uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Mai Moussa Chétima

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

"Auto-Turning Mechanisms for Vision-Based Food Inspection Systems

TITRE DE LA THÈSE / TITLE OF THESIS

Pierre Payeur

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

James Green

Nathalie Japkowicz

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Auto-Tuning Mechanisms for Vision-Based Food Inspection Systems

by:

Mai Moussa Chétima

A thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements for the degree of
Masters of Applied Science – Electrical and Computer Engineering

Ottawa-Carleton Institute of Electrical and Computer Engineering
School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-61211-8
Our file Notre référence
ISBN: 978-0-494-61211-8

Abstract

Machine vision solutions are becoming a standard for quality inspection in several manufacturing industries. In the processed-food industry where the appearance attributes of the product are essential to customer's satisfaction, visual inspection can be reliably achieved with machine vision. But such systems often involve the extraction of a larger number of features than those actually needed to ensure proper quality control, making the process less efficient and difficult to tune. This work experiments with several machine learning techniques in order to automate the initial tuning of a real-time vision-based food inspection system or to improve its performance. The impact of feature selection techniques on machine learning is also assessed. Identifying and removing as much irrelevant and redundant information as possible for a given learning scheme reduces the dimensionality of the data and allows classification algorithms to operate faster. In some cases, accuracy on classification can even be improved. The effect of filter-based and wrapper-based feature selectors is experimentally evaluated on different bakery products to identify the best performing approaches when combined with three fundamentally different machine learning strategies.

Acknowledgements

This research project would not have been possible without the support of many people. I wish to express my gratitude to my supervisor, Dr. Pierre Payeur, who was abundantly helpful and offered invaluable assistance, support and guidance. Deepest gratitude is also due to the members of the Vision, Imaging, Video and Autonomous Systems (VIVA) research laboratory, without whose knowledge and assistance this study would not have been successful.

Special thanks to the University of Ottawa for providing the financial means and laboratory facilities. I would like to also acknowledge the partial financial support of Precarn Inc., and the collaboration of Dipix Technologies Inc. to the first phase of this research.

Last, but not least, I wish to express my love and gratitude to my beloved family and friends; for their understanding, endless love and encouragement, and unconditional support through the duration of my studies.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

As for several other manufacturing sectors, the food industry has been trying to automate the quality control processes in order to decrease production costs and increase the quality and uniformity of the production. Machine vision-based systems are of particular interest when it comes to measuring the superficial characteristics of a product for classification purposes. Most of the external quality attributes of a product can be inspected visually before the packaging line and items that do not satisfy the set standards are automatically rejected. Such machine vision systems have been used over a wide variety of inspection applications in the food manufacturing industry including meat, fruits and vegetables, bakery products, and prepared consumer foods [1, 2].

Machine learning is at the core of several vision-based inspection systems. But in most applications, the inspection system is usually dependent on one classification scheme whose configuration requires an in depth knowledge of the product under inspection and is very often subjective and based on trial and errors. The exact set of features that are critical for the quality control is also difficult to determine. It is therefore difficult and time-consuming to determine on which features to focus the system's attention when configuring the inspection system in order to sustain a high production rate with a maximum of reliability. When machine learning is used, one intuitive solution is to include all features that could possibly be relevant and let the learning algorithm decide which features are in fact worthwhile [3]. A more structured way is to identify the relevant features by means of rigorous feature selection techniques and make the inspection system concentrate only over a feature space of a reduced dimension. Such feature selection techniques are often categorized as filters or wrappers. In the filter approach, the feature selector is independent of any learning algorithm and serves as a filter to sieve the irrelevant and/or redundant attributes. The wrapper feature selectors are rather integrated with a learning algorithm to actively determine the relevant attributes for that particular learning algorithm.

1.1 Motivations

The motivation for this research work is to evaluate the effectiveness of some state-of-the-art machine learning techniques and the impact of feature selection on those learning techniques for an application on real-time vision-based food inspection systems that operate on several types of bakery products such as hamburger buns, tortillas, and bread loaves. This motivation originated from two use cases presented by Dipix Technologies Inc. [4], a company specialized in the design and implementation of automated vision-based inspection solutions for the processed food industry. Those two use cases arise every time a vision-based system needs to be configured to inspect a new type of bakery product for the first time. The first use case is an attempt to answer the question "how can a system objectively identify the most relevant features out of the many parameters that are automatically extracted by the vision-based food-inspection procedure?" The second use case highlights the need to configure vision-based food inspection systems without requiring extensive knowledge about the product to be inspected, or the availability of an already built-in classifier.

The goal of the research is consequently to develop a process for the automated determination of the most suited machine learning algorithm in conjunction with the most relevant subset of features that can also automatically adapt its classification model to the type of products being inspected. This contribution represents a major evolution over the current technology where inspection parameters are mostly set through trial and error procedures for a single built in statistical classifier. It makes the configuration and maintenance of inspection systems more straightforward, even for new products, while improving the uniformity of the production and reducing the costs of system's configuration. This work aims at providing answers to the yet complex question "how can a vision-based system for quality industrial inspection be initially configured in the most accurate and economical way, while also allowing the inspection to run faster after configuration?"

1.2 Objectives

In order to propose a formal process where the vision-based food inspection system can auto-tune a classification model in a straightforward but yet robust manner which can automatically adapt to the product under inspection, the work reported here focuses on the following main objectives:

a) The evaluation of existing state-of-the-art feature selection and machine learning techniques for application in the processed food industry and their eventual conciliation. This part of the work does not aim at enhancing machine learning techniques, but rather to identify some solutions that offer high performance and should be selected for application to industrial inspection of baked food products.

b) The development of a complete data logging module embedded in the available vision-based food inspection's software. This data logging synchronized and encapsulated in the system's image processing software also handles low level interactions with the systems hardware. The extraction and storage of features from products samples by the data logging module will therefore make raw data available for experimentation.

c) The implementation of potential machine learning schemes for a vision-based food inspection system. Experimental comparative evaluation of the performance of those machine learning algorithms can then be achieved by the system. If conclusive, then machine learning algorithms should replace the current custom built classifier which is very basic and based on simple statistical and Boolean operations.

d) The integration of open-source feature selection software with the commercial software built in the vision-based food inspection system. The system can then evaluate its predicted machine learning performance with and without feature selection in order to assess the impact of feature selection on each learning algorithm.

e) The auto-tuning of the system based on the model selected by a human operator. The human operator should make his or her judgment based on

the predicted performance auto-evaluated by the system and the goals he or she wants to achieve. The inspection model should be a machine learning technique applied over the full set of features or a reduced set of features in the event that feature selection allows better performance.

f) The application of the proposed process on a wide variety of bakery products in order to validate (or invalidate) its credibility.

1.3 Thesis Outline

This thesis is organized in seven chapters, the first of which is this introductory chapter. The second chapter reviews the various applications of machine learning with specific emphasis on their applications in the world of machine vision and industrial quality control. Chapter 2 also provides reviews of feature selection techniques and briefly provides the necessary background for the last five chapters. Chapter 3 presents the rationales behind the evaluated machine learning techniques and the different feature selectors along with some implementation level details. Chapter 4 introduces the vision-based food inspection system which served as the platform for the experimental evaluation, and the three different bakery products which were used for experimentation. Results of feature-space reduction are also presented in that same chapter. Chapter 5 unites machine learning and feature selection by a comparative evaluation of the predicted performance of machine learning algorithms with and without feature selection over each of the three categories of experimental bakery products. Chapter 6 investigates auto-tuning of vision-based food-inspection systems and presents case studies of the performance of auto-tuned inspection models. Chapter 7 recapitulates the major topics discussed in this thesis while stressing their contributions to vision-based inspection and possible future work to be performed.

Chapter 2. Literature Review

In the processed-food industry where the external quality attributes of the product are inspected visually before the packaging line, several image analysis technologies have been developed and tested over the last two decades for diverse inspection applications including bakery products, meat, fruits and vegetables [1, 2]. However, vision-based inspection systems often involve the extraction of a larger number of features than those actually needed to achieve reliable quality analysis, resulting in unnecessary computations and lowering the production rate. In this literature review, the first section will present an overview of the latest applications of machine vision systems for the processed food-industry. The second section will introduce state of the art techniques used for feature-space reduction, and the third section will cover several learning schemes for machine vision applications. Finally, a summary will conclude this chapter.

2.1 Computer Vision Systems in the Food Industry

A computer vision system is generally composed of hardware for acquiring and storing digital images, software for processing images, and eventually modules for communicating results to other automated systems [5]. This section will explore recent applications of computer vision systems for the processed food industry.

2.1.1 Computer Vision Systems in the Food Industry

Computer vision systems are increasingly being used in numerous industrial inspection and quality control applications. This section will present recent applications on meat, poultry and fish, fruits and vegetables, and bakery products.

2.1.1.1. Meat, Poultry and Fish industry

Chandraratne *et al.* [6] used a machine vision system to predict lamb carcass grades (six different grades) by extracting 148 features from lamb chop images. They compared two categories of classification: a statistical classification (Discriminant Function Analysis - DFA) and an Artificial Neural Network (Multi-Layer Perceptron - MLP). DFA tries assigning individuals into certain groups identified a priori in the samples. It finds a set of linear combinations of variables whose values are as close as possible within groups and as far as possible between groups. Artificial Neural Networks on the other hand try capturing complex linear/nonlinear Multiple Inputs / Multiple Outputs (MIMO) relations given some training samples. Chandraratne *et al.*'s work [6] showed that MLP out-performed DFA by 12%, totaling a classification accuracy of 96.9%. However, dimensionality reduction has been performed on the 148 initial features using Principal Component Analysis (PCA). PCA is explained in section 2.2.1.

Basset *et al.* [7] experimented with a machine vision system in order to distinguish muscle type, age and breed of different bovine meats. According to their research, those three biological factors directly influence the structure and composition of the three different muscles they selected and cut off 26 carcasses of different breeds and different ages. Their vision system extracted 58 features. The classification was done using "*k* nearest neighbor" method, where *k* is a positive integer. This classification method pre-computes prototypes from patterns of the training samples. During classification, for every unknown pattern, the *k* nearest neighboring prototypes are considered, classifying the unknown pattern in the same class as the majority class of the *k* nearest neighbors. However, poor classification accuracy as low as 25.4% in some cases was obtained, due to correlation between many of the features extracted. Basset *et al.* [7] assumed that identifying the most relevant and uncorrelated features might improve the classification accuracy, but recognized that it is not necessarily an obvious task.

Park *et al.* [8] developed a computer vision based system for separating tumorous, bruised and skin-torn poultry carcasses from normal carcasses. The textural properties of

images are extracted in the spatial grey level domain by computing the probability that a pixel of a particular grey occurs at a specified direction and distance from its neighboring pixels. Experiments on a multispectral camera with various wavelengths (542-847 nm) demonstrated that spectral images scanned at optical wavelengths 542 nm and 700 nm produce a better distinction between normal poultry carcasses and tumorous, bruised or skin-torn carcasses in the spatial grey level domain. A neural network classifier achieved a correct classification rate of around 91.4% while distinguishing normal poultry carcasses from tumorous ones. Park and Chen [9] later used a similar approach to distinguish between unwholesome and wholesome poultry carcasses and found that a linear discriminant model was better in identifying unwholesome carcasses by achieving close to 95.6% accuracy whereas a quadratic model was better identifying wholesome poultry carcasses by reaching 97% true positive classification. The system analyzed a total of 13 extracted features.

The fish industry has also been taking advantage of machine vision systems. Storbeck and Daan [10] developed a computer vision system for recognizing different fish species moving on a conveyor belt whose direction is perpendicular to a camera as illustrated in Figure 2.1.

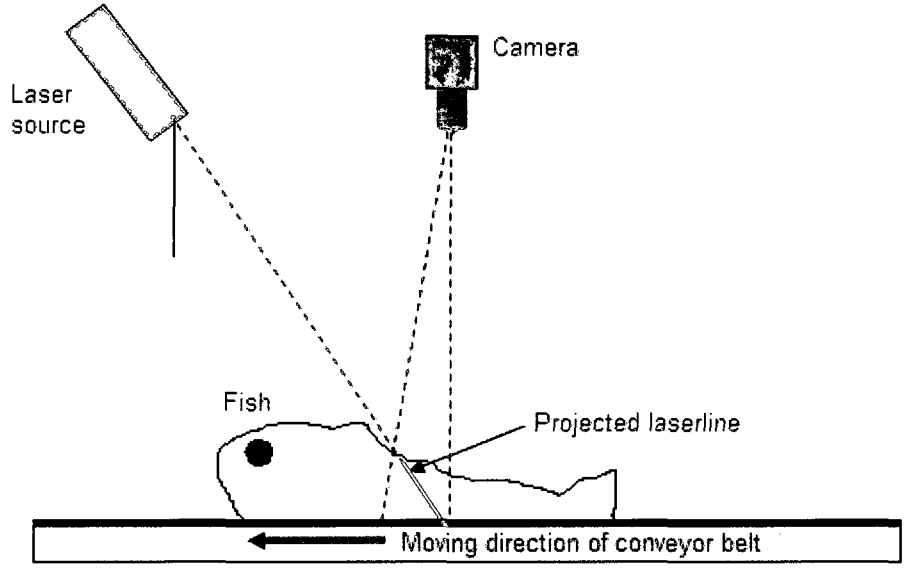


Figure 2.1. Example of a vision system for computing fish volume (adapted from [10])

The width and height of the fish at various locations are extracted while the fish moves at a speed of 0.21 m/s on a conveyor belt. Instead of using a conventional 2-dimensional camera, Storbeck and Daan [10] used a line scan camera which scans one line at a time in a direction perpendicular to the conveyor belt's moving direction. The helium-neon laser source is equipped with a cylindrical lens, therefore spreading the laser light in one direction. Under tedious configuration, the line scan camera should capture the reflection of the laser light at a certain angle; and knowing the speed of the conveyor belt, image processing algorithms combine the width given by the line scan camera and the height computed from the reflection of the laser light. Using a neural network classifier, this implementation was able to achieve more than 90% correct classification on 251 fishes covering six species of fish. This accuracy was at the cost of 15 minutes for training the network of 400 input nodes, and 7.6 ms classification time. This system was originally built by the same team of researchers for the purpose of estimating the weight of flatfishes [11]. They advanced that given a reasonably constant weight of fish per unit volume; they could compute the approximate weight of the fish. The system could estimate the fish weights within 5% of error. Storbeck and Daan [11] concluded that this error rate was sufficiently accurate for commercial purposes.

2.1.1.2. Fruits and Vegetables Industry

The fruits and vegetables industry is also among computer vision inspection systems' customers. For example, Peng and Lu [12] developed a multi-spectral imaging system for estimating apple fruits firmness and soluble solid content in real-time. They analyzed spectral images of "Golden Delicious apples" at four different wavebands, namely 680, 800, 900 and 950 nm. Six functions were empirically derived from test samples, and a linear regression method was used to select the closest function characterizing the extracted features of every apple under inspection. Finally, the firmness of the apples is computed using the pre-stored model corresponding to that function. The system achieved around 0.92% of prediction error. The major difficulty was the empirical derivation of the six classification models.

Kondo *et al.* [13] extracted four features from oranges in order to evaluate their quality. They trained a neural network using those four extracted features from test samples and their corresponding sugar content. The results were very poor due to the limited number of training samples (only five training samples) which were invalidated by test samples. Kondo *et al.* recognized that the four features they selected could not confidently allow sugar content estimation.

Shahin *et al.* [14] line-scanned sweet onions for internal defects using X-ray imaging. Their goal was at the same time to find the best classifier for their purpose between a Bayesian classifier and a neural network classifier. Their work indicated that the latter had a better accuracy rate (90%) compared to a Bayesian classifier.

2.1.1.3. Bakery Products Inspection

Machine vision systems are also used for quality inspection of numerous bakery products. Davidson *et al.* [15] implemented a vision system for extracting physical features of chocolate chip cookies such as size, shape, baked dough color, and fraction of the top surface area that was chocolate chips. Afterwards, three out of the four extracted features were subjectively used to develop four fuzzy models for predicting consumer ratings of the chocolate chips. Only two out of the four fuzzy models gave results reasonably close to consumer judgments. One possible cause is the subjectivity of the consumers in their judgment. Another possible justification is the fact that the authors subjectively sacrificed one feature out of four for complexity reasons.

Abdullah *et al.* [16] tried automating muffins' inspection by means of a computer vision system. Unable to distinguish brown muffins from dark muffins using a single threshold, they had to manually cluster the HUE histogram plot using a quadratic discriminant function. Afterwards, parameters of the discriminant quadratic function also had to be manually computed and stored from samples. The system was trained with 100 light muffins and 100 dark muffins, and tested with 100 pregraded and 100 ungraded

muffins. Moreover, Abdullah *et al.* noticed that their system generally gave better results when trained and tested using four clusters of the HUE histogram instead of a reduced number of three clusters. Given that the clusters have been intuitively determined and that only color attributes of the muffins have been considered, the authors indicated that a formal feature selection could benefit the classification of brown and dark muffins.

2.1.1.4. Summary on Food Inspection Systems and Implications

In the previous sub-sections, we have covered numerous state-of-the-art computer vision-based systems for food inspection. A lot of those systems, if not all, share several goals. Some of those goals are (1) identifying and (eventually) removing defective products and contaminants, (2) reducing human interaction and subjectivity in the process, and (3) the usage of non invasive and non destructive processes for controlling the quality of a product. However, a lot of those vision systems tend to share also the same problem of identifying which features are the most critical in determining the quality of a product, or even deciding which classification technique is best suited for a particular product. For a given classifier, very often, subjective feature reduction impedes the accuracy of product classification. In most of the cases, it comes down to choosing between (1) reducing the number of extracted and analyzed features in order to process the product faster by sacrificing the accuracy of classification, or (2) extracting and analyzing more features in order to obtain a better classification accuracy while sacrificing the product's processing time. Given the duality of the problem between guessing which features to analyze and not even formally knowing which classifier to use, our aim is to propose a formal and objective process for not only selecting the best set of features for product classification of an existing vision-based food inspection system, but instead to determine a rationale for selecting the best combination of a classifier and a subset of features (if necessary) tailored for that classifier and a particular product. Therefore, this process should automatically adapt to any type of product under classification. This will provide the existing system, that currently supports only one classifier, with a mode to auto-tune a configuration of classifier and features the first

time a new product classification is to be calibrated. In the next section, we will present an overview of several feature space reduction techniques applied in the world of Data Mining and Knowledge Discovery. We have briefly invoked several classification techniques used in machine vision systems, but close to none feature space reduction method. The next two sections will explore several feature dimensionality reduction strategies followed by a more extensive review of classification methods in the world of machine learning.

2.2 Feature Selection Techniques for Machine Learning

Several vision-based inspection systems take advantage of machine learning techniques. However for most of those inspection systems, identifying the features that need to be evaluated by the chosen machine learning technique for reliable industrial quality control has been a recurrent issue. A common approach is to guess the relevant features based on some *a priori* knowledge of the product to be inspected, or to simply let the machine learning algorithm decide which features are useful to the inspection. Another approach, known as *feature selection* (also known as feature subset selection, or attribute selection), is the process of identifying and removing as much of the irrelevant and redundant information as possible. Such formal feature selection techniques are often categorized as filters or wrappers [17, 18, 19, 20]. We will concentrate only on feature selection techniques for machine learning, as opposed to feature selection in statistics and pattern recognition [21] where most works have dealt with linear regression [22] and share the common assumption of monotonicity (increasing the number of the features can only improve performance) that does not apply to most machine learning algorithms [19].

2.2.1 Filter-based Feature Selection

In the filter approach, the feature selector is independent of the learning algorithm and serves as a filter to sieve the irrelevant and/or redundant attributes. Figure 2.2 illustrates the principle of filter-based feature selection.

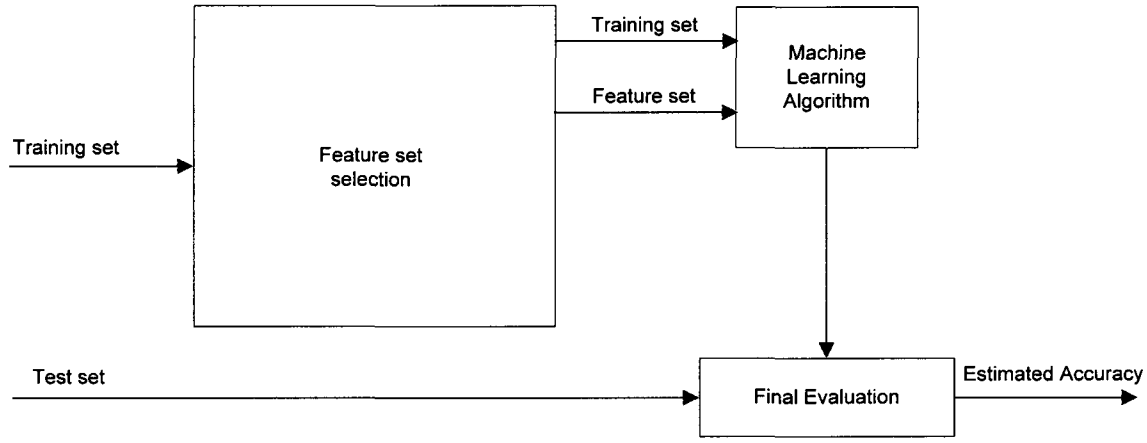


Figure 2.2. The filter feature selection approach

Typically, filters make use of available training data in order to filter out undesirable features of the data before training a machine learning algorithm. The machine learning algorithm is trained only over a reduced set of features produced by the filter-based feature selector. As we can observe from Figure 2.2, no knowledge of the machine learning algorithm to be used nor its interface are required to perform feature selection. The filter-based feature selector is supposed to optimize the dimensionality of the dataset with respect to accuracy of classification. Except for rare cases, most filter-based feature selectors operate only on discrete class problems.

Almuallim and Dietterich [3, 18] developed a filter-based feature selector known as the FOCUS algorithm and compared its performance to two other similar algorithms, namely ID3 [23, 24] and FRINGE [25]. ID3 is an algorithm which builds a decision tree by iteratively picking a subset of the attributes that describe a training dataset, adding a feature at each iteration if the feature enhances the accuracy of the dataset's description.

FRINGE on the other hand iteratively constructs new features based on the decision tree output of an induction algorithm. The iterations are stopped if no feature improves the description. It is worth mentioning that ID3 and FRINGE are machine learning algorithms which have built-in feature selection. FOCUS exhaustively examines all subsets of features and ends up selecting the minimal subset of features sufficient to determine the class of all training instances. FOCUS, ID3, and FRINGE have this preference for small sets of features, known as *min-features bias*. Experiments conducted by Almuallim and Dietterich showed that FOCUS feature selection implements min-features bias better than ID3 and FRINGE (which are machine learning techniques with built-in feature selection) while giving comparable or better accuracy. They ended up recommending the usage of a feature selector before using ID3 or FRINGE, and having those last two algorithms focus only on the relevant features for better results. However, exhaustive evaluation of all possible feature subsets could become a severe limitation when the number of features in the full dataset is relatively large. As a matter of fact, FOCUS's time complexity is $O(N^M)$ for M training instances of a dataset containing N attributes. Moreover, FOCUS works with only binary noise-free data.

Liu and Setiono [26] developed a technique known as Chi2 algorithm which is based on the chi-square (χ^2) statistic. This technique automatically discretizes continuous attributes based on χ^2 statistic and removes any attribute which values are discretized into one interval since it is presumed that this attribute has no impact on differentiating patterns. Experiments on three real biomedical related datasets and two artificially generated datasets indicated that Chi2 algorithm could reduce by up to 50% the dimension of the features on some of the datasets. One major limit of this technique is the fact that it can operate only on numeric attributes (continuous-valued attributes), meaning that datasets containing nominal attributes are out of consideration.

Modrzejewski [27] developed a feature selector based on Rough set theory. Rough set theory was originally introduced by Pawlak [28]. A Rough set is a pair of sets which give the lower and the upper approximation of the original full set. The proposed

method, called PRESET, has a time complexity of $O\left(\frac{M\left(N^2-N\right)}{2}\right)$ which is only a small fraction of time compared to an exhaustive method which has a time complexity of $O(N^M)$, where M is the number of training instances of a dataset containing N attributes. However, experimental validation of the method was able to reach only 83% classification accuracy using the reduced feature sets proposed by PRESET. Modrzejewski [27] judged the performance of PRESET relatively low in terms of accuracy given that the datasets have been randomly generated, but claims that the low complexity of this technique creates an advantage by requiring less computational resources than exhaustive feature selection techniques. The number of attributes in the full experimented original datasets varies from four to eight.

Yang and Honavar [29] used a genetic algorithm for feature subset selection. Genetic algorithms are inspired by the theory of evolution which states that in a living environment, the "best" individuals have a greater chance to survive and a greater probability to spread their genomes by reproduction. The mating of two "good" individuals causes the mixing of their genomes, which may result in a "better" offspring. The terms "good", "best", and "better" are related to the fitness of the individuals to their environment. Yang and Honovar experimented their theory using several datasets and obtained in some cases a feature compression ratio close to 50% and an accuracy close to that of the full original dataset. A neural network was trained with the reduced feature set. A noticeable fact is that among the experimented datasets, the largest datasets contained 69 attributes and achieved a compression of about 55% with only 83.5% correct classification. Nevertheless, Yang and Honavar [29] indicated that the performance of their approach provided higher generalization accuracy than several techniques for the datasets under consideration.

Hall [30, 31] introduced another heuristic filter-based feature selector called CFS (Correlation-based Feature Selection). This filter heuristically evaluates the "merit" of a subset of features by taking into account the usefulness of individual features for predicting the class label. It also considers the level of inter-correlation among features.

The algorithm originally developed to address discrete class problems has been adapted to also attack continuous class problems. Experiments on 36 datasets demonstrated that CFS reduced the number of features by more than half for 70% of the discrete class datasets, also enhancing accuracy on most of the datasets for various machine learning algorithms. The time complexity of CFS is $O\left(\frac{M\left(N^2-N\right)}{2}\right)$ for M training instances of a dataset containing N attributes.

Another filter-based attribute selector, known as Consistency-based Feature Selector, has been experimented on several datasets [32]. This filter uses an inconsistency evaluation criterion proposed by Liu and Setiono [33]. Experimentation showed Consistency-based Feature Selection to be able to remove up to 92% of the attributes of some datasets while still improving the accuracy of classification. Moreover, the consistency criterion is relatively easy to implement.

Kira and Rendell [34, 35] proposed a feature selector known as RELIEF which could deal with discrete and continuous attributes but was limited to only two-class problems (a boolean output class). RELIEF was later enhanced by Kononenko [36] to tackle noisy, incomplete and multi-class datasets. RELIEF ranks attributes based on training instances. Kononenko's experiments on both artificial and real world data showed RELIEF to be a good candidate for selecting relevant features out of imperfect data.

Another filter, called Principal Component Analysis (PCA), is a statistical technique which reduces dimensionality of data by transforming the original set of variables into new axes while retaining as much variation as possible in the original dataset. From all the filter-based feature selectors we have presented, PCA is the only unsupervised method in the sense that it does not consider the class attribute of data samples in the dimensionality reduction process. This method handles k-valued discrete attributes by converting them to k binary attributes. PCA has been experimented by Chandraratne *et al.* [6] on a vision-based system designed for lamb grade classification. Unfortunately,

researchers pointed out that Principal Component Analysis tends to increase the dimensionality of the original space when multi-valued discrete attributes are present [32].

2.2.2 Wrapper-based Feature Selection

As the name implies, wrapper feature selectors work as a wrapper around a learning algorithm and rely on the latter to determine the relevant attributes. As shown on Figure 2.3, a wrapper feature selector uses a machine learning algorithm as a black box, therefore needing only the interface of the induction algorithm. In fact, knowledge of the learning algorithm itself is not necessary [20]. The wrapper feature selector repeatedly searches for a "good" feature subset by using the actual target learning algorithm as part of the evaluation function.

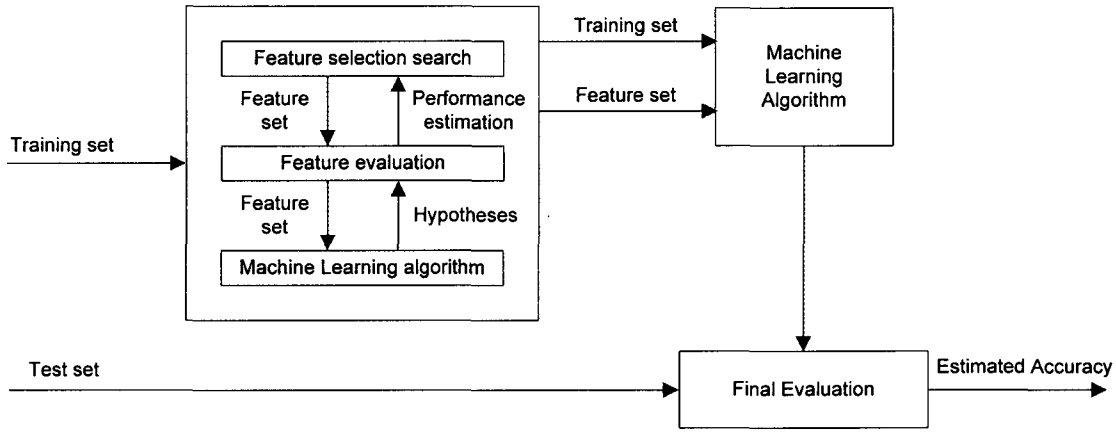


Figure 2.3. The wrapper feature selection approach

Kohavi and John [20] pragmatically evaluated the performance of wrapper feature selectors on several datasets for a decision-tree learning scheme and a Naïve Bayes learner. Their work showed wrapper selectors to significantly improve the accuracy on several datasets classification. Because wrappers select features according to the learning scheme to be used, they tend to select fewer features than filters. However,

several researchers pointed out that feature wrappers are practically limited by the time complexity of the learning algorithm [33].

Maldonado and Weber [37] evaluated the performance of feature wrappers on four datasets with Support Vector Machines (SVM) as the target learning algorithm for binary classification. SVM works by looking for the optimal hyperplane which allows the separation of training samples. SVM builds a $p - 1$-dimensional hyperplane for a p-dimensional vector of features. Unless the computed multiplier of a specific feature is null, there is no dimensionality reduction achieved by SVM. Several representations such as polynomial functions can be used to mathematically describe the hyperplane. Their experiments on the four aforementioned datasets exhibited a feature dimensionality compression ratio varying from 69.33% to 98.25% when the feature wrapper is trained with SVM as the target learning algorithm. However, Maldonado and Weber [37] did not provide experimental results on the classification accuracy of SVM when operating only over the reduced set of features.

Li *et al.* [38] have also studied the usage of feature wrappers tailored to operate with SVM as the classification algorithm. They achieved a feature reduction of more than 85% on the dataset that they have considered using the wrapper feature selection. Their aim was to enhance four different categories of known techniques of computer network intrusion detection. Their work showed that generally, not only does the wrapper feature selection reduce the computational cost of the different network collision detection techniques, but it also enables a better detection success for those latter techniques.

Wrapper feature selection has also attracted the interest of biomedical researchers. Hong and Cho's work [39] aimed at classifying cancerous human tumors based on the genes of the human subject. Given the large number of genes to be analyzed by a *k-nearest neighbor* classifier where $k=5$ (refer to section 2.1.1.1), Hong and Cho [39] explored the usage of a feature wrapper trained to operate with this classifier. Their results showed that correct classification can be achieved for up to 99.1% of the test

subjects while only using the reduced features proposed by the *k-nearest neighbor* wrapper. The approach proposed by Hong and Cho's [39] that they called *gene boosting,* is actually widely referred to as *adaBoost.* AdaBoost, short for *adaptive boosting,* is used in conjunction with machine learning algorithms in order to "boost" (improve) their performance. AdaBoost repeatedly builds classifiers using the same machine learning technique. This technique is considered adaptive as it tweaks the subsequent classifiers built in favor of the instances that those classifiers misclassified. At each iteration, the weights of misclassified instances are increased so that the new classifier focuses more on those examples. Alternatively, one can decrease the weight of the correctly classified instance.

Huang *et al.* [40] experimented wrapper feature selection approach in order to identify an optimal subset of features out of 23 commonly used features for stock trend prediction. Even though a neural network achieved a maximum classification performance as low as 69.01 % when operating only over the features selected by the feature wrapper, Huang *et al.* [40] claimed that the wrapper feature selection has improved the stock trend prediction on the two datasets that they have considered. No specific information on the number of features selected out the total of 23 was provided.

2.2.3 Feature Search Strategies

Feature selection algorithms employ several search strategies which can be grouped into two main categories: exhaustive search and heuristic search. Each category of search can generally be further broken down into forward selection and backward elimination. Forward selection starts with no variables and adds them one by one, at each step adding the one that decreases the error the most, until any further addition does not significantly decrease the error (or increases it only slightly). On the contrary, backward elimination starts with all the variables and removes them one by one, at each step removing the one that decreases the error the most, until any further removal increases the error significantly. Exhaustive and heuristic search strategies are explained below.

2.2.3.1. Exhaustive Search

Exhaustive search explores all possible subsets of M features chosen from N attributes (where $M \leq N$) in order to theoretically find the optimal combination of these M features. Figure 2.4 shows for instance all possible subsets of four attributes where an attribute is present when its circle is black and absent when its circle is white. Note that the states in the space of four features are partially ordered with each of a state's children (to the right) including one more attribute than its parents.

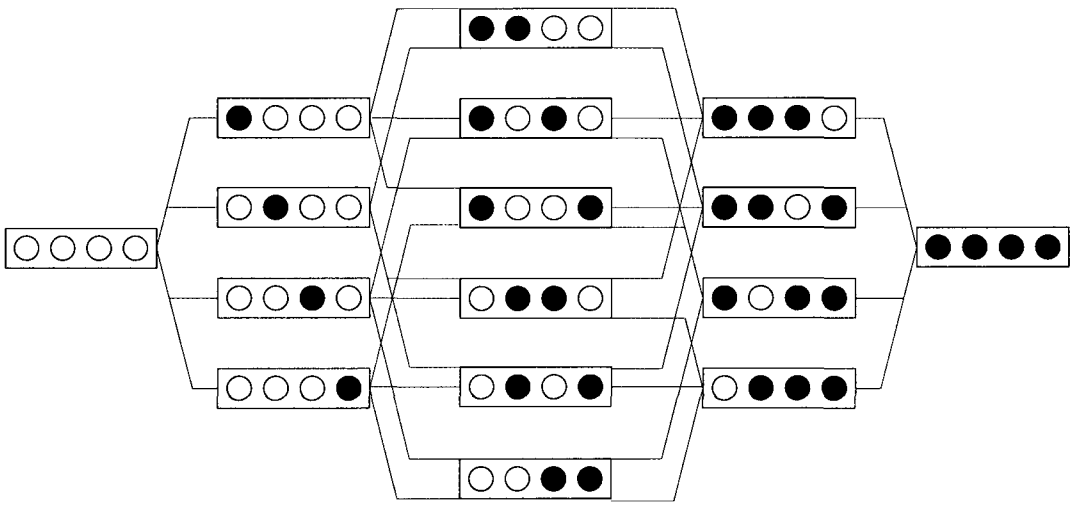


Figure 2.4. Exhaustive search: all possible subsets of four attributes (adapted from [22])

Searching the space from left to right in Figure 2.4 above would be a forward selection, and searching from right to left would be a backward elimination. However, exhaustive search is generally impractical when the number of attributes in the original dataset is relatively large. As a matter of fact, there exists 2^N possible distinct subsets of a set containing N attributes (including both the set itself and the empty set) since each attribute is either included in the subset or is not. Note that the number of subsets is the same whether the attributes are binary, discrete, or numeric. Therefore, in most implementations of exhaustive search algorithms, at some operator defined stopping point, the subset of features with the highest score discovered up to that point is selected as the satisfactory feature subset. The stopping criterion varies with the algorithm;

possible criteria include but are not limited to a subset score exceeding a threshold or a program's maximum allowed run time being surpassed. Figure 2.4 shows the $2^4 = 16$ distinct subsets of a set containing $N = 4$ attributes. The time complexity of exhaustive search makes heuristic search particularly attractive for practicability reasons.

2.2.3.2. Heuristic Search

Instead of exploring all possible subsets of features, heuristic search algorithms ignore whether the solution to the problem can be proven to be correct, but usually produce a good solution or solve a simpler problem that contains, or intersects with, the solution of the more complex problem. Heuristics are typically used when there is no known way to find an optimal solution, or when it is desirable to give up finding the optimal solution for an improvement in run time. For instance, say you are packing odd-shaped items into a box. Finding a perfect solution is a hard problem: there is essentially no way to do it without trying every possible way of packing them. What most people do, then, is: put the largest items in first, then fit the smaller items into the spaces left around them. This will not necessarily be perfect packing, but it will usually give a packing that is pretty good. It is an example of a heuristic solution. Below, we present several widely used heuristic search algorithms.

2.2.3.3. Simulated Annealing

Kirkpatrick *et al.* [41] and Cerný [42] independently described a heuristic search algorithm known as *simulated annealing.* This terminology takes its origin from a technique used in metallurgy and referred to as annealing. This technique which consists in heating and cooling a material in a controlled environment, aims at reducing the defects of that material. By heating a material, its atoms can randomly move from their initial positions (which could be local minima of the internal energy) to positions of higher energy. By slowly cooling the material in a controlled environment, the atoms will probably stand a better chance of finding a position which will lower the internal

energy of the material. By analogy, at each iteration, the simulated annealing heuristic considers some neighbor s' of the current state s, and probabilistically decides between moving the system to state s' or staying in state s. The probabilities which depend on a global parameter T (called temperature) are chosen so that the system ultimately tends to move to states of lower energy. Typically this step is repeated until the system reaches a state that is good enough for the application, or until a given computation budget has been exhausted. Simulated annealing is used in a well known stochastic neural network classifier, the *Boltzmann Machine* [43, 44].

2.2.3.4. Genetic Algorithms

Genetic algorithms are another type of heuristic search. As mentioned in section 2.2.1, genetic algorithms are inspired by the theory of evolution [29, 45]. Genetic algorithms start with a population of abstract representations (called chromosomes or the genotype of the genome) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem and evolve toward better solutions. The evolution happens in several generations. Algorithm 2.1 presents a simplified genetic algorithm.

Algorithm 2.1: A simple genetic algorithm

1. Choose initial population
2. Evaluate the fitness of each individual in the population
3. Repeat until termination: (time limit or sufficient fitness achieved)
   - 3.1. Select best-ranking individuals to reproduce
   - 3.2. Breed new generation through crossover and/or mutation (genetic operations) and give birth to offspring
   - 3.3. Evaluate the individual fitnesses of the offspring
   - 3.4. Replace worst ranked part of population with offspring

Traditionally, solutions of genetic algorithms are represented in binary as strings of 0's and 1's, but other encodings are also possible. As we can observe in the algorithm, crossing individuals of a generation is encouraged. The hypothesis behind is that mating of two "good" individuals causes the mixing of their genomes, which may result in a

"better" offspring. As mentioned in section 2.2.1, Yang and Honavar [29] took advantage of a genetic algorithm for feature selection purposes.

2.2.3.5. Hill Climbing

Hill climbing algorithms try to maximize or minimize a function $f(v)$ of discrete states v . In general, these states are represented by vertices in a graph, where edges in the graph encode nearness or similarity. Hill climbing will follow the graph from vertex to vertex, always locally increasing (or decreasing) the value of f, until a local maximum (or local minimum) x_m is reached. Hill climbing can also operate on a continuous space: in that case, the algorithm is called gradient ascent (or gradient descent if the function is minimized).

Algorithm 2.2: Hill climbing algorithm

1. Let $v \leftarrow$ initial state.
2. Expand v: apply all operators to v, giving v's children.
3. Apply the evaluation function f to each child w of v.
4. Let v' = the child w with highest evaluation $f(w)$.
5. If $f(v') > f(v)$ then $v \leftarrow v'$; goto 2.
6. Return v. // since no better child of v exists

Algorithm 2.2 above [20] shows how hill climbing operates by starting with an initial set of feature(s) (empty set of features in the case of forward selection, set of all features in the case of backward elimination). Then from that state, the algorithm computes all possible combinations of features and keeps only the one which maximizes the evaluation function f. The process is repeated until no configuration increases substantially the evaluation function f. A known weakness of hill climbing is the fact that the algorithm can get stuck at a local maximum (or minimum) instead of the target global maximum (or minimum).

