Improving protein identification in mass spectrometry imaging using machine learning and spatial spectral information

by

Soroush Shahryari Fard

A thesis submitted in partial fulfillment of the requirements for the degree of
Biochemistry with specialization in Bioinformatics

Master of Science

Ottawa Institute of Systems Biology
Department of Biochemistry, Microbiology and Immunology
Faculty of Medicine
University of Ottawa

Abstract

Mass spectrometry imaging (MSI) is a high-throughput technique that in addition to performing protein identification, can capture the spatial localization of proteins within biological tissue. Nevertheless, sample pre-processing and MSI instrumentation limit protein identification capability in MSI compared to more standard tandem mass spectrometry-based proteomics methods. Despite these limitations, the current protein identification approaches used in MSI were originally designed for standard mass spectrometry-based proteomics and do not take advantage of the spatial information acquired in MSI. Herein, I explore the benefit of using the spatial spectral information for protein identification using two objectives. For the first objective, I developed a novel supervised learning spatially-aware protein identification algorithm (SAPID) for mass spectrometry imaging and benchmarked it against ProteinProphet and Percolator, which are state-of-the-art tools for protein identification confidence assessment. I showed that SAPID identifies on average 20% more proteins at <1% false discovery rate compared to the other two algorithms. Furthermore, more proteins are identified when spatial features are used to identify proteins compared to when they are not suggesting their additional benefit. For the second objective, I used SAPID to rescue false positive and false negative protein identifications made by ProteinProphet. By examining a combination of data sampling and learning algorithms, I was able to achieve a good classification performance compared to the baseline given the extreme

imbalance in the dataset. Finally, by improving proteome characterization in MSI, our approach will help providing a better understanding of the processes taking place in biological tissues.

To my uncle Keykhosro,

(کیخسرو شهریاری فرد)

who was always loving, genuine, and thoughtful to others.

Acknowledgements

I want to thank my supervisors Dr. Mathieu Lavallée-Adam and Dr. Theodore J. Perkins who were always supportive. Their suggestions, criticisms, and encouragements throughout my Master's made me a better scientist. Mathieu, thank you for taking the chance and accepting me into your lab. I am very grateful for the opportunity and so many things I have learned from you, which applies beyond the scope of this thesis. Ted, thanks for your continuous support. You have shown me what it is like to be a critical thinker, and I hope sometimes I can achieve that, too.

I would like to thank my Thesis Advisory Committee (TAC) members Dr. Marjorie Brand and Dr. Carolina Ilkow who were always supportive and extremely helpful. I am very honoured for having you on my TAC, and this thesis would not have been possible without you. Also, special thanks to Dr. Marcel Turcotte who accepted to evaluate my thesis and provided valuable comments to improve this thesis.

I would like to thank my friends and lab mates who were always supportive and made this journey more enjoyable: Dallas Nygard, who took me to the gym for the first time in years and taught me how to play squash; Emily Hashimoto-Roth, who was always generous helping me on any project or assignment; Rachel Nadeau, who was always encouraging and supportive; Amit Scheer, who could explain complex concepts to me with ease; Iryna Abramchuk, who was always calm, friendly, and enjoyable to talk to; Francesca Barry, who taught me how to use Mac; Yun-En Chung, whose hard work and perseverance have been a model for me; Alexander Pelletier, who taught me

terminal and other cool stuff; Patrick Smyth, with whom I laughed so much; Dr. Caitlin Simopoulos, who was always available for help; Aarthie Senathirajah, who was a great classmate; Alona Petrova, who asked great questions during lab meetings; Justin Chitpin, who was always great to talk to. Also, special thanks to Shaima Kaka who was always available to address any medical questions I had about my research.

I am indebted to Dr. Robert Jack Cornett (R.I.P.) whose trust in me started my journey in research and made me become more confident in myself. Thank you Jack.

I am very grateful for the support of Ontario Graduate Scholarship and Canadian Institute of Health Research Graduate Scholarship which made this research possible.

Finally, I would like to thank all of my family, especially my mother, Jila, my father, Farshid, and my brother, Shahryar. Their love, support, and sacrifice has provided me with the opportunity to study in Canada and made me reach my goals.

Contents

List of Tables

List of Figures

Glossary

3D 3-dimentional

AA amino acids

ANN Artificial Neural Network

CV Cross-validation

deltaCN delta-correlation

Dummy CLF Dummy classifier

ENN Edited Nearest Neighbour

ESI Electrospray Ionization

FDR false discovery rate

FN false negatives

FP false positives

FPR false positive rate

Gaussian NB Gaussian Naïve Bayes

IDC Intraductal Carcinoma

LC–MS liquid chromatography–mass spectrometry

LDA Linear Discriminant Analysis

Linear SVM SVM with linear kernel

m/z mass-to-charge ratio

MALDI Matrix-Assisted Laser Desorption Ionization

MRF Markov Random Field

MS/MS Tandem Mass Spectrometry

MSI Mass Spectrometry Imaging

PEP posterior error probability

POI pixel-of-interest

PPV positive predictive value

PR precision and recall

PRAUC area under the PR curve

PSM peptide-spectrum-matches

PTM post-translational modifications

QDA Quadratic Discriminant Analysis

RBF Radial Basis Function

RBF SVM SVM with RBF kernel

ReLU Rectified Linear Unit

ROC receiver operating characteristic

ROCAUC area under the ROC curve

SAPID Spatially-Aware Protein Identification Algorithm

scRNA-seq single-cell RNA sequencing

SMOTE Synthetic Minority Oversampling Technique

SVM Supprot Vector Machine

TN true negatives

TOF time-of-flights

TP true positives

TPP Trans Proteomic Pipeline

TPR true positive rate

xCorr cross-correlation score

Chapter 1

Introduction

1.1 Protein Structure

Proteins are one of the important constituents of the cells. In fact, proteins have many critical roles in the cell such as immune reaction, structural support, regulation and signaling, among others [1]. amino acids (AA) are the building blocks of proteins, and there are 20 proteinogenic AA that form proteins in the living organisms. Each AA contains an amine group (-NH2), a carboxyl group (-COOH), and a variable functional group, or the side chain, which varies among different AAs (**Figure 1.1**).

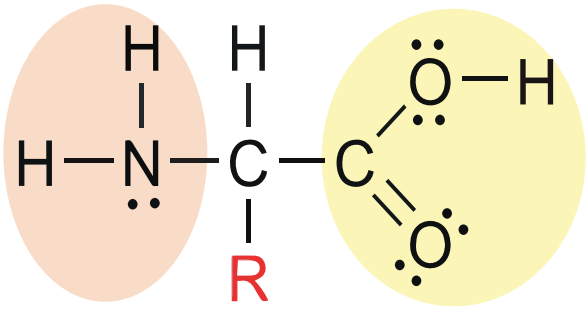

Figure 1.1: Generic structure of an amino acids — the basic block of proteins. The orange shading shows the amine group, and the yellow shading shows the carboxyl group. The variable functional group is shown with a red "R".

The side chains in AAs define their specific physical and chemical characteristics; for example, lysine and arginine are two basic AAs, and the amino

group in their side chains can become positively charged by gaining a proton, while aspartic acid and glutamic acid can become negatively charged and acidic by losing an electron [1]. Therefore, the AAs are categorised into several groups based on their properties: acidic, basic, aromatic, sulfur, uncharged hydrophilic, inactive hydrophobic, and others.

Different sequences of AAs can be linked together to form an infinite number of unique proteins with various lengths and chemical and physical properties; the linear sequence of AA in a protein is called its primary structure. This linking phenomena between adjacent AAs in proteins occur through covalent peptide bonds as shown in (**Figure 1.2**).

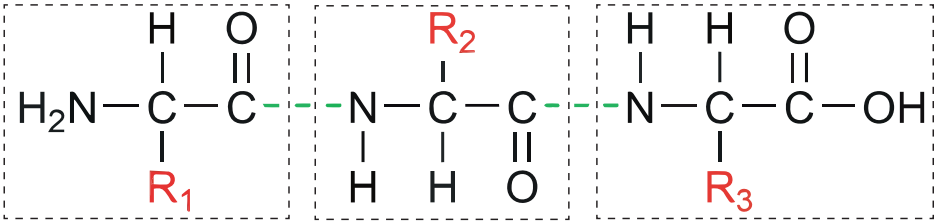

Figure 1.2: A three-peptide protein chemical structure showing the peptide bonds between the peptides. The peptide bonds are depicted as dashed green bonds, and the variable functional groups are shown with red "R". Also, each peptide is enclosed in a dashed rectangle.

1.1.1 Protein Function

Even though a protein can be characterised by its primary structure, its function is best understood by its 3-dimentional (3D) structure; The AAs side chains in a protein interact with each other through covalent and non-covalent interactions that ultimately define how it is folded into its 3D structure and what its functions are. Therefore, if there is a change in the primary structure of a given protein, the protein could fold differently and yield a loss-of-function or a gain-of-function in the protein which commonly occur in diseases such as cancer or some neurological diseases (Parkinson's disease). For example, p53 is a transcription factor with tumor-suppressing role in the cells, and it is the most frequent gene mutated in human cancers [2]. It has been shown that p53 mutation is mostly the result of only one faulty AA substitution [3]. Therefore,

identifying and characterising protein sequences is essential to understand the biological basis of diseases and reveal their biomarkers. Furthermore, it can enable the development of drugs targeting pathways where disease-causing proteins are implicated.

1.2 Mass Spectrometry-based Proteomics

The common low-throughput techniques to investigate proteins in a tissue are enzyme-linked immunosorbent assay (ELISA) and western blotting, but these methods can only analyse a few individual proteins at a time [4], [5]. In contrast to these low-throughput methods, mass spectrometry-based proteomics has been used for more than 30 years to detect the protein composition of biological samples with high sensitivity [6]. Its high throughput and sensitivity make it the method of choice for the comprehensive proteomics analysis of complex biological samples. Mass spectrometry methods are ideal to characterize gene products and their post-translational modifications (PTM) and allow the investigation of the functional state of the cells by identifying and quantifying the different proteins they express [7]. Mass spectrometry instruments (i.e. mass spectrometers) are composed of three main parts: an ionizer, one or more mass analysers, and a detector to record the number of ions coming from the mass analyser. In addition, the ionizer could be connected to an inlet device such as a liquid chromatograph that separates analytes based on their hydrophobicity [8]. The following section will be a summary of the most popular ionizers and mass analysers.

1.2.1 Ionizers

The ionizer's role is to ionize the analytes into a gaseous state which could then be transferred into the mass spectrometry instrument. There are two main types of ionizers that are used in mass spectrometry-based proteomics: Electrospray Ionization (ESI) [9] and Matrix-Assisted Laser Desorption Ionization (MALDI) [10].

Electrospray Ionization

In ESI, a high voltage is applied across a small capillary containing the analytes which produces a positively charged droplet that is drawn towards a negatively charged electrode. As the droplet is flying towards the electrode, it is heated, which evaporates the water and the electrostatic repulsion of positive ions causes the droplet to burst into smaller particles which are sprayed into the mass spectrometer [11].

Matrix-Assisted Laser Desorption Ionization

On the other hand, with MALDI, the liquid sample containing the protein mixture is mixed with a matrix that can absorb light from a laser, and then the mixture is dried up in a process which enables the co-crystallization of the matrix and the sample molecules [10]. The solid phase mixture is then bombarded with a laser beam (usually a nitrogen laser at 337 nm) that causes desorption and ionization of the sample molecules, which are then sent to the mass spectrometer [12].

1.2.2 Mass Analyzers

Mass analysers measure the mass-to-charge ratio (m/z) of the ions that enter the mass spectrometer. There are a number of mass analysers, such as Quadrupole mass analysers, time-of-flights (TOF), and Orbitrap which are among the most commonly used mass analysers [13]. Each of these mass analysers have their own advantages and disadvantages, with some being faster than others and having different levels of mass accuracy and resolution. For instance, Quadrupoles are affordable but have limited mass range and poor resolution. On the other hand, Orbitrap analysers have high resolving power that is ideal for proteomics. Furthermore, they are less expensive and require less maintenance than similar technologies [11]. Overall, all the mass analysers measure the m/z of the ions and produce a mass spectrum describing the relative abundance of each ion. Mass spectra and their use in protein identification are detailed in **Section 1.3**.

1.2.3 Liquid Chromatography

Peptides with similar molecular masses produce overlapping peaks in the mass spectrometer that makes it difficult to distinguish the peptides apart. Therefore, separation techniques such as liquid chromatography–mass spectrometry (LC–MS) can be employed to increase the dynamic range (measure of the smallest and biggest peak intensities detectable by the instrument) and reduce the chance that peptides with the same mass enter the ionizer simultaneously. The most common approach is to separate peptides based on hydrophobicity. During this process, peptides are eluted through a column packed with hydrophobic beads that causes the peptides to stick to the beads based on their hydrophobicity and allows small subset of peptides to enter the mass spectrometer at a given time [14].

1.3 Bottom-up proteomics

Ionizing intact proteins and identifying their m/z might seem like a logical approach for identifying proteins using mass spectrometers. However, ionizing large protein molecules is challenging, and mass spectrometers yield similar m/z values for large proteins of similar composition. Therefore, the proteins are first cleaved into smaller peptide molecules using enzymatic cleavages before ionization using a technique known as bottom-up or shotgun proteomics which is the most widely used approach in proteomics [15]. Despite the limitation of ionizing and analysing intact proteins, in "top-down proteomics" complex instruments and methodologies are employed to overcome these limitations [16]. Nevertheless, top-down proteomics is not the focus of this thesis and will not be explored any further. The main technique used to sequence peptides in bottom-up proteomics is called Tandem Mass Spectrometry (MS/MS).

1.3.1 Tandem mass spectrometry

Since mass spectrometers measure m/z values for each ion, the resulting m/z values can be shown as a mass spectrum with m/z values plotted against their

signal measured in the detector as shown in **Figure 1.3**.

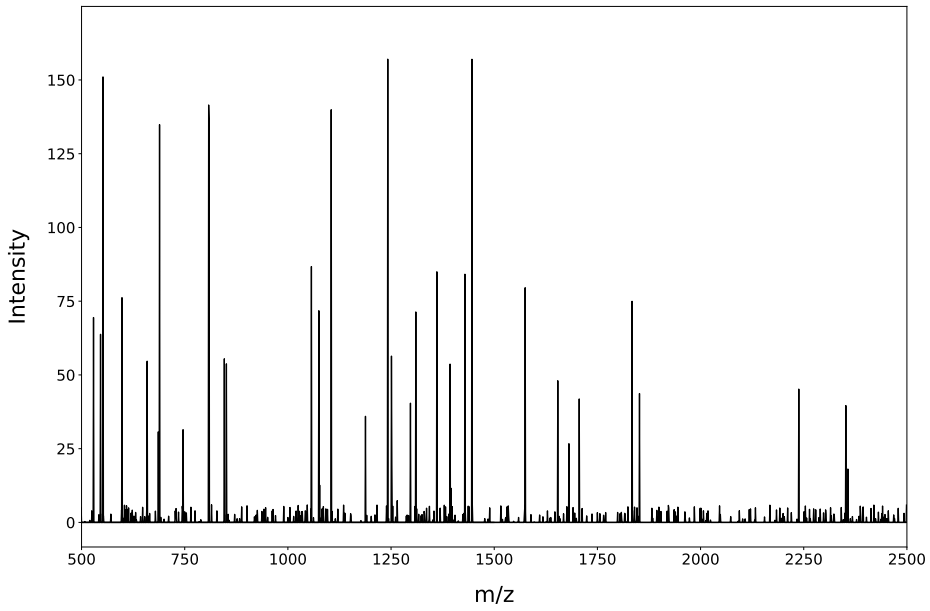

Figure 1.3: An example of a mass spectrum.

However, the mass spectrum (known as MS^1) is not sufficient to identify the AA sequence and the PTM in the peptide, so the peptide needs to be fragmented again. This step is usually done by selecting and then colliding the precursor ions from MS^1 spectrum with an inert gas such as nitrogen [17]. When the ions collide with the gas molecules, the peptide breaks along the peptide bonds (Figure 1.2) resulting in a number of sequence fragments known as b-type or y-type ions, where the charge is usually retained on the N-terminal or C-terminal portion of the fragmented ion, respectively [18]. Both of these ions are used to generate a tandem mass spectrum known as MS/MS spectrum (or MS^2 spectrum) which can be used to identify the fragmented peptides using database search algorithms [19].

1.3.2 Database search

Database search algorithms match the experimental MS/MS spectra with theoretical MS/MS spectra generated from *in silico* cleavage of a protein sequence database [20]. These algorithms match each mass spectrum with the peptide

sequence corresponding to the theoretical MS/MS spectrum that is the most similar to the experimental one and provide a score for each of these peptide-spectrum-matches (PSM). The peptide with the highest PSM score can be considered as the best match for a given mass spectrum.

SEQUEST

SEQUEST is the first automated software to match peptide mass spectra to peptide sequences [21]. This software package first calculates a Sp-Score for each PSM based on the number of ions that are matched between the experimental and theoretical mass spectra and only keeps the 500 PSM with the highest Sp-scores. Finally, SEQUEST calculates two important metrics for the selected 500 sequences to determine whether a peptide sequence is a confident match for a fragmented spectrum: cross-correlation score (xCorr), which measures the correlation between the theoretical and experimental mass spectra, and delta-correlation (deltaCN), which is the normalised difference between the best and second best PSM according to their xCorr.

Comet

Comet [22] is a direct descendent of SEQUEST with a fast cross-correlation calculation algorithm [23] implemented which allows it to score all the experimental tandem mass spectra with the peptides in the database. The efficiency in calculating xCorr scores for all PSM allows the calculation of the p-values and E-values [24]. Given that xCorr scores generated under different conditions or across multiple mass spectra are not directly comparable, Klammer et al. developed a p-value scoring scheme that can be used to compare PSM more appropriately [25]. Furthermore, the E-value of a PSM is calculated by doing a linear least squares regression on the log transform of the cumulative distribution function of the xCorr histograms [22]. Given the fast implementation of xCorr calculation, Comet can skip the preliminary scoring that is implemented in SEQUEST, however, it still calculates the Sp-Scores for the top scoring PSM for backward compatibility.

1.3.3 Confidence assessment of PSM

Database search algorithms such as Comet score all of the mass spectra with the peptide sequences in the database. The PSM with the highest score for each spectrum could be considered as the correct match. However, experimental mass spectra are typically noisy, and rarely exactly match with the theoretical mass spectra generated from the protein sequence database. Furthermore, some peptide sequences present in the analyzed sample may be absent from the database. Therefore, the PSM with the best match may be incorrect and simply using scores generated by the database search algorithm can result in many incorrectly identified mass spectra, which highlights the need for statistical inference to overcome this hurdle [26].

ProteinProphet

Coupling PeptideProphet [27] and ProteinProphet [28] is one of the most popular methods for protein identification in which confidence identification of peptides followed by confidence identification of proteins are done in sequence. More specifically, PeptideProphet generates a probability-based mixture model of correct and incorrect PSM using expectation maximization algorithm. The incorrect PSM distribution is obtained using a decoy database that is generated by shuffling or reversing sequences in a real protein database. Any PSM that are matched to decoys by the database search algorithm can be assumed as false hits and can be used to build null distributions for the scores generated by the search algorithm. The real and decoy sequences are usually combined prior to database search which is referred to as target-decoy search. Subsequently, ProteinProphet uses the confidence assessment probability and the identified peptides from PeptideProphet to estimate the probability that a given protein is identified in the sample. However, ProteinProphet only considers the PSM with highest score instead of all the PSM for a given peptide which is one of the drawbacks of ProteinProphet. This two-step process will be referred to as ProteinProphet in this thesis. Generally, the high-confidence PSM are chosen in terms of FDR that is the global estimate of the false pos-

itives resulting from the database search and can be calculated based on the total number of decoys hits (false positives) and the total number of hits [29].

Percolator

Percolator is a semi-supervised algorithm that trains a Support Vector Machine to distinguish correct and incorrect PSMs using 20 features from the database search algorithm and measurement from the mass spectrometer [30]. Percolator first trains a model to distinguish a small number of high scoring PSM from decoy PSM, and then applies the trained model on the entire dataset. This process is repeated until no new high scoring PSM are identified. Even though Percolator's performance has been regarded as state-of-art, its optimization objective is not totally clear [31]. Percolator calculates two metrics for each peptide or protein match: a q-value and a posterior error probability (PEP). The q-value is the rate of misclassification among a set of PSM, while the PEP is the probability of incorrect classification for a given PSM [32]. Similar to ProteinProphet, an FDR can be calculated for Percolator results using the number of decoy hits and total number of hits.

1.3.4 Limitations of bottom-up tandem mass spectrometry

In a conventional tandem mass spectrometry pipeline, proteins are extracted from a biological sample or tissue. The extracted proteins are digested using trypsin and the resulting solution is fed into the mass spectrometer. Therefore, with this pipeline, no information regarding the spatial localization of the molecules in a tissue can be obtained, unless the tissue is segmented using techniques such as laser capture microdissection [33]. Furthermore, despite its high-throughput proteome characterization capabilities, mass spectrometry tends to favour the identification of proteins of high abundance at the expense of low abundance proteins. Hence, proteins that are abundant in a specific tissue location but of low abundance in the overall sample are likely to be masked during the mass spectrometry analysis by proteins with a higher abundance

throughout the tissue. Identifying splice variants and protein isoforms with different PTM is still a challenging task for bottom-up MS/MS experiments [34], especially when different isoforms are distributed in different sections of a tissue. By identifying an important peptide unique to one isoform and missing other peptides associated to the other isoforms, the current bottom-up MS/MS experiments fail to detect all the variants of the proteins.

1.4 Mass spectrometry imaging

Mass Spectrometry Imaging (MSI) is an analytical technique that provides researchers with the spatial distribution of analytes, such as proteins or lipids in biological samples [35]. The application of MALDI-MSI in biological and biomedical fields has been expanded rapidly over the last decade and increasingly adopted for its molecular imaging capabilities [36], [37]. In MALDI-MSI, the biological tissue can be trypsinized *in situ* and then covered with a matrix. The laser beam is then shot at its surface at different tissue locations. The matrix absorbs the laser and ionizes the analytes at the tissue surface so that they can then be analysed by mass spectrometry [38]. This means that in addition to identifying and quantifying peptides and proteins, MALDI-MSI is capable of determining the spatial location of the analytes on the tissue. Furthermore, MSI can solve the issues discussed in section 1.3.4 and facilitate biomarker discovery. Other than MALDI-MSI, other ionization techniques such as desorption electrospray ionisation (DESI) [39] and secondary ion mass spectrometry [40] are also used in MSI, and they are coupled to conventional mass analysers such as Orbitrap, TOF, or Quadrupoles.

1.4.1 Applications of MSI

MSI has been adopted in the biomedical field over the past decade in the context of biomarker discovery, drug distribution in tissues, and microbiome studies in hospital settings [41]–[43]. For instance, Yanagisawa et al. identified a proteomics signature associated with non-small cell lung cancer and were able

to classify healthy and diseased tissues with 100% accuracy using MSI [44]. Also, Cazares et al. compared adenocarcinoma and benign prostate tissues using MALDI-MSI and identified the MEKK2 kinase as a putative biomarker to discriminate cancerous and benign tissues [45]. Lemaire et al. identified the 11S proteasome activator complex as a potential biomarker of ovarian carcinoma using MALDI mass spectrometry and validated its localization using MSI and immunocytochemistry techniques [46]. All of these approaches highlight the potential of MSI in biomedicine and precision medicine research.

1.4.2 Limitations of MSI

One major challenge of MSI is its low protein identification sensitivity. MSI is much less sensitive than traditional mass spectrometry, mainly due to a less efficient peptide ionization. Furthermore, in the commonly used MALDI-MSI, samples are directly transmitted into the mass spectrometer without liquid chromatography separation lowering dynamic range of the analysed peaks. The matrix applied on the tissue for MALDI analysis can also interfere with ionization and add noise to the acquired mass spectra [47]. Furthermore, the computational identification of peptides from MSI mass spectra has also remained highly rudimentary and mainly based on computational analyses designed for traditional mass spectrometry [48]. Peptides in mass spectrometry are typically identified from MS/MS spectra using database search as explained in **Section 1.3.2**. When identifying peptides in a MSI pixel (i.e. location where the laser was shot), these approaches fail to make use of the contextual information present in neighboring pixels, such as the peptides present in their neighborhoods. These information could improve the poor protein identification sensitivity of MSI that hinders its ability to characterize the proteome of tissues and limits its biomarker discovery capabilities, and it is the main idea investigated in this thesis.

