The School of Mathematics

THE UNIVERSITY *of* EDINBURGH

# Thorough Empirical Comparison of Statistical Learners in Binary and Multi-class Classification Settings

by

**Mohamed Donia**

Dissertation Presented for the Degree of
MSc in Computational Applied Mathematics

August 2019

Supervised by
Dr Thanasis Tsanas

## Abstract

Classification is one of the main prediction tasks demanded by a myriad of business, social and life sciences applications. Not only it saves the experts time, it can identify patterns in the data that are not possible for experts to identify. The field of statistical learning has provided various techniques to tackle the classification problem. Despite the ongoing successful research in the field, it is not practical to try all possible methods to get the best model performance. Hence, a thorough empirical comparative studies are needed to provide guidance in applications.

In this thesis, we focus on seven of the most common statistical learners used in classification problems, naive Bayes, logistic regression, $K$-nearest neighbors, random forests, adaptive boosting, linear support vector machines and radial basis function support vector machines. The aim of this study is to compare the performance of the seven classifiers when applied to real-world data sets. The data is categorized to two settings; binary-class data and multi-class data.

Based on the accuracy measure, radial basis function support vector machines has the best performance in binary classification settings whereas random forests outperformed the other classifiers in multi-class settings. Model robustness has also been monitored by cross validation and those two methods proved robust particularly in binary-class setting. The code and full analysis of all data sets are on Gihub repository called StatLearn-Comparison

# Acknowledgments

I would like to thank Dr Thanasis Tsanas not only for the thesis support but also for the friendly, warm and secure environment he creates for the team. I would like also to thank Dr Kostas Zygalakis for making it easy through this tough academic year. Last but not least, thank you my mom, thank you my dad, thank you my lovely sisters.

## Own Work Declaration

# Contents

# List of Tables

# List of Figures

# 1 Introduction

Statistical learning methods have always been there for interpreting and predicting a phenomenon based on some historical data. It is only after the technological advancements in the computer hardware and computational power that Statistical Learning has been extensively deployed to applications. It is playing a key role now in almost every single aspect of life; finance, manufacturing, development policy, social media, health care and many other areas. Algorithms have been developed to deal with learning from images, language usage and many other forms of structured and unstructured data. In a typical scenario, we want to predict either a quantitative value, such as the price of a house, or a categorical assignment such as if a specific patient has cancer or which severity level of cancer he has. The former is called a *regression* problem and the latter is *classification.* This dissertation is concerned with the classification problem, and specifically with empirically comparing the state-of-the-art methods when applied to real-world data sets. In this chapter, we provide some background on statistical learners and their use. The motivation, scope and outline are also provided.

## 1.1 Motivation and Scope

A key consideration in many applications employing machine learning techniques is deciding on the best classifier to be used for that particular setting. It is indeed, this is known as the no-free-lunch-theorem [1]. Different classifiers have different performances when applied to different types of data sets. To have a thorough comparison between classifiers, researchers ought to choose the classifiers, the type of data features they are working on and whether it is artificial or real-world data sets. Amancio et al. [2], have made a comparison between nine well-known classifiers to guide non-experts for starting with some classifiers without any fine tuning of hyperparameters. They used only the default parameters of methods in Weka commercial software explained in [3]. They have chosen to work on Naive Bayes (NB), two types of Decision Trees (DT), Random Forests (RF), K-Nearest Neighbors (KNN), Logistic Regression (LR), Support Vector Machines (SVM) and two types of networks. Entezari-Maleki et al. chose DT, KNN, LR, NB, SVM and linear classifier [4]. Both studies generated data sets for their comparison experiments, each in a different way. One study [5] has chosen to apply 33 different algorithms on 14 real data sets and 2 other artificial ones. Ten algorithms were versions of decision trees or rule-based methods and the remaining were versions of discriminant analysis and neural nets. Deciding the *best* shall be based on a performance evaluation metric. Entezari-Maleki et al compared the Area Under Curve (AUC) of Receiver Operating Curve (ROC). Amancio et al. focused on the typical accuracy metric with a standard deviation across different data sets for each classifier to have a quantification of certainty in the average accuracy measure. Detailed explanations and definitions of those metrics is provided in section 3.4. Other special studies compared different classifiers on one specific data set or problem. Carnahan et al. worked on different classifiers for predictive modeling in human factors research [6]. A lot of research has been done to improve the accuracy of classification on the breast cancer diagnostics Wisconsin data set [7], [8], [9], [10] and other cancer data sets [11].

Over the years, the list of the most common and known-to-perform-well classifiers changes and this is the reason why the previous studies choose different methods to compare. The question of the algorithm of best performance still holds and needs to addressed. If no absolute answer to be given for all the cases, engineers and practitioners need, at least, further insights on the circumstances each algorithm will probably perform the best. This study aims to provide some guidance on the choice of classification algorithms by means of empirical comparisons across diverse data sets.

## 1.2 Outline

Chapter 2 explains the theoretical and conceptual foundation of each statistical classifier used in our study. It also explains which specific settings are implemented for each method if the method has parameter variations. The methodology used to assess each classifier in terms of metrics, hyperparameter tuning and validation of results is in Chapter 3. A clear description of the data sets, the features and the number of classes is shown in Chapter 4. Chapter 5 discusses the results and compares the performance of the classifiers. Discussion of key results, rules of thumb, limitations and future work is seen in Chapter 6.

# 2 Statistical Learners

In this chapter, we describe some of the key classification algorithms which are widely used in the research literature, and which will be subsequently used for our empirical comparisons.

In order to make it easy to follow the subsections of this chapter and the following chapters, we will first agree on terminology and conventional mathematical notation that will henceforth be used. The number of entries in the data set will always be denoted $n$, the number of features $p$. The $nxp$ matrix representing the features for all the data will be in bold capital $\mathbf{X}$ and the vector comprising the classes will be lower-case bold $\mathbf{y}$. If we are referring to one element in a matrix or a vector, we will denote it with a sub-scripted non-bold lower-case letter such as $x_{ij}$ and $y_i$. Only when we mention that the features are for one instance, we will drop the first subscript to be $x_j$ instead of $x_{ij}$. The set of all classes is $\mathbb{C}$ and hence $y_i \in \mathbb{C}$. Any other notations will be defined when mentioned.

## 2.1 Naïve Bayes

Many technological advancements and new statistical methods are based on the conditional probability theorem named after its first discoverer, Thomas Bayes. Bayesian inference is one of those methods, but Naïve Bayes is a classification method that is not based on Bayesian inference but based on Bayes' theorem. Bayes' theorem describes the probability of an event given that another event has happened with some previous knowledge of the probability of the first event happening regardless of the conditions. In a more formal way:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2.1}$$

That is the probability of event A given event B happening $P(A|B)$ is calculated by knowing the probability of event B given A happening $P(B|A)$ multiplied by the probability of A $P(A)$ and divided by the probability of event B $P(B)$. Applying Bayes' theorem to our classification problem, we will classify a feature vector $\mathbf{x}$ to a specific class ($y \in \mathbb{C}$), where $\mathbb{C}$ is the set of all classes, based on the probability of the class given for the observation $\mathbf{x} = (x_1, x_2, \ldots, x_p)^T$.

$$P(y|x_1, x_2, \ldots x_p) = \frac{P(x_1, x_2, \ldots x_p|y) \cdot P(y)}{P(x_1, x_2, \ldots x_p)} \tag{2.2}$$

$P(x_1, x_2, \ldots x_p|y)$ is the probability of each feature having the observation we have it given the class $y$. The probability of each feature having the specific number we have in the observation is in fact dependent on the class and the other features. The class that has the maximum probability will be assigned to the observation. If we are predicting the probability of the classes based on one feature that we assume has a specific probability distribution, Bayes' theorem will be applied with no complexity involved. In real-life problems, the class prediction is dependent on many features and those features could have correlations between each other. The correlation and the dependence of each feature on the other features is very hard to know in practice. Hence, the naïve Bayes method assumes independence and that's why it is called naïve. Each feature will be assumed to be only dependent on the class and the formula becomes as follows:

$$P(y|x_1, x_2, \ldots x_p) = \frac{P(y) \cdot \prod_{i=1}^{p} P(x_i|y)}{P(x_1, x_2, \ldots x_p)} \tag{2.3}$$

Since the denominator is constant and does not depend on the class, we can reformulate Eq. (2.3) to be:

$$P(y|x_1, x_2, \ldots x_p) \propto P(y) \cdot \prod_{i=1}^{p} P(x_i|y) \tag{2.4}$$

And classify based on maximizing the probability:

$$\hat{y} = arg\max_{y} P(y) \cdot \prod_{i=1}^{p} P(x_i|y) \tag{2.5}$$

Another assumption is needed for each feature. Each feature should follow a probability distribution with its own parameters. For example, to calculate $P(x_i|y)$, we can assume that this feature follows a Gaussian distribution with its parameters, namely mean and variance. To estimate the mean and variance for this distribution, we use the maximum likelihood estimate for that specific class. In other orders, we calculate the mean and the variance of the data we have for that specific class. This is an estimate not claimed to be the true distribution and if it is the true distribution of the population, the parameters are not claimed to be the true parameters, but they are good enough to move forward. One of the most common assumptions about the feature distribution is to be Gaussian for all continuous variable features. The last assumption is about $P(y)$ which we call the prior probability. It is the subjective probability of how frequent this class is in the population and we also use the maximum likelihood as a good estimate. If we have an expert knowledge that the frequency in the population is different from the frequency in the dataset, we can use the expert's prior instead.
Although one would think it is an over-simplified classifier and it wouldn't perform well, naïve Bayes is reported to be working well in real-world problems specifically in text classification and spam filtering. It is worth to mention that this makes naïve Bayes extremely fast and efficient in some cases when compared to other complicated classifiers. A rigorous proof for the reason why naïve Bayes is a good classifier and the conditions for that is discussed in [12].