2.2.3.6. Best-First

Best-first search is a search algorithm which explores a graph by expanding the most promising node chosen according to some heuristic evaluation rule. Kohavi and John [20] conducted an evaluative comparison between hill climbing and best-first search algorithms and concluded that the latter was generally a more thorough technique. Algorithm 2.3 describes the principle of best search heuristic optimization.

Algorithm 2.3: Best-first algorithm

1. Begin with the OPEN list containing the start state, the CLOSED list empty, and BEST ← start state.
2. Let s = arg max $e(x)$ (get the state from OPEN with the highest evaluation).
3. Remove s from OPEN and add to CLOSED.
4. If $e(s) \geq e$(BEST); then BEST ← s.
5. For each child t of s that is not in the OPEN or CLOSED list, evaluate and add to OPEN.
6. If BEST changed in the last p expansions, goto 2.
7. Return BEST.

In Algorithm 2.3 above, the function e(.) is the heuristic evaluation of the feature subset and p is an integer to be fixed. The heuristic evaluation used depends on the goal of the application. Nonetheless, the idea is always to select the subset which maximizes the heuristic evaluation function. Best search algorithm was also used in Correlation-based Feature Selection [30, 31] and in the wrapper feature selector developed by Kohavi and John [20].

2.3 Machine Learning Techniques

Machine Learning is the study of computer algorithms that improve automatically through experience, relative to some performance metric [46]. Applications range from data mining programs that discover general rules in large data sets, to information filtering systems that automatically learn users' interests. Michalski and Stepp [47] identified three principal dimensions along with machine learning, namely the underlying learning strategies used, the representation of knowledge acquired by the system and the application domain of the system. Note that the application domain of the

system is not limited to any area of intellectual activity such as chemistry or chess [24], but can be applied to any such area. In this particular case, the underlying applications that these intellectual activities address all involve classification. We will start by introducing two major learning paradigms before presenting several learning strategies.

2.3.1 Learning Paradigms

We will consider two major paradigms of learning: *learning with a teacher*, and *learning without a teacher*. In conceptual terms, we may think of the teacher as having knowledge of the environment, with that knowledge being represented by a set of input-output examples [48]. *Learning with a teacher* is also known as *supervised learning* and *learning without a teacher* can be broken into *reinforcement learning* and *unsupervised learning*.

2.3.1.1. Supervised Learning

In order to learn a function, some machine learning techniques require to be trained from data samples represented by a combination of inputs and their corresponding desired outputs. These combinations of inputs and desired outputs are typically presented in the form of vectors. When such training is required by a learner, the technique is known as supervised learning. This expression originates from the fact that the learning process necessitates the supervision of a teacher which shows the desired output for every input from the training instances. The training should allow the learner to predict the output of the function when presented with any valid input. In order to succeed in the prediction of the function's output, the learner has to generalize from the presented data to unseen situations in a "reasonable" way [47]. The output of the function can either be continuous (regression) or a class label (classification). Figure 2.5 shows a block diagram that illustrates this type of learning [48].

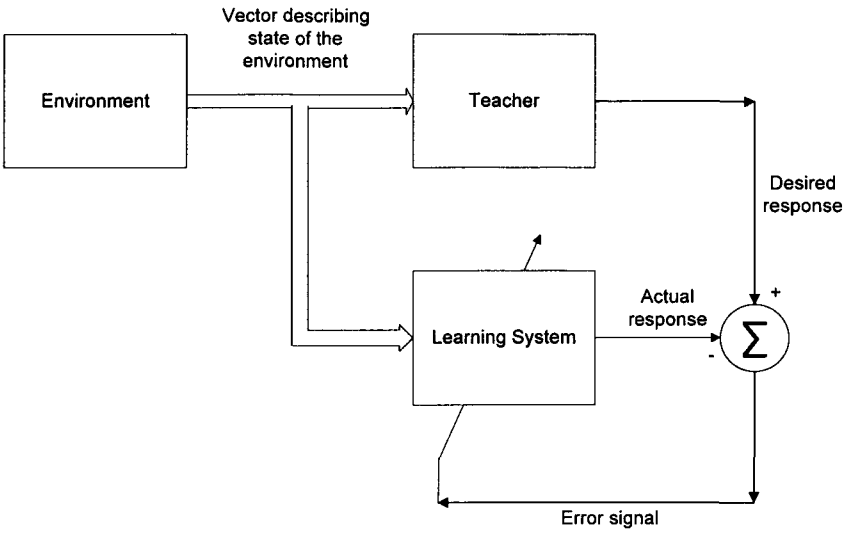


Figure 2.5. Block diagram of supervised learning

2.3.1.2. Reinforcement Learning

In reinforcement learning, the learning agent tries to discover a policy that maps the states of its environment to actions it should take in response to these states [49]. From Figure 2.6 which schematizes reinforcement learning, clear differences with supervised learning (Figure 2.5) can be observed. One major distinction is the absence of a teacher in reinforcement learning. Only input vectors describing the state of the environment are presented, but never the corresponding desired outputs. Because there is no knowledge of the desired output, the actions of the learner are never corrected. Instead, a numerical reward is given after each action. The goal of the agent is to maximize the total reward it receives over time. Reinforcement learning algorithms are highly related to dynamic programming techniques because they put a specific emphasis on online performance where the learning agent simultaneously tries to take advantage of the acquired knowledge so far and to learn from the current situation.

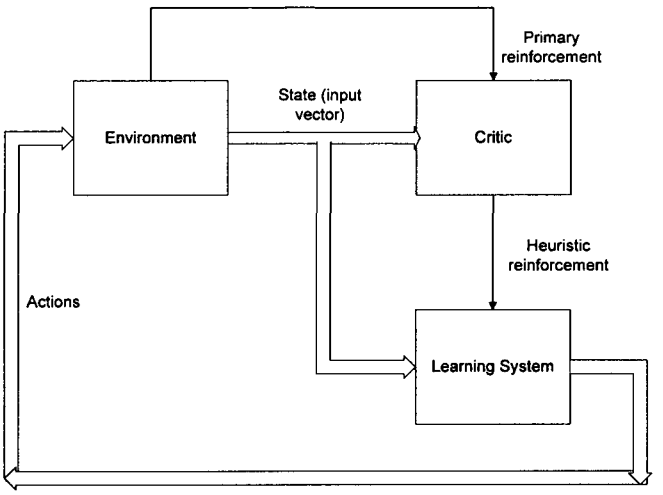


Figure 2.6. Block diagram of reinforcement learning

2.3.1.3. Unsupervised Learning

In unsupervised machine learning, the learning agent attempts to find out how the data is structured. As indicated in Figure 2.7, there is no external teacher or critic to oversee the learning process [48].

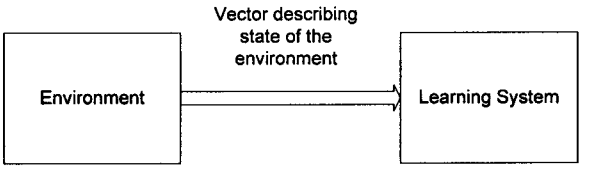


Figure 2.7. Block diagram of unsupervised learning

Principal Component Analysis (PCA) described in section 2.2.1 uses unsupervised learning in order to accomplish feature space reduction given a training dataset.

2.3.2 Learning Strategies

This subsection portrays three fundamentally different learning approaches, namely Bayesian learning, decision tree learning, and neural networks.

2.3.2.1. Bayesian Learning

As the name implies, Bayesian learning is based on applying Bayes' theorem (probability theory) where probability is used to represent uncertainty about the relationship being learned. During the learning phase, Bayesian learners build a set of probability distributions from training samples in an attempt to represent the relationships between the data input and the desired corresponding output. Those probability distributions can be viewed as the knowledge acquired prior to being exposed to the real unseen data. After seeing the data sample to be evaluated, the Bayesian learner computes a set of posterior probabilities that it combines with its prior knowledge in order to reduce the probability of an inconsistent hypothesis. This gives the Bayesian learning a high flexibility. The Bayesian learning algorithms combine training data with *a priori* knowledge to get the *a posteriori* probability of a hypothesis. It is therefore possible in theory to figure out the most probable hypothesis according to the training data. The basis for Bayesian learning algorithms is the Bayes rule:

$$p(h|D) = \frac{p(D|h)\,p(h)}{p(D)} \tag{2.1}$$

where $p(h)$, $p(D)$, $p(h|D)$ and $p(D|h)$ are respectively the prior probability of hypothesis h, the prior probability of training data D, the probability of h given D, and the probability of D given h.

As stated by John and Langley [50], Bayesian learning algorithms deal explicitly with issues of uncertainty and noise. Their predictive distribution tells you how uncertain

the prediction is. Bayesian classifiers have been widely used for several purposes such as medical diagnosis [46, 47, 50] and have been shown to be competitive with much more sophisticated induction algorithms [51]. Mitchell [46] went as far as stating that Bayesian methods provide "good standards" for evaluating other learning algorithms. One of these algorithms, the Naïve Bayes Algorithm, can be viewed as the simplest yet powerful Bayesian classifier and has been used in several classification problems [20, 32, 52, 53]. We will evaluate Naïve Bayes' fitness as a classifier for our experimental vision-based food inspection system.

2.3.2.2. Decision Tree Learning

In machine learning, a decision tree is a predictive model which maps observations about an item to conclusions about the item's target value. Such tree models are commonly referred to as *classification trees* when their outcome is discrete and *regression trees* when their outcome is continuous. In these tree structures, leaves represent classifications and branches represent conjunctions of features that lead to those classifications. The machine learning technique for inducing a decision tree from data is called decision tree learning. Figure 2.8 illustrates a simple decision tree for playing tennis. In this example adapted from [46], if the "Outlook" is *sunny* and "Humidity" is *high*, or if the "Outlook" is *rainy* and there is a *strong* "Wind", then it is not recommended to play outdoor tennis.

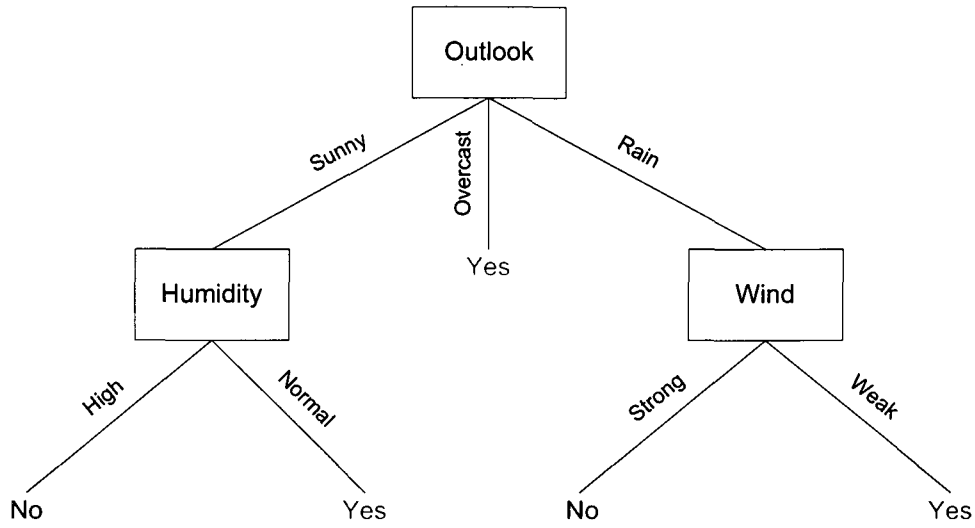


Figure 2.8. Example of a simple decision tree (adapted from [46])

Decision trees are widely used for classification purposes, for instance in equipment or medical diagnosis, or credit risk analysis [46]. Because decision tree inducers typically form their decision tree from a subset of the available attributes, some researchers view decision trees also as feature selectors. ID3 [23, 24] and FRINGE [25] presented in section 2.2.1 are examples of decision tree inducers. C4.5 [54] is also a decision tree inducer which builds its decision tree top-down by recursively finding the best single feature test to conduct at the root node of the tree. C4.5 has also proven itself to be one of the most competitive decision tree inducers in the world of machine learning for various domains of application [20, 32]. For this reason, we will evaluate the eventual benefits of C4.5 decision tree induction for classifying products inspected in the context of vision-based food inspection. A more recent approach, known as *random forest* [55], builds a combination of decision trees. Generally, the number of features M to be used (but not the features themselves) to determine the decision at a node of the trees is specified (typically much less than the total number of available features). At each node, the M features are randomly selected and the best split is computed based on those M features. Although randomly selecting the M features at a node reduces the computation during training (per opposition to selecting the best splitting feature based on information

such as entropy), random forest generally postpones this computation to evaluation time as the trees might be relatively large.

2.3.2.3. Artificial Neural Networks

An Artificial Neural Network (ANN), widely known as a Neural Network (NN), is a mathematical model whose rationale is based on biological neural networks. An artificial neural network tries to imitate the structural and functional aspects of biological neural networks. The simulated structure is composed of interconnected artificial neurons. These artificial neurons, also known as nodes, can be viewed as simple processing elements. The biological functional aspects simulated by this network of neurons are the flow of information through the network and the processing of information by the artificial neurons. Another functional aspect inspired from biological neural networks is the adaptive characteristic of the majority of artificial neural networks to modify their structure during the learning phase according to the external or internal information flowing in the network. Artificial neural networks can be used to model complex relationships between inputs and outputs or to find patterns in data. Figure 2.9 below illustrates a simple neural network containing one input layer (*i,* with two green nodes), one hidden layer (*j,* with four blue nodes) and one output layer (*k,* with one purple node). The nodes are interconnected with weighted connections. In their most widely used configuration, the value of a node N is computed using a nonlinear function of the weighted sum of all the nodes M_p contributing into that node. Every interconnection between two nodes has a weight value w_{nm} where m usually identifies the neuron on the layer from which originates the interconnection, and n represents the neuron on the layer receiving the interconnection. Artificial neural networks have been widely used in applications such as image processing, robotics, and data mining, as well as for food products inspection [6, 8, 10, 11, 13, 14].

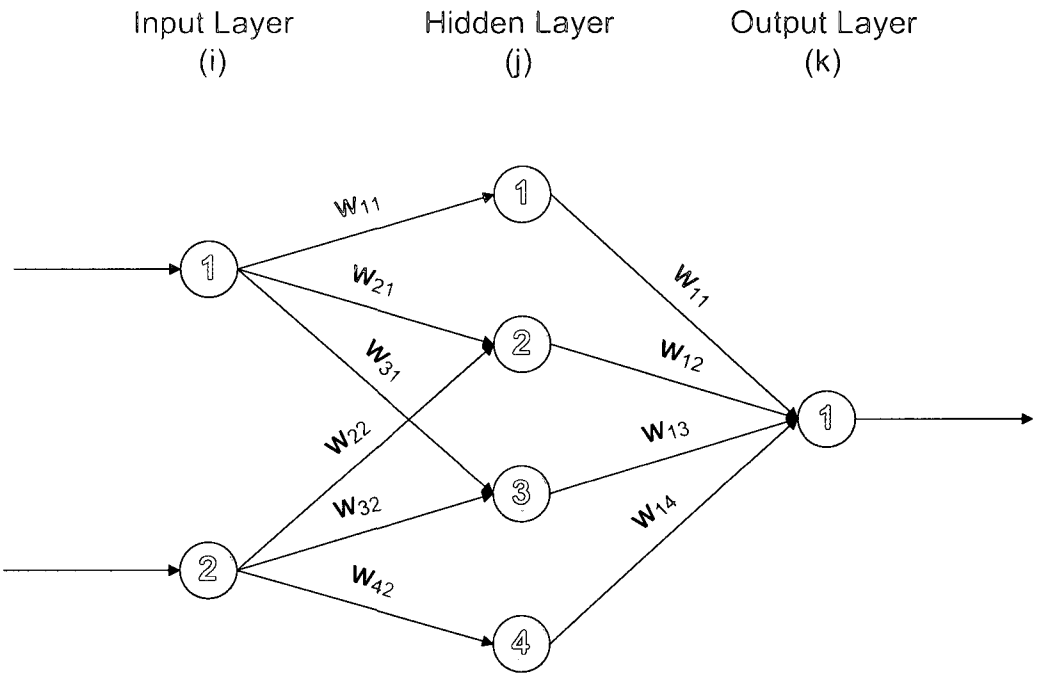


Figure 2.9. Example of a simplified neural network

A more detailed explanation of neural networks can be found in [48]. In most machine learning tasks involving numeric attributes, the Multi-Layer Perceptron (MLP) has attracted a lot of attention by its ability to capture complex nonlinear input-output relationships. In the category of artificial neural networks, we will also focus our attention on the multi-layer perceptron as a potential candidate classifier for our experimental vision-based food inspection system.

2.4 Chapter Summary

This chapter presented several fundamentally different machine learning techniques used in a wide variety of applications. We have pointed out that very often a machine learning technique is arbitrarily chosen for a given application, based on generic knowledge of the learning scheme's performance. This means for instance that a certain learning technique could be chosen for fish quality inspection because it gave great results on bovine meat inspection or vice versa, or that a neural network can be used

because this technique is known to capture complex nonlinear input-output relationships. This approach seems plausible. But what if on top of these assumptions we can actually benchmark several known machine learning techniques, and decide to adopt a particular scheme not only because it is generally known to perform well or because it performed well in similar situations, but because we have formally evaluated it to be the best candidate for a given product tied to a specific inspection system?

Machine vision systems in general, and particularly those for inspecting processed food, tend to extract a large number of features. This large number of features to process complicates the learning process for classification purposes and slows down the rate of production. Consequently, objective feature selection becomes particularly attractive when the number of features extracted is relatively large. We have seen that exhaustive search of features is generally not practically feasible and therefore heuristic feature selectors seem to be the first choice. We have also depicted filter-based feature selectors that operate independently of any learning algorithm, and wrapper-based feature selectors which select attributes based on a target learning algorithm.

We attempted introducing the necessary background for understanding the problem we are facing and the approach we ought to take in order to efficiently tackle the problem of allowing a food product inspection system to auto-tune a classifier and a combination of acceptable ranges of the analyzed features according to the product under inspection. Let us remind that our major goal is to define an appropriate process which will enable a vision-based food inspection system to automatically calibrate itself for classifying a new product. This same goal led us investigating if feature selection in conjunction with machine learning algorithms would be beneficial. One learning algorithm from each of the three learning schemes that we have presented in section 2.3.2 will be evaluated as a potential classifier for an available vision-based food inspection system which served as the experimental test framework. This experimental framework will be described in section 4.1. The candidate classifiers are: the Naïve Bayes classifier in the category of Bayesian classifiers, C4.5 in the group of decision tree inducers, and the multi-layer perceptron in the family of artificial neural networks. Consequently, three wrapper

feature selectors will also be evaluated: Naïve Bayes wrapper, C4.5 wrapper, and the MLP wrapper. Three fundamentally different and yet excellent filter-based feature selectors will also be evaluated for each of the learning algorithms: correlation-based feature selection (CFS), consistency-based feature selection and the RELIEF feature selection. The evaluated machine learning techniques and feature selection strategies will be further developed in Chapter 3.

Chapter 3. Machine Learning and Feature Selection

This chapter describes the three machine learning schemes and the four feature selection techniques evaluated in this work. The first subsection focuses on machine learning schemes while the second subsection concentrates on feature selectors. The last subsection will recapitulate the content of this chapter.

3.1 Evaluated Machine Learning Schemes

Chapter 2 presented machine learning as a viable candidate technique to evaluate for use with vision-based food inspection systems. This subsection respectively portrays the Naïve Bayes classifier in the category of Bayesian classifiers, the C4.5 in the group of decision tree inducers, and the multi-layer perceptron in the family of artificial neural networks.

3.1.1 Naïve Bayes Classifier

The Naïve Bayes Classifier is a Bayesian learning scheme particularly suited for problems dealing with a high dimensionality inputs. In order to understand the principle of Naïve Bayes, let's consider a simple classification task of objects which can be either red or green as illustrated in Figure 3.1.

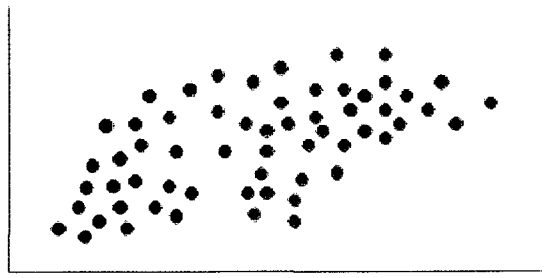


Figure 3.1. Naïve Bayes: computing prior probabilities

In the example above, there are 20 red objects and 40 green objects, and we would like to decide the class label (red or green) of new objects as they arrive, based on the known 60 samples. Since there are twice as many green objects as red, it is reasonable to believe a priori that a new unknown object is twice as likely to have membership green rather than red. In the Bayesian world, this belief is known as *prior probability* and is based on previous experience. In this example, the percentages of green and red objects are the prior probabilities and are often used to predict outcomes before they actually happen. Hence, we can write:

$$Prior\ ProbabilityOfGREEN \propto \frac{NumberOfGreenObjects}{TotalNumberOfObjects} = \frac{40}{60} = \frac{2}{3} \tag{3.1}$$

$$Prior\ ProbabilityOfRED \propto \frac{NumberOf\ RedObjects}{TotalNumberOfObjects} = \frac{20}{60} = \frac{1}{3} \tag{3.2}$$

since there are 40 green objects and 20 red objects. $\propto$ represents the proportionality operator.

To classify a new unknown object X (white object in Figure 3.2), it is reasonable to assume that the more green (or red) objects in the vicinity of X, the more likely that the new case belongs to that particular color if the objects are well clustered.

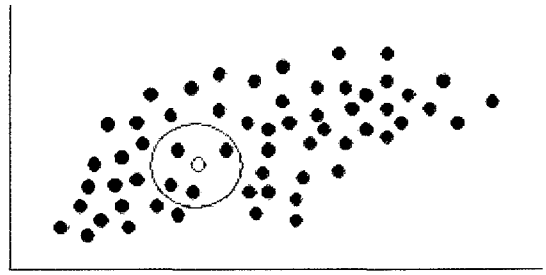


Figure 3.2. Naïve Bayes: computing posterior probabilities

This likelihood is measured by drawing around X a certain predefined area which encompasses a number of points irrespective of their class labels:

$$LikelyhoodOfXGivenGREEN \propto \frac{NumberOfGreenInTheVicinityOfX}{TotalNumberOfGreenObjects} = \frac{1}{40} \tag{3.3}$$

$$LikelyhoodOfXGivenRED \propto \frac{NumberOf\ RedInTheVicinityOfX}{TotalNumberOf\ RedObjects} = \frac{3}{20} \tag{3.4}$$

The final classification is produced by combining the prior probability and the likelihood to form a posterior probability using Bayes' rule:

$$\begin{aligned} Posterior\ ProbabilityOfXbeingGREEN \quad &\propto \quad Prior\ ProbabilityOfGREEN \\ &\times \quad LikelyhoodOfXGivenGREEN \end{aligned} \tag{3.5}$$

$$Posterior\ ProbabilityOfXbeingGREEN \quad \propto \quad \frac{2}{3} \times \frac{1}{40} = \frac{1}{60}$$

$$\begin{aligned} Posterior\ ProbabilityOfXbeingRED \quad &\propto \quad Prior\ ProbabilityOfRED \\ &\times \quad LikelyhoodOfXGivenRED \end{aligned} \tag{3.6}$$

$$Posterior\ ProbabilityOfXbeingRED \propto \frac{1}{3} \times \frac{3}{20} = \frac{1}{20}$$

The unknown object X is therefore classified as RED since its class membership achieves the largest posterior probability.

Naïve Bayes can deal with arbitrary number of independent variables whether continuous or categorical. Formally speaking, given a set of variables $X = \{x_1, x_2, ..., x_k\}$, the conditional probability for the event C_j among a set of possible outcomes $C = \{c_1, c_2, ..., c_m\}$ is computed using Bayes' rule as follows:

$$p(C_j|X) = p(C_j|x_1, x_2, ..., x_k) = \frac{p(C_j)p(x_1, x_2, ..., x_k|C_j)}{p(x_1, x_2, ..., x_k)} \tag{3.7}$$

Generally, the probability distribution in the denominator of Equation (3.7) is not directly estimated as it is simply a normalizing factor. Instead, one ignores the denominator and then normalizes so that the sum $p(C_j|x_1, x_2, ..., x_k)$ over all classes is one [50]. Assuming attributes to be conditionally independent we can rewrite $p(x_1, x_2, ..., x_k|C_j)$ as:

$$p(x_1, x_2, ..., x_k|C_j) = \prod_{i=1}^{k} p(x_i|C_j) \tag{3.8}$$

Inserting Equation (3.8) into Equation (3.7), the conditional probability is computed in a straightforward manner:

$$p(C_j|X) = p(C_j|x_1, x_2, ..., x_k) \propto p(C_j)\prod_{i=1}^{k} p(x_i|C_j) \tag{3.9}$$

Equation (3.9) is evaluated for every class C_j and the most probable class is assigned to X. Numeric attributes are modeled in several ways including normal (also known as Gaussian), lognormal, gamma, and Poisson probability density functions [56]. For experiments on the available vision-based food inspection system, we used a common choice for Naïve Bayes, the Normal distribution, which can be represented in terms of its mean and standard deviation. Therefore for continuous attributes and assuming a normal distribution, we can efficiently compute the probability of an observed value from such estimates:

$$p(X = x|C_j) = g(x, \mu, \sigma) \tag{3.10}$$

where σ is the standard deviation, μ the mean (or expected value) and the Gaussian probability density function *g(.)* is given by:

$$g(x,\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.11}$$

The presented model leaves us with a small set of parameters to estimate from training data, therefore justifying the choice of the Gaussian distribution for computing probabilities. For each class and nominal attribute, one must estimate the probability that attributes will take on each value in its domain, given the class. For each class and continuous attribute, one must estimate the mean and standard deviation of the attribute given the class. The probabilities for nominal features are estimated using frequency counts calculated from the training data. Given a dataset of *k* features, *n* training instances and *m* test instances, the time complexity of Naïve Bayes is $O(nk)$ for training and $O(mk)$ for testing.

To clarify the formal estimation process used in the experiments presented here, consider a small dataset in which there are two classes (+ and −), a nominal attribute X_1 which can take value *a* or value *b*, and a continuous attribute X_2 which can take any real value.

{(+, *a,* 1.0), (+, *b*, 1.5), (+, *a*, 2.9), (−, b, 3.2), (−, *b*, 3.5), (−, *b*, 3.7), (−, *a*, 6.3)},

Given the example above of seven training samples, Naïve Bayes will compute the following prior probabilities for each of the two possible classes (+ and −):

$$p(C=+)=3/7$$
$$p(C=-)=4/7$$

because three out of the seven training samples belong to the positive class (+) and the remaining four out seven samples belong to the negative class (−). For any new sample to be classified, Naïve Bayes will compute the two following conditional probability evaluations for all possible class labels assuming the attributes X_1 and X_2 to be conditionally independent. For the positive class:

$$p(X_1 = x_1 | C = +) = \begin{cases} 2/3 & if \quad x_1 = a \\ 1/3 & if \quad x_1 = b \end{cases}$$

because among the seven training samples, two out of the three samples belonging to the positive class $C = +$ satisfy $X_1 = a$ and one out of the three samples belonging to the positive class $C = +$ satisfies $X_1 = b$.

For the continuous attribute X_2, the following conditional probability will be estimated:

$$p(X_2 = x_2 | C = +) = g(x_2, \mu, \sigma)$$

where $g(.)$ is described by Equation (3.11); and given the three training samples belonging to the positive class (+), the expected value $\mu = \frac{1}{3}(1.0 + 1.5 + 2.9) = 1.80$ and the standard deviation $\sigma = \sqrt{\frac{1}{(3-1)}\left[(1.0-1.80)^2 + (1.5-1.80)^2 + (2.9-1.80)^2\right]} = 0.98$.

Note that the *sample standard deviation* is used for σ per opposition to the *standard deviation of the sample* (the use of *N-1* in the denominator of σ instead of the number of samples *N*). This correction, known as *Bessel's correction,* aims to correct the bias in the estimation of the variance, and some of the bias in the estimation of the standard deviation as the real population's mean is unknown and can only be estimated from samples.

The probability estimates for the negative class can be obtained similarly to those of the positive class:

$$p(X_1 = x_1 | C = -) = \begin{cases} 1/4 & if \quad x_1 = a \\ 3/4 & if \quad x_1 = b \end{cases}$$

and:

$$p(X_2 = x_2 | C = -) = g(x_2, \mu, \sigma)$$

where, given the four training samples belonging to the negative class (−), the expected value $\mu = \frac{1}{4}(3.2 + 3.5 + 3.7 + 6.3) = 4.18$ and the standard deviation $\sigma = \sqrt{\frac{1}{(4-1)}\left[(3.2-4.18)^2 + (3.5-4.18)^2 + (3.7-4.18)^2 + (6.3-4.18)^2\right]} = 1.43$. The function $g(.)$ is defined by Equation (3.11).

For any sample to be classified and having a nominal value x_1 for attribute X_1 and a continuous value x_2 for attribute X_2, Naïve Bayes will simply compute the posterior probabilities of each of the possible class labels:

$$p(C = +) = 3/7 * p(X_1 = x_1 | C = +) * p(X_2 = x_2 | C = +)$$
$$p(C = -) = 4/7 * p(X_1 = x_1 | C = -) * p(X_2 = x_2 | C = -)$$

The class achieving the maximum posterior probability is assigned to the new sample. We have achieved a java-based implementation of Naïve Bayes algorithm for the purpose of the work reported in this thesis.

3.1.2 C4.5 Decision Tree Inducer

C4.5 [54] is an algorithm that summarizes training data in the form of a decision tree (Figure 2.8). Nodes in the tree correspond to features, and branches to their associated values. The leaves of the tree correspond to classes. To classify a new instance, one simply examines the features tested at the nodes of the tree and follows the branches corresponding to their observed values in the instance. Upon reaching a leaf, the process terminates, and the class at the leaf is assigned to the instance.

In order to build its decision tree from training data, C4.5 applies a *greedy search* approach to select the next best attribute which will split best the training instances. Greedy search algorithms select the locally optimal choice at each stage with the hope of finding the global optimum. Greedy search will lead to the optimal solution only if the problem to solve exhibits an *optimal substructure.* A problem is said to exhibit optimal substructure if the optimal solution to the problem contains optimal solutions to the sub-problems [57]. C4.5 uses *information gain* defined in Equation (3.14) as the heuristic measure to guide its greedy search. The process for computing *information* gain is as follows: Given two discrete random variables X and Y, the entropy of Y before observing X is given by:

$$H(Y) = -\sum_{y \in Y} p(y) \log_2 (p(y)) \tag{3.12}$$

After observing X, the entropy of Y will be computed as follows:

$$H(Y \mid X) = -\sum_{x \in X} p(x) \sum_{y \in Y} p(y \mid x) \log_2 (p(y \mid x)) \tag{3.13}$$

where $\log_2$ represents the logarithm of base two, $p(x)$ is the probability of the random variable X taking the value x, $p(y)$ is the probability of the random variable Y taking a value y, and $p(y|x)$ is the probability of the random variable Y taking a value y given that

the random variable X has a value of x (i.e. after observing X). In information theory, Shannon entropy or information entropy is a measure of the uncertainty associated with a random variable and represents the minimum message length (usually in bits/symbol) necessary to communicate information. In the entropy equations above, the amount by which the entropy of Y decreases reflects the additional information about Y provided by X and is called the *Information Gain* [54]. Information gain is given by:

$$\begin{aligned} gain &= H(Y) - H(Y \mid X) \\ &= H(X) - H(X \mid Y) \\ &= H(Y) + H(X) - H(X,Y) \end{aligned} \tag{3.14}$$

where $H(X,Y)$ is the entropy of the pairing of the two random variables X and Y, also known as the joint entropy of X and Y, and is given by:

$$H(X,Y) = -\sum_{x \in X} \sum_{y \in Y} p(x,y) log_2 \left(p(x,y) \right) \tag{3.15}$$

where $p(x, y)$ is the probability of the random variable formed by the pair (X, Y) to take value (x, y).

Starting at the root of the tree, C4.5 recursively divides the training instances into subsets by selecting as the next best node the attribute that maximizes its information gain divided by its entropy, terminating when a given subset contains only one class. During the iterations, selection operates only on attributes for which "information is gained", that is attributes for which the entropy of the class labels in the considered subsets is less than the entropy of the class labels in the full training set (or subset if not at the root of the tree). A node is replaced by its best leaf when the estimated error of the leaf is within one standard deviation of the estimated error of the node.

The version of C4.5 used in the experiments reported in this document is the original algorithm implemented by Quinlan [54].

3.1.3 Multi-Layer Perceptron (MLP)

One of the most widely used neural classifiers today is the Multilayer Perceptron (MLP) network. As mentioned in section 2.3.2.3, MLP networks are nonlinear models inspired from biological neural networks and consist of a number of units organized into multiple layers. These units are interchangeably called neurons, nodes, or processing units. The complexity of the MLP network can be changed by varying the number of layers and the number of units in each layer. Typically it requires three or more layers of processing nodes: an input layer which accepts the input variables used in the classification procedure, one or more hidden layers, and an output layer with one node per class as illustrated in Figure 3.3 below.

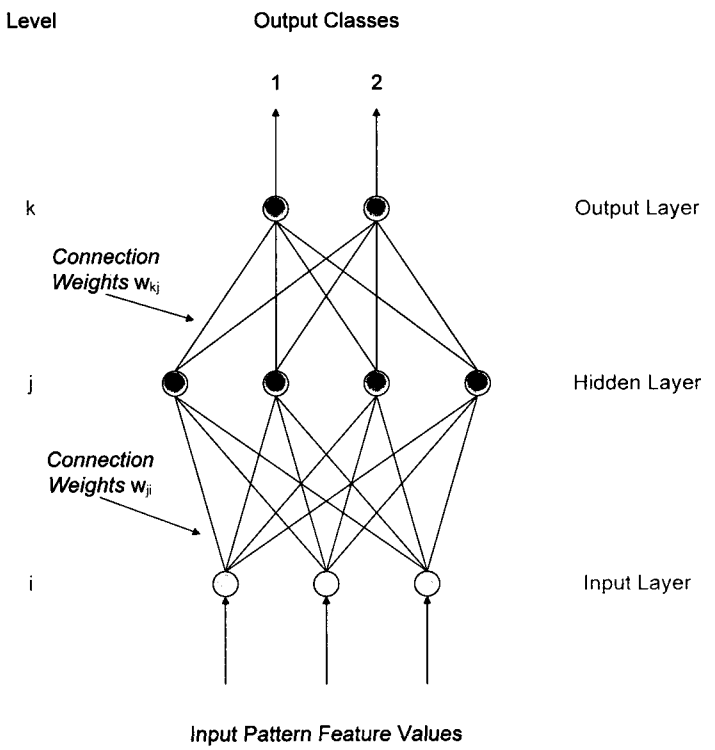


Figure 3.3. Architecture of multilayer perceptron

Each neuron (or node) is typically modeled by a *nonlinear* function. A commonly used form of nonlinearity is a *sigmoidal nonlinearity* defined by the *logistic function* [48]:

$$y_j = \varphi_j(v_j) = \frac{1}{1+e^{-v_j}} \tag{3.16}$$

where y_j is the output of neuron j and is a function $\varphi_j(.)$ of v_j; v_j being the induced local field of neuron j, that is the weighted sum of all the *(m+1) synaptic* inputs to neuron j as given by Equation (3.17). In Equation (3.17), w_{ji} denotes the synaptic weight connecting the output of neuron i to the input of neuron j:

$$v_j = \sum_{i=0}^{m} w_{ji} y_i \tag{3.17}$$

Figure 3.4 shows a plot of a standard sigmoid function $\varphi_j(v_j)$ as defined by Equation (3.16). This figure models the "S-curve" of growth of the set φ_j which could be thought of as a population [58]. The initial stage of growth is approximately exponential; then, as saturation begins, the growth slows down, and at maturity, growth stops.

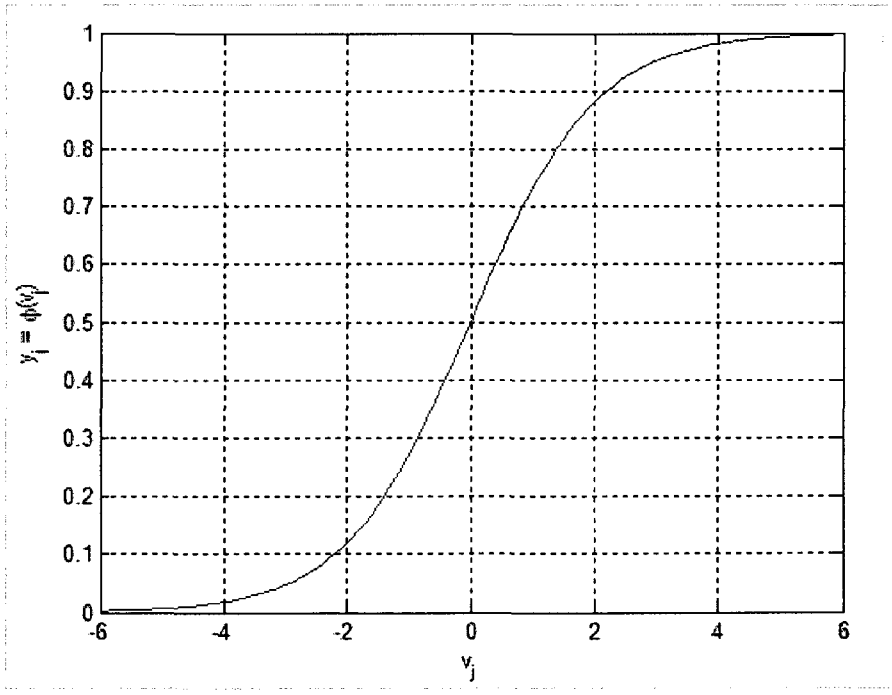


Figure 3.4. Standard logistic sigmoid function

MLP networks are trained using *error back-propagation algorithm* which essentially consists of two passes through the layers of the network: a forward pass and a backward pass. During the forward pass where the synaptic weights of the network are all fixed, an input vector is applied to the nodes of the input layer, and its effect propagates through the network layer by layer, finally producing a set of outputs as the actual response of the network. On the other hand, during the backward pass, the synaptic weights of the neurons are all adjusted using an error correction rule. During this phase, the error signal given by the difference between the desired (or target) response and the actual response produced by the network is used to update the synaptic weights in the aim of making the actual response of the network statistically closer to the desired response. Error back-propagation is often referred in the literature as back-propagation or simply back prop. During back-propagation, the correction of the synaptic weights for the n^{th} training pattern is computed using the demonstrable *generalized delta rule* given as follows [48]:

$$\begin{pmatrix} Weight \\ correction \\ \Delta w_{ji}(n) \end{pmatrix} = \begin{pmatrix} momentum \\ term \\ \alpha \Delta w_{ji}(n-1) \end{pmatrix} + \begin{pmatrix} learning \\ rate \\ \eta \end{pmatrix} . \begin{pmatrix} local \\ gradient \\ \delta_j(n) \end{pmatrix} . \begin{pmatrix} input & signal & of \\ neuron & j & \\ & y_i(n) & \end{pmatrix} \tag{3.18}$$

where w_{ji} is as described in Equation (3.17), and y_i is similarly defined in Equation (3.16) (for another layer *j*), and the positive *learning rate* η which typically lies between 0 and 1 controls the importance of the change while the *momentum term* adds a feedback loop in order to reduce risk of instability. The generally positive *momentum constant* α must satisfy $0 \le |\alpha| < 1$. For the *local gradient* $\delta_j(.)$ of neuron *j*, there are two situations to consider depending if the neuron *j* belongs to an output layer or to a hidden layer. For the former case, the local gradient for training pattern *n* is computed as:

$$\delta_j(n) = e_j(n)\varphi_j'(v_j(n)) \tag{3.19}$$

where v_j is as defined in Equation (3.17), $\phi_j'(v_j(n))$ is the derivative of the activation function $\varphi_j(v_j(n))$ associated to neuron j as defined in Equation (3.16), and $e_j(n)$ is the error signal at the output of neuron j at iteration n and is given by the difference between the known target output $d_j(n)$ and the function signal $y_j(n)$ appearing at the output of neuron j at iteration n:

$$e_j(n) = d_j(n) - y_j(n) \tag{3.20}$$

If on the other hand neuron j is located in a hidden layer, then there is no specified desired response for that neuron. In that case, the local gradient equals the product of the derivative of the activation function $\varphi_j(v_j(n))$ associated to neuron j as defined in Equation (3.16) and the weighted sum of the local gradients δ_k computed for the neurons in the next hidden or output layer that are connected to neuron j, as follows:

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n) \tag{3.21}$$

where w_{kj} denotes the synaptic weight connecting the output of neuron j to the input of neuron k. When $\alpha = 0$ in Equation (3.18), the equation is known as the *delta rule*. The new synaptic weight at iteration n is finally computed as:

$$w_{ji}(n) = w_{ji}(n-1) + \Delta w_{ji}(n) \tag{3.22}$$

where $\Delta w_{ji}(n)$ is given by Equation (3.18).