1.5 Machine learning

"Machine learning is a branch of artificial intelligence that systematically applies algorithms to synthesize the underlying relationships among data and information" [49]. In his famous book, Machine Learning (1997) [50], Tom Mitchell defines learning as follows:

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

For example, for a program that learns to recommend new videos to an online video-sharing platform users (e.g. YouTube users), the task T is recommending videos to user, performance measure P is the proportion of "appropriate" recommended videos to the users, and the training experience E is a database of videos watched by users with similar preferences. Another example could be an algorithm used in hospitals to predict the time of stroke in hospital patients, the task T is predicting the time of stroke, performance measure P is how close are the predicted time and actual time, and the training experience E is the database of previous cases of stroke and their vitals and the measured time of stroke. Therefore, these three elements change depending on the machine learning problem and should be carefully defined. In the following sub-sections, there will be a brief explanation of the types of task, experience, and performance measures used in this thesis.

1.5.1 Task (T)

In simple terms, "task" is our expectation from a given learning algorithm and should not be confused with the notion of "learning" itself [51]. For example, in our previous example of video recommendation algorithm, the task is to determine which videos to recommend out of many videos that are aligned with users' preferences, formally known as a classification task,

whereas in the stroke prediction algorithm, the task is to predict the time of stroke, formally known as a regression task. However, the process by which the learning algorithm acquire such *skills* is irrelevant to the task at hand [51]. Classification is the only type of task used in this thesis which will be further discussed.

Classification

In a classification task, the objective is to automatically label unlabeled data. The labeled or unlabeled data points that are processed by learning algorithms are called "examples" or "instances" which are the collection of features extracted from subjects, events, or processes, and these two terms are used interchangeably throughout this thesis. One instance is defined with a n-dimensional vector $x_i \in \mathbb{R}^n$ where n is the number of features for the instance i, and each feature for that instance can be written as $x_i^{(j)}, j = 1, \cdots, n$. Additionally, each instance i is associated to a label or target or class $y_i \in \mathbb{R}$. For instance, in a binary cancer diagnosis problem the labels can either be "Healthy" or "Diseased" for the biopsies, which can be represented with 0 and 1, respectively. The 0 and 1 labels can also be referred to as "Negative" and "Positive" classes, respectively. Finally, a collection of labeled instances could be represented in a dataset $\{(x_i, y_i)\}_{i=1}^N$ where N is the number of instances in the dataset, and y_i is the corresponding label for x_i [52].

The learning algorithm's task is to predict the labels corresponding to unlabeled instances. For a binary classification task, the learning algorithm, or classifier, produces a function $f : \mathbb{R}^N \to \{0, 1\}$. In other words, given a vector $x \in \mathbb{R}^n$, function f which represents the classifier, is able to predict to which group, 0 or 1, does the example x belong. It is important to note that some algorithms produce a probability distribution rather than a discrete classification result, and there needs to be a probability threshold(s) by which the classes are defined. This concept will be described in detail in the next section.

1.5.2 Experience (E)

Experience refers to the process of learning from data [51]. Supervised learning is one type of experience where the algorithm can learn from a dataset containing both the features and the labels, or targets, denoted with y_i in the previous section. The term *supervised* refers to the notion that the labels are shown by an "instructor" to the model so that the model can learn how to distinguish examples belonging to each class. A more in-depth explanation of each of the supervised classification algorithms used in this thesis follows.

Gaussian classifiers

In a binary classification problem, the probability that a new instance x has label $y = k$ where $k \in \{0, 1\}$ could be written as the conditional probability $P(y = k \mid x)$, and it could be formulated using the Bayes rule shown in **Equation 1.1**:

$$\begin{aligned} P(y = k \mid x) &= \frac{P(x \mid y = k)P(y = k)}{P(x)} \\ &= \frac{P(x \mid y = k)P(y = k)}{P(x \mid y = 1)P(y = 1) + P(x \mid y = 0)P(y = 0)} \end{aligned} \tag{1.1}$$

The above formulation can be shortened by replacing $P(x \mid y = k)$ with $f_k(x)$ and $P(y = k)$ with π_k as follows (note that π_k has nothing to do with conventional $\pi = 3.1415\ldots$, and it is only a symbol denoting the prior probability of class k):

$$P(y = k \mid x) = \frac{f_k(x)\pi_k}{\sum_y f_y(x)\pi_y} \tag{1.2}$$

Any value of k that maximizes $P(y = k \mid x)$ is considered the label for x which can be written as:

$$h(x) = \underset{k}{\operatorname{argmax}}\, P(y = k \mid x) = \underset{k}{\operatorname{argmax}}\, \frac{f_k(x)\pi_k}{\sum_y f_y(x)\pi_y} \tag{1.3}$$

The argmax function means the value of k which maximizes the value of $h(x)$. Since the denominator is a constant and independent of k, the above expression could be simplified as:

$$h(x) = \underset{k}{\operatorname{argmax}}\, f_k(x)\pi_k \tag{1.4}$$

We could assume that $f_k(x)$ for both classes are multivariate Gaussian and create a classifier $h(x)$ as follows [53]:

$$h(x) = \underset{k}{\operatorname{argmax}}\, \delta_k(x) \tag{1.5}$$

where,

$$\delta_k(x) = -\frac{1}{2}\log \mid \Sigma_k \mid -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log \pi_k$$

Σ_k is the covariance matrix for features where class is k, $\mid \Sigma_k \mid$ is its determinant, and μ_k is its mean. If it is assumed that Σ_k and μ_k can be directly estimated from the training dataset, then the above classifier is referred to as Quadratic Discriminant Analysis (QDA) which has a quadratic decision boundary as shown in the left panel of **Figure 1.4**. The decision boundary occurs where $\delta(x) = \delta_0(x)$ which is where the probability of belonging to either negative or positive class is equal.

Furthermore, if we assume that the covariance matrices are equal between the two classes $\Sigma_0 = \Sigma_1 = \Sigma$ can be simplified as:

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log \pi_k \tag{1.6}$$

which is called the Linear Discriminant Analysis (LDA) and has a linear decision boundary as shown in the right panel of **Figure 1.4**. Furthermore, the class prior probabilities π_k can be changed for each class to adjust the decision boundary of QDA and LDA in order to control the sensitivity and specificity of the classifiers which is reflected in the contours of **Figure 1.4**.

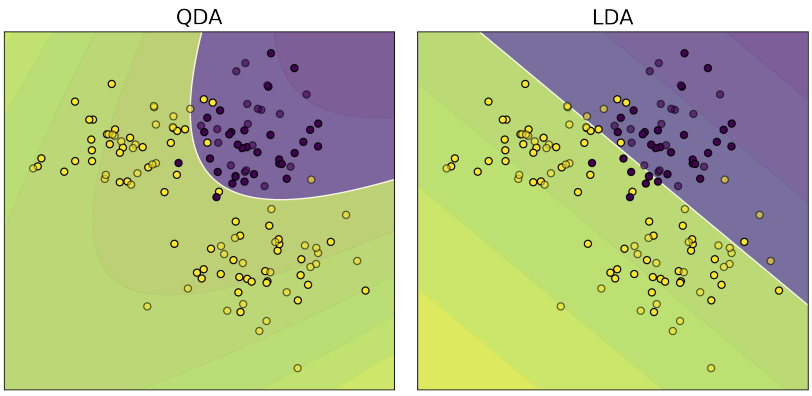

Figure 1.4: The decision boundaries of quadratic and linear discriminant analysis algorithms. Left panel shows the quadratic decision boundary of QDA; Right panel shows the linear decision boundary of LDA. The light contours show the decision boundary of the classifiers when prior probabilities π_k are changed. The yellow and purple points belong to different classes from an artificially generated imbalanced dataset with two features. The examples that were not used in the training of the models are transparent.

Naïve Bayes classifier

For Bayes rule shown in Equation 1.1, a very naïve assumption could be made regarding $P(x \mid y = k)$ stating that all the features are independent (conditional independence), that is:

$$P(x \mid y = k) = P(x^1, x^2, x^3, \ldots, x^j \mid y = k) = \prod_{j=1}^{n} P(x^j \mid k) \tag{1.7}$$

Bayes rule can then be re-written as:

$$P(y \mid x = k) = \frac{P(y = k) \prod_{j=1}^{n} P(x^j \mid k)}{P(x)} \tag{1.8}$$

Any value of k that maximizes $P(y = k \mid x)$ could be considered as the label for x which can be formulated as:

$$h(x) = \underset{k}{\operatorname{argmax}}\, P(y = k \mid x) = \underset{k}{\operatorname{argmax}}\, P(y = k) \prod_{j=1} P(x^j \mid k) \tag{1.9}$$

Note that since $P(x)$ is a constant and independent of k, it was eliminated from $h(x)$. Furthermore, maximum a posteriori estimation can be used to find the parameters maximizing $h(x)$. It turns out that similar to QDA and LDA, $P(y = k)$ can be estimated as the proportion of examples belonging to class k in the training dataset. Furthermore, different Naïve Bayes classifiers vary by the assumption they make regarding the distribution of $P(x^j \mid k)$: in the Gaussian Naïve Bayes (Gaussian NB) classifier the distribution of the features belonging to each class is assumed to be Gaussian. **Figure 1.5** shows the decision boundary of a Gaussian NB. Similar to QDA and LDA, the prior probability $P(y = k)$ can be manipulated to change the decision boundary which is illustrated with the contours in **Figure 1.5**.

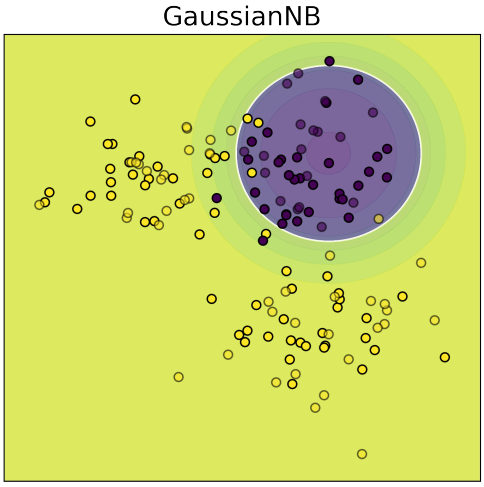

Figure 1.5: Decision boundary of Gaussian Naïve Bayes classifier. The decision boundary is shown with the white circle separating the instances from each group. The contours illustrate different decision boundary that could be created using different prior probabilities. The yellow and purple points belong to different classes from an artificially generated imbalanced dataset with two features. The examples that were not used in the training of the model are transparent.

Logistic regression classifier

The logistic regression model can be represented as follows:

$$f_{w,b}(x) = \frac{1}{1 + \mathrm{e}^{-(w^T x + b)}} \tag{1.10}$$

where w is a n-dimensional vector of parameters (weights), and b is a real number. The above function models the posterior probability $P(y \mid x)$ and is called the logistic function or sigmoid function with the range of output from 0 to 1, which makes it easy to compute probability for an observation.

The optimal values of w and b for a given classification problem needs to be calculated by maximising the likelihood of the model defined as follows:

$$\prod_{i=1\ldots n} f_{w,b}(x_i)^{y_i}(1 - f_{w,b}(x_i))^{(1-y_i)} \tag{1.11}$$

Because of the exponential nature of the logistic regression, it is much easier to maximize the logarithm of the above function which is called the log-likelihood and has the same solution as the likelihood function and is done using the gradient-descent algorithm.

The left panel in **Figure 1.6** shows the linear decision boundary of a logistic regression model, and the right panel shows the sigmoid function representing the model and the 0.5 arbitrarily probability threshold separating the two classes. By changing this threshold (i.e. moving it up and down in the figure), the number of false negative and false positive identifications can be adjusted which consequently affect sensitivity and specificity of the classification. The new decision boundaries as a result of different thresholds are shown as contour in the right panel of **Figure 1.6**.

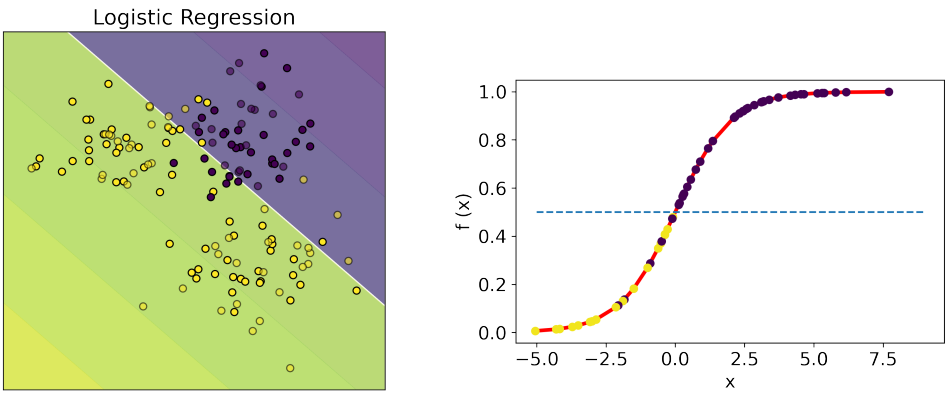

Figure 1.6: Decision boundary of a logistic regression classifier and its sigmoid function. Left panel shows the probability distribution of the logistic model distinguishing the two classes, and the contours are the new decision boundaries as a result of different thresholds; Right panel shows the sigmoid function representing the model and an arbitrary threshold line at 0.5 separating the examples from each class. The yellow and purple points belong to different classes from an artificially generated imbalanced dataset with two features. The examples that were not used in the training of the models are transparent.

Decision tree classifier

Imagine that we divide the feature space in a given dataset into m non-overlapping subsets (or partitions) $R_1, R_2, \ldots, R_m$ such that each instance only falls into one subset. We could define a function that predicts the label for an instance x as follows [54]:

$$f(x) = \sum_{i=1}^{m} c_m I \{x \in R_m\} \tag{1.12}$$

c_m is a constant value determined from the majority of the labels in subset m, and I is an identity function which returns 1 when $x \in R_m$ and returns 0 when $x \notin R_m$. Therefore, each new instance's label is determined from the subset it falls in. The left panel in **Figure 1.7** shows the decision boundary of a decision tree classifier that has divided the feature space into 5 subsets. There are several strategies to partition the feature space into m subsets, but they all try to minimize the impurity measure in them which can be calculated using Gini index or Cross-entropy. In other words, they try to find partitions

such that each partition is all or mostly one class or the other, rather than a mix of both classes. Given that there are N_m observations in a subset R_m, the proportion of observations belonging to class k in a subset R_m can be formulated as follows:

$$p_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m}^{N_m} I(y_i = k) \tag{1.13}$$

Subsequently, the impurity in a subset R_m can be calculated using Gini index or Cross-entropy as follows:

$$\text{Gini index} = \sum_{k=1}^{k} p_{mk}(1 - p_{mk}) \tag{1.14}$$

$$\text{Cross-entropy} = -\sum_{k=1}^{k} p_{mk} \log p_{mk} \tag{1.15}$$

The impurity measures quantify the homogeneity of the subsets and how well the partitions are made. Briefly, the decision tree algorithm partitions the feature space by recursively dividing the feature space and then choosing the best split that minimizes impurity measure. Therefore, by minimizing the Gini index or Cross-entropy useful partitions can be created for classifications.

Random forest classifier

In a random forest classifier, a collection of uncorrelated decision tree classifiers are built and then averaged [54]. Since decision trees are noisy and have high variance (variance is how different the estimated labels are for different training datasets), averaging many decision trees reduces the variance and improves classifications. The right panel in **Figure 1.7** shows the decision boundary of a random forest classifier which looks more complex than the decision tree classifier, since it is composed of many trees. Furthermore, the contours in the random forest plot show the probability distribution of different trees composing the forest.

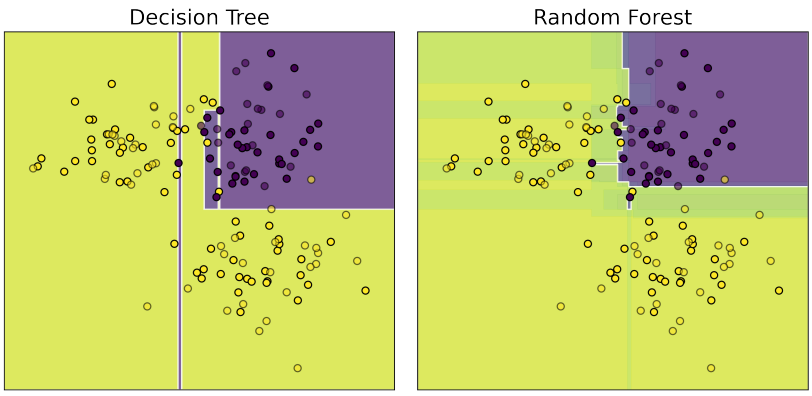

Figure 1.7: Decision boundaries of a decision tree and a random forest classifiers. Left panel shows the decision boundary of a decision tree; Right panel shows the decision boundaries of a random forest. The decision boundaries are indicated with a white line around each region. The contours in the Random Forest plot show the probability distribution of different trees composing the forest. The yellow and purple points belong to different classes from an artificially generated imbalanced dataset with two features. The examples that were not used in the training of the models are transparent.

Support vector machine

Consider a dataset with binary class instances that are linearly separable; that is, a straight line can separate the two classes perfectly. A Supprot Vector Machine (SVM) tries to find a decision function in the form of a hyperplane (or line) that not only separates the two classes, but also stays as far away as possible from the closest training examples [55] [56]. The distance between the decision boundary and the closest instance is called the margin, and SVM's objective is to maximize this margin. The points that are closer to the hyperplane are called support vectors which influence the position and orientation of the decision function. However, finding such line or hyperplane is not always possible as real-world data are noisy and some instances from each class are likely to overlap. Therefore, SVM tries to find a good balance between the width of the margin and number of instances violating the margin constraint and making sure as many points as possible are on the correct side of the margin. The left panel in **Figure 1.8** shows the decision boundary of a linear SVM and the margins which are parallel and equal distance to the decision

boundary, and their values are +1 and -1 on the decision function.

SVM can use the "kernel trick" to project the features into a higher dimensional space so that a non-linear relationship between the features could be captured [57]. Radial Basis Function (RBF) kernel is one of the most popular kernels used with SVM which can capture non-linear relationships among the data as shown in the right panel of **Figure 1.8**.

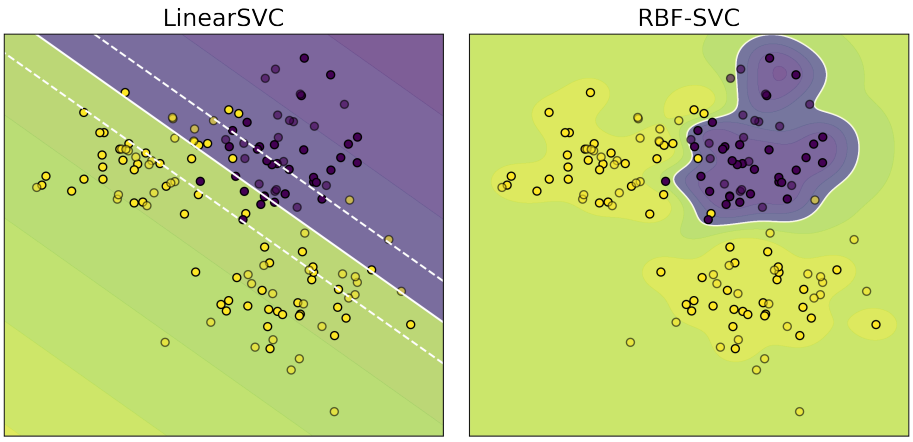

Figure 1.8: Decision boundaries of two support vector machine classifiers with linear and RBF kernel. Left panel shows the decision boundary (white line) of a linear SVM classifier and the margins (white dotted lines) which are parallel and equal distance to the decision boundary. Right panel shows the decision boundary of a SVM classifier with RBF kernel. The contours show the decision boundaries with different parameters. The yellow and purple points belong to different classes from an artificially generated imbalanced dataset with two features. The examples that were not used in the training of the models are transparent.

Artificial neural networks

An Artificial Neural Network (ANN), or neural network, is a machine learning model constructed based on the representation of a brain, in which neurons are connected to each other and propagating signals. ANN are built with different types of architecture to accomplish their goals. The Perceptron can be considered as the simplest ANN architecture as shown in **Figure 1.9**, where a piecewise step function is applied on the weighted sum of the input to produce an output for classification or regression depending on the step function [55].

However, the Perceptron can only have a linear decision boundary and capture linear relationship in the data and is not suitable for complex datasets. This limitation has been eliminated by staking multiple Perceptrons together as shown in the left panel of **Figure 1.10** which is called a Multi-layer Perceptron, or neural network that is composed of one input layer, one or more hidden layers, and one output layer. Each layer has a potentially different number of nodes, and each node can be considered as a Perceptron which applies an activation function on the weighted sum of the input from the preceding nodes. Rectified Linear Unit (ReLU) $ReLU(z) = max(0, z)$ is a popular activation function in neural network [58].

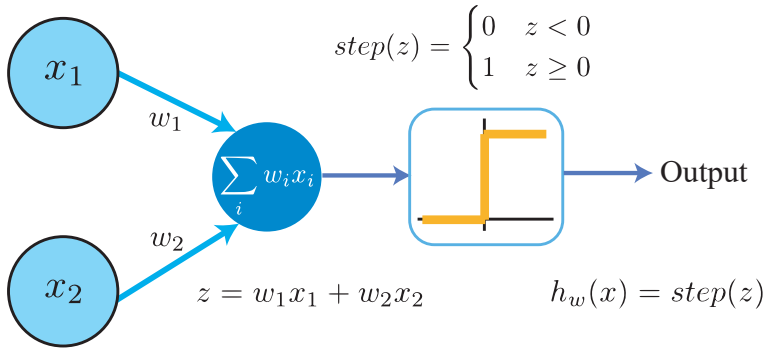

Figure 1.9: Network diagram of a perceptron with two inputs. The two inputs (x_1 *and* x_2) and their associated weights (w_1 *and* w_2) are linearly combined and then a step function is applied on the resulting sum. The outputted values could be used for classification or regression depending on the activation function.

Furthermore, depending on the classification problem, different architectures and activation functions could be used in the output layer such as the sigmoid function explained in section 1.5.2 for binary classification. The weights in a neural network are determined using the back-propagation algorithm in which a prediction is made for one of the training examples using random weight values. Subsequently, the contribution of each connection to the error is measured and the weights are tweaked to reduce the error through Gradient Descent steps [51]. Neural networks can capture non-linear relationships in the data as shown in the right panel of **Figure 1.10**.

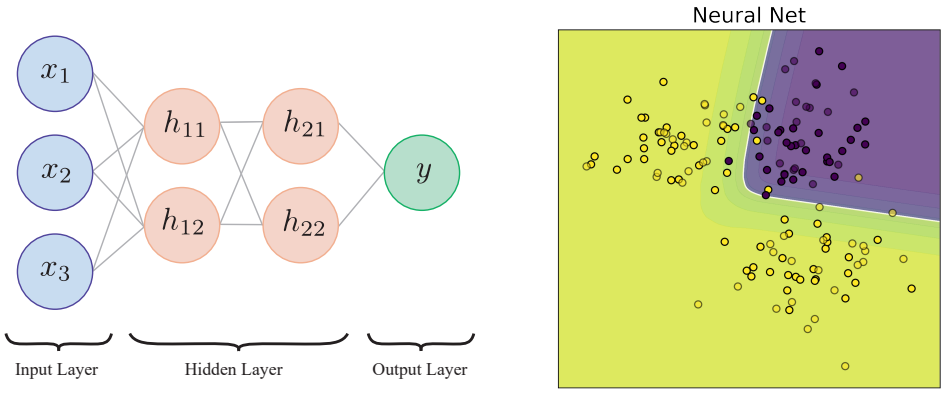

Figure 1.10: Simple architecture of a neural network, and visualization of the decision boundary of a neural network. Left panel shows 3 nodes in the input layer, 2 hidden layers with 2 nodes each, and the output layer with one node. Right panel shows the decision boundary of a neural network with sigmoid activation function as its output. The contour shows the probability of different examples as modeled by the classifier. The yellow and purple points belong to different classes from an artificially generated imbalanced dataset with two features. The examples that were not used in the training of the model are transparent.