## 2.2 Logistic Regression

Traditionally, linear regression is used to model numerical continuous response variables [1]. The classification problem as mentioned before is about assigning a class to an observation. For instance, the output shall be male or female, malignant cancer or benign, a cat or a dog or a human, low or medium or high price. In those examples, the output is not a number but a class. If we could just replace the classes with numbers to represent them such as 0 for males and 1 for females, there is an implication in such representation which is not necessarily true. The implication is there is an order between classes. This is true for the low, medium, high price classification but we need to make sure the distance between the low and medium is the same as the distance between the medium and high if we are going to code them with three consecutive integers. This implication might not be a problem in the binary classification problems but would not make sense for the cat, dog, human classification problem. Logistic regression comes as a solution to those problems in linear regression for classification. If the linear regression is estimating the parameters $\beta$ for a continuous response in the following equation:

$$\hat{y} = \beta^T \mathbf{x} \tag{2.6}$$

The logistic regression model is estimating the probability odds of the class given the observation $\mathbf{x}$. Formally,

$$\frac{P(y|\mathbf{x})}{1 - P(y|\mathbf{x})} = \beta^T \mathbf{x} \tag{2.7}$$

This would give probability values less than 0 and greater than 1 and to avoid that, in logistic regression we use the log of the odds instead of the odds. The equation becomes:

$$\log\Big(\frac{P(y|\mathbf{x})}{1 - P(y|\mathbf{x})}\Big) = \beta^T \mathbf{x} \tag{2.8}$$

With some algebraic manipulation to get the probability value, we find that

$$P(y|\mathbf{x}) = \frac{e^{\beta^T \mathbf{x}}}{1 + e^{\beta^T \mathbf{x}}} \tag{2.9}$$

To estimate $\beta$ based on the training data, we need a function to optimize. This function needs to output probability close to 1 when the class is $y$ and close to 0 when the class is not y. In this case, we typically use the likelihood function to maximize.

$$l(\beta) = \prod_{i:y_i=y} P(x_i) \prod_{i:y_i\neq y} (1 - P(x_i)) \tag{2.10}$$

The above equation is general for binary or multi-class problems as it gets the probability for each class separately considering the other classes as not in that class. In other words, it models the multi-class problem as smaller binary-class problems. This is called one-versus-rest or one-versus-all in the literature.
Since logistic regression involves optimization, the setting of the code needs specifications for the solver, the penalty method. The study of optimization is out of the scope of this work, so the default parameters in python scikit-learn will be used.

## 2.3 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a non-parametric classifier and it learns based on instances. Unlike other classifiers, it does not make any assumptions about the data distribution. It simply stores the training set and searches for the $K$ nearest neighbors to the point we want to assign a class for. It then gives a vote for the most probable class. Variations of the algorithm differ in how they decide which of the points are nearest and how the algorithm votes for the class. The simplest and most common version of the algorithm calculates the Euclidean distance, the $L_2$ norm, and does a majority vote where $K$ is an odd number to avoid ties. Other possible distance measures are $L_1$ norm and $L_\infty$ norm. Typically, all the $K$ neighbors contribute equally to the vote. Other versions consider giving more weight to the closer neighbors. The chosen value of $K$ changes the boundary dramatically and the optimal value differs from data set to another. Scaling, which is a preprocessing step explained in section 3.1, is important for such distance-based methods.
In our implementation, scaling is applied first with Euclidean distance and equal weights for the classification voting. Our implementation also searches first for the best value of $K$ and then uses it for prediction on the test set.

## 2.4 Support Vector Machines

Support Vector Machines (SVM) is another distance-based classifier that proved to perform well over the last few decades on real applications since its development in 1995 [13]. It is a development of the support vector classifier which, in turn, is a development of the maximal marginal classifier. If the points are linearly separable as shown in the example of Figure 2.1, it is easy to just draw a line, if it is a 2-dimensional feature space, between the two classes. The line is extended when the feature space is $p$-dimensional to be a hyperplane of dimension $p-1$. The hyperplane generally is:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0 \tag{2.11}$$

for parameters $\beta_0$,$\beta_1$, $\beta_2$, ... and $\beta_p$. Substituting a point $X = (X_1, X_2, \ldots, X_p)^T$ and supposing that it is not on the hyperplane, the classification will be decided upon the value. If $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0$, the point will be assigned to one class and assigned to the other class if $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p > 0$. To have a unique solution, we have to add an extra reasonable restriction. The natural restriction is to have the hyperplane exactly in the midway between the two classes with the largest distance possible to the nearest point from both classes. In other words, the unique hyperplane is the maximal marginal classifier as shown in Figure 2.2. The dashed lines are the margins and the points on the dashed lines are the support vectors as they support the classifier.

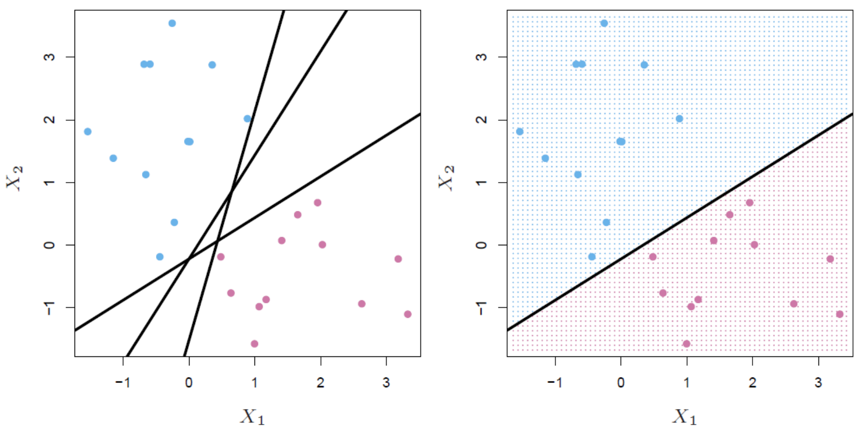


Figure 2.1: [
Linearly separable classes in binary classification setting.]Left: two linearly separable classes with multiple separating hyperplanes. Right: one hyperplane is chosen to show how the new points will be classified based on that decision, taken from [14].

To construct the maximal marginal classifier, we set up an optimization problem for $n$ points $x_1, \ldots, x_n \in \mathbb{R}^p$ with classes $\{-1, 1\}$ for a binary classification problem

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}} \quad M \tag{2.12}$$

$$\text{subject to} \quad \textstyle\sum_{j=1}^{p} \beta_j^2 = 1, \tag{2.13}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geqslant M \qquad \forall i = 1, \ldots, n. \tag{2.14}$$

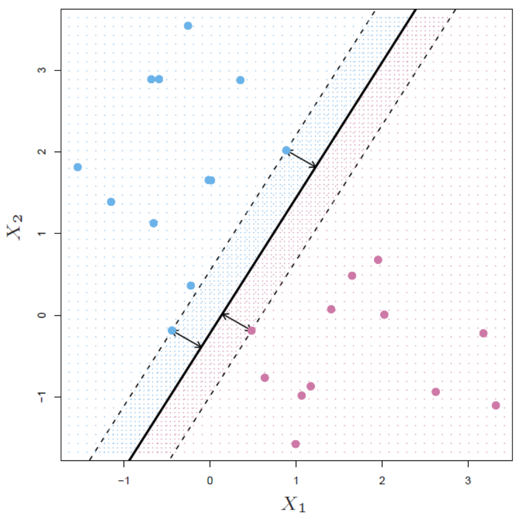


Figure 2.2: Maximal marginal classifier, taken from [14].

The optimization problem in Equations (2.12)-(2.14) is simply searching for the best parameters $\beta_0, \beta_1, \ldots, \beta_p$ that maximize $M$ which is the margin around the hyperplane $y_i(\beta_0 + \beta_1 x_{i1} +$

$\beta_2 x_{i2} + \cdots + \beta_p x_{ip}) = 0$. Equation (2.13) just makes sure that the points are at least M distance from the hyperplane and it is not a constraint on the hyperplane itself.

### 2.4.1 Support Vector Classifier

In real-life problems, the classes are almost always not linearly separable in the original feature space and no solution to the optimization problem in (2.12)-(2.14). Consequently, we need to allow for some misclassification and formally include this in the optimization problem. In some cases, the classes are linearly separable and a separating hyperplane can be found but with high variability and sensitivity to small changes. In those two cases, relieving the optimization problem can be a robust solution. This is called *support vector classifier* or *soft margin*. It is also loosely called *linear support vector machines* although the 'machines' are used to refer to this classifier when using kernels as shown in the next section.
The modified optimization problem in (2.15)-(2.18) includes the tuning parameter $C$ which allows for some points to be in the margin or even in the wrong side of the hyperplane. The *slack variables* $\epsilon_1, \ldots, \epsilon_n$ allow for each observation to violate the margin or not based on the maximum allowed violation put in $C$. You can see the influence of changing $C$ in Figure 2.3. On the top left side of the figure, when $C$ is large, a lot of points are allowed to be in the margin and on the wrong side of the hyperplane. When it decreases, the margin starts to narrow down and less points are allowed to violate the margin. This can be seen with gradual decrease of the tuning parameter $C$ in the top right, bottom left, bottom right sides of the figure respectively.

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M \tag{2.15}$$

$$\text{subject to} \quad \textstyle\sum_{j=1}^{p} \beta_j^2 = 1, \tag{2.16}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geqslant M(1 - \epsilon_i), \tag{2.17}$$

$$\epsilon_i \geq 0, \quad \textstyle\sum_{i=1}^{n} \epsilon_i \leq C, \tag{2.18}$$

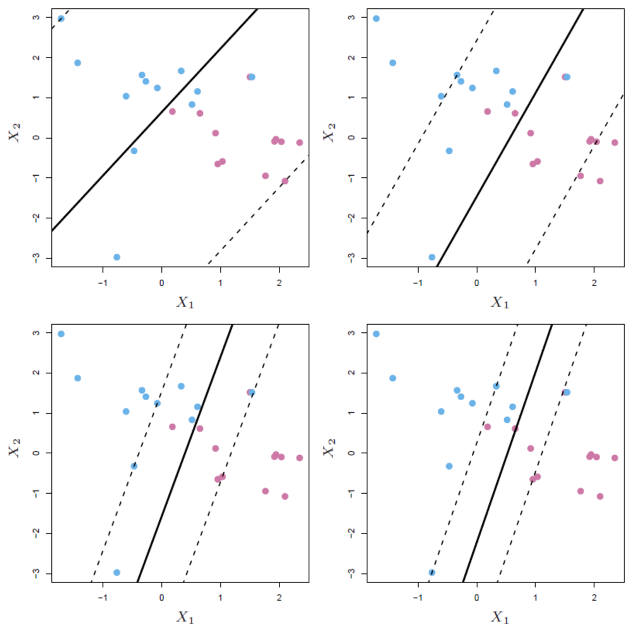


Figure 2.3: The effect of changing the tuning parameter $C$ on the margin in the support vector classifier, from [14].

### 2.4.2 Radial Basis Function Kernel

To extend the support vector classifier to separate the separable but with non-linear hyperplane, we introduce a higher dimensional feature space. Take the data in Figure 2.4 as an example of this kind of problems. Support vector machines can only create a linear separating hyperplane. The left side of the figure shows the data in the original 2-dimensional feature space. It is clearly separable but not with a linear separator. If we can move the data points into higher dimensional feature space, it can be linearly separated as in the left side of Figure 2.4. In this example, the feature space had $p = 2$ with only $X_1$ and $X_2$ as features. The new feature space has features $X_1, X_2$ and $X_1^2 + X_2^2$.