We have accomplished a java-based implementation of the MLP in order to evaluate the performance of this learning technique.

3.2 Evaluated Feature Selection Techniques

Many factors affect the success of machine learning on a given task. The representation and quality of the example data is first and foremost. Theoretically, having more features should result in more discriminating power. However, practical experience with machine learning algorithms has shown that this is not always the case. If there is too much irrelevant and redundant information present or the data is noisy and unreliable, then learning during the training phase is more difficult. Hence, identifying and removing as much irrelevant and redundant information as possible reduces the dimensionality of the data and may allow learning algorithms to operate faster and more effectively. In some cases, accuracy on future classification can be improved; in others, the result is a more compact, easily interpreted representation of the target concept. This section will present the implementation details of the four experimented feature selectors. Three of the attribute selectors are filter-based (RELIEF, correlation-based feature selection, and consistency-based feature selection) and the fourth is Kohavi and John's wrapper [20]. A java-based open source implementation of those four algorithms is available under GNU General Public Licence [59]. This open source implementation, known as the *weka* project is available in [60]. For this experimental evaluation, we used the Java-based source-code that we integrated to work with the machine learning techniques presented in section 3.1.

3.2.1 RELIEF Feature Selection

The RELIEF attribute selection technique uses general characteristics of the data to evaluate attributes and operates independently of any learning algorithm. RELIEF was first introduced by Kira and Rendell [34] as a means of estimating the "quality" of attributes with and without dependencies among them. RELIEF is an instance-based attribute ranking scheme that works by randomly sampling an instance from the data and then locating its nearest neighbors from the same and opposite classes. The neighbor from the same class is named *nearest hit* and the one from the opposite class is called

nearest miss. The values of the attributes of the nearest neighbors are compared to the sampled instance and used to update relevance scores for each attribute. As a matter of fact, RELIEF's estimate *W[A]* of attribute *A* is an approximation of the following difference of probabilities:

$$W[A] = p(\text{different value of } A \mid \text{nearest instance from different class}) - p(\text{different value of } A \mid \text{nearest instance from same class}) \quad (3.23)$$

where $p(X|Y)$ is the conditional probability of some event X, given the occurrence of some other event Y. The rationale of the RELIEF algorithm is that useful attributes should differentiate between instances from different classes and have the same value for instances from the same class.

The original version of RELIEF is limited to only two-class problems, which led Kononenko [36] to extend the original RELIEF to deal with noisy, incomplete, and multi-class datasets. The first enhancement that Kononenko addressed was to increase the reliability of probability approximation by searching the k-nearest hits/misses instead of only one near hit/miss, where k is a positive integer such that $k > 1$. The enhanced version, called RELIEF-F, finds nearest neighbors from each class different than the current sampled instance and averages their contribution for updating estimates *W[A]*, and finally weights the average with prior probability of each class as follows [32]:

Algorithm 3.1: RELIEF-F Algorithm

set all weights $W[A] := 0.0$;
for i := 1 to m do
begin
randomly select an instance R;
find k nearest hits H_j;
for each class C ≠ class(R) do
find k nearest misses $M_j(C)$
for A := 1 to NumberOfAttributes do

$$W[A] := W[A] - \sum_{j=1}^{k} diff(A, R, H_j)/(m \times k) +$$

$$\sum_{C \neq class(R)} \left[\frac{p(C)}{1 - p(class(R))} \sum_{j=1}^{k} diff(A, R, M_j(C)) \right] / (m \times k)$$

End

where *NumberOfAttributes* is the total number of attributes in the original dataset, and *diff(Attribute, Instance1, Instance2)* computes the difference between the values of *Attribute* for two instances. For discrete attributes, the difference is either 1 (the values are different) or 0 (the values are the same). For continuous attributes the difference is the actual difference normalized to the interval [0, 1]. The same function *diff(.)* is used for calculating the distance between instances to find the nearest neighbors. The total distance is simply computed as the sum of differences over all attributes. The expression "continuous attributes" refers here to features that can be measured on a continuum or scale, and can have almost any numeric value, as opposed to nominal data like good or bad, on or off, etc, or discrete data where the attribute can take a value within a limited set of values. In the algorithm of RELIEF-F above, m is a user specified number of instances. Kononenko [36] notes that the higher the value of m, the more reliable the approximation as m represents the number of instances for approximating probabilities. m can therefore not exceed the number of available training instances. But increasing the value of m comes at the expense of running time. In all experiments reported here, we set $m = 250$ and $k = 10$ as suggested in [36, 61] when the number of training instances is greater than or equal to 250. Furthermore, only attributes A with ranking $W[A]$ greater or equal to zero were considered relevant in our experiments. The reason for this choice is the fact that in computing the rank $W[A]$ of attribute A, for every randomly considered instance R belonging to class C, if an instance M_j is a nearest miss (i.e. neighbor of R

having a class different than the class of R), then the difference $diff\left(A,R,M_j(C)\right)$ should increase the score $W[A]$. On the other hand, if H_j is a nearest hit (i.e. neighbor of R having the same class as R), then the difference $diff\left(A,R,H_j\right)$ should decrease the rank $W[A]$. Basically, the bigger the difference between the value of attribute A and its nearest hits, the lower the score for attribute A (it means attribute A seems not to be consistent), and the higher the difference between the value of attribute A and its nearest misses, the higher the score of attribute A (it means attribute A seems to be consistent). The weight $W[A]$ being normalized to the interval [-1, 1], it is reasonable to consider only attributes with weights in the interval [0, 1] to be relevant (or at least neutral). Kononenko's experiments on both artificial and real world data showed RELIEF to be a good candidate for selecting relevant features out of imperfect data.

3.2.2 Correlation-based Feature Selection (CFS)

As mentioned in section 2.2.1, the Correlation-based Feature Selection (CFS) introduced by Hall [30, 31] is a filter-based attribute selector which evaluates subsets of attributes rather than individual attributes. Hall's rationale of this technique is based on the hypothesis that *"a good feature subset is one that contains features highly correlated with (predictive of) the class, yet uncorrelated with (not predictive of) each other"* [30]. The first part of this hypothesis is inspired by Gennari *et al.* [62] who stated that *"features are relevant if their values vary systematically with category membership"*. This statement has been formalized by Kohavi and John [20] who formulated that a feature V_i is said to be relevant for a class C if and only if there exists some v_i and c for which $p(V_i = v_i) > 0$ such that:

$$p\left(C = c \middle| V_i = v_i\right) \neq p\left(C = c\right) \tag{3.24}$$

where $p(X|Y)$ is the conditional probability of some event X, given the occurrence of some other event Y and is read "the probability of X, given Y".

In simple terms, the equation above implies that a feature is considered useful if it is correlated with (or predictive) of the class and irrelevant otherwise. The second part of the hypothesis is justified by the fact that empirical evidence encourages removing redundant information along with irrelevant features [20, 63, 64]. A feature is considered redundant if it is highly correlated with one or more other features. Hall compares CFS rationale to a principle in test theory [65] used to design a composite test (sum of average of individual tests) for predicting an external variable of interest. Hall gives the example stating that one can obtain a more accurate prediction of a person's success in a mechanics training course by a composite of several tests measuring a wide variety of aspects such as the person's ability to learn, ability to comprehend written material, manual dexterity and so forth, rather than any one individual test which measures a restricted scope of aspects. This subset evaluation heuristic, actually used in psychology [65, 66], is formalized by the following equation:

$$Merit_S = \frac{q\overline{r_{cf}}}{\sqrt{q + q(q-1)\overline{r_{ff}}}} \tag{3.25}$$

where $Merit_S$ is the heuristic "merit" of a feature subset S containing q features, $\overline{r_{cf}}$ is the average feature-class correlation, and $\overline{r_{ff}}$ is the average feature-feature correlation. The numerator can be interpreted as an indication of how predictive a group of features is, as for q fixed and greater than 1, the feature-class correlation average $\overline{r_{cf}}$ will be relatively large if the group of features is correlated with the class and small otherwise; and therefore the numerator theoretically allows discriminating irrelevant features. On the other hand, the denominator discriminates redundant features because in case of redundant attributes (respectively non redundant), the feature-feature correlation average $\overline{r_{ff}}$ will be large (small), which implies a larger (smaller) denominator, and therefore a smaller (larger) $Merit_S$. However, one can argue that increasing the number of features q may increase the feature-class correlation. Ghiselli [65] pointed out that it is unlikely that

a group of features which are highly correlated with the class will be at the same time in low correlation with each other. Another question one might ask is how to determine feature-class correlation and feature-feature correlation. To answer this question, an experimental comparative evaluation of three versions of CFS feature selector was conducted by Hall. The RELIEF estimation [34] introduced in section 3.2.1, the Minimum Description Length (MDL) principle [67, 68] and the Symmetrical Uncertainty (SU) [69] were tested and compared. This analysis showed that the version of CFS using symmetrical uncertainty performs the best overall. We will hence briefly introduce the principle of symmetrical uncertainty.

Symmetrical uncertainty is a modified "Information Gain" measure which was introduced in section 3.1.2 as:

$$\begin{aligned} gain &= H(Y) - H(Y \mid X) \\ &= H(X) - H(X \mid Y) \\ &= H(Y) + H(X) - H(X,Y) \end{aligned} \tag{3.26}$$

were $H(X)$ is the entropy of Y, and $H(X \mid Y)$ is the entropy of X after observing Y, $H(Y \mid X)$ is the entropy of Y after observing X, and $H(X,Y)$ is the joint entropy of the pair (X, Y). It can be seen from Equation (3.26) that the amount of information gained about Y after observing X is equal to the amount of information gained about X after observing Y; which implies the information gain to be symmetric. Unfortunately, information gain has been proven to favor features with more values [30, 31, 70]. In fact, attributes with greater number of values appear to gain more information than those with fewer values even if they are not more informative. Hall proposes a metric known as symmetrical uncertainty in order to compensate for information gain's bias toward features with more values and normalizes its value to range [0, 1] as described below:

$$\text{symmetrical uncertainty} = 2.0 \times \left[\frac{gain}{H(Y) + H(X)} \right] \tag{3.27}$$

The correlation between two features (r_{ff}) X and Y of a subset of features S is therefore computed using Equation (3.27). This same equation is used to compute the correlation between a class and a feature (r_{cf}). The average of r_{cf} and r_{ff} are afterwards used to compute the heuristic merit $Merit_S$ of the subset S of features as defined in Equation (3.25).

In Hall's work, experiments conducted on 36 datasets [30, 31] demonstrated that CFS reduces the number of features by more than half for 70% of the discrete class datasets, while also enhancing accuracy on most of the datasets for various machine learning algorithms. The time complexity of CFS is $O\left(\frac{M\left(N^2-N\right)}{2}\right)$ for M training instances of a dataset containing N attributes. Hall recommended a best first algorithm described in section 2.2.3.6 as the heuristic feature search engine for CFS.

3.2.3 Consistency-based Feature Selection

In consistency-based feature selection, training instances are projected onto the subset of attributes and then the consistency of the subset is evaluated. It is therefore common practice to use this filter-based feature selector in conjunction with a search algorithm that looks for the smallest subset with consistency. Liu and Setiono proposed an inconsistency evaluation criterion in [33]. Two instances are considered inconsistent if they match except for their class labels. The inconsistency rate of a dataset is computed as follows:

- Two instances are considered inconsistent if they match except for their class labels
- Suppose there are r possible class labels $label_1$, $label_2$, ..., $label_r$ in a certain dataset which contains N instances

- Suppose there are J distinct combinations of attribute values for a subset s of attributes (without considering the class labels of the instances)
- Suppose that D_i is the number of occurrences (or matching instances without considering the class labels of the instances) of the i^{th} combination of attribute values
- Suppose that among the D_i instances, c_1 instances belong to class label $label_1$, c_2 belong to $label_2$, ..., and c_r instances belong to class label $label_r$ such that $c_1 + c_2 + \ldots + c_r = D_i$ and let $M_i = \max\{ c_1 , c_2 , \ldots , c_r \}$

Given the definitions and hypotheses above, the *inconsistency count* is given by:

$$inconsistency\ count = (D_i - M_i) \tag{3.28}$$

This equation shows that the inconsistency count decreases when M_i increases and increases otherwise. In other words, for matching instances (without considering the class labels of the instances) in a subset S of attributes, the more the class labels match, the less inconsistent (or the more consistent) is the subset S with respect to the class. The inconsistency rate of an attribute subset S is given by the sum of all inconsistency counts divided by the total number of instances:

$$inconsistency_s = \frac{\sum_{i=1}^{J} D_i - M_i}{N} \tag{3.29}$$

where all the variables are as defined above. Several search strategies can be used to look for the smallest most consistent subset of attributes. The Best-first algorithm described in section 2.2.3.6 was used as the heuristic feature search engine for consistency-based feature selection for its high performance exhibited in [20]. Experiments [31] showed consistency-based feature selection to be able to remove up to 92% of the attributes of some datasets while still improving the accuracy of classification. Moreover, the consistency criterion is relatively easy to implement in the available food inspection system's framework.

3.2.4 Wrapper-based Feature Selection

The wrapper feature selector introduced in section 2.2.2 repeatedly searches for a relevant feature subset by using the induction algorithm as part of the evaluation function. Typically, the induction algorithm is a machine learning algorithm such as the ones discussed previously. In [20], Kohavi and John suggest using *five-fold cross validation* accuracy evaluation function for feature selection. This accuracy evaluation function is repeated multiple times. In t-fold cross-validation, where t is a positive integer, the data is randomly split into t mutually exclusive subsets (the folds) of approximately equal size as illustrated in Figure 3.5 for $t = 3$.

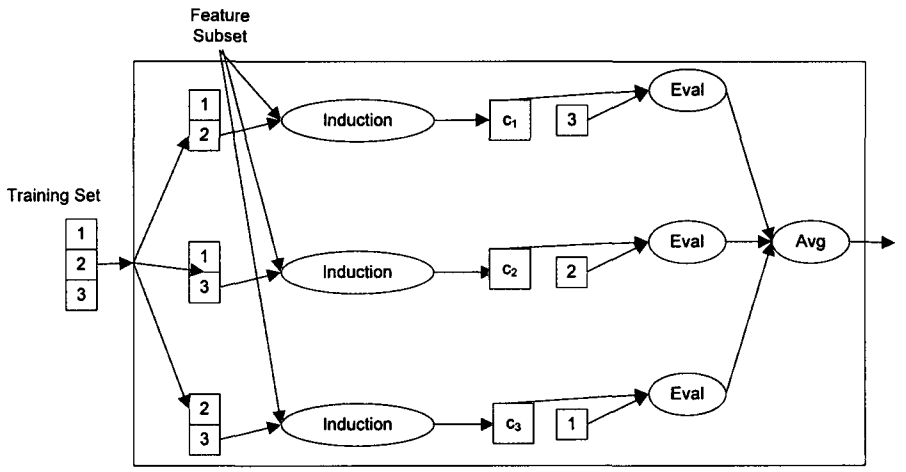


Figure 3.5. Example of *3*-fold cross-validation

The learner is trained and tested t times, each time with (t-1) training folds and the remaining one stands as the test fold. For instance for $t=3$ as illustrated in Figure 3.5, the training set {1, 2, 3} is divided into three training set/test set groups {1, 2}/{3}, {1, 3}/{2}, and {2, 3}/{1}. The induction algorithm is then trained three times with each of the three training sets {1, 2}, {1, 3} and {2, 3}, but only using a subset of the features. The feature subset is proposed by the feature search engine. Each one of the three training sets produces a certain classifier represented by "c_i" in Figure 3.5, where i is an integer value such that $1 \leq i \leq 3$. Each classifier "c_i" is tested using the remaining test set: the classifier built from training set {1, 2} is tested using test set {3}, the classifier from

{1, 3} tested using {2}, and the classifier from {2, 3} tested using {1}. The overall accuracy is then averaged over the *t* folds and if the standard deviation of the accuracy estimate is above 1% and a certain fixed number of iterations of the cross validations has not been executed; then another cross-validation is run. According to Kohavi and John [20], this heuristic seems to work well in practice and avoids multiple cross-validations for large datasets. The heuristic also forces accuracy estimation to execute cross-validation more times on small datasets than on large datasets. More cross-validation runs on small datasets will not increase the accuracy estimation time considerably because small datasets require less time to learn. The accuracy estimation time is the product of the induction algorithm's running time and the cross-validation time. A complexity penalty was added to the evaluation function in such a way that the smaller of two feature subsets that have the same estimated accuracy is always picked. More details on cross-validation can be found in [71].

3.3 Chapter Summary

This chapter presented three fundamentally different machine learning techniques and some of their implementation level details: the Naïve Bayes, the C4.5 and the MLP. Three filter-based and a wrapper feature selection techniques were also presented from the perspective of machine learning. This chapter also exhibited the integration of open source code with custom built software in order to prepare the evaluation of the impact of selected feature selection techniques on the candidate machine learning algorithms for vision-based food inspection.

Chapter 4. Experimental System and Dimensionality Reduction

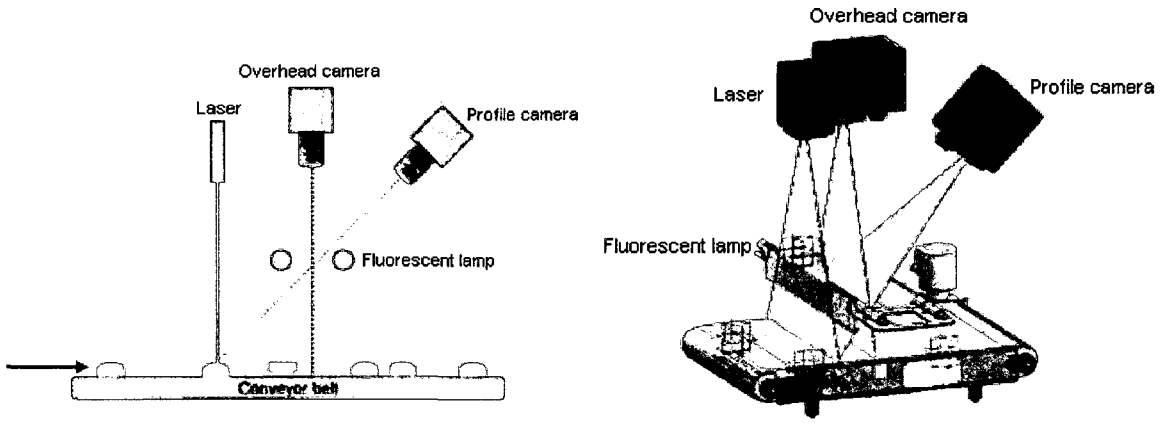
Chapter 3 detailed the evaluated machine learning techniques and the feature selection strategies selected to conduct the experimental work. The first section of this chapter presents the vision-based food inspection system which served as the platform for experimental evaluation. The second section describes the three types of bakery products for which we are experimentally trying to configure an inspection module. The last section presents and analyses the results of feature space reduction for those three products.

4.1 Experimental Food Inspection System

This section will introduce the experimental vision-based food inspection system's hardware and software.

4.1.1 Hardware Setup

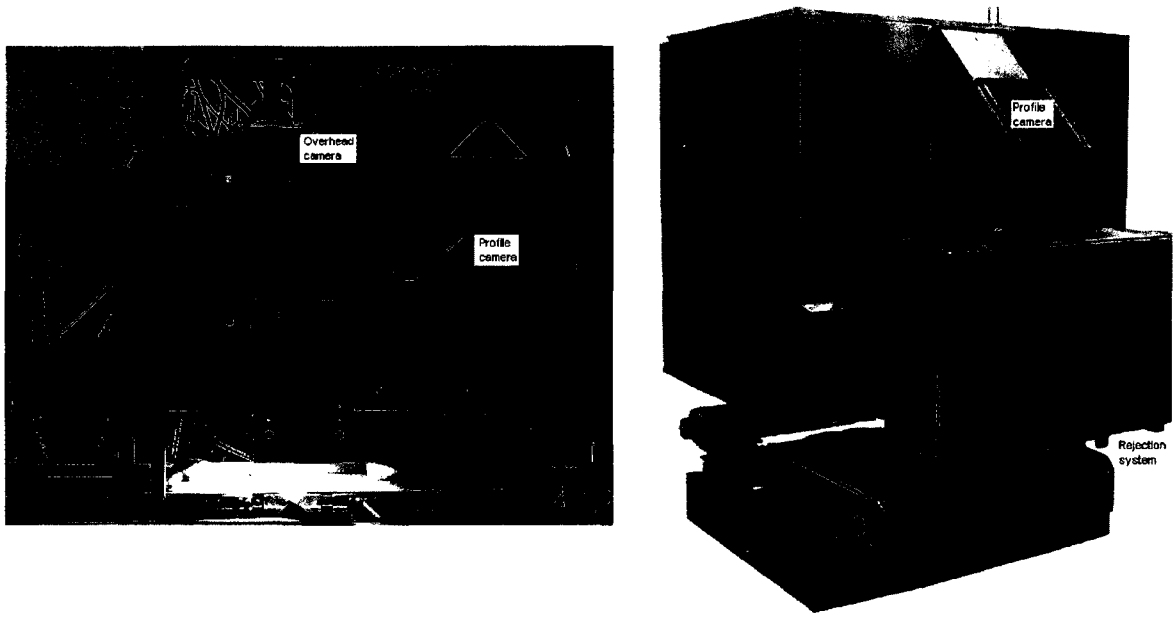
The vision-based food inspection system used for the experiments reported here is a technology developed by Dipix Technologies Inc. [4] that automates visual identification and classification of bakery products such as buns, cookies, tortillas, and pizzas. Figure 4.1 shows a conceptual view of the system.



(a) (b)

Figure 4.1. Conceptual view of Dipix food inspection system (adapted from [4])

The system is equipped with a conveyor belt which moves the bakery products on an industrial production line to the rejection and packaging systems. One camera is mounted above the conveyor belt (overhead camera) and produces real-time line scans of the top view of products moving on the conveyor belt. Two fluorescent lamps continuously illuminate the field of view of the overhead camera. A laser light strip is also projected vertically on the conveyor belt and an extra profile camera sensing the laser light in diagonal generates real-time height information on every single product. Figure 4.2 (a) shows the actual system we used for our experiments and Figure 4.2 (b) [4] shows a commercial version of the food inspection system. Some variations of the system are also equipped with a bottom camera which essentially plays the same role as the overhead camera, except that it captures images viewed from under the product. In such a case, two conveyor belts are used instead of one. The bottom camera's field of view is adjusted to coincide with the space between the two conveyor belts.



(a) (b)

Figure 4.2. Food inspection system (physical setup [4])

The system is connected to a Dual Pentium III (850 MHz) computer with 40 GB hard drive, 256 MB RAM and a 10/100 Base-TX Ethernet card. An optional air-pressure based product rejection module is attached to the system at the end of the conveyor belt. The inspection systems are built flexibly so that they can easily be inserted into different customers' production lines. Typically, a product packing system will follow the rejection module. The system used for our experiments (Figure 4.2 (a)) is from the Dipix CS18 systems' family [4], the "18" meaning that the system has a field of view of 18 inches. Typically, those food inspection systems can extract and analyze up to 200 features per product. The products' inspection happening in real-time and the conveyor belt actually running slower than its maximum velocity, the rate of production is mostly slowed down by the time the system takes to extract features and analyze a product. If for a particular product one can find a way to determine the relevant attributes that contribute to that product's classification, with respect to a learning scheme tailored for that product, then one can extract and analyze only those relevant features. This allows the conveyor belt to run faster and the system to process more products in less time. This

represents the initial motivation for the present work. The next subsection will present how the inspection system extracts features of a product.

4.1.2 Feature Extraction

As mentioned in section 4.1.1, the inspection system is equipped with a line scan camera (overhead camera) and a profile camera. Height information captured by the profile camera contains the timeframe on which it has been captured. Any signal sensed by the profile camera triggers a software interruption subroutine, alerting the system that a new product has been identified. A time stamp is also assigned to every line scan captured by the overhead camera. Figure 4.3 shows hamburger buns under inspection on the system's conveyor belt.

Figure 4.3. Example of buns under inspection

Given a constant speed of the conveyor belt, real-time image processing algorithms combine the height information from the profile camera with the data from the line scan camera in order to estimate a set of parametric features of the products under inspection. Examples of such features include color information, topping coverage, heights, diameters (for approximately elliptic products), slopes, lengths, surface area, and volume. Information extracted is presented on a graphical user interface. Figure 4.4

recapitulates the process of feature extraction where the product under inspection is a chocolate chip cookie.

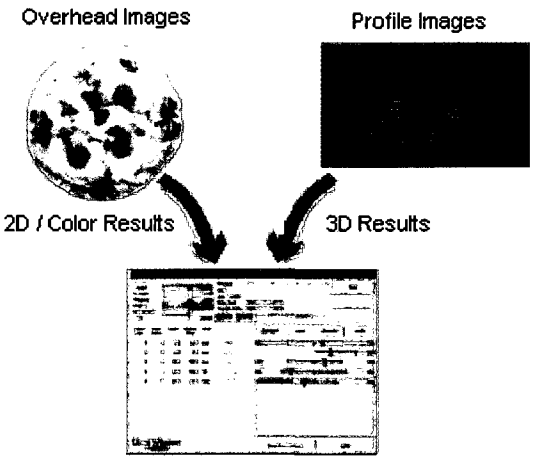


Figure 4.4. Fundamentals of feature extraction

The system analyzes the features of every product and then orders rejection of the product if it is classified as defective, or orders packaging if the product is judged acceptable. The system runs on Microsoft Windows and the inspection system's software is developed in C/C++.

4.2 Experimental Datasets and Methodology

This section gives an implementation level picture of software modules implemented in order to allow raw data extraction from the experimented bakery products. This section also portrays the different bakery products selected for experimental evaluation with the food inspection system described in section 4.1.

4.2.1 Implementation of a Data Logging Module

The food inspection system that we presented in the previous section did not have a mechanism in place to store extracted data for future analysis. For the purpose of our experiments, we developed and embedded a C++ data logging module inside the visual food inspection system's software in order to store features extracted from a product and for every inspected item. We created a parallel concurrent thread which is activated every time a new data becomes available, and sleeps for the rest of the time. The purpose behind this thread is to minimize the CPU (central Processing Unit) usage of this added module in order not to disturb the main real-time computation processes dedicated to inspection. All extracted features along with the system's decision ("accept" vs. "reject") for every single product items were saved into a text file in order to create large datasets for further analysis. We implemented two running modes for this data logging module.

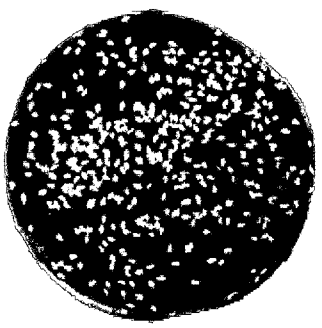
The first running mode is used when the system has already a classification rule implemented for a certain product and we wish to analyze the data and their classification by the system. In that case, the data and the decision of the system are saved. Assuming that the already existing classification rule achieves acceptable inspection results, this analysis could lead to refining the classification rule of the product or ultimately replacing the classifier with a better one.

The second running mode is used when we wish to calibrate the system for a new product or when we wish to replace an existing classifier. In this context, calibrating the system means identifying which features to extract and which rules to implement in order to objectively automate the inspection of a product. Obviously in this case, the system will not know the correct classification of the product under inspection. Therefore, typically, we ask the operator of the vision-based food inspection system to inspect the product samples in two rounds. During one of two rounds, the operator will pass on the conveyor belt a set of product samples considered as "acceptable" to the consumer. During the other round, sample products considered as "unacceptable" are

passed on the conveyor belt. The system will extract all the features from all the presented samples and will tag them as "acceptable" or "unacceptable" respectively.

4.2.2 Experimental Datasets

We conducted experiments on two categories of products: the ones for which the vision-based food inspection system already has a classifier configured for their automated classification, and one other category where the system does not have a classifier built for the product. For the category of "known products", experiments were conducted on two of the most common products inspected by the vision-based food inspection system available: buns and tortillas as illustrated in Figure 4.5 below.



(a) Seeded bun

(b) Tortilla

Figure 4.5. Example of experimented seeded bun and tortilla

Seeded buns were selected as they contain more features and more complexity than regular unseeded ones. Buns and tortillas both have "irregular" shapes as none of them has a perfectly defined geometrical shape. For both the buns and the tortillas datasets, 82 continuous features are extracted per product item, plus one boolean feature representing the decision to reject or accept the item. Each dataset contains 3287 product items. Table 4.1 summarizes the characteristics of those two datasets. All the attributes extracted can take any numeric value belonging to the set of real numbers. Data was extracted using

the first of the two running modes of the implemented data logging module as explained in section 4.2.1.

Table 4.1. Experimental datasets of "known" products

Data Set	Number of Samples	Number of Extracted Numeric Features	Number of Classes
tortillas	3287	82	2
buns	3287	82	2

In order to obtain quantitative and also qualitative product samples, we requested a video captured by equivalent versions of the vision-based food inspection from two different bakeries (one bakery for the seeded buns samples, another bakery for the tortillas samples) and the corresponding system configuration files. By qualitative samples, we mean industry standard quality products destined to reach consumers (or not reach consumers if they fail the quality inspection tests). Having those data from the industry and in large quantity will therefore serve as a basis for measuring the success of our proposed solution. The inspection system has a built in module which allows its operator to save live video of products under inspection as seen by the food inspection system. Data from the profile camera and the overhead camera of the system are saved, and can be replayed later. If the configuration files of the system on which the video has been captured are available, then any compatible food inspection system can deterministically reproduce the original system's behavior. By taking advantage of the data logging module that we developed, we can therefore mimic the system on which the images have been captured, without manipulating the actual samples, and extract the value of each attribute measured for every product.

The second category of products we experimented with is a lot different. First of all, we selected a product that is substantially different from seeded buns and tortillas. We picked a bun with a triangular shape and a non-uniform height as shown on Figure 4.6.

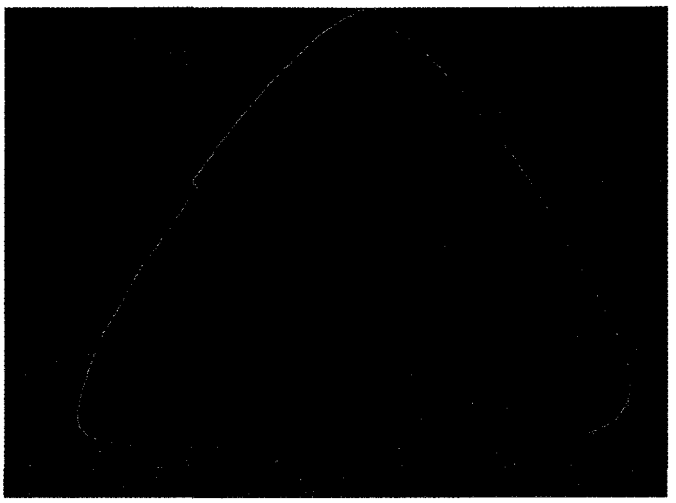


Figure 4.6. Example of experimented "9 grain ciabatta bun"

The bun is called "9 grain ciabatta bun". Given that such products were never inspected by the vision-based system used, there was not a predefined classifier available. Also, we subjectively divided the samples into "acceptable" and "unacceptable". As the products "9 grain ciabatta buns" which get to the groceries have supposedly already been classified as acceptable to the consumer, we had to create defects in some of the ciabatta buns in order to create "unacceptable" samples. As illustrated in Figure 4.7, defects created on purpose vary from holes in the buns, colors simulating over baked buns or simply a packing plastic piece which got its way on the bun, moisture on buns, broken buns or weirdly shaped buns (not triangular).

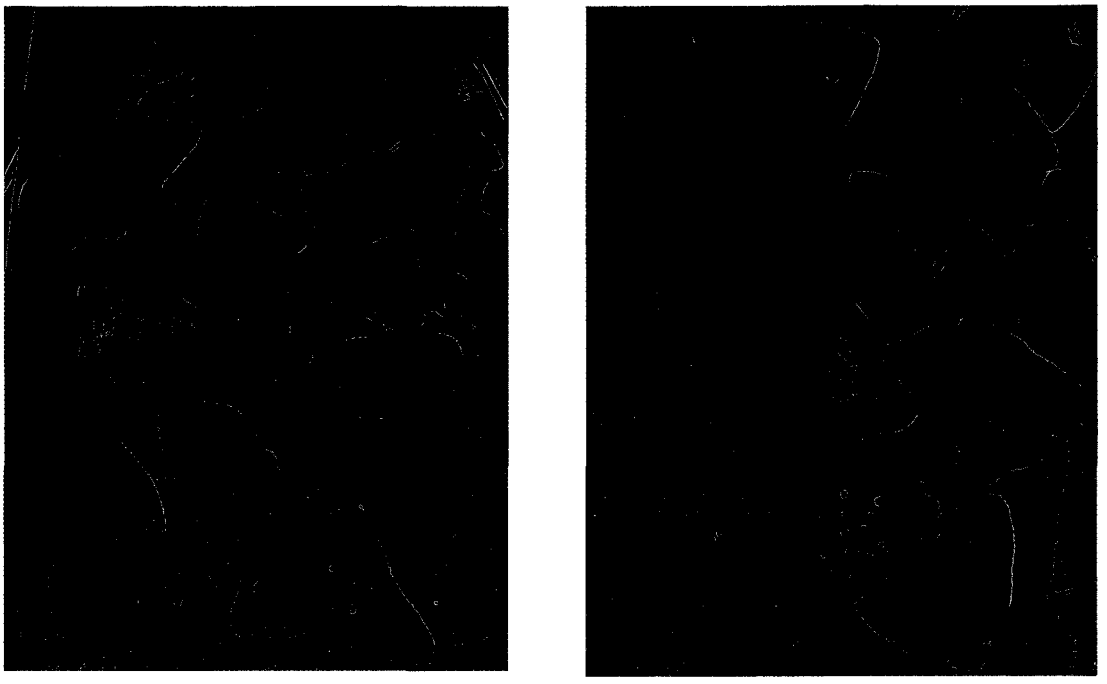


Figure 4.7. Examples of unacceptable "9 grain ciabatta buns"

Figure 4.8 shows a group of ciabatta bun samples judged "acceptable". Their shape is very close to triangular, they have approximately the same size. 160 samples containing approximately 50% of "acceptable" and 50% of "unacceptable" ciabatta buns were available. We extracted the same 82 continuous-valued features per sample as with the seeded buns and tortillas, along with the predefined boolean class attribute ("acceptable" vs. "unacceptable"). The experimental data was extracted using the second of the two running modes of the implemented data logging module as explained in section 4.2.1.

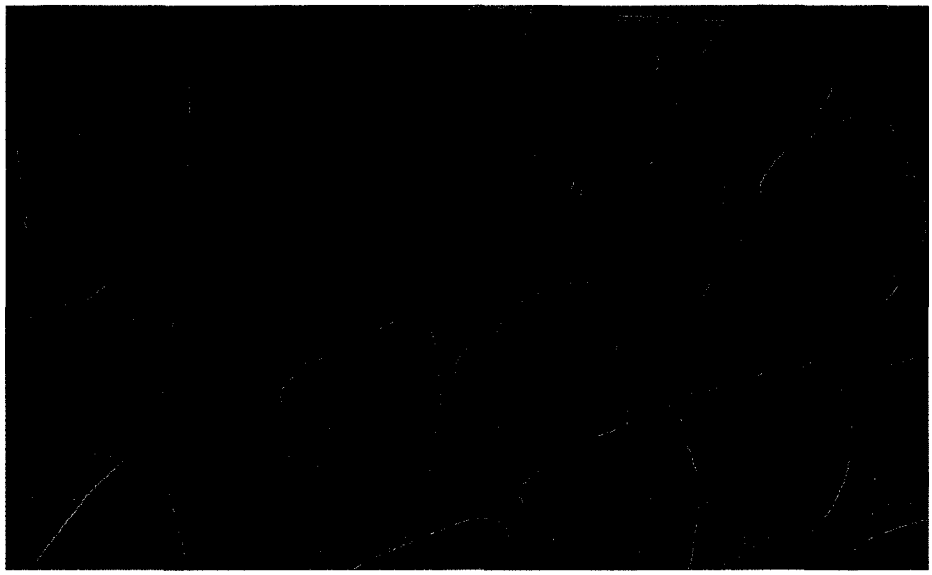


Figure 4.8. Example of acceptable "9 grain ciabatta buns"

4.2.3 Pre-processing Data

Once the data has been extracted from the training samples, they were pre-processed by another software application we built using MATLAB ® [72] This pre-processing module takes the text file containing the data extracted from samples, randomizes the samples' data, and exports the randomized data to a Comma–Separated Values (CSV) file. Randomizing data is particularly useful in the case where the inspected products have been inspected in two rounds ("acceptable" products then "unacceptable" products or vice versa) because in a real production line the defects usually come in an unordered manner. Except the first line of the CSV file, each line contains information regarding one single product. Within a line, fields are separated by commas, each field representing the value of one attribute for the product represented by that line. The fields of the first line represent the names of the features, each feature being represented by one field. The data is exported to this CSV format in order to ease up importation of the data into our open source Java application for dimensionality reduction purposes. Note that in order to evaluate the accuracy of the proposed feature reduction techniques, it is assumed that the system produces correct classification on all

items and therefore the accuracy is evaluated against the original classification of the system. Figure 4.9 shows an example of a CSV file containing the first 30 samples of the seeded buns dataset. For clarity purpose, only a few attributes are shown. Microsoft Office Excel is used to view that CSV file for ease of visual manipulation. As shown on Figure 4.9, all attributes except the class label (last column in blue labeled "---RejectedProduct") belong to the set of real numbers. It is observable from the last column that the randomization of data takes out any logical structure from the order of appearance of "acceptable" and "unacceptable" product samples.