1.5.3 Performance measure (P)

The metrics to evaluate the performance of a machine learning algorithms should be chosen according to the task that is defined for a given problem. Since classification is the focus of this thesis, only metrics used for classification will be discussed.

Let a binary classification task where each example $x_i \in \mathbb{R}^n$ and its corresponding true label (or gold standard) $y_i \in \{0, 1\}$ (0 and 1 denote negative and positive examples, respectively). A classifier f predicts the class of example x_i such that $f(x_i) \in \{0, 1\}$. The resulting prediction for each example can be categorised into 4 groups as shown in **Figure 1.11**: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) [59].

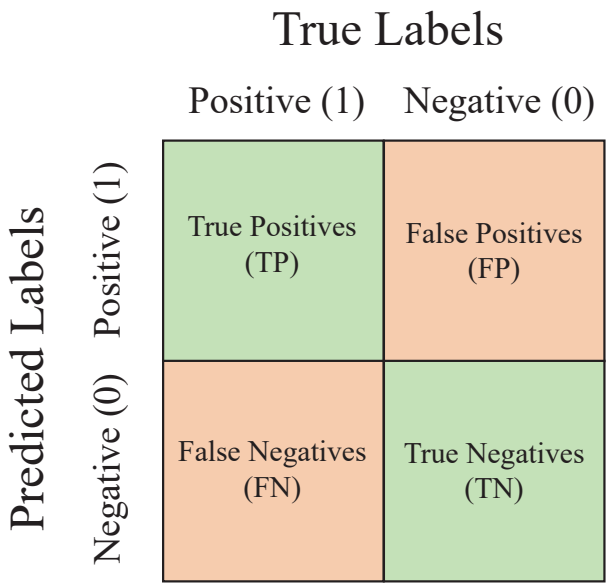

Figure 1.11: Confusion matrix with 4 categories of prediction. The green squares indicate the correct predictions, and the orange squares show the wrong predictions.

Accuracy

Accuracy is defined as the overall proportion of true predictions. However, in an imbalance dataset where the number of positive and negative examples are not similar, accuracy is not an informative metric.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP} \tag{1.16}$$

Recall

Recall (sensitivity or true positive rate (TPR)) is the proportion of positive examples that are correctly predicted to be positive.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{1.17}$$

Specificity

Specificity is the proportion of negative examples that are correctly predicted to be negative.

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{1.18}$$

Precision

Precision or positive predictive value (PPV) is the proportion of true positives with respect to all the examples that the classifiers predicted as positive.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1.19}$$

F_β measure

The F_β measure is the harmonic mean between precision and recall which measures the trade-off between the two metrics, and it is calculated as follows [60]:

$$F_\beta = (1 + \beta^2)\frac{Precision \times Recall}{(\beta^2 \times Precision) + Recall} \tag{1.20}$$

where β is the parameter that controls the importance given to precision and recall. When $\beta = 1$, also referred to as F1 measure, both precision and recall have the same importance; when $\beta = 2$, recall is twice as important as precision. When $\beta = 0.5$, precision is twice as important as recall.

Area under receiver operating characteristic curve

The receiver operating characteristic (ROC) curve summarizes the performance of a classifier at different values of TPR and false positive rate (FPR), and is generated by changing the decision boundary of the classifiers as explained previously [60]. The area under the ROC curve (ROCAUC) can be interpreted as the probability that a randomly chosen positive instance is ranked above a negative instance. Furthermore, ROCAUC is independent of threshold and often used to compare classifiers.

Area under precision and recall curve

Similar to ROC curve, the precision and recall (PR) curve summarizes the performance of a classifier at different values of precision and recall [59]. The

area under the PR curve (PRAUC) can be used to compare classifiers; however, contrary to ROCAUC, it does not have a probabilistic interpretation. Also, when there is a class imbalance and the minority class is the positive class, PRAUC is more informative than ROCAUC because it considers all of the rare events.

1.5.4 Statistical comparison of the classifiers

Comparing the performance of two or more learning algorithms with statistical tests is an important part of machine learning research. Performance is any monotonic measure of a classifier's performance on a given dataset which were described in the previous section. Simply choosing the classifier with the highest performance measure is not acceptable as the over-performance could be due to chance. There are various strategies and statistical tests developed for this purpose; however, discussing the limitations and assumptions behind each test is necessary before making any conclusion from the statistical tests.

Cross-validation (CV) is a commonly used method for the comparison of classifiers in machine learning, especially when the dataset is small. In a k-fold-CV, the training data is divided into k sets, which could be overlapping or non-overlapping. For each k iteration of the algorithm, one set is chosen for testing while the other sets will be used for training. Therefore, at each iteration, each classifier is trained on the training sets and tested on the test set. The performance of the classifiers on the test sets can be compared using tests such as paired t-test. However, this process violates the independence assumption necessary for t-test or ANOVA because the training sets overlap, and the variance of the performance measures are generally underestimated compared to when training sets were independent resulting in a high type I error probability as demonstrated by Dietterich [61]. For instance, in a 10-fold-CV experiment, 80% of the instances are shared between any pair of folds which results in high probability of finding *significance* when there is none. Alternative methods such as 5×2-fold-CV t-test [61] and corrected resample t-test [62], which incorporates overlap into variance estimation, have been pro-

posed. In this thesis, due to the presence of a sufficiently large training and testing dataset we will perform a 10-fold training and testing without undermining the assumptions underlying statistical tests which is further explained in the next chapter.

Chapter 2

General Methodology

In this chapter, the common steps and methods employed to process and generate the datasets that are used in the next two chapters are described. First, the details of the raw MSI dataset are explained followed by the downsampling scheme used to generate datasets with less spectra. Then, database search followed by confidence assessment with ProteinProphet and Percolator are explained. Afterward, the feature engineering process used to create the datasets with spatial features are explained. Finally, FDR calculation and statistical tests to compare different classifiers are described.

2.1 Dataset

We used the publicly available tandem mass spectrometry (MS/MS) dataset of mouse uterus model system that was downloaded from Mass Spectrometry Interactive Virtual Environment website `https://massive.ucsd.edu/ProteoSAFe/static/massive.jsp` with accession number MSV000084421 [63]. The dataset contains two 6×4 pixels tissue sections of mouse uterus that contain three cell types: stroma, glandular epithelium, and luminal epithelium. Furthermore, one tissue is a stromal-dominant tissue, which we used to train our algorithm, whereas the other one is a luminal-epithelium-dominant tissue, which we used to test and benchmark our models (**Figure 2.1**). Even though each of the tissues in this dataset had 24 pixels, it was the only available MS/MS dataset for MSI that we could access. The tissues were analysed with

Nanodroplet Processing in One pot for Trace Samples (nanoPOTS) coupled with laser capture microdissection at 100 μm spatial resolution followed by LC–MS analysis. The researchers used MaxQuant [64] with match-between-run setting and detected 1,764 and 2,357 unique proteins in in the stromal-dominant and luminal-epithelium-dominant tissues, respectively. Also, each raw dataset contained on average 36,000 spectra. We used the MS/MS data of all the pixels and their relative coordinates on each tissue.

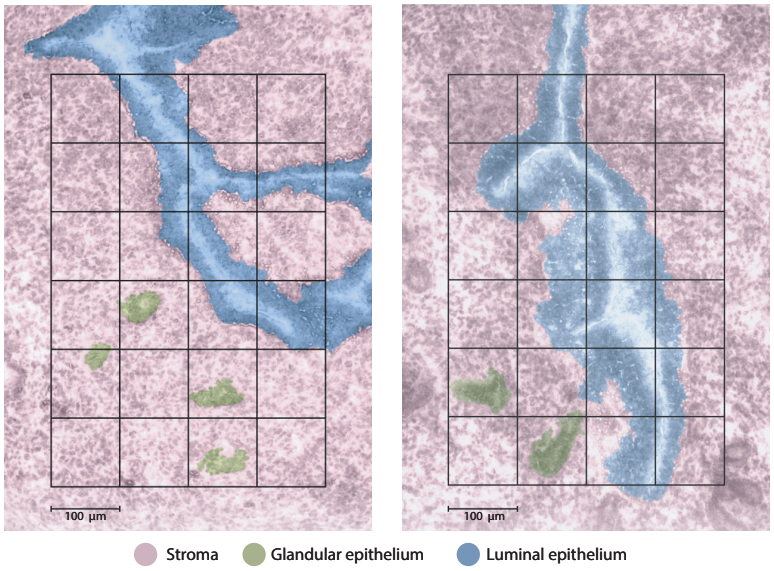

Figure 2.1: Pseudo-color optical illustration of two dataset used in this thesis. Left panel is the stromal-dominant tissue that was used to train our machine learning model; Right panels is the luminal epithelium-dominant dataset used to benchmark our model. Scale bar, 100μm. Obtained from Piehowski et al. [63]

2.2 Database search

For each pixel, the raw MS/MS instruments files (.RAW) were converted into open format (.mzML) using the msconvert tool in Trans Proteomic Pipeline (TPP) version v5.2.0 which is a free open-source software platform for proteomics data analysis [65]. In order to match the spectra in the MSI datasets with proteins, database search was performed on the converted files using Comet (version 2018.01 rev.4) [22]. Mass spectra were searched against the

UniProt *Mus Musculus* database downloaded on October 2016 and contained 16,839 proteins and 16,839 decoys generated by reversing the protein sequences. Carbamidomethylation of cysteine was set as a fixed modification and N-terminal acetylation and oxidation of methionine were allowed as variable modifications, and a maximum of two missed enzyme cleavages were allowed in a peptide. We then used ProteinProphet and Percolator for protein detection confidence assessment on the generated PSM.

2.3 Mass spectrum downsampling

In order to artificially generate MSI dataset with less number of spectra for benchmarking purposes (explained in section 3.3), we devised a downsampling scheme where certain proportions of the PSM in the database search files are randomly eliminated. Herein, we chose nine downsampling ratios: 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, and 0.90. For example, in the 0.10 downsampling file, 10% of the PSM from the original dataset were randomly eliminated. Note that *downsampling* should not be confused with the data sampling performed in machine learning analysis of the imbalanced datasets to change the number of examples in each class.

2.4 Protein detection confidence assessment

Confidence assessment of the protein matches in each pixel was done separately by analysing their database search results using ProteinProphet and Percolator.

2.4.1 ProteinProphet

We used PeptideProphet and ProteinProphet that are integrated into TPP (version v5.2.0). Minimum peptide length of 7, probability threshold of 0.0, and accurate PPM mass binning were used to run both software.

2.4.2 Percolator

Crux (version 3.2) mass spectrometry analysis toolkit was used to run Percolator (version 3.05.nightly-106-2c8457a7) on the dataset [66]. Protein detection probabilities were inferred from the picked protein-level FDR using the concatenated target-decoy search strategy.

2.5 Feature engineering

Prior to feature engineering, all the database search files were converted into mzIdentML which is an exchange standard in proteomics designed by the Proteomics Standards Initiative [67]. We used the *idconvert* tool shipped with the ProteoWizard (version 3.0.20361) which is a set of open-source cross-platform tools and libraries for proteomics [68].

The Comet database search results, now in mzIdentML format, were parsed and features were extracted and created for each protein in each pixel. Overall, the features can be divided into two categories: local (from pixel-of-interest (POI)), and spatial (from neighbouring pixels). Therefore, for a given protein on a given pixel (which we also refer to it as POI), some features are derived from the spectral information on the POI, and some features are derived from the spectral information on the adjacent pixels to POI. A naming convention used to name the features is that the local features extracted from a pixel are ended with *POI* referring to the *pixel-of-interest*, while the spatial features from the neighbouring pixels lack this suffix.

We have defined the *neighbourhood* of a POI as the most adjacent pixel in the horizontal, vertical, and diagonal directions. More specifically, if the coordinate of a given POI is (x_{POI}, y_{POI}), the set of coordinates from adjacent pixels defined by C is determined as follows:

$$C = \{(x_{adj}, y_{adj}) \mid 1 \geq x_{POI} - x_{adj} \geq -1 \; and \; 1 \geq y_{POI} - y_{adj} \geq -1, y_{adj} \in \mathbb{W}\} \tag{2.1}$$

The notion of neighbourhood is visualised in **Figure 2.2**. Therefore, at a given POI, there could be 8, 5, or 3 neighbouring pixels which we will be using

to build the spatial features.

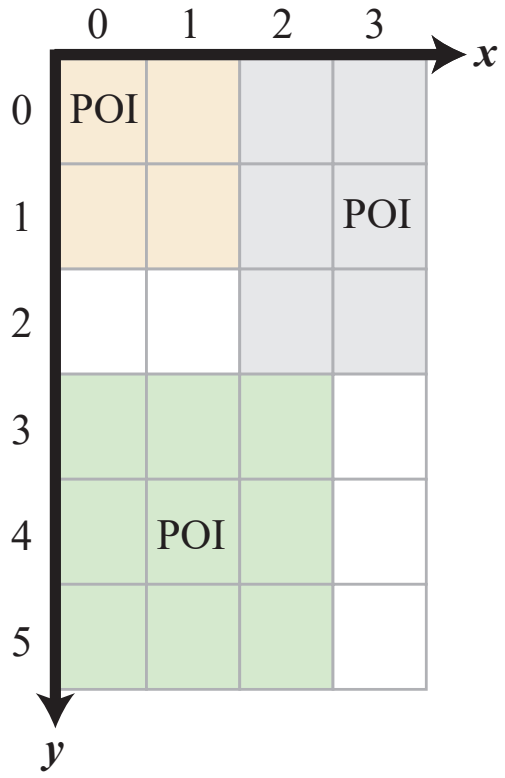

Figure 2.2: Visualising the neighbourhood definition at three randomly chosen POIs. Each of the three POIs and their neighbouring pixels are colored with the same color. The number of neighbours is the highest (8 pixels) for the green POI, is intermediate (5 pixels) for the gray POI, and is the smallest (3 pixels) for the yellow POI.

Even though the Comet database search reports 5 PSM for each spectrum, we have used the top ranked (based on xCorr) PSM to build our features. The only exception is *xCorr_5_POI* which was calculated based on the best and the worst (5^{th} ranked) xCorr for a given spectrum. It is important to remember that proteins could be associated to a single or multiple spectra (i.e. one or multiple top PSM) at a given pixel from which we derived the features. In a 6×4 pixels dataset, a protein could appear at most 24 times and at least 1 time. The list of features and their descriptions are in **Table A.1**.

2.6 Protein identification labels by ProteinProphet

We employed ProteinProphet probabilities, P_{PP}, and their 1% FDR threshold, P_{thr}, to assign labels to the proteins (non-decoys) in the dataset as shown in **Equation 2.2**:

$$f(P_{thr}, P_{PP}) = \begin{cases} 1 & P_{PP} \geq P_{thr} \\ 0 & P_{PP} < P_{thr} \end{cases} \tag{2.2}$$

All the decoys in the dataset were assigned label zero regardless of their ProteinProphet probabilities because they represent protein sequences which are incorrectly identified. Label "1" is associated to the proteins that are confidently identified in a pixel, and label "0" indicates the proteins that are not confidently identified. Therefore, the final dataset that was used for the downstream machine learning analysis has tabular format with rows being the protein instances and columns being the features, and one column for the labels. The term *instance* refers to a protein or decoy identification at a given pixel. All the *positive* instances are confidently identified proteins, but the *negative* instances could be decoys or proteins that are not confidently identified in a pixel.

2.7 Decoy-based FDR estimation

All the FDR values calculated in this thesis is based on decoy as opposed to the conventional FDR formula which is calculated using FP. To avoid confusion, this FDR value is referred to as *decoy-based FDR* which is widely used in proteomics. Even though there are several decoy-based formulas, we used the Elias and Gygi's equation [69] to calculate FDR as shown below:

$$FDR = \frac{2 \times D}{T + D} \tag{2.3}$$

where D is the number of decoy hits, and T is the number of target hits.

2.8 Data pre-processing

Given that some features had larger scales than others, each feature was normalised to the range [0,1] as follows:

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)} \tag{2.4}$$

where $x \in \mathbb{R}^{1\times m}$ is the feature vector for the dataset containing m instances, $x_i \in \mathbb{R}$ is the value of the feature for instance i, and z_i is the normalised value of x_i.

2.9 Statistical comparison of the classifiers

When comparing the performance of one or more classifiers, we performed a 10-fold training-testing as shown in **Figure 2.3**. Each training and testing fold is a randomly chosen stratified portion of the training and testing dataset that is non-overlapping (disjoint) from other folds. (Stratified indicates that the proportion of the positive and negative samples are similar in all the folds).

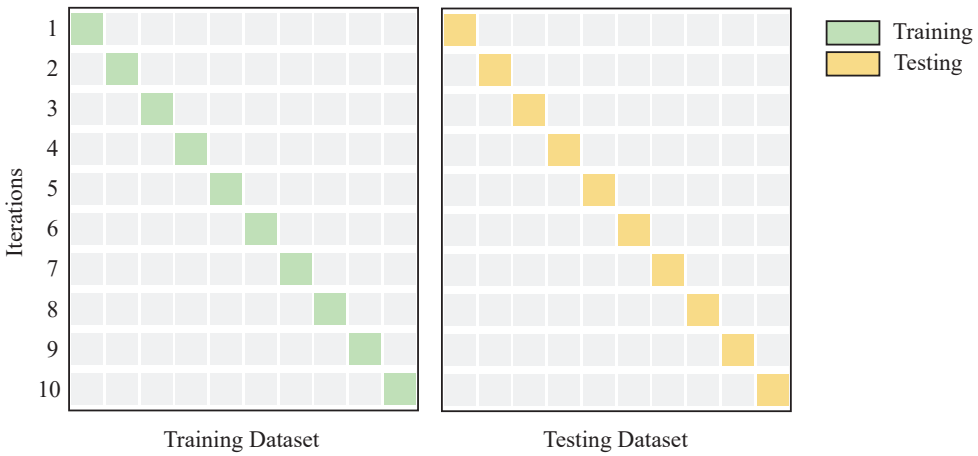

Figure 2.3: Training-testing scheme used to compare the machine learning classifiers.

The above training and testing scheme was used to ensure independence of the samples which is necessary for statistical tests. However, there are still limitation in ensuring that all the samples are statistically independent; that

is, each fold comes from the same experiment or tissue pixel which further violates the independence assumption. In this thesis, however, we assume independence of the samples. On the other hand, such scheme accounts for smaller training and testing sets that lower the performance of the learning algorithms especially those that require large training datasets such as neural networks. However, we have accepted this trade-off given that at worst, we have underestimated the performance of some classifiers rather than overestimating them.

The performance measures between two classifiers were compared using the nonparametric alternative of t-test: Wilcoxon's test [70] implemented in Pingouin [71] package version 0.3.11.

The performance measures between more than two classifiers were compared using the nonparametric alternative of one-way repeated-measure ANOVA: Friedman's test [72] which was implemented in SciPy [73] package version 1.6.2. The statistical significance level for both tests are achieved with p-value < 0.05.

Chapter 3

Protein identification using local and spatial features

The overall goal of this chapter is to explore how the addition of spatial spectral information can affect protein identification in MSI. Herein, a supervised learning model called Spatially-Aware Protein Identification Algorithm (SAPID) is introduced. SAPID integrates both spatial and local information acquired by MSI to assess protein identification at a given tissue location. This approach is based on the idea that given some evidence that a protein P is present at a location S, evidence for protein P in the neighbourhood of S increases the confidence of P's identification at S.

First, I trained four different supervised learning algorithms on the MSI dataset and compared their performance. Furthermore, three different combinations of feature sets (only spatial information, only local information, and both local and spatial information) were used to assess their effect on protein prediction. The performance metrics such as recall, specificity, precision, and ROCAUC were used to compare different combinations of learning algorithms and feature sets. Moreover, total number of proteins detected at $<1\%$ decoy-based FDR was another metric that was compared between classifiers and feature sets. The classifier that had an overall better performance was used in the second part of the analysis where SAPID was benchmarked against the other conventional algorithms on the downsampled datasets.

Since the downsampled datasets mimic MSI analysis with fewer mass spec-

tra per pixel, protein identifications in these datasets can reveal how the spatial features can help protein identification in MSI experiments, where less material is available and protein identification is more challenging. Therefore, we benchmarked SAPID against ProteinProphet and Percolator and compared the total number of proteins they can identify at <1% decoy-based FDR. The optimal outcome is when SAPID identifies more proteins compared to the other two algorithms at the same decoy-based FDR. The above milestones can be summarized as follows:

Protein identification in the original MSI dataset: We built a supervised machine learning model, SAPID, that could use both local and spatial spectral information to identify proteins in MSI datasets, and then benchmarked the algorithm against two state-of-art protein identification algorithms: ProteinProphet, and Percolator.

Protein identification in downsampled MSI dataset: The machine learning model from the previous milestone was trained and tested on the downsampled MSI datasets that were generated by downsampling the original dataset. The algorithms were then benchmarked against ProteinProphet and Percolator on the downsampled dataset.

Statement of author contribution

Soroush Shahryari Fard, Theodore J. Perkins, and Mathieu Lavallée-Adam conceived the study. Soroush Shahryari Fard, Theodore J. Perkins, and Mathieu Lavallée-Adam designed the algorithm. Soroush Shahryari Fard implemented the algorithm. Soroush Shahryari Fard analyzed the data, prepared the figures, and wrote this chapter under the supervision of Theodore J. Perkins, and Mathieu Lavallée-Adam.

3.1 Data visualization

Visualizing the dataset and the relationship between the features is often the first step in machine learning research as it gives important clues about the

instances in the dataset. I first performed hierarchical clustering to explore how similar or different are the features from the proteins that are correctly identified compared to the incorrectly identified proteins. Moreover, the hierarchical clustering analysis can reveal how closely the spatial and local features can be clustered together. I also created a correlation matrix of the features to further showcase the collinearity among the features which could explain the result of the downstream machine learning analysis.

In order to visualise the features and instances in the dataset, a hierarchical clustering with average linkage and Euclidean distance was performed on 20% of randomly selected instances, equivalent to 47,552, in the training dataset. The reason for only choosing 20% of the dataset was simply because of the limited memory on my computer. The Seaborn package [74] version 0.11.1, which wraps the Scipy [73] package's hierarchical clustering algorithm, was used to produce the clustering heatmap. Finally, the Pearson's correlation heatmap for the dataset was generated using pandas [75] version 1.1.4.

The training set contained 204,998 negative and 32,760 positive examples, and the testing set contained 219,014 negative and 42,238 positive examples. Therefore, the class imbalance for the training and testing sets were 1:6 and 1:5, respectively.

The hierarchical clustering results in **Figure 3.1** show a clear clustering of the positive and negative classes; that is, the positive and negative instances are separated in Euclidean space. Therefore, it is expected that the machine learning algorithms that have linear decision boundaries can have a good classification performance on the dataset. The top lanes that are mainly associated with the correctly identified proteins generally show higher value for both spatial and POI features compared to the other lanes. Furthermore, clustering of the features show that some of the spatial features are clustered more closely with the POI features than with other spatial features. For instance, the spatial features *top-xCorr-2* and *top-deltaCN-2* are more closely clustered with *mean-deltaCN-POI* and *mean-deltaCN-5-POI* than with other spatial features such as *top-SpecCount-1* or *num-surrounding-pixels-with-PSM*.

Since mass spectra with higher deltaCN are generally indicative of good protein identifications at a given POI and more good quality identifications are expected to have some neighbouring spectral information associated with them, the close clustering between POI and spatial features can be explained.