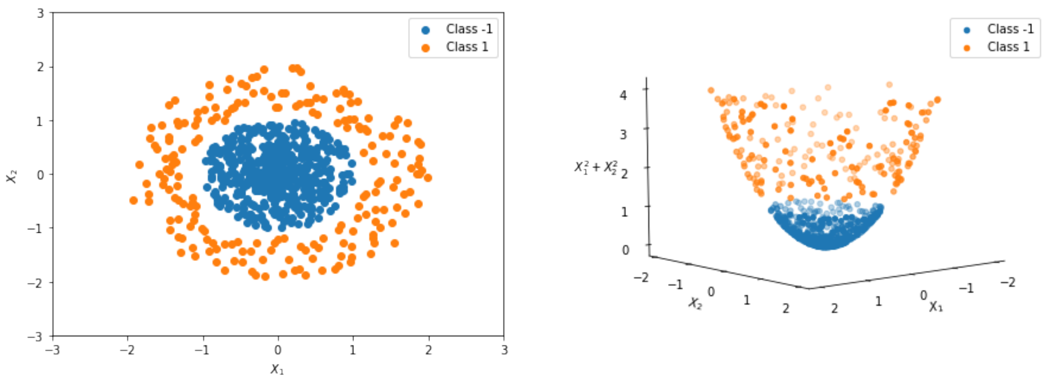


Figure 2.4: Non-linearly separable classes in 2d (left) but same data linearly separable in 3d (right).

There is no restriction of what dimensions should be added to the original feature space. It can be polynomials for each feature,$X_1^2, X_1^3, \ldots$, or interaction terms, $X_1X_2$. This can lead to a very large or infinite dimensional space if every possible dimension is considered. Due to the nature of the optimization problem in equations (2.15)-(2.18), the problem would only involve the inner products of the observations not the observations themselves [1]. It can be shown that this inner product is equivalent to some function called a *kernel*. Depending on the kernel function used, the feature space will be different. In our implementation, we use the *radial basis function* kernel in equation 2.19.

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2), \tag{2.19}$$

where $x_i$ and $x_{i'}$ are two different instances of the data and $\gamma$ is a positive constant representing the kernel width or the variance term of Gaussian distribution. It is actually a tuning parameter for each problem. Each kernel function is equivalent to the inner product of mapped observations from the original feature space to some other higher dimensional feature space. This creates the linear hyperplane in the higher dimensional space that is non-linear when mapped back to the original one.

## 2.5 Tree-Based Methods

In this section, we will discuss two of the main methods we are comparing; random forests and adaptive boosting. They are under the same section because they are both tree-based methods. Decision tree is a method used for classification and regression based on a set of splitting rules to divide the feature space. It is well-known for easy interpretation since it can be represented and summarized in a tree-like graph, see Figure 2.5.

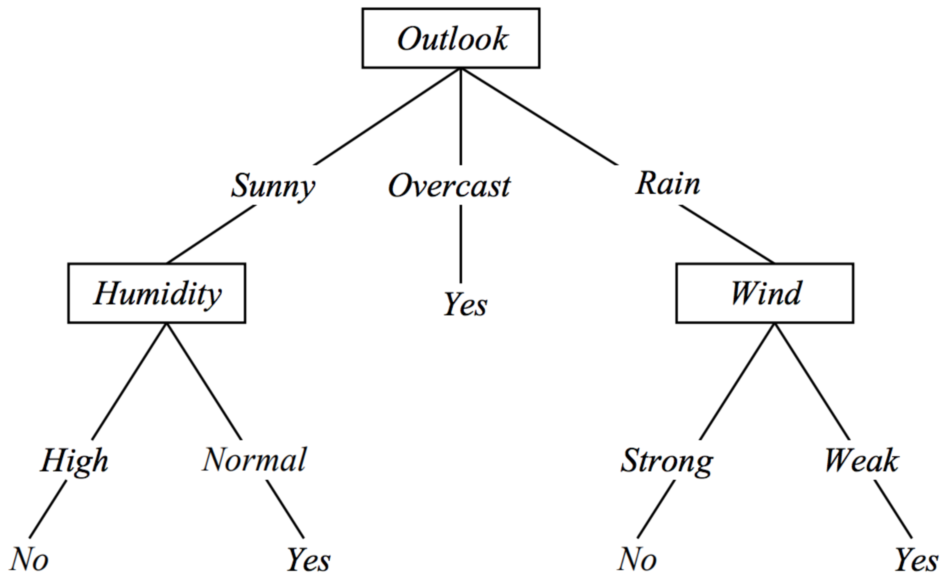


Figure 2.5: A visualization of a decision tree of whether a tennis match will be played or not based on different features (nodes), taken from [15].

The decision trees, on their own, usually do not compete with other methods in accuracy measures. Decision trees tend to overfit due to the splitting rules the algorithm deduces from the training set. There are some solutions like pruning the tree before it overfits. A slight change in the training set will result in a totally different tree and hence, different class assignment and accuracy metric. This high variation makes the decision tree a less reliable classifier than the previously mentioned ones. Aggregating different decision trees seems to reduce variance and bias and increase the predictive power of trees. Ensemble methods are those methods that aggregate classifiers such as naïve Bayes with logistic regression and Decision Trees for example, training all classifiers and vote for the majority predicted class. Here, we are just aggregating decision trees but how can the trees be different? This is discussed in the following two subsections about *Random Forests* and *Adaptive Boosting.*

### 2.5.1 Random Forests

Breiman introduced Random Forests algorithm in 2001 [16]. In Random Forests, there are two types of modifications to build up the model, a modification in the training entries and a modification in the features used for training. If we are only aggregating different decision trees with different training entries but all the features are included, we are doing what Leo Breiman called Bagging, short for Bootstrap Aggregation [17]. Bootstrapping is the process involves sampling $n$ entries from the training set with replacement. This means that every new sample will have some entries repeated if n is the same as the sample size. Around 37% of the set will be replicates. Bagging will assign a class based on each tree created to each bootstrapped training set and then have a majority voting. Another idea is combined with Bagging to form Random Forest. This idea is random selection of features or random subspace method developed by Tin Kam Hao in 1998[18]. It is the same idea of aggregating bootstrapped training sets but this time it is the same training set, but each training set has a random sample of features with replacement. This relieves the overfitting caused by considering all the features in as Decision tree. This combination appearing in Random Forests meta-algorithms proves to significantly improve the classification accuracy. Random Forests is only sensitive to the maximum number of features chosen randomly to construct the tree and the default value in scikit-learn is the floor of the square root of the number of features $\sqrt{p}$. However, random forests is fairly robust to the choice of this hyperparameter when the number of trees is relatively large.

The number of trees in our implementation was set to 500. We could also parallelize the implementation to speed the run time up using scikit-learn function arguments.

### 2.5.2 Adaptive Boosting

Instead of building a number of trees at the same time as in Random Forests, Adaptive Boosting (AdaBoost) builds trees *sequentially*. Based on the information it gets from the previous tree, it tries to boost the performance of the next tree to avoid the previously misclassified points. This takes place by giving more priority to the those points by increasing their weights, graphically represented in Figure 2.6.

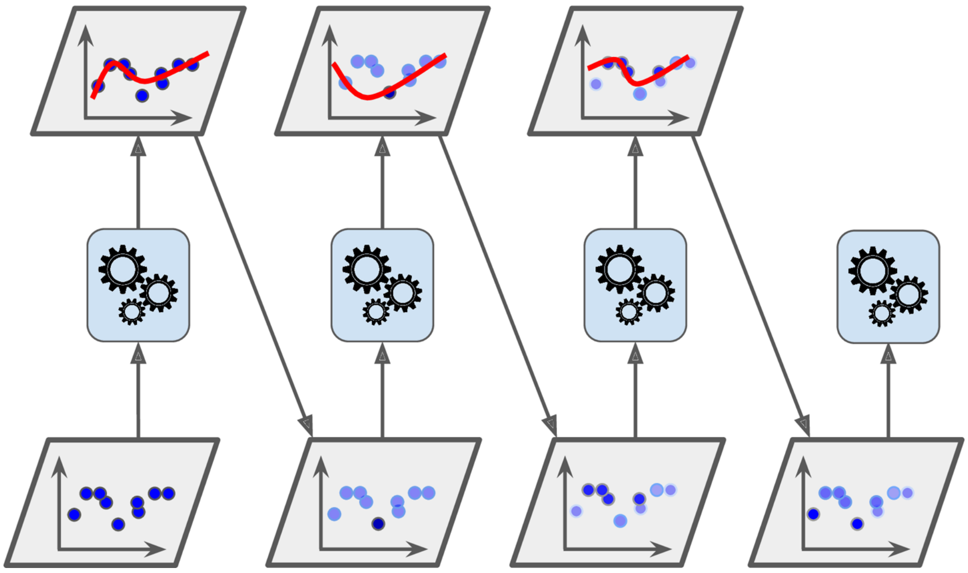


Figure 2.6: AdaBoost sequentially building trees and updating weights[19].

The algorithm starts with giving equal weights for all the points $w_i = \frac{1}{n}$. Then, it builds the first tree based on the equal weights and checks how much the weighted error rate is:

$$r_j = \frac{\sum_{i:\hat{y}_i \neq y_i}^{n} w_i}{\sum_{i=1}^{n} w_i}, \quad \text{for all trees } j = 1, 2, \ldots, T \tag{2.20}$$

then this tree should have a weight $\alpha_j$ for later voting contribution based on how much it is accurate in classification,

$$\alpha_j = \log \frac{1 - r_j}{r_j}, \tag{2.21}$$

the weights of points are then updated,

$$w_i = \begin{cases} w_i & \text{if } \hat{y}_i = y_i \\ w_i \ \exp(\alpha_j) & \text{if } \hat{y}_i \neq y_i \end{cases} \tag{2.22}$$

and normalized by dividing by $\sum_{i=1}^{n} w_i$. A new tree is built based on the new weights and the process repeats until the required number of trees $T$ or an optimal tree reached. The predictions will be based on the weighted votes for each tree. Formally:

$$\hat{y} = \underset{c}{\text{argmax}} \sum_{j:\hat{y}_j = c}^{T} \alpha_j, \quad \text{where } c \text{ is the class.} \tag{2.23}$$

In our implementation, $T$ is chosen to be 200 without any change in the default parameters, namely each decision tree is of one node and two leaves, and the learning rate is 1.

# 3 Methodology

Our methodology has been implemented in *python* programming language. The vector operations and data manipulations have been done through numpy and pandas libraries [20], [21]. The machine learning library was scikit-learn [22]. Matplotlib and seaborn libraries were used for visualizations [23], [24].

In this chapter, we discuss in detail our methodology of comparison between statistical learners, the tuning of parameters for some learners, cross validation and performance metric.

## 3.1 Data Handling

Since we are working on real-world data sets and there is no previous knowledge about the data, exploratory data analysis has been carried out for each data set before fitting the models. This step is important to know the data types (numeric, categorical), number of classes and the data size. It is also important to see the distribution of each feature because some statistical learners have some assumptions about the data distributions as seen in naive Bayes section 2.1. If the assumptions of a classifier did not match what is seen in the exploration, so we do not expect this classifier to perform well. Almost all the data used in this study was numeric real values with some exceptions of integer and categorical (0 or 1) values.

