scenarios with "marked pleocytosis" are generally felt to reflect infectious or secondary to other inflammatory disease although a wide range in nucleated cell values has been reported (median of 20, range 0-575).[lxiv] On the lower end, mild pleocytosis has been felt to be consistent with other non-vasculitic disease like Fabry (particularly in males with violaceous angiokeratoma often found on the peri-genital regions) or RCVS (particularly in relation to thunderclap headache and hemorrhage).[lxv] Cell types are also of significance where heterogeneous cellular changes with increased populations of natural killer and b-cells could be consistent with PACNS but increase in CD4+ T-Cells may suggest an alternative like a beta amyloid related angiitis.[lxvi]

ii. Proteins

The overall CSF protein content of published cases ranges widely with one review citing a median 120 mg/dL (15 to 1034).[lxvii] Suggestive values typically range from 100 – 250 mg/dL.[lxviii] In addition to quantitation of protein in general, some specific proteins have been recognized as being of potential diagnostic significance. Amyloid beta A4 protein (APP) has been demonstrated to be reduced in PACNS. Neurofilament may be increased but this is a non-specific marker of neuronal injury and may be elevated following other neuraxial insults (e.g., stroke, multiple sclerosis attack, etc.).[lxix] Von Willebrand factor antigen, a protein relevant to the co-coagulation cascade which can indicate vascular endothelial damage when elevated, has been shown to be helpful in children but not adults.[lxx] Interleukin 17 (IL-17), a proinflammatory cytokine linked granulocyte / polymorphonuclear leukocytes (PMN) driven defense against extracellular bacteria and fungi as well as other autoimmune disease like psoriasis, has been proposed to suggest vasculitis in the CNS.[lxxi]

iii. Serology / PCR

Spinal fluid serology in pursuit of an infectious etiology is not commonly applied but can be considered. Examples include the lyme CNS antibody index for which titers normalized to albumin and IgG in both serum and CSF are compared. Unfortunately, true infection with leptospirosis or syphilis can interfere with such testing.[lxxii] Venereal Disease Research Laboratory (VDRL) is a non-tremponemal test applied to the CSF seeking confirmation of a possible diagnosis of syphilis. This is often applied after a treponemal test like FTA-Abs is

positive in the serum to verify syphilis as the cause in two phased diagnostic algorithms of neurosyphilis.[lxxiii]

c. Imaging Features

i. Angiography

Digital Subtraction Angiography (DSA) and Magnetic Resonance Imaging / Angiography are the two most popular techniques for the assessment of vasculitis in the CNS. Segmental stenosis followed by segmental dilation (i.e. post-stenotic ectasia) creates the classical "string of beads" appearance in contrast to stenotic regions re-expanding to as much as a normal caliber as can be seen in RCVS (i.e., "Sausages on a line").[lxxiv] RCVS may also show a graded or diffusely thinned appearance not expected of a CNS vasculitis and can be responsive to inter-arterial calcium channel blockade (e.g., nimodipine) or Phosphodiesterase-III inhibition (e.g., milrinone).[lxxv] In addition to caliber irregularities, visualization of aneurysms may suggest a genetic vasculopathy such as COL4A, particularly when in the presence of a family history of early ischemic / hemorrhagic stroke and or aneurysm while microaneurysms could be consistent with inflammatory vasculitis.[lxxvi]

Common descriptors on angiogram of CNS vasculitis related pathology include: abrupt / tapered occlusion, microaneurysm, neovascularization, distal branch attenuation, mass effect / lesion, distal branch attenuation, isolated stenosis and cuffing.[lxxvii] While angiography can be helpful in exploring a diagnosis of vasculitis, probing the vessels implicated in small vessel vasculitis (i.e. those less than 500 micrometers in diameter) may be and often is beyond the resolution of even DSA.[lxxviii]

ii. Vessel Wall Imaging

Vessel wall imaging has emerged as an increasingly popular technique to assess for vasculitis impacting the central nervous system. First practical incarnations related to suspected GCA for concentric vessel wall thickening of axillary and temporal arteries coined the "halo sign."[lxxix] The concept was later explored in MRI and has since evolved. Although early 2D approaches required time intensive image acquisition that limited the region of intracranial vascular visualization, 3D / isometric techniques have since emerged that allow for more complete coverage with an acceptable trade-off in resolution. Analogous to the use of fat saturation to assess for optic nerve inflammation, pre and post contrast T1 weighted sequences with "flow spoiling" to degrade the intravascular signal generated by flowing blood (i.e. black blood phenomenon) focus attention on the vessel wall.[lxxx] Longer segments of with concentric vessel wall changes are expected of PACNS in contrast to the sorter eccentric changes at vascular branching points seen in atherosclerosis.[lxxxi]

iii. Body

Computed Tomography (CT) imaging of the chest, abdomen and pelvis is often applied to identify evidence of an alternative processes such as lymphadenopathy in lymphoma or systemic involvement such as renal infarction or pulmonary hemorrhage in granulomatous polyangiitis.[lxxxii] In some cases CT angiography can look directly for systemic large / medium vessel vasculitis (i.e. of the aorta and its major branches) and be combined with positron emission tomography (PET) to ascertain changes in metabolic activity within areas of vessel wall enlargement.[lxxxiii]

a. Biopsy

i. Brain

Although brain biopsy is a fast and effective tool to rule out malignancy and infection, pathology can be missed in approximately 25% of cases owing to surgical inaccessibility of lesions and / or the potential for vasculitis to "skip" regions mixing areas of pathology with apparently healthy regions.[lxxxiv] A wedge biopsy of at least 1 cubic cm of brain tissue containing both gray and white matter is a proposed standard to minimize risk of under sampling tissue while balancing the excess risk of collateral damage to healthy tissues from larger biopsies.[lxxxv] Including leptomeningeal and cortical tissue / vessels can also help explore the distribution of pathology and driving factors such as cerebral amyloid.[lxxxvi]

When pathology is visualized, transmural arterial inflammation is expected in PACNS and three histologic subtypes are recognized: granulomatous, lymphocytic and necrotic/fibrinoid.[lxxxvii] Since its original description, granulomatous angiitis has largely been linked to pathological amyloid deposition in vessel walls leading to a reclassification of many cases as a secondary process known as one of A-Beta Related Angiitis (ABRA) or Cerebral Amyloid Angiopathy Related Inflammation (CAARI).[lxxxviii] Other sources of immunological activity that may present similarly to PACNS but possess a secondary trigger include Epstein Bar Virus where evidence of viral activity can be seen within CD-20 positive B-Cells (Lymphatoid Granulomatosis mimicking a lymphocytic PACNS)[lxxxix] and mycoplasma where electron microscopy has been noted to detect irregularities / inclusions.

ii. Alternative Sites

Renal, skin, muscle, and peripheral nervous tissue may provide insights depending on evidence of alternative sites of inflammatory activity. Temporal arteries are a well-known target when giant cell arteritis is suspected.[xc] When a diagnosis of intravascular lymphoma is being entertained, skin biopsy could be considered particularly where predictive factors exist, namely: unexplained fever (over 38 Celsius), alteration of consciousness, oxygen desaturation (less than 95%), thrombocytopenia (less than 120×10^9/L), serum LDH elevation (over 800 U/L), and elevated serum sIL2R level where available (over 5000 U/mL).[xci] The site of the incisional biopsy could be random although cherry/senile angioma have been described as a site of colonization for b-cell lymphoma and including this tissue in collections may improve the yield.[xcii]

E. Discussion

As demonstrated, a vast array of candidate variables can be applied to the identification of PACNS versus non-PACNS cases. Of particular importance are those variables which separate inflammatory causes of symptoms from vasoconstriction, malignancy and infection as treatments rendered for these entities can diverge substantially.[xciii] While imaging features and clinical characteristics are a readily accessible mainstay of diagnosis for RCVS and elucidating a malignant cause would often rely on biopsy from a site less costly than the brain, ruling out infection poses a great challenge in PACNS diagnosis.[xciv] Even in those cases where brain biopsy is required to separate suspected malignancy from PACNS delays in management are not as costly as in infection and eliminating the suspicion of an infectious mediator of a PACNS appearing phenomenon can cause a disabling delay definitive therapy.[xcv], [xcvi]

While few authors comment on a specific framework for infectious disease testing to a scenario where PACNS is being considered, there is some general agreement regarding that CSF culture and a serological battery to assess for potential infection is appropriate.[xcvii] For most proposed approaches this is included in a general stage of ruling out infectious causes of vasculitis and other systemic causes of vasculitis or mimicry although the unique implications with regards to inaction or incorrect administration of immunosuppression in infectious disease has led some to specifically develop this step.[xcviii] Although infections may have characteristic features that would direct specific testing, biopsy is often relied upon to rule out infectious causes of vasculitis and has been shown to result in a diagnosis of infectious vasculitis in as much as 15% of cases.[xcix]

Triggers for considering an infectious vasculitis identified included CSF cell count of greater than 250 cells/mL, low CSF glucose. VZV was often considered an infection to exclude though PCR on CSF or via serology.[c], [ci] Other infections supported for routine testing included in screening were HIV, Hepatitis B/C and syphilis.[cii],[ciii],[civ] Testing for tuberculosis, such as by interferon gamma release assay (IGRA) of the blood and CSF, is also relevant for those at higher risk including recent or known prior infection with tuberculosis or recent real or probable exposure to an active case (i.e. close contacts, emigrated from and endemic region, positive tuberculin skin test, frequent interactions with injection drug users, persons who work or reside with high risk individuals or in high risk contexts such as hospitals, homeless shelters, correctional facilities, nursing homes, and residential homes for those with HIV) or are immunocompromised (i.e. HIV positive, substance abusers, on immunosuppressants including high dose steroids, or have comorbidities like silicosis, diabetes mellitus, severe kidney disease, low body weight, organ transplant, or head and neck cancer).[cv, cvi], [cvii] Basilar meningitis is also suggestive but the pattern is not specific and additional screening for endemic fungus and fungal culture would be warranted.[cviii],[cix],[cx] Beyond this the decision regarding testing for specific infectious etiologies is governed by risk factor identification particularly in the presence of granulomatous or lymphocytic findings on biopsy (which constitutes the majority of presumed PACNS cases).

F. Conclusions

Although many authors identify a need for comprehensive investigation to rule out an infectious source of vasculitis, there is no consensus on what constitutes a thorough evaluation.[cxi], [cxii] While there is no commentary on what level of risk determines whether testing should be performed for a specific infectious entity, a practical general principal seems if any risk factor is identified then that would warrant further analysis for the specific disease of interest although the lengths to which one must go to rule out a disease process remain unclear. In this section we have shown that a variety of diagnostic considerations and tests exists when considering a diagnosis of PACNS and their application can be quite context specific. The following section, appreciating that context specificity can drive data heterogeneity and missingness, presents an approach for addressing missing data in machine learning.

Table 1. Proposed testing in cases of suspected PACNS

Test	Purpose
Blood	

Antibodies – Bacterial	Rule out syphilis
Antibodies – Viral	Rule out VZV
Antigen Testing	Cryptococcus, Hepatitis B
Antineutrophil Cytoplasmic Antibodies	Granulomatous polyangiitis, Eosinophillic Granulomatosis with Polyangiitis (Churg Strauss)
Antinuclear Antibodies	Lupus, Mixed connective tissue disease
Antiphospholipid antibodies	Sneddon syndrome (livedo recemosa, anti-cardiolipin)
Blood Smear	Malaria, Sickle cell
C3/C4	Low C4 in cryoglobulin related vasculitis
CBC / Platelets	Thrombocytopenic Thrombotic Pupura
CRP	Elevated in systemic inflammation
Cryoglobulins	Presenting Cryoglobulinemia
ESR	Elevated in systemic inflammation and often correlating with disease activity
Extractable Nuclear Antigen Antibodies	Sjogrens Syndrome (SSA/B)
Genetics	Cerebral autosomal dominant arteriopathy with subcortical infarcts and leukoencephalopathy [CADASIL syndrome], HERNS mutations, COL4A1 mutations, TREX1 mutations
IgA	HSP with elevated IgA
Interferon gamma release assay	Assess for tuberculosis
NMO/MOG	More helpful in children in fulminant inflammation
Rheumatoid Factor	Non-specific, suggestive of systemic inflammatory process
Serologies	Aspergillosis, Hepatitis B/C, endemic fungus (histoplasmosis, blastomycosis, coccidiomycosis), HIV, VZV
Von Willebrand factor antigen	Helpful in pediatric disease to detect idiopathic inflammatory vasculitis of the CNS

CSF	
Cell Count (under 250)	Infection if over 250
Cell Count (over 5)	Mildly lymphocytic pleocytosis is common in PACNS
Culture	Could suggest nocardia, mycobacteria
Flow Cytometry	Rule out cancer
IL-17 (elevation)	May be suggestive of vasculitis
Opening Pressure	Unclear benefit / significant elevation could suggest infection
Protein (>45 mg/d)	Usually elevated in vasculitis
Polymerized Chain Reaction (PCR)	Can suggest infection with VZV, Mycobacteria, Lyme
Serology	Can suggest infectious source with specific testing for VZV
Lyme Index (CSF/Serum Serology)	Rule out lyme: <1.3 negative, >1.5 positive

Imaging	
Angiography (Irregular / Post Stenotic Ectasia)	Suggestive of vasculitis
ECHO	Myxoma, Vegetations, Marantic endocarditis (anti-phospholipid antibody syndrome)
Extracranial Vessel Involvement	GCA, Fibromuscular dysplasia
Venogram	Cerebro-venous Sinus Thrombosis in Bechet
Lobar Hemorrhage	Can indicate fibrinoid subtype vasculitis but is uncommon in PACNS
Sulcal Hemorrhage	Can indicate RCVS
Scattered Infarcts	Non-specific
Vessel Wall	Smooth Concentric changes suggestive of vasculitis but can be seen in RCVS

Biopsy	
Brain	
EBV RNA in situ hybridization	Suggests lymphatoid granulomatosis
Transmural inflammation / type of inflammation	Granulomatous inflammation, lymphocytic cellular infiltrates, and acute necrotizing vasculitis suggestive of various forms of vasculitis
β-A4 amyloid	Suggests CAA / ABRA
Culture	Could suggest nocardia, mycobacteria
Alternative Sites	
Blind Skin biopsy	To assess for intravascular lymphoma

V. Chapter 2: Informal Review on Addressing Missing Data in Classification Techniques (focus on random forest)

A. Introduction

Given the number of potential features identified and summarized in Table 1 and the anticipatedly small number of cases of suspected rare entity (i.e., PACNS), a high degree of collinearity between candidate features was expected. Although the use of traditional techniques such as clustering and applying logistic regression or principal component analysis could be considered, data preparation / model selection (e.g., stepwise regression) could be tediously laborious and / or the interpretation of the results may not be intuitive. To address this issue machine learning was explored, specifically the use of random forests.

An additional problem exists, however. In the institutional PACNS dataset, a heterogeneous sample was anticipated where various tests were completed for some but not all patients. Dissimilar diagnostic approaches were therefor expected to effectively create substantial quantities of missing data. Missing data, a ubiquitous problem in statistical analysis, is often addressed with the dismissal of variable or cases or imputation as outlined in the narrative review

presented in the instruction (see section III. C). To formally ascertain the scope of novel approaches addressing missing data in the context of random forests, a structured review was undertaken.

B. Methods

The review was conducted using the MathSciNet database. The following search parameters were applied: "Anywhere=(random forest) AND Anywhere=(missing data)." Resulting abstracts abstracts were subsequently reviewed. Articles were included that provided a technique for managing missing information for application with random forest analysis. Selected articles underwent a full text review to ensure demonstrably effective approaches to missing data were discussed and the proposed techniques were extracted by a single reviewer.

C. Results

The search criteria yielded 20 articles. Of these articles 5 discussed imputation techniques but did not use them in conjunction with random forests, 2 did not have English text available, 3 did not have a complete text available (i.e., abstracts alone), and 1 focused primarily on Bayesian techniques. For the 9 articles with relevant content, there were two general themes to the articles. One was that random forest techniques could be used to fill missing data and the other was regarding an ability to adapt the technique to accept missing data either from a testing or training perspective. See Table 2 for a summary of the techniques proposed and their purpose.

Table 2. Missing data techniques applied with respect to random forests.

Technique	Purpose†	Description
Partitioning Around Medoids Clustering and Random Forest Classification (PAMRF)[cxiii]	Impute	Data is missing for a single feature and complete for all other features. Available data is clustered around medoids by first using a silhouette technique to identify the ideal number of clusters where the cost function penalizes greater distance to the medoid. A random forest technique attempts to learn the pattern of clustering without relying on the feature that has missing data entries and then cluster IDs are assigned to the cases where data was missing. The missing data is then drawn from other entries in the cluster and this can be done multiple times to provide “multiple imputation”.
Generative Random Forest[cxiv]	Train	Generative random forests use probability circuits to reconceptualize the problem from using a deterministic conditional distribution that typically directs random forest classification into a joint distribution that evaluates the intersects of various classifications and the associated variables. In other words the problem shifts from solving for $P(Y\|X_{observed}, X_{missing})$ where $X_{observed}$ is the available data and $X_{missing}$ is the missing data to $P(Y, X_{observed}, X_{missing})$. Clearly missing data cannot be used to solve the problem because it is unavailable but the missing data

		can be marginalized to solve for P(Y, $X_{observed}$) (i.e., $\int P(Y, X_{observed}, X_{missing})\, dX_{missing} = P(Y, X_{observed})$.
Surrogate Splitting[cxv, cxvi, cxvii]	Test	Decision trees are generated and used within a random forest but where missing data is encountered at a decision node a rule is created to decide if the presented case has a missing value that is super or sub threshold (i.e., decide which branch the data point should follow). Traditionally this rule uses a single alternative surrogate feature but developing a tree around the decision is possible. If multiple values are missing, then a hierarchy is used to find the next most appropriate substitute. Although this could be applied to training theoretically, it is practically discussed most often as part of a testing phase for data.
Random Forest Imputation (e.g., missForest) [cxviii, cxix, cxx]	Impute	The random forest technique is used itself to regress / impute missing values given the status of the other available features.
Probabilistic Assignment (i.e. factorization) [cxxi]	Train, Test	Data points are assigned to child nodes based on some rule that reflects randomness with some probability of the node following one path or the other. If one extends this to a scenario where out of bag error is computed and then recomputed following a situation where one pretends the data of one feature is missing for all data points then an importance measure can be constructed.

†Purpose reflects how the missing data technique was used

While the latter three techniques are presented are reasonably straightforward, the former two are more distinct and complex. PAMRF is described by the following pseudocode:[cxxii]

1. Input:
    1. Matrix of n cases with m features (i.e., $n \times m$ matrix). Note that the algorithm assumes at least p features have complete data
2. For each class impute values for members with missing feature entries:
    1. Find the optimal number of clusters with complete case data using a silhouette base method (k-means clustering shown).[cxxiii]
        1. Compute the silhouette width for data contain k clusters where k is a value between 2 and p
            1. Partition the data into k clusters
                1. Find the clustering pattern that minimizes clustering cost function (n.b. the squared differences between the means and composite elements in k-means clustering is shown)

$$\arg\min_{k} C_{K-means} = \arg\min_{k} \sum_{j=1}^{k} \sum_{x \in S_j^*} \|x - \mu_j\|^2 \qquad (25)$$

Where $C_{K-means}$ is the k-means related cost (n.b., this is the sum of squared differences between the mean and cases / elements within the set), k is the number of clusters, S_j^* is the set of cases within the jth cluster, x is the location of a case / element within S_j^*, and μ_i is the metric indicating the center of the cluster which in the case of k-means clustering is the mean / centroid (i.e., $\mu_i = \frac{1}{|S_i^*|}\sum_{x \in S_i^*} x$)

2. For ith case
    1. Compute the average intra-cluster dissimilarity $a(i)$ of the ith case within its assigned cluster:

$$a(i) = \frac{1}{|S_i| - 1} \sum_{j \in S_i, j \neq i} d(o_i, o_j) \quad (26)$$

Where S_i represents the set of cases within the cluster to which ith case is assigned, $|S_i|$ size of that set, and $d(o_i, o_j)$ is the distance between the ith and the jth case (note distance metrics vary by implementation and may use Gower or Euclidean distance for example).

    2. Compute the smallest average inter-cluster dissimilarity $b(i)$ of the ith case to its next closest cluster

$$b(i) = \min_{i' \neq i} \frac{1}{|S_{i'}|} \sum_{j \in S_{i'}} d(o_i, o_j) \quad (27)$$

Where $S_{i'}$ represents the set of cases within a cluster to which ith case is not assigned, $|S_{i'}|$ size of that set, and $d(o_i, o_j)$ is the distance between the ith case and the jth case within the $S_{i'}$ cluster.

    3. Compute the silhouette width contribution $s(i)$ of the ith case

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad (28)$$

3. Compute the cumulative sum (i.e., silhouette) over all $s(i)$ (i.e., $S(i) = \sum_i s(i)$)

2. Select the number of clusters with the largest silhouette (i.e., $\max_k S(i)$).

2. Apply PAM clustering to find medoids
    1. Randomly select q cases to represent the medoids
    2. Divide the cases / objects to find the minimum cost function

$$C_{medoid} = \sum_{j=1}^{k} \sum_{o \in S_j^*} d(o, med_j) \quad (29)$$

Where C_{medoid} is the medoid cost function and $d(o, med_j)$ is the distance between any case / object o within the jth set S_j^*, and its assigned medoid med_j.

    3. Recursively swap the medoid with every other case / object (i.e., non-medoid object) within the set and divide the cases / objects to minimize the cost function to find the best swap
    4. Stop the process when swaps no longer reduce the cost function and assign the cases to the clusters with the resultant medoids.
3. For those cases with missing data run a random forest regression to ascertain the classify the cases by their cluster identification using only those p features that have complete data as the input feature values
4. Assign values to the missing features for a case from those that are randomly drawn from feature values for members of the same cluster.

The generative random forest is a powerful but arguably less conventional construct where within trees comprising the forest internal nodes are replaced by sum nodes and leaves are replaced with distribution nodes.[cxxiv]

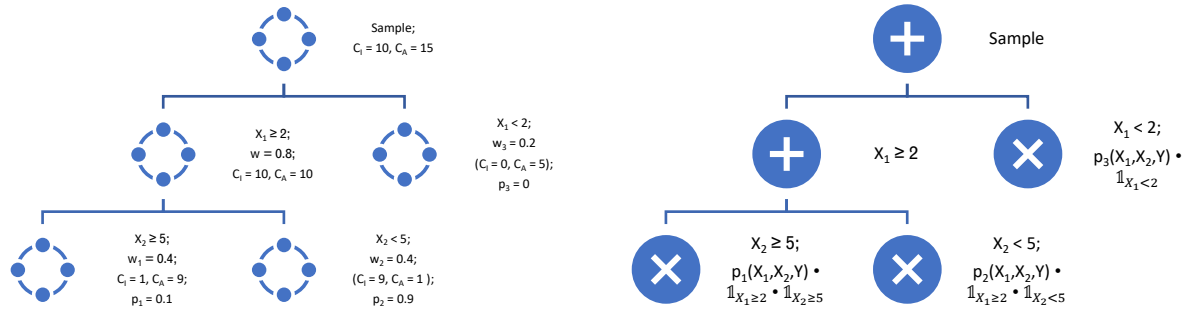


Figure 16. Example of a sample divided into two classes C_I (class of interest) and C_A (alternative class) using features X_1 and X_2 with a conventional decision tree (left) and a generative decision tree (right). Sum nodes were indicated by and encircled "+" and distribution nodes by and encircled "X". Resultant distributions are noted by p_1, p_2 , and p_3 as products with the appropriate indicator functions.

The formal conversion process described by Corriea et al. is as follows (generative decision tree pseudocode):[cxxv]

1. Input:
    1. Matrix of n cases with m features (i.e., $n \times m$ matrix).
2. Construct G' as structural copy of G (i.e., the decision tree of inters, See Figure 16) with v' as the corresponding node to v in G.
    1. For each internal node
        1. Partition the data (D_v) presented to the node by a deciding (decision) feature (X_i) at node v into data (D_u; note $D_{u,i}$ specifically references the data / feature values associated with the ith features withing the data set) to be passed to the child nodes (u)
        2. For each partition $u \in ch(v)$ (i.e., for each partition u which is an element of the children of v) compute the proportion distributed to the partition $w_{v'u'}u' = \frac{\sum_{x \in D_v} \mathbb{1}(x_i \in D_{u,i})}{|D_v|}$, where $w_{v'u'}$ is the proportion of cases with the feature value of interest versus the total in the data presented to the node v that are now distributed to node u (which are structurally the same as nodes v' and u' respectively), x_i are the feature values pertaining to the ith case, and x is the entire set of feature values.
        3. Let v' be a sum node $\sum_{u' \in ch(v')} w_{v'u'}u'$.
    2. For each leaf (i.e., terminal node)
        1. Let v' be represented by a density of $p_{v'}(x, y)$ over the support coinciding with the data presented in D_v.

When the data is complete the results of a random forest and a generative forest coincide perfectly. When a learner is trained with complete factorization the same approach as probabilistic assignment with nodal assignments proportionate to the outflow arcs generated by the data the nodal variable interest present although other methods with LearnSPM have been proposed.[cxxvi] Altering the weightings of nodal output arcs, akin to altering the rules in assignment for probabilistic assignment, have also been explored as a means of adapting to data MNAR. Appreciating the potential for significant bias using unaltered probabilistic assignments, Llerena et al. for proposed reweighting those nodes associated with unobserved values, in the context of unobserved data that is not ignorable by conservatively reweighting, such that the certainty of the classification versus the alternative is minimized.[cxxvii]

In generative random forests as described by Corriea et al., when predicting in the context of a missing feature the marginalization at any one leaf is a trivial process and the preponderance of probabilities that is extracted from an overall forest probability (assuming all trees are of equal weight):

$$p(X, Y) = \frac{1}{n_T} \sum_{i=1}^{n_T} p_{ti}(X, Y) \quad (30)$$

Where $p_{ti}(X, Y)$ is the probability derived from the appropriate node of the ith tree. From this the most probable classification can be extracted:

$$\arg\max_y P(Y|O) \quad (31)$$

Where y is the value of the class variable Y and O is the observed data variable.[cxxviii] It is of value to note that the power of the generative random forest comes from 2 critical properties of the predictive distributions formed as a probabilistic circuit these are decomposability at product nodes (i.e., leaf nodes in the case of generative random forests) and smoothness (i.e., internal nodes in the case of generative random forests). Decomposability is represented by:

$$p(x,y) = p(x)p(y) \tag{32}$$

Where x is the value of the available features. Smoothness is represented by:

$$p(x,y) = \sum_{i=1}^{n_l} w_{ni} p_{ni}(x,y) \tag{33}$$

Where w_{ni} represents the weight of the outflow arc, p_{ni} represents the joint probability of a set of features and classification at leaf node i and n_l is the number of descendent leaf nodes to a node of interest (i.e., in the case that $p(x,y)$ represents the probability of a confluence of variable values x and y for an entire tree, n_l would be the number of all the trees leaf nodes). These properties are afforded to the generative trees / forests as part of their modeling as naïve bayes networks which assume a subsequently observed event impacts the initial probability of interest with regards to observing some outcome (i.e., the model subscribes to Bayes theorem) and that features are both independent and equally contribute to the outcome (i.e., the model is "naïve").[cxxix, cxxx]

To illustrate the operation of the probabilistic circuit, briefly consider two scenarios that might be presented to the tree introduced in Figure 16, consider Figure 17:

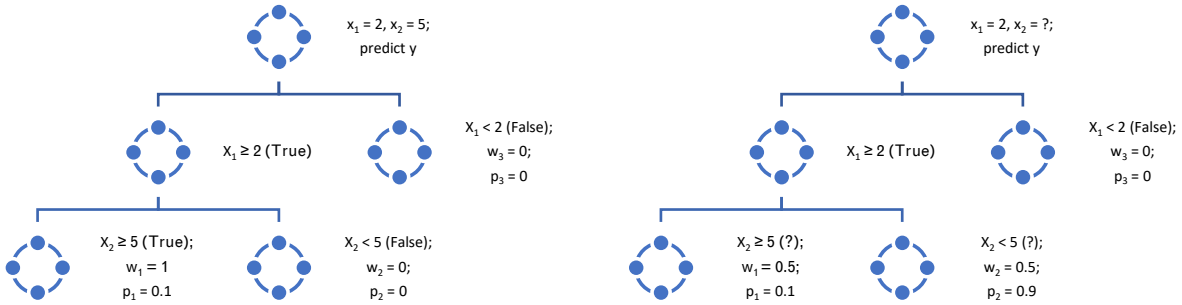


Figure 17. Two cases for prediction where $x_1 = 2$ and $x_2 = 5$ (left) and $x_1 = 2$ and x_2 is missing / unknown (right).

In the case where all values of X are known, the probability computation is trivial and is taken from the distribution of the first leaf node. The probability is therefore 0.1 or 10% (Figure 17 – left) that the case is classified as C_I. If x_2 were unknown then the distribution is represented by the confluence of applicable nodes (i.e., leaf nodes 1 and 2) and the resultant probability can be calculated per equation (33. The probability is therefore 0.5 or 50% (i.e., $w_1 p_1 + w_2 p_2 =$

$0.5 \times 0.1 + 0.5 \times 0.9 = 0.5$) that the case is classified as C_I. Simplistically, X_2 has been marginalized by weighted summation over all conceivable scenarios in the presence of x_2 missing. An alternative representation of this would be as follows:

$$\int f_{Y,X_1,X_2}(y,x_1,x_2)dx_2 = \int p(y,x_1,x_2)dx_2 = p(y,x_1) = 0.5 \quad (34)$$

Where $f_{Y,X_1,X_2}(y,x_1,x_2)$ represents the probability density function of interest.

D. Discussion

Overall, there are traditional approaches of data dismissal and imputation including techniques using random forests to impute data that can then be reused to further construct a random forest but only a limited number of approaches that adapt the algorithm to missing data. When models have been adapted, deterministic approaches with node customization as in the BEST technique or trifurcation have been popular while probabilistic approaches such as the use of probabilistic assignments and probabilistic circuits have been useful as well. While these techniques can optimize classification to varying degrees, the ability to classify using these models is not the only characteristic of importance and appreciating the individual capacity for features to infer a class may be of greater value.

To appreciate the impacts of various approaches on inference in the presence of missing data it is important to consider their potential impacts on the models in turn. With regards to variations of imputation techniques, PAMRF and missForest provided alternatives to traditional techniques with varying degrees of complexity. Of the traditional techniques previously described, single value imputation is the simplest to implement and the least computationally expensive. Unfortunately, as outlined in the missingness primer (Section III. C. 5.), imputed values are only as appropriate as the assumptions underlying their calculation.[cxxxi] For SVM, while the kernel used for determining the margin between classes does complicate discussion drawing on conventional intuition, using a mean or median value imputation technique is not so unlikely to place a point on the margin between two classes particularly if the data is split evenly between classes. In so doing the final model is likely to be altered / biased by this assumption should that point placement prove incorrect. The effect is quite different in a random forest whereby assigning all the missing data feature entries a single value, be it mean, median, mode, or some random value not tied with other feature values, creates an “island” that can bias a node formation by falling on one side of a potential threshold or another and can alter the selection of the appropriate threshold that maximizes an impurity drop (as discussed in Section III. C. 5.). If one were to instead use a k-nearest neighbors approach there may be a regression toward the mean in the event of classifying a rare / exclusionary class (i.e., a class that was typified by feature values that were normal thereby excluding alternative classification) may occur. If there are several alternative classifications (or a composite class containing a heterogenous grouping of other entities) any one or more abnormal feature(s) may be indicative of that class which could lead to missing data being filled with only non-indicative values and bias toward the exclusionary class. In more complex regressive imputation techniques such as multiple imputation, while the modelling of the missing values is more individualized, one assumes a stable pattern in the relationship between non-missing values and missingness. For cases that may have been generated /included based on the idiosyncrasies of various sources (e.g., the

decisions of several independent diagnosticians over a long period of time such as was argued to be the case in in the PACNS data) the projected distribution of appropriate missing values is likely invalid even if reasonable results are obtained. PAMRF presents a potential solution to this that endeavors to cluster those cases with similar attributes prior to imputation. Random forest-based imputation techniques on the other hand leverage their non-linear machinery to appreciate the inter-relationships between values in the dataset for its imputation. Both can adapt to clusters of cases where the reasons for and predictions of missingness may differ in the training set but suffer from their inability to deal with new predictions on cases with missing data unless all data remains available in perpetuity and the circumstances of the missing data match that of the case requiring imputation (i.e., one would need to properly impute the missing entries before then applying the model). As alluded to in the missing data primer, it is reasonable to assume that the reasons for missingness may differ from the data used to generate the imputation and classification models compounding subsequent classification error. A technique such as surrogate splitting could be used to partially offset this error by not requiring imputation by dividing cases on a surrogate variable. In some cases, a suitable alternative is obvious (e.g., height in centimeters versus height in inches) but in data sets where much missing data is expected, hierarchies to navigate nodes calling on missing data could be very complex and stable / appropriate surrogates hard to find.

When dealing with large degrees of missing data, probabilistic assignment and generative random present relatively seamless processes. In probabilistic assignment, when cases are reduced from a full weight to partial weights while travelling down the branches downstream from nodes attempting to partition the cases by a feature threshold value that is missing, the inherent assumption is that the missing data is missing but it is irrelevant (i.e., that the tendency of a case coming to a decision node to find itself in one branch or the other is modeled by the proportionate splitting of cases with known data). With many reasons for potential missingness, generative forests and its modified incarnations bring to bear a powerful concept that instances where missing data entries exist (i.e., multiple consecutive missing values within a tree) are merged to focus on partitioning and classification across known feature values. In contrast to probabilistic assignments where the focus is at a nodal level, global optimization of the tree structures can take place.

The mapping of random forest trees to a probabilistic circuit is an involved process where, at its core, this provides insights into classification by presenting a distribution of the confluence of events. There is much similarity to the probabilistic assignments. For example, consider a problem where you have access to a class Y and features X_1, X_2, X_3, X_4 and X_5. If a decision tree within a random forest is considered that serves to characterize the classification of Y given X_1 and X_2, the problem can be reframed into a probabilistic circuit (i.e., simplistically this could be seen as reframing the deterministic distribution which is possible due to decomposability).

$$f_{Y|\,X_1,X_2}(y,x_1,x_2)=\frac{f_{Y,\,X_1,X_2}(y,x_1,x_2)}{f_Y(y)} \qquad (35)$$

It is then possible to remove the impact of missing X_2 by considering all scenarios where X_2 was available (i.e., marginalizing) as shown in equation (34.

If this is applied to individual nodes as is the case in probabilistic assignment (where if, for example, 0.6 of the case is assigned to one daughter node and 0.4 is assigned to the other when a feature is missing in accordance with the overall typical frequencies of assignment [in this case 60% would typically go to the first node and 40% to the second when the feature is present]) the marginalization concept is quite similar but tree structures can be different depending on the precise rule applied for the probabilistic assignments versus generative tree optimization. In generative decision trees, even if one were to then try to re-weight nodes of tree assuming unobserved values the minimize classification certainty, in effect a distribution is still created at each node assuming a homogeneity of missingness that is drawn on to decide on how to partition the data. As mentioned previously, the missingness of the global dataset could be heterogeneous and so a subset experienced at any node could still contain a fair degree of heterogeneity, albeit likely lower the more partitioning that has taken place prior to encountering the node calling on missing data (i.e., those cases that were partitioned in similar ways by other variables likely have more similar distributions of the potential missing values than those cases that were not partitioned in a similar way).

Another issue in both probabilistic assignments and circuits, is that understanding what the importance of the variables with importance derived from either of these methods truly means in an intuitive sense. One would need to consider that the importance of variables is much different when certain other variables are missing (i.e., marginalized). If this information is not readily extractable, using that information to direct future behaviour around the collection of such data would be complex.

A major flaw of all methods described is that one way or another, an assumption is made about the nature of the missingness of the data and furthermore meaning is given in many cases to that missingness. This gives rise to several questions: When data comes from multiple sources, are the reasons for missingness sufficiently similar? For imputation-based techniques, are there sufficient features and auxiliary features to predict the missing values? In surrogate splitting, does the predictiveness of the surrogate change depending on the missing data pattern which is suggestive of different collection techniques / settings? In probabilistic techniques is it fair to assume that the cases will split in the same fashion as / or that is discernable from cases with known data? When careful consideration confirms that these techniques are not appropriate because the reasons for data missingness are MNAR / cannot be uniformly predicted it would be useful to have an approach that can consider the data at hand and make a best determination.

To put this in the context of a practical application consider the following analogy. If a random forest of trees was instead a grouping of diagnosticians a few concerns may arise from their assumptions. If one clinician said they placed importance on a variable to make a decision that was not available but knowing that 50% of the time it is was indicative of a diagnosis and 50% of the time it was not (probabilistic assignment) then one might worry about the confidence of their guess. If they assumed that it was some value based on cases with similar findings, but it was not clear certain critical features for describing how the cases were similar or not were available that might be concerning (imputation; e.g., missForest and PAMRF). If the diagnostician used another value instead that had similar properties to substitute for a feature they found to be important this would pose a challenge as well. If there were 1,000 diagnosticians and alternative might be to ask only those who had dealt with a situation with

similar parameters to give their input (novel RF variant) or ask them to do their best omitting the features that are unknown (generative random forest). It might be a better approach to simply ask the clinicians who had seen this combination of variables before what they thought.

E. Conclusions

While techniques do exist to address missing data for random forests and the classification performance of a traditional random forest approach may be inferior to generative techniques in the context of missing data for example, the interpretation is more straightforward in that the importance reflects the importance of variables when they are presented. In a practical sense this may provide less insight into the problem of "if you were to have only 5 tests to diagnose PACNS what tests would you choose," which importance estimates using probabilistic circuits would provide, but would better answer the question "are there situations in which these tests are useful." This ability to answer such a question is helpful in the development of guidelines because, when one is trying to convey the testing other clinicians might consider, one does not want to totally ignore rarer situations where features are needed to isolate a diagnosis particularly when the condition of interest is rarer to begin with. If techniques are used to impute missing data one must proceed with caution that the approach to imputation is appropriate for the data it is being applied to as this can introduce another layer of bias.

This section considered what techniques have developed to manage missing data in term of our machine learning modality of choice for the purposes of this thesis, i.e., random forests. It also has established that the novel approach proposed has not yet been characterized in the available literature and is compatible for use with known techniques. The next section proposed the novel random forest approach for comparison to established techniques in simulation and the institutional PACNS data series.

VI. Chapter 3: Construction of a Novel Approach to Random Forest Management of Missing Data

A. Introduction

Missingness, the reasons for missingness, and how to address missingness were central themes in this thesis and was the main impetus for generating the novel random forest technique. Very little data in a PACNS diagnostic data set would be expected to be missing completely at random (MCAR):

$$p(Missingness|y,x) = p(Missingess) \quad (36)$$

Where $p(Missingess)$ represents the probability of missingness and $Missingess$ represents missingness. The left combinations of missingness at random (MAR) and missingness not at random (MNAR). MAR would be formally defined as:

$$p(Missingness|y,x) = p(Missingness|y,x_{observed}) \quad (37)$$

Where $x_{observed}$ represents observed data / non-missing data. MNAR would formally be defined as:

$$p(Missingness|y,x) = p\left(Missingness|y, x_{observed}, x_{missing}\right) \quad (38)$$

Where $x_{missing}$ represents missing data.[cxxxii] While MAR is expected to represent missingness related to observed data (i.e., missingness of a feature is dependent on values that remain within the model; e.g., if ANA is negative indicating no nuclear specific antibodies were found in serum then an ENA panel looking to differentiate the targets of those antibodies would not be required and in not being performed the data for ENA would be "missing"), MNAR reflects missingness related to data that is not observed or fully incorporated into the models. Both would arise for a variety of reasons which are discussed in greater detail later alongside the modeling of missingness.

In this section, a dataset was generated from a linear system of equations and was augmented to provide source data with defined properties for characterization of the proposed novel random forest relative to established classification techniques.

B. Methods

1. Manufactured / Linear System Data Set Construction

a. Generation Procedure

The manufactured data set simulated data with a clear / linear relationship between variables and the classification outcome. A sample of n cases with F features was generated from a randomly generated cofactor vector matrix Q representing (i.e., a linear system of F equations to predict the classification of r classes of interest versus an alternative).

$$Q = \begin{pmatrix} Q_{11} & \cdots & Q_{1F} \\ \vdots & \ddots & \vdots \\ Q_{r1} & \cdots & Q_{rF} \end{pmatrix} \quad (39)$$

Where q represents the coefficient matrix where Qij represents the coefficient for the j of the ith equation and coefficients were drawn using a standard uniform distribution.

To then construct the matrix of simulated data, n values were drawn from a uniform distribution. The feature values were then solved for using the inverse matrix product with a vector (or case-wise singular values) comprised of randomly selected values.

$$Q \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_F \end{pmatrix} = \begin{pmatrix} Z_1 \\ \vdots \\ Z_r \end{pmatrix} \quad (40)$$

$$Q^{-1} \cdot Q \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_F \end{pmatrix} = \begin{pmatrix} Q_{11}^{-1} & \cdots & Q_{1r}^{-1} \\ \vdots & \ddots & \vdots \\ Q_{F1}^{-1} & \cdots & Q_{Fr}^{-1} \end{pmatrix} \cdot \begin{pmatrix} Z_1 \\ \vdots \\ Z_r \end{pmatrix} \quad (41)$$

Where there is one class of interest (i.e., $r = 1$) and one alternative class we obtained the following:

$$\begin{pmatrix} X_1 \\ \vdots \\ X_F \end{pmatrix} = \begin{pmatrix} Q_{11}^{-1} \\ \vdots \\ Q_{F1}^{-1} \end{pmatrix} \cdot Z_1 = Z_1 \begin{pmatrix} \frac{1}{Q_{11}} \\ \vdots \\ \frac{1}{Q_{1F}} \end{pmatrix} \quad (42)$$

Where Qij represents the coefficient for the j of the ith equation; r represents the number of class variables to be simulated; Z_i represents the ith of the randomly selected class probability values (note that one of the Z_i variables is a class of interest and the others may be overlapping and are not of interest to the simulated classification problem); and X_i represents the ith feature value for the given the Z value selections. The single classification probability value (Z_1) was selected and then recast into a categorical variable based on the magnitude of the value.

Not that, affording additional flexibility, the more general form of Equation (41 (i.e., the general form of the co-efficient matrix) as it seen to be implemented in python is:

$$\begin{pmatrix} Q_{11} & \cdots & Q_{1F^*} \\ \vdots & \ddots & \vdots \\ Q_{r1} & \cdots & Q_{rF^*} \end{pmatrix} \cdot \begin{pmatrix} X_{11} & \cdots & X_{1N} \\ \vdots & \ddots & \vdots \\ X_{F^*1} & \cdots & X_{F^*N} \end{pmatrix} = \begin{pmatrix} Z_{11} & \cdots & Z_{1N} \\ \vdots & \ddots & \vdots \\ Z_{r1} & \cdots & Z_{rN} \end{pmatrix} \quad (43)$$

Where N represents the total number of cases to be generated; and F^* represents the number of features available to describe the data but are not necessarily included in the model. Note that this approach was chosen due to its flexibility and that more complex relationships between values could be simulated by generating up to r random classification probabilities (Z values) and using only one of these Z values to dictate target class using some arbitrary threshold. With only one classification variable the feature variables are noted to be perfectly colinear unless randomization is introduced.

Equation (43 provides the ability to include F features that are part of the simulation and A auxiliary variables that have been left out of the simulation, where $F^* = F + A$, as well as r background classes / properties. Note that there would be a unique pseudoinverse matrix to Q so long as r is less than F^*.

Once a complete set of data was available, the augmentation process (described in the next subsection) then took place.

b. Augmentation of Simulation Data Sets

Prior to removing values to produce various degrees of missingness, feature values of cases were randomly redistributed to varying degrees creating noise / obscuring perfect predictability within the data. The first missing data mask was baseline entry mask that dictated for any given feature what the probability was that it could be missing. This is analogous to the clinical / diagnostic situation that certain tests may be considered invasive, expensive, or otherwise unavailable and that may impact the likelihood that they were performed (note this represents a form of data

missing completely at random). Each feature therefore has a certain factor that when exponentiated represents the relative probability that the feature is missing. For example, if the mask was an all-ones matrix assumed then the baseline probability of missingness would be the same across features (i.e., probability of factor missingness does not vary by case or feature). Further augmentation related to specific features of the cases was discussed as part of subsequent masks.

$$B = \begin{pmatrix} B_1 & \cdots & B_F \\ \vdots & \ddots & \vdots \\ B_1 & \cdots & B_F \end{pmatrix} = \begin{pmatrix} P_{11}^{e1} & \cdots & P_{1F}^{e1} \\ \vdots & \ddots & \vdots \\ P_{N1}^{e1} & \cdots & P_{NF}^{e1} \end{pmatrix} \quad (44)$$

Where B represents the baseline missingness mask, B_i represents the missingness exponents for the i^{th} feature and thus the B and resultantly the P^{e1} matrices are composed of N row vectors with coefficients equal to B_1, B_2,… B_F. P^{e1} represents the matrix of exponents that following exponentiation represent the un-normalized probabilities of missingness that are captured by the 1^{st} mask.

The second mask reflected the fact that missing data may be missing as the result of the ability of the party collecting the data to anticipate the value of a certain test and from this intuition omit the test (note that this data may be MAR or MNAR depending on whether the data used by the clinician to predict the missingness was retained within the predictive model or not). It is also possible, considering when cases were first assessed, that certain tests may have been more likely or less likely to be available after a certain time (generally MNAR). Although the party could be incorrect about their value prediction and time thresholds blurred (e.g., there was a transition period over which testing for a feature like neurofilament light chain serum testing became available) it would be anticipated that there could be some stable probability of missingness with certain underlying values or thresholds in values. Note that for the purposes of simulation "thresholds" could be inserted into the data frame as dummy features when a feature value threshold was triggered however in the presented simulations only the feature values themselves were used (i.e., threshold dummy variables were not specifically introduced).

$$D \cdot V = \begin{pmatrix} D_{11} & \cdots & D_{1F} \\ \vdots & \ddots & \vdots \\ D_{N1} & \cdots & D_{NF} \end{pmatrix} \cdot \begin{pmatrix} V_{11} & \cdots & V_{1F} \\ \vdots & \ddots & \vdots \\ V_{F1} & \cdots & V_{FF} \end{pmatrix} = \begin{pmatrix} P_{11}^{e2} & \cdots & P_{1F}^{e2} \\ \vdots & \ddots & \vdots \\ P_{N1}^{e2} & \cdots & P_{NF}^{e2} \end{pmatrix} = P^{e2} \quad (45)$$

Where D represents the dataset, V represent the feature value related missingness mask, D_{ij} represents data entries for the N^{th} case and F^{th} feature. V_{ij} represents the degree to which the value of the i^{th} element impacts the likelihood the j^{th} element of a case is missing, and P^{e2} represents the matrix of exponents that following exponentiation represent the un-normalized probabilities of missingness that are captured by the 2^{nd} mask.

The third mask addressed the association between missing values. Practically speaking, if one were to collect a type of sample (e.g., cerebrospinal fluid) and perform a certain test (e.g., spinal fluid cell count), it was more likely that there are other tests that would be applied to the same sample (e.g., spinal fluid total protein) or conversely if a test on a sample type were not present there was a probability that other related tests would not be completed possibly as the sample

was not available. Although a closed form solution / mask could be generated, this specific masking step for simplicity was applied iteratively. In other words, all masks were "aligned" and a data entry to eliminate selected. The third mask (dependent on what data entries are missing) was reapplied to the alignment after accounting for new missing data entries (i.e., constructed by taking the dot product of a to a dummy mask that indicates the location of previously selected missing values and a matrix that indicates the inter-relationship of missing features). Note that alignment here indicates the entry-wise summation of all mask terms that upon exponentiation of the sums of these terms which is equivalent to entry-wise multiplication of the exponentiated masks representing relative probabilities

$$M \cdot I = \begin{pmatrix} M_{11} & \cdots & M_{1F} \\ \vdots & \ddots & \vdots \\ M_{N1} & \cdots & M_{NF} \end{pmatrix} \cdot \begin{pmatrix} I_{11} & \cdots & I_{1F} \\ \vdots & \ddots & \vdots \\ I_{F1} & \cdots & I_{FF} \end{pmatrix} = \begin{pmatrix} P_{11}^{e3} & \cdots & P_{1F}^{e3} \\ \vdots & \ddots & \vdots \\ P_{N1}^{e3} & \cdots & P_{NF}^{e3} \end{pmatrix} = P^{e3} \quad (46)$$

Where M is the binary matrix representing the locations of missing data, I is the missing value inter-relation mask, M_{ij} are binary / logical variables which represent the missingness of data with respect to the N^{th} case and F^{th} feature (i.e., the value of M_{ij} is 1 if the j^{th} feature of the i^{th} case is missing and 0 if it is present). I_{ij} represents the degree to which missingness of the i^{th} element impacts the likelihood the j^{th} element of a case is missing. P^{e3} represents the matrix of exponents that following exponentiation represent the un-normalized probabilities of missingness that are captured by the 3rd mask.

$$\begin{pmatrix} P_{11}^{e1} + P_{11}^{e2} + P_{11}^{e3} & \cdots & P_{1F}^{e1} + P_{1F}^{e2} + P_{1F}^{e3} \\ \vdots & \ddots & \vdots \\ P_{N1}^{e1} + P_{N1}^{e2} + P_{N1}^{e3} & \cdots & P_{NF}^{e1} + P_{NF}^{e2} + P_{NF}^{e3} \end{pmatrix} = \begin{pmatrix} P_{11}^{eC} & \cdots & P_{1F}^{eC} \\ \vdots & \ddots & \vdots \\ P_{N1}^{eC} & \cdots & P_{NF}^{eC} \end{pmatrix} = P^{eC} \quad (47)$$

Where P^{eC} represents the matrix of exponents that following exponentiation represent the un-normalized probabilities of missingness that are captured by all probability masks. This is otherwise known as the "aligned" composite mask.

$$e^{P^{eC}} = e^{\begin{pmatrix} P_{11}^{eC} & \cdots & P_{1F}^{eC} \\ \vdots & \ddots & \vdots \\ P_{N1}^{eC} & \cdots & P_{NF}^{eC} \end{pmatrix}} = \begin{pmatrix} e^{P_{11}^{eC}} & \cdots & e^{P_{1F}^{eC}} \\ \vdots & \ddots & \vdots \\ e^{P_{N1}^{eC}} & \cdots & e^{P_{NF}^{eC}} \end{pmatrix} = \begin{pmatrix} P_{11}^{C} & \cdots & P_{1F}^{C} \\ \vdots & \ddots & \vdots \\ P_{N1}^{C} & \cdots & P_{NF}^{C} \end{pmatrix} = P^{C} \quad (48)$$

Where P^{C} represents the matrix of un-normalized probabilities of missingness that are captured by all probability masks. This is otherwise known as the raw probability mask.

$$\begin{pmatrix} \overline{M_{11}} \times P_{11}^{C} & \cdots & \overline{M_{1F}} \times P_{1F}^{C} \\ \vdots & \ddots & \vdots \\ \overline{M_{N1}} \times P_{N1}^{C} & \cdots & \overline{M_{NF}} \times P_{NF}^{C} \end{pmatrix} = \begin{pmatrix} P_{11}^{R} & \cdots & P_{1F}^{R} \\ \vdots & \ddots & \vdots \\ P_{N1}^{R} & \cdots & P_{NF}^{R} \end{pmatrix} = P^{R} \quad (49)$$

Where $\overline{M_{\iota J}}$ represents the complement of M_{ij} or the presence of data with respect to the N^{th} case and F^{th} feature (i.e., the value of M_{ij} is 0 if the j^{th} feature of the i^{th} case is missing and 1 if it is present). P^{R} represents the matrix of un-normalized probabilities of missingness that are captured

by all probability masks for the entries that have not been nullified. This is otherwise known as the refined probability mask. For the purposes of selecting missing values this is normalized as follows:

$$\frac{1}{S^R}\begin{pmatrix} P_{11}^R & \cdots & P_{1F}^R \\ \vdots & \ddots & \vdots \\ P_{N1}^R & \cdots & P_{NF}^R \end{pmatrix} = \begin{pmatrix} \frac{P_{11}^R}{S^R} & \cdots & \frac{P_{1F}^R}{S^R} \\ \vdots & \ddots & \vdots \\ \frac{P_{N1}^R}{S^R} & \cdots & \frac{P_{NF}^R}{S^R} \end{pmatrix} = \begin{pmatrix} P_{11}^N & \cdots & P_{1F}^N \\ \vdots & \ddots & \vdots \\ P_{N1}^N & \cdots & P_{NF}^N \end{pmatrix} = P^N \quad (50)$$

Where P^N represents the matrix of normalized probabilities of missingness that are captured by all probability masks for the entries that have not been nullified probabilities of missingness. S^R represents the sum of all un-normalized probabilities of missingness which is defined as follows:

$$\sum \begin{pmatrix} P_{11}^R & \cdots & P_{1F}^R \\ \vdots & \ddots & \vdots \\ P_{N1}^R & \cdots & P_{NF}^R \end{pmatrix} = \sum_{i,j}^{N,F} P_{ij}^R = S^R \quad (51)$$

Once the probability masks were aligned, the composite exponentiated mask was nullified where there was already a missing entry and all entries are linearized. A random number was selected between 0 and 1 to successively find the next entry for nullification where the value nullified was indicated by the indices for which the CDF_M was less than the randomly generated selection but the was CDF_M greater after adding the next co-efficient.

$$CDF_M = \sum_{i,j}^{N,F} P_{ij}^N = \sum_{i}^{N}\sum_{j}^{F} P_{ij}^N \quad (52)$$

Where CDF_M represents the cumulative distribution function of missingness with summation over columns within a row and then further rows are summed

The iterative nullification of data continued until a preset level of sparsity was achieved. Sparsity was defined as the proportion of data entries missing within the data matrix.

$$Sparsity = \frac{\sum_{i,j}^{N,F} A_{ij}}{\sum_{i,j}^{N,F} 1} = \frac{\sum_{i,j}^{N,F} A_{ij}}{N \times F} \quad (53)$$

Where A represents the absence of data which is 1 for any given I and j when no data is present and 0 when the data is present.