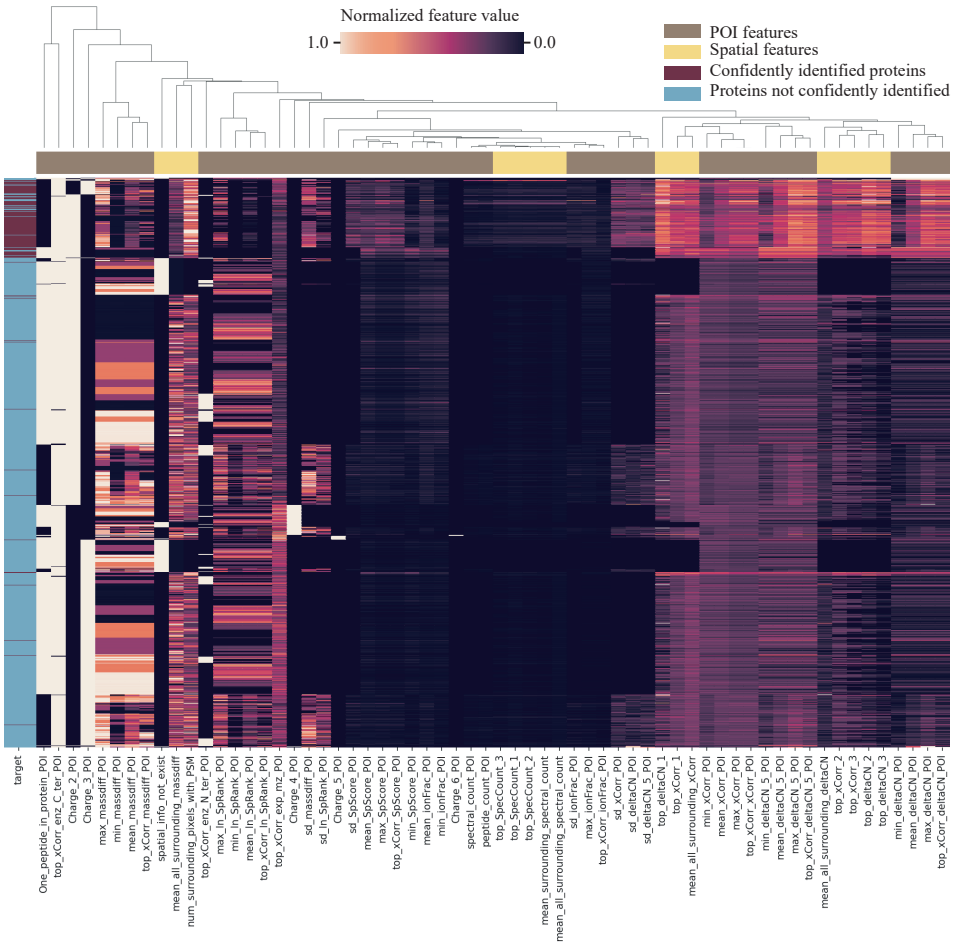

Figure 3.1: Hierarchical clustering heatmap of 20% of the training dataset. Average linkage with Euclidean distance was used to cluster the samples. Target column shows the label of the samples, and the top row shows whether the features are spatial or local. Length of the branches in the dendrogram is proportional to the distance between each feature.

The correlation matrix in **Figure 3.2** shows a strong positive correlation between some features which is in agreement with the observation made from the hierarchical clustering. For example, the features *top-SpecCount-1*, *top-*

SpecCount-2, and *top-SpecCount-3* are positively correlated with *mean-all-surrounding-spectral-count* since the latter is intentionally derived from the former features. This correlation is expected to hinder the performance of classifiers such as Gaussian NB which assume independence of the features. Furthermore, the interpretability of the models such as the weights on the logistic regression model will not be informative given the correlation among features.

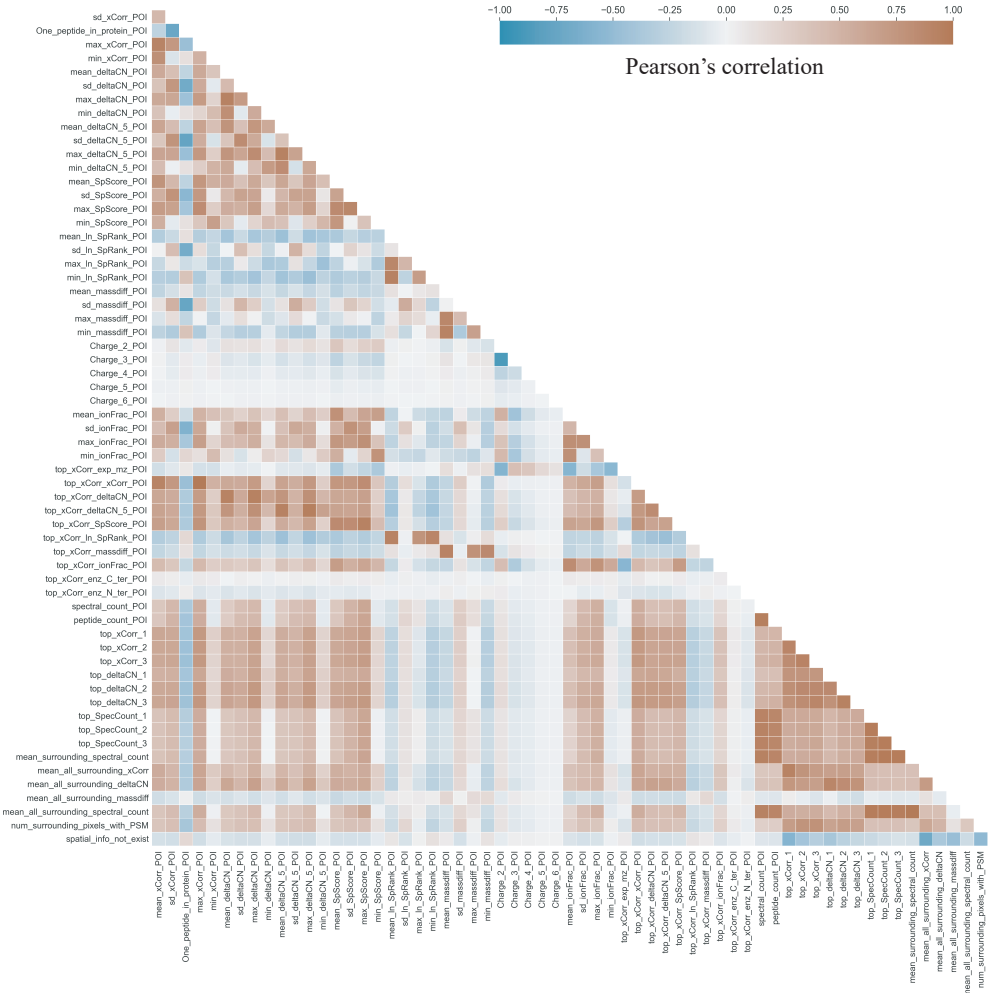

Figure 3.2: The correlation matrix of the features in the training dataset. The features are normalised, and their Pearson's correlation are indicated with the shades of blue (strong negative correlation) and brown (strong positive correlation). The legend indicates the value of Pearson's correlation.

3.2 Machine learning analysis on the full dataset

The main objectives of this section are two folds: explore which learning algorithm could perform the best protein identification task and assessing the benefit of integrating spatial features for protein identification. Thus, different combinations of four machine learning algorithms (LDA, Gaussian NB, decision tree, and logistic regression) and three features sets (POI features only, Spatial features only, and All features) were applied on the dataset using the 10-fold training and testing scheme explained in section 2.9. Scikit-Learn package [76] version 0.23.2 was used to train and test the machine learning algorithms, and hyperparameter tuning was performed using grid search with the search space and k-fold-CV shown in **Table 3.1**. The parameters that maximized the mean of F1 measure across the folds were selected for testing. For each training-testing fold, six metrics were used to compare different combinations of classifiers and feature sets (recall, specificity, precision, F1 measure, ROCAUC, and PRAUC) using the statistical tests explained in section 2.9.

Table 3.1: Summary of grid search space used for hyperparameter tuning of the machine learning models

Algorithm	**Stratified *k*-fold validation**	**Grid search space**
Linear Discriminant Analysis	5	tol: 1.0e-5, 1.0e-4, 1.0e-3, 1.0e-2
Gaussian Naïve Bayes	5	var-smoothing: 1e-8, 1e-9, 1e-10
Decision Tree	5	criterion: gini, entropy max-depth: 1, 2, 3, 4, 5, None min-sample-split: 2, 4, 6, 8 min-sample-leaf: 1, 2, 3, 5 class-weight: None, Balanced
Logistic Regression	5	penalty: L2, Elastic Net C: 0.01, 0.1, 1.0, 10 l1-ratio: 0.25, 0.5, 0.75 class-weight: None, Balanced

Furthermore, the number of correctly identified proteins at <1% decoy-based FDR was also used to compare the classifiers. However, this metric was only possible for classifiers that could obtain <1% decoy-based FDR. Finally, logistic regression was chosen as the optimal algorithm for SAPID for this classification task, and the number of identified proteins at different FDR thresholds were plotted for ProteinProphet, Percolator, and SAPID with different feature sets (All features, Spatial features only, and POI features only) to benchmark their protein identification performance.

The performance metrics of the four classifiers with each of the three feature types are depicted in **Figure 3.3**, which show that overall, the classifiers were similarly able to distinguish the correctly and incorrectly identified proteins across all the feature sets. In particular, Gaussian NB's performance metrics are lower than the other classifiers which could be because of the independence assumption that Gaussian NB requires, but that is not satisfied in the dataset. The relatively good and similar performances of the other classifiers can be due to the good separation of the two classes in Euclidean space as explained before. Thus, the classification task was not very challenging, and classifiers with linear decision boundary performed well on it. For instance, the average recall of 0.90 indicates that the classifiers were able to correctly classify 90% of the positive protein instances, and specificity of 0.97 indicates that the classifiers are able to correctly identify 97% of the negative protein instances. Moreover, precision of 0.90 indicates that 90% of the positive predictions are correct. Furthermore, F1 score of around 0.9 indicates that despite the class imbalance of around 1:5, the classifiers are able to achieve a good overall recall and precision.

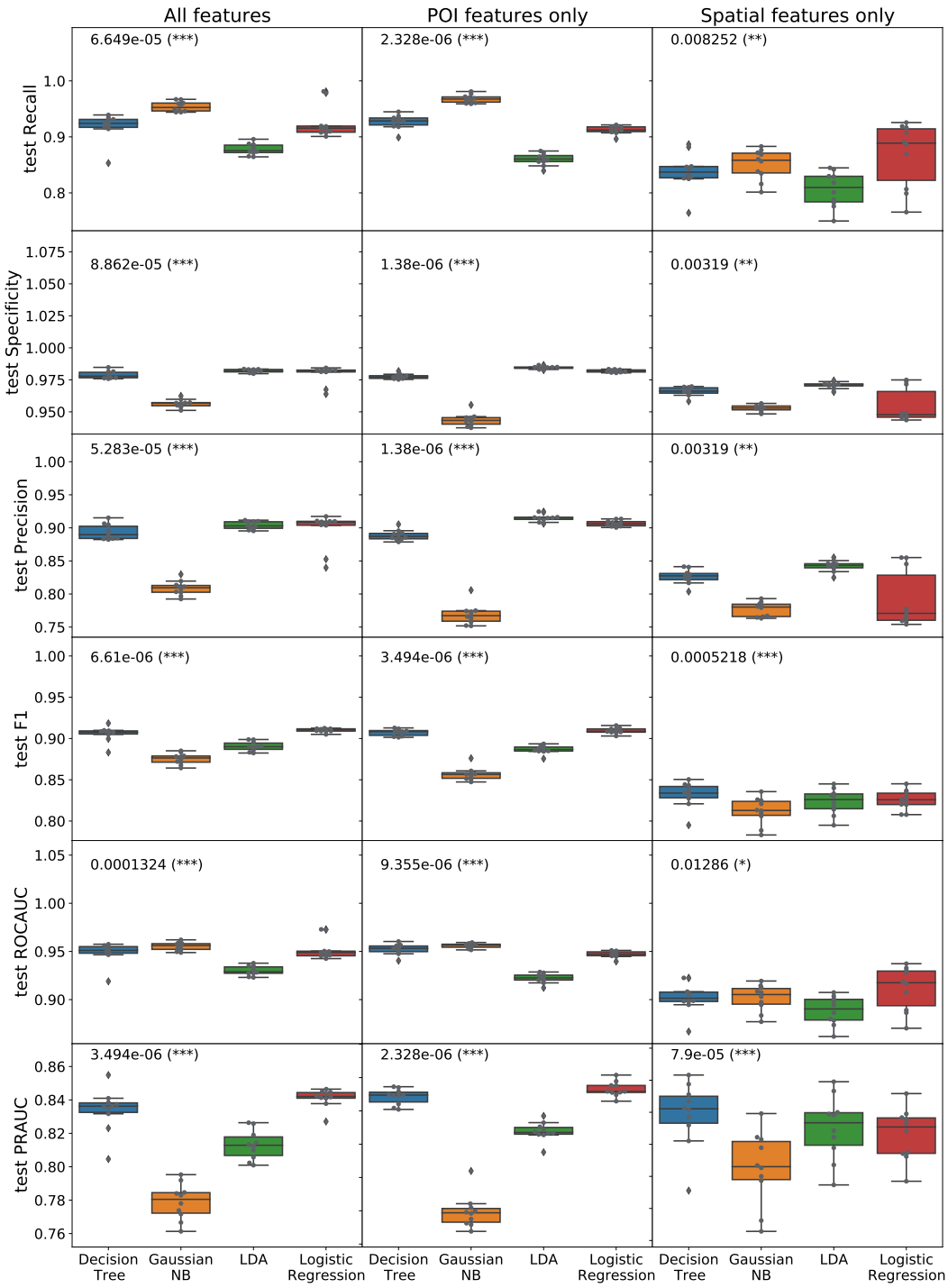

Figure 3.3: Comparing the performance of different classifiers based on different metrics and feature sets on distinguishing correct and incorrect protein matches. Non-parametric Friedman test was performed to assess statistical significance of metric differences. N=10 testing folds. p-values are also indicated by stars, *p<0.05, **p<0.01, ***p<0.001.

The ROCAUC and PRAUC are shown in **Figure 3.3** which are similar for all classifiers except for the Gaussian NB. Furthermore, the curves in **Figure 3.4** show a typical behaviour of a good learning algorithm except for the decision tree with an unstable PR curve. That is, the decision tree algorithm overfits the training sets and fluctuates depending on the training set it is trained on which is not a suitable behaviour.

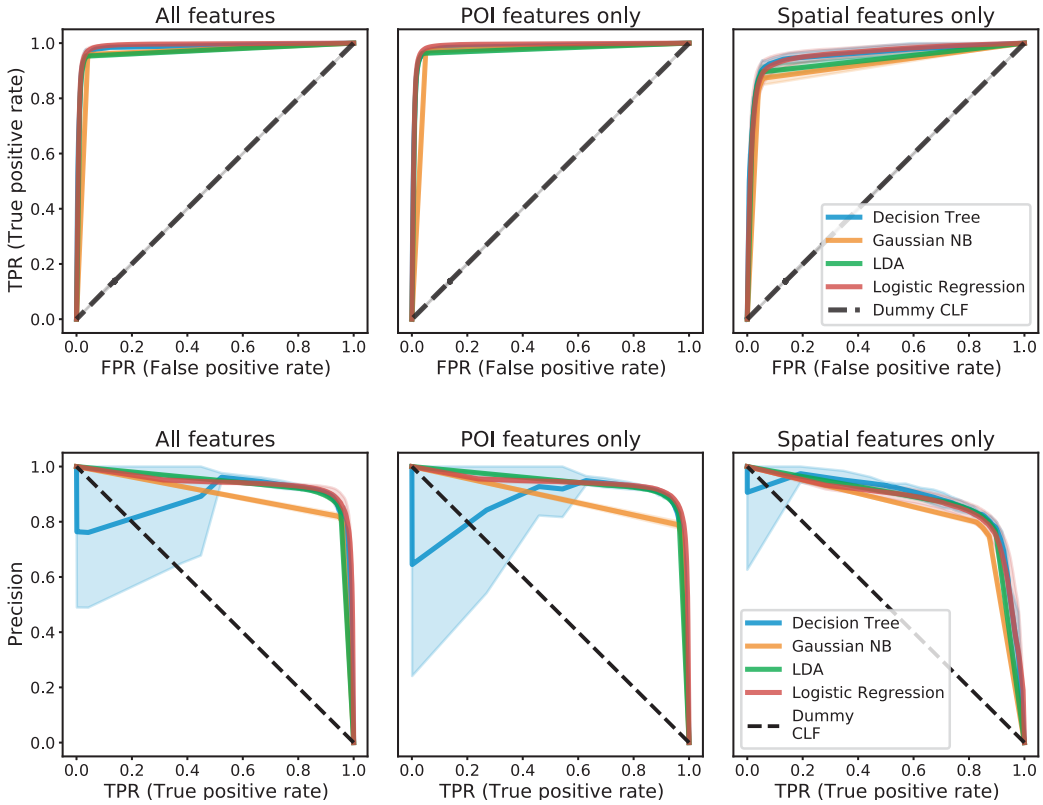

Figure 3.4: ROC and PR curves of different classifiers with different feature sets. Top panel shows the ROC curves; Bottom panel shows the PR curves of the classifiers. The shaded regions show ± standard deviation of the 10 different testing folds.

The total number of protein identifications at <1% decoy-based FDR deemed correct by the logistic regression and LDA classifiers are shown in **Figure 3.5**. Note that Gaussian NB and decision tree could not achieve protein identification at decoy-based FDR <1% and were eliminated from the figure. The statistical tests show that logistic regression identifies more proteins at <1% decoy-based FDR compared to LDA when POI features are used

or when spatial features are used. However, no difference is observed when all features are used. Even though **Figure 3.3** shows that there is little difference between LDA and logistic regression when looking at their performance metrics, **Figure 3.5** shows that logistic regression detects more proteins than LDA at <1% decoy-based FDR, and this difference is statistically significant when either local or spatial features are used. The advantage of the logistic regression can be due to its more non-parametric characteristics; that is, contrary to LDA, logistic regression does not estimate the whole joint distribution.

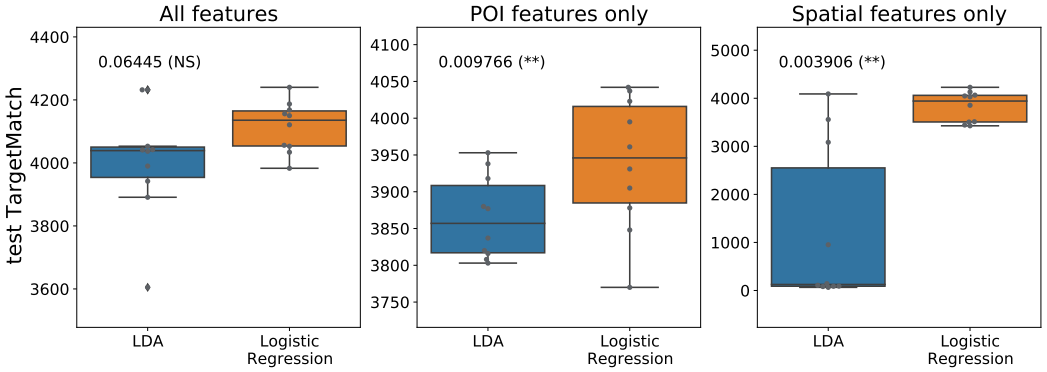

Figure 3.5: Comparing the number of proteins identified at a decoy-based FDR <1% for different feature sets with LDA and logistic regression Non-parametric Wilcoxon test was performed to assess statistical significance. N=10 testing fold. p-values are also indicated by stars, *p<0.05, **p<0.01, ***p<0.001. NS indicates p>0.05.

Figure 3.6 shows the performance metrics of the classifiers when different feature sets are used which illustrates that, overall, the classification of the protein instances is not improved with the addition of spatial information with respect to the metrics measured in the figure. Gaussian NB is the only classifier which shows an improvement in its recall, precision, F1 measure, and PRAUC with addition of spatial features compared to only when POI features are used, but the other algorithms do not show any improvement with the addition of the spatial information. Such behaviour could be because of the correlated features in the dataset; that is, when spatial features are absent, the POI features that are correlated with the spatial features can convey the same information to the learning algorithms. Interestingly, when only spatial features are used, the

classifiers perform reasonably well with average recall, specificity, precision, F1 measure, ROCAUC, and PRAUC of 0.80, 0.90, 0.80, 0.85, 0.90, and 0.70, respectively. Such observation shows that the neighbouring features of a POI alone have the potential to be used for protein identification.

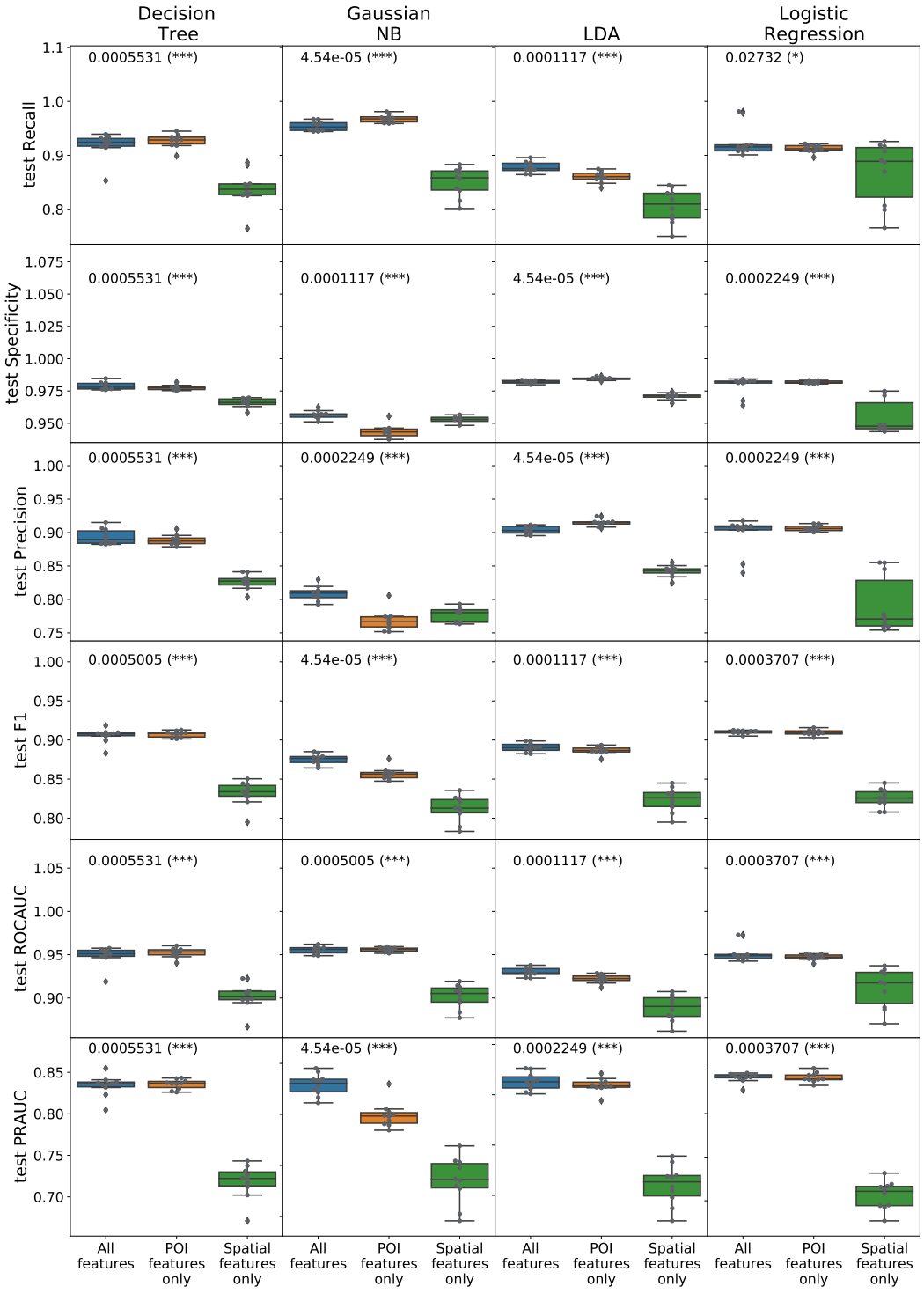

Figure 3.6: Comparing different metrics and feature sets for different classifiers on distinguishing correct and incorrect protein matches Non-parametric Friedman test was performed to assess statistical significance. N=10 testing folds. p-values are also indicated by stars, *p<0.05, **p<0.01, ***p<0.001.

Figure 3.7 shows the number of protein identifications at a decoy-based FDR <1% for logistic regression and LDA classifiers when different feature sets are used. The statistical tests show that the performance of the classifiers did not stay constant across the three features sets. For LDA, the number of correct protein identifications fluctuates a lot across the folds when spatial features are used; however, much less variation is observed in logistic regression which can be due to its less parametric nature as explained before. Moreover, addition of the spatial information did not improve number of proteins detected in LDA, but it increased that for logistic regression.