In the explorations, we make sure no missing data exists before training the models. There are multiple ways of dealing with missing data such as imputing them with means or interpolating between points [25]. The pitfalls of those traditional techniques and comprehensive review of modern techniques can be see in [26]. For the scope of this study, the instances that had missing values have been omitted. After deleting the missing data instances, 20% of the data, randomly chosen, is put aside as a test set $(\mathbf{X}_{test}, \mathbf{y}_{test})$. The remaining 80% of the data,the training set$(\mathbf{X}_{train}, \mathbf{y}_{train})$, is further split for cross validation as explained in section 3.3.

A critical preprocessing step for some algorithms is to scale the features before fitting the model in order for all the features to have the same influence on the model and not be dominated by the large-valued features. Logistic regression, $K$-nearest neighbors and support vector machines need this step [27] while the other classifiers are not distance-based and hence scaling will not make a difference. There are many ways of feature scaling discussed in the literature in many different applications [28]. A typical scaling approach is *linear scaling* or sometimes called Min-Max scaling. It means squeezing or expanding all the values for each feature to values that range between $[0, 1]$. This takes place by this transformation for each value:

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}. \tag{3.1}$$

This range can be change to $[a, b]$ where $a, b \in \mathbb{R}$ and $a < b$. In our study, we implemented typical linear scaling with $a = 0$ and $b = 1$.

## 3.2 Hyperparameter Tuning

The parameters that are not directly optimized by the classifier are called hperparameters. For instance, the support vector classifier is trying to learn the best hyperplane parameters $\beta_0, \beta_1, \ldots, \beta_p$ in equations (2.15)-(2.18), hence $\beta_0, \beta_1, \ldots, \beta_p$ are parameters. On the other hand, the penalty $C$ is not directly learned, so it is considered a hyperparameter and can be freely chosen based on the problem in hand. In our comparison setting, three out of the seven classifiers need a careful choice of hyperparameters because it affects their performance dramatically. Those classifiers are $K$-nearest neighbors, linear support vector machines and RBF-kernel support vector machines. For those classifiers, we have searched for the best hyperparameters based on the data set. For $K$-nearest neighbors, we search for the best $K$ among all integers from $K = 1$ to $K = 20$ and the $K$ with the best reported accuracy is used to train the model and test it. For support vector machines, the linear classifier has the penalty $C$ as the only hyperparameter whereas the RBF-kernel support vector machines classifier has two hyperparameters, the penalty $C$ and the kernel width $\gamma$.

In scikit-learn python's library, the implementation of support vector machines is based on LIBSVM developed in 2000 [29]. The optimal values for the $C$ and $\gamma$ to train the classifier are based on a grid search $C = [2^{-5}, 2^{-3}, \ldots, 2^{15}]$ and $\gamma = [2^{-15}, 2^{-13}, \ldots, 2^{3}]$, from the practical recommendation in [30].

## 3.3 Cross Validation

Splitting the data into training set $(\mathbf{X}_{train}, \mathbf{y}_{train})$ and test set $(\mathbf{X}_{test}, \mathbf{y}_{test})$, as described in section 3.1, allows us to assess the performance of the fitted model to a new set of data that the model has not seen while being trained. To gain more information about the model before testing it and to optimize the hyperparameters, we can deploy one of the well-known *resampling methods*. Resampling methods are tools for estimating the variability of a model if only a random subset of the data is considered in building the model. Such methods provide a measure of variation which are not accessed otherwise. One of the resampling methods is briefly discussed in section 2.5.1 which is bootstrapping. We discuss here one of the most used resampling methods, *cross validation*.

Cross validation is simply randomly splitting the training set into approximately equal-sized $k$ group, or *folds*. $k$ models will be built, each time one of the folds will be a *validation set*, the remaining $k - 1$ folds will constitute the training set. For example, if $k$ is chosen to be 5, the splitting will be as shown in Figure 3.1. It can be chosen to be equal $n$, and then called *Leave-One-Out Cross Validation* (LOOCV). This can be computationally intensive since it requires building $n$ models. In our experiment setting, we applied 10-fold cross validation which is common in the literature. For statistical confidence, the 10-fold cross validation has been repeated 10 times.

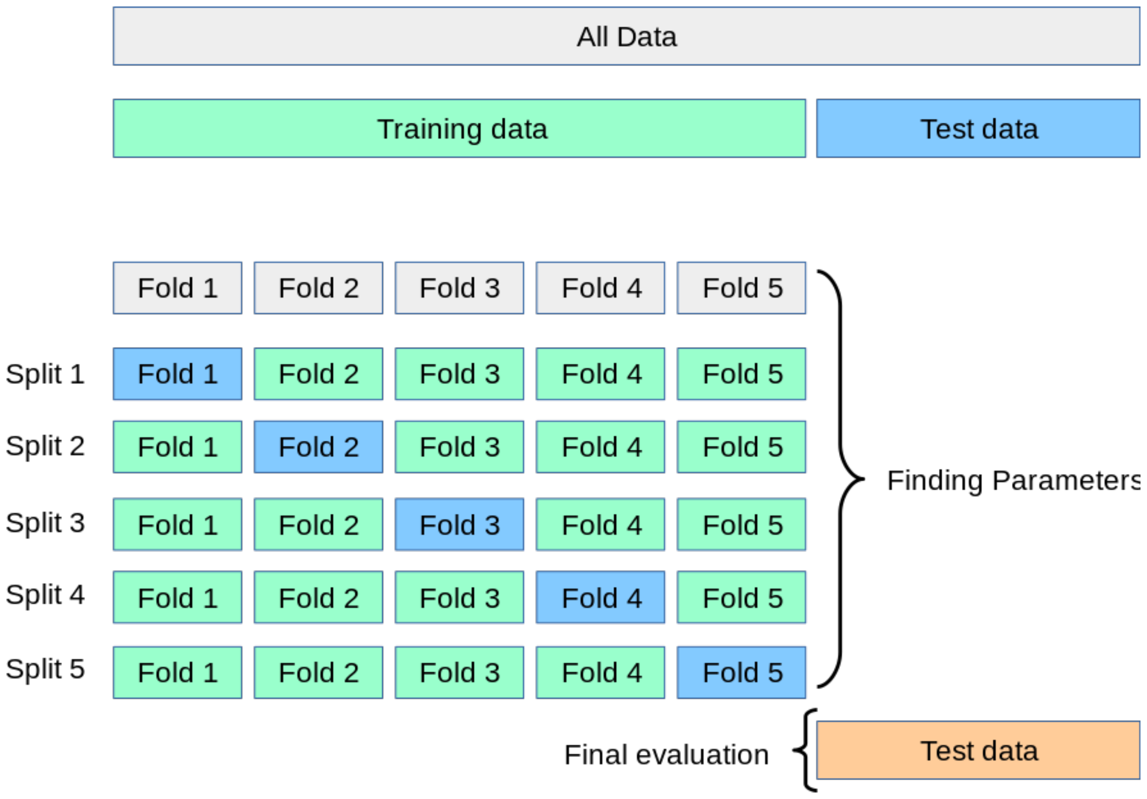
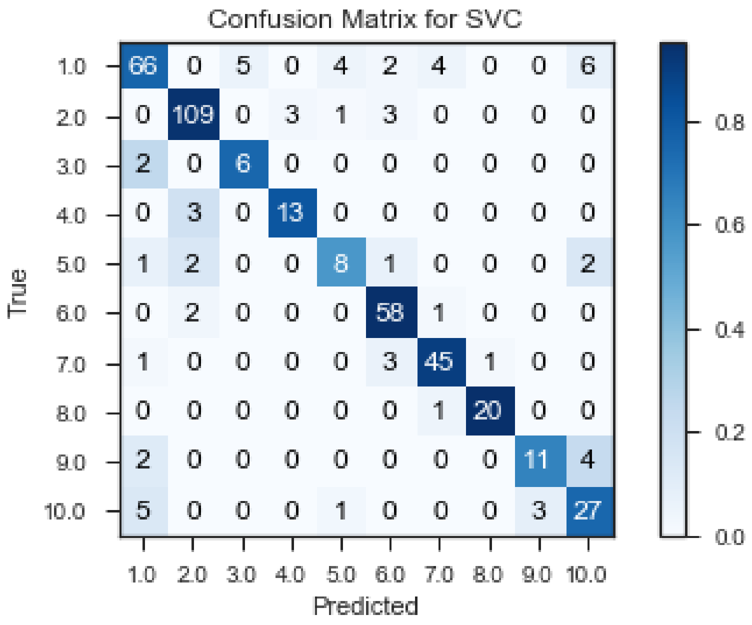


Figure 3.1: Splitting the data with 5-fold cross validation setup, taken from [22].

## 3.4 Model Evaluation and Performance Metrics

Numerous metrics have been developed to quantify the quality of classifiers. There are some metrics specific for binary classifications and some others for multi-class classification. There are also some metrics that can be used for both settings. *Accuracy* is the classical measure of performance in classification. It is the ratio of correctly classified instances to the total number of instances you were trying to classify. The other side of that same metric is the *error*, and it is the ratio of misclassified instances from all instances. Some of the commonly used metrics are f-score, Kappa statistic, area under the Receiving Operating Characteristics (ROC) curve and the fitting time [3]. Put the fitting time aside, all the classifiers are based on the numbers that can be summarized and seen in a *confusion matrix*. The confusion matrix simply shows how each instance is classified either correctly or incorrectly, illustrating which classes the classifier confuses. For example, Figure 3.2 shows a 10-class setting, the CTG data set, and how linear SVM confuses and correctly classifies the instances.



| True \ Predicted | 1.0 | 2.0 | 3.0 | 4.0 | 5.0 | 6.0 | 7.0 | 8.0 | 9.0 | 10.0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1.0 | 66 | 0 | 5 | 0 | 4 | 2 | 4 | 0 | 0 | 6 |
| 2.0 | 0 | 109 | 0 | 3 | 1 | 3 | 0 | 0 | 0 | 0 |
| 3.0 | 2 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4.0 | 0 | 3 | 0 | 13 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5.0 | 1 | 2 | 0 | 0 | 8 | 1 | 0 | 0 | 0 | 2 |
| 6.0 | 0 | 2 | 0 | 0 | 0 | 58 | 1 | 0 | 0 | 0 |
| 7.0 | 1 | 0 | 0 | 0 | 0 | 3 | 45 | 1 | 0 | 0 |
| 8.0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 20 | 0 | 0 |
| 9.0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 4 |
| 10.0 | 5 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 27 |

Figure 3.2: Confusion matrix for 10-class setting, linear SVM on CTG data set.

The diagonal entries are the correctly classified instances. The off-diagonal entries are showing details about the misclassfied instances. For example, linear SVM confuses class (1) for class (3) 5 times and for class (5) 4 times as shown in the first row but correctly classified class (1) 66 times. For the clarity and the information the confusion matrix provides, we choose to show all the results in confusion matrices and can be investigated on the Github repository. We have also reported the classifier accuracy along with their Inter-Quartile Range (IQR) for purposes of comparison.