Finally, as the features available in the wine set were strongly predictive of classification, random swaps of values between rows/cases but within columns/features were completed empirically to reduce model accuracy / introduce noise into the features.

c. *Summary of Data Generation Variable Values*

For the purposes of this simulation a sample of 178 cases (n = 178) with 13 features (F = 13) was generated (i.e., the same shape as the wine data set). A randomly generated cofactor vector matrix Q representing a linear system of 13 equations was therefore constructed to predict the classification of one class of interest versus an alternative (r = 1). The single classification probability value (Z_1) was selected to be 0.5. Thus, values less than 0.5 represent the target class and those greater than 0.5 represent the alternative.

The simulation data was generating by creating data characterized by 50% randomness across features with decreased randomization of feature 0, 4, 5 (25%) and increased randomization of features 2, 6, 12 (75%). To create a high degree of missingness a 50% nullification of data was performed. Each matrix was separately applied, as explored in the discussion, to generate perceptible shifts in variable importance measures. All masks were then applied in unison with the following missing value matrices:

$$B = (5 \quad 5 \quad 1.25 \quad 1.25 \quad 1.25 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 5 \quad 5)$$

Note that this base missingness vector was a composite of two explored vectors in the discussion.

$$V = \begin{bmatrix}
1.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.8 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

Note this value missingness matrix was a variation of the own value simulation described in the discussion with the addition of few off axis entries of similar magnitude to the on-axis entries.

$$I = \begin{bmatrix}
0 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -20 & -20 \\
20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -20 & -20 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

0	0	0	0	0	0	0	0	0	0	0	0	0
-20	-20	0	0	0	0	0	0	0	0	0	0	30
-20	-20	0	0	0	0	0	0	0	0	0	30	0

Note these missingness matrices are later applied with varying combinations to facilitate the discussion of the results of the simulation by demonstrating the results of one missingness pattern at a time.

2. General Data Preprocessing

Data was processed using standard techniques to ensure applicability of the models. Prior to training and sorting by the classification algorithms, categorical data was divided out into binary dummy variables when appropriate and feature columns were normalized. Although it would not be expected to significantly impact the construction of the random forest classifiers and thus was not a typical step in that context, this would impact the outcomes of the support vector machine-based algorithm in subsequent analysis. To simplify / maximize comparability, all algorithms had these preprocessing steps applied.

3. Strategy for Data Partitioning / Preparation

Algorithmic identification of manufactured data class (wine type / class and the diagnosis of PACNS the subsequent chapters) were modeled using recursive partitioning. All generated features in the manufactured data set (and all available features in the later presented wine data set) were used for partitioning. The data was generally divided in to "test(ing)" and "train(ing)" sets to facilitate validation and a random forest model or SVM was generated using the randomForest package using Scikit in python.[cxxxiii] Approximately 20% of the full sample was reserved in the "test" set to evaluate metrics of model accuracy including sensitivity and specificity. As the focus of the classification process was the class in general, the sample was split by presence or absence of the class (wine type of interest or the presence or absence of PACNS in the analysis later performed on the wine simulation and institutional datasets) prior to division ensuring 15 – 25% cases of the known target class (wine type in the later presented wine data set and PACNS diagnosis in the institutional set) were available in the "test" set. Sampling with replacement from a randomly selected complete factor space (i.e., data was subsampled at random to provide subsets of complete data) were used to create the learners that combined to generate the final model using the confluence of their votes on a given sample (i.e., the class or diagnosis with the largest number of votes or "hard" voting).

4. Imputation

For data set fed into all the algorithms (except the novel RF), mean value imputation was used to preserve comparability across models. A kNN approach was considered as it had been shown to perform well even when a large quantity of missing data exists.[cxxxiv] The focus of the thesis however was principally to consider the ability of the classification algorithms to negotiate missing data concerns and thus mean value imputation simplified the understanding and discussion of the strengths / weaknesses of the various applied techniques.

5. Algorithm Tuning / Simulation Parameters

a. Overview

Before providing the multifaceted missingness simulation (and subsequent simulations presented for discussion with different combinations of missing data) we discussed here the parameters for the simulations. In all simulations, the sparsity / proportion of missing data was set to 50% in accordance with contemporary studies that have considered this to be a very high degree of missingness.[cxxxv] In general, when applying the classification algorithms, the default settings set within the scikit learn package were used.[cxxxvi, cxxxvii] Justifications for parameter sections were discussed with the benefit of focused simulations in the discussion section. In each simulation the degree of distortion by missingness (or the degree of randomness, as was the case in the initial simulation presented in the discussion section) was incrementally adjusted / increased until the AUC for the median simulated ROC curve of the standard random forest technique with single value imputation was 95% or less. The random forest curve was selected for this purpose due to its demonstrated classification robustness to feature noise.[cxxxviii]

b. Parameters By Algorithm

i. SVM

In the SVM algorithm a linear kernel was used. The simulated data was generated with a single threshold and thus linear behaviour was anticipated by design. This made consideration of any kernel of greater complexity unnecessary. Similar behaviour was anticipated of the PACNS dataset in that the focus of the pathology was within the brain and thus the classification of PACNS versus all other pathologies would favor normalcy across a significant majority of features, particularly in the case of biochemical variables. The notable exception would be of those variables on the other extreme that have been specifically linked to PACNS (e.g., biopsy) or vasculitis (e.g., vessel wall enhancement) but coded characterizations would be binomial (e.g., the presence of fibrinous necrosis in blood vessels, the presence of concentric vessel wall enhancement, etc.) and thus have only the option for a single linear division. Of note, a gaussian kernel could have been a reasonable alternative as the bulk of laboratory testing is presented as "normal" depending on a gaussian relationship to "healthy average" individual. However, again in support of a linear threshold / division, it is often one extreme of any given laboratory data set that is of concern thus forming a single bound of practical relevance (e.g., creatinine values of 3 standard deviations above the mean are pathological where those 3 standard deviations below the mean do not raise suspicion of a pathological entity).

The wine data set on the other hand has been successfully analyzed with sound justification using a gaussian kernel in the past.[cxxxix] In contrast to this analysis by Cortez et al., the data set used for the purposes of the analysis presented within this thesis was reduced from a 3 class to 2 class classification problem. Assuming the feature values were homogeneous / normally distributed within a class, this could technically sandwich properties of the target class (i.e., class_0) between the two non-target original classes (i.e., class_1 and class_2) the less complex linear kernel was preferred for reasons of simplicity and comparability across simulations. It was shown in the following analysis that all models tested on the wine dataset classified the target class with excellent accuracy which supported this simplification.

c. Random forest (Standard and Novel)

i. General

With regards to the random forest design, several hyper parameters were set to allow for comparable execution. Each forest was composed of 1000 trees and 100 iterations were performed to create the composite ROC plots.

ii. Number of Trees

Random forests are generally benefitted by the addition of more trees however there is a practical trad off between the number of trees and processing time.[cxl], [cxli] In the simulation, a clear relationship was established between the features and classes and for the wine data set there was a well-established predictiveness of the features to the wine classes of interest.[cxlii] Within the PACNS data there is an opportunity for several noisy features to exist however, if both models were to have the same number of trees during a head to head analysis, it was not anticipated that the number of trees would significantly benefit the traditional or novel RF variant as the novel RF variant is a special case of the traditional RFs. While the numbers of trees could have been interrogated against error to select an ideal number for any one data set (i.e., the simulated data, wine data, and the PACNS data sets), comparable parameter application across data sets was favored. For the purposes of the presented analysis, forests of 1,000 trees were generated.

iii. Cases per node

When creating decision trees, a hyperparameter of relevance was the number of cases per node. We have used 2 as the minimum number in our simulations which meant that when a data set is divided, 2 cases must exist in each daughter arm at a minimum. With a value of 1, each tree would have another conceivable level of nodes. The value of 2 was selected primarily to facilitate improved processing speed given the number of forests generated 100 times over although theoretically this may have been of some consequence.

Considering the extreme case if, given the bootstrapped sample, there were one or no cases of either the classification of interest or its alternative the tree was automatically a null tree and all variables have equal or no reported importance for that tree. For novel trees that was more likely to happen for missingness that was associated to the classification be it direct or indirect (i.e., missingness was based on the value of the factor or other factors that are themselves predictive of outcome) as cases could be expected to have unbalanced scarcity on one or the other side of the threshold. These trees were therefore less “rich” as the number of cases they represent of one or the other class was less relative to the minimum leaf size. Although reducing the minimum samples per leaf to 1 would seem reasonable, there would still be residual effects on the assessment of variable importance particularly where randomness presents additional challenges. If there were error in the small population of cases on one or the other side of the threshold, then the impacts of that error can be more variable than when there is a larger number of cases.

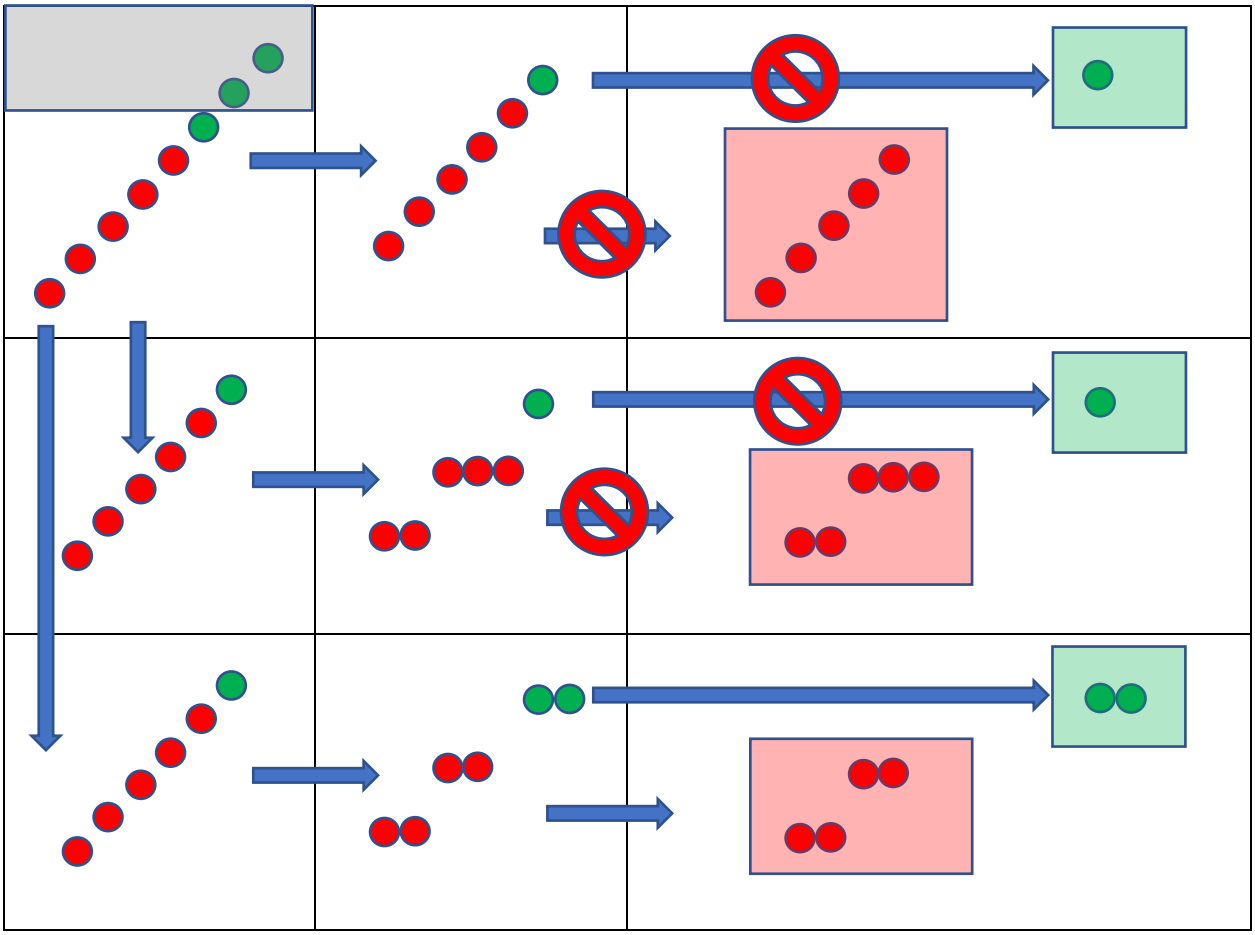


Figure 18. Potential data division with a minimum case count of 2 showing generation of a data set where missingness was value dependent and the subsequent consequences to classification. (top) raw data with value dependent dismissal of cases in the grey box and an attempt at direct classification blocked by the ideal split leaving one daughter leaf with fewer than 2 cases (i.e., a single case); (middle) data with value dependent missingness following bootstrap selection blocked again due to insufficient cases arising in a group following an ideal split; (bottom) data with value dependent missingness following bootstrap selection succeeding

iv. Cardinality Correction

For the purposes of importance measure computation, cardinality correction was applied to features with discrete values (i.e., categorical variables, ordinal variables, and nominal variables). This was not a standard approach in random forest construction and represents a novel approach to addressing biasing random forest impurity driven feature importance measures from features with discrete values of fewer levels to those with more levels or continuous features. This section briefly reviewed this issue and describes the correction process with the benefit of simulations.

Consider a brief experiment that addresses the notion that a random forest overestimates the importance of continuous variables or discrete variables with higher cardinality versus discrete variables with lower cardinality. Techniques have been developed to address this including conditional inference frameworks although a different approach is provided here.[cxliii] Simulations are shown that both preserved and failed to preserve the potential for a variable to generate ideal divisions when reduced to a bilevel variable. Consider Figure 19. Any set of real numbers without degenerate / inseparable states (i.e., ties) could be partitioned by a continuous variable

into any number of groups equal to or less than the number of data points / cases (top row) and so long as one or combination of those divisions represents an ideal split, there was no loss of information with respect to discerning the difference between classes (middle row: 2 groups perfectly divided, 3 groups perfectly divided when the lower 2 are combined, and 4 groups perfectly divided when the lower 3 are combined). One could then add back some element of randomness without interfering with this partitioning (bottom row) which would subsequently be referred to as pseudo-randomness / pseudo-cardinality. Of course, this randomness could be used to provide partitioning opportunities in non-ideal splits as exemplified in Figure 9.

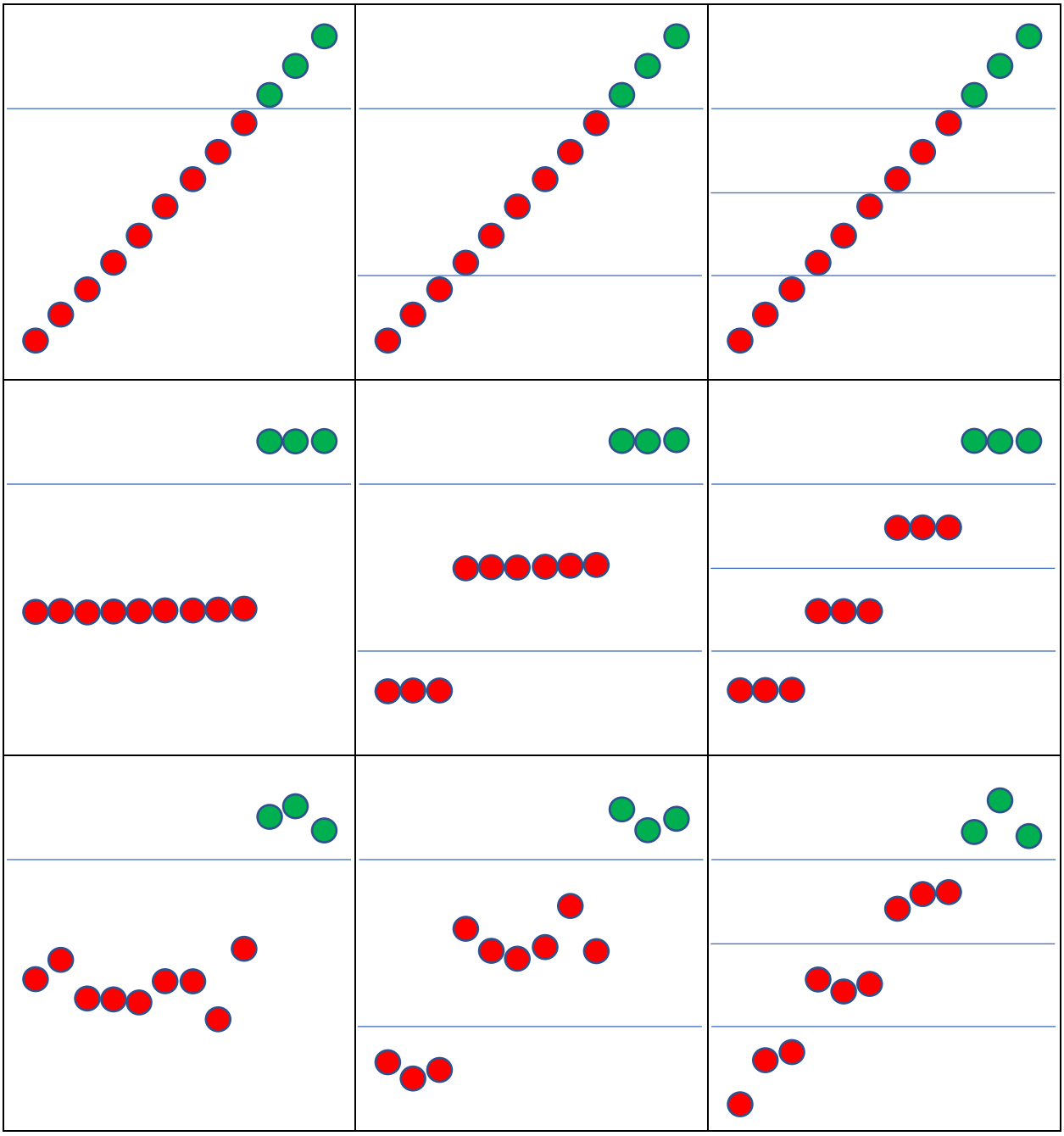


Figure 19. With the variable of interest on the vertical axis and one class in red and the other in green: continuous variable partitioned neatly into two groups (a: top-left), three groups (b: top-center), and four groups (c: top-right); discrete variable

partitioned neatly into two groups (d: mid-left), three groups (e: mid-center), and four groups (f: mid-right); and a pseudo-discrete variable partitioned neatly into two groups (g: bottom-left), three groups (h: bottom-center), and four groups (i: bottom-right).

There was a potential for lost information in discrete versus continuous presentation of a variable if from its continuous form the variable is portioned non-ideally or if excessive randomness is added back to multi-level variable (see Figure 20).

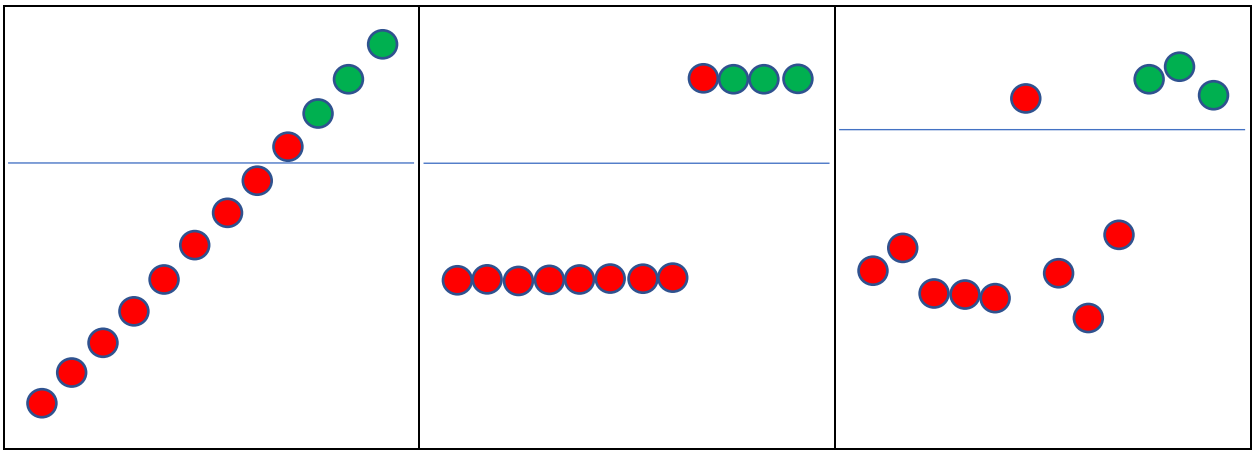


Figure 20. With the variable of interest on the vertical axis and one class in red and the other in green: neat division for a continuous variable (left); imperfect division of a bilevel variable (center); imperfect division of a bilevel variable due to excessive pseudo-randomness.

One can consider that for practical purposes that ordinality may not be preserved / retain its importance with respect to the classification of interest and that a multi-level variable may be separated into n dummy variables with pseudo-randomness (see Figure 21). While this may be used to compare the process to linear techniques, such a division is arguably not as important using non-linear techniques. If the goal is to decide on the effectiveness of a test one might have to consider corrections such as incorporating all dummy variables when the parent variable is selected to generate a tree in a random forest. Note there is also a need to introduce n as opposed to n – 1 dummy variables as well unless a node selecting the nominal variable is able to make a compound split (i.e., able to simultaneously use the data from all dummy variables to allow the random forest algorithm to see separations of the base state as achievable in one step as opposed to n – 1 steps and thus not disadvantage the partitioning out of the base state).

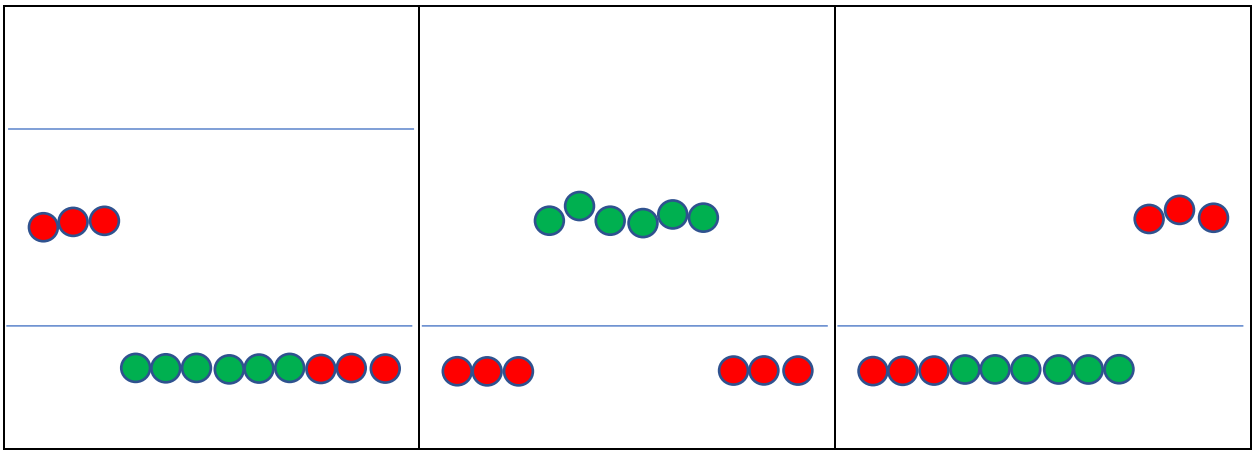


Figure 21. With the variable of interest on the vertical axis and one class in red and the other in green: original tri-level variable divided up into dummy variables with dummy bi-level variable for original first level (left); dummy bi-level variable for original second level (center); dummy bi-level variable for original third level (right)

Consider a situation where all variables possess the same potential to classify the outcome variable, the feature values are 50% randomized, and there is no missing data. The importance assessments by Gini importance are similar (see Figure 22).

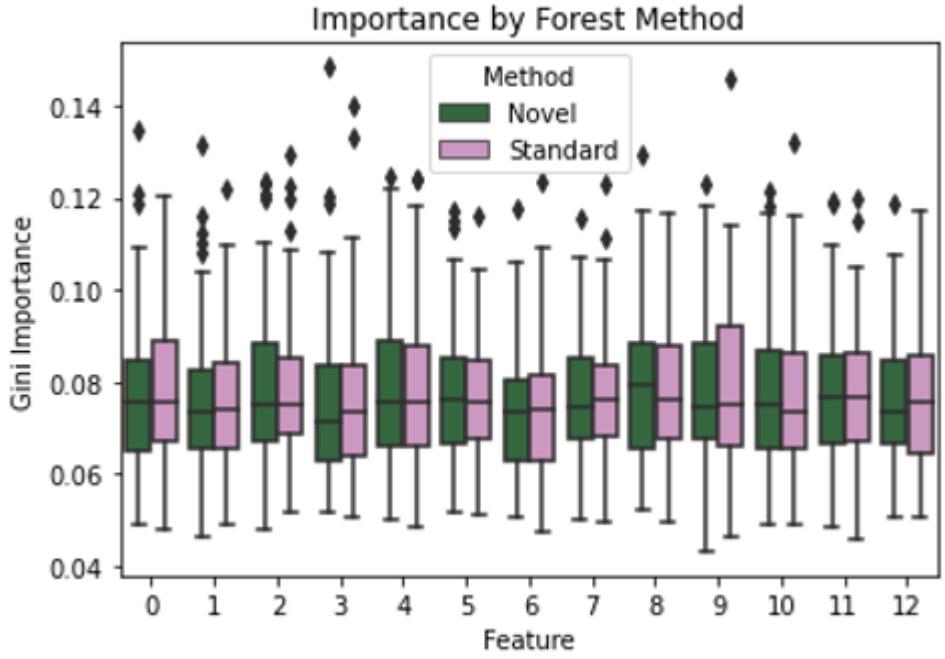


Figure 22. Feature importance where all features are linear combinations of one another and are continuous with 50% randomization of the feature values.

Now consider the scenario where one variable is swapped with its bilevel counterpart but retains its discriminative potential (i.e., an ideal split is available) as in Figure 19 d. Note that there is not any difference in importance perceived if no noise / randomness is introduced (see Figure 23).

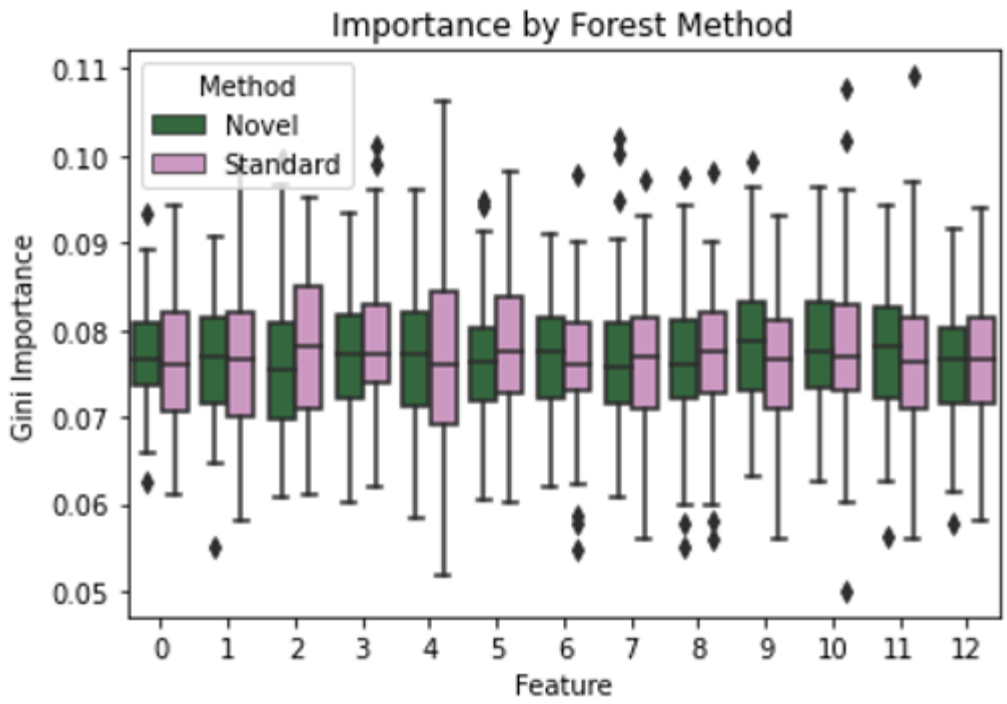


Figure 23. Feature importance where all features are linear combinations of one another and are continuous except for feature 2 (which is bilevel) with no randomization of the feature values.

Where the split is suboptimal the importance assessment is very different (see Figure 24). The importance of the second feature is effectively negated.

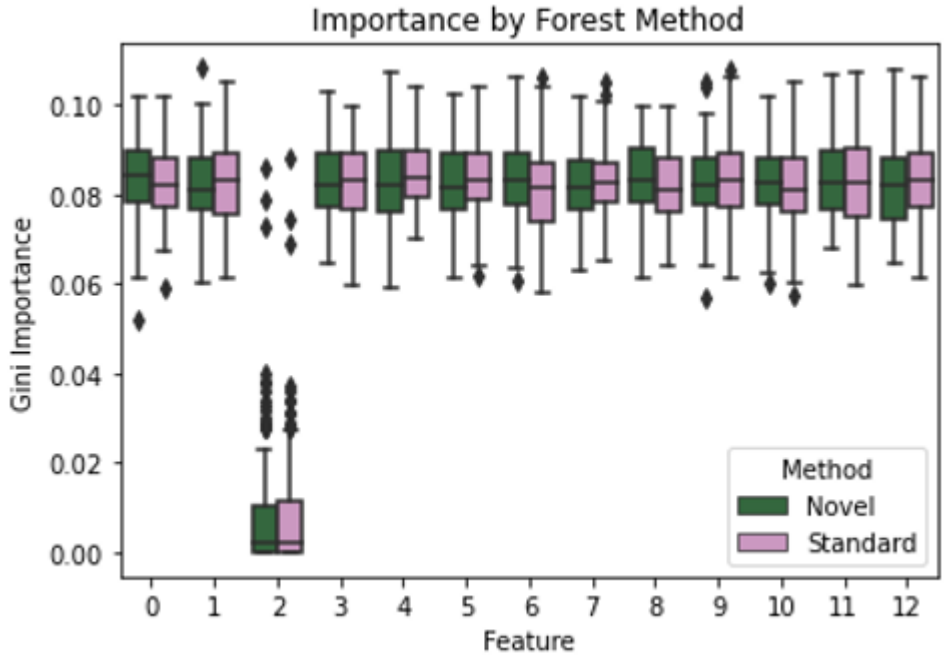


Figure 24. Feature importance where all features are linear combinations of one another and are continuous except for feature 2 (which is bilevel with an imperfect threshold) with no randomization of the feature values.

If one introduces general noise / randomness, then the importance of the bilevel feature is compromised (see Figure 25) even when an ideal split is made however the introduction of pseudo-randomness of the feature levels does alleviate this change (see Figure 25)

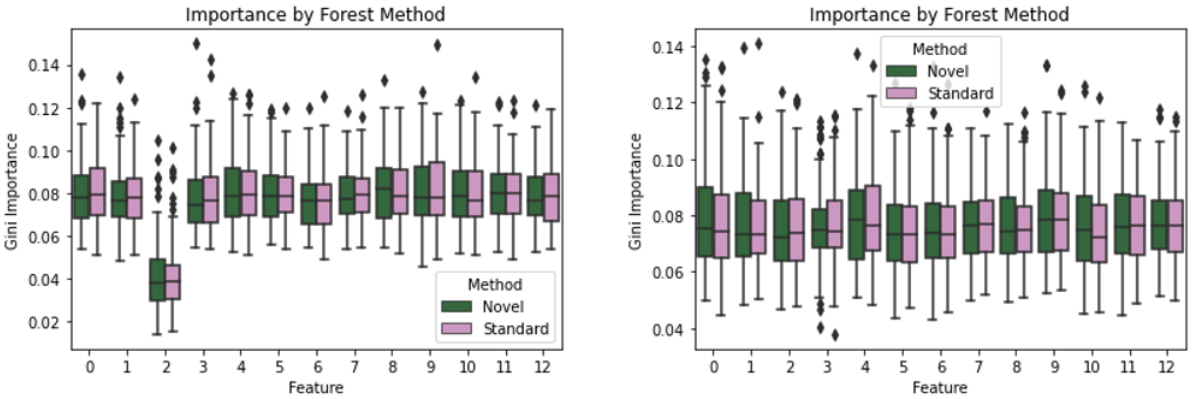


Figure 25. Feature importance where all features are linear combinations of one another and are continuous except for feature 2 (which is bilevel) with 50% randomization of the feature values; (a: left) No pseudo-randomness has been introduced; (b: right) pseudo-randomness has been introduced.

The importance of the feature can then again be reduced if its ability to make the distinction is diminished which is to be expected. See Figure 26 for an example of the same imperfect split seen in Figure 24 (i.e., a split at the median as opposed to being closer to a 2:3 split which would be ideal) where the introduction of pseudo-randomness cannot perfectly compensate for the inaccuracy of the split with respect to class identification but does so to a large degree.

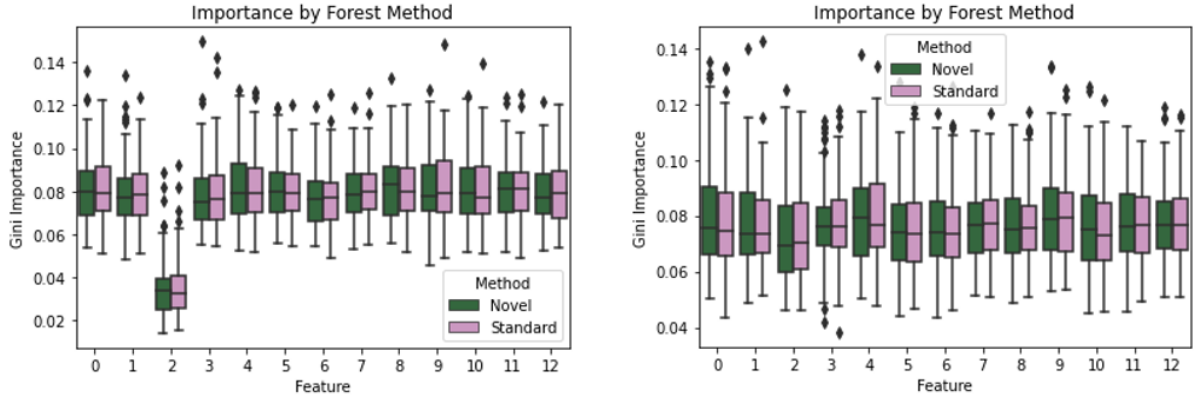


Figure 26. Feature importance where all features are linear combinations of one another and are continuous except for feature 2 (which is bilevel and has been forced to split with a non-ideal threshold at the 50th centile versus perfection at the 60th centile) with 50% randomization of the feature values; (a: left) No pseudo-randomness has been introduced; (b: right) pseudo-randomness has been introduced.

It is important to realize this loss of importance when a continuous value is transitioned to a discrete value using an inappropriate threshold is a real one (i.e., the ability for the appropriate split to be determined has been compromised) and is exacerbated by cardinality but only in so far as greater cardinality provides more opportunities for the an ideal division to exist or a threshold or thresholds that do exist may fall closer to ideal values when levels have maximal integrity (i.e., the levels can be perfectly identified). In real terms however, any given bilevel variable does not necessarily represent at its root a linear and / or continuous variable that has been mapped into two pieces and thus non-ideal splits in general reflect the true inability to partition samples into appropriate classes of interest. If a more precise feature existed (i.e., one with

higher cardinality) it would be more useful only if that precision facilitated more accurate insights into partitioning and would be expected to be more important in that setting. Recall Figure 19 where precision / the number of levels increases from left to right, but the partitioning accuracy (i.e., the accuracy of at least one division that separates on class from another of interest) is unchanged. Where no ideal split can be made, pseudo-randomness is only so helpful in correcting the issue of cardinality alone and the helpfulness is reduced the further off the threshold is (Figure 27).

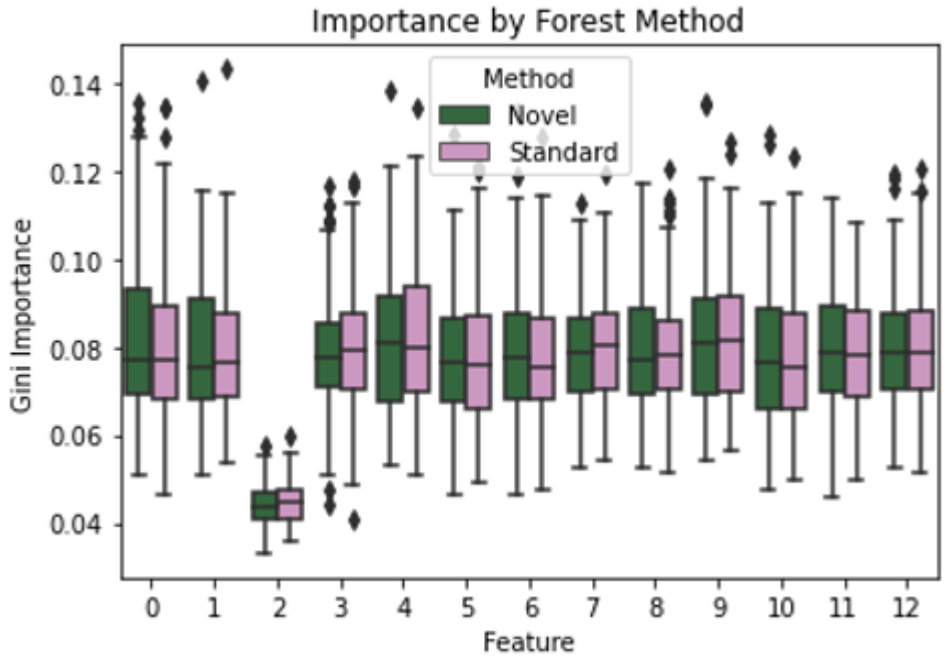


Figure 27. Feature importance where all features are linear combinations of one another and are continuous except for feature 2 (which is bilevel and has been forced to split with a non-ideal threshold at the 10th centile versus perfection at the 60th centile) with 50% randomization of the feature values and pseudo-randomness has been introduced.

To summarize, adding the random variation to the bilevel variable (i.e., pseudo-randomness) did nothing to improve real data quality, as was the case in the bottom row of Figure 19, but it provided the algorithm the opportunity to single out those errors in classification and eliminate them with the same overfitted potential as in the situation where a continuous variable is used (recall Figure 9). Pseudo-randomness does not improve accuracy and thus this artificial augmentation of precision does not alter a variable's inherent ability to predict classification.

C. Simulation Results (Composite Missingness)

1. Data Summary

The simulated data was created to emulate some of the properties of the wine dataset prior to exploring the properties of the various missing data masks before they were applied to the wine dataset. Features with equal predictive potential for the target class were selected to simplify the discussion of the feature importance results (features with equal correlation to the target class) The number of available features was selected to be 13 (the same number as in the wine dataset) and the number of cases 178. Of these cases half were of a target class. For all methods except the novel random forest approach, mean value imputation was used to complete the datasets.

2. Results

Bringing all three masks together (i.e., B, I, V) the following curves are obtained (See Figure 28):

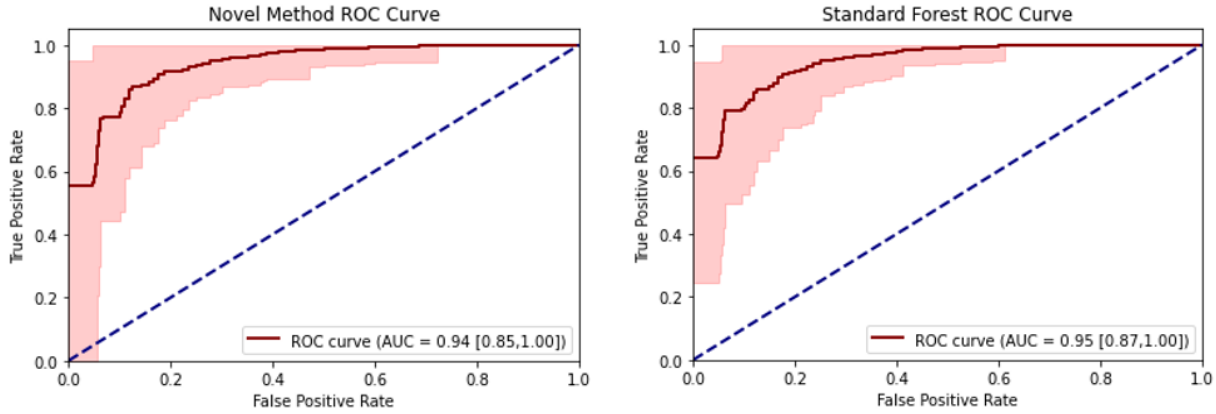


Figure 28. Receiver operating characteristic curves for a) the novel technique (left); b) the manually programmed standard random forest technique (right) using a linear system simulation after complex data augmentation.

Importance of features has greater variability with the novel approach but is more accurate compared with the standard approach as seen in Figure 53. Note again that features 0, 4, 5 are less randomized (25%) than features 2, 6, and 12 (75% randomized); all other features are 50% randomized (See Figure 29).

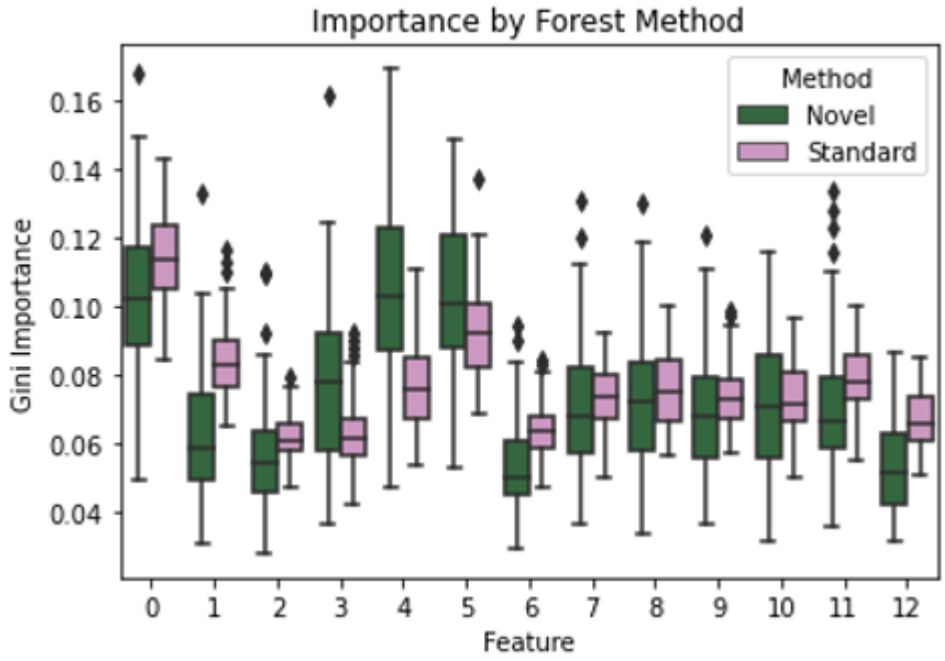


Figure 29. Feature / Factor Gini importance using a linear system simulation after complex data augmentation.

For the sake of comparison, consider where the number of cases has been increased to 500 from 178. The ROC curves are comparable, but estimates are more precise (see Figure 30).

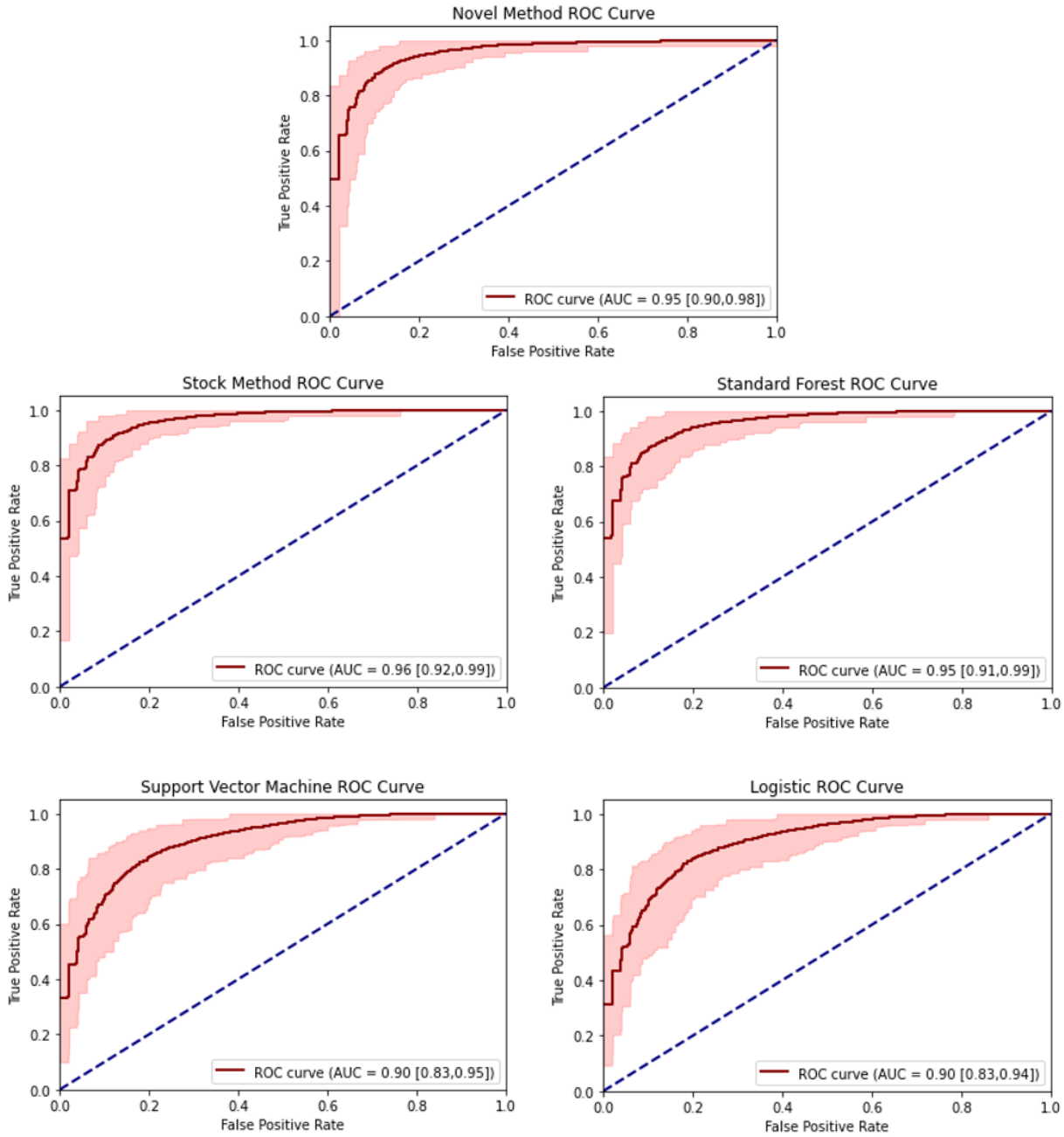


Figure 30. Receiver operating characteristic curves for a) the novel technique (top); b) the stock scikit standard random forest technique (mid-left); c) the manually programmed standard random forest technique (mid-right); d) the support vector machine technique (bottom left); e) the logistic regression technique using a linear system simulation after complex data augmentation with increased case counts (n = 500).

The results of the importance estimates are likewise similar (See Figure 31).

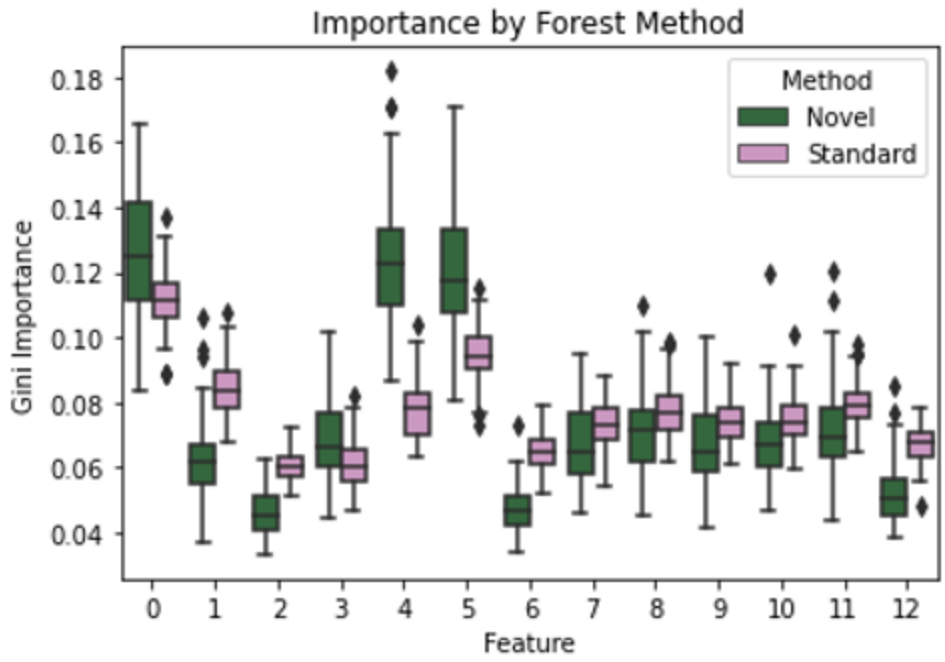


Figure 31. Feature / Factor Gini importance using a linear system simulation after complex data augmentation with increased case counts (n = 500).

Computation of permutation importance suggested that all features had no perceived importance.

D. Discussion

1. Overview

The simulated dataset incorporated several methods to generate overlapping forms of missing data. To thoroughly analyze the impacts of the different forms of missingness, particularly with respect to the appreciation of feature importance in the standard and variant random forest, the following section first decomposed the analysis to focus on various missingness masks.

In the first subsection (VI. D. 2. a.) the impacts of randomness and random missingness were reviewed with a simulation (Note that a simulation with evenly divided randomness and no missingness alone provided effectively perfect performance up to 50% randomness and thus this initial step was not discussed in detail). In the latter subsections, adjustments with additional missingness masks applied were explored. Note that in each section randomness was adjusted to a set of specified variables to view the impacts of the missingness masks on the assessment of variable importance understanding the addition of various singular and combinations of missingness mask. At times, secondary analysis was performed to demonstrate features of the novel random forest technique to facilitate interpretation of the ultimate simulation. Of note, the final / ultimate simulation presented was a combination of all three masks with the addition of increased randomness to certain target variables to ascertain the composite outcome of missingness on the result of the simulation and to explore the effects on the importance assessment.

2. Impacts of Missing Values by Decomposition of Missingness Masks

a. Base Case with Evenly Distributed Randomness and Missingness

With the simulated linear system data, where missingness is random, one can see that there was little difference in the curves produced using the scikit stock method and the manually programmed traditional random forest method versus the novel approach (see Figure 32). Note in these simulations the randomness was maintained at 50% for all variables (note randomness was incremented from 0 to 25 to 50% to obtain curves that satisfied our simulation criteria that AUC for the standard random forest approach be 95% or less) and the missingness was set to a high level of missingness as previously discussed.

$$B = (1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1)$$

The results of 50% randomness and 50% missingness are shown.

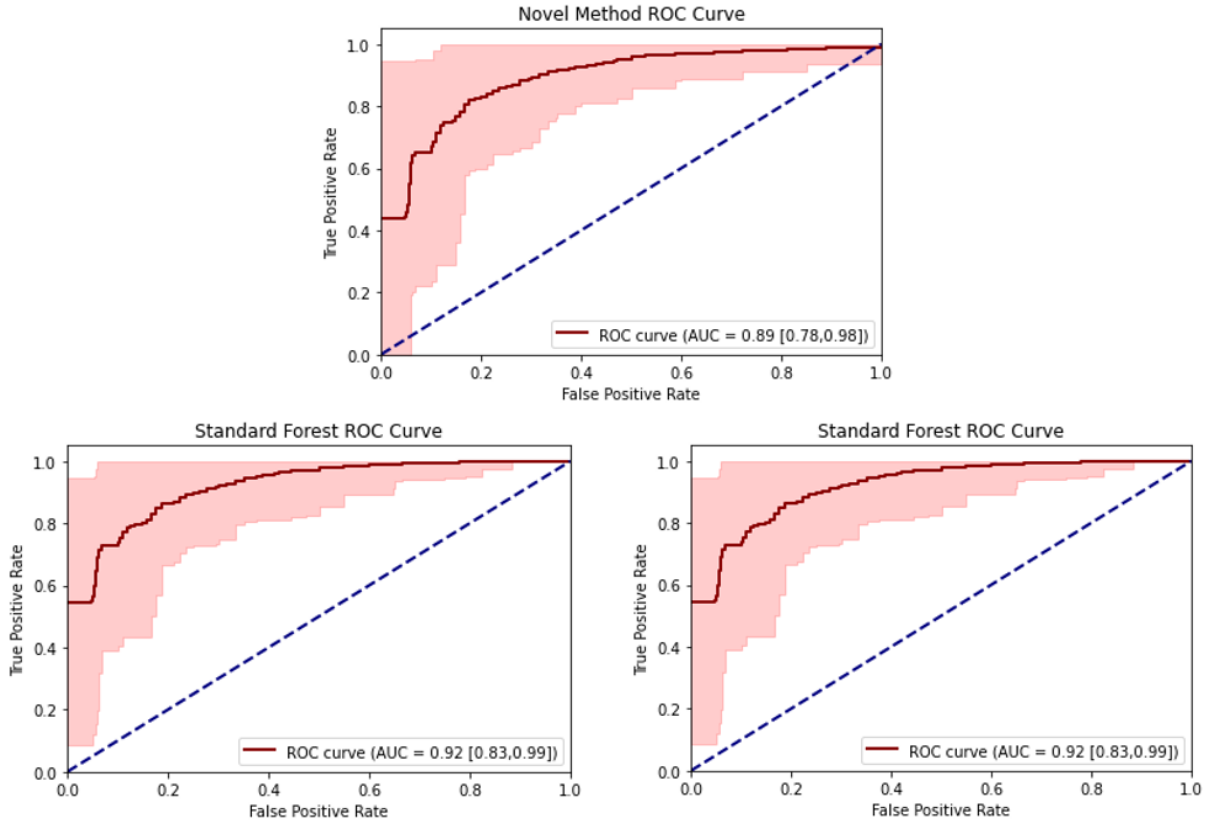


Figure 32. Receiver operating characteristic curves for a) the novel technique (top); b) the stock scikit standard random forest technique; c) the manually programmed standard random forest technique in a linear system simulation.