	M0	M1	M2	M3	M30	M31	M50	M62	M63	M74	M77	M78	---RejectedProduct
1	53.929321	46.271553	36.463341	5.426131	8.94083	7868.040527	27.92194	6.84375	2.0625	-166.404602	0.033279	0.326526	YES
2	54.715263	47.18737	36.793324	5.366962	7.164307	8058.504883	27.699055	6.1875	3.3125	36.61525	0.074289	0.392381	YES
3	46.705627	38.881031	29.572266	5.222531	5.377966	5652.021484	28.944942	6.1875	3.0625	-119.988724	0.041909	0.439124	NO
4	54.570892	46.685165	36.819248	5.530966	8.146502	8030.131348	28.152063	5.9375	3.1875	64.189911	0.051241	0.532106	YES
5	56.377487	48.165688	37.834824	5.530434	8.676251	8171.463379	27.807877	6.5625	3.3125	52.721138	0.075853	0.81845	YES
6	46.780273	39.942772	30.75	4.910675	4.420973	5769.413086	27.802809	5.9375	3.03125	117.522575	0.048684	0.660283	NO
7	46.652901	39.467091	30.62779	4.829166	5.05795	5777.987305	29.699352	5.609375	3.5625	-115.197998	0.057797	0.748522	NO
8	54.582813	46.180641	35.989586	5.61578	7.894528	7974.10791	29.002228	6.09375	2.3125	-75.663559	0.091817	0.507962	YES
9	47.605469	41.202236	32.764561	4.357922	4.316486	5801.67334	28.215027	5.8125	1.6875	168.202072	0.055372	0.441589	NO
10	56.235328	48.272408	36.905371	5.533402	7.793585	7895.521484	27.18972	6.0625	3.4375	-72.529228	0.059149	0.57197	YES
11	47.897137	40.901367	31.637075	4.863232	5.095866	5669.685742	29.255199	5.6875	3.1875	-91.51133	0.053319	0.555554	NO
12	56.296875	47.201138	34.235352	6.536513	8.598229	8256.328125	27.941866	6.1875	3.78125	74.505432	0.072946	0.796136	YES
13	45.950138	38.442532	26.558594	5.82629	6.031868	5774.611328	28.745272	5.9375	3.90625	-171.841766	0.060735	0.752284	NO
14	53.825829	46.984764	36.309658	4.631346	7.571272	7952.056152	26.963821	6.4375	3.1875	160.081299	0.12586	1.572193	YES
15	46.538818	39.614704	31.113281	4.697116	4.882159	5706.030762	28.980501	5.8125	3.1875	-22.697474	0.057325	0.640332	NO
16	55.80719	47.590103	37.543945	5.629847	8.147182	7836.733887	29.610046	6.4375	4.03125	-149.179138	0.062196	0.720004	YES
17	54.529991	46.225113	35.388672	5.776593	8.624684	7743.865234	27.910303	6.5625	3.9375	-163.011703	0.065557	0.664422	YES
18	54.109074	46.926216	37.605469	4.975187	8.666592	8394.933594	27.993412	6.5625	3.6875	71.598938	0.050232	0.420199	YES
19	47.085938	39.9591	28.725586	5.588	6.090232	5631.889648	29.285172	5.9375	0.5625	-173.819412	0.0629	0.695836	NO
20	46.727039	39.987404	31.026279	4.683486	3.812981	5627.165527	27.790897	5.234375	2.8125	126.984795	0.055549	0.34426	NO
21	57.111183	48.986446	38.172745	5.712257	8.838415	7958.979004	27.373295	6.4375	3.8125	44.820251	0.035774	0.200972	YES
22	47.911243	40.401138	30.445965	5.284449	4.48421	5820.032227	27.700928	4.984375	2.8125	53.337715	0.050615	0.370921	NO
23	45.934998	38.991165	30.010851	4.812579	4.981614	5674.227539	28.242218	5.5125	3.1875	46.797291	0.068579	0.726258	NO
24	47.661133	40.632721	30.691965	5.061804	5.025706	5840.139648	28.247894	5.71875	3.1875	175.716278	0.028178	0.128021	NO
25	54.188385	46.85574	36.798378	5.267551	7.960629	8019.664063	27.862896	6.1875	3.0625	165.914886	0.053268	0.597522	YES
26	53.891434	46.314838	35.358074	5.566445	9.035226	7973.714844	28.530775	6.6875	3.8125	-162.648239	0.078176	0.532679	YES
27	54.183319	46.023449	36.11861	5.590839	7.27359	8087.048828	28.732634	5.979167	2.5625	-92.09404	0.0826	1.109635	YES
28	55.826782	47.490189	36.361492	5.986682	8.799672	8531.379883	28.409149	6.9375	3.5625	61.668381	0.14613	0.925293	YES
29	45.974659	39.433689	30.945097	4.619423	4.822511	5661.793457	28.829775	5.609375	2.8125	67.905846	0.082891	1.112257	NO
30	46.649281	39.44574	28.59375	5.390297	6.351184	5579.717773	29.751892	5.6875	3.6875	-178.949371	0.047665	0.480825	NO
31													

Figure 4.9. Example of a dataset CSV file

As the dimensionality reduction techniques presented in section 3.2 operate better or only with discrete valued features, discretization can be used to improve the performance of the evaluated feature selection techniques. For instance, the consistency-based feature selection introduced in section 3.2.3 considers two samples to be inconsistent when they match for all features in a selected feature subset except for their class label. However, the features extracted by the inspection system being able to take almost any decimal value, it is obvious that having two samples sharing the exact same value for every single measured attribute is close to impossible. Discretization of features into a limited number of possible values or intervals, provided that those values or

intervals are still meaningful to allow distinction between noticeable different values, can therefore increase the chances of having two sample products share the same value for one or more attributes. Given those considerations, feature values discretization has been applied as a pre-processing step using a very efficient discretization technique introduced by Fayyad and Irani [73]. Essentially, the technique combines an entropy-based splitting criterion with a minimum description length stopping criterion. The best-cut point is the one that makes the subintervals as pure as possible, i.e. where the information value is smallest. This is equivalent to splitting where the information gain presented in section 3.1.2 is the largest. The method is then applied recursively on the two subintervals. The stopping criterion prescribes accepting a partition T if and only if the cost of encoding the partition and the classes of the instances in the intervals induced by T is less than the cost of encoding the classes of the instances. A Java-based open source implementation of Fayyad and Irani's disretization technique is also available in [59, 60]. Note that this implementation is from the same source as the evaluated feature selection techniques, therefore the integration of discretization and feature selection was straightforward. Also note that data discretization is used only for feature selection purposes, but not for machine learning techniques evaluation.

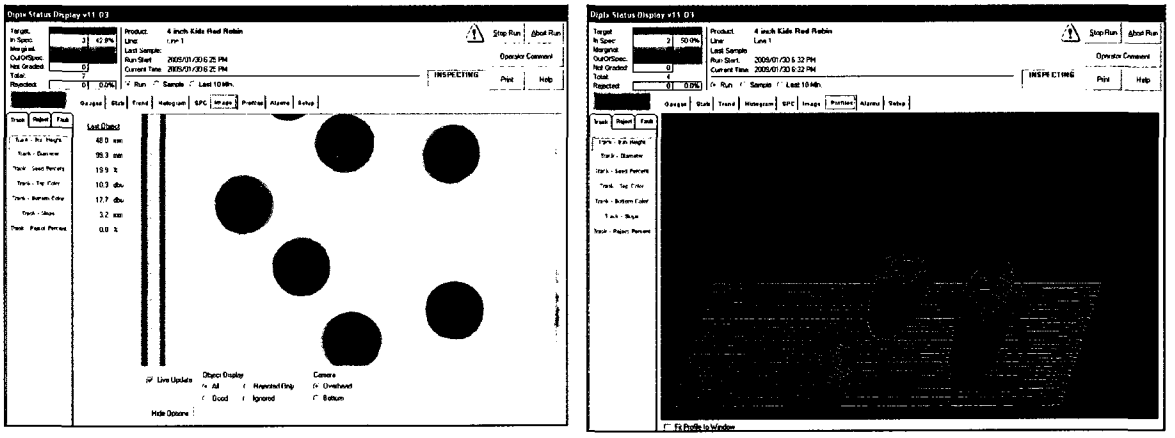
4.3 Experimental Results and Analysis

As explained previously, the vision-based food inspection system first extracts features from the sample products. Once the computed values of the features are extracted for every sample, they are then fed into the feature selection programs according to the process described in section 4.2. For the analysis, we separately consider datasets where the samples have been classified by the vision-based food inspection system, and those which samples have been classified based on human judgment. The reason for distinguishing those two categories is the fact that in the former case, the aim is to investigate if the system could use another classifier with eventually less features to analyze in order to achieve the same or a better accuracy than the current classifier. In the latter case, the idea is for the system to automatically evaluate several

configurations of classifiers and feature selection techniques for an unknown product and to present the results of the evaluation to the operator of the system for validation. In both scenarios, the operator of the system will be able to decide for a classifier quickly, based on facts from the auto-evaluation of several configurations by the system itself. This should eliminate the manual and time consuming trial-and-error process currently used to configure the system.

4.3.1 Results on Machine-Classified Datasets

The two products shown on Figure 4.5 have already customized classifiers built in the vision-based food inspection system. Therefore, in the category of machine-classified datasets, experimentation has been conducted with those two products (seeded buns and tortillas). Figure 4.10 (a) shows the overhead camera view of the seeded buns passing on the vision-based food inspection system's conveyor belt. On the other hand, Figure 4.10 (b) displays the view from the profile camera. Recall that information from both the overhead camera and the profile camera are combined in order to build a tridimensional model of every product under inspection.



(a) (b)

Figure 4.10. Overhead and profile view of seeded buns during feature extraction

Figure 4.11 illustrates the graphical user interface with the emphasis on the view of the overhead camera for the tortillas. No profile camera view is available for the tortillas inspection, the reason being that the bakery producing these tortillas has judged height information to be irrelevant in the classification of tortillas due to their thin shape.

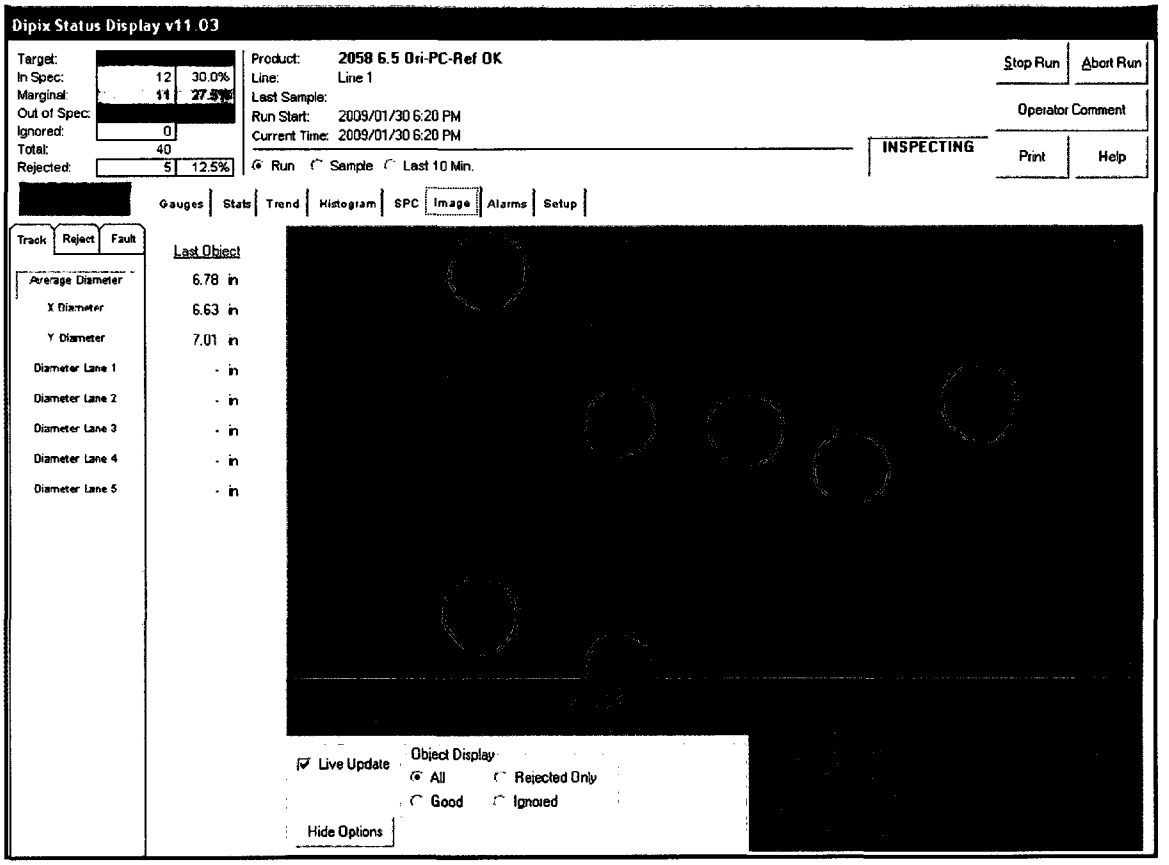


Figure 4.11. Overhead camera view of tortillas during feature extraction

After feature extraction, the data recorded by the implemented data logging module is passed to the feature selection modules to determine the minimal set of features that can be retained to correctly characterize the product and ensure proper classification as acceptable or not. Table 4.2 and Table 4.3 show the number of features experimentally selected for tortillas and seeded buns respectively, sorted in ascending order of number of features selected by the different dimensionality reduction techniques.

For the tortillas dataset, the C4.5 wrapper algorithm comes on top of the list by selecting only 4 features as being the most representative of a product's quality out of 82 features, followed equally by the CFS feature selector and the MLP wrapper that both select 5 features. The RELIEF algorithm is the last on the list as it keeps up to 33 attributes out of 82. The details of the attributes selected for the tortillas dataset are given in Appendix A.

For the seeded buns dataset, the C4.5 wrapper is still on top of the list with 2 representative features selected, followed by the consistency-based feature selection (5 features out of 82). The Naïve Bayes wrapper and the MLP wrapper both found 6 attributes to be relevant to their respective target learning algorithms. RELIEF is again the one rejecting the least attributes. The details of the actual features selected for the seeded buns dataset can be found in Appendix B.

Table 4.2. Features space reduction performance on the tortillas dataset

	Number of features selected	Number of features rejected	Feature rejection ratio (%)
C4.5 wrapper	4	78	95.12
CFS	5	77	93.90
MLP wrapper	5	77	93.90
NB wrapper	7	75	91.46
Consistency	10	72	87.80
RELIEF	33	49	59.76

Table 4.3. Features space reduction performance on the seeded buns dataset

	Number of features selected	Number of features rejected	Feature rejection ratio (%)
C4.5 wrapper	2	80	97.56
Consistency	5	77	93.90
MLP wrapper	6	76	92.68
NB wrapper	6	76	92.68
CFS	7	75	91.46
RELIEF	59	23	28.04

For both the buns and the tortillas datasets, C4.5 wrapper, MLP wrapper, Naïve Bayes wrapper, CFS and consistency-based feature selection tend to consider 10 or less features out of 82 as relevant to qualify the product, whereas RELIEF seems to be cautious by keeping many more features. The fact that only positive differences of probability were kept in the RELIEF implementation implies that weakly relevant

features are very likely to be preserved. Recall that for the RELIEF feature selection presented in section 3.2.1, the approximated difference of probability *W[A]* between the nearest hits and the nearest misses, considered as the weight of a certain attribute *A,* is normalized to the interval [-1, 1]. Features having a difference of probability in the interval [-1, 0[would be irrelevant, and those in interval [0, 1] would be neutral or relevant. We decided to keep any feature that could be relevant, even if it is weakly relevant in order not to miss any attribute worth the system's attention. One might decide to increase the lower bound of the relevant attributes' interval in order to keep only strongly relevant attributes, at the risk of missing some (weakly) relevant attributes. Kohavi and John [20] also mentioned that the RELIEF algorithm tends to keep most of the relevant features of a dataset even if they are redundant and only a fraction of them is necessary for the concept description. Moreover, wrapper feature selection techniques globally tend to select fewer features than filter-based feature selectors. This could be explained by the fact that wrappers are meant to optimize the feature selection for a particular given algorithm with which they interact during the attribute selection process.

One interesting point is the fact that all feature subsets selected by any of the filter-based feature selectors except RELIEF, are also all selected by RELIEF for the seeded buns dataset (refer to Appendix B). This remark also holds for tortillas dataset, except for the consistency-based feature selector which selected one attribute (Attribute M66) that all the other feature selectors judged irrelevant or redundant (see Appendix A).

4.3.2 Results on Human-Classified Datasets

Figure 4.12 shows an example of view during feature extraction on the "9 grain ciabatta buns". The view presented is from the perspective of the profile camera. The graphical user interface displays selected features and their value for the last sample analyzed, the mean, and the standard deviation of that feature for the previous and current samples. For the purpose of our research, the nature of the features and how they

are computed is not relevant, therefore we will describe a measured attribute only when necessary.

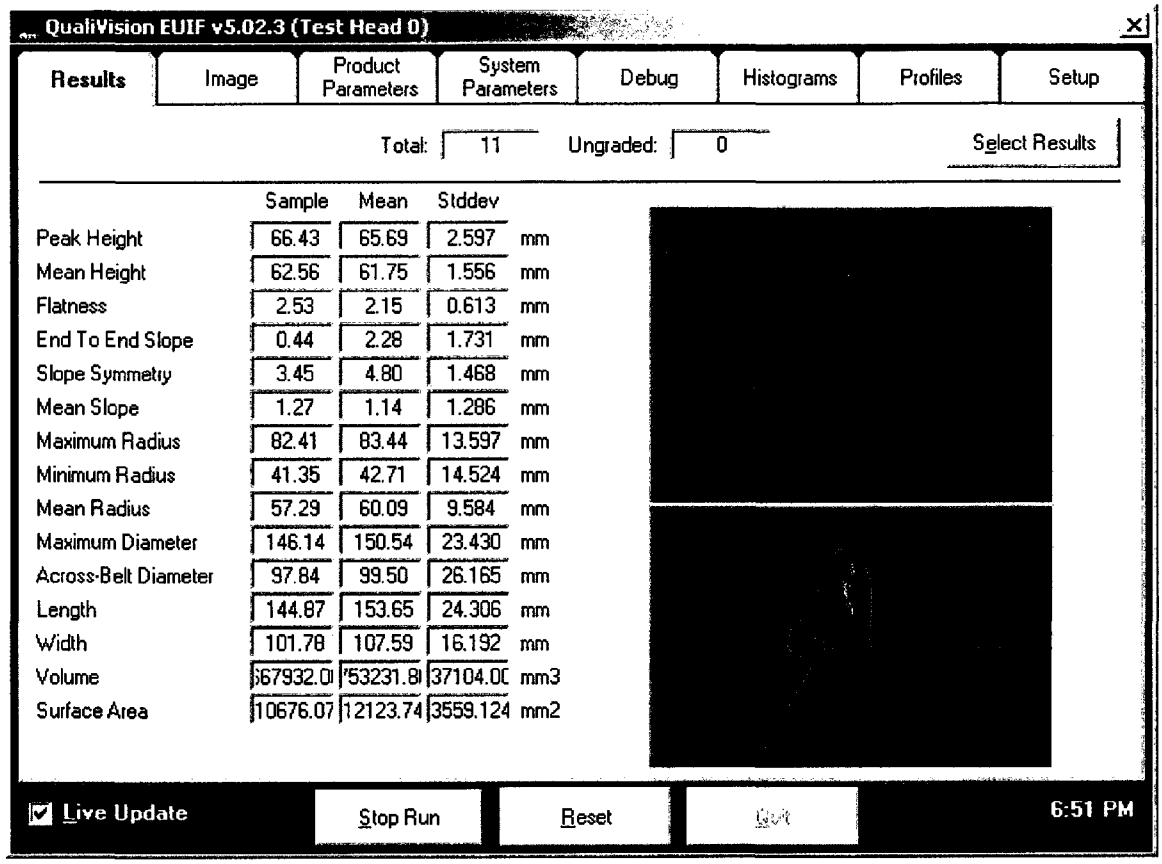


Figure 4.12. Profile view of "9 grain ciabatta" buns during feature extraction

Once the features are extracted and pre-processed, they are fed into the feature selection programs for dimensionality reduction purposes. When considering the "unknown" category of products for which the classification of the samples is done subjectively by a human operator before extracting the features using the vision-based food inspection system, the same information is presented to the different feature selection approaches. Table 4.4 shows the number of features selected for the triangular ciabatta buns, sorted in ascending order of the number of features selected.

Table 4.4. Features space reduction performance on the ciabatta buns dataset

	Number of features selected	Number of features rejected	Feature rejection ratio (%)
C4.5 wrapper	4	78	95.12
NB wrapper	4	78	95.12
MLP wrapper	9	73	89.02
Consistency	9	73	89.02
CFS	14	68	82.93
RELIEF	36	46	56.10

As with the other types of products, the feature wrappers keep the fewest attributes: C4.5 wrapper and Naïve Bayes wrapper share the first rank by selecting only 4.88% of the features (4 out of 82) as being relevant to inspect the ciabatta buns; followed by the MLP wrapper which selects 9 attributes (10.98%). The consistency-based feature selector also selects 9 features out of 82. RELIEF ends up again as the one removing the least attributes by keeping 36 features (43.90%). RELIEF is preceded by CFS which judges 14 features (17.07%) to be relevant and non redundant. All the features selected by the different techniques are either also selected by RELIEF, or have a maximum of two features which are not selected by RELIEF. Except RELIEF, all the attribute subset selectors kept less than 15 features (18.29% of all features) for all the experimented datasets. The first advocates of feature wrappers (Kohavi and John [20]) actually claimed that wrapper feature selectors are able to distinguish between weakly and strongly relevant features. They formally defined those two degrees of relevance of an attribute. They defined a feature X_i to be strongly relevant to the target concept(s) if the probability distribution of the class values, given the full feature set, changes when X_i is removed. A feature X_i is said to be weakly relevant if it is not strongly relevant and the probability distribution of the class values, given some subset S (containing X_i) of the full feature set, changes when X_i is removed. All features that are not strongly or weakly relevant are considered irrelevant.

Appendix C shows the features selected by each dimensionality reduction technique for the "9 grain ciabatta" buns.

4.3.3 Results and Analysis Summary

These experiments with three datasets of products with different characteristics point out that feature wrappers generally achieve better results than filters due to the fact that they are tuned to the specific interaction between an induction algorithm and its training data. However, they tend to be much slower than feature filters because they must repeatedly call the induction algorithm and must be re-run when a different induction algorithm is used. In our experiments, wrappers training time for five-fold cross validation reached several minutes rather than only a few seconds with filter-based selectors.

Except for RELIEF, all the attribute subset selectors kept less than 15 features (18.29% of all features) for all the experimented datasets. Nevertheless, the number of attributes selected as relevant by the different feature selectors should be interpreted with caution. For artificial datasets, it is pretty straightforward to evaluate the performance of a feature selection algorithm. However, for real world datasets such as the ones reported in our experiments, it is not necessarily clear what the relevant features are. Therefore, whether the selected features are relevant or not can only be determined indirectly.

One way is to see the effect of feature selection through a learning algorithm. Although a dataset with fewer features would be preferred for production rate enhancement, the accuracy of prediction with the reduced datasets is of capital importance. This makes us favor prediction accuracy over dimension reduction of the dataset for industrial quality inspection applications, as considered here.

4.4 Chapter Summary

This chapter described the vision-based food inspection system whose performance our work is trying to improve. We described the experimental methodology and some implementation level details for achieving the experiments on three types of bakery

products: seeded buns, tortillas, and "9 grain ciabatta buns". More specifically, experiments demonstrated that significant reduction in the dimensionality of the feature space required to properly characterize food products can be achieved. This approach offers a promising solution to replace trivial intuitive settings of industrial inspection systems currently used in practice. Experiments also showed wrapper feature selectors to generally select less attributes than filter-based feature selectors, but are drawn back by the high computational cost they require due to repetitive interaction with the target learning algorithm. Unlike artificial datasets where it is clear whether the relevant features are chosen or not, real world datasets as reported here require indirect evaluation by checking a learning algorithm's performance before and after feature space dimensionality reduction. Chapter 5 will further investigate this aspect.

Chapter 5. Learning Schemes and Classification Performance

Chapter 4 has shown that most of the feature selection techniques considered have judged less than 20% of the features to be relevant for tortillas, seeded buns and ciabatta buns proper classification. In spite of this observation, we have recognized that measuring the percentage of the feature space dimensionality reduction is not a sufficient metric to evaluate the relevance of the reduced feature space. While one can easily evaluate the relevance of the feature subset proposed by a feature selection algorithm when the datasets are artificial, this same task is not necessarily straightforward with real world datasets as reported here. Consequently, in most situations, relevance of feature space dimensionality reduction techniques is indirectly evaluated by observing the performance of a learning algorithm over the reduced feature space. In order to better guide the compromise that must be found between prediction accuracy and dimension reduction of the dataset for industrial quality inspection applications, the study involves the three different machine learning techniques introduced in section 3.1: Naïve Bayes (a probabilistic learner), C4.5 (a decision tree learner) and Multi-Layer Perceptron (MLP, a neural networks learner). Hence, the objective of this chapter will not be to find the best candidate learning scheme for a certain type of product or the best feature selector for that same product, but rather to study the performance of each machine learning scheme when performing respectively over the full set of features and over the subsets of features retained by the feature selection techniques. The operator of the vision-based food inspection system can therefore make a wise and informed decision on the inspection model best suited to fulfill his or her business objectives from the auto-evaluation of the system's predicted performance with several different models of classifiers. The first subsection will thus describe the methodology of the experiments and define some criteria for model selection. Then the following three subsections will present and analyze the results of the conducted experiments on the three products presented in section 4.2.2. Finally, the last subsection will summarize and conclude this chapter.

5.1 Experimental Methodology and Learning Evaluation Criteria

Three key performance indicators were considered in the scope of this study: the accuracy of the classifiers built by the three evaluated machine learning schemes operating over the full set of features and over the four feature subsets derived from the four feature selection techniques, the time taken by those classifiers to predict the class of an unknown product sample under inspection (testing time), and the time taken to actually build the classifiers from the training samples (training time).

The accuracy represents the percentage of products that were classified from the built classifiers (over reduced sets and full set of features) in the exact same way as the original classification of the system without feature reduction for products for which the system has already a configured classifier. For products which do not have a predefined classifier, the considered original classification would be the one based on the subjective judgment of consumers (see section 4.2.2). Formally speaking, the classification of an instance of a product produced by a classifier is considered accurate if and only if this classification matches the classification produced by the food inspection system's built in classifier (or the subjective classification if there is no built in classifier). It naturally follows that a classification which is not accurate is considered inaccurate. The accuracy of a classifier is consequently defined as the ratio of accurately classified product samples out of the total number of samples inspected by the classifier. The choice of this "ground truth" is based on the assumption that we can simply not find a better reference than the classification of commercial products inspected by an industrial quality control system, or the classification made by a consumer.

The holdout and cross-validation methods [71] are considered to evaluate the accuracy of the prediction on the class (as "acceptable" or "unacceptable") of the sample products. These two different estimation methods are considered with the aim of evaluating the worth of using one over the other in the context of vision-based food inspection. The holdout method, also called test sample estimation, separates the data into two mutually exclusive subsets: the training set and the test set, or holdout set. As

the name of the sets indicates, the training set is used for the training phase, and the test set is used later to evaluate performance. In *t*-fold cross-validation on the other hand, the data is randomly split into *t* mutually exclusive subsets (the folds) of approximately equal size as explained in section 3.2.4. For the present evaluation, 10-fold cross-validation was used for model selection as suggested by Kohavi in [71], and the cross-validation was repeated 10 times with different random seeds. The accuracy is afterward averaged over the 10 repetitions of the 10-fold cross-validation. Holdout was repeated 100 times on the datasets and the overall accuracy was also averaged. Note that Kohavi and John [20] suggest only 5-fold cross validation for feature reduction purposes where the interest is to identify relevant and non redundant features. Their experiments showed that 5-fold cross validation gives satisfactory results. The other reason is the fact that during feature selection, cross-validation is applied to every feature subset proposed by the feature search algorithm until a satisfactory subset is found, with respect to the defined stopping criterion; therefore having a relatively small number of folds accelerates the process without sacrificing a lot of quality. On the other hand, during model selection where the focus is more on finding a reliable classifier rather than finding a subset of relevant and non redundant features, the cross-validation is done only on one feature-reduced subset (or on the full set of features if no dimensionality reduction has been accomplished). In this latter case, increasing the number of folds *t* to 10 in the *t*-fold cross-validation only moderately impacts running time. Because for both cross-validation and holdout the test is done on product samples that were not seen previously, the average accuracy will also give an indication of how well the classifiers can generalize to new samples. The standard deviation of the accuracy will also indicate how close the estimated accuracies of the 100 classifiers built during the 10 repetitions of 10-fold cross-validation, and the 100 classifiers built during the 100 repetitions of holdout, are close to the average. In fact, in probability theory and statistics, standard deviation is a measure of the variability or dispersion of a population, a data set, or a probability distribution. A low standard deviation indicates that the data points tend to be very close to the same value (the mean), while high standard deviation indicates that the data are "spread out" over a large range of values. A low standard deviation of the

accuracy will therefore indicate that most of the classifiers produce an accuracy close to the average estimated accuracy.

The second key performance indicator, namely the testing time, represents the time taken by a classifier to correctly (or incorrectly) classify a product instance. The testing time therefore directly impacts the rate of production as it indicates how fast the system can process a product once the features to be analyzed are extracted. Faster processing time means that the system can inspect more products in the same amount of time as long as the vision-based food inspection system's hardware and image processing software can allow it. When Naïve Bayes and MLP learning schemes are considered, the same features used in the training process (see Table 4.2, Table 4.3, and Table 4.4) will appear in the generated classifier, therefore having an impact on the testing time. However for the C4.5 decision tree inducer, the built classifier may or may not contain all the features used in the training process as decision trees typically build their tree from a subset of the available features as mentioned in section 2.3.2.2. Therefore when it comes to evaluating the testing time of a decision tree like C4.5, the size of the generated tree should be taken into account. In this context, when we talk about the size of a tree, we refer to the number of branches in the tree plus one for the top node. For example, the size of the decision tree on Figure 2.8 is 8. Smaller decision trees are also generally preferred because they are easier to be interpreted by humans. Very often, a human operator relates the decision trees to facts that he or she knows about the concept described by the decision tree.

The third element we evaluate for every machine learning algorithm over a set or subset of features is the time taken to "learn" how to classify a certain type of product. Because the training of the classifiers is done offline, the time taken to do so does not impact the productivity of the food inspection system. It is clear that for the same accuracy and testing time performances, one will prefer the fastest learner, but when it comes to making a compromise between those three factors, one would generally be more willing to sacrifice learning time in favor of accuracy or testing time.

The decision on the right classifier will be based on those three key performance indicators. Ideally, one would like a classifier that will learn in the shortest time, classifies in the fastest time and be the most accurate. But when those three combinations do not occur simultaneously, then it comes down to the operator of the vision-based food inspection system to decide which performance he or she is willing to sacrifice to the benefit of one or more other performance aspects.

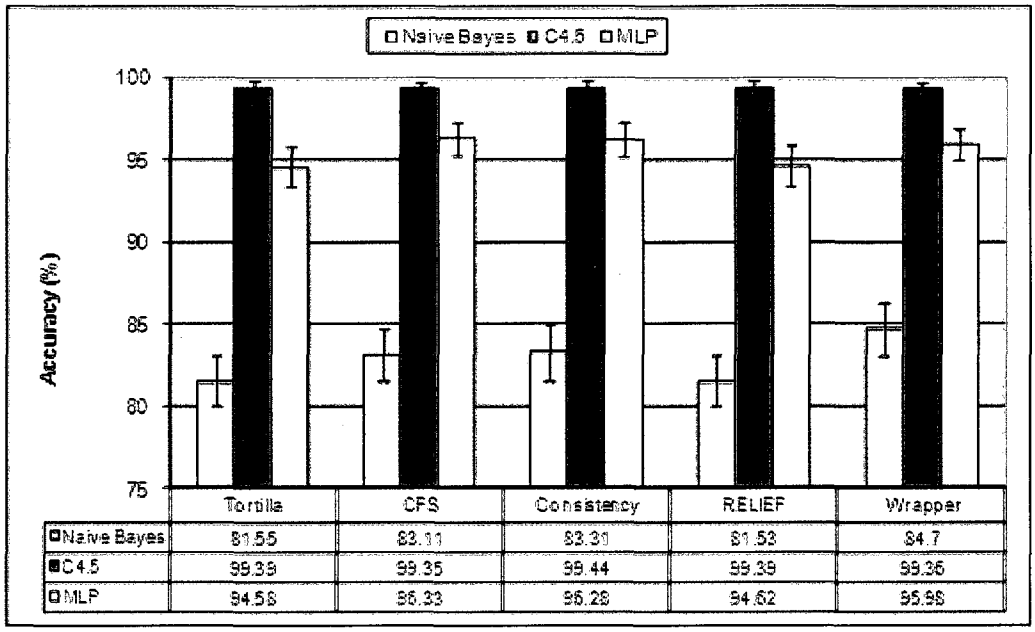
It is worth mentioning that the continuous features generated by the inspection system are discretized only for feature selection purposes. After feature selection, the reduced datasets are extracted from the original continuous datasets and then passed to the learning algorithms for accuracy estimation. The original full dimension dataset is also passed to the learning algorithms in order to evaluate if and at which cost improvement can be achieved by means of feature selection. The same train/test sets and the same folds were used for all learning schemes in order to establish a common base for comparison.

5.2 Tortillas and Model Selection

5.2.1 Classification Accuracy for Tortillas

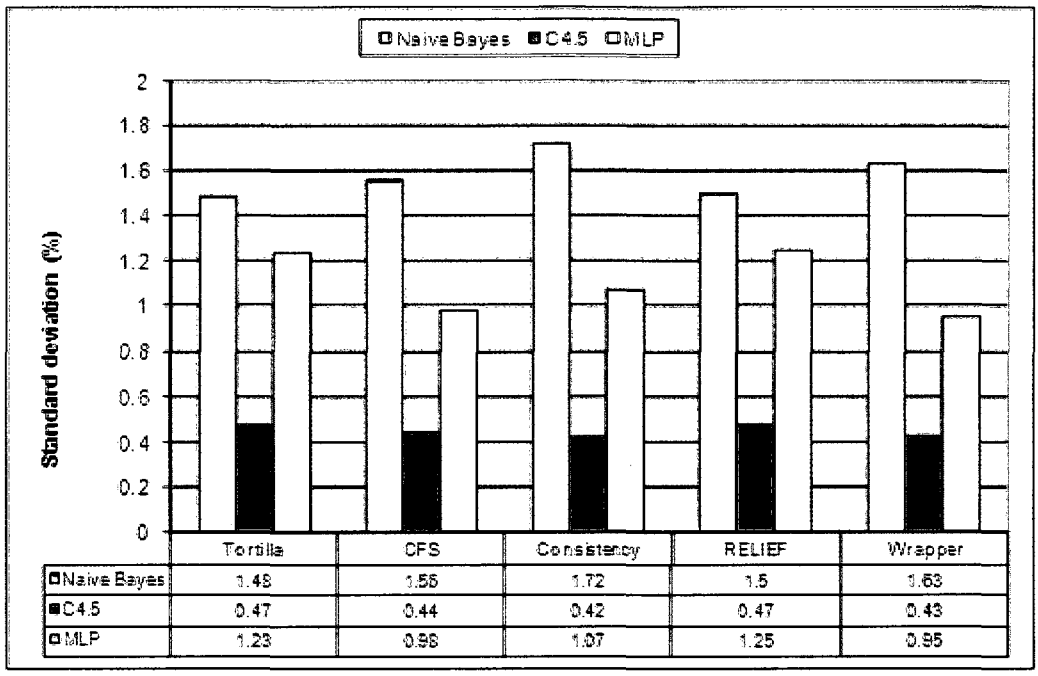
Figure 5.1 shows the accuracy estimation on the class prediction with the different feature selection techniques for the tortillas dataset evaluated using the three learning schemes: C4.5, Naïve Bayes, and MLP. The results presented on that figure are averaged from ten repetitions of the 10-fold cross-validation. The vertical lines at the edge of the bars of Figure 5.1 represent the standard deviation. Figure 5.2 shows a better view of the standard deviation resulting from the 10-fold cross-validation on the tortillas dataset. On both figures "Tortilla" represents the original full dataset which did not undergo any feature selection. Similarly, Figure 5.3 and Figure 5.4 present the results averaged over 100 repetitions of holdout accuracy estimation for the tortillas dataset. Recall from section 5.1 that 100 % accuracy of classification corresponds to the

classification produced by the original classifier built in the vision-based food inspection system, which we have considered as the "ground truth".



	Tortilla	CFS	Consistency	RELIEF	Wrapper
Naive Bayes	81.55	83.11	83.31	81.53	84.7
C4.5	99.39	99.35	99.44	99.39	99.36
MLP	94.58	96.33	96.28	94.62	95.98

Figure 5.1. Average over 10 repetitions of the 10-fold cross-validation accuracy estimation for the tortillas dataset



	Tortilla	CFS	Consistency	RELIEF	Wrapper
Naive Bayes	1.48	1.56	1.72	1.5	1.63
C4.5	0.47	0.44	0.42	0.47	0.43
MLP	1.23	0.98	1.07	1.25	0.95

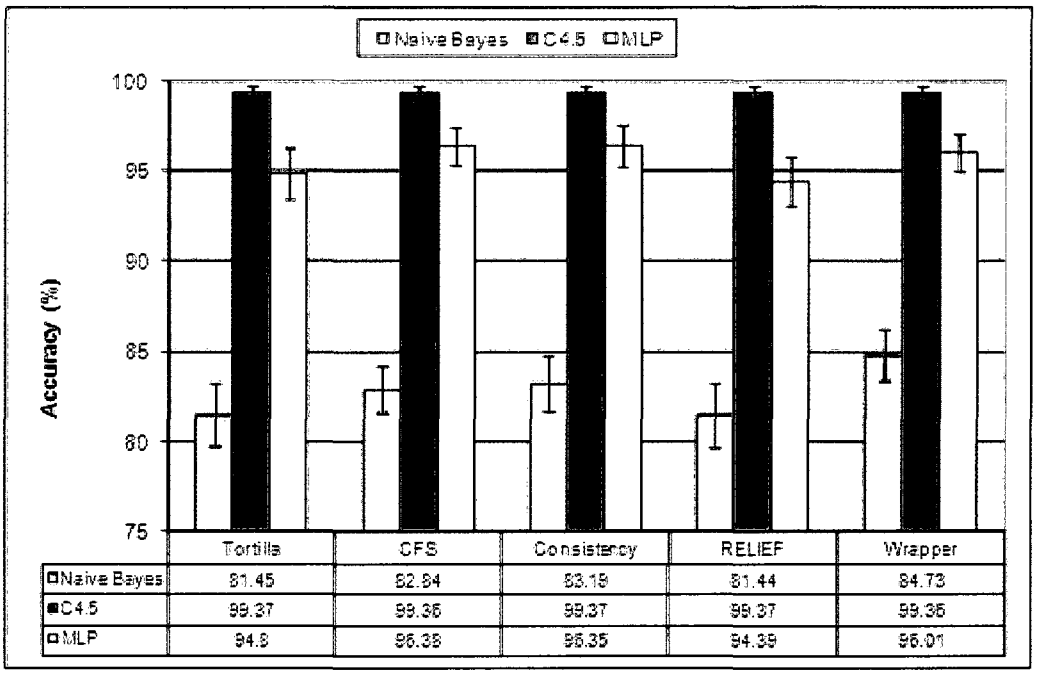
Figure 5.2. Standard deviation of 10 repetitions of the 10-fold cross-validation accuracy estimation for the tortillas dataset

Observation of both the results for cross-validation and holdout accuracy estimation tests for the tortillas dataset shows that for all the considered feature selectors, the C4.5 learning scheme globally gives the highest accuracy and the lowest standard deviation at the same time, followed closely by the MLP learner and far beyond by the Naïve Bayes learner. It has been advanced in section 5.1 that a low standard deviation indicates that the accuracies of the classifiers built during cross-validation or test sample estimation tend to be very close to the average accuracy; while high standard deviation indicates that these accuracies are "spread out" over a large range of values. It is important to note that in the experiments presented for tortillas, holdout accuracy estimation uses 90% of the data for training and 10% for test.