## 4 Data Sets

There are two paradigms to compare the performance of classifiers. One is to generate artificial data sets and the other is to use real-world ones. Creating artificial data sets is robust in terms of experiment design since you have the flexibility to choose the number of features, number of entries, number of classes and data types. The drawback of this paradigm is that you might draw conclusions about classifiers that are not pertained to real data due to the hidden characteristics of the data sets that are not taken into account when generating the artificial ones. For this reason, the comparison in this study is based on real-world challenging and diverse data sets. The data sets are chosen to include a fair range of data size, number of features and numeric real and integer data types. Some data sets have categorical data types but with only two categories. Based on the number of predicted classes, the data sets are categorized into binary-class and multi-class data sets. All the data is sourced from *UCI Machine Learning Repository* [31] unless mentioned otherwise in the following sections.

### 4.1 Binary Classification

1. **Breast Cancer Wisconsin Diagnostic:** Early diagnostic of diseases is of concern to both patients and clinicians. Modern technology can help detect the hidden pattern in features. This data set has 30 features.They are derived from images of fine needle aspirate of a breast mass. Those features represent 10 geometric and shape features of 3 nuclei which comprises 30 features for 569 cases. Each case is labeled to have a malignant (coded 1) or benign (coded 2) tumor. All features are real-valued representing nucleus radius, standard deviation of grey-scale values, perimeter, area , smoothness, concavity, concave points, symmetry, and fractal dimension. The classifier should predict if the tumor is malignant or benign based on those 30 features.

2. **Sonar:** Signal processing experts extracts some characteristic features such as energy within frequency bands. The 60 features in this data set represent this energy values integrated over a certain period of time. Bouncing sonar signals off rock and metallic cylinder at different angles resulted in the 208 entries of this data set. The classifier is supposed to predict whether the feature vector consisting of 60 energy values is for a rock or a metallic cylinder.

3. **Congressional Voting:** In the social and political sciences, the prediction of the individual's behavior in consumption or voting is critical to policy makers and governments. This data set originally had 435 congressmen in the U.S. House of Representatives' votes for republicans or democrats, this is the class to be predicted. The 16 features are other votes for the same congressmen expressing their opinion on other country issues such as if they agree to have religious groups in schools, if they support immigration or against it and so on. Some people did not vote or voted to avoid conflict and are considered null values and removed in the preprocessing step. Only 232 entries were analyzed. For more information, refer to the data set source link on github corresponding notebook.

4. **Banknote Authentication:** Fraud detection is one of the critical applications of statistical learners. This data set is taken originally from 1372 banknote-like images. The features are extracted from the images by wavelet transform tool. Although the images were $400x400$ pixels, the extracted features are only 4. The classifier should predict if the banknote is genuine or forged.

5. **Acute Inflammation:** As in the breast cancer diagnosis learning, this data set is for another disease diagnosis, acute inflammation of urinary bladder. It has 6 symptoms of 120 patients only. The symptoms act as features, one continuous integer feature which is the temperature and the remaining 5 are categorical with yes or no answers, such as if the patient has nausea, lumbar pain, etc. The data set also had another label to predict

but we only used the acute inflammation because multi-labeling is out of the scope of this thesis.

6. **Spam base:** Traditional problem in machine learning is to identify an email as spam or not based on the word usage, frequencies, length of uninterrupted sequences of capital letters, and some other features. This data set has 4601 emails with 57 features.

Table 4.1 summarizes the binary data sets used in this study.

| | $n$ | $p$ | Data Types | Missing Values |
|---|---|---|---|---|
| 1. Cancer | 569 | 30 | Real | no |
| 2. Sonar | 208 | 60 | Real | no |
| 3. Voting | 435 | 16 | Categorical | yes |
| 4. Banknote | 1097 | 4 | Real | no |
| 5. Inflammation | 120 | 6 | Categorical, Integer | no |
| 6. Spambase | 4601 | 57 | Real, Integer | yes |

Table 4.1: Binary classification data sets summary.

## 4.2 Multi-class Classification

1. **Image Segmentation:** There are seven outdoor images in this data set and they are the classes at the same time. The 2310 entries are parts of the images, each entry is a $3x3$ pixels region of the image. Some algorithms have been applied for feature extraction such as how many lines go through the region, the contrast of horizontally and vertically adjacent pixels to detect horizontal and vertical edges, the intensity mean value, the average of each color over the region (RGB), the excess of each color, and other features to comprise 19 features in total.

2. **Cardiotocography:** This data set is a health care application using signal processing techniques. Cardiotocgraphy (CTG) is known to be the way of monitoring the fetal well-being, heartbeat, uterine contractions during pregnancy. Each fetus was assessed to be normal, suspect or pathological by the consensus of three experts. The entries are also classified based on morphological patterns to 10 classes. It can be used for 3-class or 10-class classification problems. The features extracted from the CTG were 21 and represented beats per minute, number of fetal movements per second, number of uterine contractions per second, number of light and severe deceleration per second and some other measures of abnormal variability.

3. **Lee Silverman Voice Treatment (LSVT):** Parkinson's disease patients are known to have degradation in their vocal performance. LSVT Companion is a computer program to help patients through their rehabilitation process. Based on vowel phonation in different settings with diverse pitches and loudness, 14 subjects recorded 9 phonations each with total of 126 data points. The extraction of useful features out of voice signals are called dysphonia measures. For this data set, 309 dysphonia measures, i.e. features, are extracted which makes it a challenging data set with number of features almost 2.5 times the number of instances. The are 3 classes showing how acceptable the phonation is accepted,'1' indicates acceptable phonation, '2' indicates loud phonation, and '3' indicates pressed phonation. This data set is studied in [32] as a binary classification problem with '1' indicates and '2' and '3' indicate not accepted.

|  | n | p | Data Types | number ofclasses | Missing Values |
|---|---|---|---|---|---|
| Image | 2310 | 19 | Real | 7 | no |
| Cardio | 1700 | 21 | Real | 10 | no |
| LSVT | 100 | 310 | Real | 3 | no |

Table 4.2: Multi-class classification data sets summary.

# 5 Results

In this chapter, we will show the detailed results and analysis for one of the data sets to give an idea of how the procedure has been implemented for all the other data sets. For the analysis of the remaining data sets, please visit the Github repository. We will also compare the performance of the methods in terms of accuracy and model fitting time for both binary and multi-class problems.

## 5.1 Indicative Full Data Analytics Methodology on a Single Dataset

Our data set of choice is the cancer diagnostics with feature details summarized in Chapter 4. The first step in our analysis is to run exploratory data analysis. This analysis is to know the data type, data distribution, size of data, dimensionality or number of features, number of classes, classes balance, correlations between features and anything else worth to note. We can see in this data set, cancer diagnostics, that all the features are continuous and almost normally distributed and some distributions are skewed in Figure 5.1. Features are shown as numbers from '0' to '29' and refer to the extracted features from the images of a fine needle aspirate of a breast mass, as mentioned in detail in the corresponding notebook on Github. The class is '1' for malignant cancer and '2' for benign.

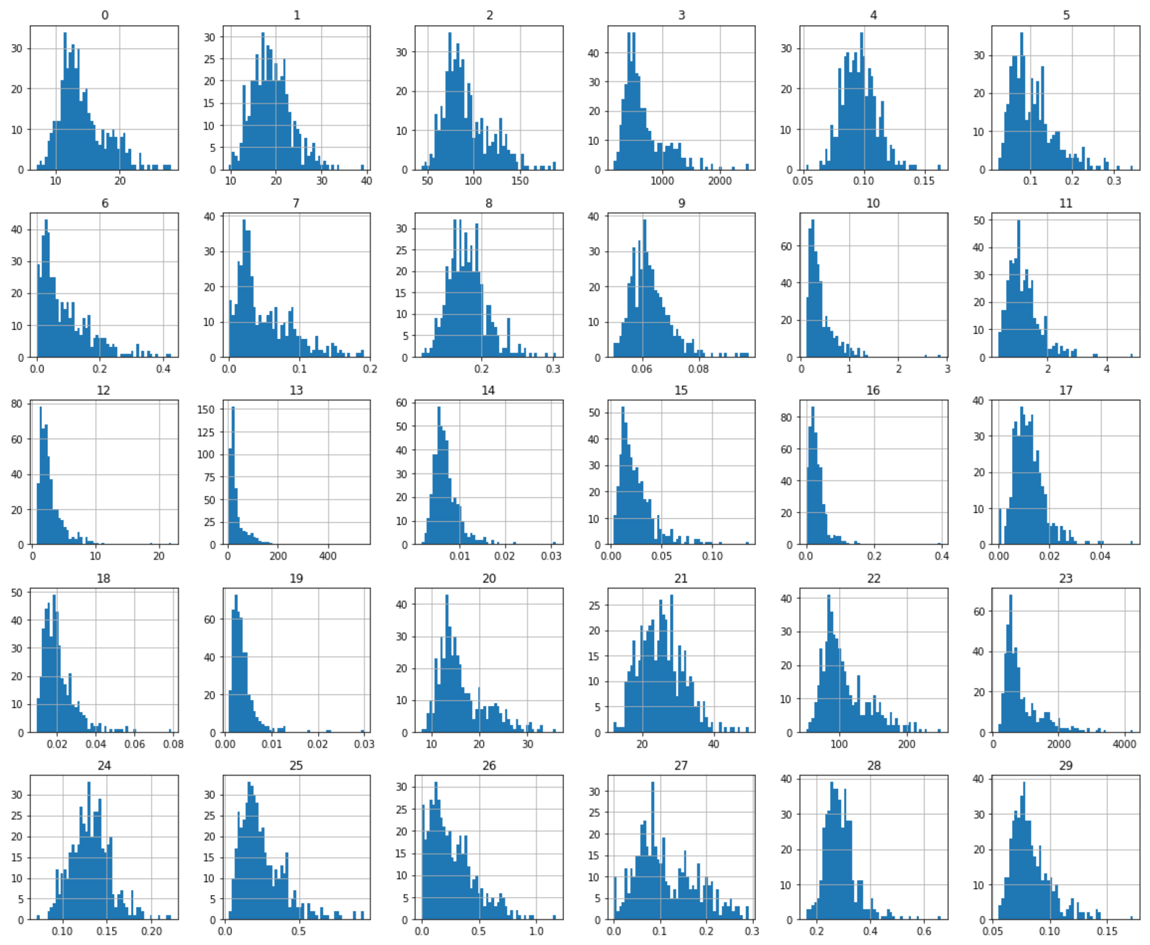


Figure 5.1: Histogram of features in Cancer Diagnostics data set showing close to Gaussian distribution for each feature with continuous numeric type of data. Features are coded from 0 to 29 and can be deciphered and known in their respective reference.

It is also insightful to visualize the correlation between features and graph the most correlated features with each others and more importantly with the response variable, the class. Figure5.2

shows the heat map of the correlation and Figure 5.3 focuses on the 8 most correlated features with the class and draws them with each other with separate colors for each class.