From a technical perspective, the standard curves used the same process with matched hyperparameters obtaining effectively identical results. The novel technique is also, as previously discussed, a special case of a standard random forest (i.e., it is a scenario where only the cases with all data present for the selected features in during generation of a given trees are used). While there is a slight advantage to the standard approach in terms of prediction accuracy, the Improvement reflects leveraging what can be gleaned from the residual data in incomplete cases without significant disruption from reduced feature thresholding accuracy. There was

however some trade off in interpreting the relevance of the findings as reflected upon later. Overall, a minimum AUC of 0.78 would seem reassuring of a preserved ability to make reasonable inference from data that was significantly degraded, however examination of conventional techniques highlights the success relates to some increased power of the random forest approach itself. Support vector machine and logistic regression-based curves indicated that these techniques have less predictive capacity under the simulation conditions (i.e., where 50% of the data is missing and 50% of the remaining data is randomized) as seen in Figure 33.

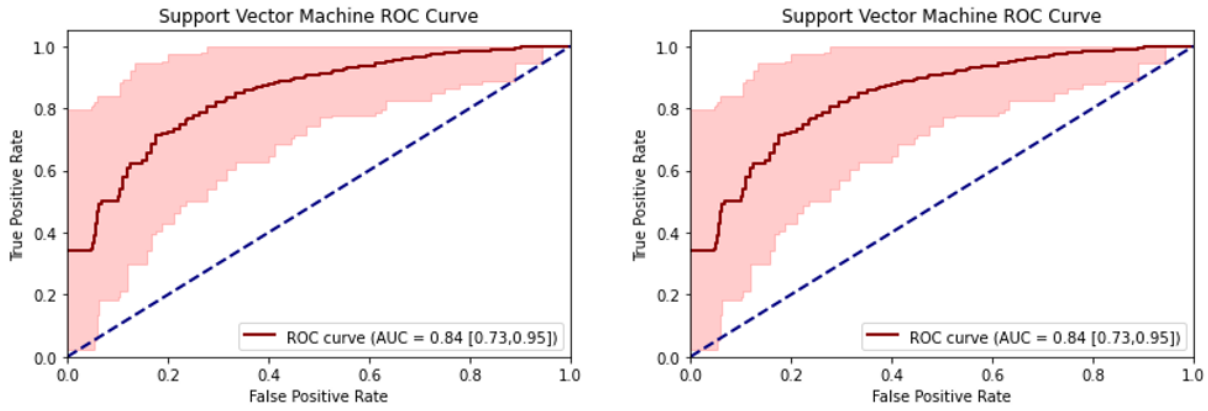


Figure 33. Receiver operating characteristic curves for a) the support vector machine technique (left) and b) the logistic regression technique (right) in a linear system simulation.

Moving from an assessment of accuracy of the predictions to the assessment of relative feature importance, in this linear system it was unsurprising that all features were roughly of the same importance as seen in Figure 34. As an equation in a linear system can be re-expressed as a non-nullified linear combination of equations within that system, such a system of equations could be reduced to a one to one correspondence between any one factor and the result (i.e., each factor can be related to the resultant classification as follows: $a_i x_i = z_j$ where a_i is the cofactor for the i^{th} feature, x_i is the value, and z_j is probability to select an ultimate classification).

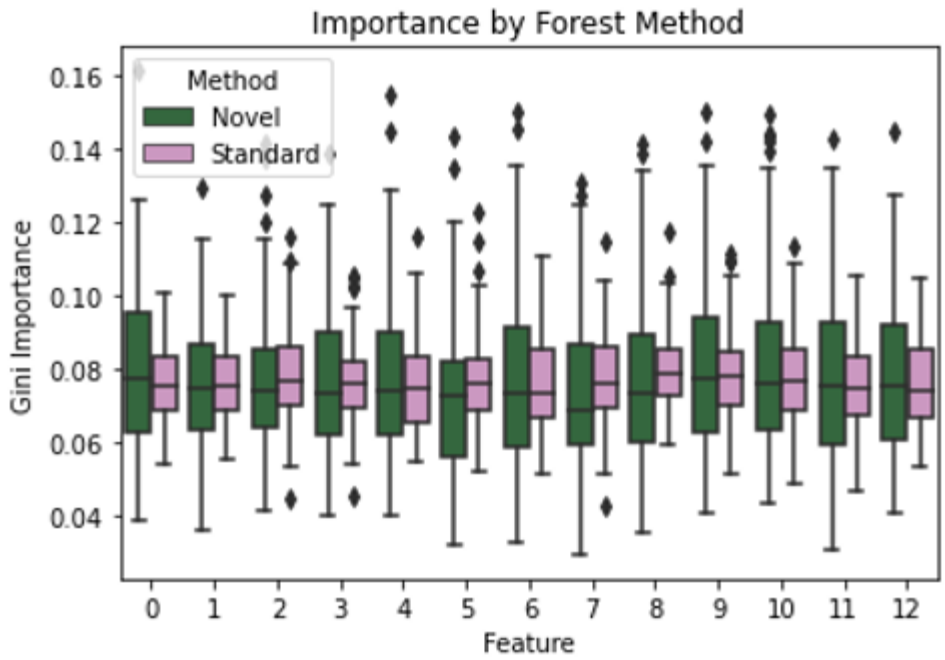


Figure 34. Feature / Factor Gini importance in a linear system simulation with even probabilities of missingness

b. Focused Randomization with Evenly Spread Missingness

All things being equal, given several features with equal correlation to the classification, those features with a greater degree of randomness would be expected to be less predictive. This concept was explored briefly in this section to demonstrate the degree to which Gini importance does respond / change when a feature is less predictive / correlated to the ultimate classification. Preserving the 50% missingness across variables If one were to focus a greater degree of randomization (75% randomization) in features 3 and 6 and less (25% randomization) in 4 and 5 one can see relative loss and gain of importance respectively (See Figure 35).

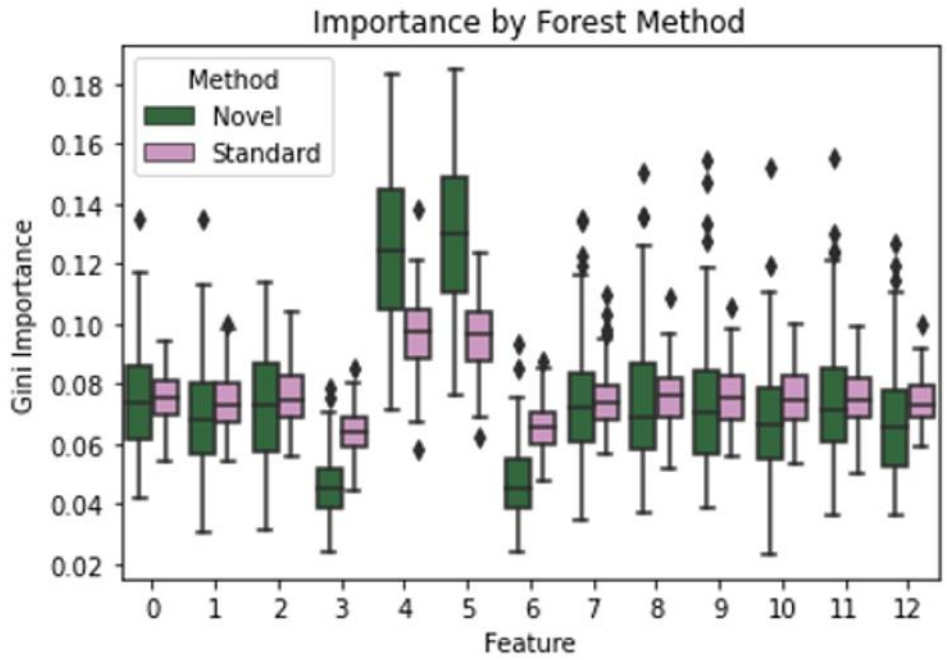


Figure 35. Feature / Factor Gini importance in a linear system simulation with increased randomness in features 4 and 5 but less in features 3 and 6.

c. Focused Missingness with Even Randomness

Now consider a scenario where each feature has 50% of its values randomized but missingness is not evenly distributed and instead focused on three factors, namely 0, 1, and 12, which are 25% more likely to be missing. The vector representing relative missingness is therefore:

$$B = (1.25 \quad 1.25 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1.25)$$

Three features were selected to have higher proportions of missingness to demonstrate a reliable link between the increase proportion of missingness and change in feature importance. The characteristic curves resemble previous results (see Figure 36):

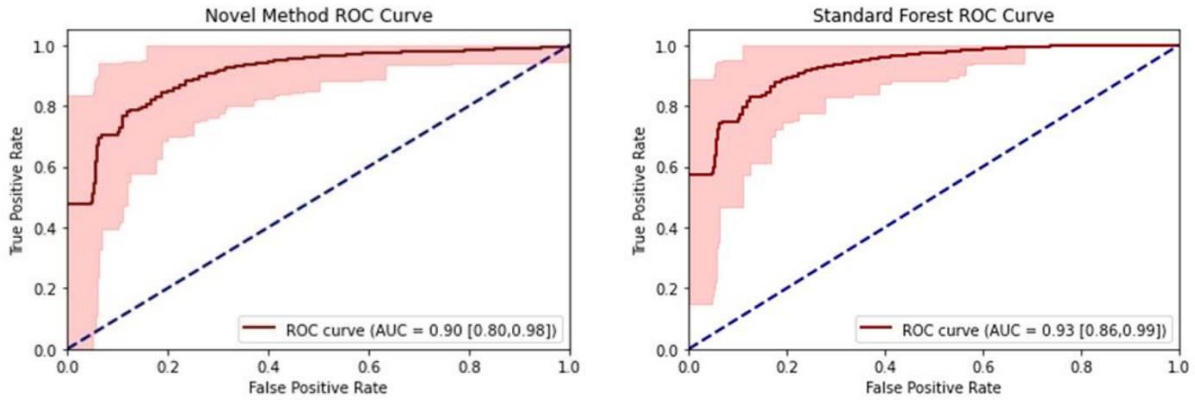


Figure 36. Receiver operating characteristic curves for a) the novel technique and b) the standard random forest technique in a linear system simulation given a 25% greater quantity of missing data for factors 1, 2, and 12.

The Gini importance assessments are different between techniques. Where one can see there is an indication that those factors with 28% more missing data are of approximately one quarter the importance (See Figure 37) in the standard approach while assessments of the novel approach are unchanged.

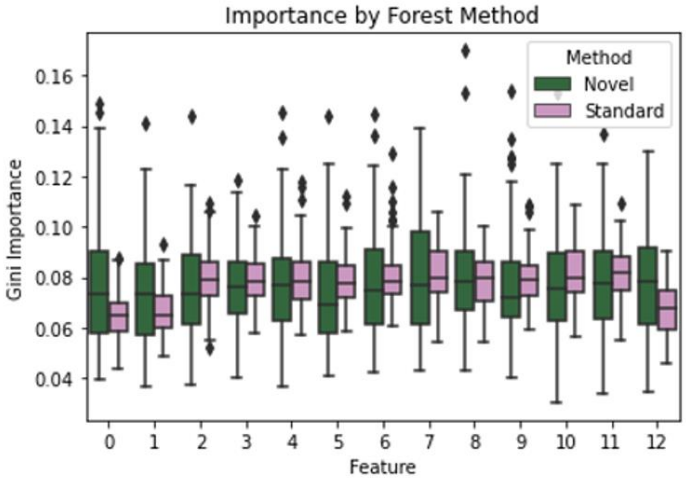


Figure 37. Feature / Factor Gini importance in a linear system simulation given a 25% greater quantity of missing data for factors 1, 2, and 12 without missingness correction.

When discussing missingness in the introduction, a comment was made regarding potential for feature importance adjustment by weighting overall Gini importance computations with the number of available cases per tree. If one corrects for the degree of missingness such that, instead of summing the importance estimations across trees and dividing by the number of trees, one were to use the number of cases that were submitted for classification by the individual tree as a weighting for the tree (i.e., the number of cases with complete data for the feature set selected for a given tree), it is clear that a large part of the change in importance is related to missingness itself. In Figure 38, note the weighting procedure significantly reduces the gap between the novel and standard techniques. Conversely, the inverse values of the weight could otherwise be used to restore / adapt the standard value to recognize evenly distributed importance of values.

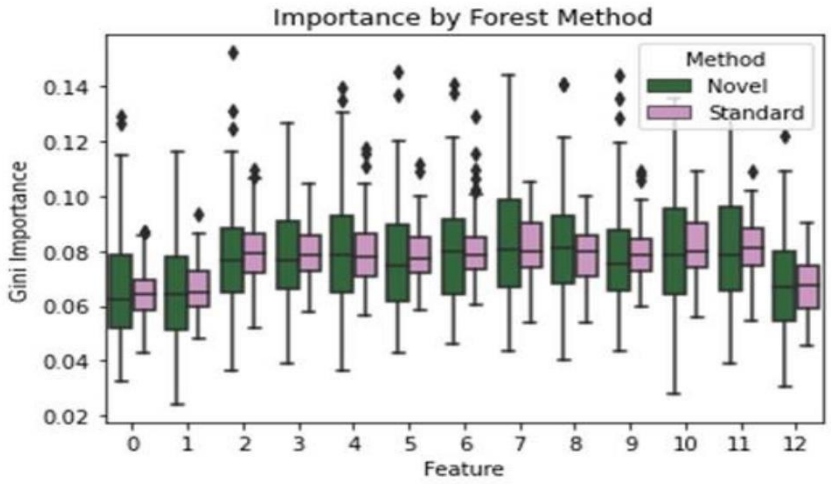


Figure 38. Feature / Factor Gini importance in a linear system simulation given a 25% greater quantity of missing data for factors 1, 2, and 12 after missingness correction.

d. Missingness of a feature value that is dependent on some feature value's missingness

While Gini importance assessment is altered by missingness at random (overshadowing the known inherent potential for all factors to predict the outcome equally), missingness where there is a dependence on the missingness of other feature values can create complex changes in the missingness assessment which was explored in this subsection. At an extreme, diametric relationships in missing variable data (i.e., if one value is missing another value is almost always present and vice-a-versa) can obstruct the creation of trees and alter the Gini importance assessment by virtue of a reduced relative quantity of trees possessing a variable. The reason for this is that trees that were randomly selected to use both of diametrically missing variables are dismissed as if one variable is present the other must be missing (i.e., no complete data entries exist). Consider the following data:

$$\left(\begin{array}{cccc|c} 0.0 & - & 4 & 1 & C_L \\ 1.0 & - & 3 & 2 & C_H \\ - & 1.5 & 8 & 3 & C_L \\ 0.5 & - & 1 & 4 & C_H \\ - & 1 & 9 & 5 & C_L \\ - & 0 & 3 & 6 & C_H \\ 0.0 & - & 3 & 7 & C_L \\ 0.5 & - & 8 & 8 & C_H \\ 0.0 & - & 7 & 9 & C_L \end{array}\right)$$

In this example, clearly trees could not be formed from complete data if the first and second features are to be incorporated simultaneously however there is a threshold for each that if one or the other is triggered, the data could be perfectly separated (0.5 and above for feature 1 and less than 1 for feature 2). This is not to say these are the only features of importance but if one considers that there are 10 possible selections of 1 – 2 features with 4 permutations including any one feature (4 choose 2 + 4 = 10: 1 alone or 1 and 2, 3 or 4; 2 alone or 2 and 3 and 4; 3 alone or 3 and 4; 4 alone), the structure of missingness would impose a 25% penalty on importance on average (trees that would attempt to used feature 1 an 2 would not exist). Of course, a similar correction factor can once more be introduced to account for this as was the case for missingness

at random in the traditional random forest technique. The following results are based on the baseline missingness vector B and missingness association matrix M:

$$B = (5 \quad 5 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 5 \quad 5)$$

$$I = \begin{bmatrix} 0 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -20 & -20 \\ 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -20 & -20 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -20 & -20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 30 \\ -20 & -20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 30 & 0 \end{bmatrix}$$

Note that this missingness matrix strongly links data absences between factors 0 and 1 creating one cluster of missingness and 11 and 12 incorporated as another cluster with the two clusters strongly desynchronized with respect to one another. Appreciate in this case, due to the exponential nature of this mask, an incremental approach to adjusting this mask was not taken and instead a large value was selected to effectively ensure the coordinated missingness of features 0 and 1 that was diametrically missing to the coordinated missingness of features 11 and 12. The resultant accuracy of the simulation remains comparable to prior simulations (see Figure 39):

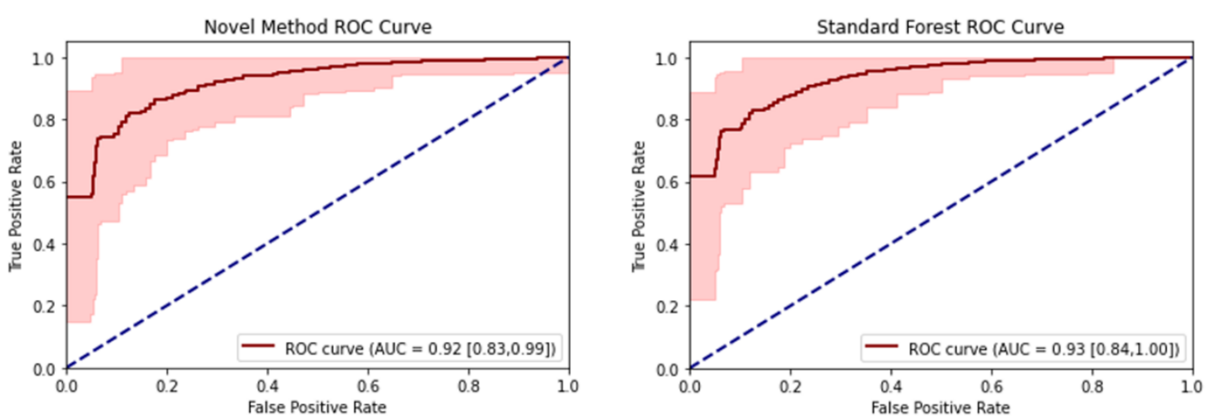


Figure 39. Receiver operating characteristic curves for a) the novel technique and b) the standard random forest technique in a linear system simulation given increased counter relatedness of missingness of features 0 and 2 with 11 and 12.

Without correction, the novel algorithm appears to detect reduced importance of features 1, 2, 11, and 12 where there is an increased counter relatedness of missingness (See Figure 40).

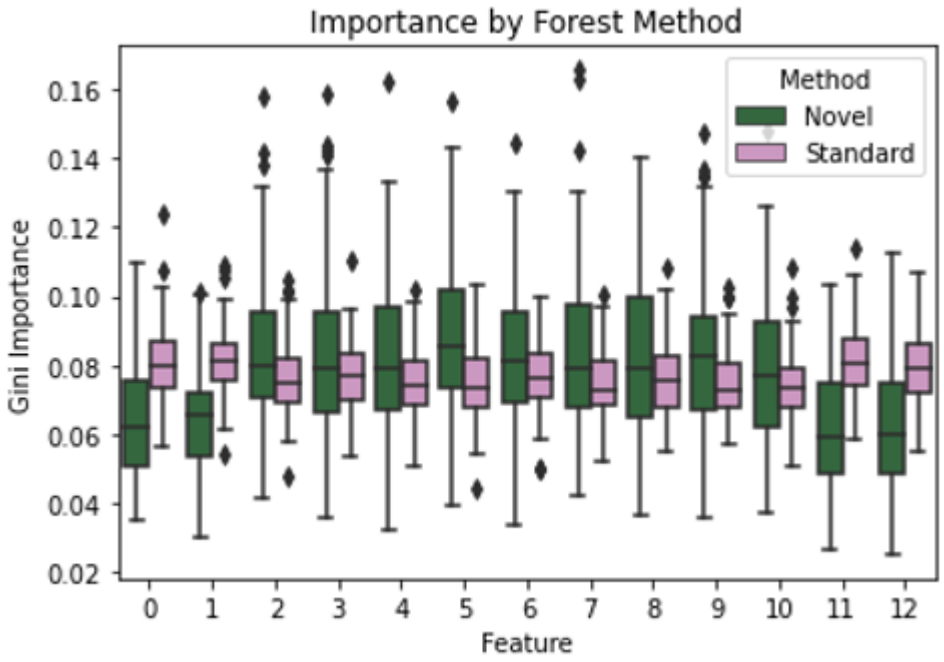


Figure 40. Feature / Factor Gini importance in a linear system simulation given increased counter relatedness of missingness of features 1 and 2 with 11 and 12.

The matrix monitoring lock-outs (i.e., a matrix that stores the number of times a feature was part of a random forest selection but the chosen variable combination could not be used to generated a tree due to a lack of complete cases) and the matrix showing the total number of each factor associated with the various tress within the ultimately generated random forest are presented in section (XI. A. 1). The result of correction for the mismatch in the number of trees is pictured in Figure 41.

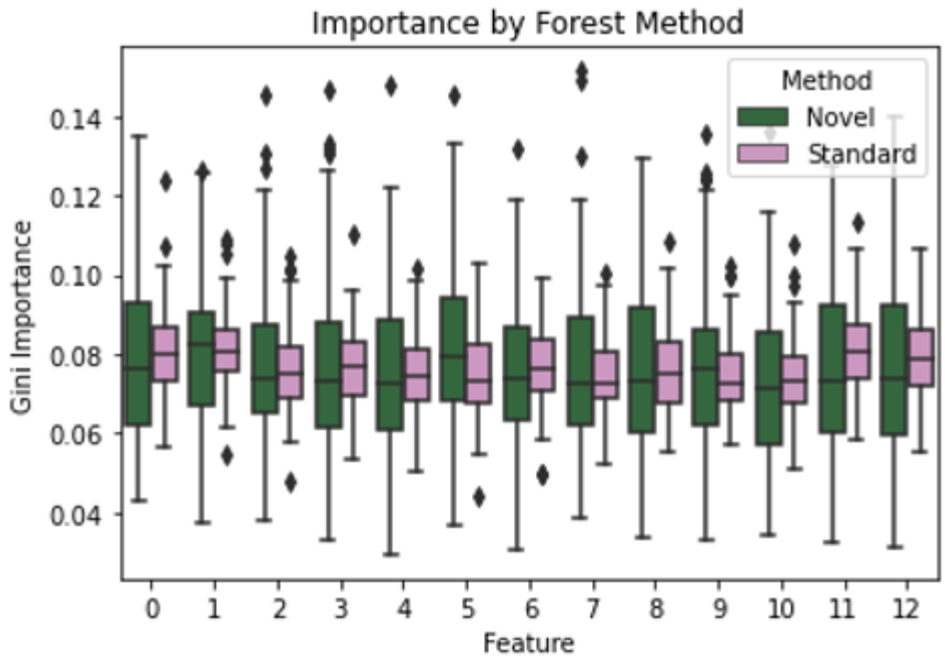


Figure 41. Feature / Factor Gini importance in a linear system simulation given increased counter relatedness of missingness of features 1 and 2 with 11 and 12 following correction.

One can see that the standard approach estimates the even distribution of importance that was anticipated as overall there are still a relatively even number of missing values per factor and this was not impacted by how the missing values themselves are arranged. In this case it would seem that the standard approach would be superior however, overall, one may still need to correct for missingness focused in certain features relative to others and the situation where there is a total or near total absence of data where another feature is present which although this is likely uncommon and easily anticipated.

e. Missingness with Value Dependence

A significant challenge occurs where the missingness is dependent on the value of the feature itself or other features (including the resultant true classification which in a linear system is fully attributable to other available features) as previously explored. Several issues are discussed in subsequent sections including the transfer of information / importance to variables with missing data dependent on another variable and transfer of information / importance from the independent variable from the standard random forest and novel random forest perspectives. The value of the two corrections performed for Gini importance (i.e., lock-out correction and proportioning* class-skewing) are described as well.

v. Own Value Related Missingness (and lock out correction)

Consider a data set with 50% randomness and 50% sparsity with the following missing value matrix applied where missing values are own value concentrated:

1.5	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

$$V = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

The related ROC plots are pictured in Figure 42.

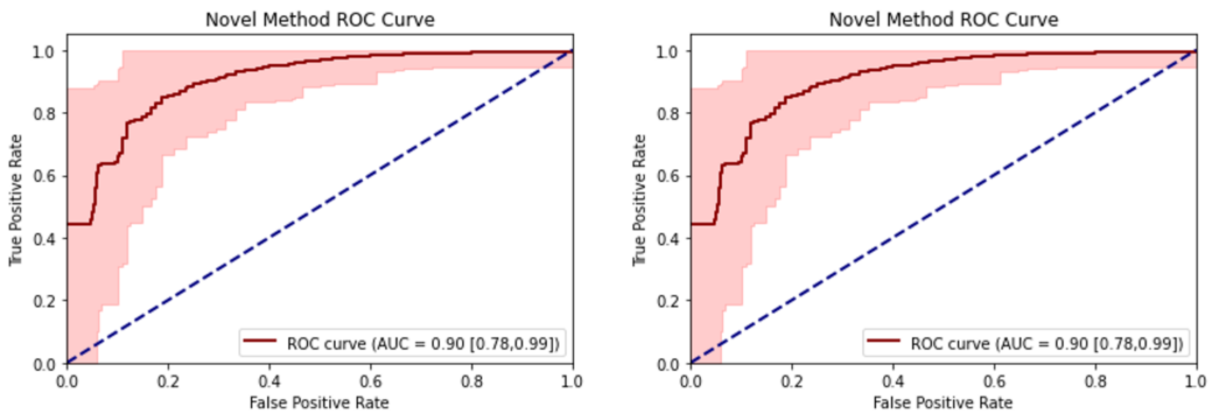


Figure 42. Receiver operating characteristic curves for a) the novel technique and b) the standard random forest technique in a linear system simulation given increased probability of missingness with value.

One might expect that if the missingness is value dependent, so long as sufficient data points surrounding relevant thresholds reflecting different classifications remain, that there would be no profound impact on an assessment of value importance itself.

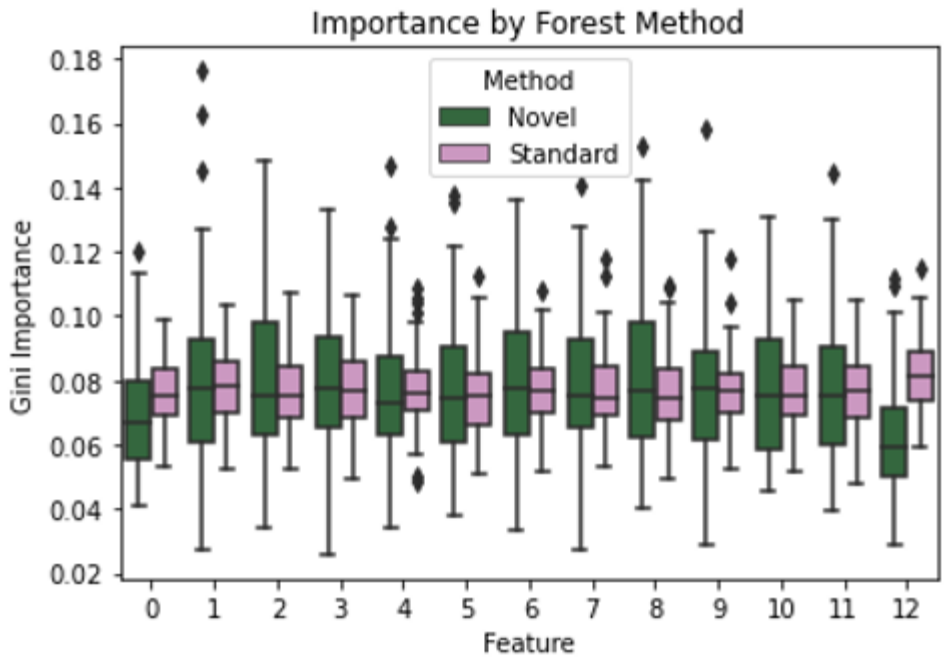


Figure 43. Feature / Factor Gini importance in a linear system simulation given increased probability of missingness associated with the values of a factor itself.

Prior to lock-out correction in the novel random forest method, the standard approach appeared to better reflect that no variable is at an advantage or disadvantage so long as the randomness of the variables does not vary between groups. The novel approach suggested that those values with a greater association with their own value perform more poorly. Note that there was a need to account for the disallowance of tree formation where too few members of one class or another were available (See Figure 44 for the corrected importance plot). This was akin to and managed through the same mechanism as the lock-out previously described where there are no feature-wise complete cases available to train a tree. This highlights a complexity of how data is presented for the creation of individual trees within a random forest especially as this correction alone may be insufficient to fully explain the importance change.

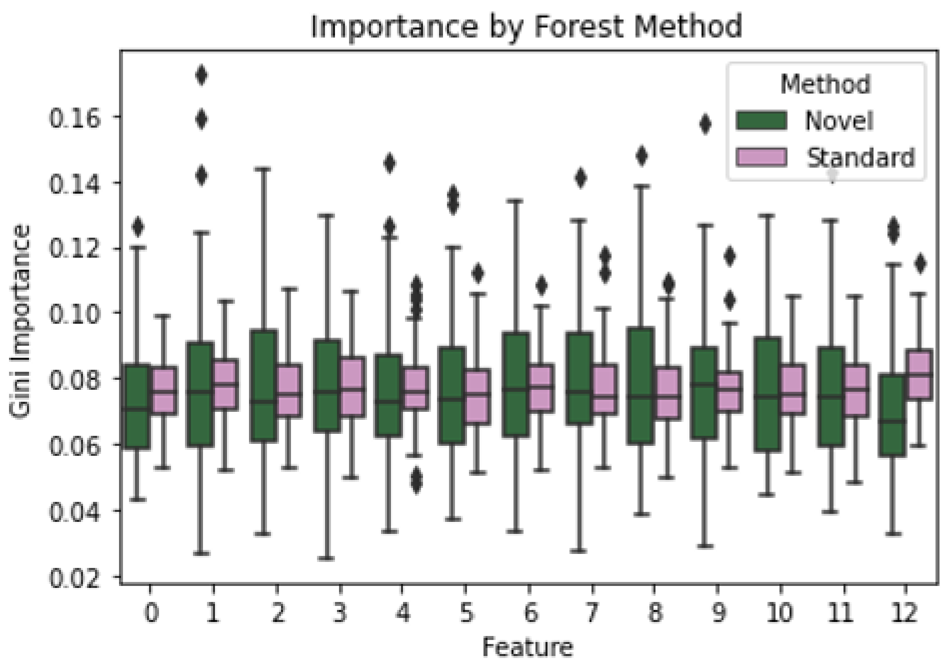


Figure 44. Feature / Factor Gini importance in a linear system simulation given increased probability of missingness associated with the values of a factor itself after lock-out correction.

vi. Own Value Related Missingness (and proportioning)

To now emphasize the impacts of proportioning on feature importance estimates, we exaggerated the own value related missingness. Consider Figure 45 where randomness / noise is removed before and after addressing proportioning (i.e., the large differences in cases of interest to non-cases presented for the fitting of tree within a random forest). Note the simulations still relied on 50% sparse data and the following value related missingness mask:

$$V = \begin{bmatrix} 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

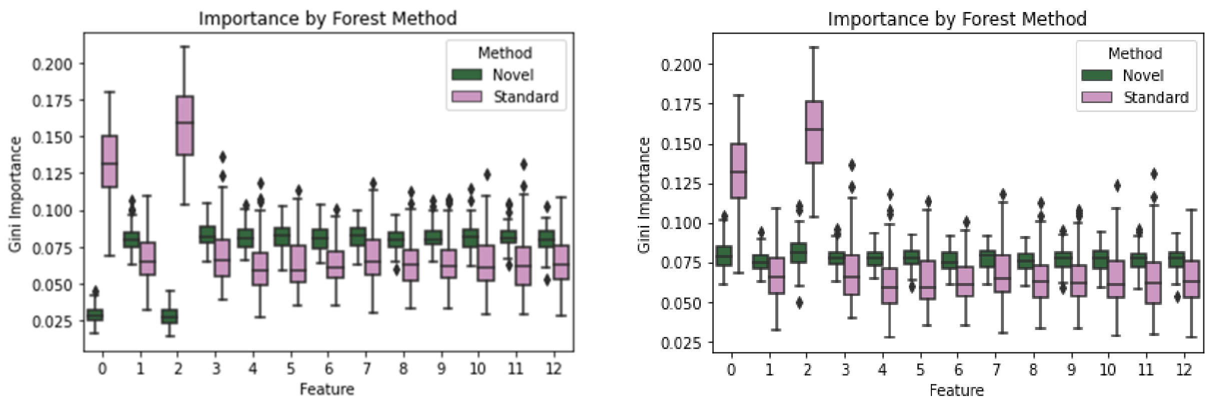


Figure 45. Feature / Factor Gini importance in a linear system simulation with no randomness / noise given increased probability of missingness associated with the values of a factor itself for factors 0 and 2 without (left) and with (right) lock-out correction correction.

As all values have the same / similar predictive potential and the result was very similar when variables are dependent on their own value (Figure 45) or adjacent variables (See Figure 46). Note that that missingness matrix that emphasizes the importance of adjacent values was as follows:

$$V = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

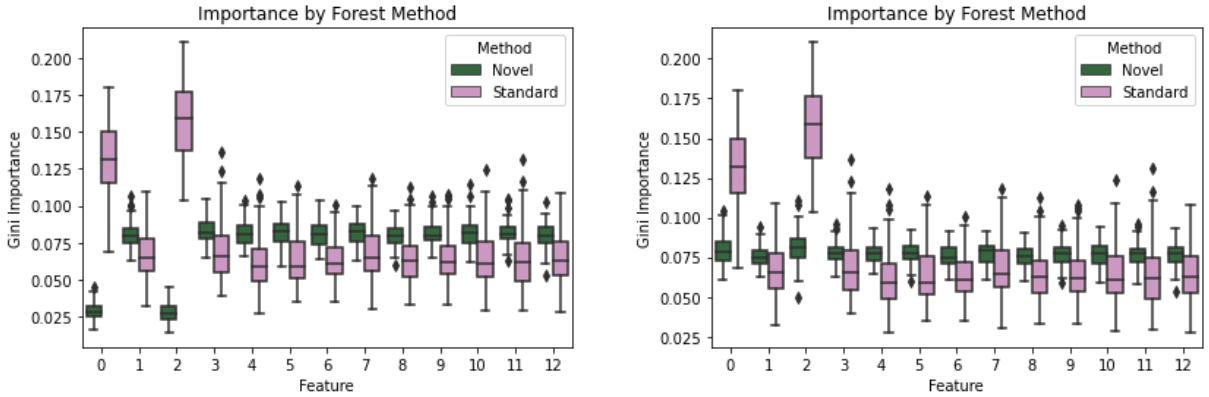


Figure 46. Feature / Factor Gini importance in a linear system simulation with no randomness / noise given increased probability of missingness associated with the values of the adjacent factors for factors 0 and 2 without (left) and with (right) proportion correction.

The effectiveness of the correction was less striking where variable randomness / noise has been introduced (50% randomness across every feature), however, a relevant effect remains (see Figure 47).

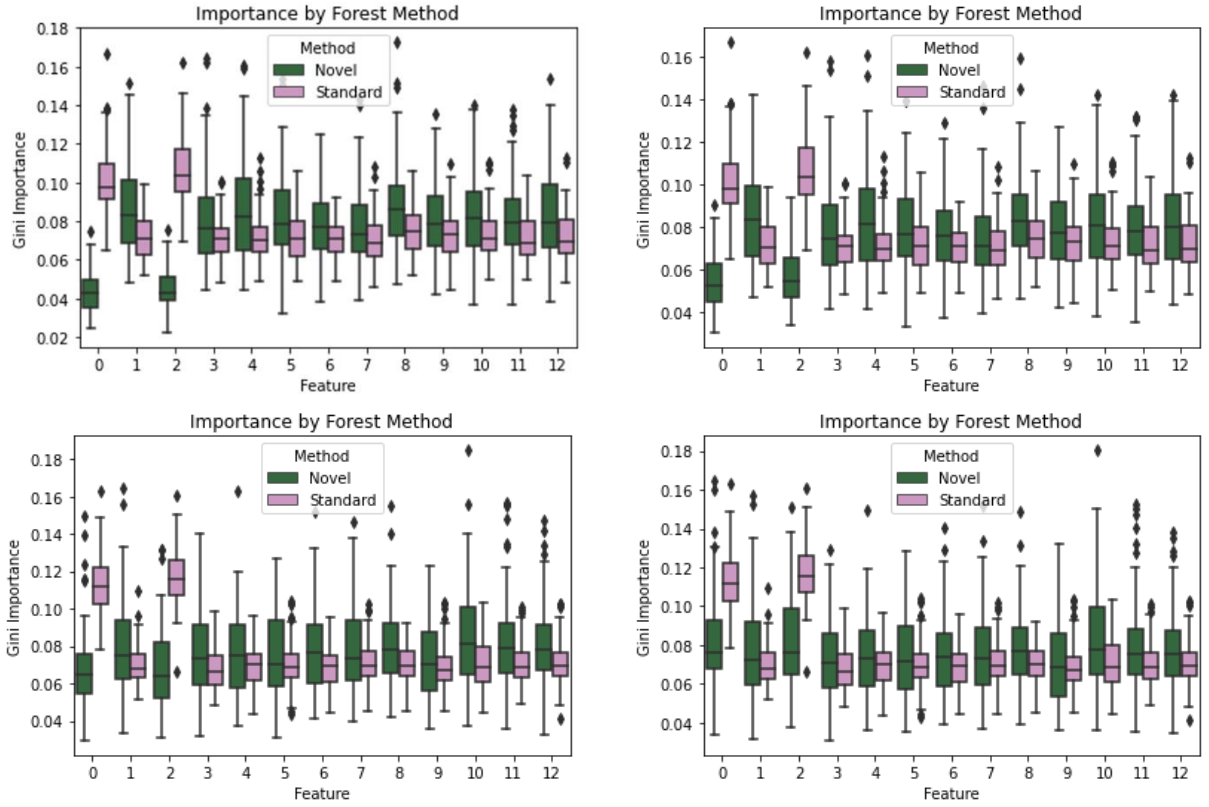


Figure 47. Feature / Factor Gini importance in a linear system simulation with 50% randomness / noise given increased probability of missingness associated with the values of the factors 0 and 2 themselves (top) or their adjacent factors (bottom) (i.e., factor 1 for 0 and 3 for 2) without (left) and with (right) proportion correction.

Returning to Figure 47, note the difference between the missingness linked to the feature's own value versus the adjacent feature's value. This difference was related to the phenomenon

previously discussed where there is a legitimate loss of importance with own value associated missingness. Consider first a scenario where a feature's own high values are missing:

$$\left(\begin{array}{cccc|c} 1 & 1 & 1 & 1 & C_L \\ 2 & 2 & 2 & 2 & C_L \\ 3 & 3 & 3 & 3 & C_L \\ 4 & 4 & 4 & 4 & C_L \\ 5 & 5 & 5 & 5 & C_L \\ 6 & 6 & 6 & 6 & C_H \\ 7 & 7 & 7 & 7 & C_H \\ 8 & 8 & 8 & 8 & C_H \\ 9 & 9 & 9 & 9 & C_H \end{array}\right) \xrightarrow{Randomize\ 50\%} \left(\begin{array}{cccc|c} 1 & 2 & 7 & 1 & C_L \\ 2 & 8 & 2 & 8 & C_L \\ 4 & 3 & 9 & 3 & C_L \\ 7 & 4 & 4 & 2 & C_L \\ 5 & 1 & 3 & 5 & C_L \\ 3 & 6 & 6 & 4 & C_H \\ 6 & 7 & 5 & 7 & C_H \\ 8 & 9 & 8 & 6 & C_H \\ 9 & 5 & 1 & 9 & C_H \end{array}\right) \xrightarrow{Own\ Value} \left(\begin{array}{cccc|c} 1 & 2 & 7 & 1 & C_L \\ 2 & 8 & 2 & - & C_L \\ 4 & 3 & 9 & 3 & C_L \\ 7 & 4 & 4 & 2 & C_L \\ 5 & 1 & 3 & 5 & C_L \\ 3 & 6 & 6 & 4 & C_H \\ 6 & 7 & 5 & - & C_H \\ 8 & 9 & 8 & - & C_H \\ 9 & 5 & 1 & - & C_H \end{array}\right)$$

In all features, high levels are associated with the C_H classification. If there is a sufficient amount of missing data and a strong enough association with the value of the factor then one can see that truly high values in feature 4 are missing and those other cases that represent a C_H classification have noise displacing their value sufficiently (i.e., these were randomized with falsely low values) so that they don't reflect the original relationship between the feature 4 value and the classification. This would have an expected negative impact on the importance of the variable. Now consider when an adjacent feature's high values are missing:

$$\left(\begin{array}{cccc|c} 1 & 2 & 7 & 1 & C_L \\ 2 & 8 & 2 & 8 & C_L \\ 4 & 3 & 9 & 3 & C_L \\ 7 & 4 & 4 & 2 & C_L \\ 5 & 1 & 3 & 5 & C_L \\ 3 & 6 & 6 & 4 & C_H \\ 6 & 7 & 5 & 7 & C_H \\ 8 & 9 & 8 & 6 & C_H \\ 9 & 5 & 1 & 9 & C_H \end{array}\right) \xrightarrow{Adjacent\ Value} \left(\begin{array}{cccc|c} 1 & 2 & 7 & - & C_L \\ 2 & 8 & 2 & 8 & C_L \\ 4 & 3 & 9 & - & C_L \\ 7 & 4 & 4 & 2 & C_L \\ 5 & 1 & 3 & 5 & C_L \\ 3 & 6 & 6 & - & C_H \\ 6 & 7 & 5 & 7 & C_H \\ 8 & 9 & 8 & - & C_H \\ 9 & 5 & 1 & 9 & C_H \end{array}\right)$$

In this case a larger amount of unrandomized / noiseless data was retained by the 4th variable. If the value related missingness coefficients were strong enough to solely create feature loss at the highest feature values of the adjacent features and 50% of those high values are random then the other 50% are falsely seen as low values and thus allow the 4th feature to retain those values which in turn have a 50/50 chance of being accurate. In short there is a clear advantage than in the previous scenario where all remaining feature 4 values representing C_H classification were either missing or random.

vii. Importance Stealing (Transfer to Dependent Feature)

Where the missingness of a value is based on another variable that is useful for classification, it would be a reasonable expectation that no information will be added so long as both variables are retained within the model. There is however a slight tendency to elevated importance for variables with missingness that is highly dependent on / correlated to the values of other variables that are predictive of classification using a standard random forest technique. Consider

(with 50% randomness and 50% sparsity) application of the missingness mask that relates feature values where feature 12 is dependent on the value of feature 0:

$$V = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The results of feature importance calculations are as follows (See Figure 48, feature 12):

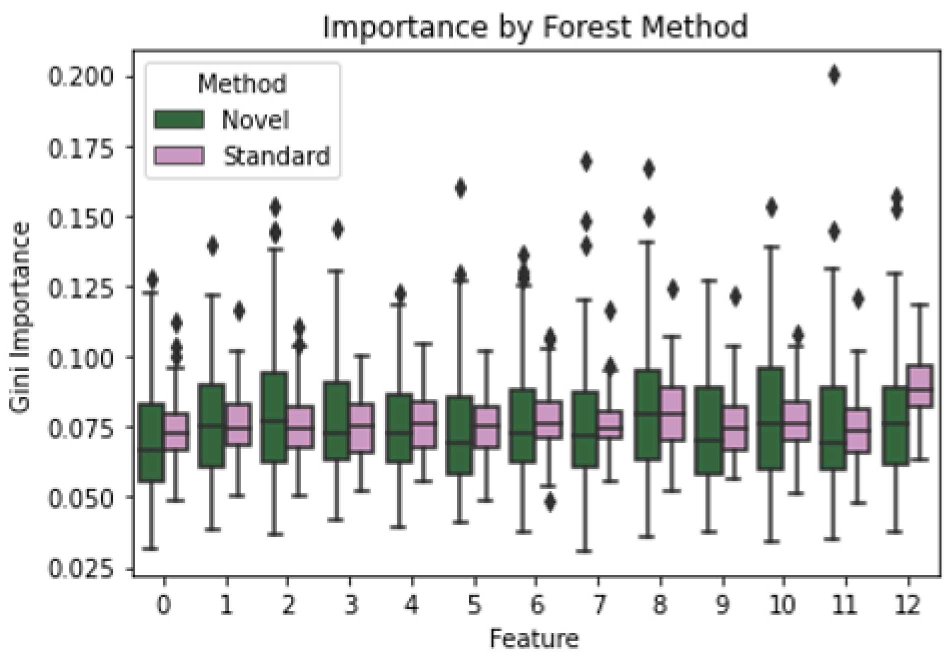


Figure 48. Feature / Factor Gini importance in a linear system simulation given increased probability of missingness associated with the values of factors other than the factor itself.

Although the difference is subtle, it likely reflects that part of the information contained by the feature has been augmented, in the case of the standard approach, by redistributing information from other features. Such a redistribution is clearer where the level of randomness is very different between two variables. Consider if there was a system of variables where one variable was perfectly predictive of outcome and another were not:

$$\left(\begin{array}{cccc|c} 1 & 8 & 4 & 1 & C_L \\ 2 & 2 & 3 & 2 & C_L \\ 3 & 3 & 8 & 3 & C_L \\ 5 & 4 & 1 & 4 & C_L \\ 5 & 6 & 9 & 5 & C_H \\ 8 & 6 & 3 & 6 & C_H \\ 7 & 3 & 3 & 7 & C_H \\ 8 & 8 & 8 & 8 & C_H \\ 9 & 6 & 7 & 9 & C_H \end{array}\right) \rightarrow \left(\begin{array}{cccc|c} 1 & 8 & - & 1 & C_L \\ 2 & 2 & - & 2 & C_L \\ 3 & 3 & - & 3 & C_L \\ 5 & 4 & - & 4 & C_L \\ 5 & 6 & 9 & 5 & C_H \\ 8 & 6 & 3 & 6 & C_H \\ 7 & 3 & 3 & 7 & C_H \\ 8 & 8 & 8 & 8 & C_H \\ 9 & 6 & 7 & 9 & C_H \end{array}\right) \xrightarrow{mean} \left(\begin{array}{cccc|c} 1 & 8 & 6 & 1 & C_L \\ 2 & 2 & 6 & 2 & C_L \\ 3 & 3 & 6 & 3 & C_L \\ 5 & 4 & 6 & 4 & C_L \\ 5 & 6 & 9 & 5 & C_H \\ 8 & 6 & 3 & 6 & C_H \\ 7 & 3 & 3 & 7 & C_H \\ 8 & 8 & 8 & 8 & C_H \\ 9 & 6 & 7 & 9 & C_H \end{array}\right)$$

Here it is clearly seen that in the original matrix, the two most predictive factors would be the first and last (i.e., fourth) features. The first feature would be able to give a near perfect separation at a value of 3 to less than 6 and the fourth a perfect separation at between 4 and 5. Now consider the second matrix, where the missingness is itself predicted by the value of the 4th feature. Although the third feature is random with respect to the outcome, and as a decision tree can call on a feature more than once, one would certainly see the importance rise for feature 3 (where missing values are replaced by feature mean values) in a tree that was tasked with utilizing either feature 2 or 3 (noting that the values in feature 2 are also random with respect to the ultimate classification). In this situation the optimal selections for a tree would be to first split off feature 3 values at 6 and below and then split off those with a value of 6 from those with lesser values. The contribution of splits related to the use of feature 3 would account for 100% of the importance attributed to splits of any available features for that particular tree (i.e., only features 2 and 3). If feature 3 was placed head to head with feature 4, feature 4 would dominate as only one split would be required to achieve complete classification although one could note that it would be a "coin flip" if instead of replacing missing values with feature means you provided an extreme value beyond the maximum or minimum (i.e., instead of replacing missing feature 6 values with 6 they were replaced with either 2 or 10 which are, respectively, less than and more than the apparent extreme values of 3 and 9). A similar situation is present if the tree is to use feature 1 or 3. In the case the mean value is used, the feature importance assessment favors splitting the group first by those cases with a feature 1 value of 5 or less and then place a division per feature 3 at between 6 and 9. If an extreme value replacement was used for feature 3 missing data on the other hand the feature would completely dominate that importance assessment for that particular tree. In the practical situation presented in Figure 46, the way in which data was shared / stolen is similar and the features were, by design, all of similar importance at baseline.

In summary, if the missingness of one variable (dependent feature) is dependent on another's value (independent feature) which in turn predicts a classification (e.g., higher ANA titers or presence of ANCA predicting a secondary source of vasculitis and making it less likely to obtain a biopsy) then the standard random forest-based computation of Gini importance leads to redistributing the importance to the dependent feature and the novel approach does not.

viii. Aside of Focused Missingness

To demonstrate the effects of focusing missingness (i.e., the transfer of importance from the independent to dependent feature) we revisit our simulations discussion proportioning (see Figure 49). All features were with 50% randomness with the exception of feature 1 which was 25% randomized and the value related missingness matrix was as follows:

$$V = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

One can see that the importance of feature 0 has risen even higher and the increase in importance of feature 1 was less substantial than would have been expected from prior simulations.

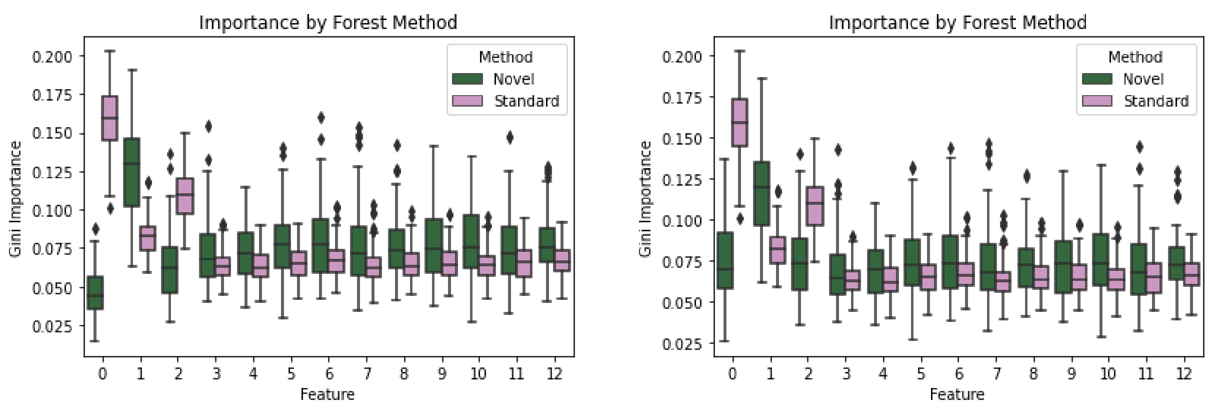


Figure 49. Feature / Factor Gini importance in a linear system simulation with randomness / noise and increased probability of missingness associated with the values of the adjacent factors for factors 0 and 2 without (left) and with (right) proportion correction. Note that the missingness of factor 0 is dependent on factor 1 and 2 on 3 where factor 1 has reduced randomness.

A remarkable peculiarity is that in examples presented in Figure 45, Figure 46, Figure 47, Figure 18, and Figure 49 there is an obvious increase in the standard estimation of importance for the features where the missingness is dependent on itself or another value. Along the lines of data stealing previously discussed, the reason for this relates to the technique used to apply value dependent missingness which normalizes the dictating feature about zero. Thus, a proportion of the values (i.e., half) fall above and its conjugate falls below zero with a symmetrical distribution. Overall, the average addition to the exponent representing the probability of missingness is 0. In other words, normalization means the likelihood that the data whose missingness is dependent on the feature value of greatest magnitude and that of the lowest magnitude are simultaneously missing is the same as two non-value dependent features simultaneously missing and resultantly all features with a symmetric value distribution, in the absence of any other determinants, will have the same proportion of missing values. Where this proportion is not value dependent this is scattered throughout the sample randomly. Where this is value dependent, it is focused on higher normalized values. If those values are in turn dependent

on a class outcome, then when incorporating them into the model as any placeholder value that place holder value will have more meaning as opposed to no meaning at all. Consider the simple example as follows with two variables with variables with random and value based missingness.

$$\begin{pmatrix} & 1 & 1 & C_L \\ \dots & 2 & 2 & C_L \\ & 3 & 3 & C_L \\ & 4 & 4 & C_L \\ \dots & 5 & 5 & C_L \\ & 6 & 6 & C_H \\ & 7 & 7 & C_H \\ \dots & 8 & 8 & C_H \\ & 9 & 9 & C_H \end{pmatrix} \xrightarrow{Random} \begin{pmatrix} & 1 & 1 & C_L \\ \dots & 2 & - & C_L \\ & 3 & 3 & C_L \\ & 4 & - & C_L \\ \dots & 5 & 5 & C_L \\ & 6 & - & C_H \\ & 7 & 7 & C_H \\ \dots & 8 & - & C_H \\ & 9 & 9 & C_H \end{pmatrix} \xrightarrow{Value} \begin{pmatrix} & 1 & 1 & \\ \dots & 2 & - & C_L \\ & 3 & 3 & C_L \\ & 4 & & C_L \\ & 5 & & C_L \\ \dots & - & \dots & C_L \\ & 5 & & C_H \\ & - & & C_H \\ - & & 7 & C_H \\ - & & - & C_H \\ - & & 9 & \end{pmatrix} \xrightarrow{Fill} \begin{pmatrix} & 1 & 1 & C_L \\ \dots & 2 & 5 & C_L \\ & 3 & 3 & C_L \\ & 4 & 5 & C_L \\ \dots & 5 & 5 & C_L \\ & 3 & 5 & C_H \\ & 3 & 7 & C_H \\ \dots & 3 & 5 & C_H \\ & 3 & 9 & C_H \end{pmatrix}$$

Clearly, the variable with the value based missingness will fare better with a potential for Gini reduction by 0.316 ($2 \times \frac{5}{9} \times \frac{4}{9} - 2 \times \frac{5}{9} \times \frac{1}{5} \times \frac{4}{5}$) versus 0.176 ($2 \times \frac{5}{9} \times \frac{4}{9} - 2 \times \frac{7}{9} \times \frac{2}{7} \times \frac{5}{7}$).

3. Composite Simulation

With regards to classification performance, the novel and standard random forest techniques had similar AUC values / ROC curves with a slight advantage to the standard technique. While the novel approach seeks to avoid the introduction of bias by avoiding imputation which may not model the data well, it in principle relies on trees that were individually biased in the same way that complete case analysis would be. Although forms of adaptive bagging have been proposed to reduce bias as trees are bagged, bias is not well addressed by standard bagging.[cxliv] In this case the tradeoff was not in favor of the novel technique.