For the C4.5 learning scheme which outperformed MLP and Naïve Bayes, results of the cross-validation point out that the full original tortillas dataset had an accuracy of 99.39% with a standard deviation of 0.47%. Consistency-based subset evaluation gave a slightly better accuracy (99.44%) than the full dataset. The RELIEF feature selector is

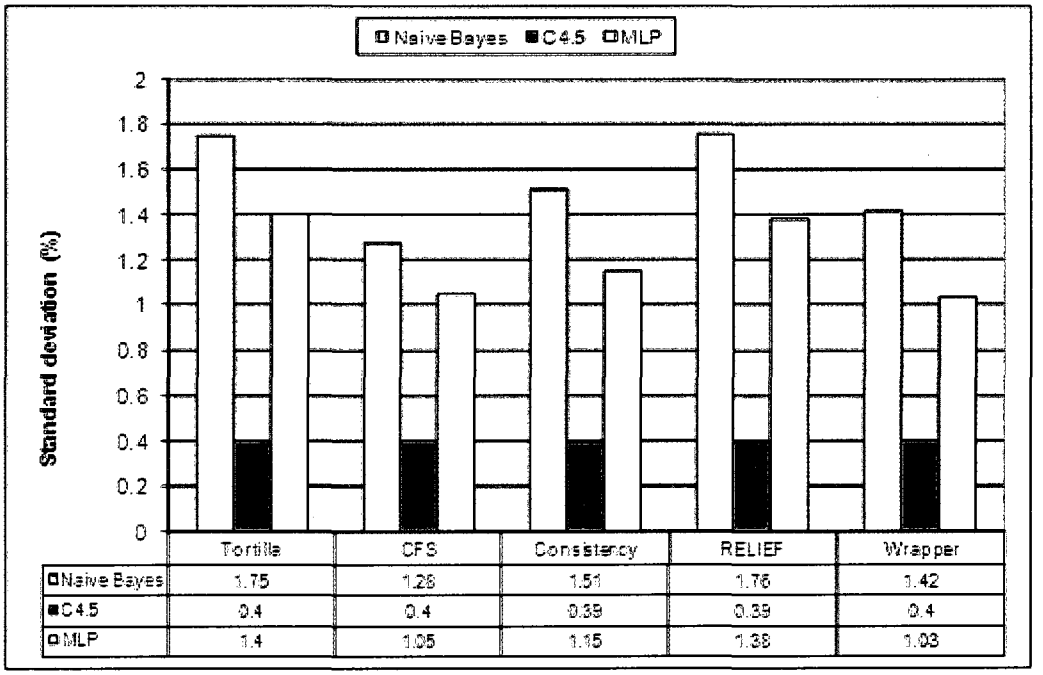
the second best by having the same accuracy estimation as the original dataset, followed by the C4.5 wrapper and the correlation-based feature selector with an accuracy of respectively 0.03% and 0.04% below that of the full dataset. The standard deviation for the C4.5 decision tree inducer is relatively low and varies from 0.42% for consistency-based feature selection to 0.47% for the full features dataset when cross-validation is used for model selection.

When holdout is the chosen technique for model selection (Figure 5.3 and Figure 5.4), C4.5 decision tree inducer once again achieves the best overall classification accuracy, followed by MLP and tailed by Naïve Bayes learner. The order of performance of the feature selectors for C4.5 learner for holdout accuracy estimation is very similar to the one for cross-validation: consistency-based feature selector is still first, except that this time it shares the first rank with RELIEF and the non-reduced dataset with an estimated accuracy of 99.37%. The C4.5 wrapper and the CFS equally dispute the fourth rank with 99.36% correct estimation. The standard deviation for C4.5 holdout accuracy estimation is slightly lower than when cross-validation is used and lies between 0.39% (consistency-based feature selection and RELIEF) and 0.40% (full features dimension, CFS, and C4.5 wrapper).



	Tortilla	CFS	Consistency	RELIEF	Wrapper
Naive Bayes	81.45	82.84	83.18	81.44	84.73
C4.5	99.37	99.36	99.37	99.37	99.36
MLP	94.8	96.38	96.35	94.39	96.01

Figure 5.3. Average over 100 repetitions of the holdout accuracy estimation for the tortillas dataset



	Tortilla	CFS	Consistency	RELIEF	Wrapper
Naive Bayes	1.75	1.28	1.51	1.76	1.42
C4.5	0.4	0.4	0.39	0.39	0.4
MLP	1.4	1.05	1.15	1.38	1.03

Figure 5.4. Standard deviation of 100 repetitions of the holdout accuracy estimation for the tortillas dataset

The second performer of the learning schemes for the tortillas dataset, namely the MLP, seems to learn better when feature selection is achieved. As a matter of fact, cross-validation results show that all of the four dimensionally reduced datasets actually gave better estimated accuracy compared to the full dimension dataset. Correlation-based feature selection worked the best for MLP by allowing an average correct classification of 96.33% (±0.98%) over the ten repetitions of the 10-fold cross-validation while the full feature dataset achieved the lowest performance of 94.58 % (±1.23%). Consistency-based feature selection closely follows CFS with a score of 96.28% (±1.07%). The MLP wrapper is the next best candidate with 95.98% (±0.95%) correct classification ratio, which is just 1.36% above RELIEF. The standard deviation for the MLP is within the interval 0.95% (MLP wrapper) to 1.25% (RELIEF), which is higher than the standard deviation of C4.5.

Holdout results (Figure 5.3 and Figure 5.4) also unanimously point the dataset which undergoes correlation-based feature selection (CFS) to be the best performer for MLP by correctly classifying 96.38% (±1.05%). As with cross-validation, consistency-based feature selection and MLP wrapper follow very closely with respective accuracy of 96.35% (±1.15%) and 96.01% (±1.03%). Only RELIEF performed less than the full features tortillas dataset with correct classification of 94.39% (±1.38%), that is 0.41% below the accuracy of the full tortillas dataset. The standard deviations of the holdout results are bounded by the interval [1.03%, 1.40%], which corresponds to the respective standard deviations of the MLP wrapper and the full dimension tortillas datasets.

By achieving correct classification of inspected products varying from 81.53% for RELIEF feature-reduced dataset to 84.7% for Naïve Bayes wrapper, Naïve Bayes learner exhibits the lowest performance among the three experimented learning schemes using cross-validation. When Naïve Bayes is used as the classifier for tortillas, cross-validation results show that this probabilistic learner performs best when Naïve Bayes wrapper feature selection is applied (84.7% ± 1.63%). Consistency and CFS also performed accuracy wise better than the full feature dimension dataset (81.55% ± 1.48%) by correctly classifying 83.31% (±1.72%) and 83.11% (±1.56%). RELIEF is the last of

the list with 81.53% (±1.5%), just 0.02% less than the performance of the full dimension tortilla dataset. The standard deviations of all the tortillas datasets for Naïve Bayes learning lie between 1.48% (full dimension dataset) and 1.72% (consistency-based feature selection).

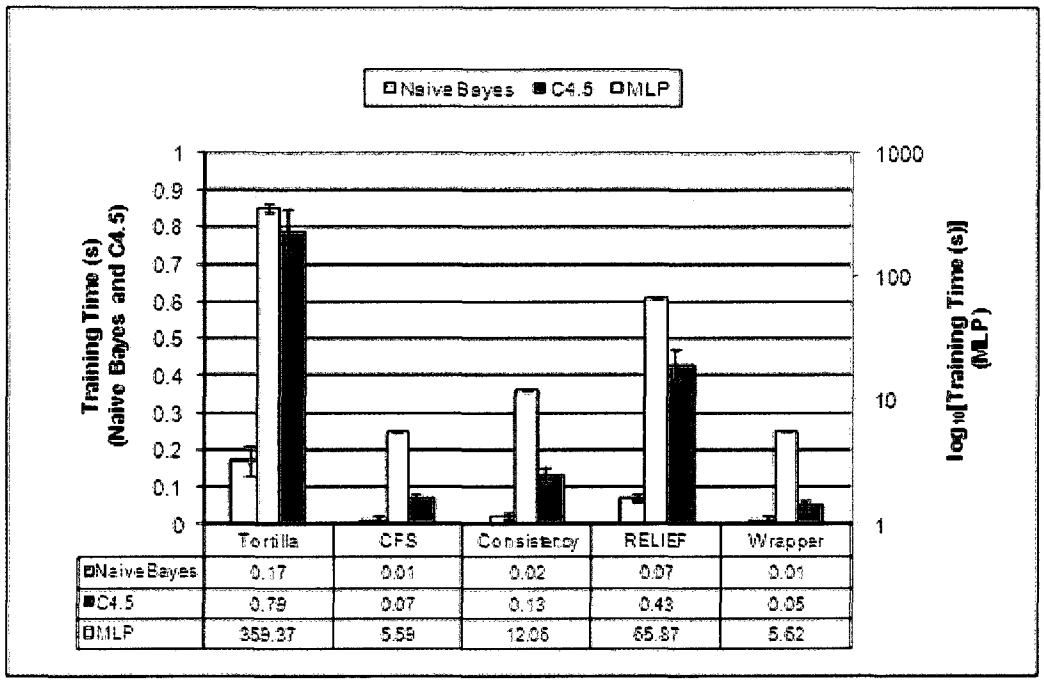
When repeated holdout is used instead of cross-validation for model selection, the order of performance of the feature selectors exploited by the Naïve Bayes learner is exactly the same as the one obtained when cross-validation is the accuracy estimation method: Naïve Bayes wrapper (84.73% ± 1.42%), consistency-based feature selection (83.19% ± 1.51%), CFS (82.84% ± 1.28%), original dataset without feature selection (81.45% ± 1.75%) and RELIEF (81.44% ±1.76%). Note that in this case also, RELIEF is performing very close to the full original dataset by providing only 0.01% less accurate classification. Naïve Bayes standard deviations are generally higher than those for MLP, which are in turn higher than those for C4.5.

The fact that the Naïve Bayes classifier gives lower accuracy estimations compared to the C4.5 and the Multi-Layer Perceptron can be attributed to the assumption that the former algorithm makes about features being conditionally independent. In fact, several of the features in the dataset are correlated, for example features such as the mean diameter of an approximately circular product and its surface area are clearly correlated. MLP was able to capture a certain rule for the classification of the products considered because of the structure inherent to the MLP which allows capturing complex input/output relationships. C4.5 giving very high classification accuracy can be explained by the fact that the vision-based food inspection system inherently uses a decision structure very close to a decision tree.

5.2.2 Training Time and Testing Time for Tortillas

Given that C4.5 decision tree inducer seems to be the one achieving the highest accuracy for the full-feature dimension and for all the considered feature reduced tortillas

datasets, the next step is to consider time-related performance of the different machine learning techniques. Figure 5.5 shows the average over ten repetitions of ten-fold cross-validation of the time in seconds taken by the three evaluated machine learning techniques to generate a classifier for the tortillas given the training samples. Exceptionally for MLP, the training time is plotted in decimal logarithmic scale in order to compensate the difference in scales with Naïve Bayes and C4.5 training times. The machine learning algorithms built five different classifiers each, one from the original full-dimension dataset and four from feature-reduced datasets (which resulted from the application of CFS, consistency-based, RELIEF, and the wrapper feature selection). Figure 5.6 on the other hand shows the time in microseconds (μs) actually taken by the generated models for tortillas (combination of a classifier and a feature subset) to classify one test sample in average. Holdout estimation results are similar to those of cross-validation and are therefore presented in Appendix D.

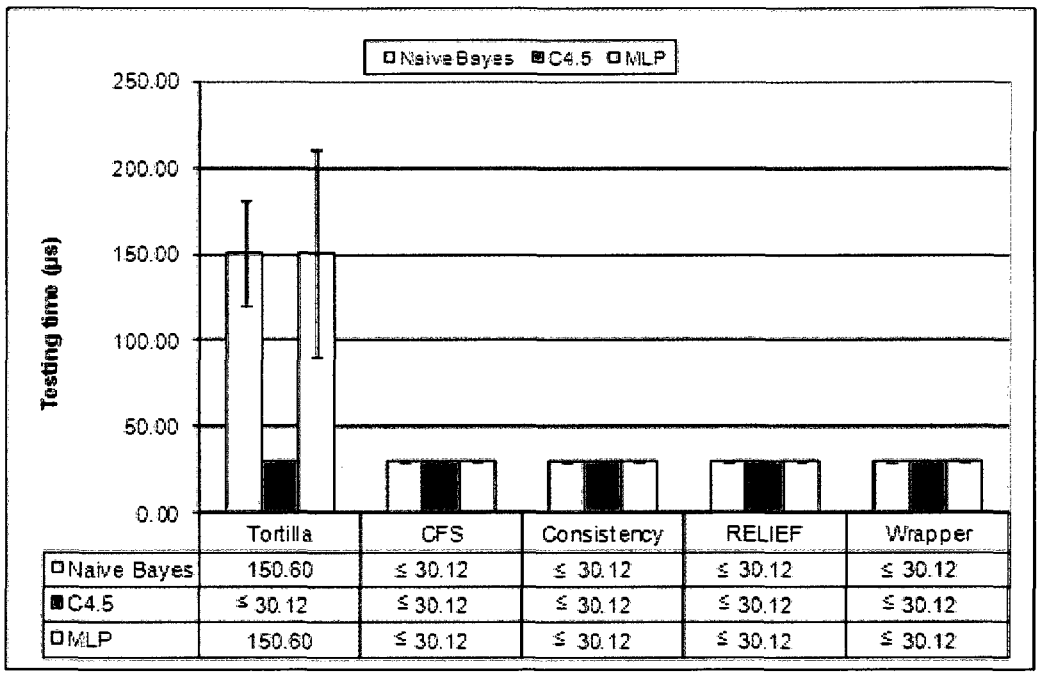


	Tortilla	CFS	Consistency	RELIEF	Wrapper
NaiveBayes	0.17	0.01	0.02	0.07	0.01
C4.5	0.79	0.07	0.13	0.43	0.05
MLP	359.37	5.59	12.06	65.87	5.62

Figure 5.5. Average training time over 10 repetitions of the 10-fold cross-validation accuracy estimation for the tortillas dataset

As illustrated by Figure 5.5, for a given feature set (or subset), the Naïve Bayes is the fastest learner, followed by the C4.5 decision tree inducer. The MLP is the slowest learner. This conclusion holds for the full dimension dataset and for the ones that have undergone filter-based feature selection because in this scenario, the feature subsets and the full feature set do not depend on a learning algorithm. One cannot compare the time performance of the classifiers built from the three datasets created by the wrapper feature selectors because in this case, the feature subset depends on the targeted learning algorithm, and therefore the datasets are potentially different for every wrapper trained with a target learning algorithm. MLP took almost 6 minutes (359.37 seconds) to train the full tortillas dataset containing 82 features, compared to 0.79 second for C4.5 and only 0.17 second for Naïve Bayes. This result is not surprising, given that Naïve Bayes only computes a priori probabilities from the training samples and therefore the training time depends mostly on the number of training samples and the number of features ($O(nk)$ for k features and n training samples as highlighted in section 3.1.1). The C4.5 decision tree inducer training uses an iterative process whose time depends on the number of features, the number of training samples and more importantly the time taken by the incorporated feature search engine. The MLP training time mostly depends on the number of features, the number of nodes in the network, the learning rate η and the momentum constant α defined in Equation (3.18), and the MLP algorithm's stopping criteria. For the same learning algorithm, the training time generally increases as the number of analyzed features increases. As an example, for the C4.5 learner, the fastest training time (50ms) occurs when the training dataset contains only the features derived from C4.5 wrapper (4 features), followed respectively by the datasets containing the features selected by CFS (5 features trained in 0.07 second), consistency-based feature selection (10 features trained in 0.13 second), RELIEF (33 features trained in 0.43 second). The full feature set without any feature dimensionality reduction took the longest training time for C4.5: 82 features trained in 0.79 second. This predictable observation (see Table 4.2 for the features space reduction performance on the tortillas datasets) corroborates the fact that the smaller the number of features to analyze for the learning schemes considered in the experiments reported here, the shorter the training time. Because the training of the classifier is done offline (meaning that it is done prior to

the actual real-time inspection process), the training time impacts the rate of production less than the testing time which should in principle correlate to the actual time taken to analyze the inspected products. For this reason, all the testing time results presented in this thesis represent the average time per product sample, and not the training time for the whole dataset. This way, the testing time presented illustrates better the time required to classify an unknown instance of tortilla. Figure 5.6 shows the average testing time taken to classify a test sample for the same combinations of classifier/feature subset for the tortillas dataset.



	Tortilla	CFS	Consistency	RELIEF	Wrapper
Naive Bayes	150.60	≤ 30.12	≤ 30.12	≤ 30.12	≤ 30.12
C4.5	≤ 30.12	≤ 30.12	≤ 30.12	≤ 30.12	≤ 30.12
MLP	150.60	≤ 30.12	≤ 30.12	≤ 30.12	≤ 30.12

Figure 5.6. Average testing time over 10 repetitions of the 10-fold cross-validation accuracy estimation for the tortillas dataset

Except for the classifiers built from the datasets containing the whole set of features, all the other classifiers generated after feature selection (CFS, consistency, RELIEF, or the wrapper feature selection) tested a product sample in an average time inferior or equal to 30.12µs. C4.5 seems to be the fastest algorithm in terms of classification time by achieving a record time of less than 30.12µs per sample, for all the considered datasets (including the full set of features). When the full set of features is

considered, MLP and Naïve Bayes test a product sample in an average time of 150.60μs, that is at least five times longer than C4.5 learning scheme. The testing time of the Naïve Bayes classifier will depend on the number of features analyzed as the number of computed probabilities will vary accordingly. The testing time of an MLP classifier will depend more on the number of nodes (which depends on the number of features to some extent as the input layer will have the same number of nodes as the number of features). The C4.5 testing time will depend on the number of leaves in the generated decision tree, which depends both on the number of features and the inter-correlation between the features and the different classes. The training and testing times when holdout accuracy estimation is used are presented in Appendix D because of their similarity with those of cross-validation.

5.2.3 Model Selection for Tortillas

For the tortillas, experimental results show that regardless of the feature selection technique employed (including no feature selection at all), the C4.5 decision tree learner seems to achieve a better accuracy, a lower standard deviation and a lower testing time than the other two experimented learning schemes (Naïve Bayes and MLP). Section 5.2.1 highlighted that accuracy wise, C4.5 performs the best when operating only over the feature subset proposed by the consistency-based feature selection technique. Cross-validation results place RELIEF feature selection and no feature selection at the second place in terms of accuracy performance, followed by C4.5 wrapper and finally by CFS, whereas holdout results tie RELIEF and no feature selection with Consistency at the first rank, followed equally and very closely by C4.5 wrapper and CFS.

When speaking about the testing time for a decision tree, the size of the tree is directly correlated to the time taken to classify an instance of a product. Figure 5.7 below illustrates the average size of the trees produced during the ten repetitions of cross-validation, and during the 100 repetitions of holdout respectively. One can quickly

observe that the average size of the decision trees is pretty much the same for cross-validation and holdout accuracy estimation tests.

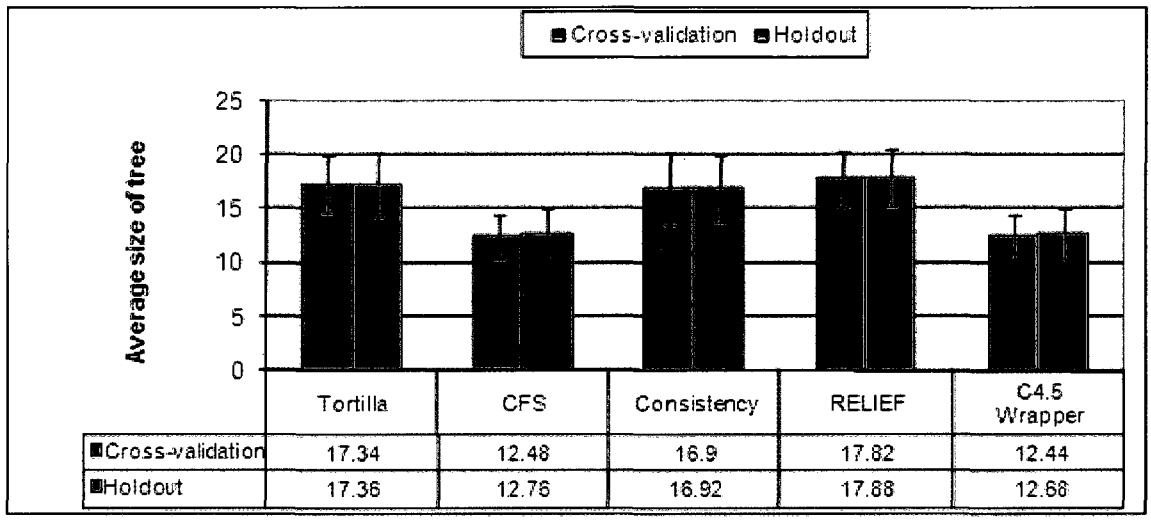


Figure 5.7. Average size of trees produced by C4.5 (Tortillas)

Figure 5.7 shows that tortillas datasets on which C4.5 wrapper or CFS feature selection has been achieved produce trees of the smallest size while those filtered by RELIEF and those with no feature selection at all produce the largest trees. Datasets filtered by consistency evaluation produce trees of size between those two extremes.

The decision of which feature selector one should combine with the C4.5 decision tree will be determined by the interests of the operator of the inspection system. For instance, an operator who prefers the highest accuracy estimation would use a consistency-based feature selection which however generally generates slightly bigger trees than when C4.5 wrapper or CFS is used. Relatively large decisions trees take slightly longer to classify a product and are less obvious for human interpretation. On the other hand, an operator who is ready to tolerate slightly lower estimation accuracy to achieve a smaller decision tree would select the C4.5 classifier built from the feature subset judged relevant by C4.5 wrapper or CFS. In this scenario, smaller trees allow faster processing by the inspection system and are also easier to interpret for a human

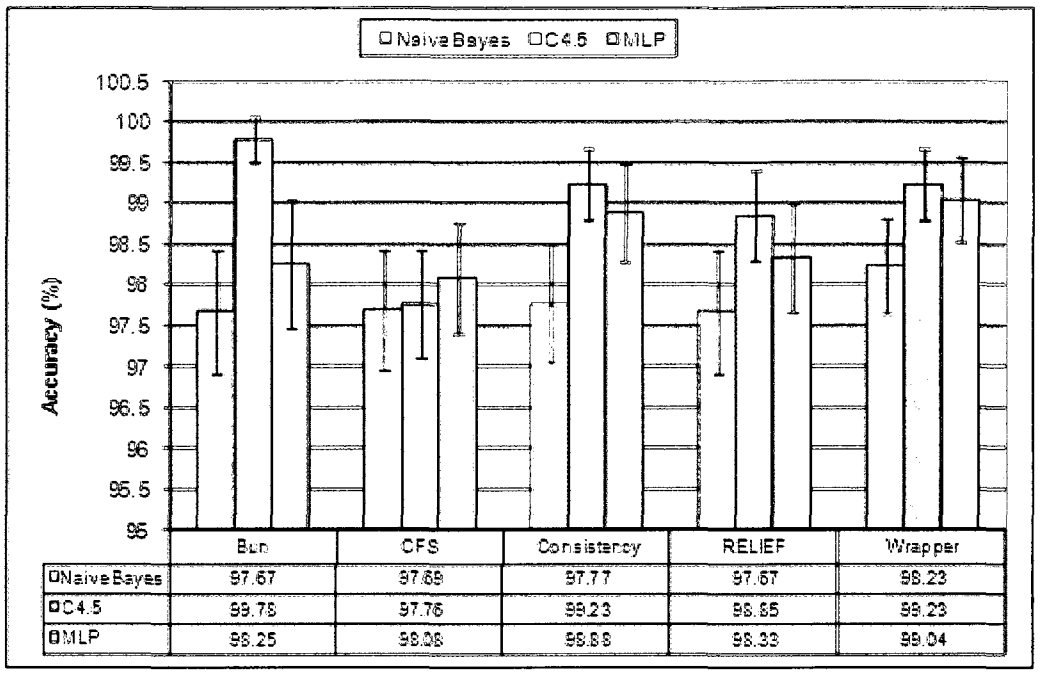
being. In both scenarios, the food inspection system should perform faster and produce more accurate classification than when no feature selection is achieved.

Given the experimental results of the predicted performance of the three evaluated learning schemes operating over the full feature dimension dataset and those containing only the features selected by the four feature selection techniques, C4.5 triumphs as the best candidate learning scheme for tortillas. The choice of the feature selector to use with C4.5 decision tree inducer will be driven by the business objectives of the company exploiting the vision-based food inspection system. The choice of cross-validation or holdout accuracy estimation test did not have a significant impact on the conclusions presented regarding model selection for tortillas.

5.3 Seeded Buns and Model Selection

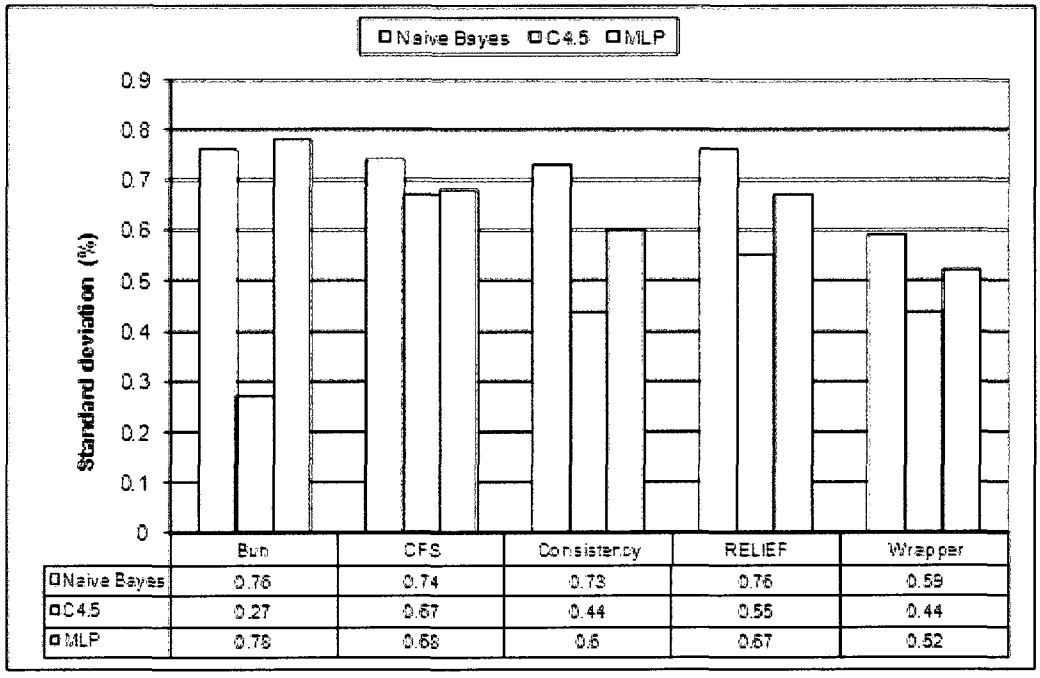
5.3.1 Classification Accuracy for Seeded Buns

Figure 5.8 shows the results averaged over 10 repetitions of 10-fold cross-validation for the seeded buns dataset. Figure 5.9 shows the standard deviations of the estimated accuracy presented by Figure 5.8. Similarly, Figure E.1 presents the classification accuracy of the three evaluated machine learning algorithms averaged over 100 repetitions of the holdout test, for the full dimension buns dataset and the four datasets on which feature space reduction has been performed. Figure E.2 displays the standard deviation of the accuracy estimates presented on Figure E.1. The holdout accuracy test results are presented in Appendix E as they are comparable to the cross-validation results. In the experiments presented for seeded buns, each repetition of holdout accuracy estimation uses 90% of the data for training and 10% for test as was done in the experiments on tortillas. The "ground truth" for classification accuracy corresponds to the original classification achieved by the classification model built in the vision-based food inspection system as justified in section 5.1. On all figures, "Bun" represents the original full dataset containing all the features.



	Bun	CFS	Consistency	RELIEF	Wrapper
Naive Bayes	97.67	97.69	97.77	97.67	98.23
C4.5	99.78	97.76	99.23	98.85	99.23
MLP	99.25	99.08	99.88	98.33	99.04

Figure 5.8. Average over 10 repetitions of the 10-fold cross-validation accuracy estimation for the seeded buns dataset



	Bun	CFS	Consistency	RELIEF	Wrapper
Naive Bayes	0.76	0.74	0.73	0.76	0.59
C4.5	0.27	0.67	0.44	0.55	0.44
MLP	0.78	0.68	0.6	0.67	0.52

Figure 5.9. Standard deviation of 10 repetitions of the 10-fold cross-validation accuracy estimation for the seeded buns dataset

From the results of both the cross-validation and the holdout, C4.5 is once again the learning scheme giving globally the highest accuracy estimation and the lowest standard deviation, followed by the MLP and the Naïve Bayes respectively. Recall that evaluation of accuracy on tortillas datasets also converged to this same conclusion. However, for all the datasets that have undergone correlation-based feature selection (CFS), MLP achieves a higher accuracy than C4.5 for seeded buns. This higher accuracy of MLP over C4.5 is nevertheless challenged by a lower standard deviation by C4.5 predictions compared to MLP predictions, meaning that C4.5 predictions are more confident than MLP predictions.

When cross-validation is performed, the original full seeded buns dataset has an accuracy of approximately 99.78% with a standard deviation of 0.27% and none of the reduced dataset is able to achieve a better accuracy for C4.5 decision tree inducer. However, the consistency-based subset evaluation and the C4.5 wrapper closely follow the original full buns dataset by both achieving an estimated accuracy only 0.55% inferior to that of the full dataset. RELIEF occupies the third place and CFS the fourth with an accuracy less than 1% lower than that of the full seeded buns dataset. Holdout results attest that exact same performance order: original full dataset leads the way (99.82% ± 0.25%), consistency and C4.5 wrapper arrive in the second place (99.26% ± 0.44%), RELIEF is fourth (98.85% ± 0.52%) and CFS is last of the list, yet a good performer with accurate classification of 97.93% ± 0.68%. Both cross-validation and holdout accuracy estimation methods show C4.5 accuracy estimates to have a standard deviation varying between 0.25% and 0.68%.

Cross-validation results with MLP as the candidate learning scheme showed that all datasets which underwent feature space reduction except CFS gave a better accuracy classification of the seeded buns compared to the original full dimension dataset. MLP wrapper has the highest accuracy (99.04% ± 0.52%), followed respectively by consistency-based feature selection, RELIEF, full original buns dataset, and CFS with accuracy estimates 0.17% below that of the full dataset. Once again, the results of repeated holdout accuracy estimation revealed the exact same order of classification.

Accuracy estimates of MLP fit their standard deviation into the interval [0.55%, 0.78%] for cross-validation and holdout, which is slightly higher than when C4.5 learning scheme is considered.

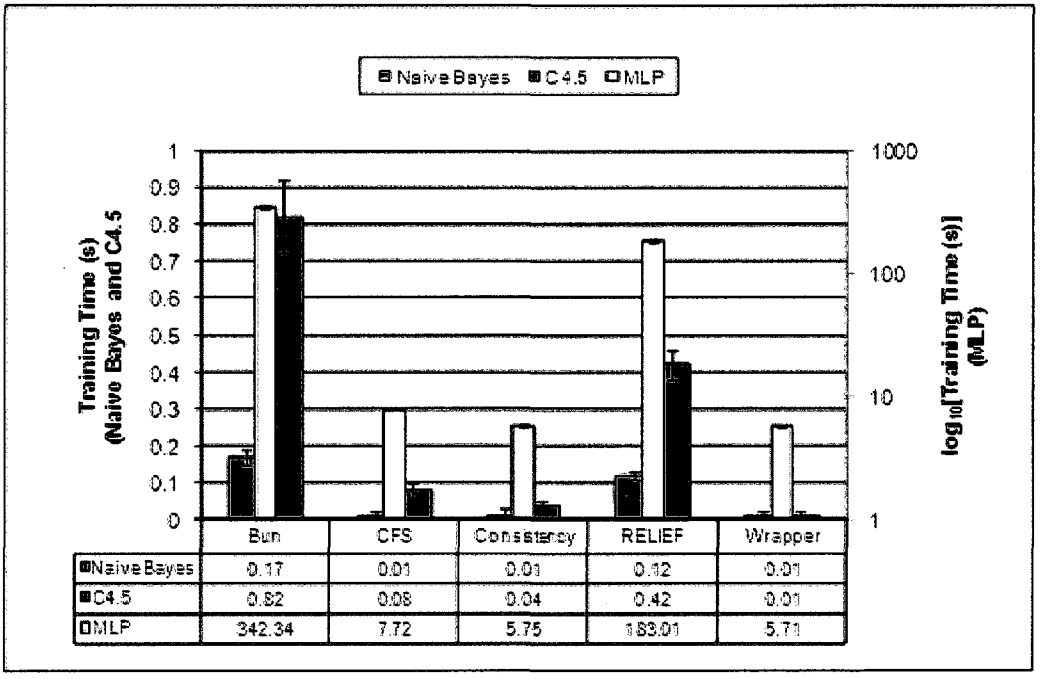
For the Naïve Bayes classifier, cross-validation showed that three out of four feature selection techniques improve the accuracy over the full original dataset: Naïve Bayes wrapper with 0.56% improvement, consistency-based feature selection with 0.10% improvement, and CFS with 0.02%. The RELIEF technique, ranked fourth, achieves the same 97.67% accuracy as with the full original seeded buns dataset. Holdout also shows three feature selectors out of four to perform better than the full dimension dataset, but with a slightly different order: Naïve Bayes wrapper still champions with 0.54% improvement, CFS with 0.03% enhancement, consistency-based method with 0.02% enhancement. RELIEF and the full dataset are the last of the list with an equally accurate classification of 97.73%. As observed with tortillas, for seeded buns also, Naïve Bayes classifier performs the best when operating only over the features selected by Naïve Bayes wrapper. Except when the full set of features is considered, Naïve Bayes tends to give a higher standard deviation than MLP.

Results of both cross-validation and holdout accuracy estimation show that two out of the three evaluated machine learning schemes (MLP and Naïve Bayes) achieve their greatest performance when operating over the feature subset retained by a feature wrapper tailored to perform with them. Even though the third learner (C4.5) reached its maximum accuracy when learning over the full set of features, this machine learning technique also happens to be the best accuracy performer for seeded buns automated inspection when feature selection is initially accomplished.

5.3.2 Training Time and Testing Time for Seeded Buns

Another aspect taken into account in the process of identifying the best classifier for seeded buns is the time taken to train the classifier to distinguish between acceptable

and unacceptable buns, and more importantly the time taken to actually classify an unknown instance of seeded bun once the classifier is trained. Figure 5.10 shows the average over ten repetitions of 10-fold cross-validation of the time taken in seconds by the three evaluated machine learning techniques (Naïve Bayes, C4.5 decision tree, and MLP) to generate a classifier for seeded buns given the training samples. The training time for MLP is plotted in decimal logarithmic scale because of its values relatively larger than Naïve Bayes and C4.5 training times. The machine learning algorithms built five different classifiers each, one from the original full-dimension dataset and four from the feature-reduced datasets (which resulted from the application of CFS, consistency-based, RELIEF, and the wrapper feature selection).



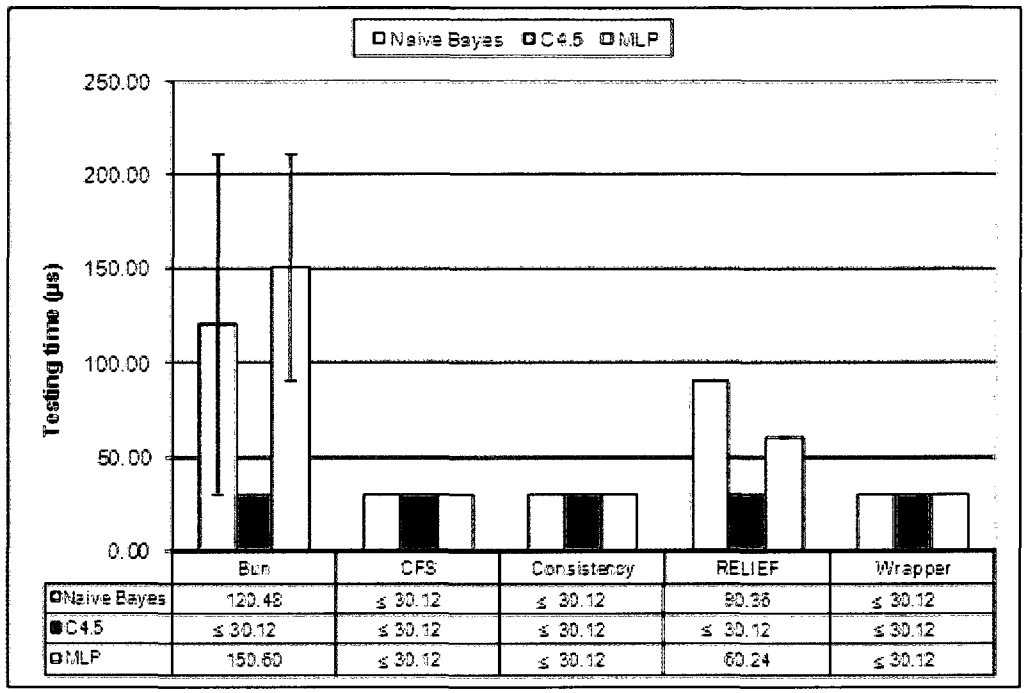
	Bun	CFS	Consistency	RELIEF	Wrapper
Naive Bayes	0.17	0.01	0.01	0.12	0.01
C4.5	0.82	0.08	0.04	0.42	0.01
MLP	342.34	7.72	5.75	183.01	5.71

Figure 5.10. Average training time over 10 repetitions of the 10-fold cross-validation accuracy estimation for the buns dataset

There is no surprise to observe again that for every dataset derived from filter-based feature selection, Naïve Bayes is the fastest to build a classifier, followed by C4.5 and finally by MLP. For the filtered-based feature selection, the feature subset produced by the consistency method trains the fastest (0.01 second for Naïve Bayes, 0.04 second for

C4.5 decision tree and 5.75 second for MLP), CFS feature subset follows (0.01 second for Naïve Bayes, 0.08 second for C4.5 and 7.72 second for MLP), and RELIEF takes the longest to train: 0.12 second for Naïve Bayes, 0.42 second for C4.5 and 183.01 seconds (a bit more than 3 minutes) for MLP. Recall from Table 4.3 that consistency-based feature selection retained 5 features, CFS kept 7 features, and RELIEF found 59 features necessary for seeded buns classification. In the case of feature wrappers, Naïve Bayes took 0.01 second in average to build a classifier from a dataset containing only the 6 features proposed by the Naïve Bayes wrapper, C4.5 took the same duration as Naïve Bayes to propose a classifier from the two features suggested by C4.5 wrapper, while MLP took 571 times more to generate a classifier from the 6 features recommended by the MLP feature wrapper. For any of the three evaluated learning schemes, the classifier built from the full set of features records the longest training time compared to when feature selection is accomplished. For instance, when no feature selection is achieved, Naïve Bayes' average training time to build a classifier over the full set of features is 17 times the duration taken when only the features selected by Naïve Bayes wrapper are used, C4.5 takes 82 more times compared to when C4.5 feature wrapper is used, and MLP takes about 60 times longer than when learning only over the feature subset reduced by MLP wrapper.

From the testing time perspective, Figure 5.11 below shows the average cross-validation performance of the three evaluated machine learning techniques operating on the full set of features and on the different subsets of features derived by means of feature selection. The testing times presented correspond to the time taken in average to classify one unknown instance of a product.



	Bun	CFS	Consistency	RELIEF	Wrapper
Naive Bayes	120.48	≤ 30.12	≤ 30.12	90.36	≤ 30.12
C4.5	≤ 30.12	≤ 30.12	≤ 30.12	≤ 30.12	≤ 30.12
MLP	150.60	≤ 30.12	≤ 30.12	60.24	≤ 30.12

Figure 5.11. Average testing time over 10 repetitions of the 10-fold cross-validation accuracy estimation for the buns dataset

It can be noticed from Figure 5.11 and Table 4.3 that for of all the three learning schemes, the classifiers built from the five features retained by the consistency-based feature selection, the seven features from CFS, the two features from C4.5 wrapper, the six features from Naïve Bayes wrapper and the six features from MLP wrapper are able to classify an unknown product instance in an average time of 30.12µs or less. Classifiers built after RELIEF feature selection and from the full set of features record the longest testing time, up to 90.36µs for RELIEF and up to 150.60µs taken in average to classify a product sample. C4.5 learning scheme generally champions the classification time while Naïve Bayes operates faster than MLP for the full dataset, but slower when RELIEF feature selection has been achieved.

Holdout accuracy estimation gave comparable results to those of cross-validation. Those test sample estimation results are attached in Appendix F for that reason.

5.3.3 Model Selection for Seeded Buns

In section 5.3.1, experimental results on seeded buns indicated that C4.5 decision tree generally gives the highest estimated accuracy compared to MLP and Naïve Bayes learning schemes when the same sets of features are considered. For a decision tree like the one generated by C4.5, the size of the tree directly impacts the time taken to classify a product. Figure 5.12 shows the average size of the trees generated by the C4.5 decision tree inducer for the full set of features and the different feature subsets suggested by the three feature filters and the feature wrapper selection techniques. It can be seen that when only the features suggested by the C4.5 wrapper or those filtered by means of the consistency-based feature selection are used to train the C4.5 learning scheme, the generated trees record the smallest average size of 5. The dataset containing the full set of features was actually able to induce C4.5 decision trees of an average size smaller than that of the datasets containing only the feature subsets proposed by CFS or RELIEF feature selection. This performance of C4.5 over the full set of features can be attributed to C4.5 discovering and using strong interaction between attributes before the data becomes too fragmented. The biggest trees were generated when C4.5 learns over the feature subsets judge relevant by RELIEF feature selection.