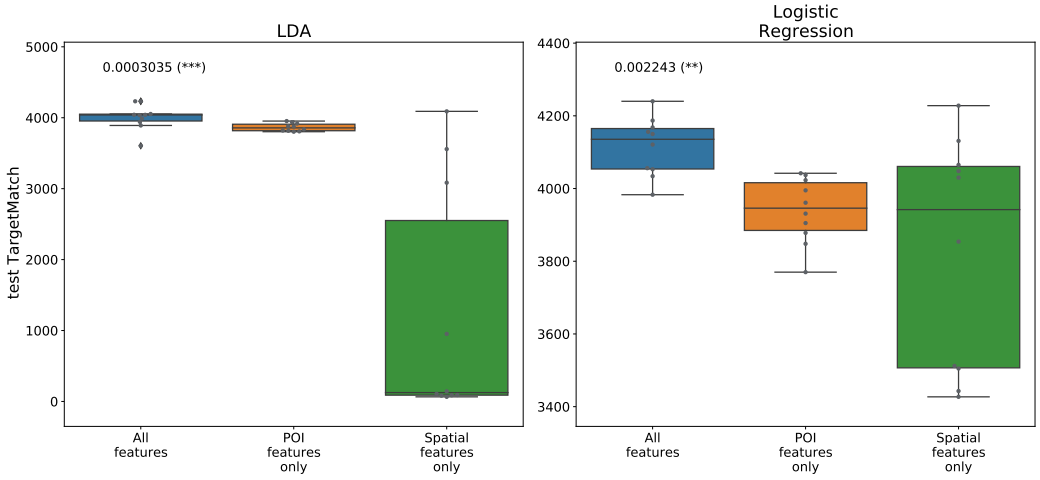

Figure 3.7: Comparing the number of protein identifications identified at decoy-based FDR <1% for LDA and logistic regression with different feature sets. Non-parametric Friedman test was performed to assess statistical significance. N=10 testing fold. p-values are also indicated by stars, $^{**}p<0.01$, $^{***}p<0.001$.

When looking at the number of proteins that are correctly identified at different decoy-based FDRs, SAPID with all features performed consistently better than ProteinProphet and Percolator as shown in **Figure 3.8**. For instance, at a <1% decoy-based FDR, SAPID with all features identifies 40,922 proteins, while ProteinProphet and Percolator identify 37,202 and 34,158 proteins, respectively. It is important to note that these values do not indicate unique proteins; instead, they indicate unique proteins at a given location/pixel. More specifically, SAPID with all features on average identified 1705 ± 118 unique

proteins per pixel, whereas ProteinProphet identified 1550 ± 114 unique proteins per pixel on the testing dataset at a decoy-based FDR <1%. Furthermore, the number of unique proteins that are only detected by either SAPID (All features) or ProteinProphet are 1484 and 843, respectively. Interestingly, SAPID with only spatial features had superior performance compared to ProteinProphet and Percolator and outperformed all the other SAPID's feature combinations at a decoy-based FDR >1.5%. Since more proteins are identified when both spatial and POI features are used compared to when only POI features are used, we can conclude that spatial features improve protein identifications.

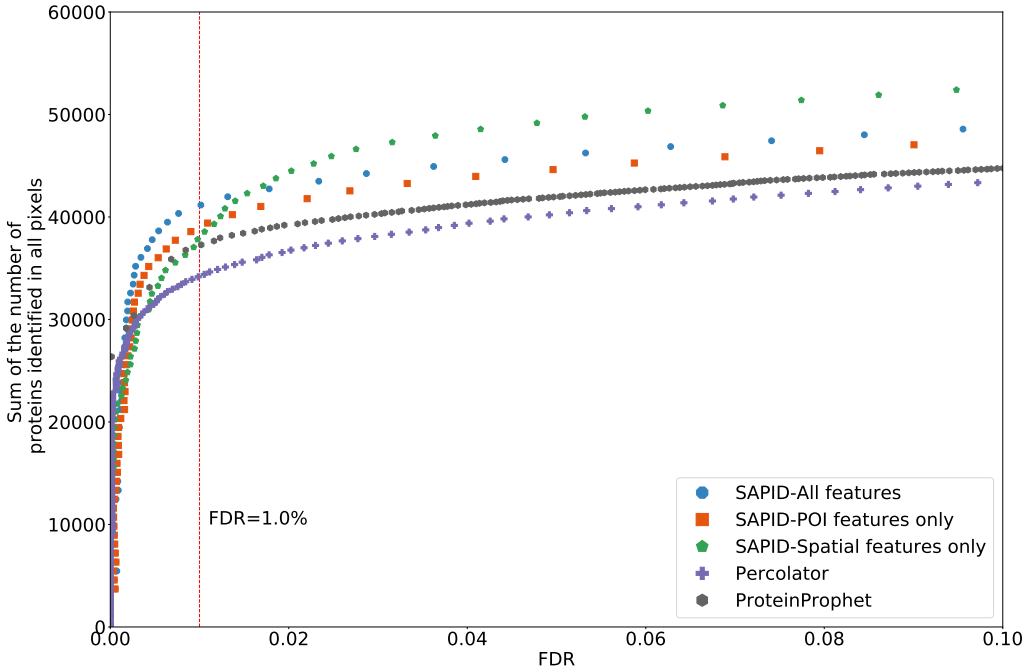

Figure 3.8: Sum of the total number of protein instances identified in the testing dataset with SAPID, Percolator, and ProteinProphet at different decoy-based FDR thresholds. Different combinations of features were used to train and test the SAPID algorithm which is indicated in the legend. The red line indicates 1% FDR.

This conclusion may seem contradictory to the previously presented results indicating that there is no benefit in adding spatial features with POI features. However, it is important to note that the previous metrics were calculated

based on the performance of the classifiers in identifying correctly identified proteins from the labels that were assigned to each instance by ProteinProphet. Although we generally expect them to be correct, this is not guaranteed. In particular, FN are possible with the limited data available in each pixel. Thus, failure for spatial features to improve classification performance may have more to do with the limitations of our instance labels. On the other hand, the decoy-based FDR results shown in **Figure 3.8** is based on the ratio of decoy identifications and non-decoy identifications as explained in 2.7. Although the decoy hits can be considered as FP, the non-decoy identifications could be biologically correct identifications. Therefore, decoy-based FDR calculation performed here cannot be compared to any of the label-based metrics reported previously, as it is like comparing apples to oranges. In short, decoy-based FDR in the context of proteomics is not equivalent to the label-based definition of FDR (or equivalently, $1 - precision$).

3.3 Machine learning on the downsampled datasets

When performing MSI analysis, one of the choices is the pixel size. Smaller pixels mean a potentially-finer understanding of tissue architecture at the molecular level. Because pixels are physically closer, it may also mean a greater reason to believe that proteins in a neighbouring pixel are also found in the POI. On the other hand, making smaller pixels probably means fewer spectra from each pixel, since less peptides will be ionized and hence cause more challenge in identifying the proteins. We cannot directly simulate spatially finer data, but we can downsample the mass spectra to simulate that there are less data per pixel that would be expected and see if SAPID can have benefits in that context. Furthermore, varying tissue quality may also cause less spectra to be acquired at a given pixel.

Therefore, the downsampled training and testing datasets with fewer number of spectra were created as explained in section 2.3. Again, downsampling

should not be confused with data sampling performed in machine learning research to create balanced datasets; that is, the downsampling ratio 0.5 (50%) reduces the number of mass spectra by 50%, but it does not necessarily change the number of training or testing instances by 50%. For each downsampled training dataset, a SAPID logistic regression model was trained and tuned as explained in the previous section and tested on the corresponding downsampled testing datasets. Subsequently, Percolator and ProteinProphet were also tested on the downsampled testing datasets to compare their performances against those of SAPID.

The number of positive and negative instances and their ratio in the downsampled datasets are shown in **Table 3.2**, which shows a small imbalance across all the downsampled datasets. As the downsampling ratio increases, the class balance decreases, which means the datasets with higher downsampling ratio can be more challenging for classification. The highest and lowest class imbalance occur at downsampling ratios 0.1, and 0.9, which correspond to class imbalance ratio of 1:5 and 1:3, respectively.

Table 3.2: Summary of the number of positive and negative instances in the downsampled datasets generated from the original testing dataset.

Downsampling ratio	Number of positive instances	Number of negative instances	Positive:Negative ratio
0.1	40,646	203,615	1 : 5.01
0.2	38,789	187,427	1 : 4.83
0.3	36,721	170,171	1 : 4.63
0.4	34,559	151,488	1 : 4.38
0.5	31,931	132,172	1 : 4.14
0.6	28,769	110,423	1 : 3.84
0.7	24,795	87,346	1 : 3.52
0.8	19,695	62,213	1 : 3.16
0.9	12,564	33,804	1 : 2.69

As explained before, we determined the gold standard protein identifications in the original (non-downsampled) training and testing datasets using ProteinProphet (Equation 2.2). Furthermore, we applied ProteinProphet on the downsampled datasets at different downsampling ratios, which allowed us to compute the fraction of proteins that ProteinProphet is able to cor-

rectly identify at different downsampling ratios. At the same time, we trained SAPID on the downsampled spectra datasets using the downsampled ProteinProphet results as labels since that is all that would be available in a situation where there are limited number of spectra. Then we were able to evaluate what fraction of proteins SAPID can identify in the downsampled datasets in comparison with the ProteinProphet output from the full dataset.

The total number of correct protein identifications across all the downsampled datasets for SAPID, Percolator, and ProteinProphet at <1% decoy-based FDR threshold are shown in **Figure 3.9**. SAPID consistently identifies more proteins than both Percolator and ProteinProphet across all the datasets, and ProteinProphet consistently identifies more proteins than Percolator across all the downsampled datasets. The better performance of SAPID shows that it can more robustly identify proteins in any dataset, though it does not suggest that spatial features are what is causing SAPID to outperform the other two algorithms; a repeated random downsampling would be required to check if the differences are statistically significant.

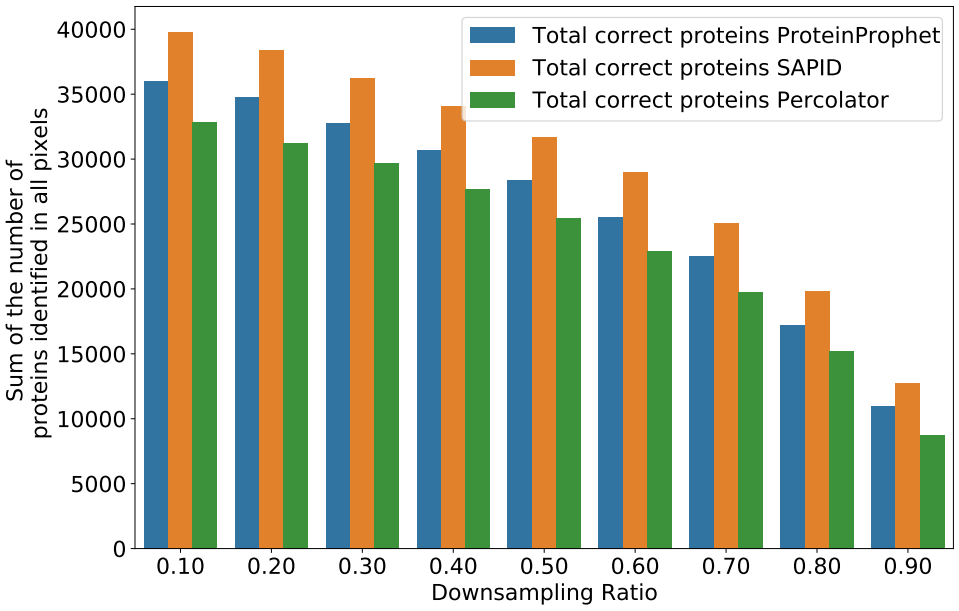

Figure 3.9: Total number of protein instances identified with ProteinProphet, SAPID, and Percolator at different downsampling thresholds at <1% decoy-based FDR threshold. Note that SAPID is a logistic regression model trained with both spatial and POI features

The percent of unique protein identifications in the downsampling datasets that are only identified by either SAPID or ProteinProphet is shown in **Figure 3.10**. Even though SAPID is trained on the results from ProteinProphet, the percent of uniquely identified proteins in SAPID is consistently higher than ProteinProphet in all of the downsampled datasets.

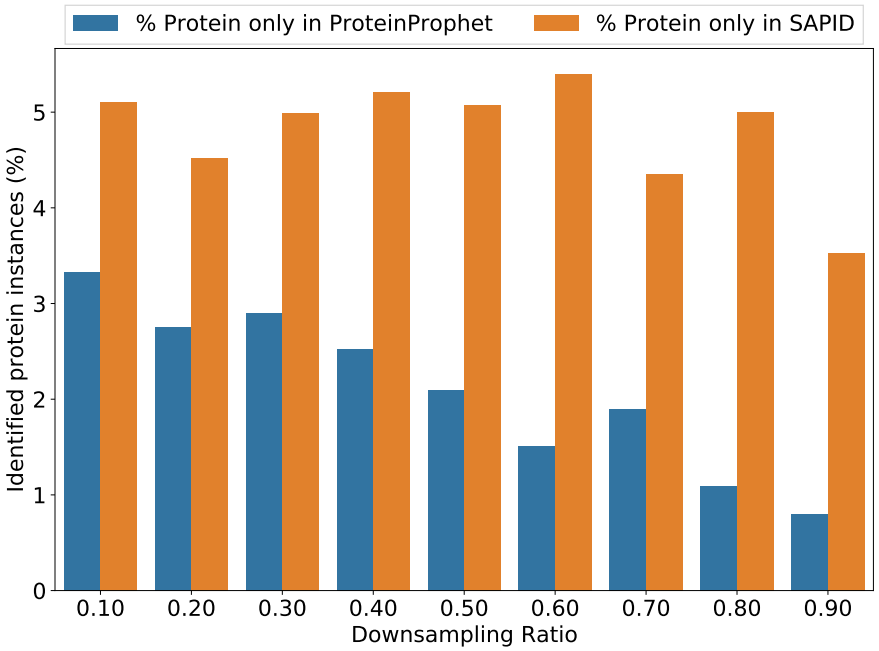

Figure 3.10: Percentage of protein instances correctly identified only by ProteinProphet or SAPID. The percentages are calculated based on the number of unique proteins identified by each algorithm that were present in the original testing dataset. SAPID is a logistic regression model trained with both spatial and POI features.

3.4 Conclusion

In this chapter, we showed that there are correlations among the spatial and local features, and that the correctly and incorrectly identified proteins can be efficiently distinguished with classifiers with linear decision boundaries such as logistic regression and LDA. Furthermore, it was demonstrated that the addition of spatial features does not improve classification scores, but it increases the number of protein identification at a given decoy-based FDR. Furthermore, we were able to show that SAPID can detect more proteins compared to ProteinProphet and Percolator at <1% decoy-based FDR on either the original dataset or on the downsampled datasets when using both spatial and POI features.

Chapter 4

False protein identification rescue in ProteinProphet

ProteinProphet is a popular protein identification confidence assessment algorithm that assigns probability to protein identifications based on peptide assignment information from the database search programs. The score assigned to each protein by ProteinProphet is computed from the probabilities that their associated peptides are correctly identified. Therefore, proteins with more peptide matches and MS/MS spectra associated with them will have higher identification probabilities. Extending a mass spectrometry experiment or using more sample volume to augment MS/MS spectra acquisition can improve protein confidence assessment by ProteinProphet; however, such process can be costly or not feasible due to experimental limitations such as in MSI. Therefore, being able to use spatial information to correct the false protein identifications committed by ProteinProphet can be beneficial without adding any extra cost or experimental processes, and it would further demonstrates the benefit of spatial information for protein identification. Thus, in this chapter, we explored whether SAPID can be used to identify the incorrectly identified or non-identified proteins by ProteinProphet; that is, we wanted to test if SAPID can distinguish the FN and TN protein identifications, and also, distinguish FP and TP protein identifications committed by ProteinProphet. The two classification problems were therefore designed to rescue the mistakes made by ProteinProphet, and their performances were measured using recall,

specificity, precision, PRAUC, and ROCAUC. These metrics illustrates the effectiveness of SAPID in improving protein identification in ProteinProphet. If the mistakes made by ProteinProphet, which only uses the local information, cannot be distinguished by SAPID effectively, we can conclude that there is no benefit in using spatial information for protein identifications in this context. On the other hand, the benefit of spatial information can be demonstrated if SAPID shows a better-than-baseline performance in identifying false identifications or false non-identifications.

We used the downsampling method to create artificial datasets with fewer mass spectra and categorized instances into true negatives and false negatives for the false negatives rescue problem, and true positives and false positives for the false positives rescue problem based on the protein identifications results from the original (non-downsampled) dataset as determined by ProteinProphet.

In order to avoid confusion, hereafter, correct protein identification in the downsampled datasets made by ProteinProphet will be referred to as True Positives, incorrect protein identifications as False Positives, correct lack of identification (non-identification) as True Negatives, and incorrect lack of identification as False Negatives; The capitalised names indicate that the labels are based on the result of ProteinProphet as opposed to SAPID. We can define the two false identification rescue problem as below:

False Negative identification: Different machine learning algorithms were trained and compared in distinguishing ProteinProphet's False Negative and True Negative protein identifications that were made on the downsampled datasets.

False Positive identification: Different machine learning algorithms were trained and compared in distinguishing ProteinProphet's False Positive and True Positive protein identifications that were made on the downsampled datasets.

Statement of author contribution

Soroush Shahryari Fard, Theodore J. Perkins, and Mathieu Lavallée-Adam conceived the study. Soroush Shahryari Fard, Theodore J. Perkins, and Mathieu Lavallée-Adam designed the algorithm. Soroush Shahryari Fard implemented the algorithm. Soroush Shahryari Fard analyzed the data, prepared the figures, and wrote this chapter under the supervision of Theodore J. Perkins, and Mathieu Lavallée-Adam.

4.1 Data visualization

Visualizing the instances and the relationship between them is an important first step to understand the data. However, prior to visualization, the datasets for the False Positive and False Negative rescue problems were created by dividing the instances in the original and downsampled datasets into four sets (**Figure 4.1**): Ω is the set of all the proteins and decoys in the original dataset, P_+ is the set of all the positive protein instances in the original dataset as defined by Equation 2.2, Ω_{ds} is the set of all the proteins and decoys in the downsampled dataset, and P_{+ds} is the set of all the positive instances in the downsampled dataset as defined by Equation 2.2.

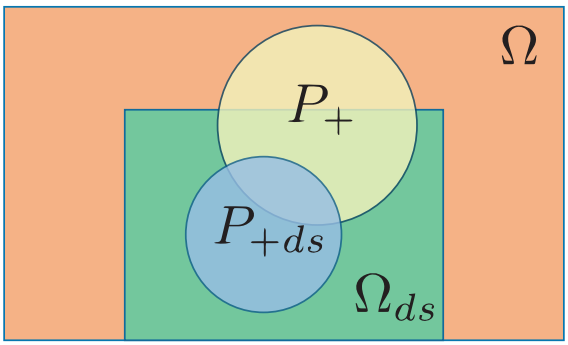

Figure 4.1: Venn diagram of the four sets created from the original and downsampled datasets to define the false protein identification rescue problems. Ω is the set of all the proteins and decoys in the original dataset, P_+ is the set of all the positive protein instances in the original dataset, Ω_ds is the set of all the proteins and decoys in the downsampled dataset, and P_{+ds} is the set of all the positive instances in the downsampled dataset. The diagram is not scaled based on the size of the sets.

Therefore, the new positive and negative instances for each objective can be defined as below:

False Negative identification: In this machine learning problem, the proteins that are falsely identified as negative (or labeled as not-identified in the downsampled dataset) by ProteinProphet are being identified. Therefore, the classification task is to receive all the negative instances identified by ProteinProphet and identify which ones are False Negatives and which ones are True Negatives:

$$\text{Positive (label=1) instances} = \Omega_{ds} \cap P'_{+ds} \cap P_{+} \tag{4.1}$$

$$\text{Negative (label=0) instances} = \Omega_{ds} \cap P'_{+ds} \cap P'_{+} \tag{4.2}$$

False Positive identification: In this machine learning problem, the proteins that are falsely identified as positive (or labeled as identified in the downsampled dataset) by ProteinProphet are being identified. Therefore, the classification task is to receive all the positive instances identified by ProteinProphet and determine which ones are False Positives and which ones are True Positives:

$$\text{Positive (label=1) instances} = P_{+ds} \cap P'_{+} \tag{4.3}$$

$$\text{Negative (label=0) instances} = P_{+ds} \cap P_{+} \tag{4.4}$$

The number of positive and negative instances and their ratio for the False Negative rescue and False Positive rescue problems are shown in **Table 4.1** and **Table 4.2**, respectively. The ratio of the positive to negative instances for both objectives show large class imbalance across all downsampling ratios. Such class distribution will generally cause the learning algorithms to predict mostly the majority class in order to reduce cost or maximize the accuracy. Furthermore, the dataset generated from the smaller downsampling ratios have bigger class imbalance than those generated from the bigger downsampling ratios. For instance, the 0.1 downsampling ratio dataset in the False

Negative rescue problem has class imbalance of 1:219, while the 0.9 downsampling ratio dataset has class imbalance of 1:7.56, and the same trend is observed for the False Positive rescue datasets; since smaller downsampling ratio is equivalent to a dataset with more mass spectra and better protein identifications, the number of False Negative and False Positive identifications will be lower resulting in a more imbalanced dataset which explains the observed trend. Nevertheless, we have chosen the 0.5 downsampling ratio dataset for the downstream analysis because it is a good middle ground. Since the class imbalance for the False Positive rescue problem is more severe than the False Negative rescue problem, the overall performance of the classifiers on the former problem is expected to be lower.

Table 4.1: Summary of the number of positive and negative instances in the downsampled testing datasets for the False Negative rescue problem.

Downsampling ratio	**Number of positive instances**	**Number of negative instances**	**Positive:Negative ratio**
0.1	924	202,691	1 : 219
0.2	1,817	185,610	1 : 102
0.3	2,745	167,426	1 : 61.0
0.4	3,489	147,999	1 : 42.4
0.5	4,163	128,009	1 : 30.7
0.6	4,680	105,743	1 : 22.6
0.7	5,073	82,273	1 : 16.2
0.8	4,952	57,261	1 : 11.6
0.9	3,948	29,856	1 : 7.56

Table 4.2: Summary of the number of positive and negative instances in the downsampled testing datasets for the False Positive rescue problem.

Downsampling ratio	**Number of positive instances**	**Number of negative instances**	**Positive:Negative ratio**
0.1	154	40,492	1 : 263
0.2	274	38,515	1 : 141
0.3	329	36,392	1 : 111
0.4	456	34,103	1 : 74.8
0.5	484	31,447	1 : 65.0
0.6	461	28,308	1 : 61.4
0.7	398	24,397	1 : 61.3
0.8	341	19,354	1 : 56.8
0.9	277	12,287	1 : 44.4

In order to visualize the features and instances for each of the two classification objectives, a hierarchical clustering with average linkage and Euclidean distance was performed on 25% of randomly selected negative examples and all the positive examples in the 0.5 downsampled ratio dataset for each objective. The reason for choosing only 25% of the negative examples was simply because of the limited memory on my computer. Seaborn package [74] version 0.11.1, which wraps the Scipy [73] package's hierarchical clustering algorithm, was used to produce the clustering heatmap.

The hierarchical clustering heatmaps of the 0.5 downsampling ratio datasets for the False Negative and False Positive rescue problems are shown in **Figure 4.2** and **Figure 4.3**, respectively. The heatmaps do not show any clustering pattern between the instances from the same group. Furthermore, similar to the clustering pattern of the features in the previous chapter, some spatial features are more closely clustered with the POI features than with other spatial features as explained in **Chapter 3**. The lack of clustering between the samples of the same class shows that both classification problems are challenging and classifiers with linear decision boundary may perform poorly on them.