Overall, the novel approach appeared to provide less biased feature importance estimates at the expense of precision (as seen by feature importance more reflective of known underlying randomization but with greater variability is the estimate itself). When a greater number of cases were available the trend towards observing the difference between more randomized and less randomized features was even clearer using the novel approach and the standard approach appeared to struggle with accurately identifying even the relative order of lesser to more randomized variables (i.e., the standard approach is more biased). Further simulations shown in appendix XIV demonstrate additional limitations of the novel approach that are not discussed in detail here but also suggest that a misperception of the most important variables in the standard random forest is not trivial and could compromise accuracy relative to the novel approach and other modelling techniques. In the end ensuring agreement of the novel and standard random forest approaches on the ordering of variable importance may be of greater practical importance to identify when such models may be at risk from the bias of missingness.

The increase in variability in the variable importance measures is likely in part explained by the reduced number of trees possessing importance estimates for any one feature, a issue that may be addressed by increasing the number of trees produced. With regards to the relatively reduced bias, assigning a value to missing data (or engaging a data handling step for missing data as used

in the BEST algorithm) can obscure appreciation of which features are most important by redistributing importance of variable through various means. This may then lead one to question if permutation importance may be a viable alternative to circumvent this issue although this technique had a critical weakness when viewing large quantities of data particularly where there are a significant number of predictive variables. With permutation importance, it is known that associations between variables can lead to reductions in perceived importance.[cxlv] Although clustering variables to avoid redundancy may be relatively straightforward in linear systems, that would be at least partly subjective and can pose a problem where predictors were non-linear particularly considering that the applicability to non-linear scenarios is a strength of machine learning approaches. Consider the following partial matrix (provided that other features do exist that have varying predictive potential):

$$\left(\begin{array}{cccc|c} \cdots & 1 & 9 & 1 & C_L \\ \cdots & 2 & 8 & 2 & C_L \\ \cdots & 3 & 7 & 3 & C_L \\ \cdots & 4 & 6 & 4 & C_L \\ \cdots & 5 & 5 & 5 & C_H \\ \cdots & 4 & 4 & 6 & C_H \\ \cdots & 3 & 3 & 7 & C_H \\ \cdots & 2 & 2 & 8 & C_H \\ \cdots & 1 & 1 & 9 & C_H \end{array}\right)$$

All three presented features in this situation possess the same ability to yielded perfect classification. C_H is identified by all those cases with a feature 1 value of 3 or more and a feature 2 or 3 value of between 3 and 7. If one were to use a traditional approach such as using Euclidean feature distance determined by hierarchical clustering of inter-feature correlations then features 2 and 3 may be easily identified as part of a group (adjusting for sense) but feature 1 would be left out of that group with near certainty (the correlation with either feature is 0). If one were to then attempt to explore feature importance by permutation an erroneous conclusion of unimportance of all three features would be made. Of course, one's suspicion that an error must exist in the analysis would be piqued by the high model accuracy without finding any value to be of particular importance, but inspection and/or implementation of additional techniques would be required to further decode the results. A labor intensive but fair approach for example might be to randomly adjust combinations of feature over all permutations of combinations. This may indeed actually be a robust way to cluster related data although this is beyond the focus of the discussion presented here. Another note about permutation importance is that, to avoid capturing the data of other variables encoded in the missing data of a given feature, permutation of the present values and then filling in the missing values best separates the value of a feature itself versus the missing data. Of course, the baseline model (e.g., random forest versus novel random forest) would also have some role in dictating the impact of missing data.

For the purposes of our simulation data, the use of permutation importance proved unhelpful. Due to a high degree of collinearity by design all feature permutation importance estimates have a mean of 0 and standard deviation of 0 with a perfect step characteristic curve (i.e., implies the data can perfectly characterize the outcome but there is a contradictory assessment that none of the features are of any value).

E. Conclusions

In this section, the novel random forest approach was developed, and simulations were used to demonstrate how it addresses potential bias introduced by missing data. Performance compared to logistic regression and SVM was shown. Standard and novel random forest approaches outperformed the other models. Novel random forest estimates of feature importance were less biased at the expense of overall model performance measured by AUC to a modest degree (i.e., ROCs remain substantially overlapped). In the following section, candidate techniques are applied to a defined data set (the wine data set) to further explore their relative performance.

VII. Chapter 4: Comparative Study on the Application of the Novel Random Forest Approach in a Defined Dataset

A. Introduction

To explore the properties of the novel random forest technique and demonstrate plausible applicability to the PACNS data set, a comparative study was conducted. The wine data is a freely available data set that was augmented to provide data with properties more in keeping with what would be expected of the PACNS data set allowing for an intuitive demonstration. The application of permutation importance in more nuanced real data as an alternative to Gini importance is also explored.

B. Methods

1. Wine Data Set Description

The wine data set contains chemical properties on a list of classified wines. There were 178 wines, 3 wine "classes," and 13 features / properties that could be used to characterize the set. The wine data set is a complete data set (i.e., a data set without any missing data). The data set was first changed to a binary problem whereby only the first class was distinguished from the other two class effectively changing this to a two-class classification problem analogous to distinguishing PACNS from all other diagnoses. A probability map was then constructed that included 4 separate components or masks to impose missingness of differing patterns. The practical analogies of each mask and their implementation is discussed in turn.

2. Data Processing

Data augmentation and processing to insert patterned missing data proceeded as described in VI B.

C. Results

First, we explored the basic properties of the available wine dataset. Note that features were referenced using the following number / name correspondence: 0: 'alcohol', 1: 'malic acid', 2: 'ash', 3: 'alcalinity of ash', 4: 'magnesium', 5: 'total phenols', 6: 'flavanoids', 7: 'nonflavanoid phenols', 8: 'proanthocyanins', 9: 'color intensity', 10: 'hue', 11: 'od280/od315 of diluted wines', and 12: 'proline'

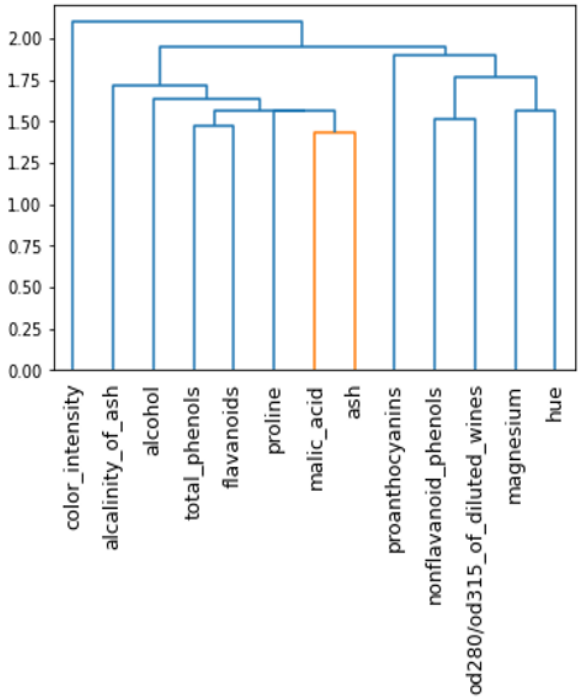


Figure 50.Hierarchical clustergram of wine data set.

Baseline performance on the wine data set was the same / excellent across methods (see Figure 51):

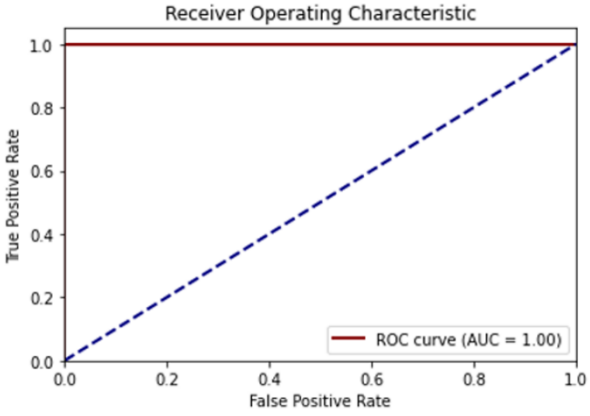


Figure 51. Receiver operating characteristic curves the novel technique / stock scikit standard random forest technique / the manually programmed standard random forest technique / the support vector machine technique / the logistic regression technique using the wine data set.

Feature importance estimates were similar by technique using novel and standard random forest techniques:

Novel – [0.096, 0.053, 0.040, 0.057, 0.056, 0.095, 0.14, 0.045, 0.051, 0.069, 0.058, 0.064, 0.18]

Standard – [0.10, 0.053, 0.036, 0.057, 0.065, 0.094, 0.13, 0.048, 0.053, 0.060, 0.051, 0.075, 0.18]

The order of importance is 12 > 6 > 0 > 5 > 9 > 11 > 10 > 3 > 4 > 1 > 8 > 7 > 2 versus 12 > 6 > 0 > 5 > 11 > 4 > 9 > 3 > 8 > 1 > 10 > 7 > 2.

Combining all forms of missingness (i.e., value, prior missingness and value based missingness with corrections reveal a retained high degree of accuracy) (see Figure 52).

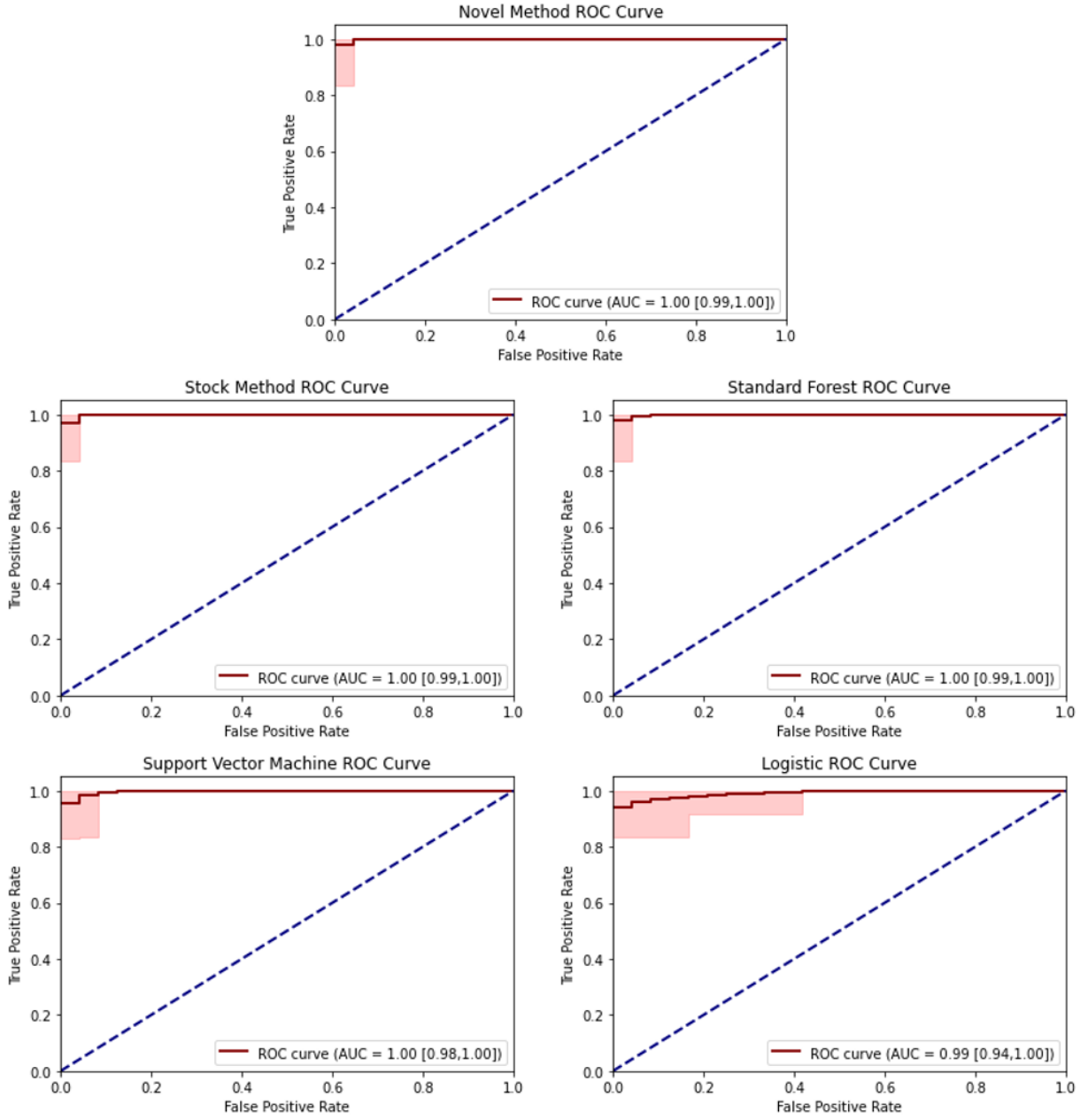


Figure 52. Receiver operating characteristic curves for a) the novel technique (top); b) the stock scikit standard random forest technique (mid-left); c) the manually programmed standard random forest technique (mid-right); .d) the support vector machine technique (bottom left); e) the logistic regression technique using the wine data set after complex data augmentation.

Importance of features follows a similar pattern (see Figure 53).

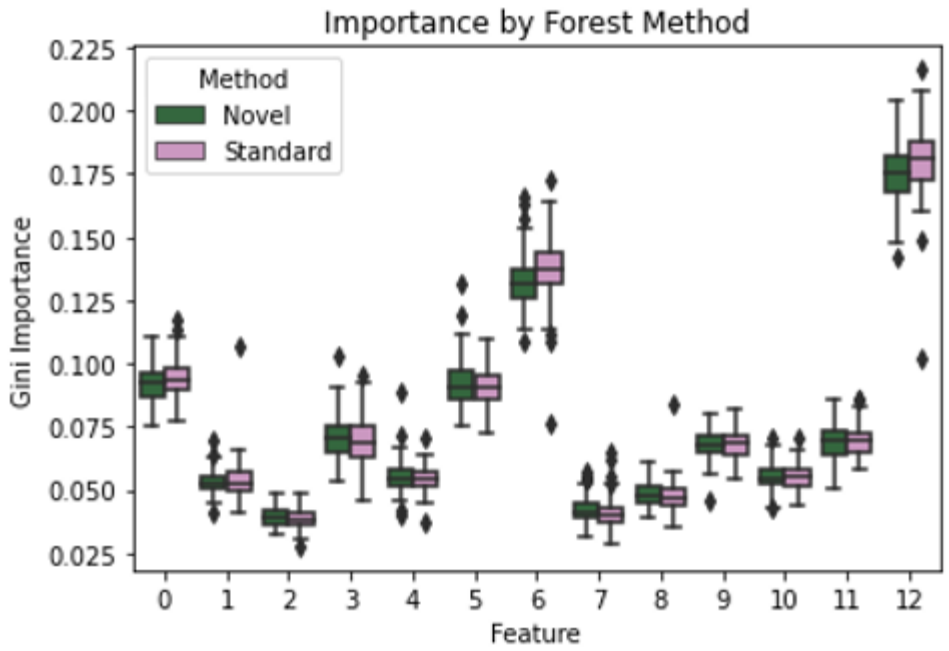


Figure 53. Feature / Factor Gini importance using the wine data set after complex data augmentation.

Note is made of agreement to original importance assessments and the most important features 12 > 6 > 0 > 5 with 3, 9 and 11 trailing behind and the least important features 7 > 2.

Using permutation importance based on accuracy shift (i.e., the default approach per the commonly available scikit learn python package) the order of mean variable importance is 12 > 6 > 5 > 0 > 11 > 1 = 2 = 4 = 7 = 8 = 9 = 10 > 3 and 12 > 3 > 1 > 6 > 0 = 11 > 2 > 4 = 5 = 7 = 8 = 9 = 10 using random forest and SVM techniques. The vectors containing mean importance are as follows:

Standard – [0.0044, 0, 0, -0.011, 0, 0.0061, 0.025, 0, 0, 0, 0, 0.011, 0.11]

SVM – [0.013, 0.025, 0.0097, 0.045, 0, 0, 0.021, 0, 0, 0, 0, 0.013, 0.18]

Using permutation importance based on ROC shift the importance estimates using a standard random forest technique are as follows (mean [standard deviation]):

0: 0.00087 [0.0015], 1: 0 [0], 2: 0 [0], 3: 0 [0], 4: 0 [0], 5: 0 [0], 6: 0.0016 [0.0027], 7: 0 [0], 8: 0 [0], 9: 0 [0], 10: 0 [0], 11: 0 [0], 12: 0.023 [0.015]

Which would suggest that factor 12 > 6 > 0 (i.e., the proline, flavonoid, and alcohol content) are important and all other features are not of importance in wine classification. If we restrict our consideration to 0: ‘alcohol’, 3: ‘alcalinity of ash’, 5: ‘total phenols’, 10: ‘hue’, and 12: ‘proline’ we find 12 > 0 > 3 > 5 > 10 without any reduction in ROC (i.e., ROC remains 1 despite variable reduction). The importance are as follows (mean [standard deviation]):

0: 0.019 [0.011], 3: 0.011 [0.009], 5: 0.007 [0.006], 10: 0 [0], 12: 0.20 [0.06]

This shows that factors 3: ‘alcalinity of ash’ and 5: ‘total phenols’ are not unimportant but their collinearity with some combination of 12: ‘proline’, 6: ‘flavonoid’, and 0: ‘alcohol content’ obscures the analysis of their importance. That said, in this example the classification characteristic curve is unchanged when substituting feature 6 for 3 and 5 and thus a model with only features 0, 6 and 12 would be more parsimonious but may not be most ideal. If for example it was less expensive to test for the combination of total phenols (5) and alcalinity of ash (3) and not fractioning out and quantifying the flavonoids (6), then the perception of an ideal testing algorithm could shift in favour of the less parsimonious model.

D. Discussion

When taken together, all techniques yield similar insights despite increasing patterned missingness although random forest techniques show they may have an edge. The difference from the simulated data likely relates to relatively less noise / randomness in the dataset. Gini and permutation importance also provide similar insights however permutation importance adds the complexity of nullifying colinear features requiring careful consideration of which variable should be included in generated models. There are ways to make such assessments more robust that require either careful consideration on the part of the user to reduce the number of variables used during modeling or increase the complexity / resources required for the calculation (i.e., using a random sampling or complete enumeration of all combinations of data sets with various variables randomized). Of course, the permutation importance can also be significantly influenced by the model at its base be it a random forest, a novel random forest or SVM. If missing data exists, the discussion surrounding data redistribution between features in the standard random forest technique could be highly relevant given this may promote collinearity that would suppress the importance of variables where missingness was not at random.

E. Conclusions

There are many readily available properties of wine but the ability to identify the type of wine correctly can be effectively reduced to measurements of proline, flavonoid and alcohol content. Other selections of properties may serve as substitutes but provide little in the way of additional discriminative power. The machine learning approaches (especially the random forest techniques) appear to provide a model with a slight tendency for greater accuracy while the random forest techniques additionally provide readily probed “feedback” to find these variables which are most important without invoking additional tools like regularization (e.g., LASSO, Ridge regression, etc.).

In this section we have shown the applicability of machine learning to a described data set and that there are subtle indications that it may be more adaptable to missing data. We also showed that permutation importance is a powerful technique, but it is limited by the collinearity of features which could make the preprocessing of data prior to modelling and the requisite justification a more intensive process. In the next section we applied the discussed machine learning techniques to the institutional PACNS sample.

VIII. Chapter 5: Modelling of the Ottawa Hospital Sample PACNS data set

A. Introduction

The final step in assessing the effectiveness of the proposed novel technique was to apply it to a real-world sample of patient cases to ascertain if it had the ability to identify patients with PACNS in a comparable way to established methods. Relative feature importance for variables describing the sample were also compared. Given the limitations in the PACNS dataset in terms of case collection (i.e., cases of suspected PACNS were not prospectively collected and instead retrospectively collected using a technique later described), and that key variables of importance such as imaging results and biopsy results were not included as they were not readily extractable from the Ottawa Hospital's database, this was considered as a purely a demonstrative example. Results interpretation / discussion therefore focused on how the results of the models may indicate the success of the models by how they coincide with the current understanding presented in contemporary literature as opposed to offering new insights.

B. Methods

1. Target Population / Data

a. Operational Definitions

The case definition for "suspected PACNS" relied on the nature in which clinicians would be anticipated to order testing to confirm or refute a diagnosis of PACNS as an indicator of their suspicions. While it is recognized that ideally cases would have been prospectively identified for the purposes of analysis and modelling, as the purpose of the modelling was to appreciate the relative importance of factors towards confirming a diagnosis, inclusion criteria were constructed for a practical case identification paradigm. Infections and systemic vasculitides are key differential diagnoses for PACNS. The combination of lumbar puncture and / or brain biopsy, which would serve to rule out or at lease significantly reduce the likelihood of infectious contribution, and ANCA testing, which when positive would lead a clinician to strongly consider if an ANCA positive systemic vasculitis were the true cause, was considered a basis for "reasonable testing" to investigate a case of suspected CNS vasculitis. This combination was used to define "potential cases of suspected CNS vasculitis" in the Ottawa Hospital Database for subsequent chart review. For a more complete description please see Appendix XI.

A case of "suspected PACNS" in the database analysis therefore included any patient for whom a diagnosis of vasculitis of the CNS was entertained (stated in the clinical notes as a possible diagnostic outcome of prescribed testing) and "reasonable testing" to pursue the diagnosis of PACNS was completed and logged. The minimum standard for "reasonable testing" was defined as the completion of at least antineutrophil cytoplasmic antibodies (ANCA) (i.e., a test to rule out systemic vasculitidies that are uncommon but know to impact the CNS) and a lumbar puncture (i.e., a test commonly used and technically necessary to rule out an infectious source of vasculitis prior to treating CNS vasculitis) or brain biopsy. The "initial diagnostic process" encompassed all testing completed within 30 days of the first documented ANCA test record at the Ottawa Hospital (i.e. 30 days before or after). Those cases, following verification by manual chart review, where the care team raised the possibility of central nervous system vasculitis and completed the minimal testing were included in the analysis.

b. Data Description

To explore quantity and character of the data warehouse case information available (i.e., to ensure viability of application of target methods), a synthetic data package employed by the Ottawa Hospital (i.e., MDClone) was used. Data synthesis is a concept within statistical analysis whereby anonymity of source data can be preserved by stochastically generating a data set using a framework that preserves inter-relationships between features.[cxlvi] Approximately 100 patients were found to have been diagnosed at the Ottawa Hospital (using data available between January 1, 1996 and August 30, 2019) with "cerebral arteritis, not otherwise specified," (CANOS) the coding for the diagnosis that best represents an idiopathic vasculitis of the central nervous system which is in effect analogous to PACNS. To try and draw on a sample where PACNS had been suspected a sample was selected from patients who had ANCA testing and CSF sampling within one month of each other (2,625 with both of these tests drawn of which 50 were CANOS) and those with ANCA testing and a brain biopsy within one month of each other (approximately 100 of which 50 were CANOS). It was expected that there could be partial / substantial overlap between patients with ANCA testing and CSF testing versus biopsy, nevertheless it was considered likely that a sufficient quantity of data would exists such that subsequent classification tasks would not be futile.

Limitations with the synthetic data ultimately required that the source data was obtained. The software / data was limited in its ability to provide only representative data pertaining to numeric / nominal simulated testing results and does not allow for the incorporation of more detailed elements that might help classify patients presenting with suspected PACNS such as initial symptoms / case history and more detailed reports such as on imaging or biopsy results. It also cannot ensure eventual diagnosis of patients who were seen at the Ottawa hospital although the causal diagnosis suspected at the end of their encounters during which their testing was obtained at the Ottawa hospital is known. For this reason, the simulated data was used purely to plan the extraction of the real / verified patient data from the Ottawa Hospital Data warehouse. All cases identified with the diagnosis of CANOS / PACNS and a selection of cases confirmed to have a suspicion of PACNS were fed into the classification algorithms for comparative analysis. To focus on separating PACNS from other vascular or infectious causes, from all cases with verification that a physician suspected vasculitis of the CNS at some point in their diagnostic workup, only patient with an eventual diagnostic code of "Diseases of the circulatory system" or "Certain infectious and parasitic diseases" were included.

c. Variable selection

The entire TOH lab hierarchy of extractable lab values was evaluated for tests that had been featured in the scoping review. A total of 4,132 were assessed for potential relevance in the context of PACNS diagnosis. This number represents the total for all separately coded tests performed at the Ottawa Hospital since the inception of its data base. Of note several variables were effective duplicates with slight variation in naming over time (e.g., CRP versus C-Reactive Protein).

2. Data extraction (selection and coding)

a. Study (case) selection / confirmation

To the cases extracted from the data warehouse, eligibility criteria were applied by one reviewer and verified by a second pass of the data.

b. Procedure

Charts were manually reviewed to identify the diagnostic conclusion of the treating clinicians as it pertained to the suspicion of CNS vasculitis. The "final" diagnosis was defined as the diagnosis that was last conveyed and documented in the available medical record. Where there were multiple diagnostic opinions, the discrepancy was noted and the "final" diagnosis reflected the treating neurologist's opinion over the rheumatologist's opinion, the treating rheumatologist's opinion over any other specialist position, and a specialist opinion over a primary care provider or generalist opinion. Once cases of "suspected PACNS" were confirmed, numeric laboratory results were extracted.

3. Theoretical Description of Modelling

a. Overview

For the planned predictive model design, cases were extracted from the TOH database that related to suspected PACNS per the operational definitions discussed. The scoping review (Section IV) informed the variables to be extracted from the selected cases and this information was used to apply one of several classification algorithms. Supervised machine learning techniques including random forest generation and support vector machines (SVM) were used (note that these are both supervised technique because outcomes are labelled to direct the learning algorithm as opposed to these having to be "guessed"). In brief, an ensemble approach was applied to create or "train" multiple independent assessors / assessment algorithms / decision trees or patterns with "training" data and then they all voted on "test" data. The majority vote of all assessors that can vote (i.e., some were not able to vote as a result of missing data) represented the prediction from the model and this was used to ascertain accuracy.

b. Context

As previously outlined, patients with "suspected PACNS" and documentation of the "initial diagnostic process" to confirm this entity accessioned within The Ottawa Hospital Data Warehouse were included / evaluated for analysis.

CSF results including Cell Counts, Protein, glucose Oligoclonal Bands, cryptococcal antigen, culture, and serologies were available for extraction. Blood testing results for Syphilis, VZV, HIV, ESR, CRP, ANA, dsDNA, C3, C4, ANCA, Cryoglobulin, Beta-2-glycoprotein Ab, Cardiolipin, and Lupus anticoagulant were also available. A sample of these variables was collected. The ultimate diagnosis of PACNS was coded as definite or probably PACNS although classification analysis focused on diagnosis of PACNS without this qualification.

Not all variables identified as candidate features by cross referencing the scoping review with the data warehouse coded testing were available in the gathered sample. Variables with textual results were available in most cases but ultimately not included in the models which served to simplify data pre-processing. In future analysis, the main body of reports discussing findings and opinions of CT, MR and conventional angiography reports could be searched and logged for the presence of keywords (affirmative context): PACNS, beading, concentric, , string-of-pearls, pearls, arteritis, vasculitis, angiitis, stroke, diffusion restriction / restricting, and vessel wall enhancement. Report of biopsies could be searched and logged for their tissue of origin (e.g. Brain, Temporal artery, Skin, etc.) and for the presence of the keywords: angiitis, arteritis, and

vasculitis. Cultures (from all sources) could be logged as normal, abnormal or contaminant (i.e. if they test positive for a typically non-pathogenic skin flora: coagulase-negative staphylococci, Corynebacterium species, Propionibacterium acnes, viridans group streptococci).

Case data regarding adolescents (under 18 years of age at the time of their first ANCA test), those who were not being investigated for PACNS, and those with known systemic cause of vasculitis prior to the onset of neurological signs / symptoms were excluded.

c. Participants/population

Inclusion criteria: patients initially being evaluated for "suspected PACNS" (i.e. vasculitis with onset of associated clinical activity within the CNS) and some form of "reasonable testing" that was available including ANCA (a test indicating that some consideration to ruling out systemic involvement of vasculitis was given) and either of CSF sampling (test indicating that infectious causes were at least minimally considered) or brain biopsy (tests indicating an in depth approach to determining cause that would cover consideration of infectious cause was given) within 30 days of each other.
Exclusion criteria: adolescents (under 18 years of age), know systemic cause of vasculitis prior to the onset of neurological signs / symptoms

d. Intervention(s), exposure(s)

Application of various machine learning and traditional statistical techniques (e.g. ANCA status to indicate systemic vasculitis, blooming on echo planar or susceptibility weighted magnetic resonance imaging to suggest amyloid related inflammatory response, Histoplasma, Aspergillus, and Blastomycosis in the blood constitute an infectious serology group for endemic fungus) to separate out mimickers to ultimately confirm the likely diagnosis of PACNS.

e. Comparator(s)/control

Not relevant.

f. Main outcome(s)

The accuracy of prediction of underlying PACNS versus other cause in patients initially suspected to have PACNS. The relative influence of various variables was extracted as well to indicate the most significant variables as it pertains to classifying PACNS.

g. Measures of effect

Sensitivity, Specificity, Accuracy, Gini Importance

h. Additional outcome

The prediction of an alternative cause of CNS vasculitis in patients initially suspected to have PACNS. Demonstration of pathology other that PACNS on biopsy serves as a gold standard of diagnosis but was often unavailable. The final diagnosis provided within the medical record served as a probable / effective diagnosis.

i. Measures of effect

Receiver operator characteristic curve / area under the curve

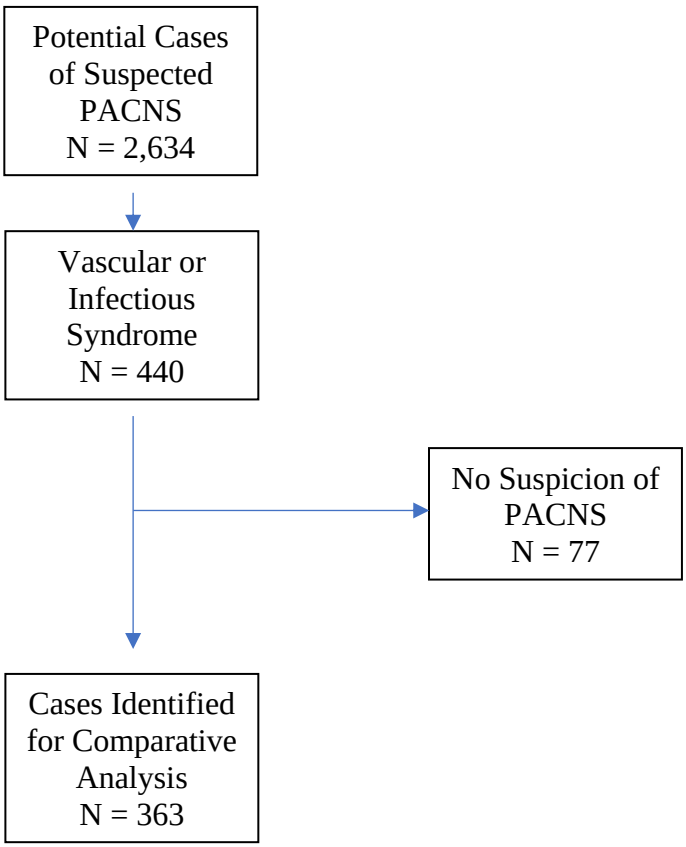
C. Results

1. Variable Selection

Of the 4,132 tests listed in the TOH Data Warehouse, 469 were considered of potential relevance to separating a case of PACNS from mimickers. Of these 469 candidate variables a sample of 23 was selected from those available variables that were found to be reported for at least one patient in the evaluated population (i.e., those cases of suspected PACNS where a vascular or infectious precipitant of the clinical was eventually diagnosed). See section XIII (Relevant Files) for the file name containing the source data.

2. Data Extracted

In total, 2,634 cases were extracted for potential review from TOH's data warehouse. Of these cases, 292 carried an encounter diagnosis attached to the relevant diagnostic testing of "Diseases of the circulatory system" and 148 carried an encounter diagnosis of "Certain infectious and parasitic diseases." On review, 77 of these cases were not in relation to assessment of a potential primarily central nervous system vasculitis and either, for the most part, possessed primarily a peripheral clinical quality / suspicion of vasculitis involvement (e.g., mononeuritis multiplex) or there were clinical stigmata of a systemic vasculitis and a non-specific neurological phenotype (e.g., delirium). Only 24 cases where PACNS was suspected, was the diagnosis confirmed. See Figure 54.



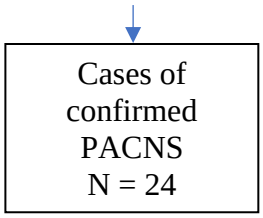


Figure 54. Case selection for comparative analysis

The sparsity of the variables (i.e., the percentage of missing values per variable) was as follows:

Table 3. Input variable sparsity of features for patients with suspected PACNS who ultimately were not diagnosed with PACNS (i.e., Non-PACNS), were diagnosed with PACNS, and the overall sparsity (i.e., Both).

Variable	Non-PACNS	PACNS	Both
Sex	0.00	0.00	0.00
ESR	10.32	4.17	9.92
Urine Protein (numeric)	92.33	91.67	92.29
Urine Protein-Result (textual)	20.06	16.67	19.83
Paraneoplastic Antibody Positivity	96.76	95.83	96.69
Cardiolipin IgG	53.69	37.50	52.62
Cardiolipin IgM	53.69	37.50	52.62
Beta-2-Glycoprotein IgG	78.47	62.50	77.41
Beta-2-Glycoprotein IgM	78.47	62.50	77.41
Compliment C3	33.63	8.33	31.96
Compliment C4	33.63	8.33	31.96
CSF Protein	1.77	0.00	1.65
CSF WBC	10.62	16.67	11.02
CRP	22.42	4.17	21.21
ANA Titer	82.60	91.67	83.20
Homocysteine	84.37	83.33	84.30
Protein C	75.22	58.33	74.10
Protein S	78.17	54.17	76.58
p-ANCA	0.00	0.00	0.00
c-ANCA	0.00	0.00	0.00
ANCA pattern 1-Result	0.00	0.00	0.00
ANCA pattern 2-Result	0.00	0.00	0.00
Total	41.19	33.33	40.67

3. Comparative Analysis

Selected variables tended fell within groupings of compliment testing and then inflammatory disease testing in general with some more gradual accumulation of unclustered variables. The urine protein numeric result served as an outlier.

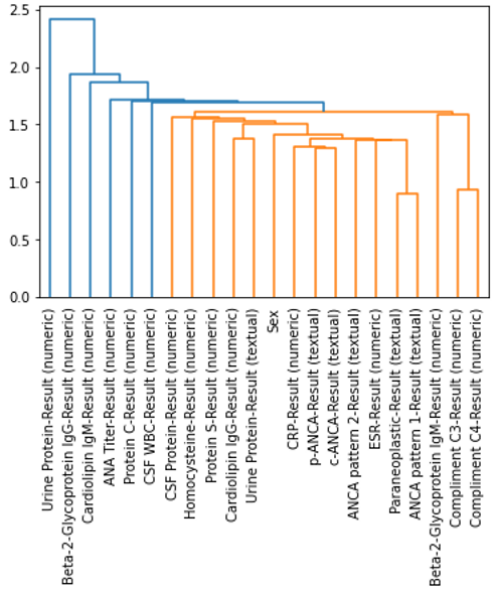


Figure 55. Clustering of select variables predictive of PACNS diagnosis.

Classification performance was comparable across classification algorithms (see Figure 56).

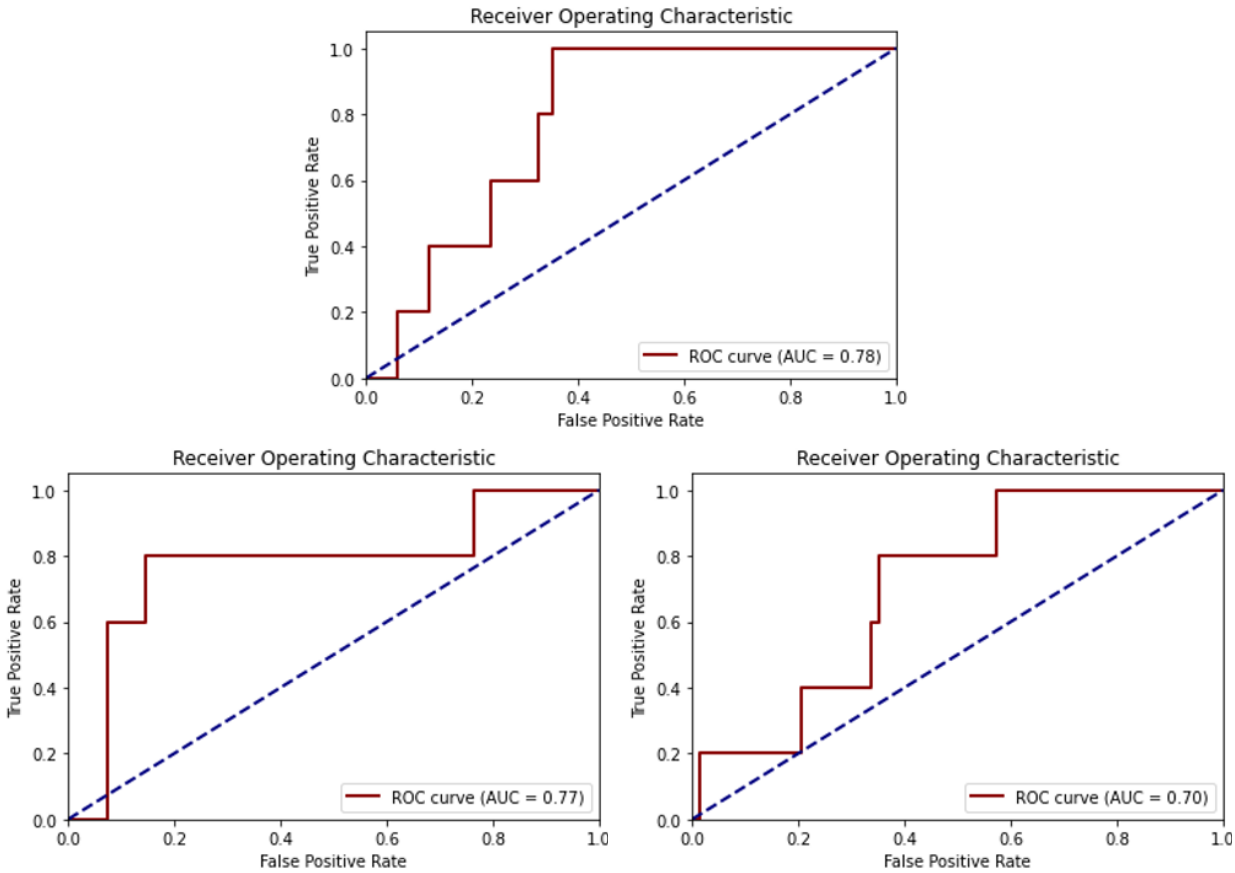


Figure 56. Receiver operating characteristic curves for a) the novel technique (top); b) the stock scikit standard random forest technique (bottom-left);.c) the support vector machine technique (bottom right) using the unaltered TOH data warehouse dataset. Note that logistic regression failed to converge at a reasonable number of maximum iterations (i.e., 10,000).

Feature importance tended to be higher for continuous variables (i.e., Beta-2-Glycoprotein, Cardiolipins, Compliment 4, Compliment 3, CRP, CSF Protein / WBC, Homocysteine, and Protein C / S) than nominal / ordinal variables (i.e., ANA, ANCA, paraneoplastic antibody presence, sex, and urine protein). See Figure 57. This tendency was less apparent when pseudo-randomness was incorporated.

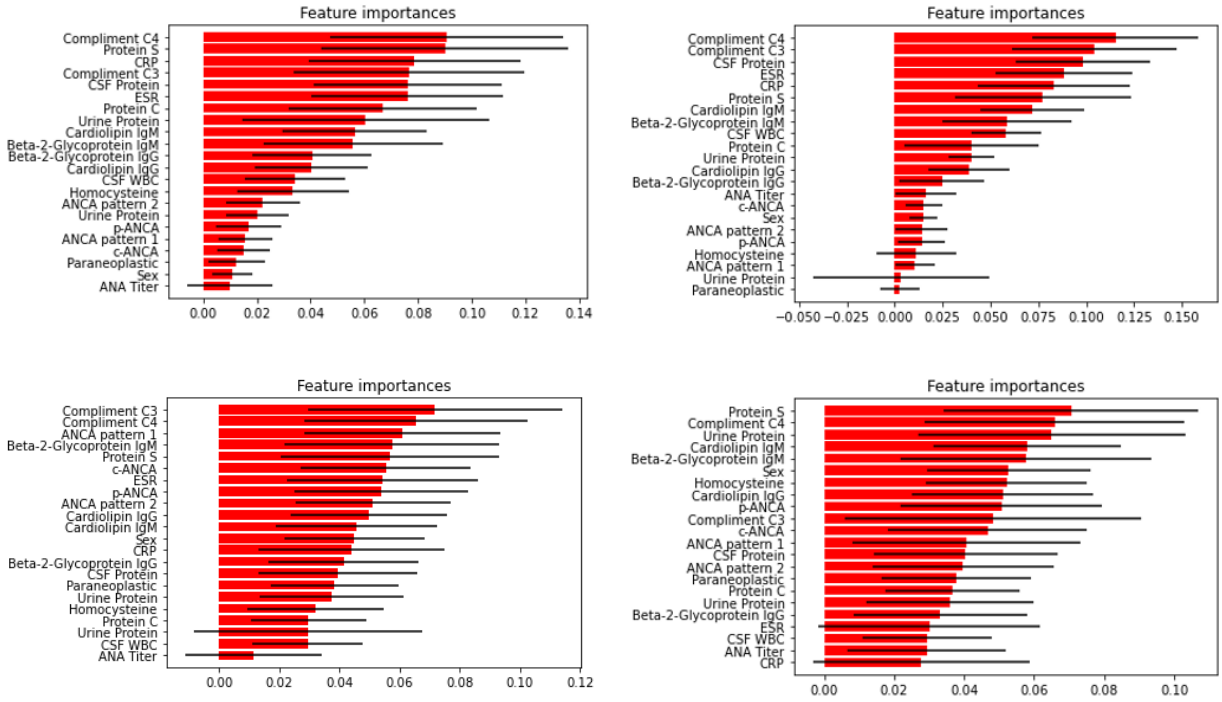


Figure 57. Gini importance related to a) the novel technique without data pseudo-randomness (top-left); b) the stock scikit standard random forest technique without data pseudo-randomness (top-right); c) the novel technique with data pseudo-randomness (bottom-left) using the unaltered TOH data warehouse dataset; d) the stock scikit standard random forest technique with data pseudo-randomness (bottom-right).

Permutation importance was unhelpful with all variables included resulting in nil importance assessed for all variables across methods suggesting a large degree of inter-correlatedness. Note that this same issue was the likely source of convergence failure in the logistic regression analysis.[cxlvii] Given the nested nature for the clustering of the features, a clear selection of features to reduce inter-correlatedness was not obvious and further analysis and was omitted.

D. Discussion

1. General Comments

Given the limitations in the features and population included in the modelling process, these models would be expected to have their own limitations with regards to diagnosing PANCS. Of note, the described initial survey of the Ottawa Hospital Data Warehouse suggested that there could be as many as 100 cases of PACNS (i.e., 100 patients were diagnosed with CANOS), only 50 were extracted by the proposed case identification technique and of those only 24 were truly

diagnosed with PACNS. There were still patterns of note with regards to the results of the various approaches however that are subsequently discussed.

2. Input Data

Except for two features (i.e., CSF WBC and ANA Titer) there was greater sparsity of feature values in the non-PACNS cases (see Table 3). This supports the view that PACNS is to some degree a diagnosis of exclusion and that patterns of missingness could conceivably be used by a modelling technique to infer a diagnosis (i.e., a case with more variables present would be more likely to represent a case of PACNS).

3. Data Clusters

The PACNS features selected appear to arrange themselves in a significantly vertical fashion with little clustering per se. Beta-2-Glycoprotein IgM and compliment C3/4 form one cluster of variables which are all indicative of inflammatory disease (specifically systemic lupus erythematosus). A second cluster including Sex, CRP, ESR, ANCA, Paraneoplastic Panel Positivity and ANCA appears to indicate systemic inflammatory disease indicators. These two clusters are then within a larger grouping that includes the presence of urine protein, Cardiolipin IgG, Protein S, Homocystine, and CSF protein that can, apart from CSF protein (which is a non-specific indicator of CNS afflicting disease), indicate a greater probability of stroke for systemic reasons. Accumulating beside this cluster are unclustered variables that are more synchronized with the larger cluster than the other unclustered variables (CSF WBC, Protein C, ANA, Cardiolipin IgM and Beta-2-Glycoprotein IgG). Outside of this resides the quantitative urine protein result which may indeed be less synchronized with other variables with respect to a diagnosis of PACNS as it may be more indicative of systemic disorders such as diabetes and hypertension and not an inflammatory condition.

4. Modelling Results

Applying the novel and standard random forest and SVM techniques showed similar results with regards to a general ability to separate PACNS cases from non-PACNS cases in this sample population. The Gini importance patterns differed, however. Without the introduction of pseudo-randomness, the categorical / nominal variables including ANA titer, ANCA patterns, Sex, and presence of protein in the urine are uniformly less important in the novel method and mostly that case in the standard random forest approach where homocysteine is featured as a variable of less importance (note however that homocysteine is reported as an integer number with a limited range). This phenomenon is further explored in section VI. 5. c. regarding cardinality correction.

When pseudo-randomness is added, both techniques identified that complement levels, which are markers of systemic inflammation including significant infection, remained important features.[cxlviii, cxlix] The presence of ANCA appeared to be relatively more important using the novel technique than the traditional random forest technique. Where the novel technique is interpreted as presenting the importance of features for those cases where the feature was available it would support the idea that ANCA has fair sensitivity and specificity for detecting an alternative condition (i.e., ANCA positive systemic vasculitis).[cl] For the data set as a whole, ANCA positive vasculitis was uncommon and probably would be tested / values present in the data set less often where there is a suspicion of PACNS as systemic manifestations would typically be apparent during initial consultation. Selective testing tendencies therefore were

likely the reason for improved performance in the novel approach. On the other hand, selective tendencies, where insightful, should serve to boost the importance of a variable in the standard random forest model (a phenomenon discussed as importance stealing in subsequent discussion). Resultantly, there are likely a variety of interactions with missingness and the "noisiness" of the variables that led to a relative decline in the feature importance. The difference in urine protein performance is likely an even more telling indicator of the impacts of applying a random forest to data where missing values have been filled versus the novel approach. Intuitively, one would expect the information that a continuous value reflects would be similar to the information possessed by its textual / ordinal counterpart. Where the novel approach is applied, the performance is in the lower range of relative importance whereas in the standard random forest approach one incarnation is a top performer (3rd most important variable) and the other a low performer (6th least important variable). A main advantage of the novel variant was that it appeared to simplify many of these complexities around missing data which was discussed when explored reasons for differences between novel and traditional random forest techniques using simulated data.

The relative importance of ANA was also of interest and regardless of correction or technique, this was a feature of lesser importance. Although the ANA titer is relevant to the diagnosis of systemic lupus erythematosus (SLE) being an "entry criterion" to considering the diagnosis of SLE per the 2019 European League Against Rheumatism/American College of Rheumatology Classification Criteria for Systemic Lupus Erythematosus, it is a non-specific test, and a positive titer of varying levels can be seen in 3 – 15% of healthy patients.[cli, clii] In fact this value appears to be age dependent and may be as high as 10 – 37% in those over age 65, which is of particular importance when considering an average age of onset of PACNS of 50.[cliii] It is also the case that Lupus itself tends to reflect a vasculopathy rather than a vasculitis per se when the nervous system is impacted and thus the presentation of Lupus related vasculopathy may less closely overlap with presenting syndrome of PACNS than other vasculitidies or systemic phenomena.[cliv] It would appear therefore that ANA is a feature of lesser importance could be a valid one.

The performance of sex serves as an informative standard to interpret the relative importance of the variables. PACNS is up to 2 times more common in men than women whereas RCVS, a top contender in the differential, is more common in women.[clv] Biological sex therefore has some predictive power and is available for all patients. In the novel approach, markers of various inflammatory disease are relatively more important whereas Sex is the 6th most important variable in the standard random forest approach with intermediate performance with respect to other inflammatory markers. Although it is possible that sex has much more discriminative potential than many of the factors presented, it is likely that the discriminative potential of the inflammatory markers is similar particularly considering the cluster analysis where sex was a part of but a relative outlier to the systemic inflammatory disease markers cluster.

Finally, the relative performance of CSF WBC was of note. Being one of the few variables that would have had the potential ability to separate infectious CNS precipitants of the presenting syndrome given that infections would typically be expected to increase CSF WBC levels more than non-infectious diagnoses, its importance was relatively poor being the worst or next to worst performer. This might reflect that infectious vasculopathies / vasculitidies are more associated with "atypical" infections of the CNS like fungus and spirochetes (where cell counts

are often in a lower “inflammatory” range) and less so more frequently clinically significant infections seen with a bacterial etiology (where a higher cell count would be expected). Regardless the novel and standard random forest techniques gave a similar perspective on the importance of his feature in this limited dataset.

E. Conclusions

With regards to diagnosis of PACNS within this limited dataset, compliment levels appeared of importance but ANA and CSF WBC levels less so. Other markers of inflammatory disease including ANCA, and Anti-phospholipid antibodies may be helpful as well although larger datasets with a more comprehensive capture of cases where PACNS was considered would be required to better understand the relevance of these observations. As it pertains to understanding how the results reflect upon the relative success of the methodologies, the clustering of the variables by their importance appeared to provide some insight into the relative function of these modeling techniques. The novel approach appeared to collect features associated with inflammation above sex which, although sex is associated with the diagnosis, it would be intuitive for these variables to have similar importance relative to a demographic feature such a sex.

In this section we have applied machine learning (i.e., random forest variants and SVM) to a sample of data and found that random forest techniques seem to perform well compared to SVM. The results of the novel random forest and the standard random forest were comparable in both accuracy and variable importance estimation although the results are potentially more intuitive particularly when cardinality correction is applied.

IX. Conclusions

A. Review of Machine Learning Approaches

The novel approach to random forest generation provides similar discriminative power to a standard approach where the missing values are filled with a single value but insights into the importance of variables can vary significantly. Where the input variables are highly predictive (see Figure 31), the Novel approach and standard random forest are demonstrably superior to more traditional techniques like logistic regression and even support vector machines despite the presence of large quantities of missing data (i.e., 50% sparsity). Even in the situation where real data with a high degree of sparsity (i.e., 35%) is used, as shown for the PACNS data, this observation still holds with similar performance metrics for the Novel and standard random forest techniques (i.e., ROC AUC of 78% versus 77%) that are superior to SVM (ROC AUC of 70%). Therefore, the novel technique dismisses some data when generating any given tree, but the nature of the random forest approach allows much of this data to still be captured in other trees within the forest to retain a strong discriminative potential that matches or exceeds contemporary classification techniques.

While the performance of the novel and standard random forest techniques performed similarly in defining classes of interest (e.g., PACNS from non-PACNS), the novel technique addresses missing data by creating trees from locally complete datasets (where less data is rejected than in complete case analysis) and avoiding additional bias that might be imposed from inaccurate imputation. Essentially, the novel technique ensures the components of the model (i.e., trees) are constructed with known data, but because random forest assembles these multiple trees into a

forest, the novel technique is not unduly biased by the dismissal of large swaths of information (as in complete case analysis) where sparsity is substantial as previously discussed. Additionally, as the component trees are only constructed from known and not imputed data, estimates of importance from these trees appear more insulated from the bias injected from missing data and its underlying pattern. As this bias can have a profound effect, the novel approach allows for a more accurate attribution of importance to every variable considered and without significant preprocessing of the data. By knowing those variables that are most important to whittle down to a diagnosis of PACNS, clinicians could tend to order those up front to converge to a diagnosis more rapidly. Additionally, where clinicians are stuck after already considering a battery of more important tests, they can also get a sense of what tests remaining (i.e., not yet considered / completed) still have some value and avoid unnecessary and in some cases invasive / expensive testing

Overall, the novel approach has properties that clarify its interpretation with respect to importance estimates and may make it a valuable technique to researchers that are considering which available factors in their data set are worth further consideration with respect to making a diagnosis.

B. Limitations

With regards to the general approach and analysis performed in this thesis there were several limitations. The measure of accuracy selected was that of an AUC for an ROC. While this does simplify comparisons is does ignore some of the more useful notions in quantitative diagnostics, in particular a tendency of the model to be more or less sensitive versus specific. In the absence of a perfect model, the bowing of the ROC curve to the top right could indicate a model that is good for screening for a diagnosis / ruling out a diagnosis (i.e., more sensitive where there is lower specificity) and bowing to the bottom left could indicate greater potential for ruling in a diagnosis (i.e., more specific where there is lower sensitivity).[clvi] Consideration of this could reframe future assessments to ascertain those models which are best able to perform in select contexts. For example, in diagnoses that tend to be more exclusionary (i.e., diagnoses that do not possess a readily accessible single or combination of tests that obviate the diagnosis to a substantial degree), such as PACNS, that a preference for models skewed to a more sensitive domain would be preferable particularly when there may later be more invasive confirmatory testing like brain biopsy. This underscores a concept in diagnostics of how one might incorporate testing in a parallel or sequential fashion to come to a diagnosis.[clvii] Clearly clustering certain tests into a model (or models) of less invasive testing in series with a test that would require brain surgery has relative appeal from a patient care perspective to the alternative of applying these in parallel. One could indeed conceive of features being placed into tiers of models that refine sensitivity versus specificity in a way that rationally limits the expense in financial and clinical terms to patients and healthcare systems.

Another limitation of the analysis in the thesis is that AUC does not provide us with the more granular insights with regards to diagnostic inaccuracy, including the impacts of missingness on single case scale. Although one could analyze cases one at a time to see when errors in diagnosis are made depending on what various patterns of missingness exist, it is difficult to know how one could consolidate such insights. Another approach may be to create predictions on a population comprised of repetitions of the one case with a single missingness pattern and varying

the missingness pattern to identify if there are cases that are more sensitive to misclassification. One could then attempt to cluster cases by diagnostic error given a missingness pattern and attempt to identify those characteristics that made the case more susceptible under different patterns of missingness. This may in time, for example, better elucidate what situations favor the novel versus standard random forest approach.

There were also limitations with regards to the datasets used to demonstrate the effectiveness of the novel approach. The simulated dataset was based on a set of highly co-linear features to ensure equal importance to each variable. This was not the only way to achieve this effect and certain patterns of additive and multiplicative as well as additional higher order interactions could have been simulated and may have been of relevance. Although the use of the wine dataset was intended to address this by providing a well characterized dataset with more realistic properties / feature interaction, the data proved to have too little noise (in the simulation represented by randomizing the feature values) to show much difference between models even with the same missingness pattern as the simulation. In essence the data was of such a high quality that even with 50% of the feature values missing, very clear inference could be drawn from the remaining data. Finally, the PACNS data has only 24 positive cases which limits the generalizability of any model that could be constructed from the data particularly where the properties of the data may be heterogeneous as previously described.