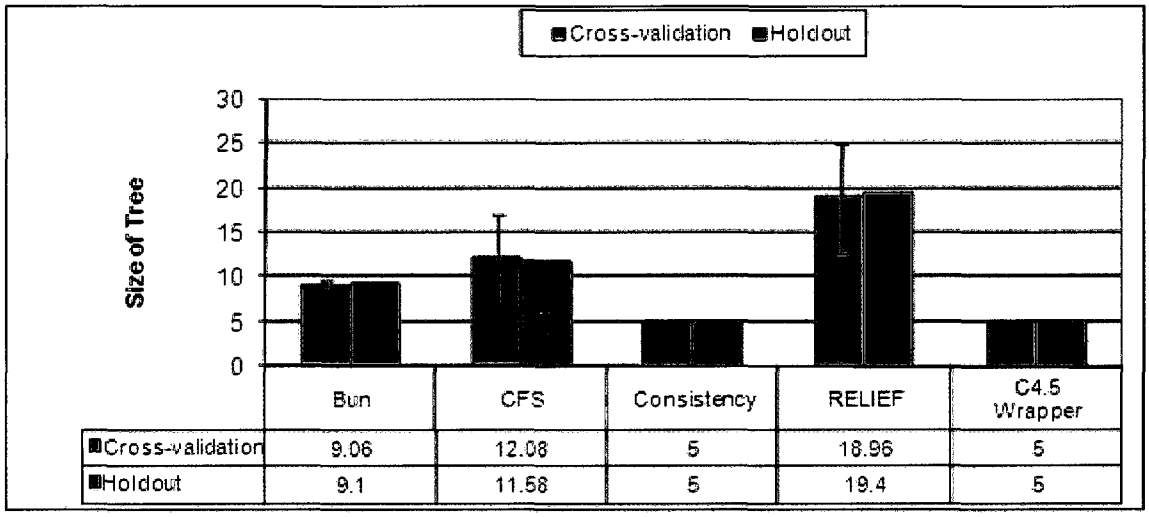


Figure 5.12. Average size of trees produced by C4.5 (Seeded buns)

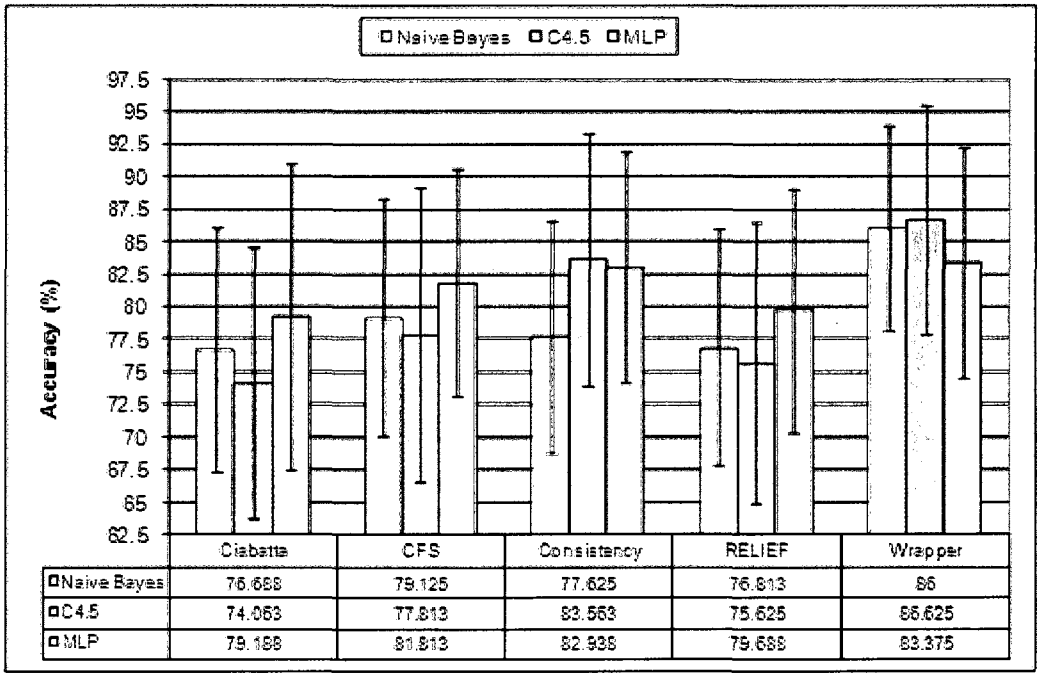
Knowing from section 5.3.1 that C4.5 achieves highest accuracy when operating over the full set of features, it would be reasonable to consider this combination as the ideal classifier in terms of accuracy. The average size of the generated trees is also relatively small, but slightly higher than the C4.5 trees generated after C4.5 wrapper or the consistency-based feature selection. The second option involving the classifiers generated from training C4.5 over a dataset containing only the features derived from consistency-based feature selection will operate a bit faster as they should have a tree in general of smaller size, but will be in average 0.55% less reliable in terms of accuracy performance. Consistency-based feature selection is proposed over the C4.5 wrapper because even though both feature selection techniques produce in average the same accuracy performance and the same size of C4.5 trees, the consistency-based method is a filter-based feature selector, and therefore requires less computational resources than the C4.5 wrapper during the feature selection process. There would be no need to consider MLP over C4.5 as a learning scheme because the former takes longer to train, longer to test, and achieves lower accuracy performance. Naïve Bayes is neither very competitive over C4.5 as it achieved the worst accuracy performance of all three evaluated machine learning techniques. The fact that C4.5 decision tree inducer achieves the highest accuracy estimates, the lowest standard deviation in terms of accuracy and the fastest classification time makes it supersede Naïve Bayes and MLP as potential machine learning candidates for automated classification of seeded buns. In the case of seeded buns also, the fact that the preconfigured classifier on the vision-based food inspection system is built more like a decision tree could account for the superior performance of C4.5 decision tree. Cross-validation and holdout accuracy estimation tests globally converged to the same conclusions for seeded buns model selection.

5.4 Ciabatta Buns and Model Selection

5.4.1 Classification Accuracy for Ciabatta Buns

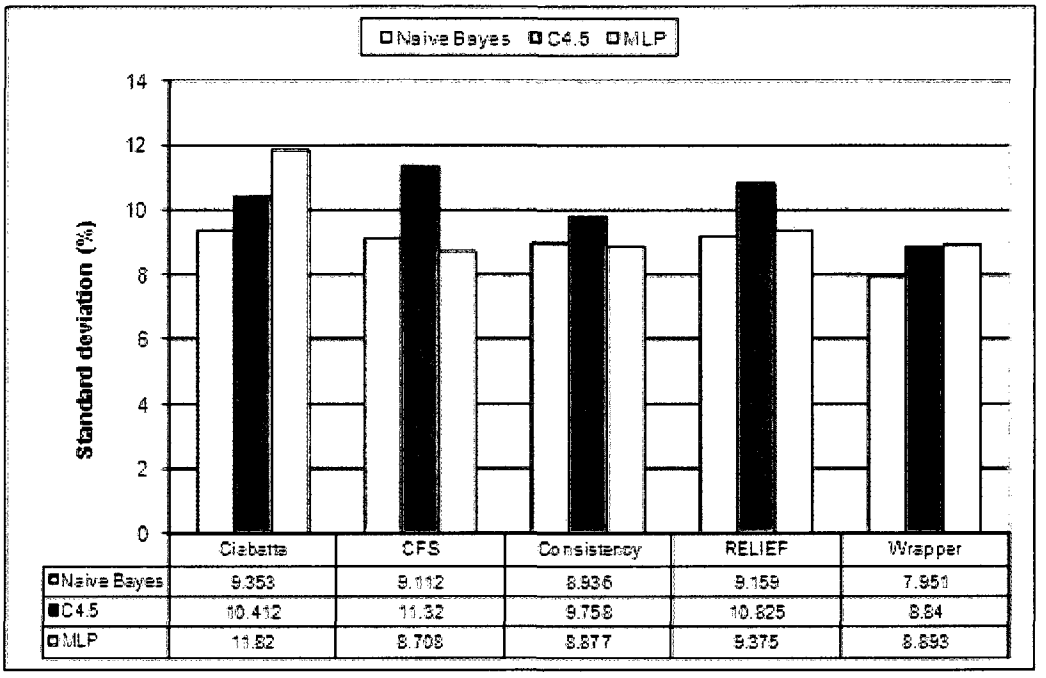
Figure 5.13 and Figure 5.15 respectively show the average results over 10 repetitions of 10-fold cross-validation accuracy estimation and the average results over

100 repetitions of holdout accuracy estimation for the ciabatta buns. Figure 5.14 plots the standard deviation of the estimated accuracy over the 10 repetitions of 10-fold cross-validation, whereas Figure 5.16 plots the standard deviation of the accuracy estimated by the 100 repetitions of holdout. Because there are fewer samples of ciabatta buns available than tortillas and seeded buns, holdout accuracy estimation for ciabatta buns uses for each iteration two third of the data for training and the remaining third for test instead of the 90% for training and 10% for test as proceeded with tortillas and seeded buns. In the case of ciabatta buns, the "ground truth" corresponding to 100% accurate classification is given by the subjective classification of a consumer as explained in section 5.1.



	Ciabatta	CFS	Consistency	RELIEF	Wrapper
Naive Bayes	76.688	79.125	77.625	76.813	86
C4.5	74.063	77.813	83.563	75.625	86.625
MLP	79.188	81.813	82.938	79.688	83.375

Figure 5.13. Average over 10 repetitions of the 10-fold cross-validation accuracy estimation for the ciabatta buns dataset

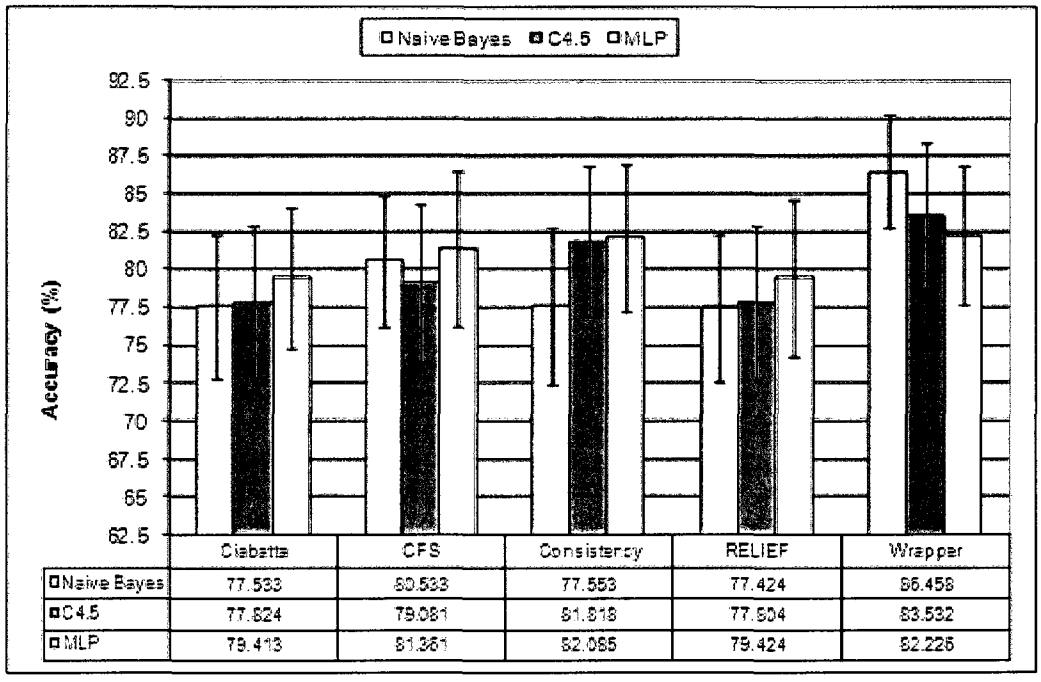


	Ciabatta	CFS	Consistency	RELIEF	Wrapper
Naive Bayes	9.353	9.112	8.936	9.159	7.951
C4.5	10.412	11.32	9.758	10.825	8.84
MLP	11.82	8.708	8.877	9.375	8.893

Figure 5.14. Standard deviation of 10 repetitions of the 10-fold cross-validation accuracy estimation for the ciabatta buns dataset

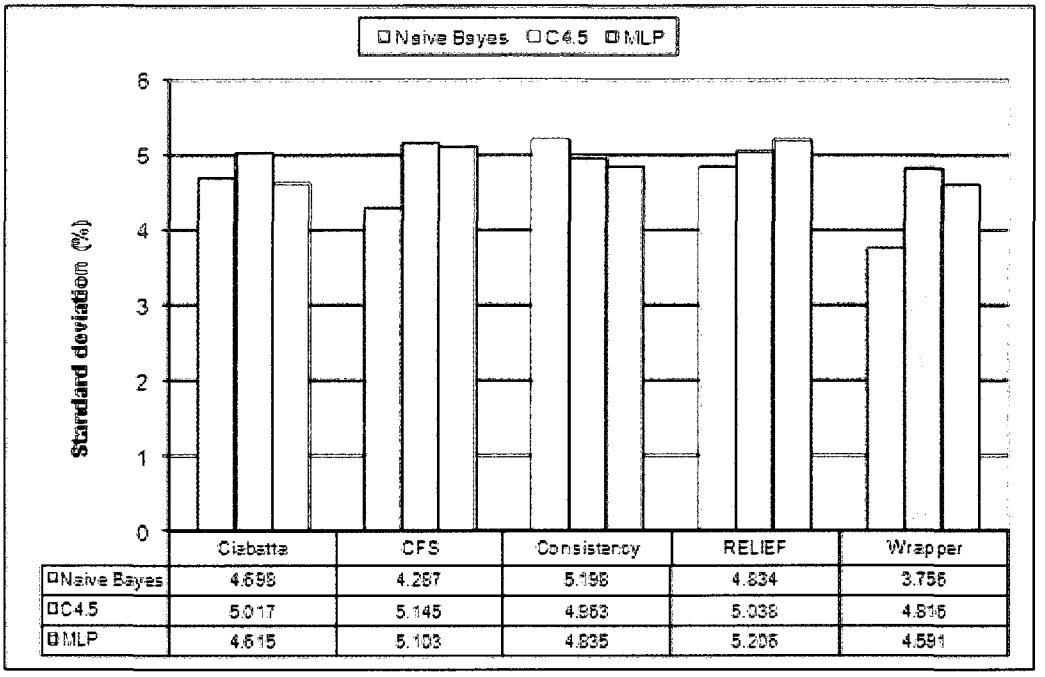
Unlike the tortillas and the seeded buns where C4.5 seems to generally perform better than the MLP and the Naïve Bayes learning schemes, results on ciabatta buns products do not unanimously exhibit the best learning scheme independently of the used feature selection. Nevertheless, both cross-validation and holdout test results show that the accuracy of almost all the reduced datasets is greater than or equal to the accuracy obtained with the full feature dimension dataset. In the case of the ciabatta buns, the accuracy estimation results for cross-validation and holdout do not converge to the same tendency as was generally witnessed in the case of the seeded buns and tortillas. The standard deviations of cross-validation results are a lot larger than those of holdout accuracy estimation: cross-validation's standard deviations are spread between 7.951% (Naïve Bayes trained over feature subset selected by Naïve Bayes wrapper) and 11.82% (MLP trained over the full feature set) while holdout tests standards deviation are mapped between 3.756% (Naïve Bayes trained over feature subset selected by Naïve Bayes wrapper) and 5.206% (MLP trained with the feature subset filtered by RELIEF). This observation indicates that the estimates of the holdout are more likely to be closer to

the averaged estimated accuracy compared to the estimates of the cross-validation method. The same observation could also explain the fact that cross-validation and holdout accuracy results diverge in the case of ciabatta buns. The fact that cross-validation gives poorer results compared to holdout for ciabatta buns could be attributed to the smaller number of ciabatta samples available. As a matter of fact, section 3.2.4 pointed out that 10 fold-cross-validation divides samples instances into 10 folds (or ten subsets of the samples), and trains over 9 folds to test on the remaining fold. If the number of samples is limited, the size of the folds could be very small; therefore fewer instances remain in the test fold, leading to poor or inconsistent accuracy estimation. For this reason, we will focus our analysis over the holdout accuracy estimation results presented in Figure 5.15 and Figure 5.16 where only 2/3 of the data is used for training and the remaining 1/3 for testing.



	Ciabatta	CFS	Consistency	RELIEF	Wrapper
Naive Bayes	77.533	80.533	77.553	77.424	86.458
C4.5	77.824	79.081	81.818	77.804	83.532
MLP	79.413	81.361	82.085	79.424	82.225

Figure 5.15. Average over 100 repetitions of the holdout accuracy estimation for the ciabatta buns dataset



	Ciabatta	CFS	Consistency	RELIEF	Wrapper
Naive Bayes	4.698	4.287	5.198	4.834	3.756
C4.5	5.017	5.145	4.953	5.038	4.816
MLP	4.615	5.103	4.835	5.205	4.591

Figure 5.16. Standard deviation of 100 repetitions of the holdout accuracy estimation for the ciabatta buns dataset

For all three learning algorithms, the ciabatta buns datasets that were reduced with a wrapper feature selection gave the greatest holdout accuracy estimation: 86.46% for the Naïve Bayes wrapper, 83.53% for the C4.5 wrapper and 82.23% for the MLP wrapper, which correspond respectively to 8.93%, 5.71% and 2.81% estimated accuracy enhancements compared to that of the full dataset. For both the C4.5 and the MLP learning schemes, the consistency-based attribute subset filter occupies the second rank with respect to the estimated accuracy from a reduced feature space, followed by CFS. For the Naïve Bayes Learner, CFS is the second best choice and the consistency-based filter the third choice. Recalling that Naïve Bayes assumes conditional independence between attributes, one could argue that the removal of correlated attributes by CFS improves the performance of Naïve Bayes. RELIEF is the last choice for all three candidate machine learning schemes. However, it is noticeable in Figure 5.15 that even though RELIEF performs the least in terms of estimated classification accuracy for a product for which the inspection system is not finely tuned, the feature subset selected by RELIEF still provides better accuracy than when all the features are considered by an

MLP classifier, and respectively only 0.02% and 0.01% lower accuracy when a C4.5 classifier or a Naïve Bayes classifier is considered. The holdout accuracy estimations show that Naïve Bayes generally gives more confident results than C4.5 and MLP as the standard deviation of Naïve Bayes seen on Figure 5.16 is the lowest of the three learning schemes, followed respectively by C4.5 and MLP's standard deviation.

5.4.2 Training Time and Testing Time for Ciabatta Buns

As proceeded for tortillas in section 5.2.2 and for seeded buns in section 5.3.2, this section presents the time-related performance of Naïve Bayes, C4.5, and MLP learning schemes over the full feature dimension dataset and the feature-reduced datasets resulting from the application of CFS, consistency-based subset selection and the wrapper-based feature selection. As justified in section 5.4.1, holdout accuracy estimation results will serve as the basis for analysis. The reader is referred to Appendix G for time-performance results exhibited by cross-validation accuracy estimation. Figure 5.17 plots the training time performance in seconds of the three evaluated machine learning techniques when trained from the original feature dimension ciabatta buns dataset and from the datasets which dimensions have been reduced with the four feature selectors. Here also, MLP training times are plotted in decimal logarithmic scale in order to be viewed on the same graph as Naïve Bayes and C4.5 training times.

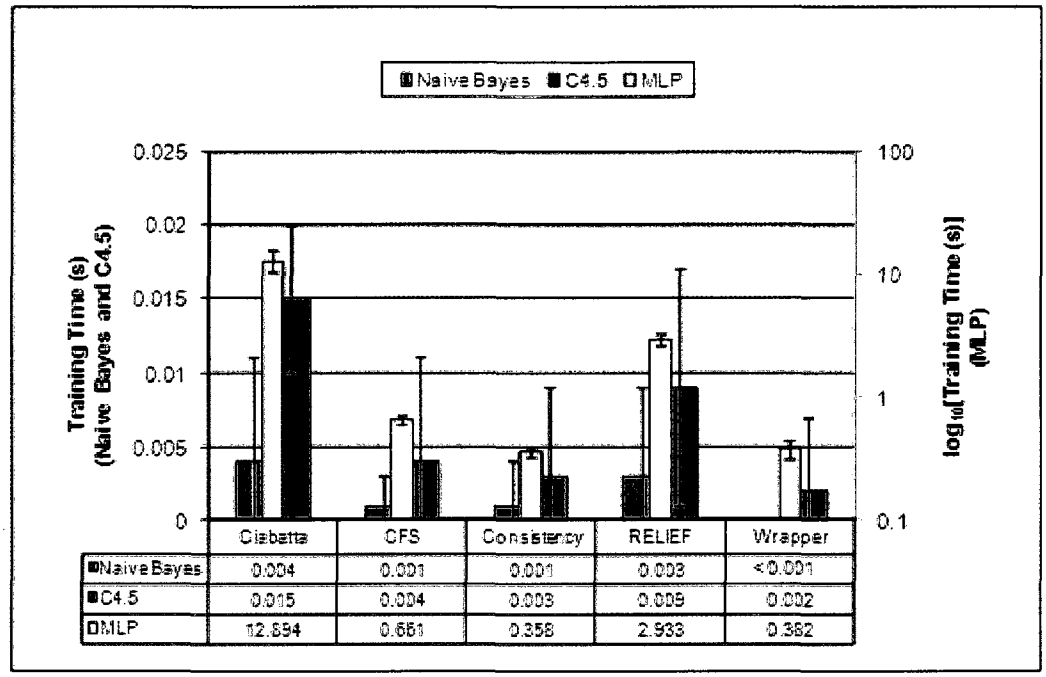
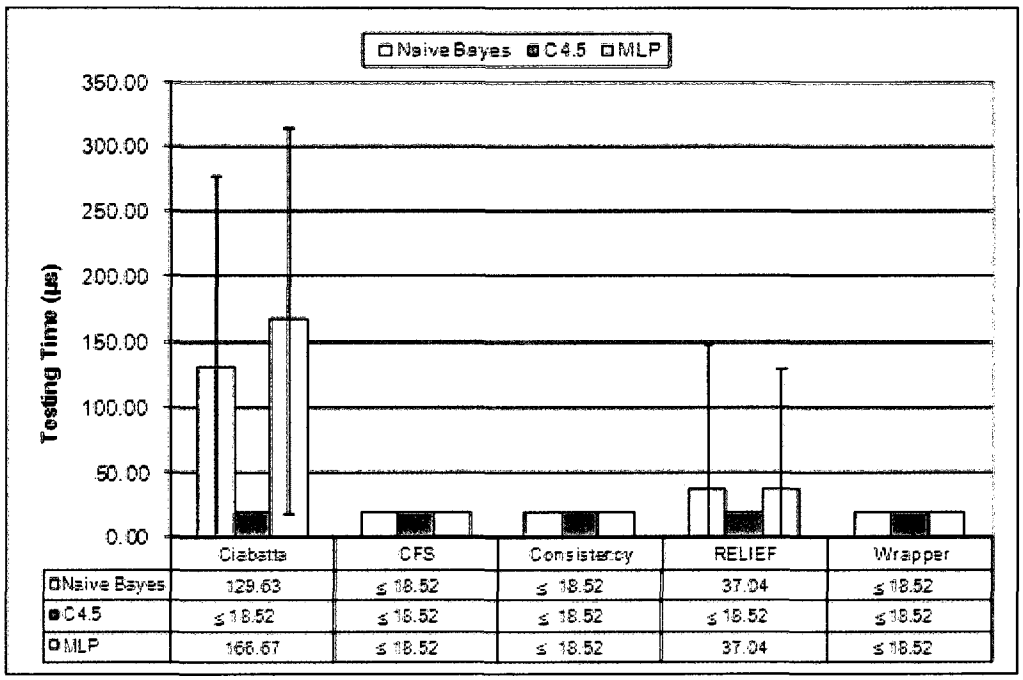


Figure 5.17. Average training time over 100 repetitions of the holdout accuracy estimation for the ciabatta buns' dataset

Whether feature selection is achieved or not prior to learning, Figure 5.17 shows that Naïve Bayes trains the fastest, followed by C4.5 and far behind by MLP. At its best performance, Naïve Bayes trains in less than 0.001 second over the dataset reduced by Naïve Bayes wrapper (dataset of four features as indicated in Table 4.4, while training in only 0.004 second in its worst performance when no feature selection is applied. Naïve Bayes also takes an average training time of 0.001 second over the 9 features suggested by the consistency method and the 14 features proposed by CFS (Table 4.4). Naïve Bayes learns the 36 features dataset produced by RELIEF in about 0.003 second in average. The second fastest learner, the C4.5 decision tree inducer takes about three times the training time of Naïve Bayes for the same datasets. C4.5 also achieves its best training time performance when learning the four features dataset reduced by the C4.5 wrapper (0.002 second) and its worst performance over the full 82 features dataset (0.015 second). The slowest learner, namely the MLP, takes approximately the same time to train the two datasets of 9 features built from consistency-based feature selection (0.358 second) and the MLP wrapper (0.382 second). MLP achieves its fastest training time

when operating over the features selected by the consistency-based method (0.358 second, or 358 times the training duration of Naïve Bayes for the same dataset) and the slowest training time when learning over the whole attributes list (12.894 second, or more than 3000 times the duration of Naïve Bayes for the same dataset).

The time taken by a classifier to classify an unknown instance of a product directly impacts the rate of productivity in the sense that it actually relates to how fast the system can process a product once the features have been extracted. Figure 5.18 below shows the average testing time per product taken by the five classifiers built by each of the three learning schemes.



	Ciabatta	CFS	Consistency	RELIEF	Wrapper
Naive Bayes	129.63	≤ 18.52	≤ 18.52	37.04	≤ 18.52
C4.5	≤ 18.52	≤ 18.52	≤ 18.52	≤ 18.52	≤ 18.52
MLP	166.67	≤ 18.52	≤ 18.52	37.04	≤ 18.52

Figure 5.18. Average testing time over 100 repetitions of the holdout accuracy estimation for the ciabatta buns' dataset

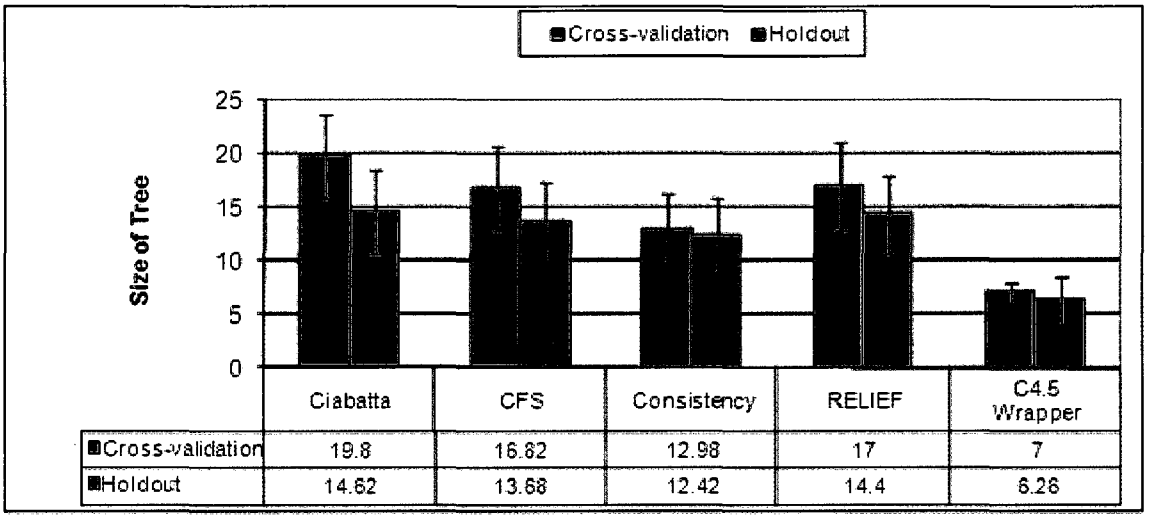
Except when RELIEF feature selection is considered and except the full set of features, all the classifiers generated by the three learning schemes were able to test an unknown instance in an average time of 18.52µs or less. C4.5 champions Naïve Bayes and MLP learning schemes in terms of testing time when the full set of features or the features

filtered by RELIEF are considered. For all the datasets, with or without feature selection, the classifiers built by C4.5 decision tree are able to classify each dataset in 18.52μs or less. For the datasets with a reduced number of features via RELIEF feature selection, Naïve Bayes and MLP classifiers take approximately the same duration to classify an instance of the 14 features dataset filtered by RELIEF (37.04μs). As usual, the dataset with the whole set of features makes the classifiers take the longest time for deciding the class membership of an unknown product instance under inspection: Naïve Bayes classifies an instance from the full feature dimension dataset in approximately 129.63μs whereas MLP takes an average testing time of 166.67μs/sample, that is almost 30% longer than Naïve Bayes testing time. We could have predicted Naïve Bayes to test faster on a subset of features compared to the full set of features because this Bayesian learning scheme's testing time complexity is $O(mk)$ for m test instances of a dataset containing k features; which means that the less features to analyze, the faster Naïve Bayes operates. This analogy also holds for the MLP network where less features to analyze means less input nodes in the MLP input layer, and therefore fewer processing units (also known as neurons or nodes) in the overall MLP network structure, which in turn consequently requires the MLP to process an instance of a product faster. However, this analogy does not hold for decision trees because the testing time increases with the size of the induced decision tree, which makes the testing time similar with or without feature selection.

5.4.3 Model Selection for Ciabatta Buns

We have seen in section 5.4.1 that unlike the case of tortillas and seeded buns where there was a clear machine learning algorithm which seems to perform better than the rest when the same sets of features are considered regardless of the feature selector, the performance of the evaluated machine learning schemes depends also on the feature selector used in the case of the ciabatta buns. Figure 5.15 clearly showed that in terms of predicted accuracy performance, each of the three experimented machine learning techniques reaches its maximum when operating over the only features selected by the wrapper feature selector trained to work with that target learning algorithm. If

classification accuracy is the most important objective of the ciabatta buns classification, Naïve Bayes classifier operating over the features selected by Naïve Bayes wrapper would be the best candidate combination of a classifier and a feature selector as this combination simultaneously gave the highest average classification accuracy and the lowest standard deviation. This combination also proposes the smallest number of features needed to classify an unknown instance (only 4 features as reported in Table 4.4) and the second fastest testing time out of the three machine learning techniques. C4.5 decision tree classifies the dataset of 4 features selected by C4.5 wrapper faster than Naïve Bayes, but with an accuracy in average 3% less than that of Naïve Bayes classifier. The trees generated by C4.5 from the features selected by the C4.5 wrapper actually happen to be of the smallest size among all the trees generated by the same learning scheme over the other datasets which did endure feature selection or not, as illustrated on Figure 5.19 below.



	Ciabatta	CFS	Consistency	RELIEF	C4.5 Wrapper
Cross-validation	19.8	16.82	12.98	17	7
Holdout	14.62	13.68	12.42	14.4	6.26

Figure 5.19. Average size of trees produced by C4.5 (ciabatta buns)

In the case of the ciabatta buns, the two best choices in terms of accuracy, testing time and number of features analyzed involve the usage of wrapper-based feature selection. Two major disadvantages of feature wrappers are the time taken during the feature selection process because of the repetitive interaction with a target learning algorithm, and of course the fact that the selected features are tailored for that target algorithm only,

imposing the feature selection process to be redone every time the target learning algorithm changes (section 3.2.4). If for one of those reasons the operator of the vision-based food inspection system decides to exclude the usage of wrapper-based feature selection techniques, then MLP and C4.5 reach their maximum performance accuracy when the set of evaluated features is filtered by the consistency-based feature selection (9 features) while Naïve Bayes thrives when operating over the 14 features filtered by CFS. In this scenario, MLP exhibits the highest average estimated accuracy when coupled with consistency-based feature selection, followed by C4.5 when joined with consistency, and by Naïve Bayes in conjunction with CFS. When the time taken to classify product instances is not the main burden, the classifier generated by MLP over the features selected by consistency would be the ideal candidate. On the other hand, the classifier generated by C4.5 over the features filtered by consistency would give the fastest classification time performance, but one would have to settle for less accurate classification than in the case of MLP. There would be no reason to consider the usage of Naïve Bayes with the features selected by CFS as the system could classify instances faster and with better accuracy when using C4.5 decision tree in conjunction with consistency-based feature selection. It is worth mentioning that the decision trees generated by C4.5 have the smallest size when wrapper feature selection is solicited (Figure 5.19). Among filter-based feature selectors, the consistency-based method gives the smallest tree for ciabatta buns, followed by CFS and RELIEF respectively. Note that even though RELIEF produces the largest C4.5 decision tree among all the feature selectors, this filter-based feature selector still produces trees in average smaller in size than the ones generated when no feature selection has been accomplished. It is also noticeable that for all the machine learning techniques evaluated on the ciabatta buns, the classifiers generated from a reduced feature subset achieve a better estimated accuracy performance (Figure 5.15), a faster training time (Figure 5.17) and a faster testing time (Figure 5.18) than when no feature selection at all is applied. Classifiers built from a reduced set of features also produce a smaller decision tree when C4.5 is the considered candidate classifier (Figure 5.19).

5.5 Chapter Summary

This chapter presented an experimental model selection for the automated inspection of three bakery products: tortillas, seeded buns, and "9 grain ciabatta" buns. The performance of three machine learning techniques, namely the C4.5 decision tree learning, the Naïve Bayes learning, and the Multi-Layer Perceptron learning, has been evaluated based on some proposed key performance indicators. The impact of feature selection on the different machine learning schemes has also been assessed. Experimental results from two different performance evaluation techniques known as *t*-fold cross-validation and test sample estimation (holdout) showed that the usage of feature selection in conjunction with machine learning techniques generally allows faster processing while achieving comparable or even higher accuracy performance in the classification. Experimental evaluation also indicated that instead of looking for the best learning scheme or the best feature selector suited for a certain type of product, one should look for the best combination of both which will meet his or her performance objectives. We could advance that for the experiments reported here, in general feature wrappers select fewer features compared to filter-based features selectors but are obviously drawn back by the time taken during their feature selection process which interacts with a target learning algorithm. Among the feature filters, consistency-based feature selection has shown superiority several times for the evaluated products. Naïve Bayes has proven to be a faster learner than C4.5 and MLP, while C4.5 inspects faster once a classifier model is learned. MLP is the slowest learner and the slowest classifier of all three machine learning schemes. Because the testing time complexity of Naïve Bayes is $O(mk)$ for m test instances of a dataset containing k features, it is clear how reducing the number of features analyzed can directly make this learning scheme operate faster. We have also observed that MLP operates faster after feature selection because a smaller number of input nodes involves a smaller network of neurons. On the other hand, we have noted that the testing time of C4.5 decision trees correlates with the size of the induced tree, which remains the same with or without feature selection. Comparative evaluation of cross-validation and holdout as accuracy estimation tests shows that both methods globally converge to the same conclusion, provided that enough product

samples are made available. The whole experimental evaluation of predicted performance of the candidate models can be done automatically by the food-inspection system itself. Table 5.1 captures the summary of the recommended models for classifications, according to the three key performance indicators established in section 5.1. The process presented in this chapter basically allows the vision-based food inspection system to auto-evaluate its predicted performance against several models of inspection, and to propose the results of the objective evaluation to an operator. The next chapter will push the research further by actually explaining how the vision-based food inspection system can auto-tune itself based on the model selected by the operator according to his or her business or performance objectives.

Table 5.1. Summary of model selection

	Tortillas		**Seeded Buns**		**Ciabatta Buns**	
Priority	Feature selector	Learning scheme	Feature selector	Learning scheme	Feature selector	Learning scheme
Accuracy	Consistency	C4.5	No feature selection	C4.5	Naïve Bayes wrapper	Naïve Bayes
Testing Time	C4.5 wrapper	C4.5	C4.5 wrapper	C4.5	C4.5 wrapper	C4.5
Training Time	CFS	C4.5	Consistency	C4.5	Naïve Bayes wrapper	Naïve Bayes

Chapter 6. Auto-Tuning

Chapter 5 has proposed a process for formally selecting a classification model for a vision-based food inspection system based on the auto-evaluation of the system of its predicted performance. The current chapter closes the loop of the auto-tuning process by studying cases where the system is actually tuned with the model selected by a human operator. The first section of this chapter will explain the experimental methodology adopted to evaluate the real performance of the proposed inspection models. The second section will study the behavior of the system once tuned with some of the models recommended for the two bakery products for which the system already possesses a tuned classifier (tortillas and seeded buns as discussed in sections 5.2 and 5.3). The second section will also experiment with a new bakery product for which the system has never been tuned to inspect, namely hot dog buns. The last section will reconcile and summarize the analysis of the predicted performance of the food inspection system against the actual performance achieved once tuned with a model. Only C4.5 decision tree models have been tuned into the system for the purpose of the work reported here. The choice of C4.5 learning scheme over Naïve Bayes and MLP is clearly justified by Table 5.1 which predicts C4.5 to be the best learning scheme for tortillas and seeded buns in terms of accuracy, classification time, and testing time. C4.5 is also predicted to be the learning scheme which classifies an unknown instance of ciabatta bun in the shortest time.

6.1 Experimental Methodology

Chapter 5 provided means of predicting the performance of a classification model which involve averaging the performance results over repetitions of cross-validation or holdout test sample estimation. Therefore when we refer to *predicted accuracy*, we make allusion to the accuracy performance of a specific classification model during either cross-validation or holdout. On the other hand, *real accuracy* refers to the accuracy

performance achieved by the vision-based food inspection system once tuned with a specific classification model. In order to tune the vision-based food inspection system with a model for visual classification of bakery products, we adopted the following steps, after feature selection (Chapter 4) and evaluation of predicted accuracy of C4.5 (Chapter 5) with and without feature selection:

1. Select the model which achieved the highest predicted accuracy for every combination of feature selector (including no feature selection) and C4.5 learning scheme,
2. Identify the leaves of the C4.5 decision tree that order rejection (or those that order acceptance) of the product,
3. Tune each of those rules as the model for classification in the vision-based food inspection system,
4. Experiment the tuned model with new samples that have not been used during the training process,
5. Compare the predicted accuracy with the real accuracy achieved by the system operating with the tuned classifier.
6. If the system already has previously been tuned for that type of product under classification, then compare the features formally selected by the proposed process to those initially considered relevant by the initial classifier (which have been previously derived by trial-and-error).

6.2 Inspection Model Auto-Tuning

This section covers the auto-tuning of two bakery products for which the system already has classification rules implemented (tortillas and seeded buns presented in section 4.2.2), and a third bakery product that the system has never been calibrated to inspect before (hot dog buns).

6.2.1 Auto-Tuning Classification Model for Tortillas

After evaluating the performance of C4.5 decision tree with and without feature selection, the decision tree achieving the highest predicted accuracy is retained as the best candidate classification model, for each of the feature selectors (including no feature selection). Figure 6.1 shows the actual C4.5 decision tree which achieved the highest classification accuracy when the full set of features is considered. This tree of size 21 has 11 leaves. Recall from section 5.1 that the size of a tree corresponds to the number of branches in the tree plus one for the top node. 7 out of those 11 leaves are labeled "reject" (in red) and the remaining 4 leaves are labeled "accept". Each of these leaves correspond to the rules for which the inspection system should classify a product instance as defective if the values of the measured features satisfy the conditions taking to a leaf labeled "reject" from the root node, or to classify a product as non-defective if its features fall into a leaf labeled "accept". The first rejection rule is as simple as "reject the tortilla if parameter M79 is greater than 2.599511" (first leaf from the top node). If the seven rules corresponding to leaves labeled "reject" are tuned, then the inspection system should reject any tortilla whose combinations of evaluated features fall into any of the seven leaves that are labeled "reject". One also has the option of tuning instead only the four rules representing the four leaves labeled "accept". In that scenario, the inspection system should reject any product instance which features values do not satisfy any of the four "acceptance" rules. When the rules from the rejection leaves are tuned, the system operates by rejecting any product that satisfies at least one rejection rule. On the other hand, if the rules from the acceptance leaves are tuned, the inspection system operates by rejecting any product that does not satisfy at least one acceptance rule. One should tune the rules that require less computation. One can observe from Figure 6.1 that even though the decision tree was generated from the full set of features, only seven distinct attributes appear in the decision tree (M79, M17, M23, M24, M59, M25, and M20). Somme features do not appear because C4.5 typically builds its decision tree from a subset of the available features that it finds relevant as mentioned in section 2.3.2.2.