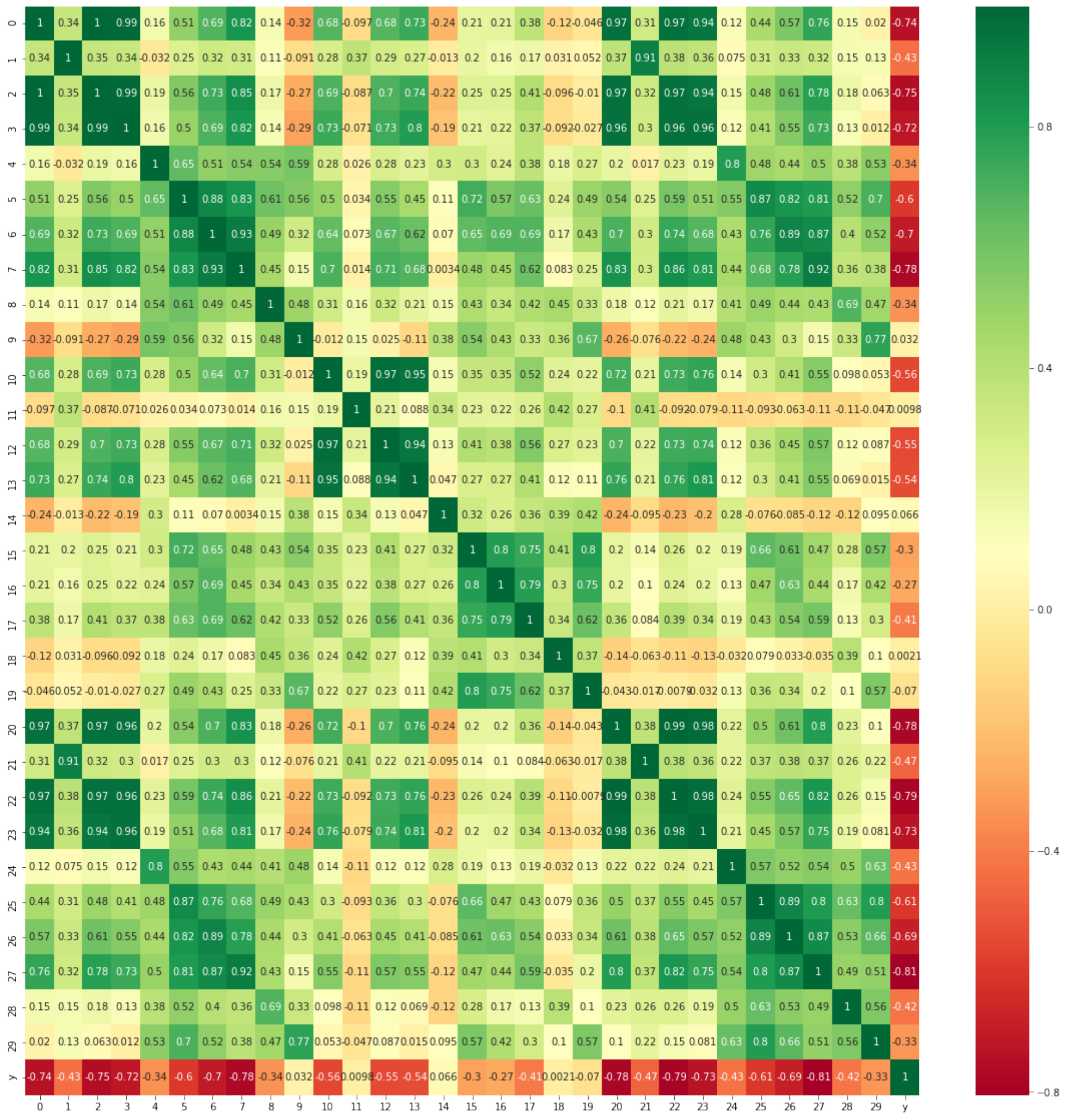


Figure 5.2: Heat map of correlations between features, and between class and features.

There appears to be very high correlations between some features. This is expected since they are geometric dimensions of cell nuclei. For example in Figure 5.3, feature 0 is the first nucleus radius and feature 2 is the perimeter of the same nucleus and they are linearly correlated with a line slope of almost $2\pi$. It is also clear that feature 3 which is the area is correlated but polynomially with order 2. Those notes are critical to know and serve to justify why naive Bayes performs the worst in all other classifiers; the features are highly correlated and naive Bayes assumes the features are independent. Figures 5.4-5.10 shows the confusion matrices to clearly see the performance of each classifier both on the total of all validation sets and on the test set.

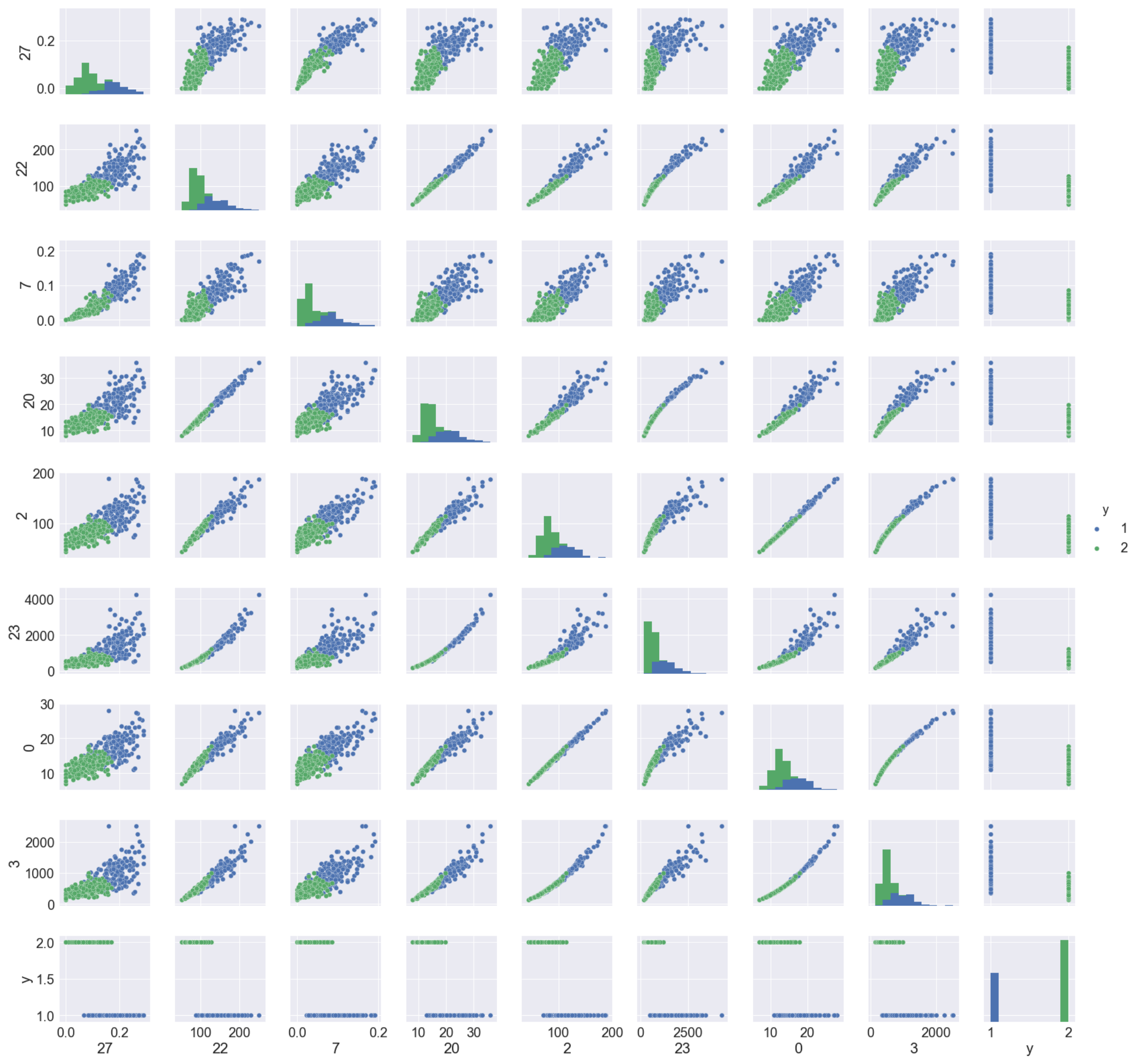
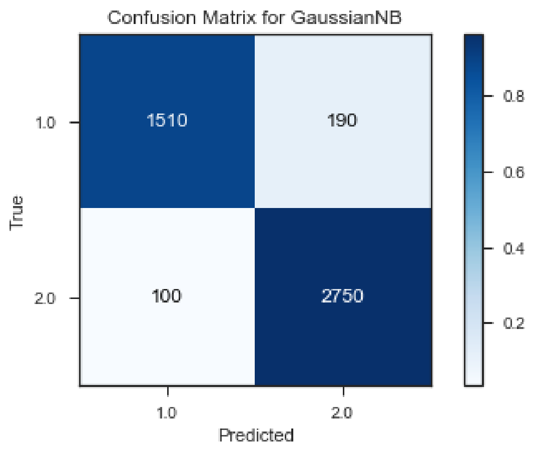


Figure 5.3: Scatter plots of the 8 most strongly correlated features with the class.

(a) Validation Set



(b) Test Set

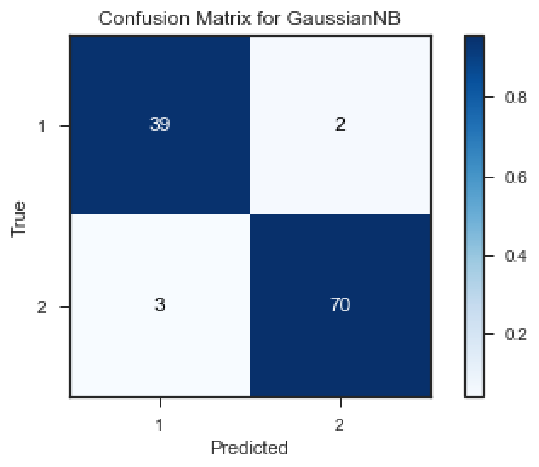
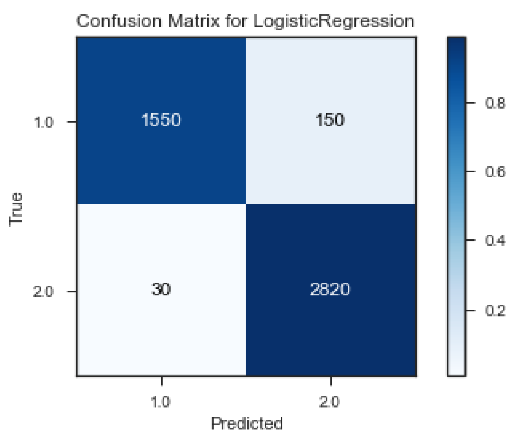


Figure 5.4: Cancer diagnostics **naive Bayes** confusion matrices for (a) validation and (b) test sets.