C. Future Steps

1. Methodological

The novel random forest design was shown to have comparable bias to a standard random forest technique in simulated and real-world data but provide more intuitive results with regards to the assessment of feature importance. Techniques for tree debiasing could be considered. The weighting of trees could be considered in this case and may benefit from incorporating insights from the lockout matrix. While adjusting the Gini importance was used here to debias the feature importance estimates in this thesis, it may improve model accuracy to instead use those insights to adjust the voting weights of the trees. Properties of combining imputation where appropriate and the novel approach could also be explored noting that the novel random forest and imputation are not incompatible. Further comparisons of the novel technique with a generative approach could be analyzed as well.

2. Clinical

The primary goal of this foundational work was to assess and propose the tools to extract insights from the heterogenous and sparse data expected of PACNS diagnosis across multiple clinicians and centers. With the resources at the Ottawa hospital, the 2600+ patients that were identified as having potentially been reviewed for PACNS could undergo case review to establish where PACNS was truly suspected and more broadly apply the proposed techniques over the 400+ variables. Further case identification could be facilitated by incorporating a search of notes, testing and imaging reports for the suspicion of PACNS and PACNS type pathology to capture the target patient population more completely. Following this, the next step in analyzing an ideal approach to PACNS diagnosis would be to engage members of the CanVasc community.[clviii]

CanVasc is a non-profit scientific network based in Canada that identifies, as a primary role, an objective to “initiate, conduct, and promote studies” related to vasculitis. By combining data

using a similar methodology to that proposed for the simulation presented on PACNS data, feature importance analysis would more rigorously highlight candidate variables that should be routinely collected in a prospective analysis. A prospective study analyzing the diagnostic approach where patients are identified as being suspected of PACNS vasculitis could then be discussed between CanVasc core members. Although additional variables would be included / required, the relative importance of various features could be verified and so long as no new significant candidate variables came to light, a model for PACNS diagnosis could be explored. Features of importance and the context of their use could then provide clinicians / CanVasc core members quantitative support for a discussion of diagnostic approach in PACNS.

Finally, with the benefit of the analysis regarding which variables could be diagnostically important, discussion and ultimate consensus regarding a diagnostic process could take the form of a Delphi / Modified Delphi approach where clinicians would first identify which tests are helpful / could be helpful for ruling out mimicry / confirmation in PACNS diagnosis.[clix] Subsequently, clinicians would disclose the initial testing they would feel is warranted in differentiating patient with PACNS. After consensus on the first round of testing, a second round could be discussed where the prior results had not elucidated any secondary cause.

X. Summary / Contributions

Modern statistical techniques have the ability to provide insight into rare disease that would not have heretofore been easily extractable. PACNS is a rare and disabling entity, particularly when treatment is delayed, and techniques are required to converge to a diagnosis rapidly. Understanding what variables are of greater significance in the diagnostic process better guides clinicians to consider reasonable alternatives when results do not align with expectations. Furthermore, developing institutional algorithms enhance future clinician's recognition of unusual entities. While often tasked with inference on large data sets, Machine learning and related methods like random forest modeling have a capacity to efficiently interrogate small quantities of data to seamlessly provide the insights that enhance such efforts. Furthermore, techniques that also adapt to missingness while retaining and minimally augmenting the small quantities of data available in rare disease could yield transformative realizations in care of patients with rare diseases.

When applying random forests, while purity / entropy-based assessments have limitations compared to permutation analysis, they also can assess a greater range of variables that could be co-linear in complex ways. One such limitation, that of a preference for continuous versus discrete variables, is readily, albeit expensively, addressed by the addition of pseudo-randomness. The novel random forest variant that was proposed in this thesis does not assume distributions of missingness, although it is compatible with having at least some missing values imputed should it be deemed appropriate (by whichever method is deemed appropriate no less). Even individual trees could be constructed using a generative approach within data subsets (i.e., generative trees could be used for training but if future data was presented to a tree that relied on data that was missing it would abstain from a predictive classification vote). In effect the novel techniques are not necessarily an alternative to the techniques presented in the review but an additional tool to be used, subject to its own biases, where it is challenging to understand the nature of the missing data. For those missing values where a method that could adapt to the

missingness is unclear, trees can be formed from subsets of data that have all features of interest present. By creating subsets of data with a randomly selected feature space before forming the trees, one is acting in accordance with a motivating principal of random forests which is to create a diverse set of trees. In some situations, very few cases will be removed when generating the tree of interest and would be less susceptible to bias. In other cases, bias could be increased where the data is MNAR (should the missingness be dependent on the classification as is seen in complete case analysis), but at most this bias for an individual tree would not exceed that of complete case analysis and the bias of the most biased trees would be counterbalanced by being aggregated with other trees with less bias. The novel random forest variant provides a balance of prediction accuracy and feature importance interpretability in the presence of multiple forms of missingness.

XI. Central Nervous System (CNS) Vasculitis Case Definition

As there are several ways that CNS vasculitis can present, with all including some form of CNS dysfunction, neuroimaging is often among the first steps in the diagnostic process. While this would make it a natural test to focus a search for suspected cases around it would not necessarily be expected to cover all presentations. Multiple infarcts, characteristically in various stages of evolution, would generally be considered the most obvious imaging manifestation of CNS vasculitis, however it is possible that a tumor like lesion or no imaging abnormality is found (e.g., as could occur in some encephalopathic presentations). It is therefore unclear whether searchable findings or useful differential diagnosis would be presented in the body of the imaging study. If one then considers completion of the definitive testing at the end of the diagnostic process regarding "suspected PACNS" (probable or possible) as a means of identifying suspected cases then it was likely that some cases will be lost due to alternative cause being discovered during the usual period of at least several days required to arrange invasive tests (i.e. the source of the presentation was found and thus the definitive testing cancelled). Therefore, a definition that incorporates testing likely to be completed and / or testing that would need to be complete when presented with the sorts of neurological conditions that can present similarly to PACNS would be expected to be more inclusive and likely to capture a more complete scope of cases (to subsequently be verified on manual chart review).

Blood work was often first ordered and would likely cover broadly secondary causes of vasculitis when PACNS might be expected. Although more general tests of inflammation would be requested, they could be ordered in a multitude of other clinical scenarios. The testing of ANCA would be more suggestive that a vasculitis was specifically being considered. Next, in the absence of any information that could draw one to suspect a specific source of vasculitis, (e.g., the patient was known to have a prior to presentation consistent with granulomatosis with polyangiitis, prior positive blood culture / septicemia, etc.) it is likely that clinicians would order more invasive testing to rapidly direct and/or ensure the safety of treatment which would include at least a search for infection of the central nervous system. Often a test to rule out infection of the nervous system would include CSF sampling although it is possible that biopsy of the brain would be favored in a tumor like presentation and therefore CSF sampling not performed.

There are two other scenarios to consider before proposing a final definition of "suspected PANCS." It is also possible that a biopsy of a secondary site other than the brain could be considered sufficient to have suggested a diagnosis (e.g., diffuse b-cell lymphoma in a random skin biopsy). This "loss" of cases may be countered by a need to treat with medications that might suppress the immune system and thus some CSF sampling may still be required to pursue further treatment unless otherwise contraindicated. Finally, it is possible that a biopsy is done without a suspicion of PACNS but then the suspicion of PACNS arises. Here the usual flow of testing would be reversed in that the blood work to rule out secondary vasculitis would be ordered only after the biopsy raised the suspicion. See Figure 58 for details.

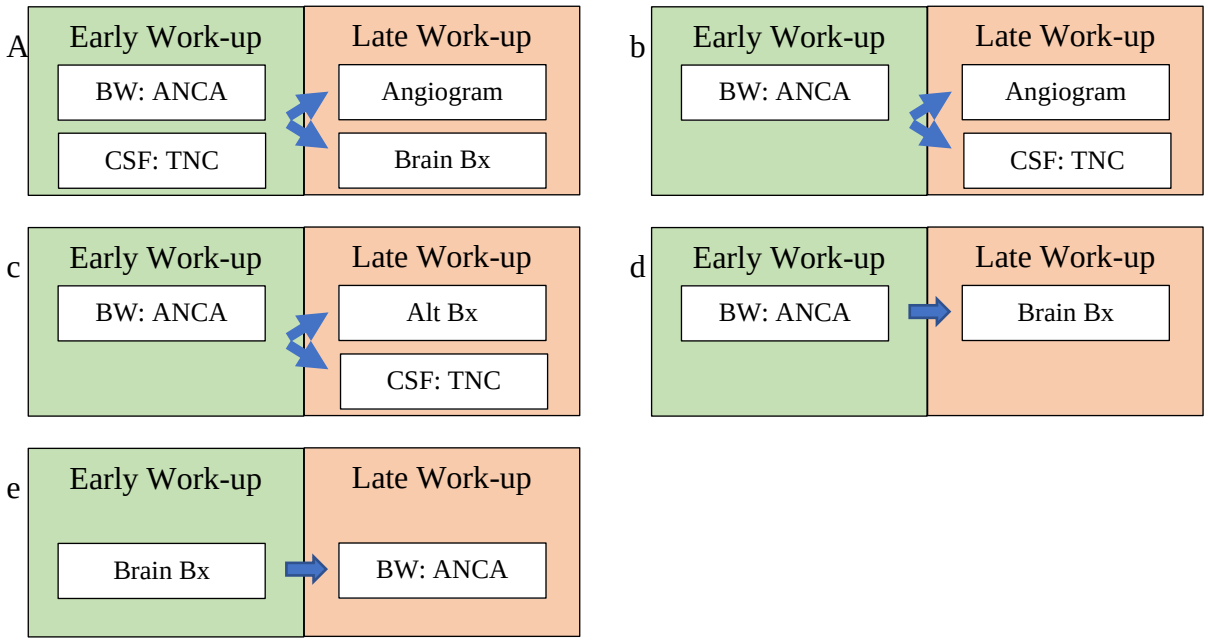


Figure 58. a) the typical pattern of vasculitis work-up when it involves the central nervous system to rule out certain diagnoses in the early phase and then confirm and PACNS diagnosis with angiography (probable PACNS) and/or biopsy of the brain (definite PACNS). B) the work-up pattern whereby there was early consideration of vasculitis but the extent of involvement of the CNS was not known and ruling out an infection in the absence of a biopsy was ultimately require. C) a scenario where an alternative tissue site was used to confirm a diagnosis but ultimately CSF testing would ultimately be required to proceed with treatment. D) in this patten there has been a direct move toward biopsy which can rule our infection. E) in this pattern the biopsy was done and the need to consider a diagnosis of vasculitis arose after the results of the biopsy were known. Alt: Alternative (i.e. alternative tissue sampling other than the brain); ANCA: Anti-neutrophil Cytoplasmic Antibodies; BW: Blood Work; Bx: Biopsy; CSF: cerebrospinal fluid; TNC: Total Nucleated Cells

XII. Supporting data

A. Linear Simulation

1. Dependent Missingness

j. Lockout Matrix

7242	631	659	623	638	627	660	632	622	599	620	3796	3744
631	7097	606	598	659	595	586	628	616	575	640	3708	3725
659	606	1265	0	0	0	0	0	0	0	0	644	621
623	598	0	1221	0	0	0	0	0	0	0	597	624
638	659	0	0	1297	0	0	0	0	0	0	640	657
627	595	0	0	0	1222	0	0	0	0	0	620	602
660	586	0	0	0	0	1246	0	0	0	0	633	613
632	628	0	0	0	0	0	1260	0	0	0	605	655
622	616	0	0	0	0	0	0	1238	0	0	635	603
599	575	0	0	0	0	0	0	0	1174	0	571	603
620	640	0	0	0	0	0	0	0	0	1260	664	596
3796	3708	644	597	640	620	633	605	635	571	664	7189	634
3744	3725	621	624	657	602	613	655	603	603	596	634	7153

k. Composite Matrix

17202	3011	3181	3051	3119	3164	3038	3092	3202	3120	3128	0	0
3011	16874	3060	3056	3062	2970	3083	3025	3015	3066	3148	0	0
3181	3060	22994	3752	3763	3633	3747	3725	3744	3640	3696	3081	3104
3051	3056	3752	22935	3804	3673	3749	3806	3775	3723	3633	2994	3105
3119	3062	3763	3804	23149	3736	3833	3805	3737	3633	3788	3100	3123
3164	2970	3633	3673	3736	22818	3684	3642	3663	3697	3680	3122	3098
3038	3083	3747	3749	3833	3684	23252	3812	3725	3750	3816	3145	3178
3092	3025	3725	3806	3805	3642	3812	23162	3763	3700	3804	3136	3096
3202	3015	3744	3775	3737	3663	3725	3763	22968	3756	3550	3105	3014
3120	3066	3640	3723	3633	3697	3750	3700	3756	23000	3823	3092	3103
3128	3148	3696	3633	3788	3680	3816	3804	3550	3823	22980	3081	3076
0	0	3081	2994	3100	3122	3145	3136	3105	3092	3081	17086	3133
0	0	3104	3105	3123	3098	3178	3096	3014	3103	3076	3133	17112

l. Adjustment factors

(1.232111 1.256061 0.921752 0.924123 0.91558 0.928862 0.911525 0.915066 0.922796 0.921512 0.922314 1.240476 1.238591)

2. Value Dependent Missingness

a. Alternate Variable Dependence

1.5	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0.75	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

b. Direct Variable Dependence

1.5	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0.8	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	3

3. Multifactorial Missingness

a. Variable Value Dependence Matrix

1.5	0	0	0	0	0	0	0	0	0	0	0	1.5
0	0	0	0	0	0	0	0	0	0	0	0	0

0	0	3	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.75	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

XIII. Relevant Files

A. TOH Data Warehouse Lab Hierarchy

1. Description

A listing of all test names for tests collected at the Ottawa Hospital with annotation indicating which tests were featured in the systematic review of PACNS diagnosis.

2. File

Brooks_John_2023_TOH-Data-Warehouse_Lab_hierarchy_Anotated.xlsx

XIV. Additional Simulations

In the following simulations the impacts of inference from the missingness pattern were reviewed by training the model given one missingness pattern (i.e., the missingness pattern presented in out ultimate simulation in the case of the first series of simulations) and replacing it with another (See Figure 59).

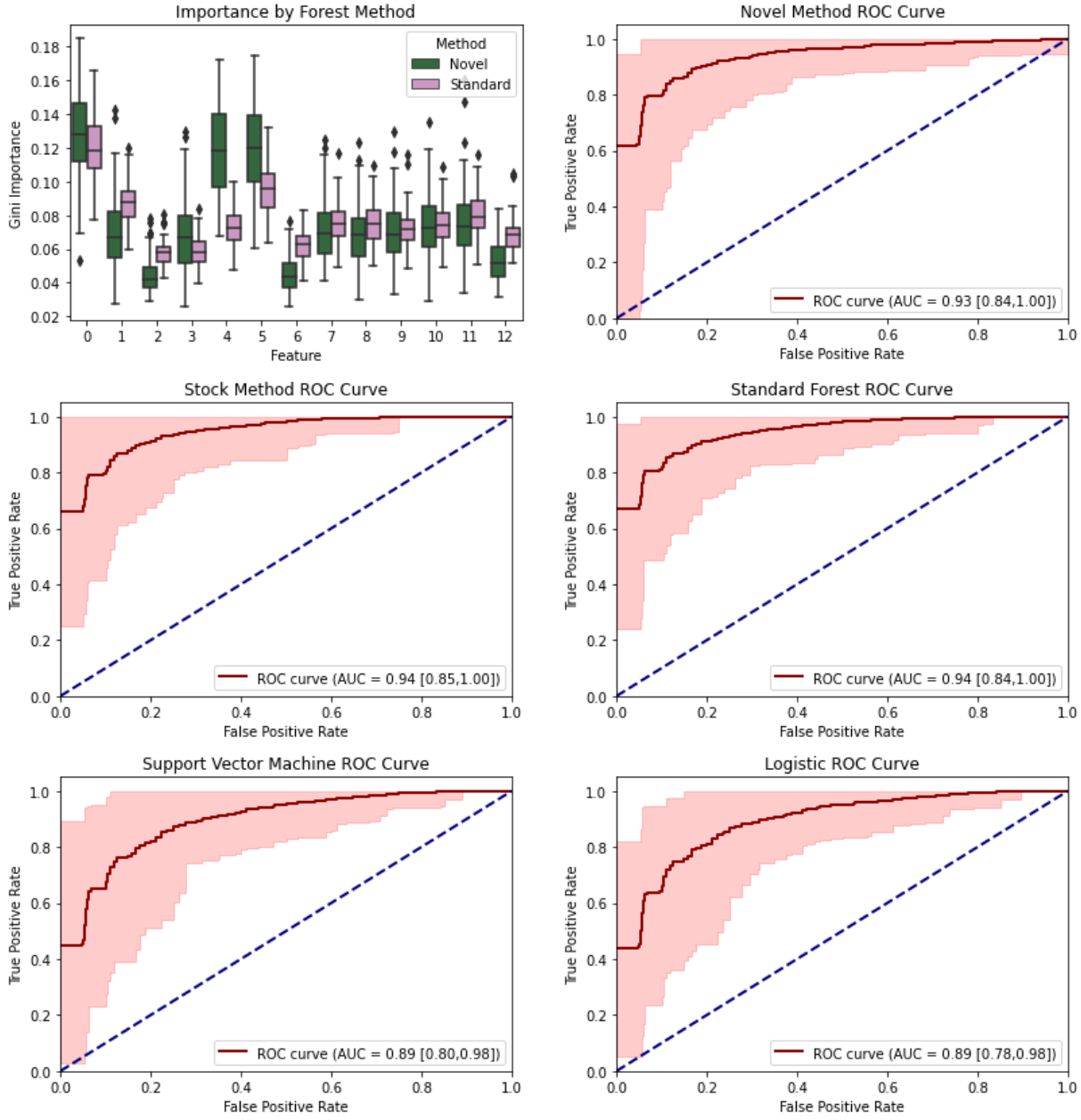


Figure 59. Simulation with complex missingness pattern and random missingness (50% missingness) at predict time. Random forests are composed of 1000 trees.

Forest models continued to be superior however the random forest models trained with mean value substitution appeared to be robust to this change in missingness pattern. If the percentage of missing values were increased, then the standard / stock approach continue to outperform the novel approach (see Figure 60).

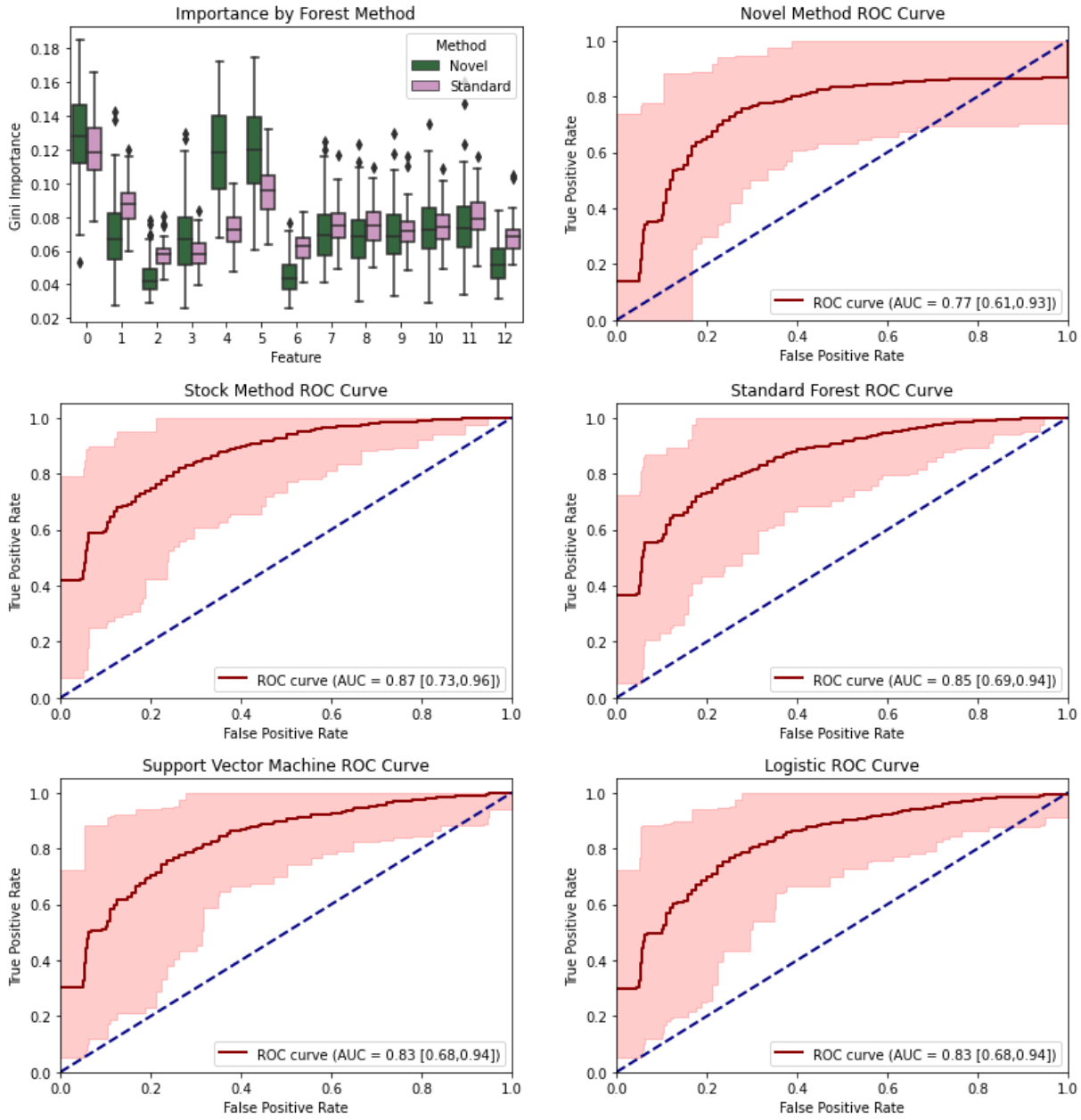


Figure 60. Simulation with complex missingness pattern and random missingness (75% missingness) at predict time. Random forests are composed of 1000 trees.

There is a notable drop in the ROC for the novel method that relates to default voting of the random forests. This could reflect the cumulative error created associated with the bias imposed by complete case analysis for each tree. One should recall that error is reduced in bagging by reducing variance (not bias) by aggregating the results of multiple assessments. The issue appeared to be related to the abstention voting system, however. Consider a model that was trained with missing data imputed by a feature mean value (Figure 61). Note that the performance curve for the novel random forest is nearly unchanged implying that training with tree wise complete data versus mean value imputation made little difference to simulation outcome.

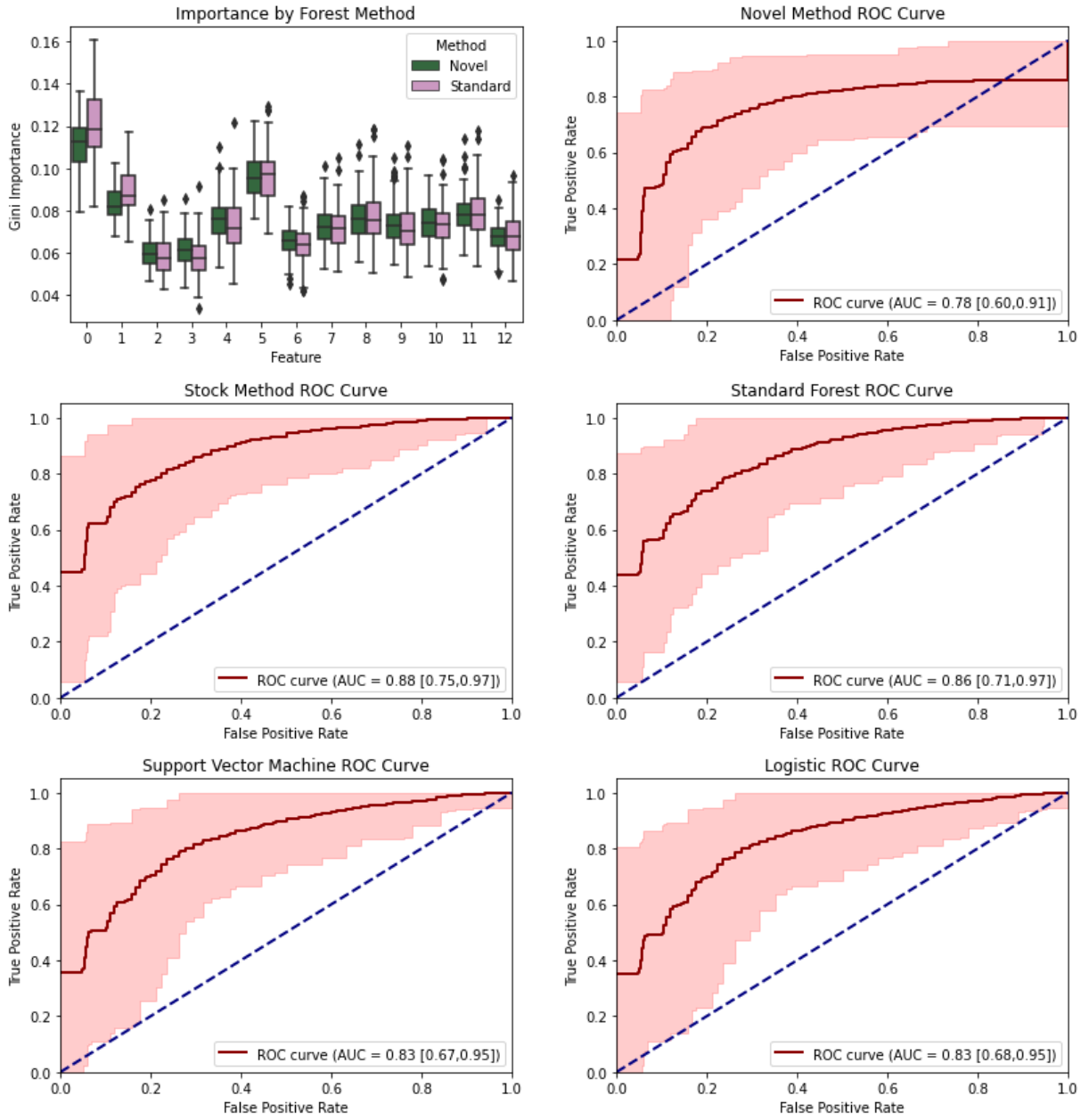


Figure 61. Simulation with complex missingness pattern and random missingness (75% missingness) at predict time. Random forests are composed of 1000 trees and the novel technique was trained with mean value imputation.

The number of trees associated with the random forest model appeared to make the greatest difference (see Figure 62 and Figure 63). This can be understood from the likelihood that a random forest with abstention voting may find itself without being able to make a vote (i.e., the forest may default to voting against the cases representing the diagnosis of interest because all trees must abstain). Consider that in a forest with 13 features and if one may select up to 3 features to compose a tree then there are 455 possible trees:

$$Possible\ trees = \frac{(Select + F - 1)}{(Select)(F-1)} = \frac{(\sqrt{13} + 13 - 1)}{(\sqrt{13})(13-1)} = \frac{(3 + 13 - 1)}{(3)(13-1)} = 455 \quad (54)$$

There are however 2,197 possible draws. Of those draws, 13 result in drawing on only one feature, 468 on two features and 1716 drawing on all different features:

$$Possible\ draws = F^3 = 13^3 = 2{,}197 \quad (55)$$

$$Draws\ 3\ Same = F = 13 \quad (56)$$

$$Draws\ 2\ Same = F = 13 \times 1 \times 12 + 13 \times 12 \times 2 = 468 \quad (57)$$

$$Draws\ Different = F = 13 \times 12 \times 11 = 1716 \quad (58)$$

Assuming no lock-outs occurred we can compute the probability a tree will abstain on average:

$$Probability\ of\ non-abstention\ \ with\ 50\%\ missing\ = \frac{13}{2197 \times 2^1} + \frac{468}{2197 \times 2^2} + \frac{1716}{2197 \times 2^3} = \frac{2}{13} = \frac{1}{6.5} > \frac{1}{2^3} = \frac{1}{8} \quad (59)$$

$$Probability\ of\ non-abstention\ \ with\ 75\%\ missing \sim \frac{1}{37} > \frac{1}{4^3} = \frac{1}{64} \quad (60)$$

In the case of 75% random missingness it is therefore much more likely to see substantial abstentions and thus the rseulf of a random forest may rely on far fewer trees than in the standard approach. This could bre addressed by increasing the number of trees but, in the case of of 75% random missingness, a substantial increase in the number of trees would be required to generate a comperable voting complement. For illustrative purposes consider a situation where there were fewer trees (see Figure 62):

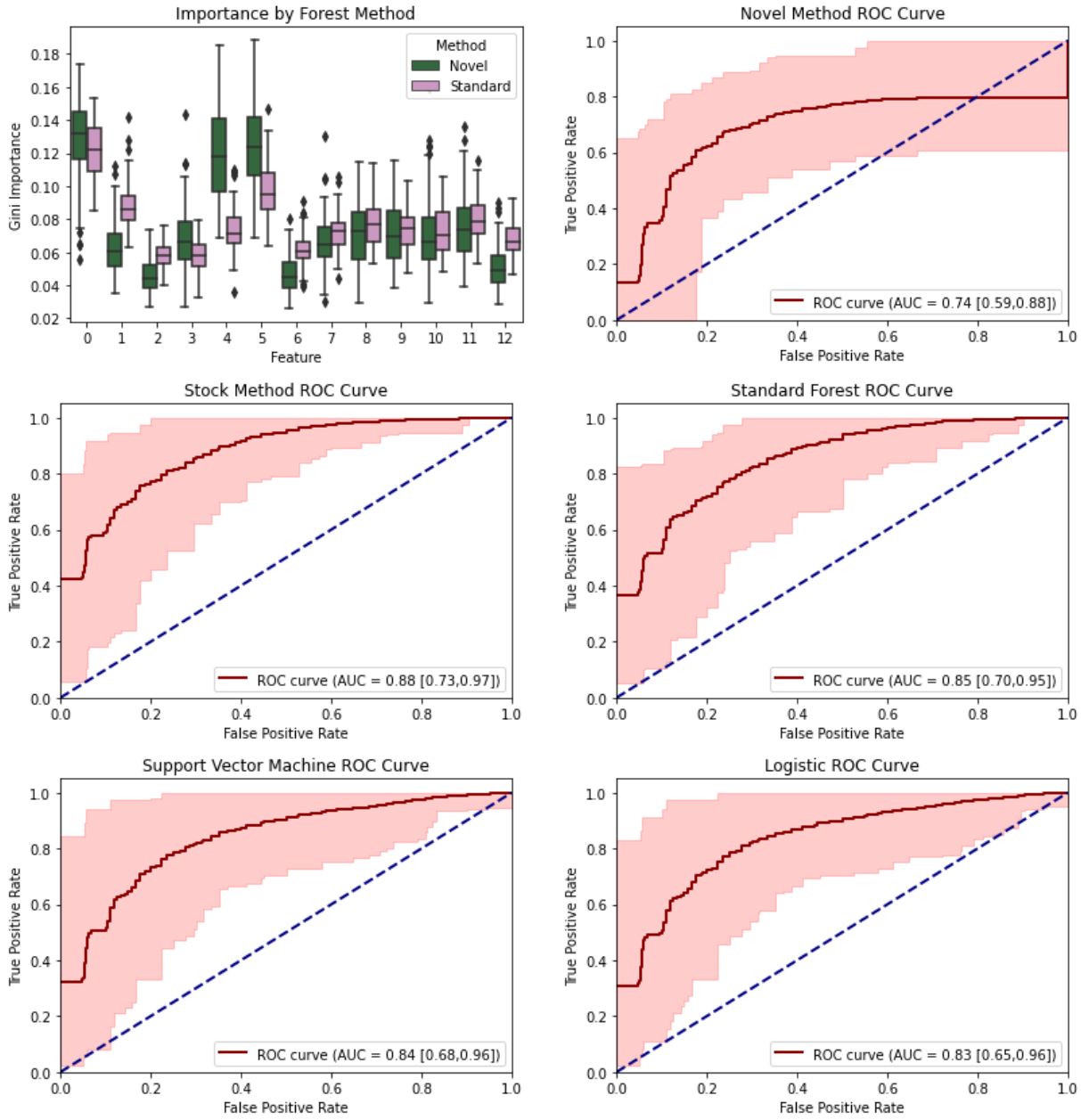


Figure 62. Simulation with complex missingness pattern and random missingness (75% missingness) at predict time. Random forests are composed of 500 trees.

Consider now the situation were there were more trees available for voting (see Figure 63):

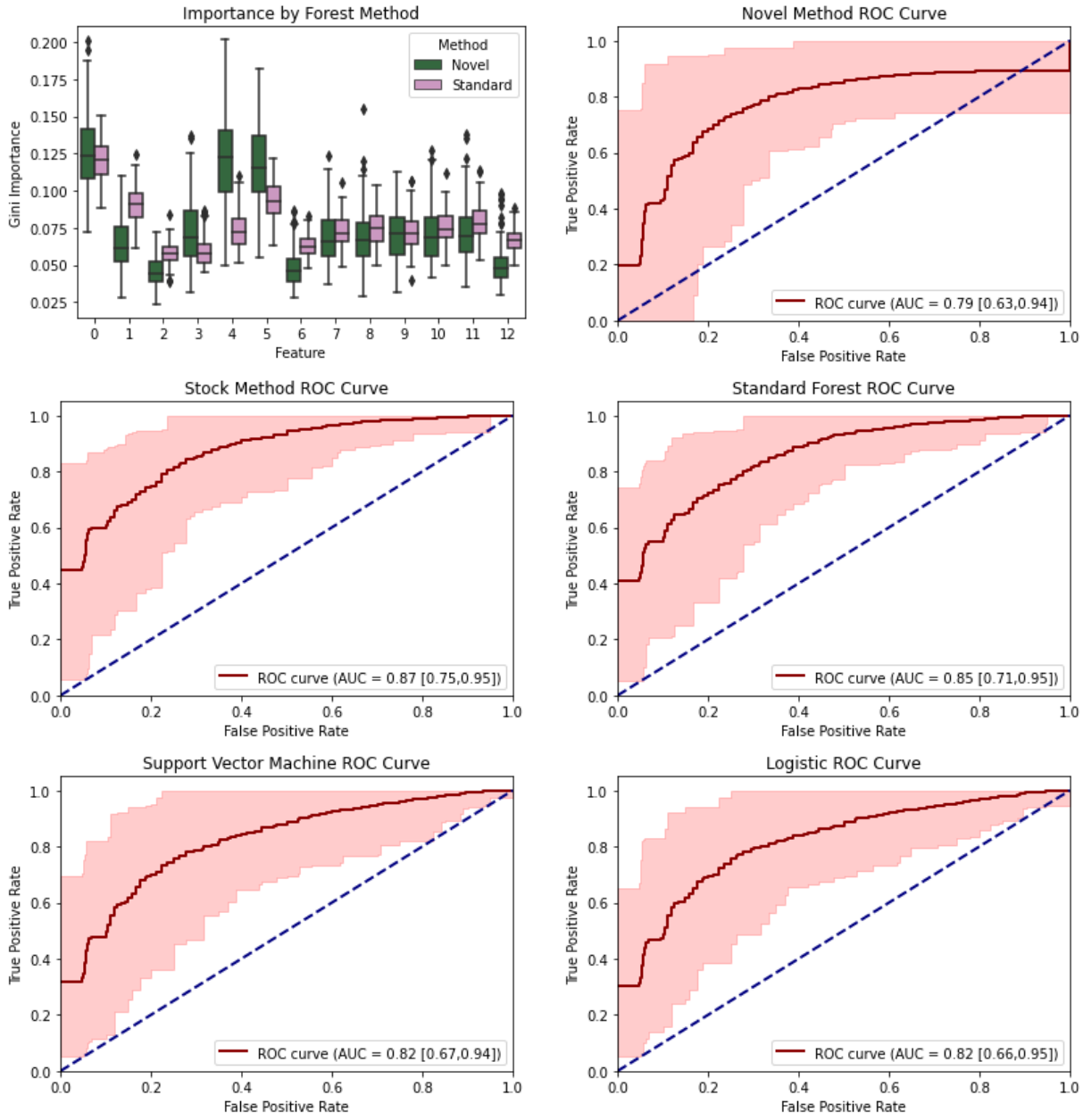


Figure 63. Simulation with complex missingness pattern and random missingness (75% missingness) at predict time. Random forests are composed of 2000 trees.

A significant improvement in the ROC is noted. Returning to the scenario where there was 50% random missingness the number of tree do not make a large difference (Figure 64):

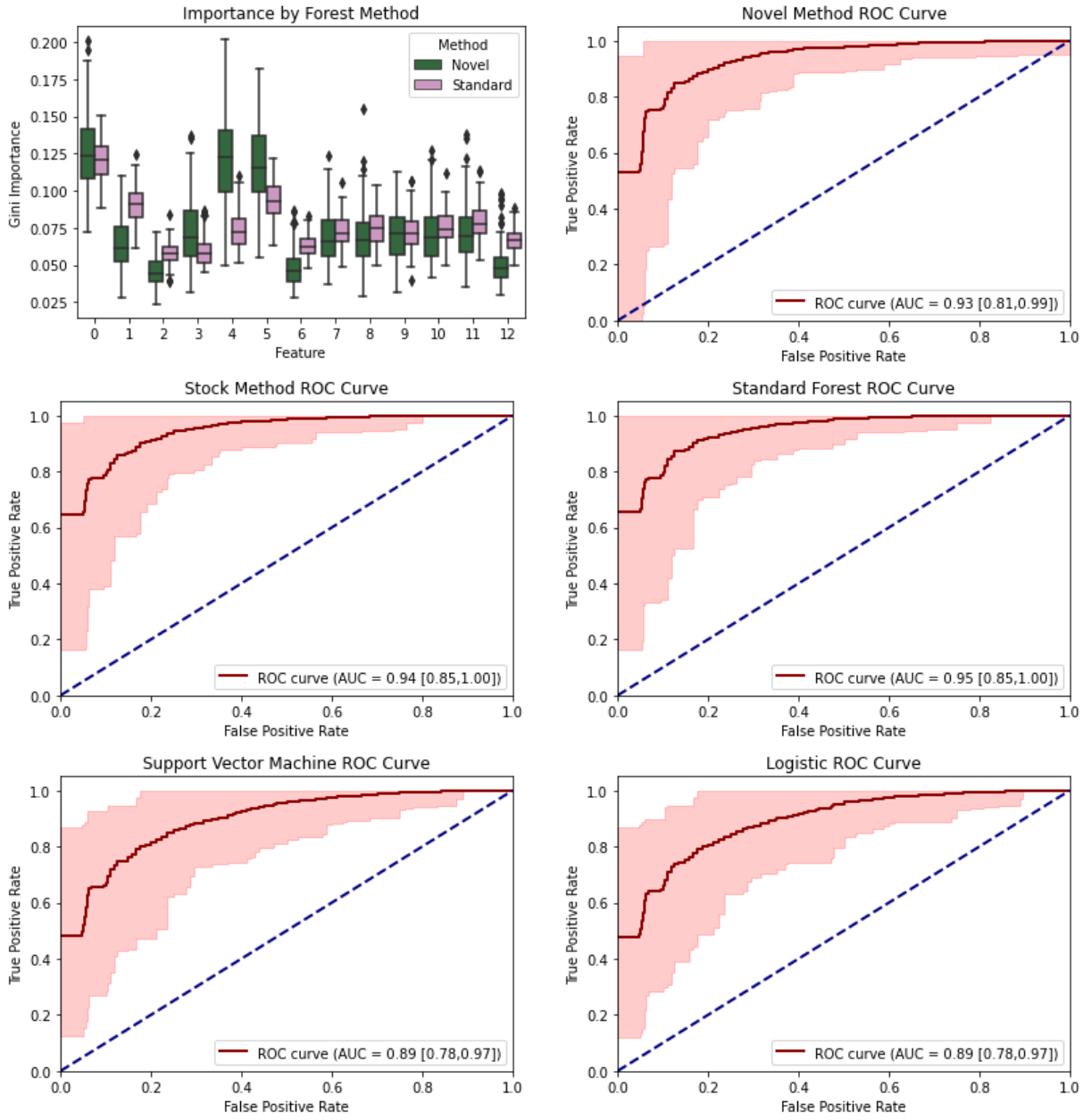


Figure 64. Simulation with complex missingness pattern and random missingness (50% missingness) at predict time. Random forests are composed of 2000 trees.

In all cases so far, the traditional random forest models outperformed however a problem existed where the standard techniques undervalued critical features. Considered was the situation where all features had missingness that was dependent on feature seven and that feature 7 was less randomized compared to the other features (25% versus 75%) (see Figure 65):

0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

$$V = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

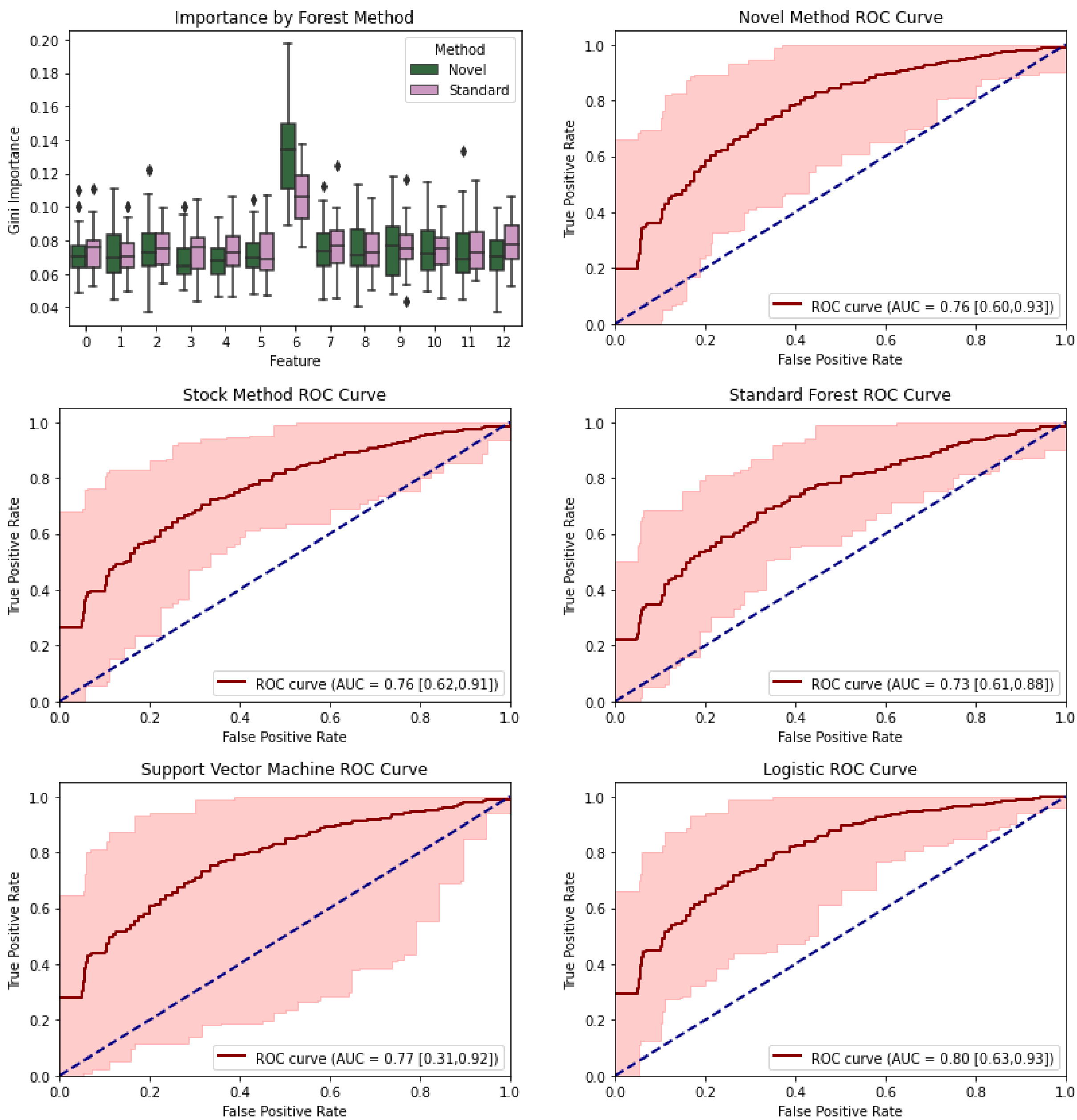


Figure 65. Extreme simulation where the missingness of every feature is dependent on the 7th feature with 50% random missingness at predict time.

Finally, if feature 7 was also possess of greater baseline missingness then the importance of the 7^{th} feature is suppressed in the standard approach and the model accuracy faltered (see Figure 66):

$$B = (1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1.5 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1)$$

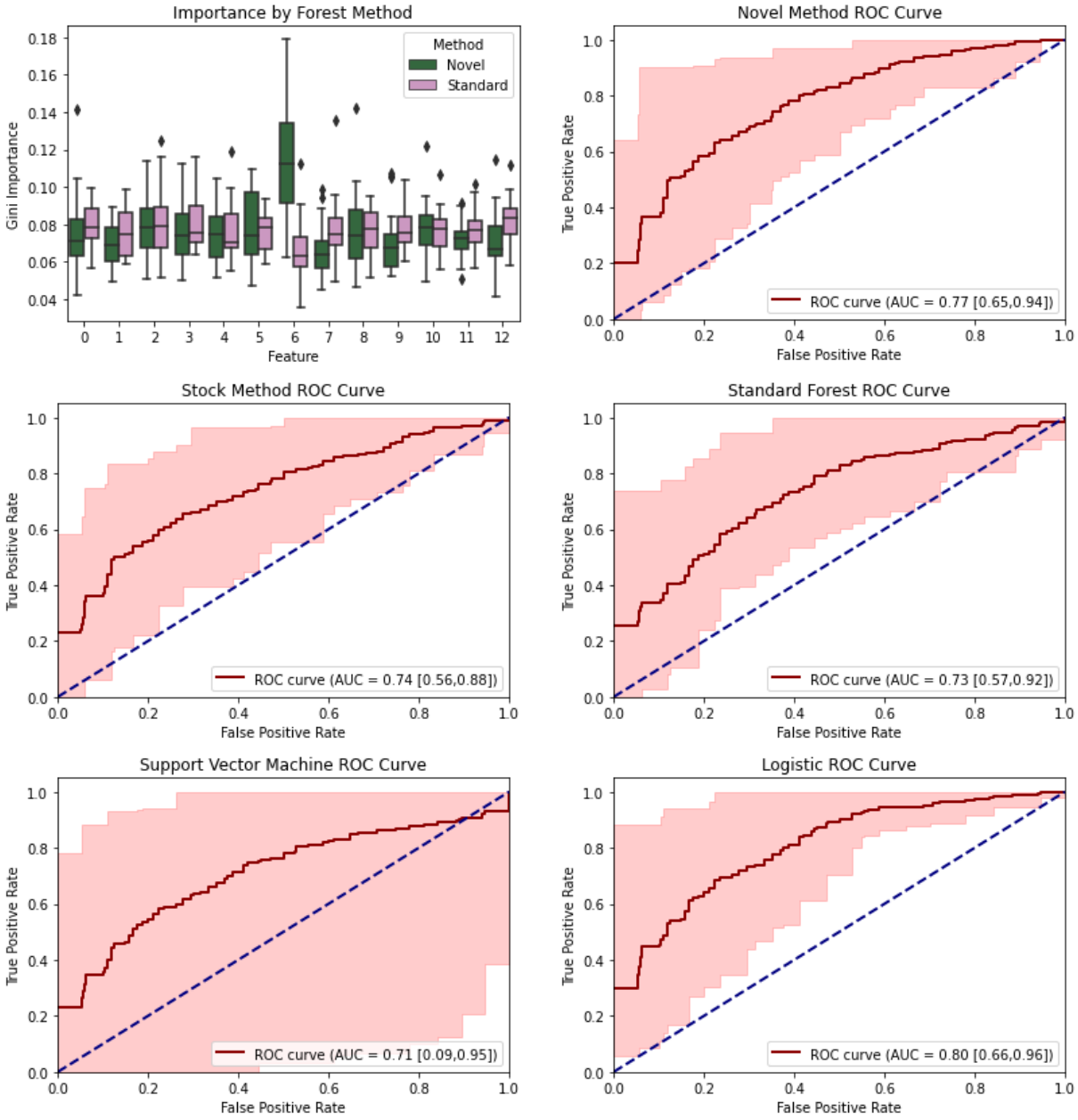


Figure 66. Extreme simulation where the missingness of every feature is dependent on the 7^{th} feature and greater baseline missingness of the 7^{th} feature with 50% random missingness at predict time.