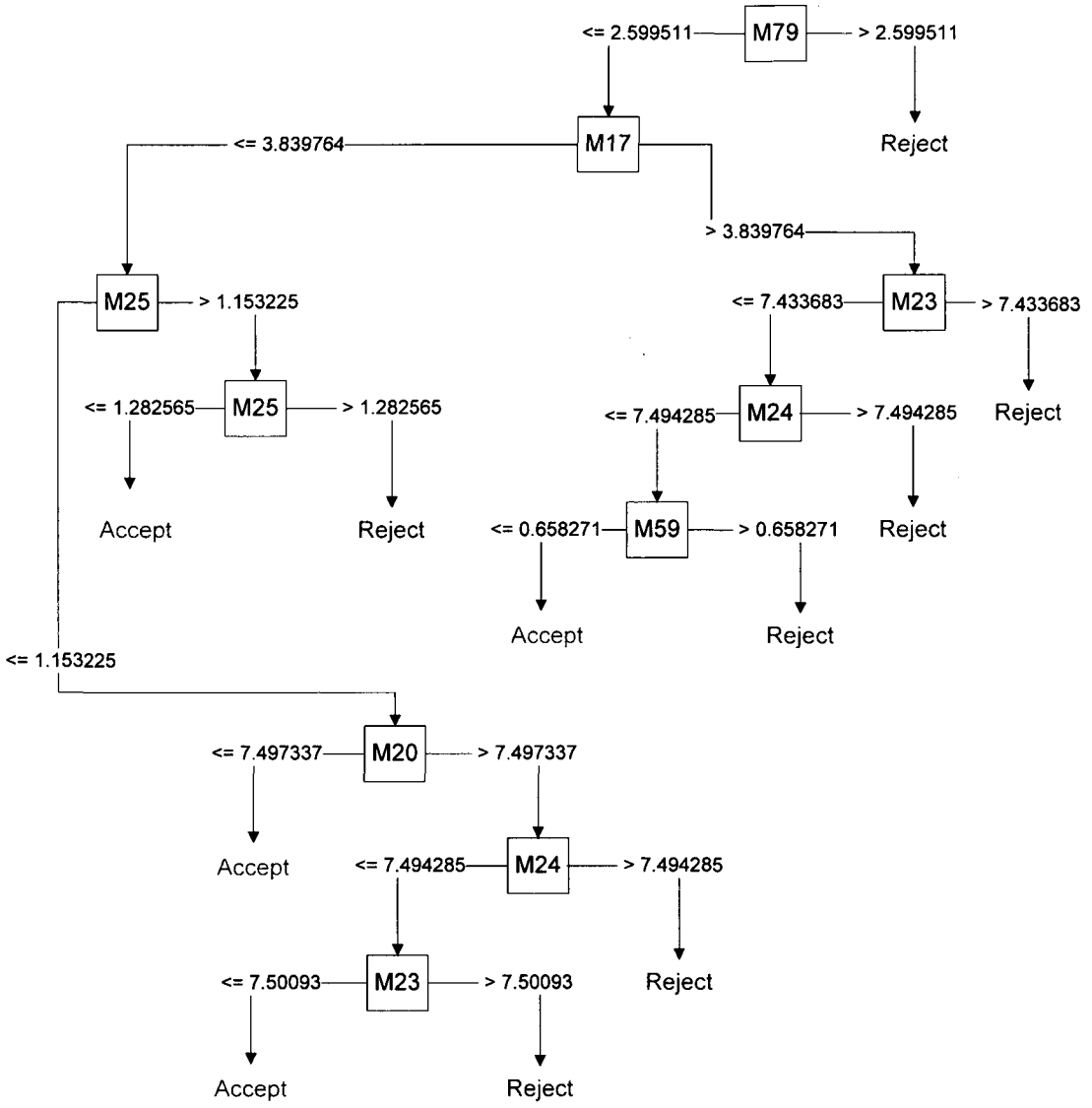


Figure 6.1. C4.5 decision tree induced from the full set of features (tortillas)

Because the vision-based food inspection system allows an operator to enter custom set of rules, we created the following seven rules corresponding to the 7 rejection leaves, where IIF means *if and only if* and M201 to M207 are parameters that are either true or false depending on the result of the corresponding IIF statement. Note that we could have tuned the only 4 acceptance rules.

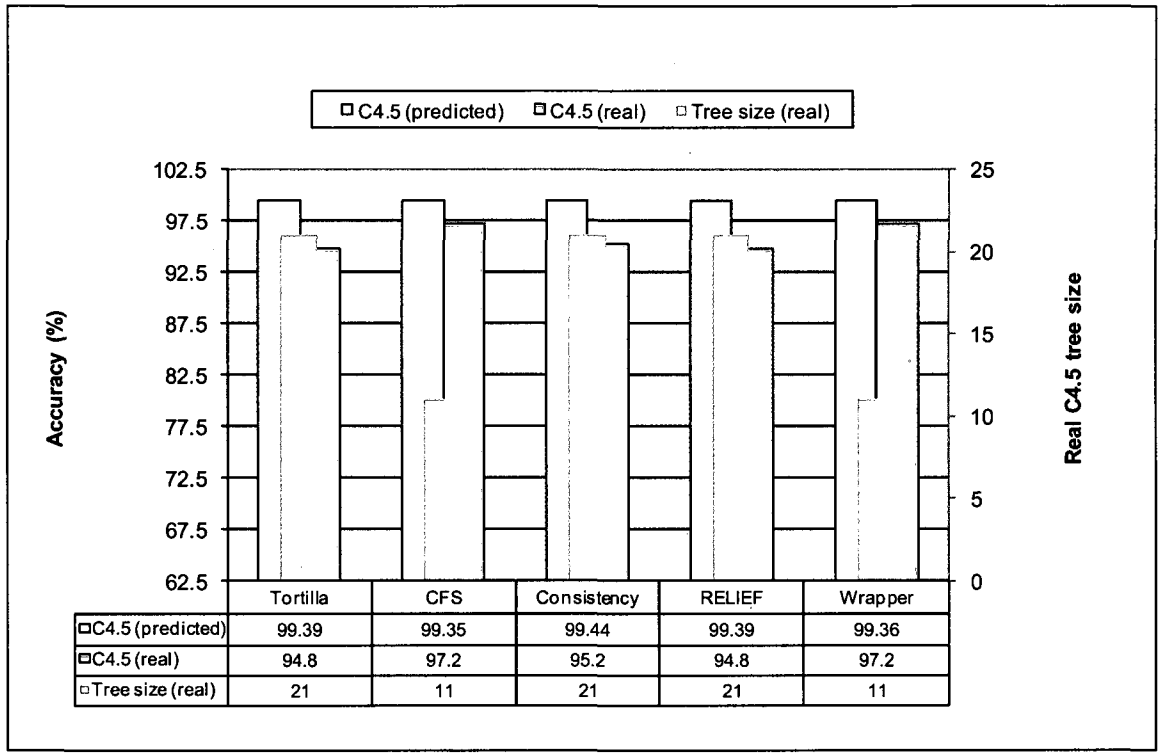
Table 6.1. Example of auto-tuned rules for tortillas classification (based on full set of features)

M201:	IIF((M79 > 2.599511), True, False)
M202:	IIF((M79 <= 2.599511) AND (M17 > 3.839764) AND (M23 > 7.433683), True, False)
M203:	IIF((M79 <= 2.599511) AND (M17 > 3.839764) AND (M23 <= 7.433683) AND (M24 > 7.494285), True, False)
M204:	IIF((M79 <= 2.599511) AND (M17 > 3.839764) AND (M23 <= 7.433683) AND (M24 <= 7.494285) AND (M59 > 0.658271), True, False)
M205:	IIF((M79 <= 2.599511) AND (M17 <= 3.839764) AND (M25 > 1.153225) AND (M25 > 1.282565), True, False)
M206:	IIF((M79 <= 2.599511) AND (M17 <= 3.839764) AND (M25 <= 1.153225) AND (M20 > 7.497337) AND (M24 > 7.494285), True, False)
M207:	IIF((M79 <= 2.599511) AND (M17 <= 3.839764) AND (M25 <= 1.153225) AND (M20 > 7.497337) AND (M24 <= 7.494285) AND (M23 > 7.50093), True, False)

The rules above mean that the vision-based food inspection system should reject a tortilla whenever any of the parameters M201 to M207 has a value of "True". Note that all those rules are mutually exclusive. For all the decision trees induced from each of the feature selectors, the same approach is followed and the rules corresponding to each decision tree are tuned into the vision-based food inspection system.

Once the models are tuned into the system, 500 new samples of tortillas were inspected by each of the five models: C4.5 decision tree induced from the full set of features, CFS, consistency-based feature selection, RELIEF, and C4.5 wrapper. The 500 samples were first analyzed by the classifier initially tuned by the bakery producing those tortillas in order to obtain the correct classification (in the sense of the bakery expectations) that their system would have produced. For every product that is inspected with the new tuned model, its classification is considered accurate if it matches the classification from the bakery's initial classifier, and inaccurate otherwise. The "ground truth" corresponding to 100% accuracy is therefore represented by the classification

produced by the bakery's initial classifier as explained in section 5.1. The same 500 tortilla samples were presented to every C4.5 tuned-model. Figure 6.2 below plots the real accuracy achieved by the auto-tuned system against the accuracy predicted by cross-validation for the tortillas. On Figure 6.2, "Tortilla" represents the full set of features without any feature selection. The sizes of the C4.5 decision trees tuned in the system are also plotted on a secondary axis.



	Tortilla	CFS	Consistency	RELIEF	Wrapper
C4.5 (predicted)	99.39	99.35	99.44	99.39	99.36
C4.5 (real)	94.8	97.2	95.2	94.8	97.2
Tree size (real)	21	11	21	21	11

Figure 6.2. C4.5 predicted accuracy vs. real accuracy (tortillas)

For all the considered models, the real accuracy achieved is a bit lower than the accuracy predicted by cross-validation. However, the maximum difference between the predicted accuracy and the real accuracy (with respect to the tortillas bakery model) is 4.59% and corresponds to the C4.5 model generated based on the full set of feature and also to the C4.5 model induced based on features selected by RELIEF (which happens to be the feature selector removing the least attributes as was concluded in section 4.3.3). The decision tree based on CFS achieved 2.15% lower accuracy in practice, the C4.5 tree induced based on C4.5 wrapper feature selection over-predicted the real accuracy by

about 2.16%. The C4.5 decision tree induced after consistency-based feature selection achieved an accuracy about 4.5% lower than its predicted accuracy. In practice, C4.5 wrapper and CFS achieve the best accuracy performance for C4.5. Not only do those two feature selectors give a better accuracy in practice, but also the C4.5 decision trees that maximize their accuracy estimates have a smaller size than when RELIEF or consistency-based feature selection are considered, or when no feature selection at all is achieved. Recall that smaller decision trees are usually preferable because they are easier for a human operator to interpret. Smaller decision trees also obviously have relatively less leaves, and therefore this implies that there are less classification rules to tune into the vision-based food inspection system. For instance, Figure 6.3 below shows the C4.5 model induced after wrapper-based feature selection which is incontestably more compact and involves only four rejection rules to tune (or two acceptance rules), compared to seven rejection rules (or four acceptance rules) when the full set of features is considered.

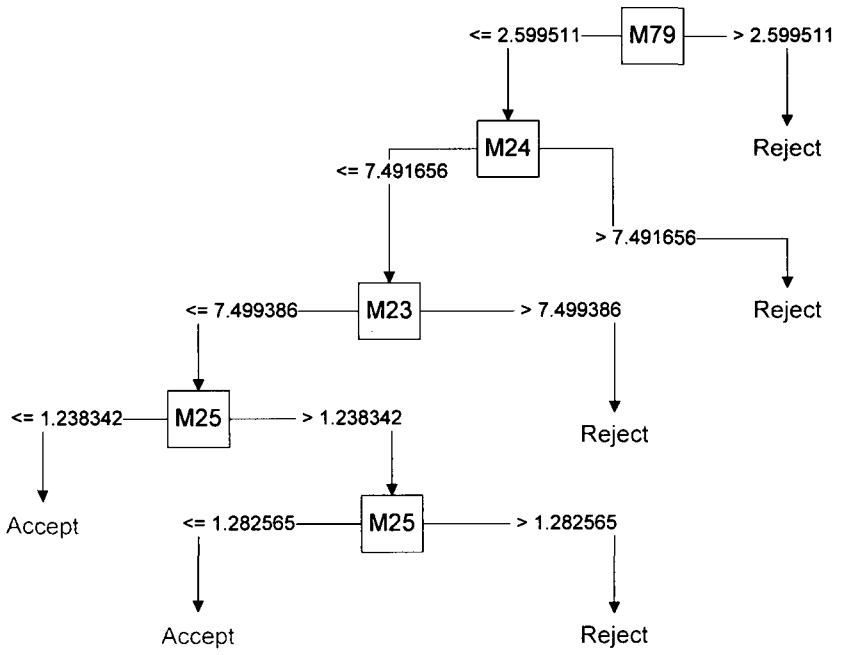


Figure 6.3. C4.5 decision tree induced after wrapper feature selection (tortillas)

Even though the order of performance of the different feature selectors combined with C4.5 decision tree learning is slightly different when the system operates with auto-tuned rules (compared to the order of performance predicted by 10-fold cross-validation), the experimental results show that all the C4.5 models generated after feature selection achieve an accuracy greater or equal to when the full set of features is considered.

The last step of our experiments for the tortillas dataset is to visualize and compare the features that have been formally retained as relevant by means of feature selection to those selected by trial-and-error by a human operator. Table 6.2 below shows the number of features judged relevant by the different feature selection techniques that also appear in the list of 9 features derived by trial-and-error for tortillas classification. Those 9 features are actually the features that were used to classify the tortillas with the initially preset classifier. All the classification accuracy performances presented for tortillas are compared to the classification produced by this preset classifier.

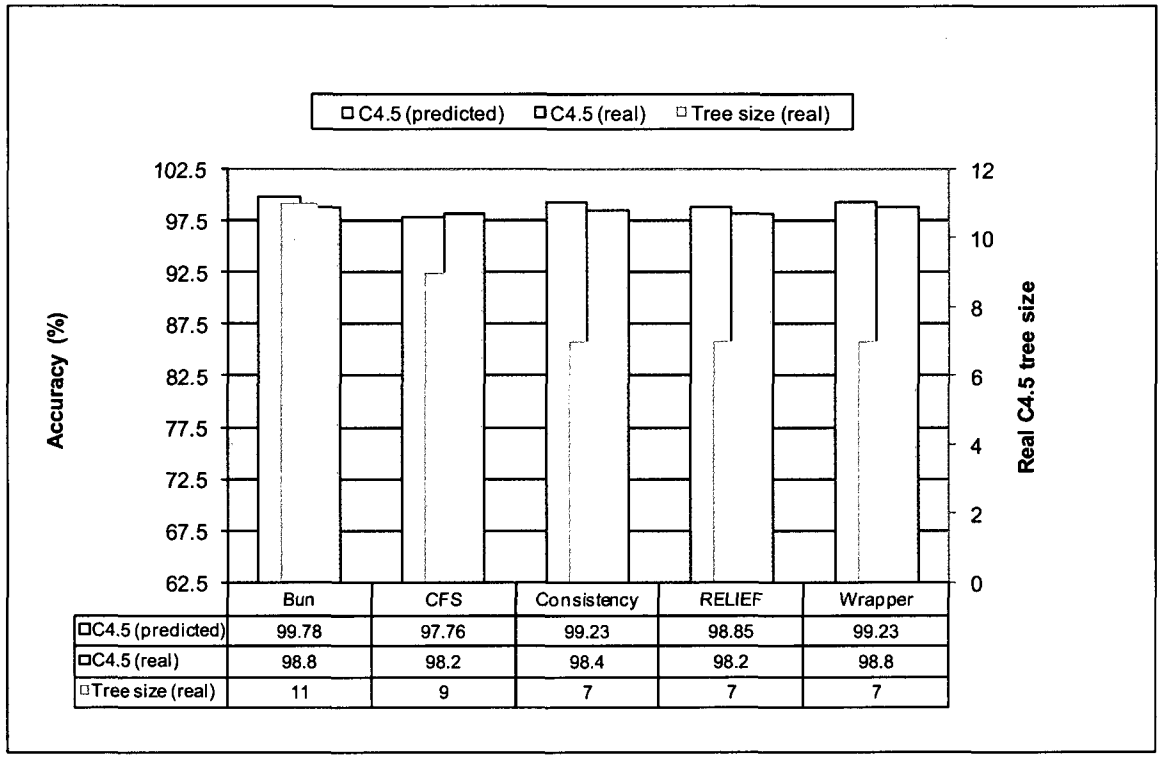
Table 6.2. Comparison of formally selected features and features selected by trial-and-error (*tortillas*)

	CFS	Consistency	RELIEF	Naïve Bayes wrapper	C4.5 wrapper	MLP wrapper
Number of matching features (formal feature selection / trial-and-error)	1 / 9	5 / 9	9 / 9	2 / 9	1 / 9	1 / 9
Total number of features (formal feature selection)	5	10	33	7	4	5

From Table 6.2, we can see that all the feature selectors have found some features that were initially considered relevant by a human operator. RELIEF has found all the attributes used in the preset model. Regardless of the number of matching features, the results from Table 6.2 and Figure 6.2 indicate that even though the features selected by trial-and-error are partially different from those formally selected as relevant, comparable or even greater accuracy can be performed with formal feature selection.

6.2.2 Auto-Tuning Classification Model for Seeded Buns

In order to tune a classification model for seeded buns, the exact same methodology was followed as in the case of tortillas. The experiment also involved 500 new samples of seeded buns that have not been used during the model learning phase. Figure 6.4 shows a comparison between the predicted accuracy performance of the C4.5 decision trees and their real accuracy performance achieved when the system is configured with the decision tree achieving the highest predicted accuracy. Here also, the size of the tuned tree is indicated for every model. In Figure 6.4, "Bun" refers to the dataset containing the full original set of features.



	Bun	CFS	Consistency	RELIEF	Wrapper
C4.5 (predicted)	99.78	97.76	99.23	98.85	99.23
C4.5 (real)	98.8	98.2	98.4	98.2	98.8
Tree size (real)	11	9	7	7	7

Figure 6.4. C4.5 predicted accuracy vs. real accuracy (seeded buns)

Observation of Figure 6.4 shows that all the accuracy performances achieved with the tuned vision-based food inspection system over the 500 new samples differ less than 1% from the accuracy predicted using cross-validation. The full set of features

achieves the best performance as was predicted by cross-validation (98.8% of the 500 new seeded buns samples were correctly classified compared to the predicted 99.78%). However, the second best choice for C4.5 decision tree learning according to our predictions (C4.5 wrapper) now reaches the same performance accuracy as with the full set of features. The decision tree generated after consistency-based feature selection achieves 98.4% correct classification with the 500 samples, which is 0.83 less than the predicted accuracy. The decision tree generated from the full set of features has the largest size (11), whereas the one based on C4.5 wrapper feature selection proposes the smallest size (7). The former tree had only three leaves ordering rejection of the seeded buns while the latter exhibited even fewer leaves modeling unacceptable seeded buns (two leaves). Because those two decision trees achieve the same accuracy performance, the smaller tree should be preferred because of the same reasons advocated in section 6.2.1.

When the features selected formally following our proposed process are compared to the 10 features considered relevant by a human operator, RELIEF is again the feature selector having the most features in common (7 out of 10) as illustrated by Table 6.3. Table 6.3 and Figure 6.4 show that formal feature selection can allow comparable performance even though the features selected are not necessarily the ones perceived as relevant by a human operator.

Table 6.3. Comparison of formally selected features and features selected by trial-and-error (seeded buns)

	CFS	Consistency	RELIEF	Naïve Bayes wrapper	C4.5 wrapper	MLP wrapper
Number of matching features (formal feature selection / trial-and-error)	2 /10	3 / 10	7 / 10	1 / 10	1 / 10	1 / 10
Total number of features (formal feature selection)	7	5	59	6	2	5

6.2.3 Auto-Tuning Classification Model for Hot Dog Buns

Another type of product for which we evaluated the performance of the system auto-tuned with the corresponding classification models, is hot dog buns. For this

particular type of bakery product, the vision-based system has never been configured for its inspection. This product belongs, like the ciabatta buns, to the category that we have called "unknown products" in section 4.2.2. Figure 6.5 shows an example of the aforementioned hot dog bun.

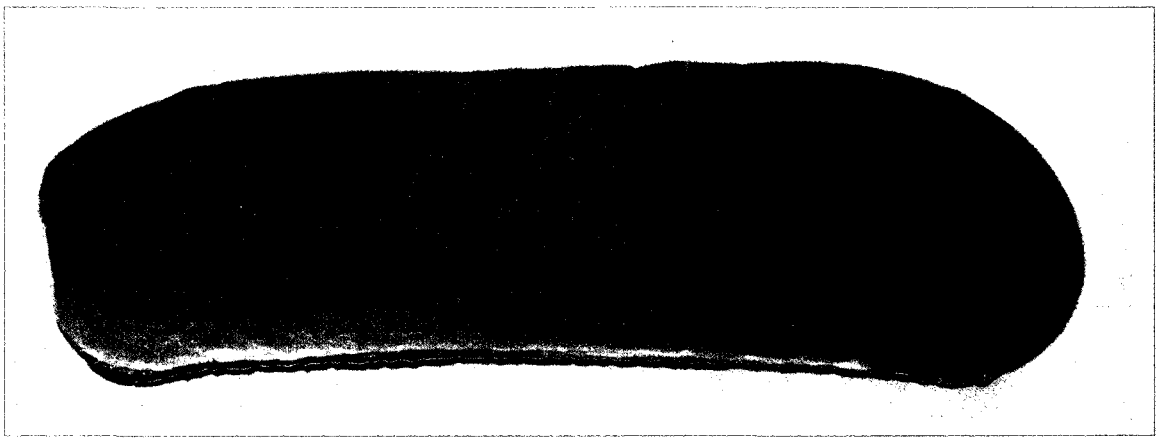


Figure 6.5. Example of experimented hot dog bun

Further experiments on these hot dog buns, which present a shape very different from the three other products experimented so far (tortillas, seeded buns, and "9 grain ciabatta buns"), provide a case where more variability exists in the types of products inspected. This variability creates an additional use case which will either validate or invalidate our conclusions on the capability of auto-tuning a vision-based food inspection system.

199 samples of hot dog buns were used to anticipate the performance of the three machine learning techniques with and without feature selection. As in the case of ciabatta buns, defects had to be created among the hot dog samples in order to create hot dog buns of unacceptable quality to the consumer. Among the 199 samples, 112 samples (56.28%) were initially graded acceptable and the remaining 87 samples (43.72%) were graded unacceptable by a consumer. The collection of the data is done using the second of the two running modes of the implemented data logging module which was presented in section 4.2.1. Table 6.4 below summarizes, sorted in ascending order of the number of features selected, the feature dimensionality performance achieved by the four evaluated

feature selectors: CFS, consistency-based feature selection, RELIEF and the wrapper technique (trained to learn with Naïve Bayes, MLP, and C4.5). The details of the selected features can be found in Appendix H.

Table 6.4. Feature space reduction on the hot dog buns dataset

	Number of features selected	Number of features rejected	Feature rejection ratio (%)
NB wrapper	2	80	97.56
C4.5 wrapper	3	79	96.34
MLP wrapper	10	72	87.80
CFS	12	70	85.37
Consistency	14	68	82.93
RELIEF	50	32	39.02

As we could have predicted, the feature wrapper technique selects the smallest set of feature as being necessary for hot dog buns inspection: Naïve Bayes wrapper retains only 2 features out of 82 (rejecting 97.56% of the features), C4.5 wrapper judges 3 features to be relevant (3.66% of the features), and MLP wrapper claims requiring 10 features out of 82 to classify hot dog buns with a maximum of accuracy. CFS necessitates 12 features out of 82 while consistency-based feature selection advocates for 14 features out of 82 to be necessary for correct classification. As usual, RELIEF feature selection is more careful and selects the largest number of 50 features out of 82 (60.98% of all the available features).

Regarding time related performance, we know by now that for the same set of features, Naïve Bayes learning scheme generally learns faster than C4.5, which in turn learns faster than MLP. We also know that because the learning process is done prior to the actual operation of the vision-based food inspection system, the learning time does not directly impact the rate of production. Per opposition to the training time, the time taken to actually classify an instance of a hot dog bun after learning and tuning a classification model directly impacts the rate of production because the classification is done during the real time inspection of the bakery products. Sections 5.2.2, 5.3.2, and 5.4.2 have shown that for the same set of features, C4.5 decision tree generally presents faster testing time than Naïve Bayes while MLP takes the longest to classify an instance.

For classification accuracy estimations, 100 hundred repetitions of test sample estimation were accomplished with 2/3 of the 199 hot dog buns for training, and the remaining 1/3 for testing as in the case of ciabatta buns (section 5.4). This accuracy estimations averaged over the 100 hundred repetitions of holdout, give the predicted accuracy performance of each of the three learning schemes (Naïve Bayes, C4.5, and MLP) when operating over the full set of features, the subset of features selected by CFS, those retained by consistency-based feature selection, RELIEF, and the feature wrapper (trained to work with each of the three learning schemes). During the repetitions of holdout, the model achieving the highest accuracy for every feature selector (including no feature selection at all) when C4.5 is the target learning algorithm is retained as the best candidate for C4.5 decision tree learning. The best candidate of each subset (and the full set) of features for C4.5 is then tuned in the vision-based food inspection system as was accomplished in sections 6.2.1 and 6.2.2. The next step consists of evaluating the real performance of the vision-based food inspection system tuned with the best C4.5 model for each of the four feature selectors and the full set of features. 100 new samples that have not been used in the model generation are presented to the tuned system for this evaluation.

Figure 6.6 shows the average accuracy predicted by the 100 hundred repetitions of holdout vs. the accuracy performance achieved by the vision-based food inspection when classifying previously unseen hot dog buns based on the tuned model (model from best C4.5 candidate during repeated holdout). The size of the C4.5 decision trees corresponding to each of the tuned C4.5 models is also displayed. In Figure 6.6 "Hot dog" represents the dataset containing the full set of features which did not undergo any feature selection whereas the vertical lines at the edge of average predicted accuracies represent the standard deviation of the predictions.

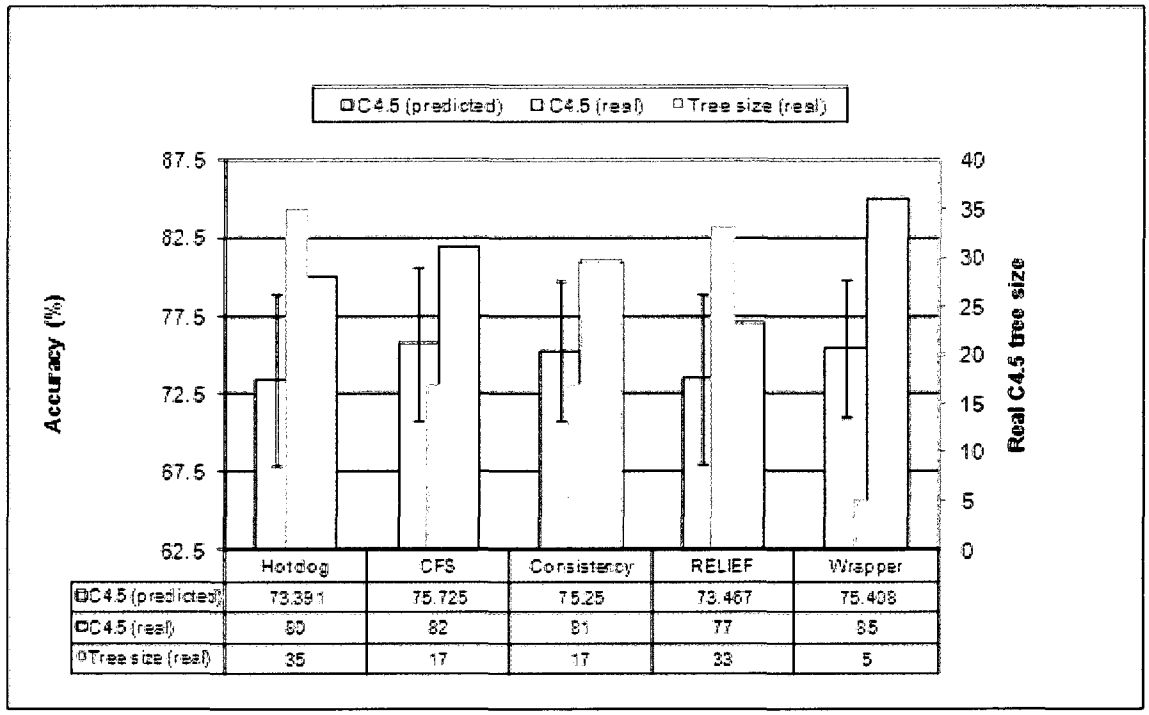


Figure 6.6. C4.5 predicted accuracy vs. real accuracy (hot dog buns)

From Figure 6.6, we can observe that the classification accuracy performance of the auto-tuned vision-based food inspection is significantly higher than the accuracy predicted by test sample estimation. This difference between predictions and reality could be explained by the fact that the predicted accuracy is averaged over 100 repetitions of holdout accuracy test whereas the real accuracy is evaluated with the model that achieved the best predicted accuracy. This observation implicitly indicates that among other models generated during the repeated holdout test, both the model predicting the worst accuracy and the one predicting the best accuracy contributed in computing the average predicted accuracy. The relatively large standard deviations of the predictions also indicate that the averaged predictions are spread out over a large range of values instead of being very close to the mean value; which could also account for the large difference between the predicted average accuracy and the accuracy achieved by the auto-tuned system. Accuracy wise, holdout predictions for C4.5 show CFS to perform 0.317% better than C4.5 wrapper feature selection (75.408%) whereas the tuned-vision-based food inspection system shows C4.5 feature wrapper to accomplish

85% correct classification, which is 3% above CFS feature selection. Both the holdout results and the food inspection system's results put consistency-based feature selection to be the third best candidate to use conjointly with C4.5 for hot dog buns inspection. While the accuracy of C4.5 decision trees averaged over repeated holdout cross validation shows that the full set of features performs the least, just behind RELIEF feature selection, the experiments on the best C4.5 decision trees induced from those two sets of features swaps their performance accuracy. As in the case of the ciabatta buns, the accuracy performance of hot dog buns is also lower than the one achieved with tortillas or seeded buns. This observation can be attributed to two major reasons: the first reason being the limited number of training samples, and the second being the fact that the initial distinction between acceptable and unacceptable hot dog buns samples has been done according to a human judgment, and not a machine. Therefore human judgment tends to show more inconsistency. In addition to showing poorer results compared to C4.5 wrapper, CFS, and consistency-based feature selection, the full set of features and RELIEF feature selection have also induced much larger C4.5 decision trees for hot dog buns. As a matter of fact, the C4.5 tree generated from the full set of features is of size 35, compared to only a size of 5 when wrapper feature selection is accomplished prior to inducing a C4.5 decision tree. The C4.5 tree generated from the full set of features required the tuning of 9 rejections rules in the vision-based food inspection system per opposition to only 2 rejection rules when the feature subset derived from C4.5 wrapper feature selection is considered.

6.3 Chapter Summary

This chapter aimed at validating the estimated accuracy of a combination of feature selector (or the full set of features) and a machine learning scheme. For this proof of concept, C4.5 decision tree learning was considered among the three presented machine learning techniques because of its superior performance observed in Chapter 5. After assessing the predicted performance of a feature selector combined with C4.5 decision tree learning, the model achieving the highest predicted accuracy for each of the feature

selectors was tuned into the experimental vision-based food inspection system. The model consists of a set of rules to be evaluated in order to decide the rejection or the acceptance of a bakery product sample under inspection. Three different bakery products were selected to conduct the experiments: tortillas, seeded buns, and hot dog buns. While the vision-based food inspection system already has preset classification models for tortillas and seeded buns, no preset model existed for the hot dog buns, and therefore initial inspection of the samples had to be done manually. The results of the experiments globally showed that the accuracy of the inspection system auto-tuned with an automatically selected model is generally comparable to the accuracy predicted by cross-validation or holdout. For the hot dog buns, the auto-tuned model actually achieved an accuracy performance significantly higher than the one predicted by test sample estimation (holdout). These additional experiments consequently validate the process of auto-tuning a vision-based food inspection that we have presented by clearly showing that the performance achieved with the experimental auto-tuned system is comparable or even better than the predictions achieved by cross-validation or holdout.

When cross-validation or holdout results do not show a clear superiority of one combination of a learning scheme and a feature selection technique (or no feature selection), the decision of which feature selector to use and which learning scheme to use should be made based on the performance of the system tuned with a specific model for each combination of feature selector and a learning scheme. Therefore cross-validation or holdout should be used only to distinguish between combinations of machine learning schemes and feature selection techniques that have very distinguishable level of performance. Afterwards, auto-tuning the best candidate models and evaluating their performance over few new samples will lead to a more unbiased decision of a model.

While the features retained by means of feature selection techniques may not necessarily correspond to the ones that a human operator may find relevant, most of the auto-tuned classification models presented in this chapter achieved a performance accuracy very close or sometimes even higher when formal feature selection is accomplished compared to no feature selection at all. The proposed process thus

eliminates the current subjective, repetitive trial-and-error based and time consuming process, which requires an in-depth knowledge of the product to be inspected, by being totally objective, non-repetitive, fast, and not requiring any knowledge of the products. These observations hence compel to replacing the current classification software built in the vision-based food inspection system with the classification model that meets the most the business objectives of its operator. It is however necessary to emphasize that having training data that covers as many cases as possible is a definite key in helping the algorithms generalize. Therefore particular caution in choosing the data samples shall be applied to prevent poor generalization of the learner.

Chapter 7. Conclusion and Future Work

This research work focused on the development of a formal and robust process for auto-tuning a vision-based food inspection system for industrial quality control. The first subsection of this final chapter will summarize the main topics discussed in this thesis while the second subsection highlights its major contributions. The last subsection opens the door to possible future enhancements.

7.1 Summary

This thesis was inaugurated by introducing the reader to the problem of configuring a vision-based food inspection system to achieve proper inspection of products in an industrial environment. Chapter 1 advanced that in many cases, vision-based food inspection systems have a built-in generic classification model. Moreover, its configuration to automatically discriminate products into preset categories remains cumbersome and time consuming, and is mainly based on trials and errors. In addition, it is not always clear on which features to focus the system's attention, and therefore subjective feature selection is accomplished according to the system operator's knowledge and experience. Chapter 1 proposed to evaluate the use of machine learning techniques as potential classifiers for vision-based inspection systems. If machine learning turns out to be achieving reasonable results, then its usage will eliminate the iterative trial-and- error configuration process of the system as machine learning schemes will only require training samples in order to automatically "learn" and propose a classification model. Chapter 1 also studied the impact of feature selection techniques on machine learning algorithms with the aim of reducing the time taken to inspect a product, therefore allowing a higher rate of production. Finally, the research was directed towards the development of mechanisms that will allow an existing vision-based food inspection system to automatically adapt its classification model to the type of products being inspected.

An extensive review of the current utilizations of vision-based inspection systems for quality control in the processed food industry was presented in Chapter 2. The analysis of those systems showed machine learning techniques to be more and more solicited. For many of the presented systems, it is however recognized that proper selection of the features to analyze is a key aspect to reliable classification. Chapter 2 therefore presented fundamentally different categories of machine learning strategies such as Bayesian learning, decision tree learning, and multi-layer perceptron learning. Several feature selections techniques were also presented in Chapter 2, and were grouped into two main philosophical approaches: the filter approach where the feature selector is independent of any learning algorithm and serves as a filter to sieve the irrelevant and/or redundant attributes, and the wrapper approach where the feature selectors are rather integrated with a learning algorithm to actively determine the relevant attributes for that particular learning algorithm.

Chapter 3 further developed the rationale and implementation level details behind each of the three evaluated machine learning techniques: Naïve Bayes (a probabilistic learner), C4.5 (a decision tree learner) and Multi-Layer Perceptron (MLP, a neural networks learner). The four evaluated feature selectors were also presented in depth: correlation-based feature selection, consistency-based feature selection, RELIEF feature selection and the feature wrapper. The integration of open source code with custom built and commercial software was also discussed.

Chapter 4 introduced the vision-based food inspection system which served as the experimental platform, and the three different categories of bakery products with which the experiments have been conducted. That same chapter also explains how a data logging module has been built and incorporated to the existing system's software. Finally, experimental results on feature-space reduction showed that feature wrappers generally retain less features than filter-based features selectors, but obviously take more time to execute than the latter due to their repetitive interaction with a target learning algorithm. Apart from the RELIEF algorithm, all feature selectors reduced the feature space dimensionality by more than 80%. However, Chapter 4 finally invites to interpret

with caution the number of features found relevant by means of feature selection as ultimately, the classification performance of the classification model built from those features should also be taken into account.

In Chapter 5, the performance of each of the three machine learning algorithms, with and without feature selection, was investigated in the aim of finding the most suited classification model for each of the three experimented bakery products. It was shown that Naïve Bayes is the fastest learner of all tree learning algorithms, followed by C4.5 decision tree and tailed by MLP. But when it comes to applying what the classifier has "learned", C4.5 is the fastest to produce a decision, followed by Naïve Bayes and MLP respectively. Experimental results showed C4.5 to be the best performing machine learning scheme in terms of accuracy for both seeded buns and tortillas, independently of the feature selector used. For the ciabatta buns, there was no clear machine learning taking the edge in terms of accuracy, but results unanimously showed that all the three machine learning schemes reach their best performance when operating only on the features selected by the wrapper technique. Most of the feature-reduced datasets provided by the attribute selectors gave a predicted classification accuracy very close to the accuracy achieved with the full datasets, and even higher when extracted with either the consistency-based feature selection technique or the feature wrappers with a target learning algorithm. This evaluation with realistic datasets extracted from bakery products demonstrates the relevance of integrating machine learning techniques and feature selectors into the vision-based food inspection system. With this solution, the system can evaluate several candidate machine learning algorithms for a given product and propose a model which readily focuses on fewer features that get automatically identified according to the characteristics of the product being inspected. Moreover, it still provides inspection decisions and classification of a comparable reliability as with the full set of measured features while allowing for a higher rate of production with shorter configuration time and less effort.

Chapter 6 took the evaluation further by going back to tuning the vision-based food inspection system with the specific rules of the C4.5 decision tree achieving the most

predicted accuracy for all the different feature selectors. C4.5 decision tree learning was chosen above Naïve Bayes and MLP because of its high performance exhibited in Chapter 5. The most accurate C4.5 decision trees generated from the full set of features and the four different feature selection techniques were tuned into the vision-based food inspection system for three different bakery products and the real performance was compared to the predicted one. Two out of the three bakery products are among the ones studied in Chapter 4 and Chapter 5 while the third product is an additional use case. The experiments show that the performance of the auto-tuned inspection system on new test samples generally follows the same trend as the performance predicted with cross-validation or repeated holdout, with only slight differences. The performance of the auto-tuned system was even significantly higher than the predictions in the case of the hot dog buns. The fact that performance is evaluated over the system tuned with the model achieving the highest predicted accuracy and the relatively large standard deviation of the prediction could account for this significant difference between prediction and reality. Those additional experiments however flagged out that when accuracy predictions average over cross-validation or repeated holdout are very close for different combinations of learning scheme and feature selector, it might be wiser to study the performance of the system tuned with the specific rules of the best classifier of each learning scheme with and without feature selection before finally deciding on the classification model to use.