(a) Validation Set



(b) Test Set

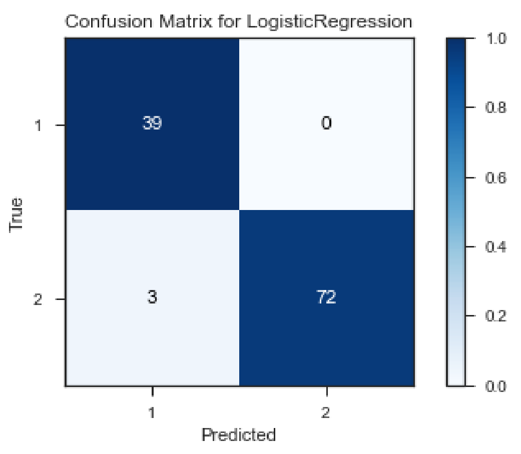
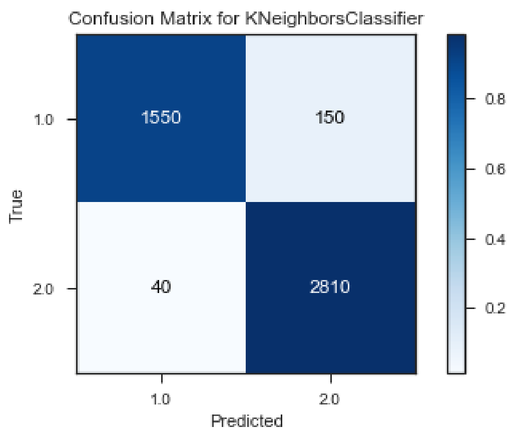


Figure 5.5: Cancer diagnostics **logistic regression** confusion matrices for (a) validation and (b) test sets.

(a) Validation Set



(b) Test Set

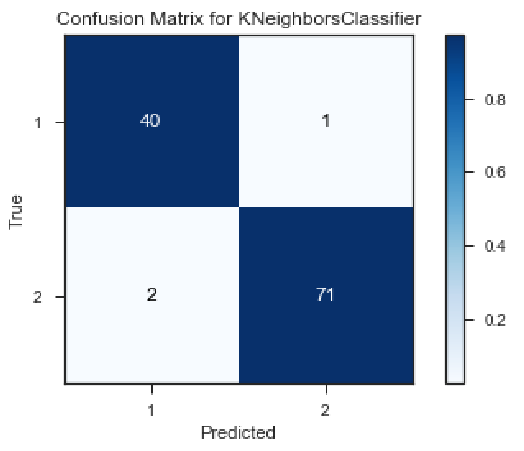
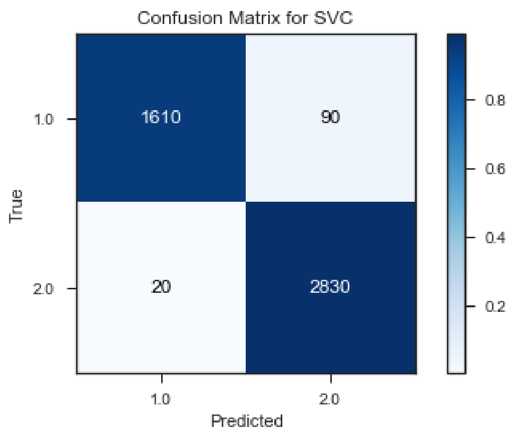


Figure 5.6: Cancer diagnostics $K$**-nearest neighbors** confusion matrices for (a) validation and (b) test sets.

(a) Validation Set



(b) Test Set

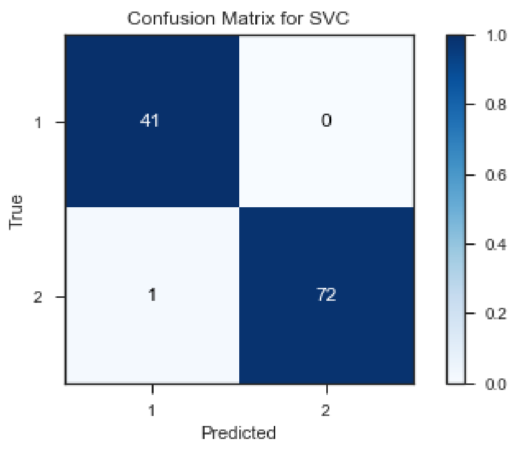
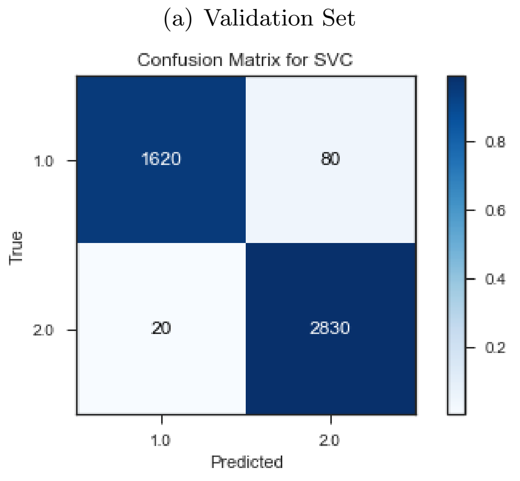


Figure 5.7: Cancer diagnostics **linear SVM** confusion matrices for (a) validation and (b) test sets.



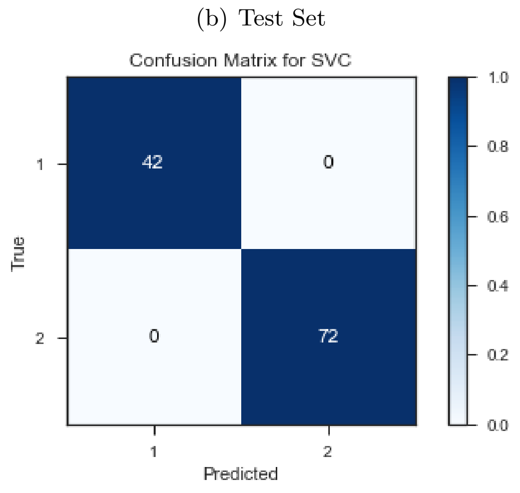
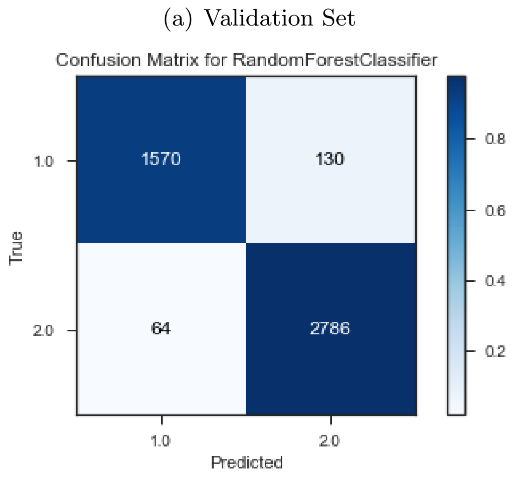


Figure 5.8: Cancer diagnostics **RBF-kernel SVM** confusion matrices for (a) validation and (b) test sets.



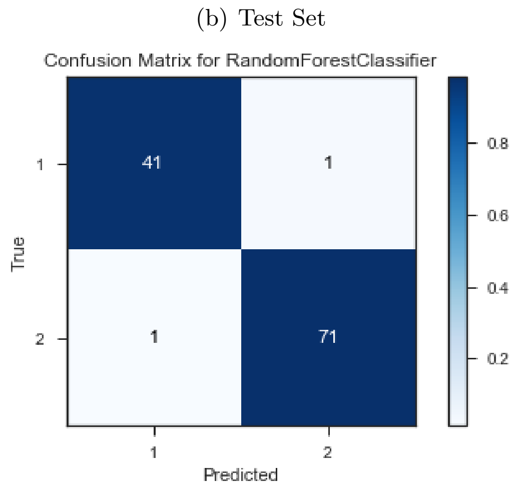
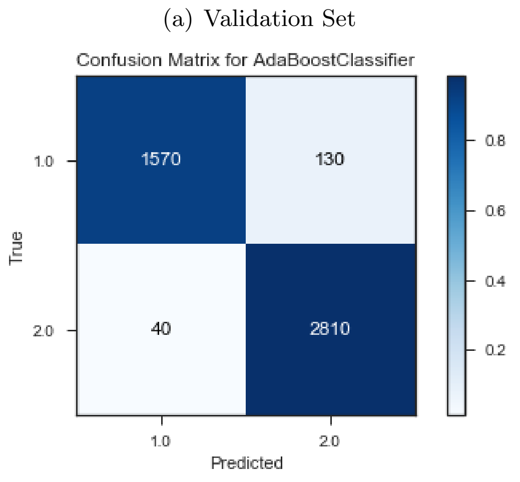


Figure 5.9: Cancer diagnostics **random forests** confusion matrices for (a) validation and (b) test sets.



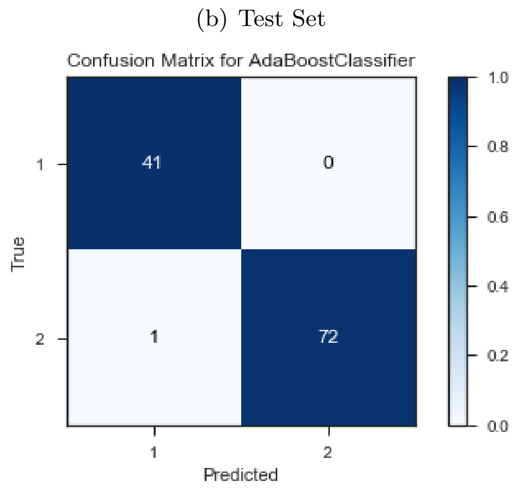


Figure 5.10: Cancer diagnostics **AdaBoost** confusion matrices for (a) validation and (b) test sets.

This data set is showing relatively high accuracy for most of the classifiers with no less than 91% accuracy for any of the classifiers. Figure 5.11 shows that for this data set linear SVM and RBF-kernel SVM have the same accuracy and the same confidence. They also had the same accuracy on the test set.

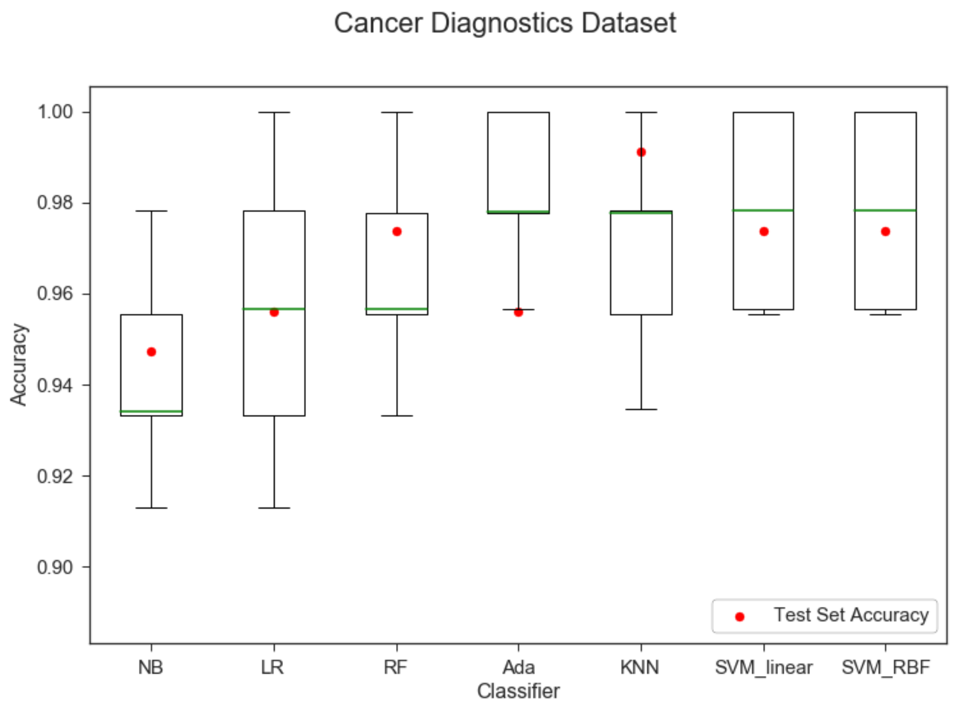


Figure 5.11: Cancer diagnostics summary accuracy results showing box plots for the cross validation accuracy (median: green bar) and test set accuracy (red dots).