XV. Code

```
# -*- coding: utf-8 -*-
"""
Random Forest Vesus Support Vector Machines
John Brooks MD FRCPC
"""

################################################################################
"""
References:
   https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python
   https://www.askpython.com/python/examples/impute-missing-data-
values#:~:text=%20Impute%20missing%20data%20values%20by%20MEAN%20,the%20values%20have%20been%20imputed%20or...%20More%20
   https://realpython.com/knn-python/
   https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html
   https://medium.com/@kyawsawhtoon/a-guide-to-knn-imputation-95e2dc496e
   https://datagy.io/pandas-conditional-column/
   https://stackoverflow.com/questions/35530640/pandas-use-value-if-not-null-else-use-value-from-next-column
   https://machinelearningmastery.com/calculate-feature-importance-with-python/
"""
################################################################################

################################################################################
"""
Declarations
"""
################################################################################

# Initialize
## Datat reaad in
import pandas as pd
import numpy as np
from math import floor

## Imputation machinery
from sklearn.impute import KNNImputer

## Learning algorhthym
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.metrics import roc_auc_score

# from itertools import compress
import matplotlib.pyplot as plt
import seaborn as sns

## For visualization
### depricateded: from sklearn.externals.six import StringIO
from six import StringIO
from IPython.display import Image
import pydotplus
from sklearn.tree import export_graphviz

## Alternates
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from sklearn.linear_model import LogisticRegression

## Play data
from sklearn.datasets import load_wine
import random

## Other tools
import statistics
from scipy.cluster import hierarchy
from sklearn.utils import shuffle
from sklearn.inspection import permutation_importance
from collections import Counter

## Display tools
# import time
import progressbar

################################################################################
"""
Global Variables
"""
################################################################################

# Set initial random seed
randomMasterInitial = 5
randomMaster = randomMasterInitial

################################################################################
```

```
"""
Selections
"""
################################################################################

# Data For Processing
################################################################################

dataChoices = 'fresh'

if dataChoices == 'fresh':
   dataSetSelect = 'generate'
   dataSetSelectForCurves = 'fabricate'
   bootstrap = False

elif dataChoices == 'real':
   dataSetSelect = 'real'
   dataSetSelectForCurves = 'data through'
   bootstrap = False

elif dataChoices == 'import':
   dataSetSelect = 'real'
   dataSetSelectForCurves = 'data augment'
   bootstrap = True

elif dataChoices == 'wine':
   dataSetSelect = 'load'

   trimData = True
   keepInSet = ['alcohol', 'alcalinity_of_ash', 'total_phenols', 'hue', 'proline']

   dataSetSelectForCurves = 'data augment'
   bootstrap = True

else:
   # Select data real, load, or generate
   dataSetSelect = 'real'

   # Select 'fabricate' or 'data augment' or 'data through'
   dataSetSelectForCurves = 'data through'

   # To fix a data alteratiopn pattern and only generate curves by selecting cases with replacement
   bootstrap = False

# Function Select
################################################################################
singleIter = False
runNewApproach = False
printOut = True
permuteImportGo = False
permuteImport = 'Custom'
permutationIterationsSelected = 100

runMainCurves = True
runExperimentSelect = False
runDifferentPredictMissingness = True
sparseDefiniteProportion = 0.5
runTrainWithFill = False

# Run plots / Plot Variables
################################################################################
runPlots = True
selectLowCentile = 0.025
selectHighCentile = 0.975
includeStockImportance = False

# Experiment Variables
################################################################################
experimentalVariableSelect = 2
engagePseudorandomSelect = False
perfectSplitSelect = True

# Empty list (i.e., "[]") will prompt split at median
centileSelect = []

# Tree / Forest Specifications
################################################################################
# When splitting at a node in a decision tree the resultant leaves must have at least 2 cases
desiredTreeNumberSelect = 1000
desiredIterationsSelected = 25
minimumNumberPerLeaf = 2

# When incorporating importances for trees in a forest together
## Options are 'mean' or 'raw score'
treeImportanceCombination = 'raw score'

# When incorporating importances for trees in a forest together are the importances weighted by the number of cases they aimed to seperate
## Options are 'Avg' or 'Raw'
```

```
importanceAdjustment = 'Raw'

# Create importance adjustment
adjustmentFlag = 'First'

# Data Augment specifications
################################################################################
# Pattern selections
randomPattern = 'extreme'
selectPattern = 'extreme'
# shuffleBeforeMissing = True

# Options mean, median, 3NN
fillMethodSelect = 'mean'

# Randomness
################################################################################
if randomPattern == 'even':
   shuffleSelection = 0.5
   shuffleType = 'none'
elif randomPattern == 'extreme':
   shuffleSelection = 0.5
   shuffleType = 'extreme'
elif randomPattern == 'complex':
   shuffleSelection = 0.5
   shuffleType = 'complex'
elif randomPattern == 'none':
   shuffleSelection = 0
   shuffleType = 'none'
elif randomPattern == 'specific':
   shuffleSelection = 0.5
   shuffleType = 'pattern1'
elif randomPattern == 'single one':
   shuffleSelection = 0.5
   shuffleType = 'single one'
else:
   shuffleSelection = 0
   shuffleType = 'none'

# Missingness
################################################################################
paramArray = pd.DataFrame()

if selectPattern == 'even':
   sparsitySelection = 0.5
   probabilityBaseVector = 'none'
   valueMatrixApply = 'none'
   missingCorrelationMatrixApply = 'none'
elif selectPattern == 'extreme':
   sparsitySelection = 0.5
   probabilityBaseVector = 'extreme'
   valueMatrixApply = 'extreme'
   missingCorrelationMatrixApply = 'none'
elif selectPattern == 'complex':
   sparsitySelection = 0.5
   probabilityBaseVector = 'balenced'
   valueMatrixApply = 'complex'
   missingCorrelationMatrixApply = 'complex'
elif selectPattern == 'correlated':
   sparsitySelection = 0.5
   probabilityBaseVector = 'balenced'
   valueMatrixApply = 'none'
   missingCorrelationMatrixApply = 'complex'
elif selectPattern == 'value':
   sparsitySelection = 0.5
   probabilityBaseVector = 'none'
   valueMatrixApply = 'complex'
   missingCorrelationMatrixApply = 'none'
elif selectPattern == 'own value':
   sparsitySelection = 0.5
   probabilityBaseVector = 'none'
   valueMatrixApply = 'own'
   missingCorrelationMatrixApply = 'none'
elif selectPattern == 'off value':
   sparsitySelection = 0.5
   probabilityBaseVector = 'none'
   valueMatrixApply = 'off'
   missingCorrelationMatrixApply = 'none'
elif selectPattern == 'no alterations':
   sparsitySelection = 0
   probabilityBaseVector = 'none'
   valueMatrixApply = 'none'
   missingCorrelationMatrixApply = 'none'
else:
   sparsitySelection = 0
   probabilityBaseVector = 'none'
   valueMatrixApply = 'none'
   missingCorrelationMatrixApply = 'none'
```

```
##############################################################################
"""
Selection Interpretations
"""
##############################################################################

# Select shuffle list
if shuffleType == 'complex':
    # If the shuffle list is empty then the shuffle select will dominate
    shuffleList = [0.25, 0.5, 0.75, 0.5, 0.25, 0.25, 0.75, 0.5, 0.5, 0.5, 0.5, 0.5, 0.75, 0, 0]
elif shuffleType == 'extreme':
    shuffleList = [0.75, 0.75, 0.75, 0.75, 0.75, 0.75, 0.25, 0.75, 0.75, 0.75, 0.75, 0.75, 0.75, 0, 0]
elif shuffleType == 'pattern1':
    shuffleList = [0.5, 0.5, 0.5, 0.75, 0.25, 0.25, 0.75, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0, 0]
elif shuffleType == 'single one':
    shuffleList = [0.5, 0.25, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.25, 0.5, 0.5, 0.5, 0, 0]
else:
    shuffleList = []

# Select probability base matrix
if probabilityBaseVector == 'balenced':
    dpb = pd.DataFrame([5,5,1.25,1.25,1.25,1,1,1,1,1,1,5,5,0,0]).T
elif probabilityBaseVector == 'diametric':
    dpb = pd.DataFrame([5,5,1,1,1,1,1,1,1,1,1,5,5,0,0]).T
elif probabilityBaseVector == 'subtle':
    dpb = pd.DataFrame([1.25,1.25,1,1,1,1,1,1,1,1,1,1,1.25,0,0]).T
elif probabilityBaseVector == 'none':
    dpb = pd.DataFrame([1,1,1,1,1,1,1,1,1,1,1,1,1,0,0]).T
elif probabilityBaseVector == 'extreme':
    dpb = pd.DataFrame([1,1,1,1,1,1,1.5,1,1,1,1,1,1,0,0]).T
else:
    dpb = pd.DataFrame([1,1,1,1,1,1,1,1,1,1,1,1,1,0,0]).T

valueInCorrelationMatrix = 'x'

if valueMatrixApply == 'complex':
    valueInCorrelationMatrix = [[1.5,0,0,0,0,0,0,0,0,0.0,0,0,0],
                                [0.0,0,0,0,0,0,0,0,0,0.0,0,0,0],
                                [0.0,0,3,0,0,0,0,0,0,0.0,0,0,0],
                                [0.0,0,0,0,0,0,0,0,0,0.0,0,0,0],
                                [0.0,0,0,0,0,0,0,0,0,0.0,0,0,0],
                                [0.0,0,0,0,0,0,0,0,0,0.0,0,0,3],
                                [0.0,0,0,0,0,0,0,0,0,0.8,0,0,0],
                                [0.0,0,0,0,0,0,0,0,1,0.0,0,0,0],
                                [0.0,0,0,0,0,0,0,0,0,0.0,0,0,0],
                                [0.0,0,0,0,0,0,0,0,0,0.0,0,0,0],
                                [0.0,0,0,0,0,0,0,0,0,0.0,0,0,0],
                                [0.0,0,0,0,0,0,0,0,0,0.0,0,0,0],
                                [0.0,0,0,0,0,0,0,0,0,0.0,0,0,0]]
elif valueMatrixApply == 'own':
    valueInCorrelationMatrix = [[9,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,15,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0]]
elif valueMatrixApply == 'off':
    valueInCorrelationMatrix = [[0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [9,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,15,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0]]
elif valueMatrixApply == 'extreme':
    valueInCorrelationMatrix = [[0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [1,1,1,1,1,1,1,1,1,1,1,1,1],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0],
```

```
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0]]
else:
    ## Note that the last column does not matter as class labels must be present
    ## Note that the ith column maps to the missingness of the ith factor for a given case
    valueInCorrelationMatrix = [[0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0]]

# Pad zeros
paddingVariables = 2
valueInCorrelationMatrix = np.array(valueInCorrelationMatrix)
valueInCorrelationMatrix = np.c_[valueInCorrelationMatrix, np.zeros([len(valueInCorrelationMatrix),paddingVariables])]
valueInCorrelationMatrix = np.r_[valueInCorrelationMatrix, np.zeros([paddingVariables,len(valueInCorrelationMatrix)+paddingVariables])].tolist()

# create missingness dependency matricies
################################################################################
# Generates two populations with data mostly missing for two entries

if missingCorrelationMatrixApply == 'complex':
    missingCorrelationMatrix = [[0,20,0,0,0,0,0,0,0,0,0,0,-20,-20],
                              [20,0,0,0,0,0,0,0,0,0,0,0,-20,-20],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [-20,-20,0,0,0,0,0,0,0,0,0,0,0,30],
                              [-20,-20,0,0,0,0,0,0,0,0,0,0,30,0]]
else:
    missingCorrelationMatrix = [[0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0,0,0,0,0,0]]

# Pad zeros
missingCorrelationMatrix = np.array(missingCorrelationMatrix)
missingCorrelationMatrix = np.c_[missingCorrelationMatrix, np.zeros([len(missingCorrelationMatrix),paddingVariables])]
missingCorrelationMatrix = np.r_[missingCorrelationMatrix, np.zeros([paddingVariables,len(missingCorrelationMatrix)+paddingVariables])].tolist()

################################################################################
"""
Constructs
"""
################################################################################

# Create progress bar pattern
progressBars = [' [',
        progressbar.Timer(format= 'elapsed time: %(elapsed)s'),
        '] ',
        progressbar.Bar('*'),' (',
        progressbar.ETA(), ') ',
        progressbar.Percentage(),
        ]

################################################################################
"""
Create Visualization Functions
"""
################################################################################

# Create a plot of a decision tree
def plotDecisionTree(clfN, cols, plotName):
```

```
    dot_data = StringIO()
    export_graphviz(clfN, out_file=dot_data,
                    filled=True, rounded=True,
                    special_characters=True, feature_names = cols,class_names=['0','1'])
    graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
    graph.write_png(plotName)
    Image(graph.create_png())

# Create a plot of an ROC
def plotROCTree(fpr, tpr, rocFinal):
    # ROCs
    # Note: Can make receiver-operator curves for each class if more than one
    plt.figure()
    lw = 2

    ### Draw on plot
    plt.plot(
        fpr,
        tpr,
        color="darkred",
        lw=lw,
        label="ROC curve (AUC = %0.2f)" % rocFinal,
    )
    plt.plot([0, 1], [0, 1], color="navy", lw=lw, linestyle="--")

    ### Set up plot
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title("Receiver Operating Characteristic")
    plt.legend(loc="lower right")
    plt.show()

# Create a plot of an ROC with multiple curves
def plotROCTreeFill(statsInFrame, rocFinal, plotNameIn='Receiver Operating Characteristic'):

    plt.figure()
    lw = 2

    # Label string
    labelIng = "ROC curve (AUC = {:0.2f} [{:0.2f},{:0.2f}])".format(rocFinal[0], rocFinal[1], rocFinal[2])

    # Draw on plot
    plt.plot(
        statsInFrame['falsePositiveRate'],
        statsInFrame['mean'],
        color="darkred",
        lw=lw,
        label=labelIng,
    )
    plt.plot([0, 1], [0, 1], color="navy", lw=lw, linestyle="--")

    # Create a filled region in the plot
    plt.fill_between(statsInFrame['falsePositiveRate'],
                     statsInFrame['LowCentile'],
                     statsInFrame['HighCentile'],
                     facecolor="pink", # The fill color
                     color='red',      # The outline color
                     alpha=0.2)        # Transparency of the fill

    # Set up plot
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title(plotNameIn)
    plt.legend(loc="lower right")
    plt.show()

################################################################################
"""
Data Generation Functions
"""
################################################################################

# To create a definite set we will create inter-relationships between variables at random
# | a1 b1 c1 | |x| = |r|
# | a1 b1 c1 |*|y| = |r|
# | a1 b1 c1 | |z| = |r|
# r will be slected at random and then the system of equations will allow for back solving

def creatArtificalFrame (numberColumnsDesired, numberRowsDesired, thresholdSelect = 0.5, paraMeterFrameIn = pd.DataFrame(), randomSeed = 12345):

    # Instantiate random number genereator
    rng = np.random.default_rng(randomSeed)
    randomSeed += 1
```

```
    if ((paraMeterFrameIn.shape[0] == 0) or (paraMeterFrameIn.shape[0] != paraMeterFrameIn.shape[1])):

        # randomly create a parameter matrix
        parameterArray = rng.uniform(low=0.0,
                                     high=1.0,
                                     size=numberColumnsDesired**2).reshape(numberColumnsDesired,
                                                                           numberColumnsDesired)

    else:

        # Reform array
        parameterArray = paraMeterFrameIn.to_numpy()

    # Find the inverse matrix
    inverseParameters = np.linalg.inv(parameterArray)

    # Create a set of probabilities for a catagorization
    solutionsRandom = rng.uniform(low=0.0, high=1.0, size=numberRowsDesired)

    # Initiate a solution bearer
    solutionsList = np.array([np.nan]*numberColumnsDesired)

    # Iteratively solve the system of equations to create a matrix of appropriate values
    for currentSolution in solutionsRandom:
        solutionVector = np.array([currentSolution]*numberColumnsDesired).reshape(numberColumnsDesired,1)
        currentRow = np.dot(inverseParameters, solutionVector).reshape(1,numberColumnsDesired)
        solutionsList = np.vstack((solutionsList, currentRow))

    # Shape the resultant frame
    solutionsList = np.delete(solutionsList, 0, axis=0)
    catagoricalVector = solutionsRandom.reshape(numberRowsDesired,1) > thresholdSelect
    solutionsList = np.hstack((solutionsList, catagoricalVector))
    frameReturn = pd.DataFrame(solutionsList)

    return frameReturn, pd.DataFrame(parameterArray), randomSeed

# Create data with specified characteristics
def dataCreationLine (numFeatures = 13, numSamples = 178, paramArrayFunct = pd.DataFrame(), targetIn = [], randomMasterFunct = 0):

    # Create the data for the matrix
    deFunct, paramArrayFunct, randomMasterFunct = creatArtificalFrame(numFeatures,
                                                  numSamples,
                                                  paraMeterFrameIn = paramArrayFunct,
                                                  randomSeed = randomMasterFunct)

    # Generate characteristics to emulate real input data
    ## Rename matrix columns
    deFunct.rename(columns = {(deFunct.shape[1] - 1): targetIn[0]}, inplace = True)
    newVasc = []

    ## Provide names to the binary values being explored
    for i in deFunct[targetIn[0]]==0:
        if i:
            newVasc.append('Primary')
        else:
            newVasc.append('Not')
    deFunct[target[0]] = newVasc

    ## Place MRN
    deFunct['MRN'] = 0

    ## Provide a list of categorical entries
    cat_entitiesFunct = []

    return deFunct, paramArrayFunct, cat_entitiesFunct, randomMasterFunct

# Bin a variable
def binAVariable (dataFrameIn, variableSelected = 0, partitionThresholds = [], typeThresholds = 'centiles', numberPartitions = 2, engagePseudorandom = engagePseudorandomSelect,
randomMasterFunct = randomMaster):

    # Initialize
    dataFrameOut = dataFrameIn.copy()
    columnData = dataFrameIn.iloc[:,variableSelected]

    # Create thresholds
    if len(partitionThresholds) == 0:
        partitionThresholdsUse = np.divide([*range(0, numberPartitions, 1)], numberPartitions).tolist()
        quantS = columnData.quantile(partitionThresholdsUse).tolist()
    else:
        partitionThresholdsUse = partitionThresholds.copy()
        if (typeThresholds == 'centiles') and ((min(partitionThresholds) >= 0) and (max(partitionThresholds) <= 1)):
            if not (0 in partitionThresholds):
                partitionThresholdsUse.append(0)

            quantS = columnData.quantile(partitionThresholdsUse).tolist()
        else:
            minIn = min(columnData)
```

```
            if not (minIn in partitionThresholds):
                partitionThresholdsUse.append(minIn)

            quantS = partitionThresholdsUse

        # Sort quantiles
        quantS.sort()

        # Adjust column
        columnData = np.array(columnData)

        # New values
        columnOut = np.array([0]*len(columnData))

        # reset values
        loopIndex = 0
        for currentThresholdLoop in quantS:
            columnOut[(columnData >= currentThresholdLoop).tolist()] = loopIndex
            loopIndex += 1

        # Adjust values if appropriate
        if engagePseudorandom:
            randomNumberGenerator=np.random.RandomState(randomMasterFunct)
            randomSelection = randomNumberGenerator.uniform(low=0, high=0.5, size=len(columnData))
            columnOut = columnOut + randomSelection
            randomMasterFunct += 1

        dataFrameOut.iloc[:,variableSelected] = columnOut

        return dataFrameOut, quantS, randomMasterFunct

    ###############################################################################
    """
    Data Preparation
    """
    ###############################################################################

    # Prepare data
    ## Set standards
    ### Main outcome column
    target  = ['Vasculitis']

    ###############################################################################
    ###############################################################################

    ## Target selector
    if(dataSetSelect == 'real'):

        ## Set codeword for main grouping of interest / stratifying variable
        codeword = ['PACNS']
        preserveWords = ['Diseases of the circulatory system',
                         'Certain infectious and parasitic diseases']

        ## Import data and store original data
        dr = pd.read_excel (r'C:\Users\johnb\OneDrive\Desktop\vasculitisData02.xlsx')
        colsRead = dr.columns.tolist()
        keepList = ['Diagnosis','MRN','Sex',
                    'ESR-Result (numeric)',
                    'Vasculitis Dx-Diagnosis',
                    'Urine Protein-Result (numeric)','Urine Protein-Result (textual)',
                    'Paraneoplastic-Result (textual)',
                    'Cardiolipin IgG-Result (numeric)',
                    'Cardiolipin IgM-Result (numeric)',
                    'Beta-2-Glycoprotein IgG-Result (numeric)',
                    'Beta-2-Glycoprotein IgM-Result (numeric)',
                    'Compliment C3-Result (numeric)',
                    'Compliment C4-Result (numeric)',
                    'CSF Protein-Result (numeric)','CSF WBC-Result (numeric)',
                    'CRP-Result (numeric)','ANA Titer-Result (numeric)',
                    'Homocysteine-Result (numeric)','Protein C-Result (numeric)',
                    'Protein S-Result (numeric)','p-ANCA-Result (textual)',
                    'c-ANCA-Result (textual)','ANCA pattern 1-Result (textual)',
                    'ANCA pattern 2-Result (textual)']

        for val in keepList: colsRead.remove(val)
        dr = dr.drop(colsRead, inplace=False, axis=1)

        keepRowsIn = dr['Vasculitis Dx-Diagnosis'] == ''
        for pWordIn in preserveWords: keepRowsIn = keepRowsIn | (dr['Vasculitis Dx-Diagnosis']==pWordIn)

        df = dr.copy()
        dr = dr.drop(dr[~keepRowsIn].index)

        # Drop all exempted diagnoses
        dr = dr.drop(dr[dr["Diagnosis"].str.startswith("*")].index)

        ## Preprocess MDClone data
        ### Preprocess Diagnosis
```

```
    dxColumnName = ['Diagnosis']

    #### Create new column that gives the target classes
    conditions = [
        dr[dxColumnName] != codeword,
        dr[dxColumnName] == codeword
    ]

    values = ['Not', 'Primary']

    dataSR = dr.copy()
    dataSR[target] = np.select(conditions, values)

    ### Select only particular rows to review
    # dataSR = dr[dr['Include Flag 1'] == True]
    # dataSR.index = range(len(dataSR))

    dataSR.index = range(len(dataSR))

    # Merge columns with before and after results if applicable
    # de = dataSR.assign(ESR=np.where(np.isnan(dataSR['ESR Before-Result (numeric)']),
    #                     dataSR['ESR After-Result (numeric)'],
    #                     dataSR['ESR Before-Result (numeric)']))

    ### Select columns
    de = pd.DataFrame(dataSR, columns= ['MRN',
                                'Sex',
                                'ESR-Result (numeric)',
                                'Urine Protein-Result (numeric)','Urine Protein-Result (textual)',
                                'Paraneoplastic-Result (textual)',
                                'Cardiolipin IgG-Result (numeric)',
                                'Cardiolipin IgM-Result (numeric)',
                                'Beta-2-Glycoprotein IgG-Result (numeric)',
                                'Beta-2-Glycoprotein IgM-Result (numeric)',
                                'Compliment C3-Result (numeric)',
                                'Compliment C4-Result (numeric)',
                                'CSF Protein-Result (numeric)','CSF WBC-Result (numeric)',
                                'CRP-Result (numeric)','ANA Titer-Result (numeric)',
                                'Homocysteine-Result (numeric)','Protein C-Result (numeric)',
                                'Protein S-Result (numeric)','p-ANCA-Result (textual)',
                                'c-ANCA-Result (textual)','ANCA pattern 1-Result (textual)',
                                'ANCA pattern 2-Result (textual)',
                                'Vasculitis'])

    cat_entities = ['Sex','Urine Protein-Result (textual)','Paraneoplastic-Result (textual)',
                'p-ANCA-Result (textual)',
                'c-ANCA-Result (textual)','ANCA pattern 1-Result (textual)',
                'ANCA pattern 2-Result (textual)']

    # Factorize
    for catEntry in cat_entities:

        # Start with factorization
        de[catEntry] = pd.factorize(de[catEntry])[0]

    # Reveal Sparsities
    if printOut:
        print('Sparsitiy \n')
        sparsitiesComp = de.isna().sum() / de.shape[0]
        sparsitiesCompList = sparsitiesComp.tolist()
        sparsitiesCompList.pop(0)
        sparsitiesCompList.pop()
        print(sparsitiesComp)
        aveSparsityInput = sum(sparsitiesCompList)/len(sparsitiesCompList)
        print("Total Input Sparsity:", "%.2f" % aveSparsityInput)

elif(dataSetSelect == 'load'):

    ## Load wine data
    de = load_wine(as_frame=True).frame

    ## Remap the wine data to emulate real data characteristics
    de.rename(columns = {'target': target[0]}, inplace = True)
    newVasc = []
    for i in de[target[0]]==0:
        if i:
            newVasc.append('Primary')
        else:
            newVasc.append('Not')
    de[target[0]] = newVasc
    de['MRN'] = 0
    cat_entities = []

    if trimData == True:
        allInSet = ['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium','total_phenols', 'flavanoids', 'nonflavanoid_phenols','proanthocyanins', 'color_intensity',
'hue','od280/od315_of_diluted_wines', 'proline']
        de = de.drop(list(set(allInSet)-set(keepInSet)), inplace=False, axis=1)
```

```
elif(dataSetSelect == 'generate'):

   ## Generate clean data
   de, paramArray, cat_entities, randomMaster = dataCreationLine(13, 178, targetIn = target, randomMasterFunct = randomMaster)

elif(dataSetSelect == 'sim'):

   ## Import data and store original data
   de = pd.read_excel (r'C:\Users\johnb\OneDrive\Desktop\VasculitisSim.xlsx')
   cat_entities = ['Infection']

elif(dataSetSelect == 'sim_no_abs'):

   ## Import data and store original data without an absolute predictor column
   de = pd.read_excel (r'C:\Users\johnb\OneDrive\Desktop\VasculitisSim.xlsx')
   de = de.drop('Abs', inplace=False, axis=1)
   cat_entities = ['Infection']

elif(dataSetSelect == 'real_data'):

   ## Import data and store original data
   de = pd.read_excel (r'C:\Users\johnb\OneDrive\Desktop\vasculitisData.xlsx')
   colsRead = de.columns.tolist()
   keepList = ['Diagnosis','MRN','Sex','ESR Before-Result (numeric)','ESR After-Result (numeric)',
            'Vasculitis Dx-Diagnosis',
            'Urine Protein-Result (numeric)','Urine Protein-Result (textual)',
            'Paraneoplastic-Result (textual)','Cardiolipin-Result (numeric)',
            'Beta-2-Glycoprotein-Result (numeric)','Compliment-Result (numeric)',
            'CSF Protein-Result (numeric)','CSF WBC-Result (numeric)',
            'CRP-Result (numeric)','ANA Titer-Result (numeric)',
            'Homocysteine-Result (numeric)','Protein C-Result (numeric)',
            'Protein S-Result (numeric)','p-ANCA-Result (textual)',
            'c-ANCA-Result (textual)','ANCA pattern 1-Result (textual)',
            'ANCA pattern 2-Result (textual)']

   for val in keepList: colsRead.remove(val)

   de = de.drop(colsRead, inplace=False, axis=1)
   de = de.drop(de[de['Vasculitis Dx-Diagnosis']!='Diseases of the circulatory system'].index)
   cat_entities = ['Urine Protein-Result (textual)','Paraneoplastic-Result (textual)',
                'p-ANCA-Result (textual)',
                'c-ANCA-Result (textual)','ANCA pattern 1-Result (textual)',
                'ANCA pattern 2-Result (textual)']

else:

   ## Import data and store original data
   de = pd.read_excel (r'C:\Users\johnb\OneDrive\Desktop\VasculitisSim.xlsx')
   cat_entities = ['Infection']

################################################################################
"""
Data Augmentation - Functions
"""
################################################################################

# Create sparse data directly without any patterns to the missingness
def sparseDefinite(sparsity, dfin, targetIn, randomStateSelect=1):

   # Excludes
   excludesFunct = targetIn + ['MRN']

   # Make excludes
   includesFunct = [j for j in dfin.columns if j not in excludesFunct]

   ## Shift the target to the end
   dfout = dfin.loc[:, includesFunct].copy()
   dfout[excludesFunct] = dfin[excludesFunct]

   ## Select without replacement indicies and seperate out row / column data
   fullSize = dfout.loc[:, includesFunct].shape[0]*dfout.loc[:, includesFunct].shape[1]
   requiredSelections = round(fullSize*sparsity) - dfin.isna().sum().sum()

   ## Find data for nullification
   if requiredSelections > 0:
      random.seed(randomStateSelect)
      choicesSelect = random.sample(range(0, fullSize,1), requiredSelections)
      randomStateSelect += 1
      colSparse = choicesSelect//np.array(dfout.loc[:, includesFunct].shape[0])
      rowSparse = choicesSelect%np.array(dfout.loc[:, includesFunct].shape[0])
      errorInd = False
   else:
      errorInd = True

   ## Remove values
   for indexNaN in range(0,len(colSparse),1):
      dfout.iat[rowSparse[indexNaN], colSparse[indexNaN]] = np.nan
```

```
    return dfout, errorInd, randomStateSelect

# Find the index in a matrix where a threshold (probability) has been exceeded
def findIndexWithThreshold(dfin, targetValue):

    #Ensue probability compatibile matrix entered
    dfuse = dfin.copy()/dfin.sum().sum()

    #Initialize accumulator
    totalValue = 0

    #Accumulate values to determin when theshold exceeded
    for rowInd in range(0,dfuse.shape[0],1):
        for colInd in range(0,dfuse.shape[1],1):
            totalValue = totalValue + dfuse.iloc[rowInd, colInd]
            if totalValue > targetValue:
                break
        if totalValue > targetValue:
            break

    # Return indicies where threshold probability exceeded
    return rowInd, colInd

# Create sparse data directly with defined patterns to the missingness
def sparseProbabilistic(sparsity, baseProbabilities, valueCorrelation, missingCorrelation, dfin, dfpresent, targetIn, factorReduction = [], factorShift = [], randomStateSelect=1):
    # In value correlation the probability of missingess is correlated to values of other vairables
    # In missigness correlation the probability of missigness is correlated
    # the matrcies are both 13 x 13

    # Select without replacement indicies and seperate out row / column data
    fullSize = (dfin.shape[1] - 1)*dfin.shape[0]
    requiredSelections = round(fullSize*sparsity) - dfin.isna().sum().sum()

    if requiredSelections > 0:

        # Create Random Number Generator and select random numbers
        randomNumberGenerator=np.random.RandomState(randomStateSelect)
        randomSelection = randomNumberGenerator.uniform(low=0, high=1, size=requiredSelections)
        randomStateSelect += 1

        cat_vars = dfin[targetIn]
        cat_dums = pd.get_dummies(cat_vars, drop_first=True)

        if len(targetIn) == cat_dums.shape[1]:
            cat_dums.columns = targetIn

        # Drop columns that will not be useful to analysis / text catagorical variables
        dfuse = dfin.copy()
        dfuse = dfin.drop(targetIn, axis=1)
        dfuse = pd.concat([dfuse, cat_dums], axis=1).astype(float)

        # Create Probability Maps
        if len(factorShift) != len(valueCorrelation):
            factorShift = [0]*len(valueCorrelation)

        # Create Probability Maps
        if len(factorReduction) > 0:
            valCorMx = ((dfuse - factorShift)/factorReduction).dot(valueCorrelation)
        else:
            valCorMx = (dfuse - factorShift).dot(valueCorrelation)

        missMx = pd.DataFrame(np.ones([dfpresent.shape[0],dfpresent.shape[1]])) - dfpresent
        presentMx = dfpresent.copy()

        # Iterate to solve
        ## Initialize index collector
        indexSparse = []
        dfout = dfuse.copy()

        for selectedValue in randomSelection:
            misCorMx = missMx.dot(missingCorrelation)

            # Consolidate Probability Maps
            probabMx = pd.DataFrame(np.exp(baseProbabilities + valCorMx + misCorMx)*presentMx)
            probabMx = probabMx / probabMx.sum().sum()

            # Resolve indicies
            foundIndicies = findIndexWithThreshold(probabMx, selectedValue)

            # Persist Findings
            indexSparse.append(foundIndicies)
            dfout.iat[foundIndicies] = np.nan
            missMx.iloc[foundIndicies] = 1
            presentMx.iloc[foundIndicies] = 0

    else:
        dfout = dfin.copy()
```

```
    # print('FS')
    # print(factorShift)
    # print('FR')
    # print(factorReduction)
    # print('DU')
    # print(dfuse)
    # print('V')
    # print(valCorMx)
    # print('P')
    # print(probabMx)

    return dfout, randomStateSelect

# Find the indicies where the column does not have nan values
def findNonNan (df, columnSelectFunct):
    #indexNanFunct = df[columnSelectFunct].index[df[columnSelectFunct].apply(np.isnan)]

    indexNanFunct = df.loc[df.iloc[:,columnSelectFunct].isnull()].index.tolist()


    return list(set(df.index.values.tolist()) - set(indexNanFunct))

# Data shuffling algorithm to introduce noise into data
def randomAdjustArray(dfinfun, exceptionCols, fractionRandomized=0.5, randomEntryList=[], randomStateSelect=1):

    # Select Random Track
    randomFunctIter = randomStateSelect

    # Copy frame to block overwrite
    dfiff = dfinfun.copy()

    # Decode fraction if fixed
    # randomBreak = floor(dfiff.shape[0] * fractionRandomized)

    if dfiff.shape[1] != len(randomEntryList):
        randomEntryList = [fractionRandomized]*dfiff.shape[1]

    # Shuffle column by column
    for featureIndex in range(dfiff.shape[1]):

        if not (dfiff.columns[featureIndex] in exceptionCols):

            currentColumnFunct = dfiff.columns[featureIndex]
            selectNonNanRange = np.where(dfinfun[currentColumnFunct].notnull())[0].tolist()

            ##### Decode fraction when nans already exist
            randomBreak = floor(len(selectNonNanRange) * randomEntryList[featureIndex])

            random.seed(randomFunctIter)
            arrF = random.sample(selectNonNanRange, randomBreak)
            randomFunctIter += 1
            shuffledPieces = shuffle(dfinfun[currentColumnFunct][arrF], random_state = randomFunctIter)
            randomFunctIter += 1
            dfiff.loc[arrF, currentColumnFunct] = shuffledPieces.values

    ### Return a modified matrix
    return dfiff, randomFunctIter

################################################################################
"""
Data Augmentation - Execute
"""
################################################################################

def levelDataPerfectly (dataFrameFunctIn, experimentalVariable = experimentalVariableSelect):

    deFunct = dataFrameFunctIn.copy()

    variableToAdjust = experimentalVariable

    if max(deFunct[variableToAdjust]) < 0:
        bound1 = min(deFunct[variableToAdjust][deFunct['Vasculitis'] == 'Primary'])
        bound2 = max(deFunct[variableToAdjust][deFunct['Vasculitis'] == 'Not'])
        perfectThreshold = np.mean([bound1, bound2])

    else:
        bound1 = max(deFunct[variableToAdjust][deFunct['Vasculitis'] == 'Primary'])
        bound2 = min(deFunct[variableToAdjust][deFunct['Vasculitis'] == 'Not'])
        perfectThreshold = np.mean([bound1, bound2])

    expData = binAVariable(deFunct, variableToAdjust, [perfectThreshold], 'absolute', 2, False)
    deFunct = expData[0]

    return deFunct

def addRandomToData (dataFrameFunctIn, experimentalVariable = experimentalVariableSelect, randomMasterFunct = randomMaster, highSelect = 0.5):

    dataFrameOut = dataFrameFunctIn.copy()
```

```
    columnData = dataFrameFunctIn.iloc[:,experimentalVariable].copy()
    randomNumberGenerator=np.random.RandomState(randomMasterFunct)
    randomSelection = randomNumberGenerator.uniform(low=0, high=highSelect, size=len(columnData))
    columnData = columnData + randomSelection
    dataFrameOut.iloc[:,experimentalVariable] = columnData

    return dataFrameOut

def addRandomToWholeFrame (dataFrameFunctIn, randomMasterFunct = randomMaster):

    dataFrameOut = dataFrameFunctIn.copy()
    for currentTitle in range(dataFrameOut.shape[1]):
        dataFrameOut = addRandomToData(dataFrameOut,currentTitle,randomMasterFunct + 1 + currentTitle,0.005)

    randomMasterFunct = randomMasterFunct + 1 + currentTitle+ 1 + currentTitle
    return dataFrameOut, randomMasterFunct

def dataAugmentation (dataFrameFunctIn, targetIn  = target, cat_entitiesFunct = cat_entities, randomMasterFunct = 0, fillMethod = fillMethodSelect, sparsitySelect = sparsitySelection,
shuffleSelect = shuffleSelection, shuffleListIn = shuffleList, dpbIn = dpb, valueInCorrelationMatrixIn = valueInCorrelationMatrix, missingCorrelationMatrixIn = missingCorrelationMatrix,
runExperiment = runExperimentSelect, experimentalVariable = experimentalVariableSelect, engagePseudorandom = engagePseudorandomSelect, perfectSplit = perfectSplitSelect):

    # Run Experiment
    if runExperiment:
        if perfectSplit:
            shuffledData = levelDataPerfectly(dataFrameFunctIn)
        else:
            shuffledData, _, _ = binAVariable(dataFrameFunctIn, experimentalVariable, partitionThresholds = centileSelect, typeThresholds = 'centiles', numberPartitions = 2, engagePseudorandom
= False)
    else:
        shuffledData = dataFrameFunctIn.copy()

    # Note special columns
    ## Identifiers / data that does not convey clinical value
    recordid  = ['MRN']

    # Create a list of all exemptions
    exemptionColumns = recordid + targetIn

    if (shuffleSelect > 0) or (len(shuffleListIn) > 0):
        # Ensure valid shuffle request
        if shuffleSelect > 1:
            shuffleSelect = 1

        # Shuffle the data matrix
        shuffledData, randomMasterFunct = randomAdjustArray(shuffledData, exemptionColumns, shuffleSelect, shuffleListIn, randomMasterFunct)

    # probability based sparsity
    if sparsitySelect > 0:
        # Create value dependency matricies
        ###########################################################################
        # Create a base matrix by repeating the base selection over the number of cases
        setBaseProbabilityMatrix = pd.concat([dpbIn]*dataFrameFunctIn.shape[0], ignore_index=True)

        # Create presence matrix where one is not initially available
        maintainPresence = pd.DataFrame(np.ones([dataFrameFunctIn.shape[0],dataFrameFunctIn.shape[1]-len(exemptionColumns)]))

        # Zero a number of final columns within the target zone
        maintainPresence[list(range(dataFrameFunctIn.shape[1]-len(exemptionColumns),
                            dataFrameFunctIn.shape[1]))] = np.zeros([dataFrameFunctIn.shape[0],len(exemptionColumns)])

        # Create feature value normalizer
        maxesColumn = dataFrameFunctIn.drop(exemptionColumns, axis = 1).max()
        minsColumn = dataFrameFunctIn.drop(exemptionColumns, axis = 1).min()
        logicalSave = maxesColumn < (minsColumn * -1)
        senseHold = np.ones(len(logicalSave))
        senseHold[logicalSave] = -1

        # Simple normalization for value base missingness
        normalizerForFeatures = (maxesColumn - minsColumn)/2
        middleForFeatures = (minsColumn + normalizerForFeatures).tolist() + [0]*len(exemptionColumns)
        normalizerForFeatures = (senseHold*normalizerForFeatures).tolist() + [1]*len(exemptionColumns)

        # Sparsify the matrix
        sp, randomMasterFunct = sparseProbabilistic(sparsitySelect,
                                    setBaseProbabilityMatrix,
                                    valueInCorrelationMatrixIn,
                                    missingCorrelationMatrixIn,
                                    shuffledData,
                                    maintainPresence,
                                    targetIn,
                                    normalizerForFeatures,
                                    middleForFeatures,
                                    randomMasterFunct)
    else:
        sp = shuffledData

    # Keep original data
```

```
dor = shuffledData.copy()

# replace the target values
# dep = sp.drop(sp.columns[len(sp.columns)-1],axis=1)

dep = sp.drop(targetIn[0], inplace=False, axis=1)
dep[targetIn[0]]=dataFrameFunctIn[targetIn[0]]

dor = shuffledData.drop(targetIn[0], inplace=False, axis=1)
dor[targetIn[0]]=dataFrameFunctIn[targetIn[0]]

# select the desired data to process
df = dep.copy()

## Store desired columns im Manipulation Matricies
### Use one hot encoding / dummy veriables to regularize text columns
#### First decide if catagorical entities exist

if not not cat_entitiesFunct:

   cat_dummies_s = pd.get_dummies(df[cat_entitiesFunct], drop_first=True)
   cat_dummies_o = pd.get_dummies(df[cat_entitiesFunct], drop_first=True)

   #### Drop columns that will not be useful to analysis / text catagorical variables
   dataDummyFunct = df.drop(cat_entitiesFunct, axis=1)
   dataDummyFunct = pd.concat([dataDummyFunct, cat_dummies_s], axis=1)

   dor = dor.drop(cat_entitiesFunct, axis=1)
   dor = pd.concat([dor, cat_dummies_o], axis=1)


else:
   #### Create standard
   dataDummyFunct = df.copy()

## Organize data
colsFunct = [col for col in dataDummyFunct.columns if col not in exemptionColumns]

## Impute missing data
# Add pseudorandomness if appropriate
if (engagePseudorandom) and (runExperiment):
   nonNullColumn = (~dataDummyFunct.loc[:,experimentalVariable].isnull()).tolist()

## Impute missing data
dataOut = dataDummyFunct.copy()

if fillMethod == 'mean':
   ### Using Mean / Median
   for i in colsFunct:

      fillerSelect = dataOut.loc[:,i].mean()

      if pd.isna(fillerSelect):
         dataOut.loc[:,i]=0
      else:
         dataOut.loc[dataOut.loc[:,i].isnull(),i]=fillerSelect

elif fillMethod == 'median':
   ### Using Mean / Median
   for i in colsFunct:
      fillerSelect = dataOut.loc[:,i].median()

      if pd.isna(fillerSelect):
         dataOut.loc[:,i]=0
      else:
         dataOut.loc[dataOut.loc[:,i].isnull(),i]=fillerSelect

elif fillMethod == '3NN':
   ### Using 3 nearest neighbors
   #### Instantiate imputer object
   imputer = KNNImputer(n_neighbors=3)

   #### Temporarily adjust target data for clustering
   tar_variable = dataOut[targetIn]
   tar_dummies = pd.get_dummies(tar_variable, drop_first=True)
   tarDummy = dataOut.drop(exemptionColumns, axis=1)
   tarDummy = pd.concat([tarDummy, tar_dummies], axis=1)

   #### Apply KNN imputation algorithm
   tarDummy = pd.DataFrame(imputer.fit_transform(tarDummy),columns = tarDummy.columns)
   tarDummy = tarDummy.drop(tar_dummies.columns, axis=1)

   #### Restore the structure of the original data
   tarDummy = pd.concat([dataOut[exemptionColumns].reset_index(drop=True), tarDummy], axis=1)
   dataOut = tarDummy.copy()

else:
   ### Alternative
```

```
        dataOut = dataDummyFunct.copy()
        dataOut[colsFunct] = dataOut[colsFunct].fillna(value=dataDummyFunct[colsFunct].mean())

    # Add pseudorandomness if appropriate
    if (engagePseudorandom) and (runExperiment):
        dataDummyFunct.loc[nonNullColumn,experimentalVariable] = np.array(addRandomToData(dataDummyFunct.loc[nonNullColumn])[experimentalVariable]).tolist()
        dataOut.loc[nonNullColumn,experimentalVariable] = np.array(dataDummyFunct.loc[nonNullColumn,experimentalVariable]).tolist()
        randomMasterFunct += 1

    return dataDummyFunct, dataOut, colsFunct, randomMasterFunct, dor

# Check missingness ratios
def checkMissingRatios(p, columnSelectFunct):
    propPrimary = p[columnSelectFunct][p['Vasculitis']=='Primary'].isnull().sum() / (p['Vasculitis'] == 'Primary').sum()
    propNot = p[columnSelectFunct][p['Vasculitis']=='Not'].isnull().sum() / (p['Vasculitis'] == 'Not').sum()
    return(propPrimary, propNot)

# Create data packages with raw data and imputed data seperated in to test / train splits
def dataPackageGeneration(dataFrameFunctIn, targetIn  = [], cat_entitiesFunct = [], randomMasterFunct = 0):
    dataDummy, dataMean, cols, randomMasterFunct, dataOriginal = dataAugmentation(dataFrameFunctIn,
                                                targetIn,
                                                cat_entitiesFunct,
                                                randomMasterFunct)

    ###############################################################################
    """
    Data For Novel Ensemble
    """
    ###############################################################################

    ## Select the data
    dataSelected = dataDummy

    ## Divide the data
    x = dataSelected[cols]
    y = np.ravel(dataSelected[targetIn])
    x_use_Funct, x_out_Funct, y_use_Funct, y_out_Funct = train_test_split(x,
                                            y,
                                            test_size = .20,
                                            random_state=randomMasterFunct,
                                            stratify=y)

    randomMasterFunct += 1

    ###############################################################################
    """
    Established Ensemble (Random Forest) - Data
    """
    ###############################################################################

    ## Select the data
    dataSelected = dataMean

    ## Divide the data
    x = dataSelected[cols]
    y = np.ravel(dataSelected[targetIn])

    if runDifferentPredictMissingness == True:

        # Create random missingness pattern
        dOr, _, randomMasterFunct = sparseDefinite(sparseDefiniteProportion, dataOriginal, targetIn, randomStateSelect=randomMasterFunct)

        # Ammend novel data x values
        ## Select the data
        dataSelected = dOr

        xOr = dataSelected[cols]
        xOrF = xOr.fillna(dataMean.mean(numeric_only=True))

        x_out_Funct = xOr.iloc[x_out_Funct.index].copy()

        # Ammend standard data values
        ## Use prior indicies and generated data pattern for training
        x_use_f_Funct = x.iloc[x_use_Funct.index].copy()
        y_use_f_Funct = y[x_use_Funct.index].copy()

        ## Use prior indicies and random data pattern for training
        x_out_f_Funct = xOrF.iloc[x_out_Funct.index].copy()
        y_out_f_Funct = y[x_out_Funct.index].copy()

    else:
        ## Use prior indicies
        x_use_f_Funct = x.iloc[x_use_Funct.index].copy()
        y_use_f_Funct = y[x_use_Funct.index].copy()
        x_out_f_Funct = x.iloc[x_out_Funct.index].copy()
        y_out_f_Funct = y[x_out_Funct.index].copy()

    return x_use_Funct, x_out_Funct, y_use_Funct, y_out_Funct, x_use_f_Funct, y_use_f_Funct, x_out_f_Funct, y_out_f_Funct, randomMasterFunct
```

```
def dataAugmentationToFrame (dataFrameFunctIn, targetIn  = target, cat_entitiesFunct = cat_entities, randomMasterFunct = 0, fillMethod = fillMethodSelect, sparsitySelect =
sparsitySelection, shuffleSelect = shuffleSelection, shuffleListIn = shuffleList, dpbIn = dpb, valueInCorrelationMatrixIn = valueInCorrelationMatrix, missingCorrelationMatrixIn =
missingCorrelationMatrix, runExperiment = runExperimentSelect, experimentalVariable = experimentalVariableSelect, engagePseudorandom = engagePseudorandomSelect, perfectSplit =
perfectSplitSelect):

    # Run Experiment
    if runExperiment:
        if perfectSplit:
            shuffledData = levelDataPerfectly(dataFrameFunctIn)
        else:
            shuffledData, _, _ = binAVariable(dataFrameFunctIn, experimentalVariable, partitionThresholds = centileSelect, typeThresholds = 'centiles', numberPartitions = 2, engagePseudorandom
= False)
    else:
        shuffledData = dataFrameFunctIn.copy()

    # Note special columns
    ## Identifiers / data that does not convey clinical value
    recordid  = ['MRN']

    # Create a list of all exemptions
    exemptionColumns = recordid + targetIn

    if (shuffleSelect > 0) or (len(shuffleListIn) > 0):
        # Ensure valid shuffle request
        if shuffleSelect > 1:
            shuffleSelect = 1

        # Shuffle the data matrix
        shuffledData, randomMasterFunct = randomAdjustArray(shuffledData, exemptionColumns, shuffleSelect, shuffleListIn, randomMasterFunct)

    # probability based sparsity
    if sparsitySelect > 0:
        # Create value dependency matricies
        ###########################################################################
        # Create a base matrix by repeating the base selection over the number of cases
        setBaseProbabilityMatrix = pd.concat([dpbIn]*dataFrameFunctIn.shape[0], ignore_index=True)

        # Create presence matrix where one is not initially available
        maintainPresence = pd.DataFrame(np.ones([dataFrameFunctIn.shape[0],dataFrameFunctIn.shape[1]-len(exemptionColumns)]))

        # Zero a number of final columns within the target zone
        maintainPresence[list(range(dataFrameFunctIn.shape[1]-len(exemptionColumns),
                        dataFrameFunctIn.shape[1]))] = np.zeros([dataFrameFunctIn.shape[0],len(exemptionColumns)])

        # Create feature value normalizer
        maxesColumn = dataFrameFunctIn.drop(exemptionColumns, axis = 1).max()
        minsColumn = dataFrameFunctIn.drop(exemptionColumns, axis = 1).min()
        logicalSave = maxesColumn < (minsColumn * -1)
        senseHold = np.ones(len(logicalSave))
        senseHold[logicalSave] = -1

        # Simple normalization
        normalizerForFeatures = (maxesColumn - minsColumn)/2
        middleForFeatures = (minsColumn + normalizerForFeatures).tolist() + [0]*len(exemptionColumns)
        normalizerForFeatures = (senseHold*normalizerForFeatures).tolist() + [1]*len(exemptionColumns)

        # Sparsify the matrix
        sp, randomMasterFunct = sparseProbabilistic(sparsitySelect,
                                    setBaseProbabilityMatrix,
                                    valueInCorrelationMatrixIn,
                                    missingCorrelationMatrixIn,
                                    shuffledData,
                                    maintainPresence,
                                    targetIn,
                                    normalizerForFeatures,
                                    middleForFeatures,
                                    randomMasterFunct)
    else:
        sp = shuffledData

    # replace the target values
    dep = sp.drop(sp.columns[len(sp.columns)-1],axis=1)
    dep[targetIn[0]]=dataFrameFunctIn[targetIn[0]]

    # select the desired data to process
    df = dep.copy()

    return df

def bootSelection(dataFrameFunctIn, targetIn  = target, cat_entitiesFunct = cat_entities, randomMasterFunct = 0, fillMethod = 'mean'):

    # Create a boot select
    rng=np.random.RandomState(randomMasterFunct)

    #indexingLookup = np.array(indicesAvailable)
    indexSelect = rng.choice(dataFrameFunctIn.index.tolist(), dataFrameFunctIn.shape[0])
```

```
# indexSelect = rng.choice(list(range(dataFrameFunctIn.shape[0])), dataFrameFunctIn.shape[0])

# Note special columns
## Identifiers / data that does not convey clinical value
recordid  = ['MRN']

# Create a list of all exemptions
exemptionColumns = recordid + targetIn

# Create copy of input data to avoid back-editing
df = dataFrameFunctIn.loc[indexSelect].copy()
df.index = list(range(df.shape[0]))

## Store desired columns im Manipulation Matricies
### Use one hot encoding / dummy veriables to regularize text columns
#### First decide if catagorical entities exist

if not not cat_entitiesFunct:

   cat_variables = df[cat_entitiesFunct]
   cat_dummies = pd.get_dummies(cat_variables, drop_first=True)

   #### Drop columns that will not be useful to analysis / text catagorical variables
   dataDummyFunct = df.drop(cat_entitiesFunct, axis=1)
   dataDummyFunct = pd.concat([dataDummyFunct, cat_dummies], axis=1)

else:
   #### Create standard
   dataDummyFunct = df.copy()

## Organize data
colsFunct = [col for col in dataDummyFunct.columns if col not in exemptionColumns]

## Impute missing data
dataOut = dataDummyFunct.copy()

if fillMethod == 'mean':
   ### Using Mean / Median
   for i in colsFunct:
      fillerSelect = dataOut.loc[:,i].mean()

      if pd.isna(fillerSelect):
         dataOut.loc[:,i]=0
      else:
         dataOut.loc[dataOut.loc[:,i].isnull(),i]=fillerSelect

elif fillMethod == 'median':
   ### Using Mean / Median
   for i in colsFunct:
      fillerSelect = dataOut.loc[:,i].median()

      if pd.isna(fillerSelect):
         dataOut.loc[:,i]=0
      else:
         dataOut.loc[dataOut.loc[:,i].isnull(),i]=fillerSelect

elif fillMethod == '3NN':
   ### Using 3 nearest neighbors
   #### Instantiate imputer object
   imputer = KNNImputer(n_neighbors=3)

   #### Temporarily adjust target data for clustering
   tar_variable = dataOut[targetIn]
   tar_dummies = pd.get_dummies(tar_variable, drop_first=True)
   tarDummy = dataOut.drop(exemptionColumns, axis=1)
   tarDummy = pd.concat([tarDummy, tar_dummies], axis=1)

   #### Apply KNN imputation algorithm
   tarDummy = pd.DataFrame(imputer.fit_transform(tarDummy),columns = tarDummy.columns)
   tarDummy = tarDummy.drop(tar_dummies.columns, axis=1)

   #### Restore the structure of the original data
   tarDummy = pd.concat([dataOut[exemptionColumns].reset_index(drop=True), tarDummy], axis=1)
   dataOut = tarDummy.copy()

else:
   ### Alternative
   dataOut = dataDummyFunct.copy()
   dataOut[colsFunct] = dataOut[colsFunct].fillna(value=dataDummyFunct[colsFunct].mean())

################################################################################
"""
Data For Novel Ensemble
"""
################################################################################

## Select the data
dataSelected = dataDummyFunct
```

```
    ## Divide the data
    x = dataSelected[colsFunct]
    y = np.ravel(dataSelected[targetIn])
    x_use_Funct, x_out_Funct, y_use_Funct, y_out_Funct = train_test_split(x,
                                                y,
                                                test_size = .20,
                                                random_state=randomMasterFunct,
                                                stratify=y)

    randomMasterFunct += 1

    ################################################################################
    """
    Established Ensemble (Random Forest) - Data
    """
    ################################################################################

    ## Select the data
    dataSelected = dataOut

    ## Divide the data
    x = dataSelected[colsFunct]
    y = np.ravel(dataSelected[targetIn])

    ## Use prior indicies
    x_use_f_Funct = x.iloc[x_use_Funct.index].copy()
    y_use_f_Funct = y[x_use_Funct.index].copy()
    x_out_f_Funct = x.iloc[x_out_Funct.index].copy()
    y_out_f_Funct = y[x_out_Funct.index].copy()

    return x_use_Funct, x_out_Funct, y_use_Funct, y_out_Funct, x_use_f_Funct, y_use_f_Funct, x_out_f_Funct, y_out_f_Funct, randomMasterFunct



################################################################################
"""
Novel Ensemble - Create Ensemble - Function
"""
################################################################################

# Importance processing
def importanceWeighter(treesIn, weightingIn=[], normalizeFunct=treeImportanceCombination):

    importanceOut = []

    if len(weightingIn) != len(treesIn):
        weightingIn = [1]*len(treesIn)

    if normalizeFunct == 'mean':
        for treeCurrent in range(len(treesIn)):
            importanceOut.append(weightingIn[treeCurrent]*treesIn[treeCurrent].feature_importances_)
    elif normalizeFunct == 'raw score':
        for treeCurrent in range(len(treesIn)):
            importanceOut.append(weightingIn[treeCurrent]*treesIn[treeCurrent].tree_.compute_feature_importances(normalize=False))
    else:
        for treeCurrent in range(len(treesIn)):
            importanceOut.append(weightingIn[treeCurrent]*treesIn[treeCurrent].tree_.compute_feature_importances(normalize=False))

    importanceSummary = sum(importanceOut)
    importanceSummary = importanceSummary / sum(importanceSummary)

    return importanceSummary, importanceOut

# Function to generate Ensemble
def customGenerateEnsemble(x_funct, y_funct, random_base = 1, numberTrees = 1000, importanceApproach = importanceAdjustment, minPerLeafIn = minimumNumberPerLeaf):

    ### Create a recreatable random state
    random_augmenter = 0
    maxRunNumber = numberTrees

    ### Create templates
    featureTemplate = list(range(0,x_funct.shape[1],1))

    ### Create Running List of Classifiers, Features, Out Data
    clfLFunct = []
    featuresListFunct = []
    rawCorrectOOBFunct = []
    outOfBagFunct = []
    correctOOBFunct = []
    totalOOBFunct = []
    voteResultFunct = []
    numUsed = []
    numRetained = []
    allOfOneType = []

    indiciesAvaliableRetain = []
```

```
# Create Lock out trackers
lockOutMatrix = np.zeros((x_funct.shape[1], x_funct.shape[1]))
lockOutMatrixX = np.zeros((x_funct.shape[1], x_funct.shape[1]))
nonLockOutMatrix = np.zeros((x_funct.shape[1], x_funct.shape[1]))

# Set maximum number of features to be included in any one tree
maxIncFeatures = floor(np.sqrt(x_funct.shape[1]))

### Generate Classifiers
for treeCount in range(0,numberTrees,1):

    #### Initialize Boot Variables
    runSelect = True
    runMaintain = True
    runNumber = 0

    ### Run loop to select features where there are at least some complete datapoints
    while runSelect and runMaintain:

        ### Adjust Random State
        randomState = random_base + random_augmenter
        rng=np.random.RandomState(randomState)

        ## Adjust augmenter for next use
        random_augmenter += 1

        #### Increment Run Number
        runNumber += 1

        ### chose feature select
        featureSelect = np.unique(list(rng.randint(low = 0,
                                    high=x_funct.shape[1],
                                    size=maxIncFeatures))).tolist()

        ### data sample
        #### punch out data to find rows that will be rejected because of missing data
        dataSample = x_funct.copy()
        dataSample = x_funct.iloc[:,featureSelect]
        indicesAvailable = np.where(np.logical_not(dataSample.isnull().any(axis=1)))[0]

        indiciesAvaliableRetain += [np.array(indicesAvailable)]

        # Ensure that there are entries in at least two clases
        if len(indicesAvailable) > 0:
            if (sum(y_funct[indicesAvailable] == 'Primary') > 0) and (len(np.unique(y_funct[indicesAvailable])) > 1):
                runSelect = False

                # Note the non-interferences
                updateIndiciesFunct = featureSelect*len(featureSelect)
                sortedIndicies = updateIndiciesFunct.copy()
                sortedIndicies.sort()
                nonLockOutMatrix[updateIndiciesFunct,sortedIndicies] += 1
            else:

                # Note the interferences
                updateIndiciesFunct = featureSelect*len(featureSelect)
                sortedIndicies = updateIndiciesFunct.copy()
                sortedIndicies.sort()
                lockOutMatrixX[updateIndiciesFunct,sortedIndicies] += 1
        else:

            # Note the interferences
            updateIndiciesFunct = featureSelect*len(featureSelect)
            sortedIndicies = updateIndiciesFunct.copy()
            sortedIndicies.sort()
            lockOutMatrix[updateIndiciesFunct,sortedIndicies] += 1

        if runNumber > maxRunNumber:
            runMaintain = False
            return 'Fail', 'Fail', 'Fail', 'Fail', 'Fail', 'Fail', 'Fail', 'Fail', 'Fail', 'Fail', (random_base + random_augmenter), 'Fail', 'Fail', 'Fail', 'Fail', 'Fail'

    #### Reset Boot Variables
    runSelect = True
    runMaintain = True
    runNumber = 0

    ### Sample with replacement
    #### Sample
    # Alt: sampleSelect = list(rng.randint(low = 0,high=dataSelected.shape[0]-1,size=dataSelected.shape[0]))
    # Start by ensuring 1 entry from the target class and 1 from another class
    indexingLookup = np.array(indicesAvailable)
    valuePrimary = rng.choice(indexingLookup[y_funct[indicesAvailable] == 'Primary'], minPerLeafIn)
    valueNot = rng.choice(indexingLookup[y_funct[indicesAvailable] != 'Primary'], minPerLeafIn)

    seedSample = np.concatenate((valuePrimary, valueNot),axis=None)
    numberSampleToAdd = x_funct.shape[0] - minPerLeafIn*2
```

```
        # Select the remainder of the entries
        if numberSampleToAdd > 0:
            sampleSelectRandom = rng.choice(indicesAvailable, (x_funct.shape[0] - minPerLeafIn*2))
            sampleSelect = np.concatenate((seedSample,sampleSelectRandom),axis=None)
        else:
            numberSampleToAdd = seedSample

        # Catalogue the unique entries used
        samplesSelected = np.unique(sampleSelect)

        # Number Used
        numUsed.append(len(samplesSelected))

        ##### Find samples not selected
        samplesNotSelected = np.array(list(set(indicesAvailable) - set(samplesSelected)), dtype=int)

        # Number Out of Bag
        numRetained.append(len(samplesNotSelected))

        ### Nulify select columns
        ##### Initialize: Note the stop number is not included
        featuresNotSelected = featureTemplate.copy()

        ##### Chip away