7.2 Contributions

This thesis proposes a process for auto-tuning a vision-based food inspection system for industrial quality inspection by means of machine learning techniques in conjunction with feature selection algorithms. More specifically, this research work contributes to:

a) Feature space dimensionality reduction

Chapter 4 evaluated the effectiveness of formal feature selection applied in the automated vision-based inspection of products for the processed food industry where a large number of visual attributes is generally analyzed. Extensive experimental evaluation was conducted on several types of bakery products in order to assess the achievable level of compression of the feature space dimensionality. Those experiments using several feature selection techniques showed that for all types of products reported here and in most cases, more than 80% of the features extracted for analysis are potentially redundant or not necessary to the proper inspection of the processed products. Chapter 5 covered additional criteria going beyond the number of features retained as potentially relevant in order to better judge the usefulness of formal feature selection techniques.

b) Implementation of a data logging module

This work also implemented and embedded a software data logging module into an existing vision-based food inspection system. Chapter 4 explained in detail how the data logging module collects at a very low level the values of all the features computed by the available system and for all the inspected products in a seamless manner which does not disturb the usual framework's real-time operations. This same module allows the storage of the data extracted from the products under inspection for later analysis. The data logging module's ease of operation and the formatting of the extracted data into a standard CSV file give this module versatility and flexibility which allow the data to be manipulated in a straightforward manner with a wide variety of software applications. This implementation readily found application in the commercial product of Dipix Technologies Inc. [4] (now Montrose Technologies Inc.) as the new piece of software was included in the solution provided to the company's customers and is currently running in some bakery plants.

c) *Automated classifier determination with machine learning*

Another major contribution of this work is the proposal of a formal evaluation of machine learning algorithms for use with a vision-based food inspection system. The proposed solution distinguishes itself from other vision-based food inspection systems configuration by providing a status of predicted performance of several candidate machine learning techniques that automatically adapt to the product to be inspected, compared to the systems in [7, 8, 9, 10, 11] where a classifier is generally fixed for a product. Not only does this work propose a process for anticipating the performance of machine learning techniques, but it also studies the impact of feature selection on those machine learning schemes. Beyond the feature-space reduction performance evaluated in Chapter 4, Chapter 5 goes further by forecasting if the features considered relevant by the feature dimensionality reduction techniques actually allow acceptable inspection of bakery products once their interaction has been captured by a machine learning technique. Means of predicting accuracy and time-related performances of the candidate inspection models are also proposed.

d) *Development and experimental evaluation of specific procedures for inspection system's auto-tuning on various products*

Most of the current vision-based inspection systems are built on extensive knowledge of the product to be inspected. One of the most important contributions of this research work is the proposal of a formal process to replace the current configuration process which is very tedious and requires certain knowledge of the products to be inspected. Experiments on a variety of bakery products demonstrate that by following the rigorous process proposed in this work, one can eliminate the cumbersome trials and errors, the subjective feature selection, and the classification model calibration which require an extensive knowledge of the products. Experiments on products for which the vision-based food inspection system already possesses a preset classification model show that

comparable accuracy can potentially be achieved with the proposed automated tuning process. New classification model can also be quickly auto-determined and tuned for types of products that have never been inspected. The proposed solution therefore suppresses the level of effort and knowledge required in the case of Peng and Lu [12] where one has to build six experimental models of apple firmness, or in the case of Abdullah *et al.* [16] where the system's operator has to manually cluster the different classes of muffins, or ultimately in the case of Davidson *et al.* [15] who simply had to subjectively sacrifice some features for complexity reasons.

e) Integration with vision-based food inspection system and validation

Chapter 6 showed how the system can go from predicting its performance using different models to actually auto-tuning a specific model and its set of classification rules into a vision-based food-inspection system. Chapter 6 also took a step further and compared the performance of the auto-tuned models on new products to the one predicted using the proposed process. This additional validation step demonstrated that the performance predicted using cross-validation or repeated holdout is usually a close enough approximation of the real performance achieved with the auto-tuned system. Chapter 6 emphasizes that when the average predicted performances of two different types of models present comparable performance, then further comparative evaluation with the actual most promising classifier of each type of potential model tuned into the system should be accomplished with new real product samples. The integration of open source code with commercial and custom built software has been made possible by tuning a classification model into the vision-based food inspection system.

For the problem of initial configuration of a vision-based food inspection system on a new product, it is no longer necessary to be restricted to one classifier for one product and try to guess on which features this classifier should focus on. Neither is it required to

select the presumably relevant attributes of a product and try to configure a classifier to discriminate products based on those attributes. Using the technology developed in this work, it becomes possible to have the system evaluate by itself several combinations of products, machine learning algorithms, and feature selectors. The system then presents a summary of predicted performance to a human operator who can then base his or her decision on objective facts to select the inspection model that best suits his or her inspection objectives. This thesis therefore proposes a set-based methodology which progressively carries out the evaluation of a set of potential solutions, gradually eliminating the less promising ones until a clear solution outstands. The pain of configuring vision-based based inspection system can therefore be alleviated and done in a trivial yet robust manner.

Part of the work reported in this thesis has been published in [74] and [75].

7.3 Future Work

This thesis evaluated and proposed a process for auto-tuning a vision-based food inspection system. Even though experimental evaluation of the proposed process was conclusive, there is still room for improvement. The first possible enhancement could be incorporating more machine learning and more feature selection techniques in the library of available potential candidates. By having a wider variety of learning algorithms and feature selectors, the inspection system increases its chance of finding the optimal model for classification.

The lower performance results achieved in the case of ciabatta buns and hot dog buns because of the limited number of samples available for training compels to the exploration of *online machine learning* strategies that learn one instance at a time. One could for example start the learning process with a small number of samples using a wrapper feature selection targeting a specific learning algorithm. The inspection can then start with a very coarse classification model that will continue learning and refining

while the number of available samples increases. Other machine learning techniques such as *adaboost* (section 2.2.2) which increase weights of misclassified instances during the learning process could also be explored in order to "boost" the performance of machine learning techniques.

Another aspect that can be improved is to actually take into account the computational cost of the different features. In the performance analysis presented in this thesis, we considered the cost of extracting all the features to be the same; which is generally not the case in reality. If for instance a certain weight is attributed to every feature extracted in order to characterize the effort taken by the inspection system to compute that feature, then it might be possible in some cases to give up on a classifier that decides faster if we know that another classifier which operates slower but gives a better performance accuracy can actually compensate this differential time during the extraction of the required features. A subsequent enhancement would be the implementation of some rules that can even decide on the optimal classifier based on the profile of the system's operator, rules that would combine different performance indicators.

References

1. J. Blasco, N. Aleixos, and E. Molto, "Computer Vision Detection of Peel Defects in Citrus by Means of a Region Oriented Segmentation Algorithm", *Journal of Food Engineering*, vol. 81, no. 3, pp. 535-543, 2007.

2. M. R. Chandraratne, D. Kulasiri, and S. Samarasinghe, "Classification of Lamb Carcass Using Machine Vision: Comparison of Statistical and Neural Network Analyses", *Journal of Food Engineering*, vol. 82, no. 1, pp. 26-34, 2007.

3. H. Almuallim and T.G. Dietterich, "Learning with Many Irrelevant Features", *Proc. of the 9th National Conf. on Artificial Intelligence,* pp. 547-552, AAAI Press, 1991.

4. Dipix Technologies Inc., now Montrose Technologies Inc. http://www.montrose-tech.com/

5. E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision,* Upper Saddle River, NJ 07458: Prentice Hall, 1998.

6. M.R. Chandraratne, D. Kulasiri, and S. Samarasinghe, "Classification of lamb carcass using machine vision: Comparison of statistical and neural network analyses", *Journal of Food Engineering*, vol. 82, no. 1 pp. 26-34, 2007.

7. O. Basset, B. Buquet, S. Abouelkaram, P. Delachartre, and J. Culiolib, "Application of texture image analysis for the classification of bovine meat", *Food Chemistry,* vol. 69, no. 4, pp. 437-445, 2000.

8. B. Park, Y. R. Chen, M. Nguyen, and H. Hwang, "Characterizing multispectral images of tumorous, bruised, skin-torn, and wholesome poultry carcasses", *Transactions of the ASAE*, vol. 39, no. 5, pp. 1933-1941, 1996.

9. B. Park and Y. R. Chen, "Co-occurrence Matrix Texture Features of Multi-spectral Images on Poultry Carcasses", *Journal of Agricultural Engineering Research*, vol. 78, no. 2, pp. 127-139, 2001.

10. F. Storbeck and B. Daan, "Fish species recognition using computer vision and a neural network", *Fisheries Research*, vol. 51, no. 1, pp. 11-15, 2001.

11. F. Storbeck and B. Daan, "Weight estimation of flatfish by means of structured light and image analysis", *Fisheries Research,* vol. 20, no. 2, pp. 99-193, 1991.

12. Y. Peng and R. Lu "Prediction of apple fruit firmness and soluble solids content using characteristics of multispectral scattering images", *Journal of Food Engineering*, vol. 82, no. 2, pp. 142-152, 2007.

13. N. Kondo , U. Ahmad, M. Monta, and H. Murase, "Machine vision based quality evaluation of Iyokan orange fruit using neural networks" *Computers and Electronics in Agriculture*, vol. 29, no. 1 -2, pp.135-147, 2000.

14. M. A. Shahin, E. W. Tollner, R. D. Gitaitis, D. R. Sumner, and B. W. Maw, "Classification of sweet onions based on internal defects using image processing and neural network techniques", *Transactions of the American Society of Agricultural Engineers (ASAE)*, vol. 45, no. 5, pp. 1613-1618, 2002.

15. V. J. Davidson, J. Ryks, and T. Chu, "Fuzzy models to predict consumer ratings for biscuits based on digital image features", *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 1, pp. 62-67, 2001.

16. M. Z. Abdullah, S. A. Aziz, and A. M. Mohamed, "Quality Inspection of Bakery Products Using a Color-based Machine Vision System", *Journal of Food Quality*, vol. 23, no. 1, pp. 39-50, 2000.

17. K. Kira and L.A. Rendell, "The feature selection problem: Traditional methods and a new algorithm", *AAAI-92, Proceedings of the Ninth National Conference on Artificial Intelligence,* pp.129-134, AAAI Press/The MIT Press, 1992.

18. H. Almuallim and T.G. Dietterich, "Learning Boolean concepts in the presence of many irrelevant features", *Artificial Intelligence*, vol. 69, no. 1-2, pp. 269-305, September 1994.

19. G.H. John, R. Kohavi, and K. Pfleger, "Irrelevant feature and the subset selection problem", *Machine Learning: Proceedings of the Eleventh International Conference,* pp. 121-129, Morgan Kaufmann Publishers, 1994.

20. R. Kohavi and G.H. John, "Wrappers for Feature Subset Selection", *Artificial Intelligence,* vol. 97, pp. 273-324, December 1997.

21. P. M. Narendra and K. Fukunaga, "A Branch and Bound Algorithm for Feature Subset Selection", *IEEE Transactions on Computers*, vol. C-26, no. 9, pp. 917-922, September 1977.

22. P. Langley, "Selection of relevant features in machine learning", *Proceedings of the AAAI Fall Symposium on Relevance*, New Orleans: AAAI Press, pp.140-144, 1994.

23. C.E. Queiros and E.S. Gelsema, "On feature selection", *Proceedings Seventh International Conference on Pattern Recognition*, Montreal, Quebec, pp. 128-130, 1984.

24. J. R. Quinlan, "Induction of decision trees", *Machine Learning*, vol. 1, no. 1, pp. 81-106, 1986.

25. G. Pagallo and D. Haussler, "Boolean Feature Discovery in Empirical Learning", *Machine Learning*, vol. 5, no. 1, pp. 71-99, 1990.

26. H. Liu and R. Setiono, "Chi2: Feature Selection and Discretization of Numeric Attributes", *Proceedings of the 7th IEEE International Conference on Tools with Artificial Intelligence*, pp. 388-391, 1995.

27. M. Modrzejewski, "Feature selection using rough sets theory", *Proceedings of the European Conference on Machine Learning*, vol. 667, pp.213-226, 1993.

28. Z. Pawlak, *Rough Sets: Theoretical Aspects of Reasoning about Data*, Kluwer Academic Publishers, 1991.

29. J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm", *IEEE Intelligent Systems*, vol. 13, no. 2, pp.44-49, 1998.

30. M. A. Hall, *Correlation-Based Feature Selection for Machine Learning*, PhD thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 1998.

31. M. A. Hall, "Correlation-Based Feature Selection for Discrete and Numeric Class Machine Learning", *Proc. of the 17th Intl Conf. on Machine Learning,* pp. 359-366, 2000.

32. M.A. Hall and G. Holmes, "Benchmarking Attribute Selection Techniques for Discrete Class Data Mining", *IEEE Transactions on Knowledge and Data Engineering,* vol. 15, no. 6, pp. 1437-1447, 2003.

33. H. Liu and R. Setiono, "A Probabilistic Approach to Feature Selection: a Filter Solution", *Proc. of the 13th Intl Conf. on Machine Learning,* pp. 319-327, 1996.

34. K. Kira and L. Rendell, "A Practical Approach to Feature Selection", *Proc. of the 9th Intl Conf. on Machine Learning,* pp. 249-256, Morgan Kaufmann Publishers, 1992.

35. K. Kira and L. Rendell, "The Feature Selection Problem: traditional methods and a new algorithm", *Proceedings of the Tenth National Conference on Artificial Intelligence* (AAAI-92), San Jose, California, pp.129-134, 1992.

36. I. Kononenko, "Estimating Attributes: Analysis and Extensions of RELIEF", *Proc. of the 7th European Conference on Machine Learning*, pp. 171-182, Berlin; New York: Springer-Verlag Publisher, 1994.

37. S. Maldonado and R. Weber, "A wrapper method for feature selection using Support Vector Machines", *Information Sciences*, vol. 179, no. 13, pp. 2208-2217, June 2009.

38. Y. Li, J.-L. Wang, Z.-H. Tian, T.-B. Lu, and C. Young, "Building lightweight intrusion detection system using wrapper-based feature selection mechanisms", *Computers & Security*, vol. 28, no. 6, pp. 466-475, September 2009.

39. J.-H. Hong and S.-B. Cho, "Gene boosting for cancer classification based on gene expression profiles", *Pattern Recognition*, vol. 42, no. 9, pp. 1761-1767, September 2009.

40. C. J. Huang, D. X. Yang, and Y. T. Chuang, "Application of wrapper approach and composite classifier to the stock trend prediction", *Expert Systems With Applications*, vol. 34, no. 4, pp. 2870-2878, May 2008.

41. S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

42. V. Cerný, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm", *Journal of Optimization Theory and Applications*, vol. 45, no. 1, pp. 41-51, 1985.

43. D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A Learning Algorithm for Boltzmann Machines", *Cognitive Science*, vol. 9, no. 1, pp. 147-169, 1985.

44. G. E. Hinton and T. J. Sejnowski, "Learning and Relearning in Boltzmann Machines", *Parallel distributed processing: explorations in the microstructure of cognition \ D. E. Rumelhart, J. L. McClelland, and the PDP Research Group*, vol. 1: Foundations, pp 282-317. Cambridge: MIT Press, 1986.

45. R. R. Bies, M. F. Muldoon, B. G. Pollock, S. Manuck, G. Smith, and M. E. Sale , "A Genetic Algorithm-Based, Hybrid Machine Learning Approach to Model Selection", *Journal of Pharmacokinetics and Pharmacodynamics*, vol. 33, no. 2, pp. 93-221, 2006.

46. T. Mitchell, *Machine learning*, McGraw Hill, 1997.

47. R. S. Michalski, and R. E. Stepp, "Learning from observation: Conceptual clustering", *Machine learning: an artificial intelligence approach, editors: Ryszard S. Michalski, Jaime G. Carbonell, Tom M. Mitchell*, Los Altos, Calif.: M. Kaufmann Publishers, 1983.

48. S. Haykins, *Neural networks: a comprehensive foundation*, 2nd Edition, Prentice Hall 1999.

49. Wikipedia, Reinforcement Learning, http://en.wikipedia.org/wiki/Reinforcement_learning, visited on July 18th, 2009.

50. G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers", *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann Publishers, San Mateo, 1995.

51. P. Clark and T. Niblett, "The CN2 Induction Algorithm", *Machine Learning*, vol. 3, no. 4, pp. 261-283, 1989.

52. J. R. Anderson and M. Matessa, "Explorations of an Incremental, Bayesian Algorithm for Categorization", *Machine Learning*, vol. 9, pp. 275-308, 1992.

53. P. Domingos and M. Pazzani, "Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier", *Proc. of the 13th Intl Conf. on Machine Learning*, Bari, Italy, Morgan Kaufmann, pp. 105-112, 1996.

54. J. R. Quinlan, *C4.5: Programs for Machine Learning*, San Mateo, California: Morgan Kaufmann Publishers, 1993.

55. L. Breiman, "Random Forests", *Machine Learning,* vol. 45, no. 1, pp. 5-32, 2001.

56. T. Hill and P. Lewicki, *STATISTICS Methods and Applications*, StatSoft, Tulsa, OK, 2007.

57. T. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction à l'algorithmique*, Paris : Dunod, 2002. (Translated from English by X. Cazin and G.-L. Kocher, Original English title: *Introduction to algorithms*).

58. N. A. Gershenfeld, *The nature of mathematical modeling*, Cambridge, New York: Cambridge University Press, 1999.

59. I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

60. Weka: Data Mining Software in Java, http://www.cs.waikato.ac.nz/ml/weka/

61. M. R.-Sikonja and I. Kononenko, "An adaptation of Relief for attribute estimation in regression", *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 296-304, 1997.

62. J. H. Gennari, P. Langley, and D. Fisher, "Models of Incremental Concept Formation", *Artificial Intelligence,* vol. 40, no. 4, pp. 11-61, 1999.

63. P. Langley and S. Sage, "Induction of Selective Bayesian Classifiers", *Proc. of the 10th Conf. on Uncertainty in Artificial Intelligence,* Seattle, W.A, Morgan Kaufmann Publishers, pp. 399-406, 1994.

64. R. Kohavi and D. Sommerfield, "Feature Subset Selection Using the Wrapper Method: Overfitting and Dynamic Search Space Topology", *Proc. of the 1st Intl Conf. on Knowledge Discovery and Data Mining,* AAAI Press, pp. 192-197, 1995.

65. E. E. Ghiselli, *Theory of psychological Measurement, McGraw-Hill,* New York, 1964.

66. R. B. Zajonic, "A note on group judgements and group size", *Human Relations,* vol. 15, no. 2, pp.177-180, May 1962.

67. J. R. Quinlan, "Inferring decision trees using the minimum description length principal", *Information and Computation,* vol. 80, no. 3, pp. 227-248, March 1989.

68. J. Rissanen, "Modeling by shortest data description", *Automatica,* vol. 14, no. 6, pp. 465-471, 1978.

69. W. H. Press, B.P. Flannery, S.A. Teukolski, and W.T. Vetterling, *Numerical recipes in C++: The art of Scientific Computing*, New York; Cambridge: Cambridge University Press, 2002.

70. I. Kononenko, "On biases in estimating multi-valued attributes", *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI95),* vol. 2, pp.1034-1040, Montreal, Quebec, Canada August 20-25, 1995.

71. R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection", *Proc. of the Intl Joint Conf. on Artificial Intelligence*, Montreal, QC, Morgan Kaufmann, pp. 1137-1143, 1995.

72. MATLAB ®, "The MathWorks", *www.mathworks.com.*

73. U. M. Fayyad and K.B. Irani, "Multi—Interval Discretization of Continuous-Valued Attributes for Classification Learning", *Proc. of the 13th Intl Joint Conf. on Artificial Intelligence*, Morgan Kaufmann, pp. 1022-1027, 1993.

74. M. M. Chetima and P. Payeur, "Feature Selection for a Real-Time Vision-based Food Inspection System", *IEEE International Workshop on Robotic and Sensors Environments*, pp. 120-125, 2008.

75. M. M. Chetima and P. Payeur, "Feature Space Dimensionality Reduction for Real-Time Vision-Based Food Inspection", *Sensors & Transducers Journal*, vol. 5, Special Issue, pp.86-103, March 2009.

Appendix A. Selected Features for Tortillas Dataset

The nature of the attribute parameters has been voluntarily omitted for confidentiality reasons.

Attributes	RELIEF	CFS	Consistency	NB wrapper	C4.5 wrapper	MLP wrapper
M0						
M1						
M2						
M3						
M4						
M5						
M6						
M7						
M8						
M9						
M10						
M11						
M12						
M13						
M14						
M15						
M16						
M17	**Yes**		**Yes**			
M18	**Yes**		**Yes**			
M19	**Yes**			**Yes**		**Yes**
M20	**Yes**					
M21	**Yes**					**Yes**
M22	**Yes**			**Yes**		
M23	**Yes**	**Yes**	**Yes**		**Yes**	

M24	Yes	Yes	Yes	Yes	Yes	Yes
M25	Yes	Yes	Yes	Yes	Yes	Yes
M26	Yes		Yes			
M27	Yes		Yes			
M28						
M29	Yes					
M30	Yes					
M31	Yes			Yes		
M32						
M33	Yes					
M34	Yes					
M35						
M36	Yes					
M37						
M38						
M39						
M40						
M41						
M42	Yes					
M43	Yes					
M44	Yes					
M45	Yes					
M46						
M47	Yes					
M48	Yes	Yes				
M49						
M50	Yes					
M51						
M52	Yes					
M53						
M54						

M55						
M56						
M57						
M58						
M59	**Yes**					
M60						
M61						
M62	**Yes**					
M63	**Yes**					
M64	**Yes**					
M65						
M66			**Yes**			
M67						
M68						
M69						
M70						
M71						
M72						
M73						
M74	**Yes**			**Yes**		
M75						
M76						
M77	**Yes**		**Yes**			
M78	**Yes**					
M79	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
M80						
M81						

Appendix B. Selected features for Seeded Buns Dataset

The nature of the attribute parameters has been voluntarily omitted for confidentiality reasons.

Attributes	RELIEF	CFS	Consistency	NB wrapper	C4.5 wrapper	MLP wrapper
M0	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
M1	**Yes**					
M2	**Yes**					
M3	**Yes**	**Yes**		**Yes**		
M4	**Yes**					
M5						**Yes**
M6						
M7	**Yes**					
M8	**Yes**			**Yes**		
M9	**Yes**					
M10	**Yes**					
M11						
M12	**Yes**	**Yes**				
M13	**Yes**		**Yes**		**Yes**	**Yes**
M14	**Yes**					
M15	**Yes**			**Yes**		
M16	**Yes**			**Yes**		
M17	**Yes**		**Yes**			
M18	**Yes**					
M19	**Yes**					
M20	**Yes**					
M21	**Yes**					
M22	**Yes**					
M23	**Yes**					

M24	Yes					
M25	Yes					
M26	Yes	Yes				
M27	Yes					
M28	Yes	Yes				
M29	Yes					
M30	Yes					
M31	Yes					
M32						
M33	Yes					
M34	Yes					
M35						
M36	Yes					
M37	Yes	Yes				
M38						
M39						
M40	Yes					
M41						
M42	Yes					
M43	Yes		Yes			
M44	Yes					
M45	Yes					
M46	Yes					
M47	Yes					
M48	Yes					
M49						
M50	Yes					
M51	Yes					
M52						
M53	Yes					
M54	Yes					Yes

M55						Yes
M56						
M57	Yes					
M58	Yes					
M59	Yes					
M60	Yes					
M61	Yes					
M62	Yes			Yes		
M63	Yes					
M64						
M65	Yes	Yes	Yes			Yes
M66						
M67						
M68						
M69						
M70						
M71						
M72						
M73						
M74	Yes					
M75						
M76	Yes					
M77	Yes					
M78	Yes					
M79						
M80	Yes					
M81	Yes					

Appendix C. Selected Features for Ciabatta Buns Dataset

The nature of the attribute parameters has been voluntarily omitted for confidentiality reasons.

Attributes	RELIEF	CFS	Consistency	NB wrapper	C4.5 wrapper	MLP wrapper
M0	Yes					
M1	Yes					
M2	Yes	Yes	Yes			
M3	Yes			Yes		Yes
M4	Yes	Yes				Yes
M5			Yes			
M6		Yes			Yes	
M7						
M8					Yes	
M9						
M10		Yes				
M11						
M12						
M13	Yes	Yes				
M14	Yes	Yes				
M15	Yes		Yes			
M16	Yes					
M17	Yes		Yes		Yes	
M18	Yes					Yes
M19	Yes	Yes				
M20	Yes					
M21	Yes	Yes				
M22	Yes					
M23	Yes					

M24	Yes					
M25	Yes	Yes	Yes	Yes		
M26	Yes	Yes				
M27	Yes					Yes
M28	Yes					
M29	Yes					
M30						Yes
M31						
M32	Yes					
M33						
M34						
M35						
M36						
M37						
M38	Yes					
M39						
M40						
M41						
M42	Yes					
M43	Yes	Yes				
M44	Yes			Yes		Yes
M45	Yes					
M46						
M47						
M48						
M49	Yes					
M50						
M51						
M52	Yes					
M53						
M54						

M55						
M56						
M57	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
M58	**Yes**					
M59	**Yes**		**Yes**			
M60						
M61						
M62						
M63						
M64						
M65						
M66						
M67						
M68						
M69						
M70						
M71						
M72						
M73						
M74	**Yes**	**Yes**	**Yes**			**Yes**
M75						
M76						
M77	**Yes**					**Yes**
M78	**Yes**	**Yes**	**Yes**			
M79						
M80						
M81						

Appendix D. Holdout Training Time and Testing Time for Tortillas Dataset

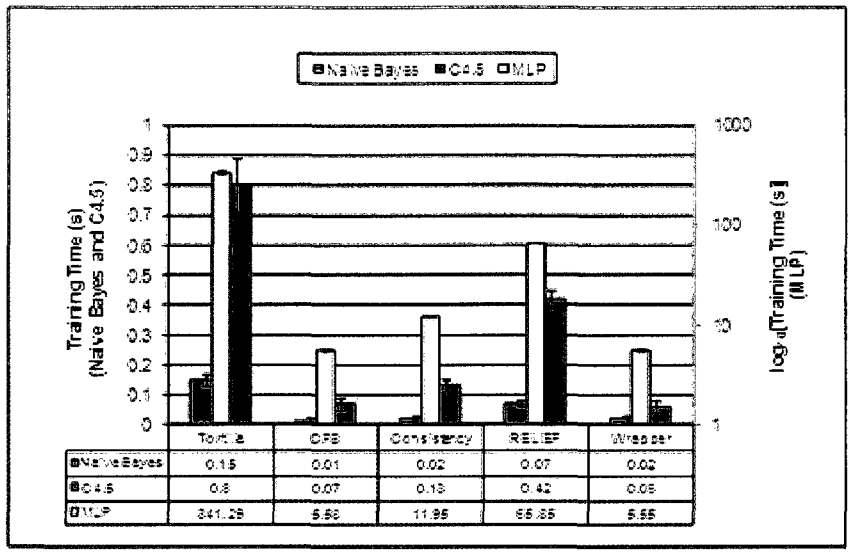


Figure D.1. Average training time over 100 repetitions of the holdout accuracy estimation for the tortillas dataset

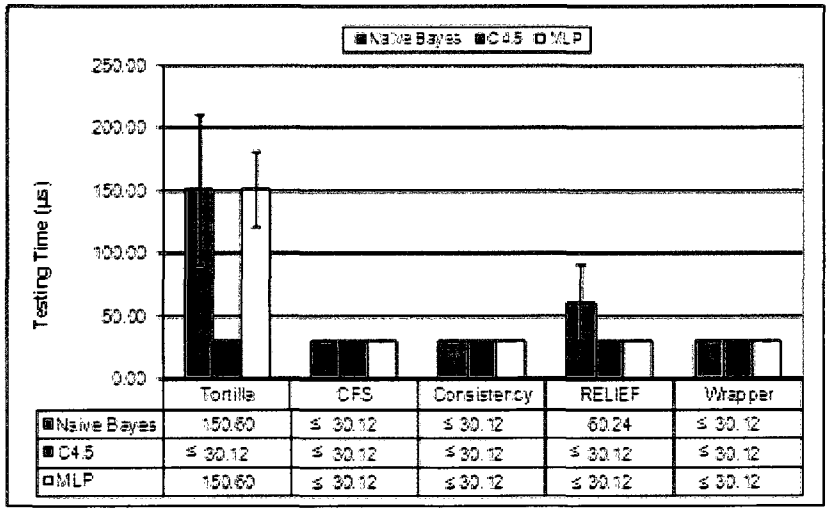


Figure D.2. Average testing time over 100 repetitions of the holdout accuracy estimation for the tortillas dataset

Appendix E. Holdout Accuracy Estimation Results for the Seeded Buns Dataset

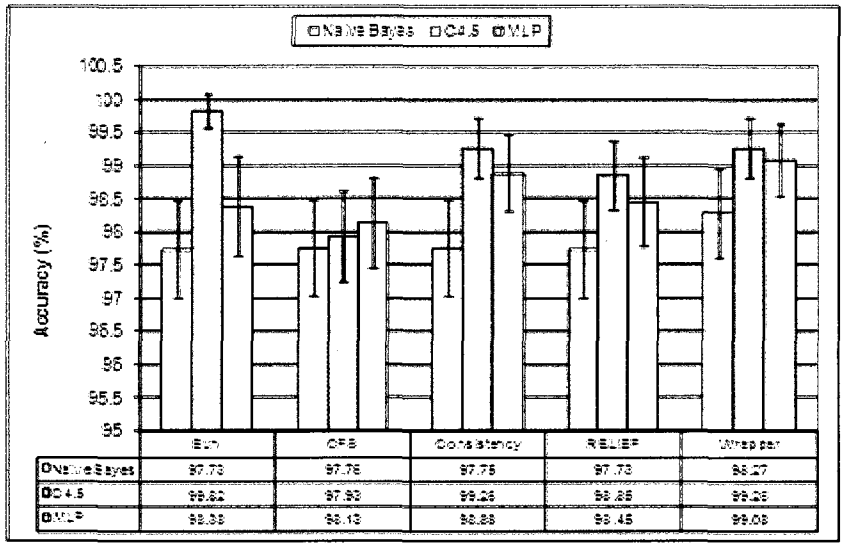


Figure E.1. Average over 100 repetitions of the holdout accuracy estimation for the seeded buns dataset

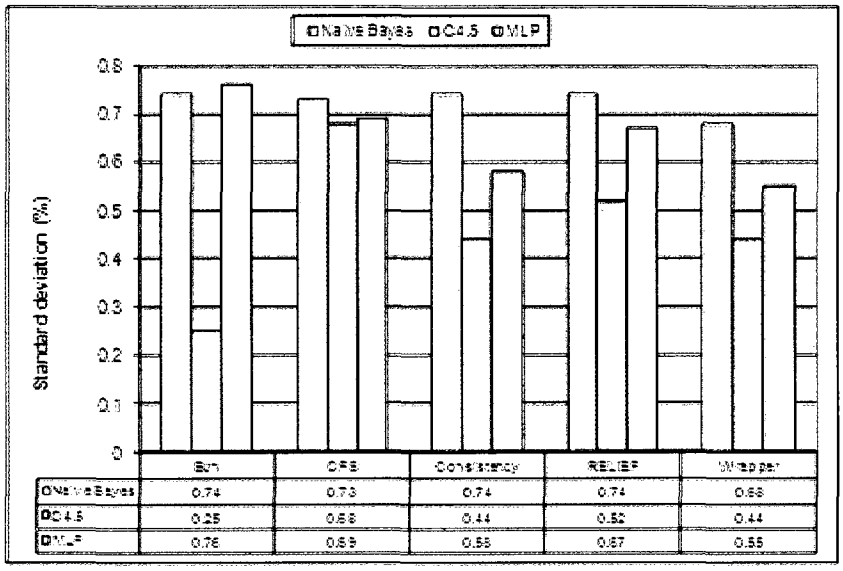


Figure E.2. Standard deviation of 100 repetitions of the holdout accuracy estimation for the buns dataset

Appendix F. Holdout Training Time and Testing Time for the Seeded Buns Dataset

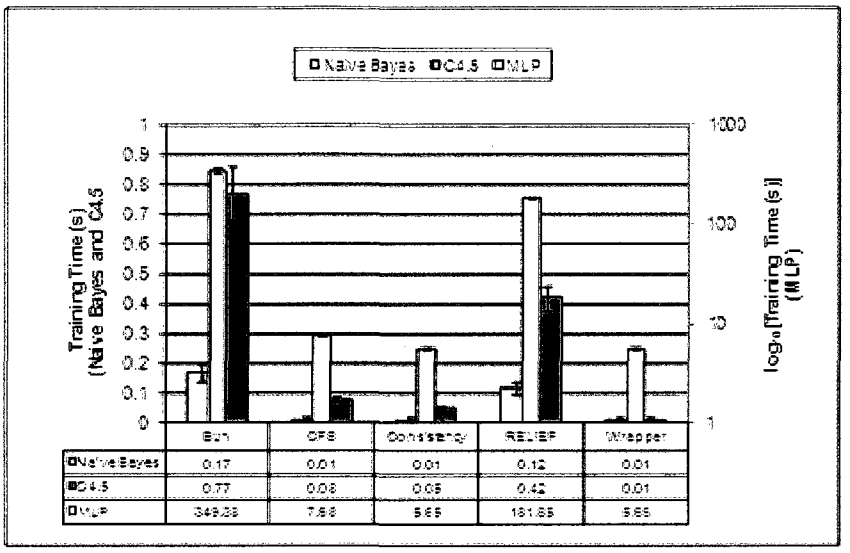


Figure F.1. Average training time over 100 repetitions of the holdout accuracy estimation for the seeded buns dataset

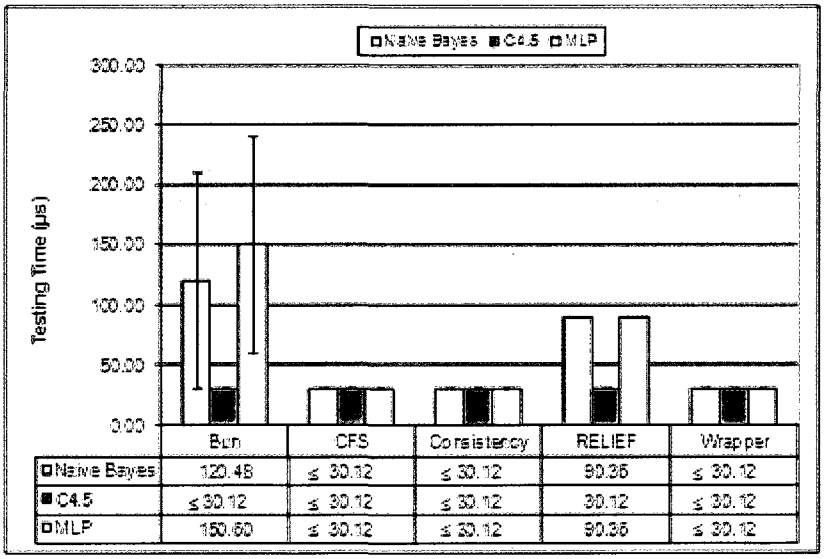


Figure F.2. Average testing time over 100 repetitions of the holdout accuracy estimation for the seeded buns dataset

Appendix G. Cross-Validation Training Time and Testing Time for Ciabatta Buns Dataset

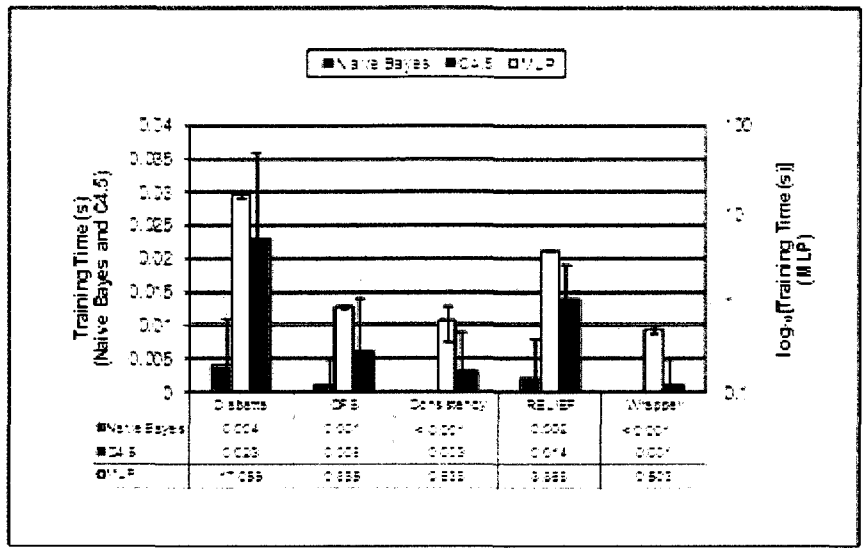


Figure G.1. Average training time over 10 repetitions of the 10-fold cross-validation accuracy estimation for the ciabatta buns dataset

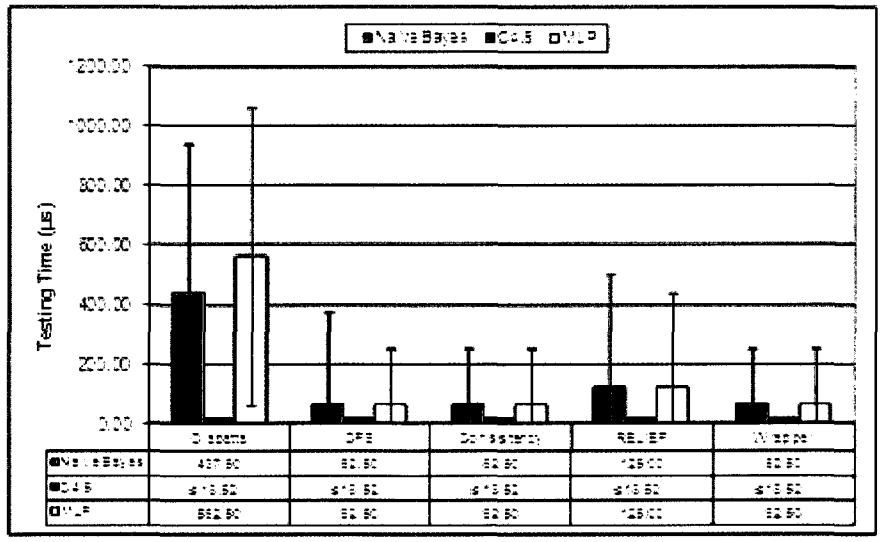


Figure G.2. Average testing time over 10 repetitions of the 10-fold cross-validation accuracy estimation for the ciabatta buns dataset

Appendix H. Selected Features for Hot Dog Buns Dataset

The nature of the attribute parameters has been voluntarily omitted for confidentiality reasons.

Attributes	RELIEF	CFS	Consistency	NB wrapper	C4.5 wrapper	MLP wrapper
M0	Yes		Yes			
M1	Yes		Yes			
M2	Yes		Yes			
M3	Yes		Yes			
M4	Yes	Yes	Yes			Yes
M5	Yes		Yes			
M6	Yes	Yes				
M7	Yes					
M8	Yes					Yes
M9						
M10	Yes		Yes			
M11						
M12	Yes	Yes	Yes			
M13	Yes					
M14	Yes	Yes				
M15	Yes	Yes				
M16	Yes	Yes	Yes	Yes	Yes	Yes
M17	Yes					
M18	Yes					
M19	Yes					
M20	Yes	Yes	Yes			Yes
M21	Yes					Yes
M22	Yes					
M23	Yes	Yes	Yes			Yes

M24	Yes	Yes				Yes
M25	Yes	Yes	Yes			
M26	Yes					
M27	Yes					
M28	Yes	Yes	Yes			
M29	Yes					
M30						
M31						
M32	Yes				Yes	Yes
M33						
M34						
M35	Yes					
M36						
M37						
M38	Yes					
M39						
M40						
M41						
M42	Yes					
M43	Yes					
M44	Yes					
M45	Yes					
M46						
M47						
M48						
M49	Yes					
M50						
M51						
M52	Yes					
M53						
M54						

M55						
M56						
M57	**Yes**					
M58	**Yes**	**Yes**				
M59	**Yes**					
M60						
M61						
M62	**Yes**			**Yes**		
M63	**Yes**					
M64	**Yes**					
M65	**Yes**					
M66	**Yes**				**Yes**	
M67	**Yes**					
M68						
M69						
M70						
M71						
M72						
M73						
M74	**Yes**					
M75						
M76						
M77	**Yes**					**Yes**
M78	**Yes**					
M79	**Yes**		**Yes**			**Yes**
M80						
M81						