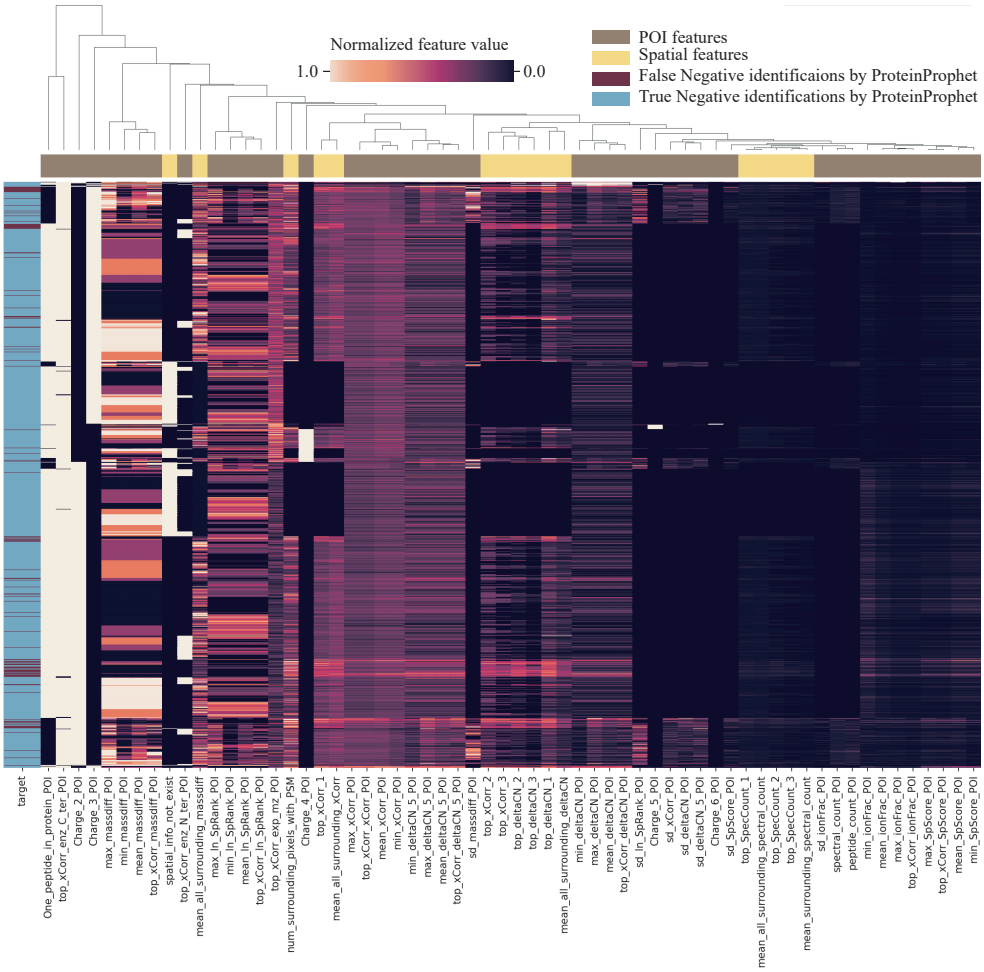

Figure 4.2: Hierarchical clustering heatmap of the False Negative and True Negative instances for the 0.5 downsampling ratio dataset. Average linkage with Euclidean distance was used to cluster all the False Negative samples and 25% of the randomly chosen True Negative samples. Target column shows the label of the instances, and the top row shows whether the feature is spatial or local.

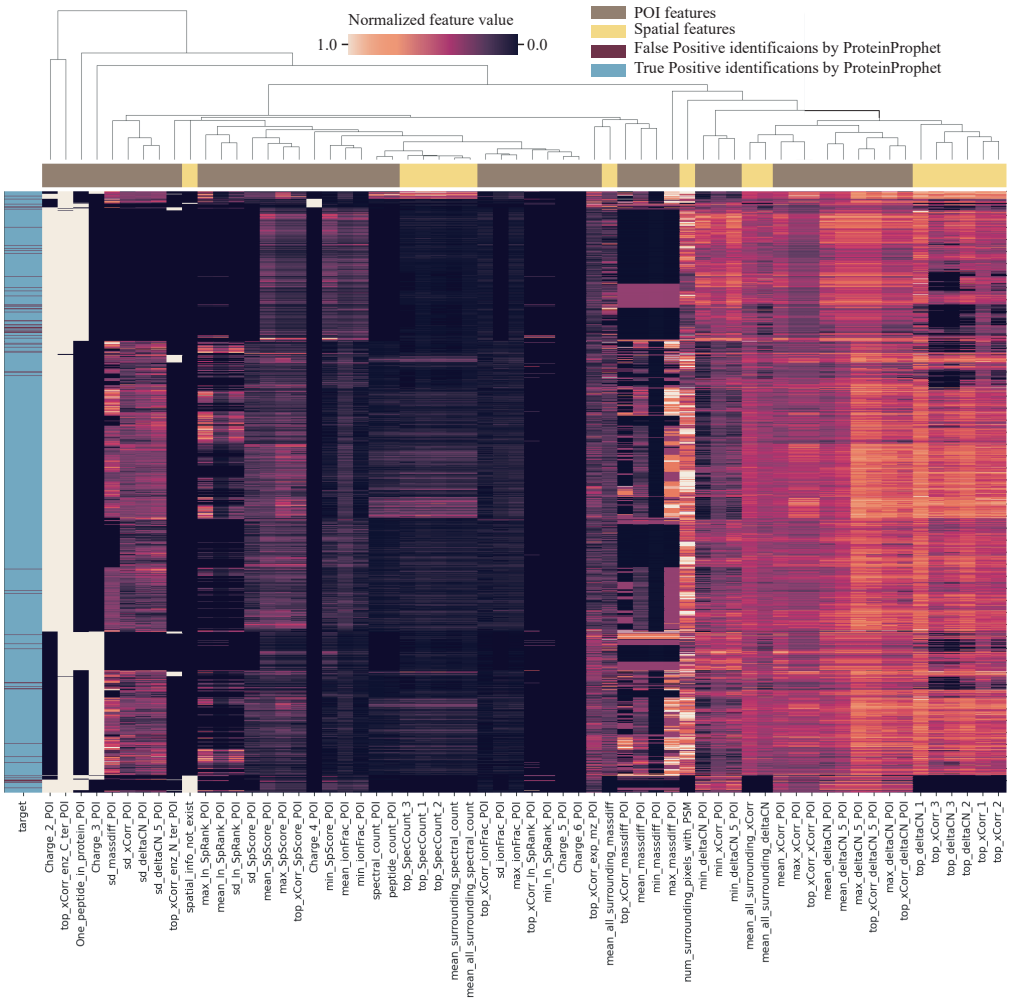

Figure 4.3: Hierarchical clustering heatmap of the False Positive and True Positive instances for the 0.5 downsampling ratio dataset. Average linkage with Euclidean distance was used to cluster all the False Positive samples and 25% of the randomly chosen True Positive samples. Target column shows the label of the instances, and the top row shows whether the feature is spatial or local.

4.2 Machine learning on the downsampled dataset

In the previous section, we visualized the instances and their relationships for the False Negative and False Positive rescue problems. Furthermore, we showed that the datasets for both problems are imbalanced. In this section, for

each problem we trained multiple machine learning algorithms and compared their performance on the 0.5 downsampling ratio dataset. The goal of this section is to show how well SAPID can identify the False Positive and False Negatives protein identifications that were committed by ProteinProphet. To test SAPID for each problem, the 10-fold training and testing scheme explained in section 2.9 was applied on both objectives with all the features and nine machine learning algorithms: logistic regression, LDA, QDA, Gaussian NB, decision tree, random forest, SVM with linear kernel (Linear SVM), SVM with RBF kernel (RBF SVM), and artificial neural network. Furthermore, a Dummy classifier (Dummy CLF) which mimics a random classifier with stratified class output ratio was included in the testing to indicate the baseline performance. Scikit-Learn package [76] version 0.23.2 was used to implement the machine learning algorithms, and hyperparameter tuning was performed using grid search with the search space and k-fold-CV shown in **Table 4.3**. The parameters that maximized the mean of F1 measure across the folds were selected for testing, and the performance of the classifiers were compared using Friedman's test.

The performance metrics of different classifiers for the False Negative rescue problem is shown in **Figure 4.4**. The p-values from the Friedman's test are significant across all the metrics which indicates that there is at least one classifier that statistically significantly perform differently than the others. Overall, LDA, random forest, and Linear SVM have better performance compared to the other classifiers. QDA has lower ROCAUC, PRAUC, F1 measure, precision, and specificity than the other classifiers, and they are similar to the Dummy CLF with random outcome. When the features are collinear, or highly correlated as in our dataset, the determinant of the covariance matrix is very small and close to zero which is highly susceptible to fluctuation in the new testing examples [77]. Furthermore, its inverse cannot be computed. As explained in section 1.5.2, QDA assumes different covariance matrices for each class, while LDA assumes a similar one. Thus, QDA is more likely to be affected by the collinear features in the dataset, and it could be concluded

Table 4.3: Summary of grid search space used for hyperparameter tuning of the machine learning models.

Algorithm	**Stratified *k*-fold validation**	**Grid search space**
LDA	2	tol: 1.0e-5, 1.0e-4, 1.0e-3, 1.0e-2
QDA	2	tol: 1.0e-5, 1.0e-4, 1.0e-3, 1.0e-2
Logistic Regression	2	penalty: L2, Elastic Net L1-ratio: 0.25, 0.5, 0.75 c: 0.01, 0.1, 1.0, 10 class-weight: None, Balanced
Gaussian NB	2	var-smoothing: 1e-8, 1e-9, 1e-10
Decision Tree	2	criterion: gini, entropy max-depth: 1,2,3,4,5, None min-samples-split: 2,4,6,8 min-samples-leaf : 1,2,3,5, class-weight: None, Balanced
Linear SVM	2	c: 0.01, 0.1, 1.0, 10 class-weight: None, Balanced
RBF SVM	2	c: 0.01, 0.1, 1.0, 10 class-weight: None, Balanced
Random Forest	2	n-estimator: 50, 100, 200 max-depth: 2, 5, 10 min-sample-split: 2, 5, 10 min-sample-leaf: 2, 5, 10 class-weight: None, Balanced
Artificial Neural Network	2	activation: ReLU hidden_layer_nodes_num: 5, 10 hidden_layer_layers: 3, 4 learning_rate_init: 0.001, 0.01, 0.1 alpha: 0.0001, 0.001, 0.01, 1.0, 10

that the multicollinear features in our dataset have caused the low performance of QDA. In fact, the Scikit-Learn gave a warning that the variables are collinear. Logistic regression has an average performance compared to the other algorithms, but it exhibits high variance across the folds which is not ideal. Even though LDA and logistic regression have similar performances, LDA has higher F1 measure and PRAUC which was not expected given that LDA relies on more assumptions compared to logistic regressions.

Random forest's better performance in terms of F1 measure, ROCAUC, and PRAUC compared to the other classifiers can be due to its ability to capture complex class distributions, especially compared to the simpler classifiers with linear decision boundaries. Furthermore, random forest has smaller variance across all the metrics compared to decision tree.

Although one expects to see a better performance from RBF SVM than the Linear SVM given the non-linear nature of the classes, there is little difference between the two models. Given the lengthy training time of RBF SVM, Linear SVM is to be preferred in this case.

Recall and ROCAUC for neural network are lower than the other classifiers, which can be due to the large class imbalance and the small number of positive training examples in the dataset. The implementation of the neural network in Scikit-Learn does not support class weights which can be the reason for its poor performance. Furthermore, the grid-search was performed with only two hidden layer sizes (3 and 4) and two node counts (5 and 10), which is limited; therefore, having more diverse architecture could potentially improve its performance.

On average, classifiers had recall and specificity of 0.75, and 0.90, respectively. These values suggest that SAPID can correctly identify 75% of ProteinProphet's False Negative protein identifications and 90% of ProteinProphet's True Negative protein identifications. Moreover, the average precision of around 0.5 suggest that only 50% of the positive calls are actually TP (or False Negative protein identifications) which is reasonable given the class imbalance of 1:30. Since SAPID can identify three-quarter of the False

Negative protein identifications correctly with sufficient precision and specificity, we can conclude that spatial information is beneficial in rescuing False Negative protein identifications committed by ProteinProphet.

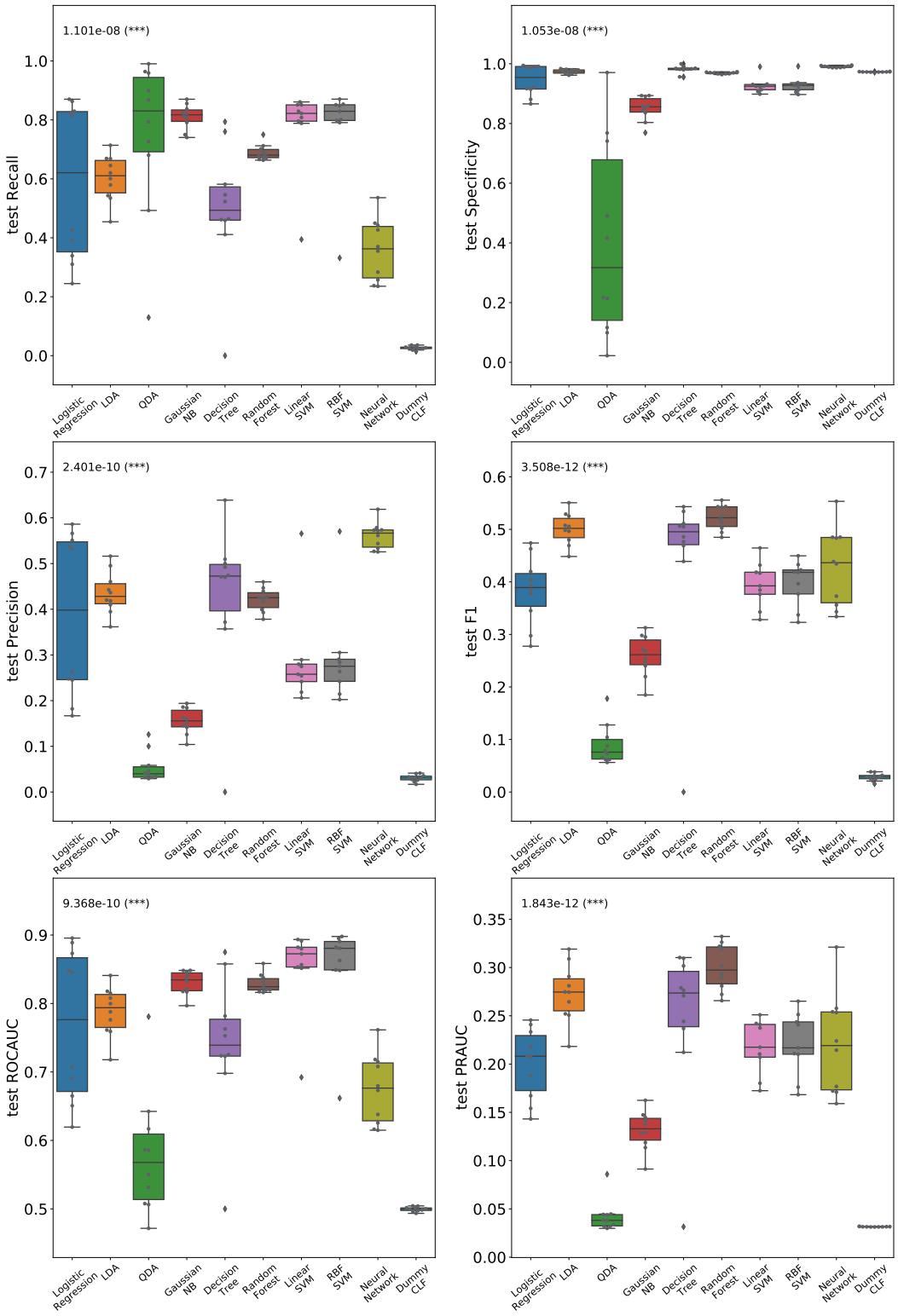

Figure 4.4: Summary of the metrics for 10 different classifiers in distinguishing False Negative and True Negative protein identifications for the 0.5 downsampling ratio dataset. Non-parametric Friedman test was performed to calculate the p-values. N=10 testing folds. p-values are also indicated by stars, $^{***}p<0.001$.

Figure 4.5 shows the performance metrics of the classifiers for the False Positive rescue problem. QDA and neural network show consistently poor performances that are comparable to the Dummy CLF with stratified output. The poor performance of QDA is likely due to the collinear features for which computing covariance matrix is not possible. Again, Scikit-Learn gave a warning indicating the features are collinear. On the other hand, the poor performance of the neural network could be because of the incapability of the Scikit-Learn's implementation to accept weights when training on an imbalanced dataset. The lack of diverse architectures with more options for number of nodes and layers could also have contributed to its poor performance. It is important to note, however, that neural network's poor performance can also be due to the very small (<500) number of positive instances in the dataset which gets further reduced to <50 samples when performing the 10-fold training testing scheme. Compared to the other classifiers, neural networks require more examples for training, and they have poor performance when there are limited number of examples, such as in this case.

Similar to the False Negative rescue problem, logistic regression exhibits high variance across the folds in the False Positive rescue problem. Moreover, it has a comparable performance with LDA, but LDA has smaller variance, which is more ideal.

The PRAUC, F1 measure, and precision of random forest are higher than the other classifiers, though its ROCAUC and recall are lower compared to RBF SVM and Gaussian NB. Thus, random forest results in more FP but less FN compared to the latter two classifiers. Interestingly, decision tree did not perform well and had much worse performances compared to the random forest. In contrast, the performance of random forest and decision tree were very close in the False Negative rescue problem (**Figure 4.4**). However, since the False Negative rescue problem is less imbalanced (1:31) than the False Positive rescue problem (1:65), we expect to see less variation between decision tree and random forest in the former problem.

Similar to the False Negative rescue problem, Linear SVM or RBF SVM

showed comparable performance in the False Positive rescue problem. However, both SVM algorithms are more computationally expensive compared to the other algorithms.

On average, the recall and specificity of classifiers are 0.65, and 0.90, respectively. Thus, SAPID can correctly identify 65% of the False Positive protein identifications and 90% of the True Positive protein identifications. The average precision of 0.15 indicates that 15% of the False Positive protein identifications are actually correct identifications. However, given the extreme class imbalance of the dataset (1:65) and the baseline performance of less than 1%, such performance is acceptable. The average ROCAUC is around 0.70 which is 20% more than the baseline. However, the average PRAUC is around 0.05 which is only 3% more than the baseline. All in all, SAPID and its use of spatial information can be used to distinguish False Positive and True Positive protein identifications committed by ProteinProphet.

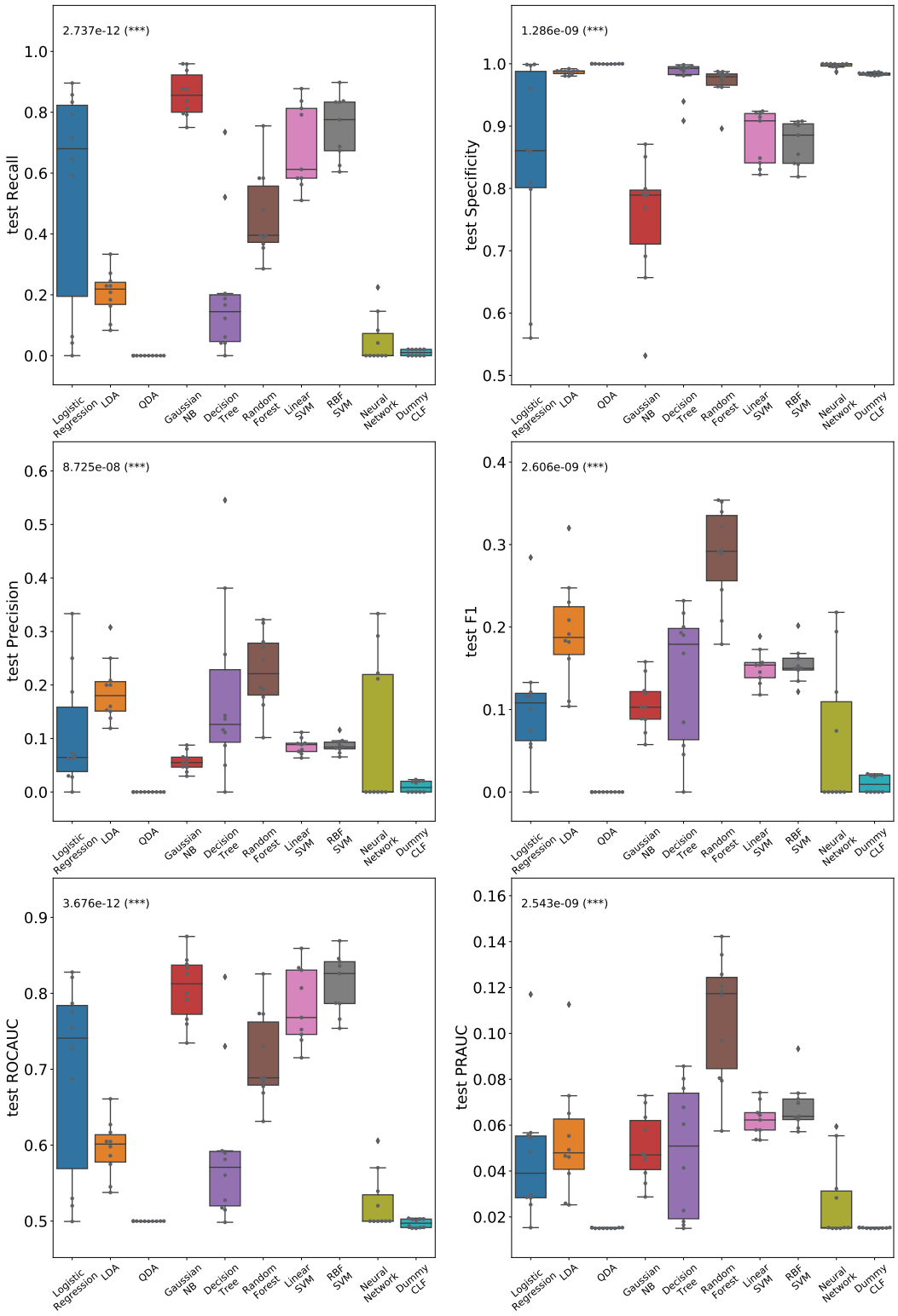

Figure 4.5: Summary of the metrics for 10 different classifiers in distinguishing False Positives and True Positives for the 0.5 down-sampling ratio dataset. Non-parametric Friedman test was performed to calculate the p-values. N=10 testing folds. p-values are also indicated by stars, $^{***}p<0.001$.

4.3 Data sampling to improve classification performance

In the previous section, we compared the performance of 10 different classifiers on both False Negative and False Positive rescue problems. Since there is a large class imbalance in both problems, we sought to investigate how data sampling techniques can affect performances of the classifiers for both problems. Briefly, data sampling seeks to balance the number of examples in each class by eliminating examples from the majority class or increasing examples from the minority class, or a mix of both. Therefore, 11 data sampling methods were used to test if data sampling could improve classification for the imbalanced datasets for both objectives: Over Sample 0.1, Over Sample 0.5, Over Sample 1, Under Sample 0.1, Under Sample 0.5, Under Sample 1, Synthetic Minority Oversampling Technique (SMOTE) 0.1, SMOTE 0.5, SMOTE 1, SMOTE Tomek, and SMOTE Edited Nearest Neighbour (ENN) were used (the number following each name indicates the ratio of minority to majority classes that the algorithms seek to achieve). Note that under sampling should not be confused with the downsampling we performed to create datasets with lower number of mass spectra, so we will capitalize the over sampling and under sampling methods in this thesis to prevent confusion with downsampling. Furthermore, we have only tested data sampling techniques with LDA, random forest, and SVM algorithms for each of the false identification rescue problem since these algorithms have shown an overall better performance compared to the other algorithms we have tested in the previous section.

Briefly, Over Sampling and Under Sampling are random sampling with replacement. SMOTE [78] algorithm is an oversampling technique, which generates new synthetic samples from the minority class by creating new samples on a line passing through at least one of the 5 nearest neighbours of the minority class samples. Since SMOTE may produce noisy data when the two classes cannot be separated well, SMOTE Tomek [79] and SMOTE ENN [80] are two algorithms that perform Under Sampling following the SMOTE over-

sampling. SMOTE Tomek removes the Tomek's links which are the samples from the minority class that are very close to the majority class samples in Euclidean space. On the other hand, SMOTE ENN is based on the Edited Nearest Neighbour algorithm, which identifies and removes noisy synthetic data where the majority of their 3 nearest neighbours have different labels than the samples themselves. Imbalanced-Learn package [81] version 0.8.0 was used for data sampling, while training, hyper-parameter tuning, and testing were performed using the same strategy explained in section 4.2. For each of the 10 training-testing fold, four metrics were used to compare different data sampling techniques (recall, specificity, ROCAUC, PRAUC) using the Friedman's test.