        for i in sorted(featureSelect, reverse=True):
            del featuresNotSelected[i]

        ### Discern which data to use and which data to calculate out of bag metrics on
        #### Resolve X
        x_mod = x_funct.copy()

        # Nullify extraneous data rendering it unable to be used for data partitioning
        if len(featuresNotSelected) > 0:
            x_mod.iloc[:,np.array(featuresNotSelected)] = 0

        x_oob = x_mod.iloc[samplesNotSelected,:]
        x_mod = x_mod.iloc[sampleSelect,:]

        #### Resolve Y
        y_oob = y_funct[samplesNotSelected]
        y_mod = y_funct[sampleSelect]

        #### Track proportion that are of a single classification
        # propPN = (y_mod == 'Primary').sum()/((y_mod == 'Primary').sum()+(y_mod != 'Primary').sum())
        allOfOneType += [[(y_mod == 'Primary').sum(),(y_mod != 'Primary').sum()]]

        ### Adjust Random State
        randomState = random_base + random_augmenter

        ## Generate a classifier
        clfN = DecisionTreeClassifier(
            min_samples_leaf=minPerLeafIn,
            random_state=randomState
            )

        ## Adjust augmenter for classifier
        random_augmenter += 1

        # Train Decision Tree Classifer
        clfN = clfN.fit(x_mod,y_mod)

        # Find correct and total out of bag guesses and store
        if len(y_oob) > 0:
            oPred = clfN.predict(x_oob)
            voteResultFunct.append(oPred)
            iscorrect = (y_oob == oPred)
            rawCorrectOOBFunct = rawCorrectOOBFunct + iscorrect.tolist()
            correctOOBFunct.append(sum(iscorrect))
            totalOOBFunct.append(len(y_oob))
            outOfBagFunct = outOfBagFunct + samplesNotSelected.tolist()

        ### Add to the running list
        featuresListFunct.append(featureSelect)
        clfLFunct.append(clfN)

    ### Extract Importances (In built)
    # Assign weights to importance estimations depending on missingness
    if importanceApproach == 'Avg':
        giniImportanceFunct, gImportanceFull = importanceWeighter(clfLFunct, numUsed)
    elif importanceApproach == 'Raw':
        giniImportanceFunct, gImportanceFull = importanceWeighter(clfLFunct)
    else:
        giniImportanceFunct, gImportanceFull = importanceWeighter(clfLFunct)

    return clfLFunct, featuresListFunct, outOfBagFunct, voteResultFunct, rawCorrectOOBFunct, correctOOBFunct, totalOOBFunct, giniImportanceFunct, gImportanceFull, indiciesAvaliableRetain,
(random_base + random_augmenter), numUsed, numRetained, lockOutMatrix, lockOutMatrixX, nonLockOutMatrix, allOfOneType

# Determine the out of bag error
```

```
def decodeOOBError(sampleNumbers, correctValues):

    correctVoteProportionFunct = []

    dataFunct = {'SN': sampleNumbers, 'CV': correctValues}
    pdFunct = pd.DataFrame(dataFunct)

    for sampleSelectedIndex in list(set(sampleNumbers)):
        ofInterestComponent = pdFunct.CV[pdFunct.SN == sampleSelectedIndex]
        correctVoteProportionFunct += [sum(ofInterestComponent)/len(ofInterestComponent)]

    oobErrorFunct = sum(1 for i in range(len(correctVoteProportionFunct)) if correctVoteProportionFunct[i] >= 0.5)/len(correctVoteProportionFunct)

    return oobErrorFunct

################################################################################
"""
Novel Ensemble - Vote with Ensemble - Function
"""
################################################################################

# Function for orderly voting
def customVoteEnsemble(xDataIn, ensembleList, featuresListed):

    # Initialize voting structures
    votesCreatedOut = []
    nullField = xDataIn.isnull()

    # Create useful copy
    x_out_reduced = xDataIn.copy()
    x_out_reduced[nullField] = 0

    ## Create and modify votes
    for treeCount in range(0,len(ensembleList),1):

        ### Create naive votes where voters using NaN data will have to be later changed
        votesCreatedOut.append(ensembleList[treeCount].predict(x_out_reduced))

        ### Adjust votes that relied on the adjustment for NaN data
        for sampleCount in range(0,xDataIn.shape[0],1):
            if  nullField.iloc[sampleCount,featuresListed[treeCount]].any():
                votesCreatedOut[treeCount][sampleCount]='Abstain'

        ## Sumarize the votes
        voteDf = pd.DataFrame(votesCreatedOut).T
        voteBool = voteDf == 'Primary'
        abstainBool = voteDf == 'Abstain'

        ## Numerators
        numeratorValues = voteBool.sum(axis=1)

        ## Denominators
        denominatorValues = len(ensembleList) - abstainBool.sum(axis=1)

        ## Adjust to avoid infinite values
        numeratorValues[denominatorValues == 0] = 0
        denominatorValues[denominatorValues == 0] = 1

        ## Lazy divide
        voteProba = np.array(numeratorValues / denominatorValues)

    return votesCreatedOut, voteProba

# Change voting probabilities into pleurality
def convertToPlurality(voteProbabilities, probabilityThreshold = 0.5, baseClassName = 'Not', selectClassName = 'Primary'):
    minimumOfPluralityVote = np.array([baseClassName] * len(voteProbabilities), dtype='O')
    minimumOfPluralityVote[voteProbabilities >= probabilityThreshold] = selectClassName

    return minimumOfPluralityVote

# Function for providing the majority decision
def customVoteEnsemblePlurality(xDataIn, ensembleList, featuresListed):
    _, voteProba = customVoteEnsemble(xDataIn, ensembleList, featuresListed)
    minimumOfPluralityVote = np.array(['Not'] * xDataIn.shape[0], dtype='O')
    minimumOfPluralityVote[voteProba >= 0.5] = 'Primary'

    return minimumOfPluralityVote

## create custom roc curve
def roc_curve_custom(outLogic, voteProbabilityFunct):

    ## Compute true / false positive rates for every level available
    rocSegmentsUsed = np.concatenate([voteProbabilityFunct,[1],[0]])
    rocSegmentsUsed = np.unique(sorted(rocSegmentsUsed, reverse=False))

    ## Obtain logic for traget class
    #outLogic = y_out == 'Primary'
```

```
    ## Compute true and false positive rates
    tprFunct = []
    fprFunct = []

    ### Iterate through
    for segi in rocSegmentsUsed:
        iPred = voteProbabilityFunct > segi
        FP = sum(iPred & ~outLogic)
        TP = sum(iPred & outLogic)
        TN = sum(~iPred & ~outLogic)
        FN = sum(~iPred & outLogic)

        if TP == 0 and FN == 0:
            tprFunct.append(1)
        else:
            tprFunct.append(TP/(TP + FN))

        if FP == 0 and TN == 0:
            fprFunct.append(1)
        else:
            fprFunct.append(FP/(FP + TN))

    # Cap the curve
    tprFunct = [1] + tprFunct
    tprFunct.append(0)

    fprFunct = [1] + fprFunct
    fprFunct.append(0)

    ## finalize lists and convert
    tprFunct = np.array(tprFunct)
    fprFunct = np.array(fprFunct)

    ## Accuracy
    accuracyFunct = sum((voteProbabilityFunct >= 0.5) == outLogic)/len(outLogic)

    return fprFunct, tprFunct, accuracyFunct

# Removes unnecessary points from ROC curves and regularizes them
def refineROCCurveDataToLists(tprIn, fprIn):

    # Adjust curve
    frameIROC = pd.DataFrame({'a': fprIn, 'b': tprIn})
    frameXROC = frameIROC.sort_values(['a', 'b'], axis=0, ascending=True).reset_index(drop = True)

    newA = []
    newB = []

    # Find and drag forward points to allow for a single x-value column
    for indexOn in range(0,len(frameXROC)-1,1):
        if frameXROC.b[indexOn] == frameXROC.b[indexOn + 1]:
            if frameXROC.a[indexOn] != frameXROC.a[indexOn + 1]:
                if newB == []:
                    newA += [frameXROC.a[indexOn]]
                    newB += [frameXROC.b[indexOn]]
                elif newB[-1] != frameXROC.b[indexOn]:
                    newA += [frameXROC.a[indexOn]]
                    newB += [frameXROC.b[indexOn]]
        else:
            if frameXROC.a[indexOn] == frameXROC.a[indexOn + 1]:
                if newA != []:
                    if newA[-1] != frameXROC.a[indexOn]:
                        newA += [frameXROC.a[indexOn]]
                        newB += [frameXROC.b[indexOn]]
            else:
                newA += [frameXROC.a[indexOn]]
                newB += [frameXROC.b[indexOn]]
                newA += [frameXROC.a[indexOn + 1]]
                newB += [frameXROC.b[indexOn]]

    newA += [frameXROC.a.tolist()[-1]]
    newB += [frameXROC.b.tolist()[-1]]

    frameFunctROC = pd.DataFrame({'a': newA, 'b': newB})

    # Neutralize sort
    frameFunctROC = frameFunctROC.sort_values(['a', 'b'], axis=0, ascending=False).reset_index(drop = True)

    return frameFunctROC.b.tolist(), frameFunctROC.a.tolist()

# Plot importances
# https://stackoverflow.com/questions/44511636/plot-feature-importance-with-feature-names
def plotImportancesWithBars(giniImportanceIn, giniImportanceFIn, indiciesIn):
    std = np.std(giniImportanceFIn, axis=0)
    indices = np.argsort(giniImportanceIn)

    # Plot the feature importances of the forest
    plt.figure()
```

```
    plt.title("Feature importances")
    plt.barh(range(len(giniImportanceIn)), giniImportanceIn[indices],
           color="r", xerr=std[indices], align="center")

    # If you want to define your own labels,
    # change indices to a list of labels on the following line.
    plt.yticks(range(len(giniImportanceIn)), indiciesIn[indices])
    plt.ylim([-1, len(giniImportanceIn)])
    plt.show()

################################################################################
################################################################################
################################################################################
"""
Mutliple Methods - Execute
"""
################################################################################
################################################################################
################################################################################

if singleIter == True:
    ################################################################################
    """
    Create a sample data set
    """

    ################################################################################
    randomMaster = randomMaster + 1000000
    x_use, x_out, y_use, y_out, x_use_f, y_use_f, x_out_f, y_out_f, randomMaster = dataPackageGeneration(de,
                                                                    target,
                                                                    cat_entities,
                                                                    randomMasterFunct = randomMaster)

    ################################################################################
    """
    Cluster Features
    """
    ################################################################################

    # Discern clusters
    ## Compute correlation matricies
    ### Note that where correlation can not be discerned no values were present simultaneously and thus it is assumed that no correlation would exist
    featureCorrelations = x_out.corr().fillna(0)
    distance_matrix = hierarchy.linkage(featureCorrelations, 'single')

    ## Create a dendrogram
    dn = hierarchy.dendrogram(distance_matrix, labels=x_out.columns.tolist(), leaf_rotation=90)
    ## Display the dendogram
    plt.show()

    ## Create list of features that should be used
    featureUse = ['malic_acid', 'alcalinity_of_ash', 'magnesium', 'ash', 'alcohol']

    ################################################################################
    """
    Adjust Data
    """
    ################################################################################

    indiciesToExplore0 = x_out.columns
    indiciesToExplore1 = []
    indiciesToUse = []
    for strElement in indiciesToExplore0: indiciesToExplore1.append(strElement.replace('-Result (numeric)', ''))
    for strElement in indiciesToExplore1: indiciesToUse.append(strElement.replace('-Result (textual)', ''))
    indiciesToUse = np.array(indiciesToUse)

    x_use, randomMaster = addRandomToWholeFrame(x_use)
    x_use_f, randomMaster = addRandomToWholeFrame(x_use_f)

    ################################################################################
    """
    Novel Ensemble - Execute
    """
    ################################################################################

    # Generate the ensemble
    clfL, featuresList, outOfBag, vroob, rawCorrect, correctOOB, totalOOB, giniImportance, giniImportanceFout, _, randomMaster, numUsedIter, numRetainedIter, lockOutIter, lockOutXIter,
nonLockOutIter, allOfOneTypeIter = customGenerateEnsemble(x_use, y_use, randomMaster, numberTrees = desiredTreeNumberSelect)

    # Importance calculate
    ### Extract Importances
    # Assign weights to importance estimations depending on missingness
    if importanceAdjustment == 'Avg':
        giniImportance, giniImportanceFout = importanceWeighter(clfL, numUsedIter)
    elif importanceAdjustment == 'Raw':
        giniImportance, giniImportanceFout = importanceWeighter(clfL)
    else:
        giniImportance, giniImportanceFout = importanceWeighter(clfL)
```

```
## Collect the votes
votesCreated, voteProbability = customVoteEnsemble(x_out, clfL, featuresList)

## Compute the AUC
rocFinal = roc_auc_score(y_out, voteProbability)

## Compute accuracy
metrics.accuracy_score(y_out, convertToPlurality(voteProbability))

# Generate curve
fpr, tpr, accuracyNovel = roc_curve_custom(y_out == 'Primary', voteProbability)
tpr, fpr = refineROCCurveDataToLists(tpr, fpr)

# Create the plot
plotROCTree(fpr, tpr, rocFinal)

# Reveal Out of Bag Score
if printOut:
   print('Novel \n')
   print(sum(correctOOB)/sum(totalOOB))
   print(decodeOOBError(outOfBag, rawCorrect))

   # Reveal Feature Importances
   print(giniImportance)

# Plot importances
plotImportancesWithBars(giniImportance, giniImportanceFout, indiciesToUse)

################################################################################
"""
Established Ensemble (Random Forest) - Execute
"""
################################################################################

# Perform random forest classification
## Instantiate a random forest classifier object
### There will be 10,000 trees and no parallel processing
clfb = RandomForestClassifier(n_estimators=1000,
                      min_samples_leaf=minimumNumberPerLeaf,
                      random_state=randomMaster,
                      oob_score=True,
                      n_jobs=-1)

randomMaster += 1

## Train the blank random forest classifier object
clfb.fit(x_use_f, y_use_f)

# Apply The Full Featured Classifier To The Test Data
y_predr = clfb.predict_proba(x_out_f)[:,1]

## Compute the AUC
rocFinalr = roc_auc_score(y_out_f, y_predr)

## Compute accuracy
metrics.accuracy_score(y_out, convertToPlurality(y_predr))

## Create curve data
fprr, tprr, _ = metrics.roc_curve(y_out_f == 'Primary',  y_predr)

# Create the plot
plotROCTree(fprr, tprr, rocFinalr)

# Reveal Out of Bag Score
if printOut:
   print('\n')
   print('Established \n')
   print(clfb.oob_score_)

   # Reveal Feature Importances
   print(clfb.feature_importances_)

giniImportanceFoutX = []
for newElement in clfb.estimators_: giniImportanceFoutX.append(newElement.feature_importances_)

# Plot importances
plotImportancesWithBars(clfb.feature_importances_, giniImportanceFout, indiciesToUse)

################################################################################
"""
Established Ensemble (New Approach) - Execute
"""
################################################################################

if runNewApproach == True:
   # Generate the ensemble
```

```
    clfLX, featuresListX, outOfBagX, vroobX, rawCorrectX, correctOOBX, totalOOBX, giniImportanceX, giniImportanceFoutX, _, randomMaster, numUsedIterX, _, _, _, _, _ =
customGenerateEnsemble(x_use_f, y_use_f, randomMaster, numberTrees = desiredTreeNumberSelect)

    # Importance calculate
    ### Extract Importances
    # Assign weights to importance estimations depending on missingness
    if importanceAdjustment == 'Avg':
      giniImportanceX, giniImportanceFoutX = importanceWeighter(clfLX, numUsedIterX)
    elif importanceAdjustment == 'Raw':
      giniImportanceX, giniImportanceFoutX = importanceWeighter(clfLX)
    else:
      giniImportanceX, giniImportanceFoutX = importanceWeighter(clfLX)

    ## Collect the votes
    votesCreatedX, voteProbabilityX = customVoteEnsemble(x_out_f, clfLX, featuresListX)

    ## Compute the AUC
    rocFinalX = roc_auc_score(y_out_f, voteProbabilityX)

    ## Compute accuracy
    metrics.accuracy_score(y_out, convertToPlurality(voteProbabilityX))

    # Generate curve
    fprX, tprX, accuracyNovelX = roc_curve_custom(y_out_f == 'Primary', voteProbabilityX)
    tprX, fprX = refineROCCurveDataToLists(tprX, fprX)

    # Create the plot
    plotROCTree(fprX, tprX, rocFinalX)

    # Reveal Out of Bag Score
    if printOut:
      print('\n')
      print('Modified Approach \n')
      print(sum(correctOOBX)/sum(totalOOBX))
      print(decodeOOBError(outOfBagX, rawCorrectX))

      # Reveal Feature Importances
      print(giniImportanceX)

##############################################################################
"""
Support Vector Machines / Regression - Execute
"""
##############################################################################

# Instantiate a svm Classifier with a linear kernel
clfsb = svm.SVC(kernel='linear', probability=True)

# Train the model using the training sets
clfsb.fit(x_use_f, y_use_f)

# Predict the response for test dataset
y_preds = clfsb.predict_proba(x_out_f)[::,1]

## Compute the AUC
rocFinals = roc_auc_score(y_out_f, y_preds)

## Compute the AUC
metrics.accuracy_score(y_out, convertToPlurality(y_preds))

## Create curve data
fprs, tprs, _ = metrics.roc_curve(y_out_f == 'Primary',  y_preds)

# Create the plot
plotROCTree(fprs, tprs, rocFinals)

##############################################################################
"""
Logistic Regression
"""
##############################################################################

# # Instantiate a svm Classifier with a linear kernel
# clflr = LogisticRegression()

# # Train the model using the training sets
# clflr.fit(x_use_f, y_use_f)

# # Predict the response for test dataset
# y_preds = clflr.predict_proba(x_out_f)[::,1]

# ## Compute the AUC
# lgrAUCROC = roc_auc_score(y_out_f, y_preds)

# ## Compute accuracy
# lgrAcc = metrics.accuracy_score(x_out_f, convertToPlurality(y_preds))

# ## Create curve data
```

```
    # lgrfpr, lgrtpr, _ = metrics.roc_curve(y_out_f == 'Primary',  y_preds)

    # # Create the plot
    # plotROCTree(lgrfpr, lgrtpr, lgrAUCROC)

################################################################################
"""
Permutation importance - Function
"""
################################################################################

# Analyse permutation importance
def permutationImportance(xDataIn, xDataNull, yDataIn, featuresListed,
                          abstainModel, randomForestModel, svmModel,
                          iterationsSelected=5, progressBarTemplate = progressBars,
                          seedSelectRandom=1):

    # Initialize progress bar using with statement - will use start statement later in more complex function
    with progressbar.ProgressBar(max_value=xDataIn.shape[1]*iterationsSelected+1, widgets=progressBarTemplate) as bar:

        # Initialize the progress bar
        bar.update(0)
        progressPoint = 1

        # Importance accumulators
        svImportanceList = []
        rfImportanceList = []
        ceImportanceList = []
        random_augmenter = 0

        # Baseline Accuracy
        svBa = metrics.accuracy_score(yDataIn, svmModel.predict(xDataIn))
        rfBa = metrics.accuracy_score(yDataIn, randomForestModel.predict(xDataIn))
        ceBa = metrics.accuracy_score(yDataIn, customVoteEnsemblePlurality(xDataNull,
                                                abstainModel,
                                                featuresListed))

        # Neutralize Indicies
        xInFull = xDataIn.copy()
        xInFull.index = range(len(xInFull))

        xInNull = xDataNull.copy()
        xInNull.index = range(len(xInNull))

        for featureIndex in range(0,xDataIn.shape[1],1):

            # Feature Lists
            svImportanceFeatureList = []
            rfImportanceFeatureList = []
            ceImportanceFeatureList = []

            # Get plausible indicies
            indexSelections = findNonNan(xInFull, featureIndex)
            indexAvailable = indexSelections.copy()

            for repNumber in range(0,iterationsSelected,1):

                # Show Progress
                bar.update(progressPoint)
                progressPoint += 1

                # Initialize run data
                xIteration = xInFull.copy()
                xIterationNull = xInNull.copy()

                # Current random state
                currentRandomState = seedSelectRandom + random_augmenter

                # Make random selections
                rngFunc=np.random.RandomState(currentRandomState)
                rngFunc.shuffle(indexSelections)

                # Update Augmenter
                random_augmenter += 1

                # Adapt permuted frames
                xIteration.iloc[indexAvailable,featureIndex] = xIteration.iloc[indexSelections,featureIndex].values
                xIterationNull.iloc[indexAvailable,featureIndex] = xIterationNull.iloc[indexSelections,featureIndex].values

                # Iterative Accuracies
                svImportanceFeatureList.append(svBa - metrics.accuracy_score(yDataIn, svmModel.predict(xIteration)))
                rfImportanceFeatureList.append(rfBa - metrics.accuracy_score(yDataIn, randomForestModel.predict(xIteration)))
                ceImportanceFeatureList.append(ceBa - metrics.accuracy_score(yDataIn, customVoteEnsemblePlurality(xIterationNull,
                                                abstainModel,
                                                featuresListed)))

            # Accumulate by features
            svImportanceList.append(svImportanceFeatureList)
```

```
        rfImportanceList.append(rfImportanceFeatureList)
        ceImportanceList.append(ceImportanceFeatureList)

        # Finish bar
        bar.update(progressPoint)

    return svImportanceList, rfImportanceList, ceImportanceList, svBa, rfBa, ceBa, (seedSelectRandom + random_augmenter)

# Perform bootstrap on a list of values
# https://www.datacamp.com/community/tutorials/bootstrap-r
def meanListsBootstrap(listSet, baseEstimatesWithUnshuffledData):

    # Initialize output list
    outList = []

    for listIn in listSet:

        #Initialize output
        bootsList = []

        # Find bootstrap metrics
        bootEst = np.mean(listIn)

        # note: s/sqrt(n) is used for error in the stock python method
        bootSe = statistics.stdev(listIn)*np.sqrt(len(listIn)-1)/np.sqrt(len(listIn))

        # Persist metrics
        bootsList.append(bootEst)
        bootsList.append(bootSe)

        # Find non-wald / bootstrap bias adjusted intervals
        ## Note these intervals do not apply because this is not a random sampling of full cases (we have only redistributed one column)
        #bootsList.append(2*baseEstimatesWithUnshuffledData - bootEst - 1.96*bootSe)
        #bootsList.append(2*baseEstimatesWithUnshuffledData - bootEst + 1.96*bootSe)

        outList.append(bootsList)

    return(outList)

# Combine lists into array
def createDataFrameFromListOfLists(listOfListsIn):
    return pd.DataFrame(np.transpose(np.array(listOfListsIn)))

# Melt an Importance frame
def meltImportanceFrame(importanceFrameIn, varNameIn='Feature', valueNameIn='Importance'):
    return pd.melt(importanceFrameIn, var_name=varNameIn, value_name=valueNameIn)

# Attach melted frames
def attachMeltedFrames(meltedFrameIn1, meltedFrameIn2, indexingColumnIn = 'Method', f1Identity = '1', f2Identity = '2'):

    outputFrameOut = meltedFrameIn1.copy()
    attachFrameFun = meltedFrameIn2.copy()

    if not(indexingColumnIn in outputFrameOut.columns.tolist()):
        outputFrameOut[indexingColumnIn] = f1Identity
    if not(indexingColumnIn in attachFrameFun.columns.tolist()):
        attachFrameFun[indexingColumnIn] = f2Identity

    return outputFrameOut.append(attachFrameFun)

################################################################################
"""
Permutation importance - Execute
"""
################################################################################

if (singleIter == True) and (permuteImport == 'Custom') and (permuteImportGo == True):
    svImport, rfImport, ceImport, svBase, rfBase, ceBase, randomMaster = permutationImportance(x_out_f,
                                                x_out,
                                                y_out_f,
                                                featuresList,
                                                clfL,
                                                clfb,
                                                clfsb,
                                                iterationsSelected=permutationIterationsSelected,
                                                progressBarTemplate = progressBars,
                                                seedSelectRandom=randomMaster)

    # Generate Plot
    pImportNovel = meltImportanceFrame(createDataFrameFromListOfLists(ceImport))
    pImportStock = meltImportanceFrame(createDataFrameFromListOfLists(rfImport))
    pImportSVMce = meltImportanceFrame(createDataFrameFromListOfLists(svImport))

    pImportPlots = attachMeltedFrames(pImportNovel, pImportStock, indexingColumnIn = 'Method', f1Identity = 'Novel', f2Identity = 'Stock')
    pImportPlots = attachMeltedFrames(pImportPlots, pImportSVMce, indexingColumnIn = 'Method', f1Identity = '', f2Identity = 'SVM')

    sns.boxplot(x="Feature", y="Importance", hue="Method", data=pImportPlots, linewidth=1.5, palette="cubehelix").set(title='Permutation Importance by Method')
```

```
# Alternative approaches
elif (singleIter == True) and (permuteImportGo == True):
    pISRF = permutation_importance(clfb, x_out_f, y_out_f, scoring = 'roc_auc', n_repeats=permutationIterationsSelected, random_state=0)
    pISVM = permutation_importance(clfsb, x_out_f, y_out_f, scoring = 'roc_auc', n_repeats=permutationIterationsSelected, random_state=0)

# Get bootstrap error
#svInfo = pd.DataFrame(meanListsBootstrap(svImport, 0))
#rfInfo = pd.DataFrame(meanListsBootstrap(rfImport, 0))
#ceInfo = pd.DataFrame(meanListsBootstrap(ceImport, 0))

################################################################################
"""
Process Multiple Curves - Functions
"""
################################################################################

# Removes unnecessary points from ROC curves
def refineROCCurveData(tprIn, fprIn):

    # Adjust curve
    frameIROC = pd.DataFrame({'a': fprIn.tolist(), 'b': tprIn.tolist()})
    frameXROC = frameIROC.sort_values(['a', 'b'], axis=0, ascending=True).reset_index(drop = True)

    newA = []
    newB = []

    # Find and drag forward points to allow for a single x-value column
    for indexOn in range(0,len(frameXROC)-1,1):
        if frameXROC.b[indexOn] == frameXROC.b[indexOn + 1]:
            if frameXROC.a[indexOn] != frameXROC.a[indexOn + 1]:
                if newB == []:
                    newA += [frameXROC.a[indexOn]]
                    newB += [frameXROC.b[indexOn]]
                elif newB[-1] != frameXROC.b[indexOn]:
                    newA += [frameXROC.a[indexOn]]
                    newB += [frameXROC.b[indexOn]]
        else:
            if frameXROC.a[indexOn] == frameXROC.a[indexOn + 1]:
                if newA != []:
                    if newA[-1] != frameXROC.a[indexOn]:
                        newA += [frameXROC.a[indexOn]]
                        newB += [frameXROC.b[indexOn]]
            else:
                newA += [frameXROC.a[indexOn]]
                newB += [frameXROC.b[indexOn]]
                newA += [frameXROC.a[indexOn + 1]]
                newB += [frameXROC.b[indexOn]]

    newA += [frameXROC.a.tolist()[-1]]
    newB += [frameXROC.b.tolist()[-1]]

    frameFunctROC = pd.DataFrame({'a': newA, 'b': newB})
    return frameFunctROC

# Compress duplicate points
def compressDuplicateXs(frameXROC):

    rowKeep = []
    uniqueX = list(frameXROC.a.unique())

    for currentUnique in uniqueX:

        # Find the rows for which the current unique value is found
        currentRows = [i for i, x in enumerate(frameXROC.a == currentUnique) if x]

        if len(currentRows) > 1:
            rowKeep += [currentRows[0]] + [currentRows[-1]]

        elif len(currentRows) == 1:
            rowKeep += [currentRows[0]]

    return frameXROC.loc[rowKeep].reset_index(drop = True)

# Merge two frames where there are different dependent values
def mergeNAs(pdA1, pdA2):

    # combine the prior x values in series and drop the x value columns from the original matrix
    newA = pdA1.a.values.tolist() + pdA2.a.values.tolist()
    dropPdA1 = pdA1.drop(['a'], axis = 1)
    dropPdA2 = pdA2.drop(['a'], axis = 1)

    dropPdA1.columns = list(range(dropPdA1.shape[1]))
    dropPdA2.columns = list(range(dropPdA2.shape[1]))

    # Create buffer segements to add to the begining / end of series to be merged
    a1Add = pd.DataFrame(index=range(dropPdA2.shape[0]),columns=range(dropPdA1.shape[1]))
    a2Add = pd.DataFrame(index=range(dropPdA1.shape[0]),columns=range(dropPdA2.shape[1]))
```

```
    # Snap on the buffer series to the beginning or the end for the y values to ensure chte matrix matches the y values to orginal x values
    CR1 = dropPdA1.append(a1Add).reset_index(drop = True)
    CR2 = a2Add.append(dropPdA2).reset_index(drop = True)

    # Combine the two matricies of y values
    filledDataFrame = CR1.merge(CR2, left_index = True, right_index = True)

    # Add the x values / a column back in and rename the columns
    newCols = ['a'] + list(range(filledDataFrame.shape[1]))
    filledDataFrame.insert(0,'a',newA)
    filledDataFrame.columns = newCols

    # Sort the frame to ensure the x values (i.e., a values) are ordered
    mergedDataFrameSort = filledDataFrame.sort_values(newCols, axis=0, ascending=True).reset_index(drop = True)

    # Fill in the buffer values to ensure sharp steps
    filledDataFrame = mergedDataFrameSort.fillna(method='ffill').fillna(method='bfill').drop_duplicates().reset_index(drop = True)

    return compressDuplicateXs(filledDataFrame)

# Merge two lists
def mergeLists(frameInFunct, ListsInFunct):

    # format add frame
    dfAdder = pd.DataFrame({'a': ListsInFunct[0],
                    frameInFunct.shape[1]: ListsInFunct[1]}).sort_values(by=['a',frameInFunct.shape[1]],
                                                    ascending=[True, True])

    sortingList = ['a'] + list(range(1,(frameInFunct.shape[1]+1)))

    initialreturn = pd.merge(frameInFunct, dfAdder, on="a", how="outer").sort_values(by=sortingList, ascending=[True]*len(sortingList)).reset_index(drop = True)

    deduplicatedReturn = compressDuplicateXs(initialreturn)

    return deduplicatedReturn

# Fill with a linear interprolation
def fillLinear(frameInFunct):

    # find the bookend y values
    lastY = frameInFunct.fillna(method='bfill')
    firstY = frameInFunct.fillna(method='ffill')

    # Create a frame with x data but preserving missingness
    xData = ((frameInFunct + 1) / (frameInFunct + 1))
    xFullFilled = pd.DataFrame(np.array([list(frameInFunct.a)]*frameInFunct.shape[1]).T)

    # Creat matrix with current x values
    xCurrent = xFullFilled.rename(columns={0:"a"})

    # Create frame for extremes
    xBase = xData * xCurrent.copy()

    # find the bookend x values
    lastX = xBase.fillna(method='bfill')
    firstX = xBase.fillna(method='ffill')

    # Compute slopes between points
    slopesFunct = ((lastY - firstY) / (lastX - firstX)).fillna(0)

    # Interprolate
    outputFunct = firstY + slopesFunct * (xCurrent - firstX)
    outputFunct.a = frameInFunct.a

    return outputFunct

#tyu = pd.DataFrame({'a': nAc[0][0], 1: nAc[0][1]})

# Find the limitartion curve regions given input test runs
def curveStatsForPlot(pdInFunct, LowCentile = selectLowCentile, HighCentile = selectHighCentile):

    prepedFrame = pdInFunct.drop(['a'], axis=1)

    xCol = pdInFunct.a
    meanCol = prepedFrame.mean(axis=1)
    quantL = prepedFrame.quantile(LowCentile,axis=1)
    quantH = prepedFrame.quantile(HighCentile,axis=1)

    return pd.DataFrame({'falsePositiveRate': xCol.values,
                    'mean': meanCol.values,
                    'LowCentile': quantL.values,
                    'HighCentile': quantH.values})

################################################################################
"""
Process Multiple Curves - Chaining Functions
"""
################################################################################
```

```
# Create single iteration of complete data run / analysis run
def singleIterationFit(x_use_Funct, y_use_Funct, x_out_Funct, y_out_Funct, x_use_f_Funct, y_use_f_Funct, x_out_f_Funct, y_out_f_Funct, randomMaster_Funct = 0, desiredTreeNumber =
desiredTreeNumberSelect, minPerLeafIn = minimumNumberPerLeaf):

    ##########################################################################
    """
    Novel
    """
    ##########################################################################

    # Generate the ensemble
    clfL, featuresList, outOfBag, _, rawCorrect, _, _, novelGini, giniImportanceFNovel, _, randomMaster_Funct, numUsedOut, numRetainedOut, novelLockOut, novelLockOutX, novelNonLockOut,
allOfOneTypeOut = customGenerateEnsemble(x_use_Funct, y_use_Funct, randomMaster_Funct, numberTrees = desiredTreeNumber)

    # Update Random seed
    randomMaster_Funct += 1

    # Capture out of bag error
    novelOOB = decodeOOBError(outOfBag, rawCorrect)

    ## Collect the votes
    votesCreated, voteProbability = customVoteEnsemble(x_out_Funct, clfL, featuresList)

    ## Compute the AUC
    novelAUC = roc_auc_score(y_out_Funct, voteProbability)

    ## Compute accuracy
    novelAcc = metrics.accuracy_score(y_out_Funct, convertToPlurality(voteProbability))

    # Generate curve
    # novelfpr, noveltpr, _ = roc_curve_custom(y_out_Funct == 'Primary', voteProbability)
    novelfpr, noveltpr, _ = metrics.roc_curve(y_out_f_Funct == 'Primary', voteProbability)

    ##########################################################################
    """
    Stock Random Forest
    """
    ##########################################################################

    # Perform random forest classification
    ## Instantiate a random forest classifier object
    ### There will be 10,000 trees and no parallel processing
    clfb = RandomForestClassifier(n_estimators=desiredTreeNumber,
                          min_samples_leaf=minPerLeafIn,
                          random_state=randomMaster_Funct,
                          oob_score=True,
                          n_jobs=-1)

    # Update Random seed
    randomMaster_Funct += 1

    ## Train the blank random forest classifier object
    clfb.fit(x_use_f_Funct, y_use_f_Funct)

    ## Extract out of bag error
    stockRFOOB = clfb.oob_score_

    # Apply The Full Featured Classifier To The Test Data
    y_predr_Funct = clfb.predict_proba(x_out_f_Funct)[::,1]

    ## Compute the AUC
    stockRFAUC = roc_auc_score(y_out_f_Funct, y_predr_Funct)

    ## Compute accuracy
    stockAcc = metrics.accuracy_score(y_out_Funct, convertToPlurality(y_predr_Funct))

    ## Create curve data
    stockRFfpr, stockRFtpr, _ = metrics.roc_curve(y_out_f_Funct == 'Primary', y_predr_Funct)

    stockGini = clfb.feature_importances_

    ##########################################################################
    """
    Manual Random Forest
    """
    ##########################################################################

    # Generate the ensemble
    clfLX, featuresListX, outOfBagX, _, rawCorrectX, _, _, manualGini, giniImportanceFManual, _, randomMaster_Funct, _, _, _, _, _, _ = customGenerateEnsemble(x_use_f_Funct, y_use_f_Funct,
randomMaster_Funct, numberTrees = desiredTreeNumber)

    # Update Random seed
    randomMaster_Funct += 1

    # Capture out of bag error
    manualRFOOB = decodeOOBError(outOfBagX, rawCorrectX)
```

```
    ## Collect the votes
    votesCreatedX, voteProbabilityX = customVoteEnsemble(x_out_f_Funct, clfLX, featuresListX)

    ## Compute the AUC
    manualRFAUC = roc_auc_score(y_out_f_Funct, voteProbabilityX)

    ## Compute accuracy
    manualRFAcc = metrics.accuracy_score(y_out_Funct, convertToPlurality(voteProbabilityX))

    # Generate curve
    # manualRFfpr, manualRFtpr, _ = roc_curve_custom(y_out_f_Funct == 'Primary', voteProbabilityX)
    manualRFfpr, manualRFtpr, _ = metrics.roc_curve(y_out_f_Funct == 'Primary', voteProbabilityX)

    ###############################################################################
    """
    Support Vector Machine
    """
    ###############################################################################

    # Instantiate a svm Classifier with a linear kernel
    clfsb = svm.SVC(kernel='linear', probability=True)

    # Train the model using the training sets
    clfsb.fit(x_use_f_Funct, y_use_f_Funct)

    # Predict the response for test dataset
    y_preds_Funct = clfsb.predict_proba(x_out_f_Funct)[::,1]

    ## Compute the AUC
    svmAUC = roc_auc_score(y_out_f_Funct, y_preds_Funct)

    ## Compute accuracy
    svmAcc = metrics.accuracy_score(y_out_Funct, convertToPlurality(y_preds_Funct))

    ## Create curve data
    svmfpr, svmtpr, _ = metrics.roc_curve(y_out_f_Funct == 'Primary',  y_preds_Funct)

    ###############################################################################
    """
    Logistic Regression
    """
    ###############################################################################

    # Instantiate a svm Classifier with a linear kernel
    clflr = LogisticRegression()

    # Train the model using the training sets
    clflr.fit(x_use_f_Funct, y_use_f_Funct)

    # Predict the response for test dataset
    y_preds_Funct = clflr.predict_proba(x_out_f_Funct)[::,1]

    ## Compute the AUC
    lgrAUC = roc_auc_score(y_out_f_Funct, y_preds_Funct)

    ## Compute accuracy
    lgrAcc = metrics.accuracy_score(y_out_Funct, convertToPlurality(y_preds_Funct))

    ## Create curve data
    lgrfpr, lgrtpr, _ = metrics.roc_curve(y_out_f_Funct == 'Primary',  y_preds_Funct)

    return clfL, novelAUC, novelOOB, novelAcc, noveltpr, novelfpr, novelGini, giniImportanceFNovel, novelLockOut, novelLockOutX, novelNonLockOut, numUsedOut, clfb, stockRFAUC,
stockRFOOB, stockAcc, stockRFtpr, stockRFfpr, stockGini, clfLX, manualRFAUC, manualRFOOB, manualRFAcc, manualRFtpr, manualRFfpr, manualGini, giniImportanceFManual, clfsb, svmAUC,
svmAcc, svmtpr, svmfpr, clflr, lgrAUC, lgrAcc, lgrtpr, lgrfpr, randomMaster_Funct, featuresList, allOfOneTypeOut

## Generate stats on a list
def listStatistics (listInFunct, lowerCent, upperCent):
    return np.mean(listInFunct), np.percentile(listInFunct, lowerCent), np.percentile(listInFunct, upperCent)

## Prepare new data
def cleanData (numberFeaturesFunct = 13, numberCasesFunct = 178, paramArrayFunct = pd.DataFrame(), targetIn = [], randomMasterFunct = 0):

    deFunct, paramArrayFunct, cat_entitiesFunct, randomMasterFunct = dataCreationLine(numberFeaturesFunct, numberCasesFunct, paramArrayFunct = paramArrayFunct, targetIn = targetIn,
randomMasterFunct = randomMasterFunct)

    x_use_Funct, x_out_Funct, y_use_Funct, y_out_Funct, x_use_f_Funct, y_use_f_Funct, x_out_f_Funct, y_out_f_Funct, randomMasterFunct = dataPackageGeneration(deFunct,
                                                                                                                                                            targetIn,
                                                                                                                                                            cat_entitiesFunct,
                                                                                                                                                            randomMasterFunct)

    return x_use_Funct, x_out_Funct, y_use_Funct, y_out_Funct, x_use_f_Funct, y_use_f_Funct, x_out_f_Funct, y_out_f_Funct, paramArrayFunct, randomMasterFunct, deFunct

# Run simulations multiple times to generate ROC curve
def curveGenerationRoutine (dataFrameIn = pd.DataFrame(), paramArrayFunct = pd.DataFrame(), numFeaturesIn = 13, numberCasesIn = 178, sparsitySelect = 0.5, shuffleSelect = 0.5,
desiredIterations = 100, progressBarTemplate = progressBars, randomSeedFunct = randomMaster + 10000):

    if desiredIterations < 1:
        desiredIterations = 1
```

```
    # Initialize progress bar
    bar = progressbar.ProgressBar(max_value=desiredIterations*2, widgets=progressBarTemplate).start()
    bar.update(0)

    # Initalize Accumulators
    novelOOBAccumulator = []
    novelAUCAccumulator = []
    novelAccAccumulator = []
    novelRegAccumulator = []
    novelAPRAccumulator = []

    stockRFOOBAccumulator = []
    stockRFAUCAccumulator = []
    stockRFAccAccumulator = []
    stockRFAPRAccumulator = []

    manualRFOOBAccumulator = []
    manualRFAUCAccumulator = []
    manualRFAccAccumulator = []
    manualRFRegAccumulator = []
    manualRFAPRAccumulator = []

    svmAUCAccumulator = []
    svmAccAccumulator = []
    svmAPRAccumulator = []

    lgrAUCAccumulator = []
    lgrAccAccumulator = []
    lgrAPRAccumulator = []

    featuresSelectedComplete = []

    # Store random seed with data accumulation to allow for recreation
    dataRandomSeedAccumulator = [randomSeedFunct]

    # Generate initial data
    if dataFrameIn.shape[0] == 0:

        xFuse, xFout, yFuse, yFout, xFuse_f, yFuse_f, xFout_f, yFout_f, paramArrayFunct, randomSeedFunct, _ = cleanData (numberFeaturesFunct = numFeaturesIn,
                                                                                                                          numberCasesFunct = numberCasesIn,
                                                                                                                          paramArrayFunct = paramArrayFunct,
                                                                                                                          targetIn = target,
                                                                                                                          randomMasterFunct = randomSeedFunct)

    else:
        if bootstrap:
            xFuse, xFout, yFuse, yFout, xFuse_f, yFuse_f, xFout_f, yFout_f, randomSeedFunct = bootSelection(dataFrameFunctIn = dataFrameIn,
                                                                                                            targetIn  = target,
                                                                                                            cat_entitiesFunct = cat_entities,
                                                                                                            randomMasterFunct = randomSeedFunct)

        else:
            xFuse, xFout, yFuse, yFout, xFuse_f, yFuse_f, xFout_f, yFout_f, randomSeedFunct = dataPackageGeneration(dataFrameFunctIn = dataFrameIn,
                                                                                                                    targetIn  = target,
                                                                                                                    cat_entitiesFunct = cat_entities,
                                                                                                                    randomMasterFunct = randomSeedFunct)

    # Run iteration
    randomSeedFunct = randomSeedFunct + 100000

    # Update Bar
    bar.update(1)

    # Adjust input data if training with filled data
    if runTrainWithFill == True:
        xFuse = xFuse_f

    # First run to initialize
    singleResult = singleIterationFit(xFuse,
                                      yFuse,
                                      xFout,
                                      yFout,
                                      xFuse_f,
                                      yFuse_f,
                                      xFout_f,
                                      yFout_f,
                                      randomSeedFunct)

    novelRegi, novelAUCi, novelOOBi, novelAcci, noveltpri, novelfpri, novelGinii, novelFGinii, novelLockOuti, novelLockOutXi, novelNonLockOuti, novelNumUsedi, stockRegi, stockRFAUCi,
stockRFOOBi, stockRFAcci, stockRFtpri, stockRFfpri, stockGinii, manualRegi, manualRFAUCi, manualRFOOBi, manualRFAcci, manualRFtpri, manualRFfpri, manualGinii, manualFGinii, svmRegi,
svmAUCi, svmAcci, svmtpri, svmfpri, lgrRegi, lgrAUCi, lgrAcci, lgrtpri, lgrfpri, randomSeedFunct, featuresListi, allOfOneTypei = singleResult

    # Run and store novel technique
    novelRegAccumulator += [novelRegi]
    nRFROCdf = refineROCCurveData(noveltpri, novelfpri)
    novelAPRAccumulator += [[noveltpri, novelfpri]]
    novelOOBAccumulator += [novelOOBi]
```

```
novelAUCAccumulator += [novelAUCi]
novelAccAccumulator += [novelAcci]
novelImpAccumulator = pd.DataFrame(novelGinii)
allOfOneTypeAccumulator = [np.array(allOfOneTypei)]
novelLockOutAccumulator = [novelLockOuti]
novelLockOutXAccumulator = [novelLockOutXi]
novelNonLockOutAccumulator = [novelNonLockOuti]
novelNumUsed = [novelNumUsedi]

novelFImpAccumulator = [novelFGinii]

# Run and store stock technique
sRFROCdf = refineROCCurveData(stockRFtpri, stockRFfpri)
stockRFAPRAccumulator += [[stockRFtpri, stockRFfpri]]
stockRFOOBAccumulator += [stockRFOOBi]
stockRFAUCAccumulator += [stockRFAUCi]
stockRFAccAccumulator += [stockRFAcci]
stockRFImpAccumulator = pd.DataFrame(stockGinii)

# Run and store manual technique
manualRFRegAccumulator += [manualRegi]
mRFROCdf = refineROCCurveData(manualRFtpri, manualRFfpri)
manualRFAPRAccumulator += [[manualRFtpri, manualRFfpri]]
manualRFOOBAccumulator += [manualRFOOBi]
manualRFAUCAccumulator += [manualRFAUCi]
manualRFAccAccumulator += [manualRFAcci]
manualRFImpAccumulator = pd.DataFrame(manualGinii)
manualFImpAccumulator = [manualFGinii]

# Run and store support vector machine technique
mSVROCdf = refineROCCurveData(svmtpri, svmfpri)
svmAPRAccumulator += [[svmtpri, svmfpri]]
svmAUCAccumulator += [svmAUCi]
svmAccAccumulator += [svmAcci]

# Run and store support logistic regression technique
mLRROCdf = refineROCCurveData(lgrtpri, lgrfpri)
lgrAPRAccumulator += [[lgrtpri, lgrfpri]]
lgrAUCAccumulator += [lgrAUCi]
lgrAccAccumulator += [lgrAcci]

# Keep features
featuresSelectedComplete.append(featuresListi)

# Update UI Bar
bar.update(2)

if desiredIterations > 1:

    # Set runs
    for i in range(desiredIterations - 1):

        # Store random seed with data accumulation to allow for recreation
        dataRandomSeedAccumulator += [randomSeedFunct]

        # Regenerate data
        if dataFrameIn.shape[0] == 0:

            xFuse, xFout, yFuse, yFout, xFuse_f, yFuse_f, xFout_f, yFout_f, _, randomSeedFunct, _ = cleanData (numberFeaturesFunct = numFeaturesIn,
                                                                    numberCasesFunct = numberCasesIn,
                                                                    paramArrayFunct = paramArrayFunct,
                                                                    targetIn = target,
                                                                    randomMasterFunct = randomSeedFunct)

        else:
            if bootstrap:
                xFuse, xFout, yFuse, yFout, xFuse_f, yFuse_f, xFout_f, yFout_f, randomSeedFunct = bootSelection(dataFrameFunctIn = dataFrameIn,
                                                                        targetIn  = target,
                                                                        cat_entitiesFunct = cat_entities,
                                                                        randomMasterFunct = randomSeedFunct)
            else:
                xFuse, xFout, yFuse, yFout, xFuse_f, yFuse_f, xFout_f, yFout_f, randomSeedFunct = dataPackageGeneration(dataFrameFunctIn = dataFrameIn,
                                                                        targetIn  = target,
                                                                        cat_entitiesFunct = cat_entities,
                                                                        randomMasterFunct = randomSeedFunct)

        # Run iteration
        randomSeedFunct = randomSeedFunct + 100000

        # Update UI Bar
        bar.update((i + 1)*2 + 1)

        # Adjust input data if training with filled data
        if runTrainWithFill == True:
            xFuse = xFuse_f

        # Run another set
        singleResult = singleIterationFit(xFuse,
```

```
                        yFuse,
                        xFout,
                        yFout,
                        xFuse_f,
                        yFuse_f,
                        xFout_f,
                        yFout_f,
                        randomSeedFunct)

        novelRegi, novelAUCi, novelOOBi, novelAcci, noveltpri, novelfpri, novelGinii, novelFGinii, novelLockOuti, novelLockOutXi, novelNonLockOuti, novelNumUsedi, stockRegi, stockRFAUCi, stockRFOOBi, stockRFAcci, stockRFtpri, stockRFfpri, stockGinii, manualRegi, manualRFAUCi, manualRFOOBi, manualRFAcci, manualRFtpri, manualRFfpri, manualGinii, manualFGinii, svmRegi, svmAUCi, svmAcci, svmtpri, svmfpri, lgrRegi, lgrAUCi, lgrAcci, lgrtpri, lgrfpri, randomSeedFunct, featuresListi, allOfOneTypei = singleResult

        # Accumulate Lockout
        novelLockOutAccumulator += [novelLockOuti]
        novelLockOutXAccumulator += [novelLockOutXi]
        novelNonLockOutAccumulator += [novelNonLockOuti]
        novelNumUsed += [novelNumUsedi]
        allOfOneTypeAccumulator += [np.array(allOfOneTypei)]

        # Store Variables
        novelRegAccumulator += [novelRegi]
        nRFROCdf = mergeNAs(nRFROCdf, refineROCCurveData(noveltpri, novelfpri))
        novelAPRAccumulator += [[noveltpri, novelfpri]]

        novelOOBAccumulator += [novelOOBi]
        novelAUCAccumulator += [novelAUCi]
        novelAccAccumulator += [novelAcci]

        novelImpAccumulator = novelImpAccumulator.merge(pd.DataFrame(novelGinii, columns=[('n_'+str(i))]), left_index = True, right_index = True)
        novelFImpAccumulator += [novelFGinii]

        sRFROCdf = mergeNAs(sRFROCdf, refineROCCurveData(stockRFtpri, stockRFfpri))
        stockRFAPRAccumulator += [[stockRFtpri, stockRFfpri]]

        stockRFOOBAccumulator += [stockRFOOBi]
        stockRFAUCAccumulator += [stockRFAUCi]
        stockRFAccAccumulator += [stockRFAcci]
        stockRFImpAccumulator = stockRFImpAccumulator.merge(pd.DataFrame(stockGinii, columns=[('n_'+str(i))]), left_index = True, right_index = True)

        manualRFRegAccumulator += [manualRegi]
        mRFROCdf = mergeNAs(mRFROCdf, refineROCCurveData(manualRFtpri, manualRFfpri))
        manualRFAPRAccumulator += [[manualRFtpri, manualRFfpri]]

        manualRFOOBAccumulator += [manualRFOOBi]
        manualRFAUCAccumulator += [manualRFAUCi]
        manualRFAccAccumulator += [manualRFAcci]
        manualRFImpAccumulator = manualRFImpAccumulator.merge(pd.DataFrame(manualGinii, columns=[('n_'+str(i))]), left_index = True, right_index = True)
        manualFImpAccumulator += [manualFGinii]

        mSVROCdf = mergeNAs(mSVROCdf, refineROCCurveData(svmtpri, svmfpri))
        svmAPRAccumulator += [[svmtpri, svmfpri]]
        svmAUCAccumulator += [svmAUCi]
        svmAccAccumulator += [svmAcci]

        mLRROCdf = mergeNAs(mLRROCdf, refineROCCurveData(lgrtpri, lgrfpri))
        lgrAPRAccumulator += [[lgrtpri, lgrfpri]]
        lgrAUCAccumulator += [lgrAUCi]
        lgrAccAccumulator += [lgrAcci]

        # Keep features
        featuresSelectedComplete.append(featuresListi)

        # Update UI Bar
        bar.update((i + 2)*2)

    # Move to new line after completing progress bar
    print('\n')

    # Adjust column names
    novelImpAccumulator.columns = range(novelImpAccumulator.shape[1])
    stockRFImpAccumulator.columns = range(stockRFImpAccumulator.shape[1])
    manualRFImpAccumulator.columns = range(manualRFImpAccumulator.shape[1])

    return novelRegAccumulator, nRFROCdf, novelAPRAccumulator, novelOOBAccumulator, novelAUCAccumulator, novelAccAccumulator, novelImpAccumulator, novelFImpAccumulator, novelLockOutAccumulator, novelLockOutXAccumulator, novelNonLockOutAccumulator, novelNumUsed, sRFROCdf, stockRFAPRAccumulator, stockRFOOBAccumulator, stockRFAUCAccumulator, stockRFAccAccumulator, stockRFImpAccumulator, manualRFRegAccumulator, mRFROCdf, manualRFAPRAccumulator, manualRFOOBAccumulator, manualRFAUCAccumulator, manualRFAccAccumulator, manualRFImpAccumulator, manualFImpAccumulator, mSVROCdf, svmAPRAccumulator, svmAUCAccumulator, svmAccAccumulator, mLRROCdf, lgrAPRAccumulator, lgrAUCAccumulator, lgrAccAccumulator, featuresSelectedComplete, allOfOneTypeAccumulator, dataRandomSeedAccumulator

################################################################################
"""
Process Multiple Curves - Execute
"""
################################################################################

# finds stats across rows
def listStatsByRow(dataFrameFunctIn, lowerCentIn = 2.5, higherCentIn = 97.5):
```

```
    outFrameFunct = pd.DataFrame(listStatistics(novelImpAccumulatoro.iloc[0].tolist(), lowerCentIn, higherCentIn))

    if dataFrameFunctIn.shape[0] > 1:
        for currentIndexFunct in range(dataFrameFunctIn.shape[0] - 1):

            frameIterFunct = pd.DataFrame(listStatistics(novelImpAccumulatoro.iloc[currentIndexFunct + 1].tolist(), lowerCentIn, higherCentIn))
            frameIterFunct.columns = [currentIndexFunct + 1]
            outFrameFunct = outFrameFunct.merge(frameIterFunct, left_index = True, right_index = True)

    return outFrameFunct.T

# Look at the distribution of importance calculations
def interogateImportances(importanceConstruct):
    importanceSummary = np.array([0]*len(importanceConstruct[0][0])).astype(float)
    for iterGroup in importanceConstruct:
        for tree in iterGroup:
            importanceSummary += (tree > 0).astype(float)

    return importanceSummary

# Initiate program
# outputMain = curveGenerationRoutine(paramArrayFunct = paramArray,
#                                     numFeaturesIn = 13,
#                                     numberCasesIn = 500,
#                                     desiredIterations = 100)

if runMainCurves:

    ## Set target
    ################################################################################
    if dataSetSelectForCurves == 'fabricate':
        dataFrameForProcessing = pd.DataFrame()
    elif dataSetSelectForCurves == 'data augment':
        dataFrameForProcessing = dataAugmentationToFrame(de)
    elif dataSetSelectForCurves == 'data through':
        dataFrameForProcessing = de
    else:
        dataFrameForProcessing = de
    ################################################################################

    # Enter an empty dataFrameIn for data fabriaction
    outputMain = curveGenerationRoutine(dataFrameIn = dataFrameForProcessing,
                                        paramArrayFunct = paramArray,
                                        numFeaturesIn = 13,
                                        numberCasesIn = 178,
                                        sparsitySelect = sparsitySelection,
                                        shuffleSelect = shuffleSelection,
                                        desiredIterations = desiredIterationsSelected,
                                        progressBarTemplate = progressBars,
                                        randomSeedFunct = randomMaster + 10000)

try:
    outputMain
except NameError:
    outputMain = None

if (outputMain is not None) and (runPlots):

    # Redistribute values
    novelRego, nRFROCdfo, nAc, novelOOBAccumulatoro, novelAUCAccumulatoro, novelAccAccumulatoro, novelImpAccumulatoro, novelFImpAccumulatoro, novelLockOutAccumulatorOut,
novelLockOutXAccumulatorOut, novelNonLockOutAccumulatorOut, novelNumUsedOut, sRFROCdfo, sAc, stockRFOOBAccumulatoro, stockRFAUCAccumulatoro, stockRFAccAccumulatoro,
stockRFImpAccumulatoro, manualRego, mRFROCdfo, mAc, manualRFOOBAccumulatoro, manualRFAUCAccumulatoro, manualRFAccAccumulatoro, manualRFImpAccumulatoro,
manualFImpAccumulatoro, mSVROCdfo, vAc, svmAUCAccumulatoro, svmAccAccumulatoro, mLRROCdfo, lAc, lgrAUCAccumulatoro, lgrAccAccumulatoro, featuresSelectedCompleteOut,
allOfOneTypeAccumulatorOut, dataRandomSeedsOut = outputMain

    novelImportancePerRun = []
    correctionFactors = []
    correctedNovelImportancePerRun = []

    #importanceAdjustment = 'Avg'
    treeImportanceCombination = 'raw score'
    treeImportanceCombination = 'mean'
    includeStockImportance = False

    ### Extract Importances
    # Assign weights to importance estimations depending on missingness
    if importanceAdjustment == 'Avg':
        for i in range(len(featuresSelectedCompleteOut)):
            novelImportancePerRuni, _ = importanceWeighter(novelRego[i], weightingIn=novelNumUsedOut[i], normalizeFunct=treeImportanceCombination)
            novelImportancePerRun += [novelImportancePerRuni]
    elif importanceAdjustment == 'Raw':
        for i in range(len(featuresSelectedCompleteOut)):
            novelImportancePerRuni, _ = importanceWeighter(novelRego[i], weightingIn=[], normalizeFunct=treeImportanceCombination)
            novelImportancePerRun += [novelImportancePerRuni]
    else:
        for i in range(len(featuresSelectedCompleteOut)):
            novelImportancePerRuni, _ = importanceWeighter(novelRego[i], weightingIn=[], normalizeFunct=treeImportanceCombination)
```

```
        novelImportancePerRun += [novelImportancePerRuni]

# Adjust importances
## Create adjustment factors if desired

for i in range(len(novelNonLockOutAccumulatorOut)):
    correctionFactors += [novelNonLockOutAccumulatorOut[i].trace()/(novelNonLockOutAccumulatorOut[i].shape[0]*novelNonLockOutAccumulatorOut[i].diagonal())]

# Note that there is some correction to factor 2 but no a lot compared to the other locked out factors (see lock out matricies)
print(sum(correctionFactors))

for i in range(len(novelNonLockOutAccumulatorOut)):
    adjustedImportanceI = novelImportancePerRun[i].T*correctionFactors[i]
    correctedNovelImportancePerRun += [adjustedImportanceI]

adjustedAccumulator = pd.DataFrame(correctedNovelImportancePerRun)

# if adjustmentFlag == 'First':

#     # Use the number of non lock outs to correct importance
#     adjustmentForTrees = mergedNovelNonLockOutAccumulatorOut.trace()/(mergedNovelNonLockOutAccumulatorOut.shape[0]*mergedNovelNonLockOutAccumulatorOut.diagonal())
#     adjustedAccumulator = novelImpAccumulatoro.T*adjustmentForTrees
# elif adjustmentFlag == 'Second':

#     # Use the number of entries in the importance array to adjust importance
#     numArrayAdjust=interogateImportances(novelFImpAccumulatoro)

#     # Prevent division by zero
#     numArrayAdjustSum = numArrayAdjust.sum()
#     numArrayAdjust = numArrayAdjust + (numArrayAdjust < 1).astype(float)

#     # Adjsut
#     adjustmentForTrees = numArrayAdjust.sum()/(len(numArrayAdjust)*numArrayAdjust)
#     adjustedAccumulator = novelImpAccumulatoro.T*adjustmentForTrees
# elif adjustmentFlag == 'Third':
#     keys = list(range(novelImpAccumulatoro.shape[0]))
#     values = [0]*len(keys)
#     numbersOut = Counter(dict(zip(keys, values)))
#     for i in range(len(featuresSelectedCompleteOut)):
#         for j in range(len(featuresSelectedCompleteOut[i])):
#             numbersOut.update(featuresSelectedCompleteOut[i][j])
# else:

#     # Do not adjust importance
#     adjustedAccumulator = novelImpAccumulatoro.T

# Combine Plots
featureImportancePlot=pd.melt(adjustedAccumulator, var_name='Feature', value_name='Gini Importance')
featureImportancePlot['Method']='Novel'
standardImportance=pd.melt(manualRFImpAccumulatoro.T, var_name='Feature', value_name='Gini Importance')
standardImportance['Method']='Standard'
featureImportancePlot = featureImportancePlot.append(standardImportance)

# Option to include stock
if includeStockImportance == True:
    stockImportance=pd.melt(stockRFImpAccumulatoro.T, var_name='Feature', value_name='Gini Importance')
    stockImportance['Method']='Stock'
    featureImportancePlot = featureImportancePlot.append(stockImportance)

# Correct indicies
featureImportancePlot.reset_index(drop=True)

# Make the plot
sns.boxplot(x="Feature", y="Gini Importance", hue="Method", data=featureImportancePlot, linewidth=1.5, palette="cubehelix").set(title='Importance by Forest Method')

# Accuracy Stats
nRFplotStats = curveStatsForPlot(nRFROCdfo)
sRFplotStats = curveStatsForPlot(sRFROCdfo)
mRFplotStats = curveStatsForPlot(mRFROCdfo)
svmplotStats = curveStatsForPlot(mSVROCdfo)
lgrplotStats = curveStatsForPlot(mLRROCdfo)

nOALH = listStatistics(novelOOBAccumulatoro, 2.5, 97.5)
nAALH = listStatistics(novelAUCAccumulatoro, 2.5, 97.5)

sOALH = listStatistics(stockRFOOBAccumulatoro, 2.5, 97.5)
sAALH = listStatistics(stockRFAUCAccumulatoro, 2.5, 97.5)

mOALH = listStatistics(manualRFOOBAccumulatoro, 2.5, 97.5)
mAALH = listStatistics(manualRFAUCAccumulatoro, 2.5, 97.5)

vAALH = listStatistics(svmAUCAccumulatoro, 2.5, 97.5)

lAALH = listStatistics(lgrAUCAccumulatoro, 2.5, 97.5)

listStatsByRow(novelImpAccumulatoro)

if printOut:
```

```
    print('\n')
    print('*** Simulation Statistics Summary ***')
    print("Novel OOB:", nOALH)
    print("Novel Accuracy:", listStatistics(novelAccAccumulatoro, 2.5, 97.5))

    print("Stock OOB:", sOALH)
    print("Stock Accuracy:", listStatistics(stockRFAccAccumulatoro, 2.5, 97.5))

    print("Manual OOB:", mOALH)
    print("Manual Accuracy:", listStatistics(manualRFAccAccumulatoro, 2.5, 97.5))

    print("SVM Accuracy:", listStatistics(manualRFAccAccumulatoro, 2.5, 97.5))

    print("Logistic Accuracy:", listStatistics(manualRFAccAccumulatoro, 2.5, 97.5))

plotROCTreeFill(nRFplotStats, nAALH, 'Novel Method ROC Curve')
plotROCTreeFill(sRFplotStats, sAALH, 'Stock Method ROC Curve')
plotROCTreeFill(mRFplotStats, mAALH, 'Standard Forest ROC Curve')
plotROCTreeFill(svmplotStats, vAALH, 'Support Vector Machine ROC Curve')
plotROCTreeFill(lgrplotStats, lAALH, 'Logistic ROC Curve')
```

XVI. Table of Figures

XVII. Glossary

ABRA	A-Beta Related Angiitis
alignment	the entry-wise summation of all mask terms that upon exponentiation of the sums of these terms which is equivalent to entry-wise multiplication of the exponentiated masks representing relative probabilities
ANA	Anti-nuclear Antibody
ANCA	Antineutrophil cytoplasmic antibody
APP	Amyloid Precursor Protein
Box and whisker plot[clx]	A plot where the central dash represents the median, the box represents the inter-quartile range (i.e., the central two quartiles or 50% of the data), the whiskers represent the extreme values not exceeding 1.5 times the inter-quartile range with respect to the distance from the median.
C3/4	compliment ¾
CAARI	Cerebral Amyloid Angiopathy Related Inflammation
CADASIL	Cerebral autosomal dominant arteriopathy with subcortical infarcts and leukoencephalopathy
CD	Cell differentiation
C_H	represents the class where there is a higher underlying value of the perfectly predictive feature
C_L	represents the class where there is a lower underlying value of the perfectly predictive feature
closed convex sets	sets containing all their limiting points
CNS	Central nervous system
COL4A	Collagen 4A
Common condition	a condition that is not rare (please see the definition of rare condition)
credal set	a set of probability distributions or possible probability distributions where the true distribution is not precisely known
CSF	Cerebro-spinal fluid
CT	Computed Tomography
data cleaning	the process of limiting the impacts on further analysis of missing or errant data within a source data set
descendent nodes	all nodes downstream to a particular node
DNA	Deoxy-ribonucleic acid
DSA	Digital Subtraction Angiography
EBV	Epstein Barr Virus
ENA	Extractable Nuclear Antigen
ESR	erythrocyte sedimentation rate
Estimator	a process or equation that when applied to input data provides a prediction on some value of interest
exclusionary diagnosis	a diagnosis in which a target diagnosis is achieved by eliminating plausible alternatives

FTA-Abs	fluorescent treponemal antibody absorption
GANS	Granulomatous angiitis of the nervous system
generative statistical techniques	techniques that generate hypotheses to be contrasted with formative 167echniques which focus on testing a hypothesis
Gini importance	a feature importance metric in decision trees that pertains to the degree to which a classification can be refined / impurity dropped by a variable
greedy algorithm	an algorithm that focuses on local optimization within any / every individual step
HERNS	Hereditary endotheliopathy with retinopathy, nephropathy, and stroke
HIV	human immunodeficiency virus
hyperplane	a boundary with a multi-dimensional variable space used in the case of support vector machines to separate two classes
interpretability	a property of estimators whereby the exact process by which an estimate is made can be transparently demonstrated
lock-out	a concept proposed in novel random forest where a prospective tree can not be generated because two few cases possess the compliment of variables that were selected to generate a tree
medoids	elements within a set that are used to represent the central element of a set
MPO	Myeloperoxidase Antibodies
MRI	Magnetic Resonance Imaging
noisy features	Features possessed of little predictiveness of the target class of a case
out of bag	those cases within the original dataset which were not available to a model at training time
outflow arc	the branch size moving away from an internal node of a probabilistic circuit
PACNS	Primary angiitis of the central nervous system
PACNS	Primary angiitis of the CNS
PCR	Polymerase chain reaction
PET	positron emission tomography
PMN	polymorphonuclear leukocytes
PR3	Anti-proteinase 3
prediction	the process of providing a prediction using the model
prediction time	the part of the modelling process where a prediction is rendered
proportioning	a concept proposed in novel random forest where the proportions of classifications are skewed by case rejection due to cases not possessing complete case information for the selected features at the time of training a tree
pseudo-randomness	noise added to the continuous data with ties or categorical data of a feature that emulates continuous data without ties without distorting the relative ordering of previously non-equal feature values
Rare conditions (i.e., rare medical condition)	a medical ailment afflicting fewer than 1 in 2,000 to 1 in 200,000 depending on regulatory agency
RCVS	Reversible cerebro-vascular constriction syndrome

RF	Random forest: a classification / regression technique that utilizes the confluence of the results of multiple component decision trees to determine an ultimate classification
RNA	Ribonucleic acid
sIL2R	Soluble interleukin 2 receptor
Sparsity	a property of data indicating a relative quantity of missing entries / information
SSA/B	Sjogren Syndrome
Suspected PACNS	was defined as those patients where PACNS or a subcategory of PACNS (i.e. Granulomatous angiitis of the CNS or GANS) was articulated by the author as a potential diagnostic consideration or condition of concern
SVM	Support vector machines: a classification / regression technique that models the interface between two classes using one or a number of kernel functions
Traditional classification methodologies	those methodologies which rely on optimization of a convex cost function across all input data to regress a singular dividing hyperplane between classes (e.g., SVM, logistic regression, etc.)
training	the process of generating a model
training time	the part of the modelling process where the model is trained / constructed
TREX1	Three Prime Repair Exonuclease 1
Variant RF	Variant Random Forest Technique: a classification method proposed during the thesis that constructs a random forest ensemble in the usual way but 1) trains individual trees with listwise deletion of cases possessing missing values specifically related to the features used by that tree for its individual classification effort and 2) uses only the votes of those trees that rely on features with complete data to vote on / participate in the prediction of the case's class
VDRL	Venereal Disease Research Laboratory
VWF	Von Willebrand factor
VZV	Varicella zoster virus
β-A4 amyloid	Amyloid Beta

XVIII. Notation

n_T	number of trees
n_t	number of unique cases drawn on when developing a tree
n	number of cases within the dataset
m	number of feature within the dataset
l	number of features selected to complete a tree
$lock_{non}$	non lock out matrix
$lock_{out}$	lock out matrix
n_{ci}	cases that possess no missing values across the l selected features for training of the i^{th} tree
ε_i'	the SVM error term indicating the distance from the marginal terms to the dividing hyperplane
w^T	transpose of the SVM weighting vector
b	a regression constant
t_i	classification pertaining to the i^{th} set of x features
x_i	i^{th} set of x features
α_i	LaGrange multiplier
β	LaGrange multiplier
$\hat{\theta}$	the prediction of the class / classification of the case of interest
ι	impurity
f_i	frequency of the i^{th} class
C	total number of classes
Im_j	importance of the j^{th} node
wt_j	weight pertaining to the number of cases sent to the j^{th} node
ι_j	impurity of case classifications within the j^{th} node
wt_{Lj}	weight pertaining to the number of cases sent down the left branch to the j^{th} node
wt_{Rj}	weight pertaining to the number of cases sent down the right branch to the j^{th} node
ι_{Lj}	residual impurity of case classifications sent down the left branch of the j^{th} node
ι_{Rj}	residual impurity of case classifications sent down the right branch of the j^{th} node
F_i	importance of feature importance I across the random forest
N_i	normalized importance of feature i
R_i	reduced importance of feature i
$dist_{a-b}$	overall distance between points a and b
$n_{available}$	number of available features
$n_{complete}$	number of features with complete data
$d_{sep}(x_{ai}, x_{bi})$	the distance measure between feature I of x_a and x_b
E	a matrix with random values
C_L	class associated with lower feature values
C_H	class associated with higher feature values
ι_{Before}	impurity before a split

ι_{After}	impurity after a split
w_s	weight / number of cases assigned to either daughter branch
sb	branch identifier
n_b	number of branches from a node
$\iota_{After\ Gate}$	impurity after gating
$X_{observed}$	those feature values that are observed
$X_{missing}$	those feature values that are missing
Y	classification values associated with X
p	number of features with complete data
k	number of clusters
$C_{K-means}$	k-means related cost
μ_j	measure of the center of the cluster
S_j^*	the set of cases within the j^{th} cluster
$\lvert S_i^* \rvert$	the size of the set pertaining to the j^{th} cluster
$a(i)$	intra-cluster dissimilarity
$d(o_l, o_j)$	distance between object / cases l and j
S_i	the set of cases containing the i^{th} point
$S_{i'}$	the set of cases not containing the i^{th} case that is the next closest to the i^{th} case
$b(i)$	Inter-cluster dissimilarity
$s(i)$	silhouette contribution from the i^{th} case
$S(i)$	the cumulative silhouette
C_{medoid}	medoid cost function
G	a decision tree
G'	a structural copy of a decision tree
v	a node within a decision tree
D_v	data pertaining to node v with tree G
u	child node of node v with tree G
$D_{u,i}$	data pertaining to the i^{th} feature node u
v'	child node of node v with tree G'
$ch(v)$	all child nodes of node v
$w_{v'u'}$	number of cases sent partitioned to node u' from node v'
$p_{v'}(x, y)$	probability density distributed at node v'
$p(x, y)$	joint probability function of x and y
w_{ni}	the weight of the outflow arc
p_{ni}	the joint probability of a set of features and classification at leaf node i
n_l	the number of descendent leaf nodes to a node of interest
Q	a linear system of F equations to predict the classification of r classes of interest versus an alternative

Z	array of r class probabilities
r	the number of classes
F^*	the number of all relevant features required to characterize the classification of an object of interest to certain classifications of interest
A	the number of auxiliary variables
B	baseline missingness mask
D	dataset matrix
V	feature value related missingness mask
M	missingness array
I	missingness interrelation array
CDF_M	missingness cumulative distribution function
S^R	sum of all unnormalized missingness
P^{e1}	exponent mask for base missingness
P^{e2}	exponent mask for value based missingness
P^{e3}	exponent mask for missingness based missingness
P^{eC}	combined exponent mask
P^C	combined exponentiated missingness mask
P^R	raw probability of missingness matrix
P^N	normalized probability of missingness matrix

XIX. References

[i] Calabrese, L.H. and Mallek, J.A., 1988. Primary angiitis of the central nervous system. Report of 8 new cases, review of the literature, and proposal for diagnostic criteria. *Medicine*, *67*(1), pp.20-39.

[ii] Hajj-Ali, R.A. and Calabrese, L.H. (no date) *Primary Angiitis of the Central Nervous System*. Available at: https://www.clevelandclinicmeded.com/medicalpubs/diseasemanagement/rheumatology/angiitis-of-central-nervous-system/ (Accessed: October 10, 2022).

[iii] MacLaren, K., Gillespie, J., Shrestha, S., Neary, D. and Ballardie, F.W., 2005. Primary angiitis of the central nervous system: emerging variants. *Qjm*, *98*(9), pp.643-654.

[iv] Kidd, D. (2019) *Primary Angiitis of the Central Nervous System (PACNS)*, *Vasculitis UK*. Available at: https://www.vasculitis.org.uk/about-vasculitis/primary-angiitis-of-the-central-nervous-system (Accessed: October 10, 2022).

[v] Beuker, C., Strunk, D., Rawal, R., Schmidt-Pogoda, A., Werring, N., Milles, L., Ruck, T., Wiendl, H., Meuth, S., Minnerup, H. and Minnerup, J., 2021. Primary Angiitis of the CNS: A Systematic Review and Meta-analysis. *Neurology-Neuroimmunology Neuroinflammation*, *8*(6).

[vi] Berlit, P. and Krämer, M., 2019. Primary angiitis of the CNS (PACNS) and Behçet disease. *Neurological Research and Practice*, *1*(1), pp.1-7.

[vii] Sundaram, S., Menon, D., Khatri, P., Sreedharan, S.E., Jayadevan, E.R., Sarma, P., Pagnoux, C. and Sylaja, P.N., 2020. Primary angiitis of the central nervous system: Clinical profiles and outcomes of 45 patients. *Yearbook of Medicine 2020*, *67*, p.296.

[viii] Salvarani, C., Brown Jr, R.D., Calamia, K.T., Christianson, T.J., Weigand, S.D., Miller, D.V., Giannini, C., Meschia, J.F., Huston III, J. and Hunder, G.G., 2007. Primary central nervous system vasculitis: analysis of 101 patients. *Annals of Neurology: Official Journal of the American Neurological Association and the Child Neurology Society*, *62*(5), pp.442-451.

[ix] Cho, T.A. and Jones, A., 2020. CNS vasculopathies: Challenging mimickers of primary angiitis of the central nervous system. *Best Practice & Research Clinical Rheumatology*, *34*(4), p.101569.

[x] Sun, L.I., Zhu, L., Zhao, T., Wang, D., Ma, D., Zhang, R. and Fang, S., 2016. A rare case of tumor-mimicking primary angiitis of the central nervous system. *Molecular and Clinical Oncology*, *4*(5), pp.827-829.

[xi] Ronen, J.A., Nguyen, A., Mueller, J.N. and Lee, H., 2018. Intracranial atherosclerosis versus primary angiitis of the central nervous system: a case report. *Cureus*, *10*(7).

[xii] Stoecklein, V.M., Kellert, L., Patzig, M., Küpper, C., Giese, A., Ruf, V., Weller, J., Kreth, F.W. and Schöberl, F., 2021. Extended stereotactic brain biopsy in suspected primary central nervous system angiitis: good diagnostic accuracy and high safety. *Journal of neurology*, *268*(1), pp.367-376.

[xiii] Hajj-Ali, R.A. and Langford, C.A. (2016) *Primary Angiitis of the Central Nervous System*, *Musculoskeletal Key*. Available at: https://musculoskeletalkey.com/primary-angiitis-of-the-central-nervous-system-2/ (Accessed: October 10, 2022).

[xiv] Coronel-Restrepo, N., Bonilla-Abadía, F., Cortes, O.A., Izquierdo, J.H., Shinchi, A.M., Bravo, J.C., Tobón, G.J. and Cañas, C.A., 2013. Primary angiitis of the central nervous system: a report of three cases from a single colombian center. *Case Reports in Neurological Medicine*, *2013*.

[xv] Gioia, L., Poppe, A. and Lanthier, S., 2011. Primary Angiitis of the Central Nervous System-Clinical Approaches, Challenges and Controversies. *Eur Neurol Rev*, *6*, pp.181-186.

[xvi] Sarti, C., Picchioni, A., Telese, R., Pasi, M., Failli, Y., Pracucci, G., Cammelli, D. and Inzitari, D., 2020. "When should primary angiitis of the central nervous system (PACNS) be suspected?": literature review and proposal of a preliminary screening algorithm. *Neurological Sciences*, *41*(11), pp.3135-3148.

[xvii] Kadoba, K., Nishimura, K., Waki, D., Okada, T., Kumazawa, T., Saito, R., Murabe, H. and Yokota, T., 2021. Black-blood magnetic resonance imaging suggesting central nervous system vasculitis in moyamoya syndrome associated with systemic lupus erythematosus. *Immunological Medicine*, *44*(4), pp.270-273.

[xviii] Eiden, S., Beck, C., Venhoff, N., Elsheikh, S., Ihorst, G., Urbach, H. and Meckel, S., 2019. High-resolution contrast-enhanced vessel wall imaging in patients with suspected cerebral vasculitis: Prospective comparison of whole-brain 3D T1 SPACE versus 2D T1 black blood MRI at 3 Tesla. *PloS one*, *14*(3), p.e0213514.

[xix] Niederberger, M. and Spranger, J., 2020. Delphi technique in health sciences: a map. *Frontiers in public health*, *8*, p.457.

[xx] Wang, J., Zhou, J., Liu, J., Wonka, P. and Ye, J., 2014. A safe screening rule for sparse logistic regression. *Advances in neural information processing systems*, *27*.

[xxi] Walker, D.A. and Smith, T.J., 2020. Logistic regression under sparse data conditions. *Journal of Modern Applied Statistical Methods*, *18*(2), p.25.

[xxii] Midi, H., Sarkar, S.K. and Rana, S., 2010. Collinearity diagnostics of binary logistic regression model. *Journal of interdisciplinary mathematics*, *13*(3), pp.253-267.

[xxiii] Couronné, R., Probst, P. and Boulesteix, A.L., 2018. Random forest versus logistic regression: a large-scale benchmark experiment. *BMC bioinformatics*, *19*, pp.1-14.

[xxiv] Fisher, A.J., Medaglia, J.D. and Jeronimus, B.F., 2018. Lack of group-to-individual generalizability is a threat to human subjects research. *Proceedings of the National Academy of Sciences*, *115*(27), pp.E6106-E6115.

[xxv] Anonymous. How robust are your data?. *Nat Cell Biol* **11**, 667 (2009). https://doi.org/10.1038/ncb0609-667a

[xxvi] Feng, J. and Simon, N., 2017. Sparse-input neural networks for high-dimensional nonparametric regression and classification. *arXiv preprint arXiv:1711.07592*.

[xxvii] Ish-Horowicz, J., Udwin, D., Flaxman, S., Filippi, S. and Crawford, L., 2019. Interpreting deep neural networks through variable importance. *arXiv preprint arXiv:1901.09839*.

[xxviii] anon. (n.d.): Random Forests - University of Wisconsin–Madison. Retrieved am 26.03.2023 from https://pages.cs.wisc.edu/~matthewb/pages/notes/pdf/ensembles/RandomForests.pdf.

[xxix] Ho, T.K., 1998. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, *20*(8), pp.832-844.

[xxx] anon. (n.d.): Sklearn.ensemble.randomforestclassifier. *scikit*. Retrieved am 20.03.2023 from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[xxxi] anon. (n.d.): Sklearn.ensemble.randomforestregressor. *scikit*. Retrieved am 12.03.2023 from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html.

[xxxii] Breiman, L., 1996. Bagging predictors. Machine learning, 24, pp.123-140.

[xxxiii] Pratap, Jujjavarapu R (2020): „Ensemble voting classifiers and random forests in sci-Kit Learn". *Medium*. Analytics Vidhya Retrieved am 15.03.2023 from https://medium.com/analytics-vidhya/ensemble-voting-classifiers-and-random-forests-in-sci-kit-learn-ed0ee6a81a12.

[xxxiv] Little, R.J., 1992. Regression with missing X's: a review. *Journal of the American statistical association*, *87*(420), pp.1227-1237.

[xxxv] King, G., Honaker, J., Joseph, A. and Scheve, K., 2001. Analyzing incomplete political science data: An alternative algorithm for multiple imputation. *American political science review*, *95*(1), pp.49-69.

[xxxvi] Sterne, J.A., White, I.R., Carlin, J.B., Spratt, M., Royston, P., Kenward, M.G., Wood, A.M. and Carpenter, J.R., 2009. Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls. *Bmj*, *338*.

[xxxvii] Pigott, T.D., 2001. A review of methods for missing data. *Educational research and evaluation*, *7*(4), pp.353-383.

[xxxviii] anon. (n.d.): *How SVM works*. Retrieved am 20.03.2023 from https://www.ibm.com/docs/en/spss-modeler/saas?topic=models-how-svm-works.

[xxxix] Smola, A.J. and Schölkopf, B., 2004. A tutorial on support vector regression. *Statistics and computing*, *14*(3), pp.199-222.

[xl] Oleszak, M. (2021) *SVM kernels: What do they actually do?*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/svm-kernels-what-do-they-actually-do-56ce36f4f7b8 (Accessed: October 10, 2022).

[xli] Gandhi, Rohith (2018): Support Vector Machine - introduction to machine learning algorithms. *Medium*. Towards Data Science Retrieved am 13.03.2023 from https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

[xlii] Hastie, T., Tibshirani, R., Friedman, J.H. and Friedman, J.H., 2009. *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2, pp. 1-758). New York: springer.

[xliii] Nembrini, S., König, I.R. and Wright, M.N., 2018. The revival of the Gini importance?. *Bioinformatics*, *34*(21), pp.3711-3718.

[xliv] anon. (n.d.): Permutation importance vs random forest feature importance (MDI). *scikit*. Retrieved am 24.03.2023 from https://scikit-learn.org/stable/auto_examples/inspection/plot_permutation_importance.html.

[xlv] Strobl, C., Boulesteix, A.L., Zeileis, A. and Hothorn, T., 2007. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC bioinformatics*, *8*(1), pp.1-21.

[xlvi] anon. (n.d.): Machine learning | google developers. *Google*. Google Retrieved am 11.03.2023 from https://developers.google.com/machine-learning/decision-forests/random-forests.

[xlvii] Bhatia, Navnina (2019): What is out of Bag (OOB) score in Random Forest?. *Medium*. Towards Data Science Retrieved am 24.03.2023 from https://towardsdatascience.com/what-is-out-of-bag-oob-score-in-random-forest-a7fa23d710.

[xlviii] Breiman, L., 2001. Random forests. *Machine learning*, *45*(1), pp.5-32.

[xlix] Emmanuel, T., Maupong, T., Mpoeleng, D., Semong, T., Mphago, B. and Tabona, O., 2021. A survey on missing data in machine learning. *Journal of Big Data*, *8*(1), pp.1-37.

[l] Stewart, T.G., Zeng, D. and Wu, M.C., 2018. Constructing support vector machines with missing data. *Wiley Interdisciplinary Reviews: Computational Statistics*, *10*(4), p.e1430.

[li] Upadhyaya, S., 2014. Comparison of Classification Rates among Logistic Regression, Neural Network and Support Vector Machines in the Presence of Missing Data.

[lii] Tang, F. and Ishwaran, H., 2017. Random forest missing data algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, *10*(6), pp.363-377.

[liii] Azur, M.J., Stuart, E.A., Frangakis, C. and Leaf, P.J., 2011. Multiple imputation by chained equations: what is it and how does it work?. *International journal of methods in psychiatric research*, *20*(1), pp.40-49.

[liv] Beretta, L. and Santaniello, A., 2016. Nearest neighbor imputation algorithms: a critical evaluation. *BMC medical informatics and decision making*, *16*(3), pp.197-208.

[lv] Tang, F. and Ishwaran, H., 2017. Random forest missing data algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, *10*(6), pp.363-377.

[lvi] Ishwaran, H., Kogalur, U.B., Blackstone, E.H. and Lauer, M.S., 2008. Random survival forests. *The annals of applied statistics*, *2*(3), pp.841-860.

[lvii] Beaulac, C. and Rosenthal, J.S., 2020. BEST: A decision tree algorithm that handles missing values. *Computational Statistics*, *35*(3), pp.1001-1026.

[lviii] Salzberg, S.L., 1994. C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993.

[lix] Stewart, T.G., Zeng, D. and Wu, M.C., 2018. Constructing support vector machines with missing data. *Wiley Interdisciplinary Reviews: Computational Statistics*, *10*(4), p.e1430.

[lx] Edgell, R.C., Sarhan, A.E., Soomro, J., Einertson, C., Kemp, J., Shirani, P., Malmstrom, T.K. and Coppens, J., 2016. The role of catheter angiography in the diagnosis of central nervous system vasculitis. *Interventional Neurology*, *5*(3-4), pp.194-208.

[lxi] Küker, W., 2007. Cerebral vasculitis: imaging signs revisited. *Neuroradiology*, *49*(6), pp.471-479.

[lxii] Twilt, M. and Benseler, S.M., 2012. The spectrum of CNS vasculitis in children and adults. *Nature reviews Rheumatology*, *8*(2), pp.97-107.

[lxiii] Adams Jr, H.P., 2014. Cerebral vasculitis. *Handbook of clinical neurology*, *119*, pp.475-494.

[lxiv] Hajj-Ali, R.A., 2010. Primary angiitis of the central nervous system: differential diagnosis and treatment. *Best Practice & Research Clinical Rheumatology*, *24*(3), pp.413-426.

[lxv] Kraemer, M. and Berlit, P., 2021. Primary central nervous system vasculitis–An update on diagnosis, differential diagnosis and treatment. *Journal of the Neurological Sciences*, *424*, p.117422.

[lxvi] Strunk, D., Schmidt-Pogoda, A., Beuker, C., Milles, L.S., Korsukewitz, C., Meuth, S.G. and Minnerup, J., 2019. Biomarkers in vasculitides of the nervous system. *Frontiers in Neurology*, *10*, p.591.

[lxvii] Hajj-Ali, R.A., 2010. Primary angiitis of the central nervous system: differential diagnosis and treatment. *Best Practice & Research Clinical Rheumatology*, *24*(3), pp.413-426.

[lxviii] Birnbaum, J. and Hellmann, D.B., 2009. Primary angiitis of the central nervous system. *Archives of neurology*, *66*(6), pp.704-709.

[lxix] Ruland, T., Wolbert, J., Gottschalk, M.G., König, S., Schulte-Mecklenbeck, A., Minnerup, J., Meuth, S.G., Groß, C.C., Wiendl, H. and Meyer zu Hörste, G., 2018. Cerebrospinal fluid concentrations of neuronal proteins are reduced in primary angiitis of the central nervous system. *Frontiers in neurology*, *9*, p.407.

[lxx] Twilt, M. and Benseler, S.M., 2012. The spectrum of CNS vasculitis in children and adults. *Nature reviews Rheumatology*, *8*(2), pp.97-107.

[lxxi] Xu, S. and Cao, X., 2010. Interleukin-17 and its expanding biological functions. *Cellular & molecular immunology*, *7*(3), pp.164-174.

[lxxii] Anon., 2022. *LNBAI - Mayo Clinic Laboratories*. [online] Available at: <https://www.mayocliniclabs.com/test-catalog/Specimen/63249> [Accessed 21 February 2022].

[lxxiii] Henao-Martínez, A.F. and Johnson, S.C., 2014. Diagnostic tests for syphilis: New tests and new algorithms. *Neurology: Clinical Practice*, *4*(2), pp.114-122.

[lxxiv] Kraemer, M. and Berlit, P., 2021. Primary central nervous system vasculitis–An update on diagnosis, differential diagnosis and treatment. *Journal of the Neurological Sciences*, *424*, p.117422.

[lxxv] Dutra, L.A., de Souza, A.W.S., Grinberg-Dias, G., Barsottini, O.G.P. and Appenzeller, S., 2017. Central nervous system vasculitis in adults: an update. *Autoimmunity reviews*, *16*(2), pp.123-131.

[lxxvi] Küker, W., 2007. Cerebral vasculitis: imaging signs revisited. *Neuroradiology*, *49*(6), pp.471-479.

[lxxvii] Edgell, R.C., Sarhan, A.E., Soomro, J., Einertson, C., Kemp, J., Shirani, P., Malmstrom, T.K. and Coppens, J., 2016. The role of catheter angiography in the diagnosis of central nervous system vasculitis. *Interventional Neurology*, *5*(3-4), pp.194-208.

[lxxviii] Hajj-Ali, R.A., 2010. Primary angiitis of the central nervous system: differential diagnosis and treatment. *Best Practice & Research Clinical Rheumatology*, *24*(3), pp.413-426.

[lxxix] Schmidt, W., Kraft, H., Völker, L., Vorpahl, K. and Gromnica-Ihle, E., 1995. Colour Doppler sonography to diagnose temporal arteritis. *The Lancet*, *345*(8953), p.866.

[lxxx] Eiden, S., Beck, C., Venhoff, N., Elsheikh, S., Ihorst, G., Urbach, H. and Meckel, S., 2019. High-resolution contrast-enhanced vessel wall imaging in patients with suspected cerebral vasculitis: Prospective comparison of whole-brain 3D T1 SPACE versus 2D T1 black blood MRI at 3 Tesla. *PLoS One*, *14*(3), p.e0213514.

[lxxxi] Küker, W., 2007. Cerebral vasculitis: imaging signs revisited. *Neuroradiology*, *49*(6), pp.471-479.

[lxxxii] Rice, C.M. and Scolding, N.J., 2020. The diagnosis of primary central nervous system vasculitis. *Practical Neurology*, *20*(2), pp.109-114.

[lxxxiii] Quinn, K.A., Ahlman, M.A., Malayeri, A.A., Marko, J., Civelek, A.C., Rosenblum, J.S., Bagheri, A.A., Merkel, P.A., Novakovich, E. and Grayson, P.C., 2018. Comparison of magnetic resonance angiography and 18F-fluorodeoxyglucose positron emission tomography in large-vessel vasculitis. *Annals of the rheumatic diseases*, *77*(8), pp.1165-1171.

[lxxxiv] Hammad, T.A. and Hajj-Ali, R.A., 2013. Primary angiitis of the central nervous system and reversible cerebral vasoconstriction syndrome. *Current atherosclerosis reports*, *15*(8), pp.1-9.

[lxxxv] Hajj-Ali, R.A. and Calabrese, L.H., 2013. Primary angiitis of the central nervous system. *Autoimmunity reviews*, *12*(4), pp.463-466.

[lxxxvi] Limaye, K., Samaniego, E.A. and Adams, H.P., 2018. Diagnosis and treatment of primary central nervous system angiitis. *Current treatment options in neurology*, *20*(9), pp.1-14.

[lxxxvii] Berlit, P. and Krämer, M., 2019. Primary angiitis of the CNS (PACNS) and Behçet disease. *Neurological Research and Practice*, *1*(1), pp.1-7.

[lxxxviii] Byram, K., Hajj-Ali, R.A. and Calabrese, L., 2018. CNS vasculitis: an approach to differential diagnosis and management. *Current Rheumatology Reports*, *20*(7), pp.1-7.

[lxxxix] Dutra, L.A., de Souza, A.W.S., Grinberg-Dias, G., Barsottini, O.G.P. and Appenzeller, S., 2017. Central nervous system vasculitis in adults: an update. *Autoimmunity reviews*, *16*(2), pp.123-131.

[xc] Powers, W.J., 2015. Primary angiitis of the central nervous system: diagnostic criteria. *Neurologic clinics*, *33*(2), pp.515-526.

[xci] Matsue, K., Abe, Y., Kitadate, A., Miura, D., Narita, K., Kobayashi, H., Takeuchi, M., Enzan, N., Tanaka, A. and Takeuchi, K., 2019. Sensitivity and specificity of incisional random skin biopsy for diagnosis of intravascular large B-cell lymphoma. *Blood, The Journal of the American Society of Hematology*, *133*(11), pp.1257-1259.

[xcii] Adachi, Y., Kosami, K., Mizuta, N., Ito, M., Matsuoka, Y., Kanata, M., Akiyama, H., Murao, T., Li, M., Ieki, R. and Ikehara, S., 2014. Benefits of skin biopsy of senile hemangioma in intravascular large B-cell lymphoma: A case report and review of the literature. *Oncology letters*, *7*(6), pp.2003-2006.

[xciii] Hajj-Ali, R.A. and Calabrese, L.H. (2022) *Primary angiitis of the central nervous system in adults*, *UpToDate*. Available at: https://www.uptodate.com/contents/primary-angiitis-of-the-central-nervous-system-in-adults (Accessed: October 10, 2022).

[xciv] Singhal, A.B., Topcuoglu, M.A., Fok, J.W., Kursun, O., Nogueira, R.G., Frosch, M.P. and Caviness Jr, V.S., 2016. Reversible cerebral vasoconstriction syndromes and primary angiitis of the central nervous system: clinical, imaging, and angiographic comparison. *Annals of neurology*, *79*(6), pp.882-894.

[xcv] de Boysson, H., Boulouis, G., Dequatre, N., Godard, S., Néel, A., Arquizan, C., Detante, O., Bloch-Queyrat, C., Zuber, M., Touzé, E. and Bienvenu, B., 2016. Tumor-like presentation of primary angiitis of the central nervous system. *Stroke*, *47*(9), pp.2401-2404.

[xcvi] Mansueto, G., Lanza, G., Fisicaro, F., Alaouieh, D., Hong, E., Girolami, S., Montella, M., Feola, A. and Di Napoli, M., 2022. Central and Peripheral Nervous System Complications of Vasculitis Syndromes From Pathology to Bedside: Part 1—Central Nervous System. *Current Neurology and Neuroscience Reports*, pp.1-23.

[xcvii] Hajj-Ali, R.A. and Calabrese, L.H., 2014. Diagnosis and classification of central nervous system vasculitis. *Journal of autoimmunity*, *48*, pp.149-152.

[xcviii] Kraemer, M. and Berlit, P., 2021. Primary central nervous system vasculitis–An update on diagnosis, differential diagnosis and treatment. *Journal of the Neurological Sciences*, *424*, p.117422.

[xcix] Rice, C.M. and Scolding, N.J., 2020. The diagnosis of primary central nervous system vasculitis. *Practical Neurology*, *20*(2), pp.109-114.

[c] Berlit, P. and Krämer, M., 2019. Primary angiitis of the CNS (PACNS) and Behçet disease. *Neurological Research and Practice*, *1*(1), pp.1-7.

[ci] Beuker, C., Schmidt, A., Strunk, D., Sporns, P.B., Wiendl, H., Meuth, S.G. and Minnerup, J., 2018. Primary angiitis of the central nervous system: diagnosis and treatment. *Therapeutic advances in neurological disorders*, *11*, p.1756286418785071.

[cii] Limaye, K., Samaniego, E.A. and Adams, H.P., 2018. Diagnosis and treatment of primary central nervous system angiitis. *Current treatment options in neurology*, *20*(9), pp.1-14.

[ciii] Hajj-Ali, R.A. and Calabrese, L.H., 2014. Diagnosis and classification of central nervous system vasculitis. *Journal of autoimmunity*, *48*, pp.149-152.

[civ] Hajj-Ali, R.A. and Calabrese, L.H., 2014. Diagnosis and classification of central nervous system vasculitis. *Journal of autoimmunity*, *48*, pp.149-152.

[cv] Anon., 2022. *TB Risk Factors | Basic TB Facts | TB | CDC*. [online] Available at: <https://www.cdc.gov/tb/topic/basics/risk.htm> [Accessed 21 February 2022].

[cvi] Wan, C. and Su, H., 2017. A Closer Look at Angiitis of the central nervous system. *Neurosciences Journal*, *22*(4), pp.247-254.

[cvii] Powers, W.J., 2015. Primary angiitis of the central nervous system: diagnostic criteria. *Neurologic clinics*, *33*(2), pp.515-526.

[cviii] Berlit, P. and Kraemer, M., 2014. Cerebral vasculitis in adults: what are the steps in order to establish the diagnosis? Red flags and pitfalls. *Clinical & Experimental Immunology*, *175*(3), pp.419-424.

[cix] Salvarani, C., Brown Jr, R.D. and Hunder, G.G., 2012. Adult primary central nervous system vasculitis. *The Lancet*, *380*(9843), pp.767-777.

[cx] "TB Risk Factors." Centers for Disease Control and Prevention, Centers for Disease Control and Prevention, 18 Mar. 2016, https://www.cdc.gov/tb/topic/basics/risk.htm.

[cxi] Hajj-Ali, R.A. and Calabrese, L.H., 2014. Diagnosis and classification of central nervous system vasculitis. *Journal of autoimmunity*, *48*, pp.149-152.

[cxii] Hammad, T.A. and Hajj-Ali, R.A., 2013. Primary angiitis of the central nervous system and reversible cerebral vasoconstriction syndrome. *Current atherosclerosis reports*, *15*(8), pp.1-9.

[cxiii] Gu, Y., Preisser, J.S., Zeng, D., Shrestha, P., Shah, M., Simancas-Pallares, M.A., Ginnis, J. and Divaris, K., 2022. Partitioning around medoids clustering and random forest classification for GIS-informed imputation of fluoride concentration data. *The annals of applied statistics*, *16*(1), p.551.

[cxiv] Llerena, J.V., Mauá, D.D. and Antonucci, A., 2021, September. Cautious Classification with Data Missing Not at Random Using Generative Random Forests. In *European Conference on Symbolic and Quantitative Approaches with Uncertainty* (pp. 284-298). Springer, Cham.

[cxv] Hillis, T., Guarcello, M.A., Levine, R.A. and Fan, J., 2021. Causal inference in the presence of missing data using a random forest-based matching algorithm. *Stat*, *10*(1), p.e326.

[cxvi] Hapfelmeier, A., Hothorn, T., Riediger, C. and Ulm, K., 2014. Estimation of a predictor's importance by random forests when there is missing data: RISK prediction in liver surgery using laboratory data. *The International Journal of Biostatistics*, *10*(2), pp.165-183.

[cxvii] Hapfelmeier, A. and Ulm, K., 2014. Variable selection by Random Forests using data with missing values. *Computational Statistics & Data Analysis*, *80*, pp.129-139.

[cxviii] Mozharovskyi, P., Josse, J. and Husson, F., 2020. Nonparametric imputation by data depth. *Journal of the American Statistical Association*, *115*(529), pp.241-253.

[cxix] Solaro, N., Barbiero, A., Manzi, G. and Ferrari, P.A., 2018. A simulation comparison of imputation methods for quantitative data in the presence of multiple data patterns. *Journal of Statistical Computation and Simulation*, *88*(18), pp.3588-3619.

[cxx] Tang, F. and Ishwaran, H., 2017. Random forest missing data algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, *10*(6), pp.363-377.

[cxxi] Hapfelmeier, A., Hothorn, T., Ulm, K. and Strobl, C., 2014. A new variable importance measure for random forests with missing data. *Statistics and Computing*, *24*(1), pp.21-34.

[cxxii] Gu, Y., Preisser, J.S., Zeng, D., Shrestha, P., Shah, M., Simancas-Pallares, M.A., Ginnis, J. and Divaris, K., 2022. Partitioning around medoids clustering and random forest classification for GIS-informed imputation of fluoride concentration data. *The annals of applied statistics*, *16*(1), p.551.

[cxxiii] Rousseeuw, P.J., 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, *20*, pp.53-65.

[cxxiv] de Campos, Cassio (n.d.): Generative random forests - decampos.nl. Retrieved am 13.03.2023 from https://decampos.nl/siptaschool2022/gefs.pdf.

[cxxv] Correia, A., Peharz, R. and de Campos, C.P., 2020. Joints in random forests. *Advances in neural information processing systems*, *33*, pp.11404-11415.

[cxxvi] Van den Broeck, G., Di Mauro, N., and Vergari, A. Tractable probabilistic models: Representations, algorithms, learning, and applications. http://web.cs.ucla.edu/ ˜guyvdb/slides/TPMTutorialUAI19.pdf, 2019. Tutorial at UAI 2019

[cxxvii] Llerena, J.V., Mauá, D.D. and Antonucci, A., 2021, September. Cautious Classification with Data Missing Not at Random Using Generative Random Forests. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty: 16th European Conference, ECSQARU 2021, Prague, Czech Republic, September 21–24, 2021, Proceedings* (pp. 284-298). Cham: Springer International Publishing.

[cxxviii] Correia, A.H., Peharz, R. and de Campos, C., 2020. Towards Robust Classification with Deep Generative Forests. *arXiv preprint arXiv:2007.05721*.

[cxxix] van de Schoot, R., Depaoli, S., King, R., Kramer, B., Märtens, K., Tadesse, M.G., Vannucci, M., Gelman, A., Veen, D., Willemsen, J. and Yau, C., 2021. Bayesian statistics and modelling. *Nature Reviews Methods Primers*, *1*(1), p.1.

[cxxx] anon. (n.d.): What is naïve Bayes. *IBM*. Retrieved am 22.03.2023 from https://www.ibm.com/topics/naive-bayes.

[cxxxi] Hong, S. and Lynn, H.S., 2020. Accuracy of random-forest-based imputation of missing data in the presence of non-normality, non-linearity, and interaction. *BMC medical research methodology*, *20*(1), pp.1-12.

[cxxxii] Mack, C., Su, Z. and Westreich, D., 2018. Managing missing data in patient registries: addendum to registries for evaluating patient outcomes: a user's guide.

[cxxxiii] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J., 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, pp.2825-2830.

[cxxxiv] Batista, G.E. and Monard, M.C., 2002. A study of K-nearest neighbour as an imputation method. *His*, *87*(251-260), p.48.

[cxxxv] Madley-Dowd, P., Hughes, R., Tilling, K. and Heron, J., 2019. The proportion of missing data should not be used to guide decisions on multiple imputation. *Journal of clinical epidemiology*, *110*, pp.63-73.

[cxxxvi] anon. (n.d.): Sklearn.ensemble.randomforestclassifier. *scikit*. Retrieved am 20.03.2023 from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[cxxxvii] anon. (n.d.): API reference. *scikit*. Retrieved am 20.03.2023 from https://scikit-learn.org/stable/modules/classes.html#module-sklearn.svm.

[cxxxviii] Agjee, N.E.H., Mutanga, O., Peerbhay, K. and Ismail, R., 2018. The impact of simulated spectral noise on random forest and oblique random forest classification performance. *Journal of Spectroscopy*, *2018*.

[cxxxix] Cortez, P., Cerdeira, A., Almeida, F., Matos, T. and Reis, J., 2009. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*, *47*(4), pp.547-553.

[cxl] Oshiro, T.M., Perez, P.S. and Baranauskas, J.A., 2012. How many trees in a random forest?. In *Machine Learning and Data Mining in Pattern Recognition: 8th International Conference, MLDM 2012, Berlin, Germany, July 13-20, 2012. Proceedings 8* (pp. 154-168). Springer Berlin Heidelberg.

[cxli] Probst, P. and Boulesteix, A.L., 2017. To tune or not to tune the number of trees in random forest. *The Journal of Machine Learning Research*, *18*(1), pp.6673-6690.

[cxlii] Aeberhard, S., Coomans, D., and de Vel, O., 1992. Comparison of Classifiers in High Dimensional Settings, *Tech. Rep. no. 92-02, Dept. of Computer Science and Dept. of Mathematics and Statistics, James Cook University of North Queensland. (Also submitted to Technometrics).*

[cxliii] Hothorn, T., Hornik, K. and Zeileis, A., 2006. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical statistics*, *15*(3), pp.651-674.

[cxliv] Breiman, L., 1999. *Using adaptive bagging to debias regressions* (p. 16). Technical Report 547, Statistics Dept. UCB.

[cxlv] anon. (n.d.): Permutation importance with multicollinear or correlated features. *scikit*. Retrieved am 14.03.2023 from https://scikit-learn.org/stable/auto_examples/inspection/plot_permutation_importance_multicollinear.html.

[cxlvi] Barse, E.L., Kvarnstrom, H. and Jonsson, E., 2003, December. Synthesizing test data for fraud detection systems. In *19th Annual Computer Security Applications Conference, 2003. Proceedings.* (pp. 384-394). IEEE.

[cxlvii] Allison, P.D., 2008, March. Convergence failures in logistic regression. In *SAS Global Forum* (Vol. 360, No. 1, p. 11).

[cxlviii] Morgan, B.P. and Harris, C.L., 2015. Complement, a target for therapy in inflammatory and degenerative diseases. *Nature reviews Drug discovery*, *14*(12), pp.857-877.

[cxlix] Nakae, H., Endo, S., Inada, K. and Yoshida, M., 1996. Chronological changes in the complement system in sepsis. *Surgery today*, *26*(4), pp.225-229.

[cl] Mandl, L.A., Solomon, D.H., Smith, E.L., Lew, R.A., Katz, J.N. and Shmerling, R.H., 2002. Using antineutrophil cytoplasmic antibody testing to diagnose vasculitis: can test-ordering guidelines improve diagnostic accuracy?. *Archives of Internal Medicine*, *162*(13), pp.1509-1514.

[cli] Raj, P., Li, Q.Z., Karp, D., Olsen, N., Sivils, K., James, J., Kelly, J., Lauwerys, B., Gregersen, P. and Wakeland, E., 2013. Antinuclear antibodies in general population: what does that mean?(P4535).

[clii] Aringer, M., Costenbader, K., Daikh, D., Brinks, R., Mosca, M., Ramsey-Goldman, R., Smolen, J.S., Wofsy, D., Boumpas, D.T., Kamen, D.L. and Jayne, D., 2019. 2019 European League Against Rheumatism/American College of Rheumatology classification criteria for systemic lupus erythematosus. *Arthritis & rheumatology*, *71*(9), pp.1400-1412.

[cliii] Antinuclear Antibodies (ANA) (n.d.) *Antinuclear antibodies (ANA)*. Available from: https://rheumatology.org/patients/antinuclear-antibodies-ana (accessed 9 April 2023).

[cliv] Ellis, S.G. and Verity, M.A., 1979, February. Central nervous system involvement in systemic lupus erythematosus: a review of neuropathologic findings in 57 cases, 1955–1977. In *Seminars in arthritis and rheumatism* (Vol. 8, No. 3, pp. 212-221). WB Saunders.

[clv] Bernstein, J.E., Podkovik, S., Kashyap, S., Ghanchi, H. and Ananda, A.K., 2020. Primary Angiitis of the Central Nervous System Presenting as a Cerebral Mass Lesion: A Case Report and Literature Review. *Cureus*, *12*(6).

[clvi] Nahm, F.S., 2022. Receiver operating characteristic curve: overview and practical use for clinicians. *Korean journal of anesthesiology*, *75*(1), pp.25-36.

[clvii] Franco, F. and Di Napoli, A., 2016. Evaluation of diagnostic tests in parallel and in series. *Giornale di Tecniche Nefrologiche e Dialitiche*, *28*(3), pp.212-215.

[clviii] Anon, 2022. About CanVasc. *CanVasc*. Available at: https://canvasc.ca/about-canvasc/ [Accessed May 23, 2022].

[clix] Trevelyan, E.G. and Robinson, N., 2015. Delphi methodology in health research: how to do it?. *European Journal of Integrative Medicine*, *7*(4), pp.423-428.

[clx] Government of Canada, Statistics Canada (2021): „4.5 measures of Dispersion 4.5.2 visualizing the box and whisker plot“. *4.5.2 Visualizing the box and whisker plot*. Retrieved am 27.03.2023 from https://www150.statcan.gc.ca/n1/edu/power-pouvoir/ch12/5214889-eng.htm.