The performance of LDA, random forest, and Linear SVM with different data sampling treatments for the False Negative rescue problem are shown in **Figure 4.6**. The Friedman tests p-values show that at least one of the data sampling techniques statistically significantly affects the performance of the classifiers. As expected, LDA and random forest follow a trend in which the data sampling techniques that generate outputs with more class balance result in a higher recall and lower specificity. For instance, in LDA, Over Sample 0.1 results in lower recall than Over Sample 0.5, and Over Sample 1.0. Similarly, Under Sample 0.1 has lower recall than Under Sample 0.5 or Under Sample 1.0. Such a trend can be explained easily because the positive class is the minority class, balancing the class ratios causes the classifiers to focus more on the positive examples compared to when the original imbalanced dataset is used. As a result, we expect to see an increase in true positives and a decrease in true negatives which, in turn, results in a higher recall and a lower specificity. Linear SVM, however, does not show a similar behaviour under different data sampling techniques. As explained in section 1.5.2, support vectors are the points that influence the position and orientation of the SVM's decision boundary that separates the two classes in feature space. So, as long as the support vectors do not change, adding or removing training examples do not affect SVM's performance, which is what we observe. In fact, SVM is

robust against outliers for this reason. Furthermore, SMOTE did not appear to be more beneficial compared to the random Over Sampling and Under Sampling. SMOTE Tomek and SMOTE ENN resulted in similar performance as SMOTE 1.0 which means that Tomek and ENN were not effective at detecting any outliers that were produced with SMOTE. However, in the future, it could be worth trying the algorithms with different number of nearest neighbours.

Overall, the ROCAUC values for LDA and random forest are improved with the data sampling techniques, especially when the final class ratios are balanced. On the other hand, the PRAUC values show a reverse trend where they are lowered with all the data sampling techniques. In fact, improving ROCAUC does not necessarily improves PRAUC [82]. ROCAUC value is based on the ratio of false positives to the number of true negatives; however, PRAUC compares the ratio of false positives to the true positives. Therefore, the later is affected more by the class imbalance when the positive class is the minority class.

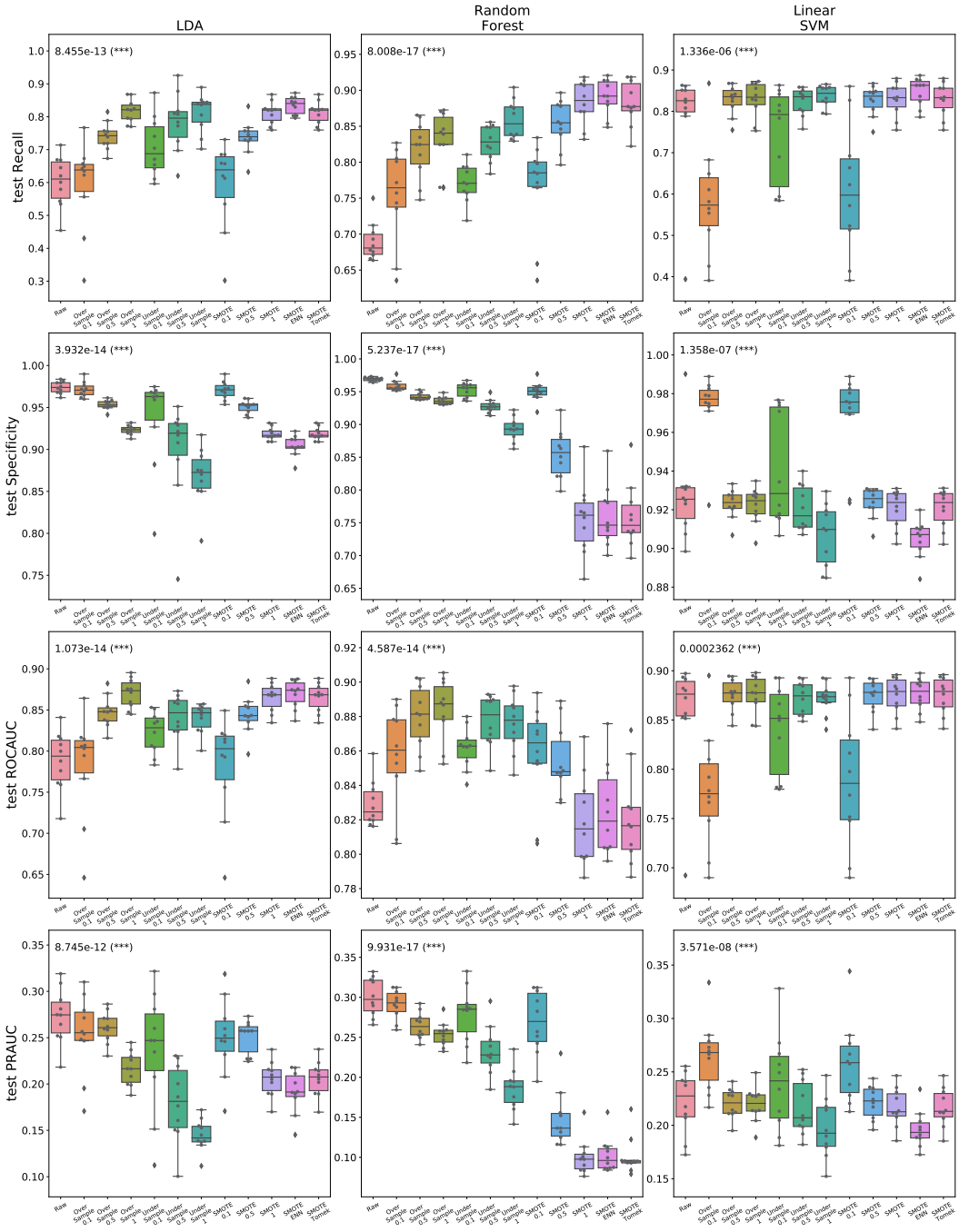

Figure 4.6: Comparing different data sampling approaches on the False Negative rescue classification task with LDA, random forest, and SVM for the 0.5 downsampling ratio dataset. Non-parametric Friedman test was performed to assess statistical significance. N=10 testing folds. p-values are also indicated by stars, $***p<0.001$. The original dataset with no data sampling is indicated with *Raw*.

The performance metrics of the 3 classifiers with different data sampling

treatments for the False Positive rescue problem are shown in **Figure 4.7**. The Friedman test p-values show that at least one of the data sampling techniques statistically significantly affect the performance of the classifiers compared to the other methods. Similar to the False Negative rescue problem, recall is generally increased and specificity decreased with the data sampling techniques that produce more balanced datasets. Furthermore, as explained previously, the RBF SVM showed little variation in performance among different data sampling techniques.

In contrast to the False Negative rescue problem where the ROCAUC and PRAUC values could not be increased at the same time with the data sampling techniques, for this problem we can see that Over Sample 0.1 and Over Sample 0.5 improved both values for LDA and random forest compared to when no data sampling was performed. SMOTE algorithms also resulted in similar performance compared to the other Over Sampling techniques indicating that its data synthesis is not beneficial for this problem. The performance of the classifiers on the datasets produced by SMOTE Tomek and SMOTE ENN were similar to SMOTE 1.0, which indicates that the addition of Tomek and ENN Under Sampling did not improve the result of SMOTE algorithm for our classifiers.

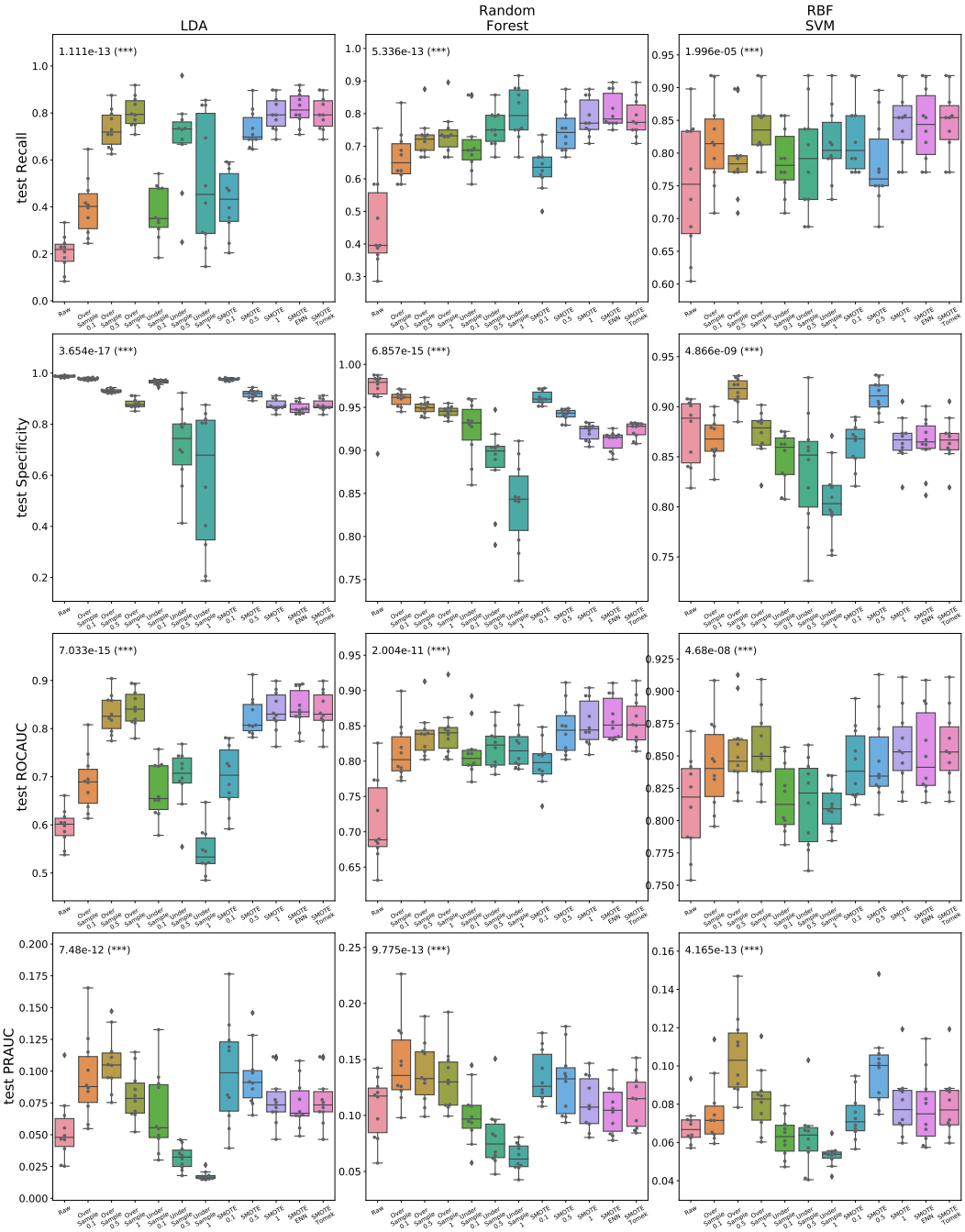

Figure 4.7: Comparing different data sampling approaches on the False Positive rescue classification task with LDA, random forest, and SVM for the 0.5 downsampling ratio dataset. Non-parametric Friedman test was performed to assess statistical significance. N=10 testing folds. p-values are also indicated by stars, $^{***}p<0.001$. The original dataset with no data sampling is indicated with *Raw*.

4.4 Conclusion

In this chapter the possibility of using both spatial and POI features to improve ProteinProphet's protein identification in MSI datasets was explored. We devised a downsampling scheme to create datasets with fewer spectra that could be used to intentionally induce known mistakes in ProteinProphet's identification that can then be assessed. The first objective was to rescue the False Negative identifications by distinguishing the False Negative and True Negative protein identifications, and the second objective was to identify False Positive and True Positive protein identifications. For both objectives, 10 machine learning algorithms were tested. We observed that LDA, random forest, and SVM are the most ideal classifiers for both of the rescue problems and data sampling could improve their performance. Furthermore, the datasampling techniques seem to improve classification performances. Although we did not test the performance of SAPID on the false identification rescue problems with different set of features in this chapter, the result of the previous chapter combined with this chapter suggests that spatial information can be beneficial at improving protein identification in MSI

Chapter 5

General discussion

The central theme of this thesis was to investigate the benefit of incorporating spatial information for protein identification in MSI. To my knowledge, this is the first attempt at exploring this notion to date, and it can pave the way for future research in this area. We have investigated the effect of spatial information with two different research objectives; first, we trained several machine learning classifiers with different sets of features to identify the best performing model and determine the effect of spatial information on protein identification. Then, we benchmarked the machine learning algorithms, a logistic regression-based model called SAPID, that uses both spatial and local information against ProteinProphet and Percolator on downsampled datasets. For the second objective, we explored whether ProteinProphet's protein identifications can be improved by identifying/rescuing False Negative and False Positive identifications using a classifier trained with both spatial and local information. To achieve this, we first used the ProteinProphet's identifications on a 0.5 downsampling ratio dataset and the original dataset to determine the mistakes made by ProteinProphet. Then, we trained different classifiers to identify those mistakes. Overall, both objectives show the added benefit of integrating spatial information with local information for protein identification.

5.1 Protein identification with local and spatial features

One important result of this thesis was to illustrate the relatedness between spatial and POI features. This showed that spatial features are informative when looking at the POI features and possibly implying a biological pattern underlying the tissue. Such biological pattern can be explored and validated in the future. Moreover, we have shown that at a decoy-based FDR <1%, SAPID was able to identify 10% more proteins than ProteinProphet, and 20% more than Percolator, indicating the advantage of using spatial information in the context of protein identification. Furthermore, we showed that SAPID can identify more proteins compared to ProteinProphet and Percolator at a decoy-based FDR <1% on the downsampled datasets containing fewer mass spectra than the original dataset, suggesting the advantage of spatial information in MSI. Interestingly, we also discovered that the spectral features alone could be reasonably useful for protein identification at a given POI. Even though we used statistical tests to compare the classifiers, the difference in protein identification performance between SAPID, ProteinProphet, and Percolator was not evaluated using a statistical approach with randomly conducted repeated experiments due to a lengthy duration of each iteration. However, such experiments should be done in the future to statistically evaluate the result of our study.

5.2 False Positive and False Negative rescue

We have shown that both False Positive and False Negative rescue problems are challenging classification tasks. After training and testing several machine learning algorithms for both the False Negative and the False Positive rescue problems, we were able to achieve an average recall and specificity of 0.75, and 0.90 for the False Negative rescue, and recall and specificity of 0.65, and 0.90 for the False Positive rescue problem. Given the large imbalance in the dataset, such numbers indicate a great advantage of SAPID compared to the state-of-

art ProteinProphet, which only uses POI information for protein inference. Although we showed that SAPID with both spatial and POI features can be more effective at protein identification in the previous objective, it would be interesting to also evaluate how SAPID with either feature sets can perform at protein rescue task. To tackle the very large imbalance in the datasets in this objective, we used some data sampling techniques that improved the performance of some, but not all, of the classifiers. Overall, the results of this objective show that SAPID with both spatial and local information can be effective at identifying the mistakes made by ProteinProphet.

5.3 Implications

MSI is a technique that can produce spatial maps of molecules such as proteins, lipids, or metabolites in complex biological tissues. Currently, MSI data analysis approaches generally do not include the available spatial information and treat each pixel separately in the context of protein identification. However, the result of our study suggests that integrating spatial information in the data analysis pipeline for MSI can improve protein detection. This notion could be extended into other MSI-based omics technologies such as lipidomics and metabolomics to improve their sensitivity without adding extra cost or experimental procedures. Identification of molecules with these technologies is more challenging than the characterization of proteins. The use of spatial information could therefore contribute to tackling this challenge. Consequently, improving data analysis for MSI translates into discovering more biological molecules that can reveal the biological pathways or mechanisms taking place in the biological tissues. A broader implication of our result and the basis of SAPID are their application in other spatial omics technologies, which are not based on mass spectrometry such as in spatial transcriptomics. In contrast to bulk transcriptomics, which can only provide limited information about heterogeneous biological tissues with complex types and functions, spatial transcriptomics allows keeping biological context and cellular localization

into perspective, which is more advantageous in complex diseases. A combination of single-cell RNA sequencing (scRNA-seq) with spatial transcriptomics have gained momentum recently to study gene expression in heterogenous tissues [83]. However, since scRNA-seq has low efficiency and is prone to more noise compared to the bulk approach [84], a similar algorithm to SAPID could be implemented to use spatial information in order to improve scRNA-seq data analysis in the context of spatial transcriptomics. We hope such an algorithm can enhance our understanding of molecules present in complex tissues.

5.4 Biological applications

The overall significance of this thesis was to demonstrate the usefulness of spatial information for protein identification in MSI. Increasing the protein identification sensitivity of mass spectrometry instruments is an important step in advancing science and our understanding of the proteome of biological samples, whether these are cell lines or complex tissue. It can also help shedding light on the biological mechanisms of disease such as cancer. Indeed, whole tumours or tissue biopsies are usually extracted from solid-tumor cancer patients for diagnostic or characterization purposes. These objectives often require the identification of over-expressed or under-expressed proteins and their PTM. Therefore, improving our ability to identify proteins can enhance our diagnostic capabilities. For example, MSI can be particularly useful to identify protein biomarkers in the context of prostate cancer. Indeed, Intraductal Carcinoma (IDC) is a lesion characterized by the growth of tumors in pre-existing prostate ducts that has been associated with a bad prognosis and recurrence of prostate cancer [85], [86]. However, there are currently no biomarkers for IDC. Establishing IDC's proteomic profile could lead to the identification of such biomarkers. However, a major challenge in characterizing the proteome of IDC is that in >75% of the specimens, IDC lesions account for less than 5% of the tumour volume [87], yielding very small amounts of proteins. MSI is an excellent choice for IDC proteomics analysis compared to the conventional

LC–MS experiments, since it can analyze proteins *in situ* in IDC instead of analyzing the prostate tumour sample as a whole, thereby maximizing IDC protein detection. However, the small amount of proteins available in IDC still makes protein identification challenging. Hence, my algorithm, by increasing protein identification in MSI, can significantly improve our ability to identify biomarkers in biological samples such as IDC in prostate cancer.

The intuition for using spatial information is based on the presumption that the local regions in the analyzed tissues usually follow a biological pattern and express a similar set of molecules. With such samples, my algorithm can use spatial information to guide protein identifications, or any other biological molecules for that matter. However, if a tissue is extremely heterogenous or if the MSI instrument has very low resolution and neighbouring pixels capture very different cell types, we expect SAPID to not be as beneficial since the spatial information will not properly inform protein identification. I would expect that state-of-the-art algorithms presented in this thesis for benchmark purposes (ProteinProphet and Percolator) are likely going to be more effective in such a context. Nevertheless, given that no data collection protocol changes are required to execute my algorithm, this new method allows any researcher around the world to potentially improve their already existing instruments regardless of their budget.

5.5 Limitations

There are several factors limiting the findings of our experiments. Firstly, the training and testing datasets were each composed of 24 pixels with 100 μm resolution. The limited number of pixels and the coarse resolution resulted in a dataset with small tissue space to evaluate the robustness of SAPID. Furthermore, given that the datasets were 6×4 in terms of dimensions, 16/24 (66%) of the pixels were located on the edges of the sample, which restricted the number of neighbouring pixels we can define for each POI.

Secondly, the gold standard protein identification, which I used to train

my algorithms was based on the results of ProteinProphet. There are some intrinsic limitations associated with ProteinProphet. For example, it uses parsimony-driven weighting, where proteins with unique peptides always dominate those without unique peptides, even when the unique peptides have small probabilities [88]. Labeling mistakes are often inevitable in supervised machine learning research. However, false identifications could be reduced by combining the results of multiple protein identification algorithms using a majority voting system, which could directly improve SAPID's performance. Finally, the lack of statistical significance testing when comparing the performance of SAPID against ProteinProphet and Percolator is a limitation which should be addressed in the future.

5.6 Future directions

The future research for this project should address the limitations of our study. For instance, SAPID should be applied on larger datasets with finer resolution to investigate whether it can outperform ProteinProphet and Percolator. Furthermore, performing repeated random experiments for statistical significance testing could increase our confidence about the results of this study. Also, the biological significance of the proteins that are identified with SAPID and their pattern on the tissue should be explored using functional enrichment analysis.

Apart from the machine learning approach we employed to use spatial information for protein identification, it could be interesting to explore Markov Random Field (MRF) approaches to improve protein identification by integrating spatial information in the analysis pipeline. MRF is a graph-based approach where the nodes are random variables connected with edges describing their conditional independence [89]. The local connections in a MRF model propagates information throughout the graph, which means that all the nodes in the graph influence each other. This is a powerful characteristic of MRFs, which have been used in image analysis and even in spatial transcriptomics to detect spatial organization of the cells in the tissue [90]. The advantage of MRF

models compared to our machine learning approach is that a MRF considers all the pixels at once rather than treat one pixel at the time. Furthermore, a MRF can model the connections between different proteins, rather than considering each protein independently as in SAPID, which can become useful when studying proteins that are co-expressed and protein-protein-interactions in a given tissue. Finally, we have defined the neighbourhood of a pixel as the most adjacent pixel in the horizontal, vertical, and diagonal directions. It could be worth investigating what will happen when the window by which the neighbourhood is defined is extended. However, this requires a dataset with larger dimension than the dataset we have used.

References

[1] W. Widłak, *Molecular Biology-Not Only for Bioinformaticians*. Springer, 2013, vol. 8248.

[2] P. A. Muller and K. H. Vousden, "P53 mutations in cancer," *Nature cell biology*, vol. 15, no. 1, pp. 2–8, 2013.

[3] C. Zhang, J. Liu, D. Xu, T. Zhang, W. Hu, and Z. Feng, "Gain-of-function mutant p53 in cancer progression and therapy," *Journal of molecular cell biology*, vol. 12, no. 9, pp. 674–687, 2020.

[4] W. F. Vann, A. Sutton, and R. Schneerson, "Enzyme-linked immunosorbent assay," *Methods in enzymology*, vol. 184, pp. 537–541, 1990.

[5] D. Egger and K. Bienz, "Protein (western) blotting," *Molecular biotechnology*, vol. 1, no. 3, pp. 289–305, 1994.

[6] J. R. Yates III, "A century of mass spectrometry: From atoms to proteomes," *nature methods*, vol. 8, no. 8, pp. 633–637, 2011.

[7] C. E. Parker, V. Mocanu, M. Mocanu, N. Dicheva, and M. R. Warren, *Mass Spectrometry for Post-Translational Modifications*, ser. Frontiers in Neuroscience. CRC Press/Taylor &; Francis, Boca Raton (FL), 2010, ch. 6, ISBN: 9781420076257. [Online]. Available: `http://europepmc.org/books/NBK56012`.

[8] C. Lane, "Mass spectrometry-based proteomics in the life sciences," *Cellular and Molecular Life Sciences CMLS*, vol. 62, no. 7, pp. 848–869, 2005.

[9] A. P. Bruins, "Mechanistic aspects of electrospray ionization," *Journal of Chromatography A*, vol. 794, no. 1-2, pp. 345–357, 1998.

[10] M. Karas and F. Hillenkamp, "Laser desorption ionization of proteins with molecular masses exceeding 10,000 daltons," *Analytical chemistry*, vol. 60, no. 20, pp. 2299–2301, 1988.

[11] A. M. Haag, "Mass analyzers and mass spectrometers," vol. 919, pp. 157–169, 2016.

[12] A. El-Aneed, A. Cohen, and J. Banoub, "Mass spectrometry, review of the basics: Electrospray, maldi, and commonly used mass analyzers," *Applied Spectroscopy Reviews*, vol. 44, no. 3, pp. 210–230, 2009.

[13] C. E. Parker, M. R. Warren, and V. Mocanu, "Mass spectrometry for proteomics," in *Neuroproteomics*, O. Alzate, Ed., CRC Press, 2010, ch. 5, pp. 71–90.

[14] Y. V. Karpievitch, A. D. Polpitiya, G. A. Anderson, R. D. Smith, and A. R. Dabney, "Liquid chromatography mass spectrometry-based proteomics: Biological and technological aspects," *Annals of Applied Statistics*, vol. 4, no. 4, pp. 1797–1823, 2010, ISSN: 19326157. DOI: 10.1214/10-AOAS341.

[15] J. R. Yates, "Mass spectral analysis in proteomics," in *Annual Review of Biophysics and Biomolecular Structure*, vol. 33, 2004, pp. 297–316. DOI: 10.1146/annurev.biophys.33.111502.082538.

[16] K. Tanaka, H. Waki, Y. Ido, *et al.*, "Protein and polymer analyses up to m/z 100 000 by laser ionization time-of-flight mass spectrometry," *Rapid Communications in Mass Spectrometry*, vol. 2, no. 8, pp. 151–153, 1988, ISSN: 10970231. DOI: 10.1002/rcm.1290020802.

[17] J. Mitchell Wells and S. A. McLuckey, "Collision-induced dissociation (cid) of peptides and proteins," in *Biological Mass Spectrometry*, ser. Methods in Enzymology, vol. 402, Academic Press, 2005, pp. 148–185. DOI: https://doi.org/10.1016/S0076-6879(05)02005-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0076687905020057.

[18] J. R. Yates, "Mass spectrometry and the age of the proteome," *Journal of Mass Spectrometry*, vol. 33, no. 1, pp. 1–19, 1998.

[19] M. Mann, R. C. Hendrickson, and A. Pandey, "Analysis of proteins and proteomes by mass spectrometry," *Annual Review of Biochemistry*, vol. 70, no. 1, pp. 437–473, 2001, PMID: 11395414. DOI: 10.1146/annurev.biochem.70.1.437. [Online]. Available: https://doi.org/10.1146/annurev.biochem.70.1.437.

[20] J. A. Paulo, "Practical and Efficient Searching in Proteomics: A Cross Engine Comparison.," *WebmedCentral*, vol. 4, no. 10, 2013, ISSN: 2046-1690. DOI: 10.9754/journal.wplus.2013.0052.

[21] J. K. Eng, A. L. McCormack, and J. R. Yates, "An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database," *Journal of the American Society for Mass Spectrometry*, vol. 5, no. 11, pp. 976–989, 1994, ISSN: 18791123. DOI: 10.1016/1044-0305(94)80016-2.

[22] J. K. Eng, T. A. Jahan, and M. R. Hoopmann, "Comet: An open-source ms/ms sequence database search tool," *Proteomics*, vol. 13, no. 1, pp. 22–24, 2013. DOI: 10.1002/pmic.201200439. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/pmic.201200439.

[23] J. K. Eng, B. Fischer, J. Grossmann, and M. J. MacCoss, "A fast SEQUEST cross correlation algorithm," *Journal of Proteome Research*, vol. 7, no. 10, pp. 4598–4602, 2008, ISSN: 15353893. DOI: 10.1021/pr800420s.

[24] J. K. Eng, M. R. Hoopmann, T. A. Jahan, J. D. Egertson, W. S. Noble, and M. J. MacCoss, "A Deeper Look into Comet - Implementation and Features," *Journal of the American Society for Mass Spectrometry*, vol. 26, no. 11, pp. 1865–1874, 2015, ISSN: 18791123. DOI: 10.1007/s13361-015-1179-x.

[25] A. A. Klammer, C. Y. Park, and W. S. Noble, "Statistical calibration of the SEQUEST Xcorr function," *Journal of Proteome Research*, vol. 8, no. 4, pp. 2106–2113, 2009, ISSN: 15353893. DOI: 10.1021/pr8011107.

[26] K. Ma, O. Vitek, and A. I. Nesvizhskii, "A statistical model-building perspective to identification of MS/MS spectra with PeptideProphet.," *BMC bioinformatics*, vol. 13 Suppl 1, 2012, ISSN: 14712105. DOI: 10.1186/1471-2105-13-S16-S1.

[27] A. Keller, A. I. Nesvizhskii, E. Kolker, and R. Aebersold, "Empirical statistical model to estimate the accuracy of peptide identifications made by ms/ms and database search," *Analytical Chemistry*, vol. 74, no. 20, pp. 5383–5392, 2002, PMID: 12403597. DOI: 10.1021/ac025747h. [Online]. Available: https://doi.org/10.1021/ac025747h.

[28] A. I. Nesvizhskii, A. Keller, E. Kolker, and R. Aebersold, "A statistical model for identifying proteins by tandem mass spectrometry," *Analytical Chemistry*, vol. 75, no. 17, pp. 4646–4658, 2003, PMID: 14632076. DOI: 10.1021/ac0341261. [Online]. Available: https://doi.org/10.1021/ac0341261.

[29] S. Aggarwal and A. K. Yadav, "False discovery rate estimation in proteomics," in *Statistical Analysis in Proteomics*, K. Jung, Ed. New York, NY: Springer New York, 2016, pp. 119–128, ISBN: 978-1-4939-3106-4. DOI: 10.1007/978-1-4939-3106-4_7. [Online]. Available: https://doi.org/10.1007/978-1-4939-3106-4_7.

[30] L. Käll, J. D. Canterbury, J. Weston, W. S. Noble, and M. J. MacCoss, "Semi-supervised learning for peptide identification from shotgun proteomics datasets," *Nature Methods*, vol. 4, no. 11, pp. 923–925, 2007, ISSN: 15487091. DOI: 10.1038/nmeth1113.

[31] M. Spivak, J. Weston, L. Bottou, L. Käll, and W. S. Noble, "Improvements to the percolator algorithm for peptide identification from shotgun proteomics data sets," *Journal of Proteome Research*, vol. 8, no. 7, pp. 3737–3745, 2009, ISSN: 15353893. DOI: 10.1021/pr801109k.

[32] L. Käll, J. D. Storey, M. J. MacCoss, and W. S. Noble, "Posterior error probabilities and false discovery rates: Two sides of the same coin," *Journal of Proteome Research*, vol. 7, no. 1, pp. 40–44, 2008, PMID: 18052118, ISSN: 15353893. DOI: 10.1021/pr700739d. [Online]. Available: https://doi.org/10.1021/pr700739d.

[33] M. R. Emmert-Buck, R. F. Bonner, P. D. Smith, *et al.*, "Laser capture microdissection," *Science*, vol. 274, no. 5289, pp. 998–1001, 1996.

[34] M. Stastna and J. E. Van Eyk, "Analysis of protein isoforms: Can we do it better?" *Proteomics*, vol. 12, no. 19-20, pp. 2937–2948, 2012, PMID: 22888084. DOI: 10.1002/pmic.201200161.

[35] R. M. Caprioli, T. B. Farmer, and J. Gile, "Molecular imaging of biological samples: localization of peptides and proteins using maldi-tof ms," *Analytical Chemistry*, vol. 69, no. 23, pp. 4751–4760, 1997, PMID: 9406525. DOI: 10.1021/ac970888i. [Online]. Available: https://doi.org/10.1021/ac970888i.

[36] J. D. Watrous and P. C. Dorrestein, "Imaging mass spectrometry in microbiology," *Nature Reviews Microbiology*, vol. 9, no. 9, pp. 683–694, 2011. DOI: 10.1038/nrmicro2634.

[37] B. Balluff, C. Schöne, H. Höfler, and A. Walch, "Maldi imaging mass spectrometry for direct tissue analysis: Technological advancements and recent applications," *Histochemistry and Cell Biology*, vol. 136, no. 3, pp. 227–244, 2011. DOI: 10.1007/s00418-011-0843-x.

[38] K. Schwamborn and R. M. Caprioli, "Molecular imaging by mass spectrometry — looking beyond classical histology," *Nature Reviews Cancer*, vol. 10, no. 9, pp. 639–646, 2010. DOI: 10.1038/nrc2917.

[39] Z. Takáts, J. M. Wiseman, B. Gologan, and R. G. Cooks, "Mass spectrometry sampling under ambient conditions with desorption electrospray ionization," *Science*, vol. 306, no. 5695, pp. 471–473, 2004, ISSN: 00368075. DOI: 10.1126/science.1104404.

[40] C. R. Anderton and L. J. Gamble, "Secondary Ion Mass Spectrometry Imaging of Tissues, Cells, and Microbial Systems," *Microscopy Today*, vol. 24, no. 2, pp. 24–31, 2016, ISSN: 1551-9295. DOI: 10.1017/s1551929516000018.

[41] L. H. Cazares, D. A. Troyer, B. Wang, R. R. Drake, and O. J. Semmes, "Maldi tissue imaging: From biomarker discovery to clinical applications," *Analytical and Bioanalytical Chemistry*, vol. 401, no. 1, pp. 17–27, 2011. DOI: 10.1007/s00216-011-5003-6.

[42] E. Bessède, M. Angla-gre, Y. Delagarde, S. Sep Hieng, A. Ménard, and F. Mégraud, "Matrix-assisted laser-desorption/ionization biotyper: Experience in the routine of a university hospital," *Clinical Microbiology and Infection*, vol. 17, no. 4, pp. 533–538, 2011, ISSN: 1198-743X. DOI: https://doi.org/10.1111/j.1469-0691.2010.03274.x. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1198743X14632699.

[43] A. Nilsson, T. E. Fehniger, L. Gustavsson, *et al.*, "Fine mapping the spatial distribution and concentration of unlabeled drugs within tissue micro-compartments using imaging mass spectrometry," *PLOS ONE*, vol. 5, no. 7, pp. 1–8, Jul. 2010. DOI: 10.1371/journal.pone.0011411. [Online]. Available: https://doi.org/10.1371/journal.pone.0011411.

[44] K. Yanagisawa, Y. Shyr, B. J. Xu, *et al.*, "Proteomic patterns of tumour subsets in non-small-cell lung cancer," *The Lancet*, vol. 362, no. 9382, pp. 433–439, 2003. [Online]. Available: https://doi.org/10.1016/S0140-6736(03)14068-8.

[45] L. H. Cazares, D. Troyer, S. Mendrinos, *et al.*, "Imaging mass spectrometry of a specific fragment of mitogen-activated protein kinase/extracellular signal-regulated kinase kinase kinase 2 discriminates cancer from uninvolved prostate tissue," *Clinical Cancer Research*, vol. 15, no. 17, pp. 5541–5551, 2009. DOI: 10.1158/1078-0432.CCR-08-2892.

[46] R. Lemaire, S. Ait Menguellet, J. Stauber, *et al.*, "Specific maldi imaging and profiling for biomarker hunting and validation: Fragment of the 11s proteasome activator complex, reg alpha fragment, is a new potential ovary cancer biomarker," *Journal of Proteome Research*, vol. 6, no. 11, pp. 4127–4134, 2007. DOI: 10.1021/pr0702722.

[47] B. K. Kaletaş, I. M. van der Wiel, J. Stauber, *et al.*, "Sample preparation issues for tissue imaging by imaging ms," *Proteomics*, vol. 9, no. 10, pp. 2622–2633, 2009, ISSN: 16159853. DOI: 10.1002/pmic.200800364.

[48] E. Gemperline, B. Chen, and L. Li, "Challenges and recent advances in mass spectrometric imaging of neurotransmitters," *Bioanalysis*, vol. 6, no. 4, pp. 525–540, 2014, PMID: 24568355, ISSN: 17576180. DOI: 10.4155/bio.13.341. [Online]. Available: https://doi.org/10.4155/bio.13.341.

[49] M. Awad and R. Khanna, *Efficient learning machines: theories, concepts, and applications for engineers and system designers*. Springer nature, 2015, ISBN: 978-1-4302-5990-9. DOI: 10.1007/978-1-4302-5990-9_1.

[50] T. Mitchell, *Machine Learning*, ser. McGraw-Hill International Editions. McGraw-Hill, 1997, ISBN: 9780071154673.

[51] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, ser. Adaptive Computation and Machine Learning series. MIT Press, 2016, ISBN: 9780262035613. [Online]. Available: https://www.deeplearningbook.org/.

[52] A. Burkov, *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019, ISBN: 9781999579517. [Online]. Available: http://themlbook.com/.

[53] L. Wasserman, *All of Statistics: A Concise Course in Statistical Inference*, ser. Springer Texts in Statistics. Springer, 2004, ISBN: 9780387402727.

[54] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*, ser. Springer Series in Statistics. Springer New York, 2009, ISBN: 9780387848587. [Online]. Available: https://hastie.su.domains/ElemStatLearn/.

[55] Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019, ISBN: 9781492032649.

[56] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[57] M. N. Murty and R. Raghava, "Kernel-based svm," in *Support Vector Machines and Perceptrons: Learning, Optimization, Classification, and Application to Social Networks*. Springer International Publishing, 2016, pp. 57–67, ISBN: 978-3-319-41063-0. DOI: 10.1007/978-3-319-41063-0_5. [Online]. Available: https://doi.org/10.1007/978-3-319-41063-0_5.

[58] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning*, ser. ICML'10, Haifa, Israel: Omnipress, 2010, pp. 807–814, ISBN: 9781605589077.

[59] H. He and Y. Ma, *Imbalanced learning : foundations, algorithms, and applications*. John Wiley & Sons, Inc., 2013, ISBN: 9781118646106. DOI: 10.1002/9781118646106.

[60] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, *Learning from Imbalanced Data Sets*. 2018. DOI: 10.1007/978-3-319-98074-4.

[61] T. G. Dietterich, "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms," *Neural Computation*, vol. 10, no. 7, pp. 1895–1923, 1998, ISSN: 0899-7667. DOI: 10.1162/089976698300017197. eprint: https://direct.mit.edu/neco/article-pdf/10/7/1895/814002/089976698300017197.pdf. [Online]. Available: https://doi.org/10.1162/089976698300017197.

[62] C. Nadeau and Y. Bengio, "Inference for the generalization error," *Machine Learning*, vol. 52, no. 3, pp. 239–281, 2003, ISSN: 08856125. DOI: 10.1023/A:1024068626366.

[63] P. D. Piehowski, Y. Zhu, L. M. Bramer, *et al.*, "Automated mass spectrometry imaging of over 2000 proteins from tissue sections at 100-μm spatial resolution," *Nature Communications*, 2020, ISSN: 20411723. DOI: 10.1038/s41467-019-13858-z.

[64] J. Cox and M. Mann, "MaxQuant enables high peptide identification rates, individualized p.p.b.-range mass accuracies and proteome-wide protein quantification," *Nature Biotechnology*, vol. 26, no. 12, pp. 1367–1372, 2008, ISSN: 10870156. DOI: 10.1038/nbt.1511.

[65] E. W. Deutsch, L. Mendoza, D. Shteynberg, *et al.*, "A guided tour of the Trans-Proteomic Pipeline," *Proteomics*, vol. 10, no. 6, pp. 1150–1159, 2010, ISSN: 16159853. DOI: 10.1002/pmic.200900375.

[66] C. Y. Park, A. A. Klammer, L. Käli, M. J. MacCoss, and W. S. Noble, "Rapid and accurate peptide identification from tandem mass spectra," *Journal of Proteome Research*, vol. 7, no. 7, pp. 3022–3027, 2008, ISSN: 15353893. DOI: 10.1021/pr800127y.

[67] A. R. Jones, M. Eisenacher, G. Mayer, *et al.*, "The mzIdentML data standard for mass spectrometry-based proteomics results," *Molecular and Cellular Proteomics*, vol. 11, no. 7, 2012, ISSN: 15359476. DOI: 10.1074/mcp.M111.014381.

[68] C. Matthew C., M. Brendan, B. Robert, *et al.*, "A Cross-platform Toolkit for Mass Spectrometry and Proteomics," *Nature Biotechnology*, vol. 30, no. 10, pp. 1–8, 2012. DOI: 10.1038/nbt.2377.

[69] J. E. Elias and S. P. Gygi, "Target-decoy search strategy for mass spectrometry-based proteomics.," *Methods in Molecular Biology*, vol. 604, pp. 55–71, 2010, ISSN: 19406029. DOI: 10.1007/978-1-60761-444-9_5.

[70] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bulletin*, vol. 1, no. 6, p. 80, 1945, ISSN: 00994987. DOI: 10.2307/3001968.

[71] R. Vallat, "Pingouin: statistics in Python," *Journal of Open Source Software*, vol. 3, no. 31, p. 1026, 2018, ISSN: 2475-9066. DOI: 10.21105/joss.01026.

[72] M. Marozzi, "Testing for concordance between several criteria," *Journal of Statistical Computation and Simulation*, vol. 84, no. 9, pp. 1843–1850, 2014, ISSN: 15635163. DOI: 10.1080/00949655.2013.766189.

[73] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020, ISSN: 15487105. DOI: 10.1038/s41592-019-0686-2.

[74] M. Waskom, "Seaborn: Statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021, ISSN: 2475-9066. DOI: 10.21105/joss.03021.

[75] T. pandas development team, *Pandas-dev/pandas: Pandas*, version 1.1.4, 2020. DOI: 10.5281/zenodo.3509134. [Online]. Available: https://doi.org/10.5281/zenodo.3509134.

[76] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[77] T. Næs and B. H. Mevik, "Understanding the collinearity problem in regression and discriminant analysis," *Journal of Chemometrics*, vol. 15, no. 4, pp. 413–426, 2001, ISSN: 08869383. DOI: 10.1002/cem.676.

[78] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002, ISSN: 10769757. DOI: 10.1613/jair.953.

[79] G. Batista, A. Bazzan, and M.-C. Monard, "Balancing training data for automated annotation of keywords: A case study.," Jan. 2003, pp. 10–18. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.10.2192&rep=rep1&type=pdf.

[80] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, 2004, ISSN: 1931-0145. DOI: 10.1145/1007730.1007735. [Online]. Available: https://doi.org/10.1145/1007730.1007735.

[81] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, pp. 1–5, 2017, ISSN: 15337928.

[82] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06, Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 233–240, ISBN: 1595933832. DOI: 10.1145/1143844.1143874. [Online]. Available: https://doi.org/10.1145/1143844.1143874.

[83] S. Longo, M. Guo, A. Ji, and P. Khavari, "Integrating single-cell and spatial transcriptomics to elucidate intercellular tissue dynamics," *Nature Reviews Genetics*, vol. 22, no. 10, pp. 627–644, 2021. DOI: 10.1038/s41576-021-00370-8.

[84] G. Chen, B. Ning, and T. Shi, "Single-cell rna-seq technologies and related computational data analysis," *Frontiers in Genetics*, vol. 10, p. 317, 2019, ISSN: 1664-8021. DOI: 10.3389/fgene.2019.00317. [Online]. Available: https://www.frontiersin.org/article/10.3389/fgene.2019.00317.

[85] R. Cohen, W. Chan, S. Edgar, *et al.*, "Prediction of pathological stage and clinical outcome in prostate cancer: An improved pre-operative model incorporating biopsy-determined intraductal carcinoma.," *British journal of urology*, vol. 81, no. 3, pp. 413–418, 1998.

[86] K. Kimura, T. Tsuzuki, M. Kato, *et al.*, "Prognostic value of intraductal carcinoma of the prostate in radical prostatectomy specimens," *The Prostate*, vol. 74, no. 6, pp. 680–687, 2014. DOI: 10.1002/pros.22786. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/pros.22786.

[87] C. C. Guo and J. I. Epstein, "Intraductal carcinoma of the prostate on needle biopsy: Histologic features and clinical significance," *Modern pathology*, vol. 19, no. 12, pp. 1528–1535, 2006. DOI: 10.1038/modpathol.3800702.

[88] Y. F. Li and P. Radivojac, "Computational approaches to protein inference in shotgun proteomics," *BMC bioinformatics*, vol. 13, no. 16, pp. 1–19, 2012, ISSN: 14712105. DOI: 10.1186/1471-2105-13-S16-S4.

[89] S. Z. Li, *Markov Random Field Modeling in Image Analysis*, 4205. 2009, p. 371, ISBN: 978-1-84800-278-4. DOI: 10.1007/978-1-84800-279-1.

[90] Q. Zhu, "A hidden markov random field model for detecting domain organizations from spatial transcriptomic data," *Methods in Molecular Biology*, vol. 1935, pp. 251–268, 2019, ISSN: 1064-3745. DOI: 10.1007/978-1-4939-9057-3_16. [Online]. Available: https://doi.org/10.1007/978-1-4939-9057-3_16.

Appendix A

List of features

Table A.1: Summary of the features used in to build the machine learning models

Feature Number	Feature Name	Description
1 – 4	(mean-sd-max-min)_ xCorr_POI	Cross correlation of PSMs corresponding to each protein in POI.
5 – 8	(mean-sd-max-min) deltaCN_POI	Fractional difference (delta correlation) between the best and second best PSMs corresponding to the protein in POI.
9 – 12	(mean-sd-max-min) deltaCN_5_POI	Fractional difference (delta correlation) between the best and fifth best PSMs corresponding to the protein in POI. Note for spectrums with less than 5 PSMs, the worst PSMs was used.
13 – 16	(mean-sd-max-min) SpScore_POI	SpScore is calculated based on the number of fragments ions in the MS/MS spectrum that match experimental data for PSMs correspo nding to the protein in POI.
17 – 20	(mean-sd-max-min) ln_SpRank_POI	Natural logarithm of the SpRank, which is the ranking of PSMs' SpScore for PSMs correspo nding to the protein in POI.
21 – 24	(mean-sd-max-min) massdiff_POI	The mass difference between the experimental and theoretical mass-to-charge ratios for PSMs corresponding to the proteins in POI.
25 – 28	(mean-sd-max-min) ionFrac_POI	Fraction of fragment ions that were matched to a theoretical spectrum.
29 – 33	Charge_(2,3,4,5,6) POI	Mode of the predicted charges corresponding to the PSMs in POI.

34 – 43	top_xCorr (exp_mz, xCorr, deltaCN, deltaCN_5, SpScore, ln_SpRank, massdiff, ionFrac, enz_C_ter, enz_N_ter) POI	10 features from the PSMs with highest xCorr extracted in POI. exp_mz indicates the experimental mass-to-charge ratio. enz_(C, N)_terminus is a Boolean value indicating if there is a tryptic (C,N) terminus on the peptide sequence corresponding to the PSM with the highest xCorr.
44	spectral_count_POI	Number of PSMs associated with a protein in POI.
45	peptide_count_POI	Number of unique PSMs associated with a protein in POI.
46 – 48	top_xCorr_(1,2,3)	The highest xcorr scores of matches associated with a given protein from POI's adjacent pixels. Note: the three values must come from 3 different pixels. In case there is <3 pixels, the missing values will be filled with zero.
49 – 51	top_deltaCN_(1,2,3)	The highest delta correlation (deltaCN) of matches associated with a given protein from POI's adjacent pixels. Note: the three values must come from 3 different pixels. In case there is <3 pixels, the mussing values will be filled with zero.
52 – 54	top_SpecCount_(1,2,3)	The highest spectral counts of matches associated with a given protein from POI's adjacent pixels. Note: the three values must come from 3 different pixels. In case there is <3 pixels, the missing values will be filled with zero.
55	mean_surrounding_ spectral_count	The average spectral counts associated with all the surrounding pixels. Note: surrounding pixels with no PSMs associated with a given protein will be given a Spec_Count of zero.
56	mean_all_surrounding_ spectral_count	The average spectral counts associated with the surrounding pixels that contain the PSMs. Note: in case there is no PSMs in any of the surrounding pixels, the value will be zero.
57	mean_all_surrounding_ xCorr	The average of all the xCorrs corresponding to a protein in the neighboring pixels. Note: surrounding pixels with no PSMs associated with a given protein are not considered.

58	mean_all_surrounding_ deltaCN	The average of all the deltaCN corresponding to a protein in the neighboring pixels. Note: surrounding pixels with no PSMs associated with a given protein are not considered.
59	mean_all_surrounding_ massdiff	The average of all the massdiff corresponding to a protein in the neighboring pixels. Note: surrounding pixels with no PSMs associated with a given protein are not considered.
60	num_surrounding_ pixels_with_PSM	Number of neighbouring pixels that have PSMs associated with a given protein in POI.
61	One_peptide_in_ protein_POI	Boolean value indicating there is only one PSM available in the POI for a given protein
62	spatial_info_not_exist	Boolean value indicating there is no spectral information available for a given protein on the surrounding pixels.