## 5.2 Classifier Comparisons in Binary Classification Data Sets

In binary classifications, we had 6 data sets. One of them has been shown in detail, section 5.1. For brevity and to save on space, we present only the final classification accuracy comparisons for the remaining 5 data sets rather than the full methodology used to explore the data.

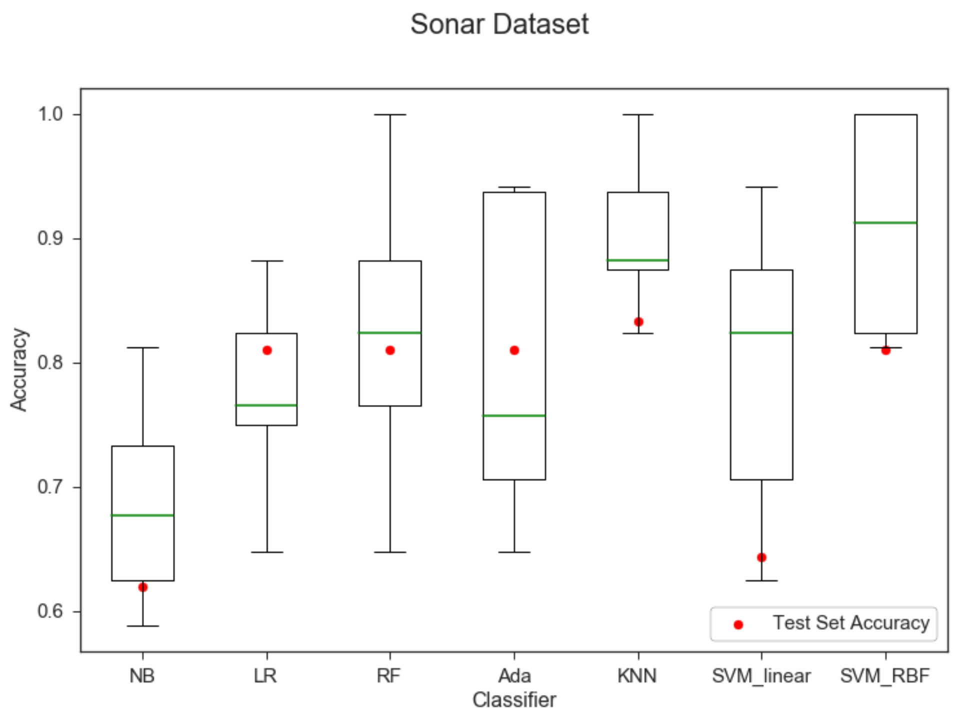


Figure 5.12: Sonar summary accuracy results showing box plots for the cross validation accuracy (median: green bar) and test set accuracy (red dots).

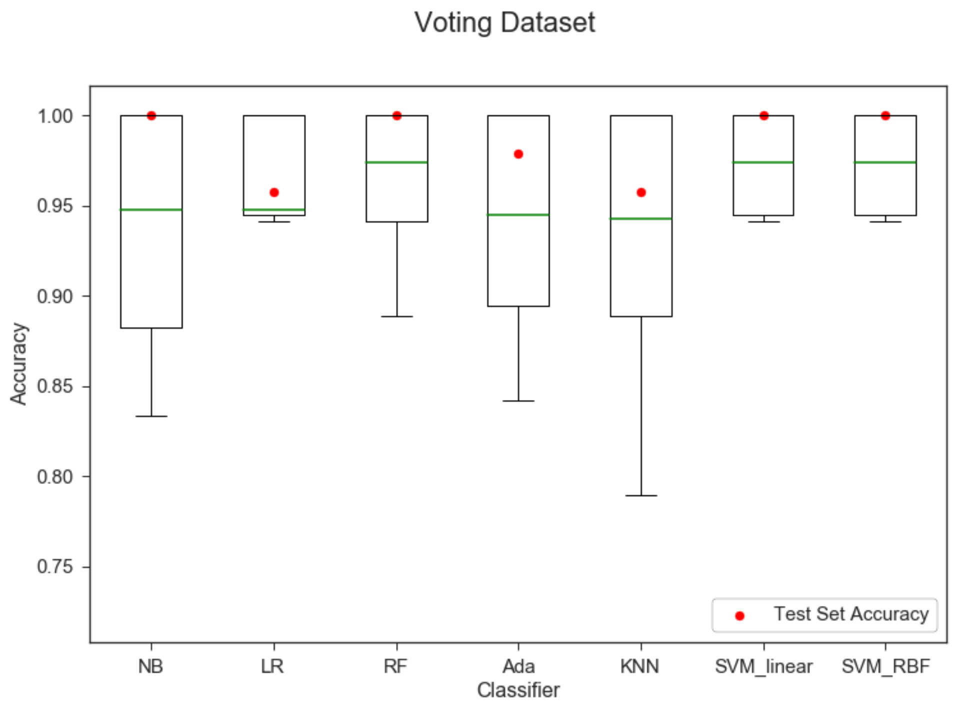


Figure 5.13: Voting summary accuracy results showing box plots for the cross validation accuracy (median: green bar) and test set accuracy (red dots).

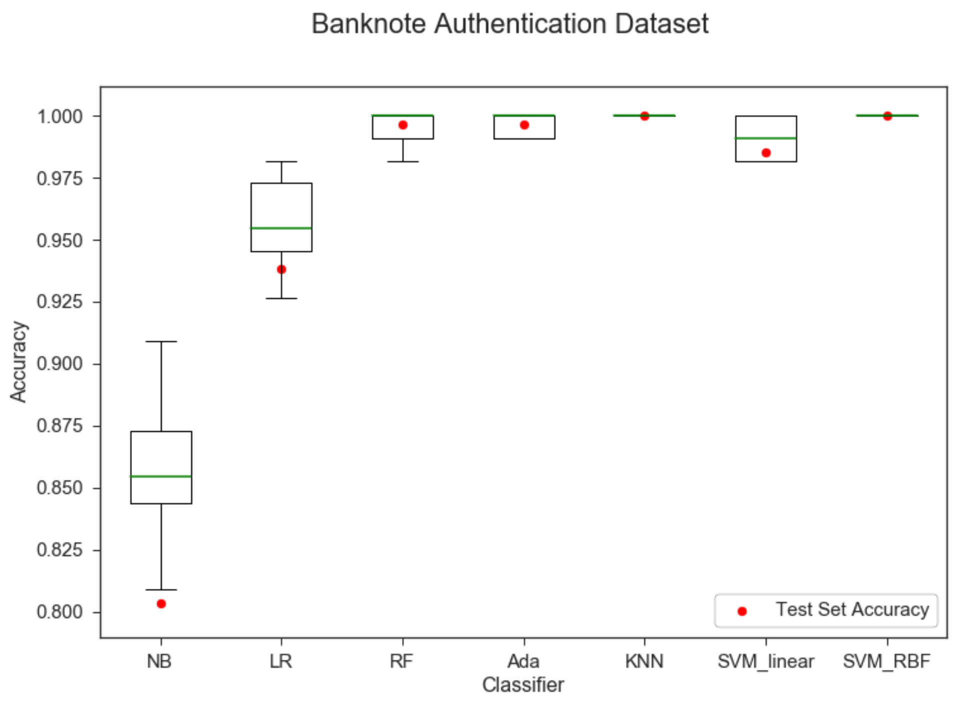


Figure 5.14: Banknote authentication summary accuracy results showing box plots for the cross validation accuracy (median: green bar) and test set accuracy (red dots).

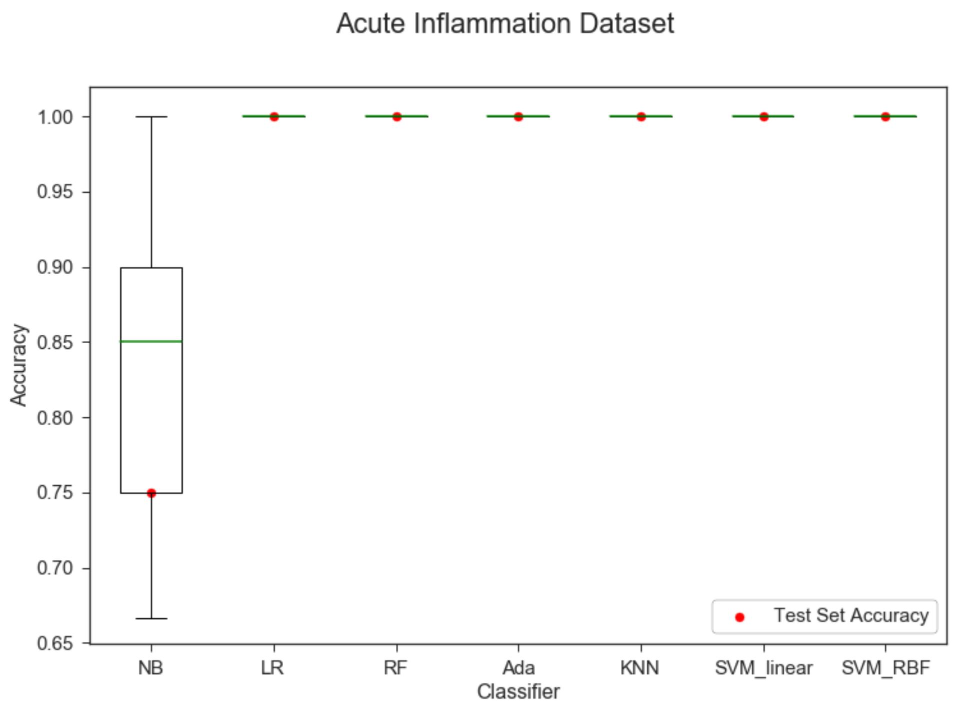


Figure 5.15: Acute inflammation summary accuracy results showing box plots for the cross validation accuracy (median: green bar) and test set accuracy (red dots).

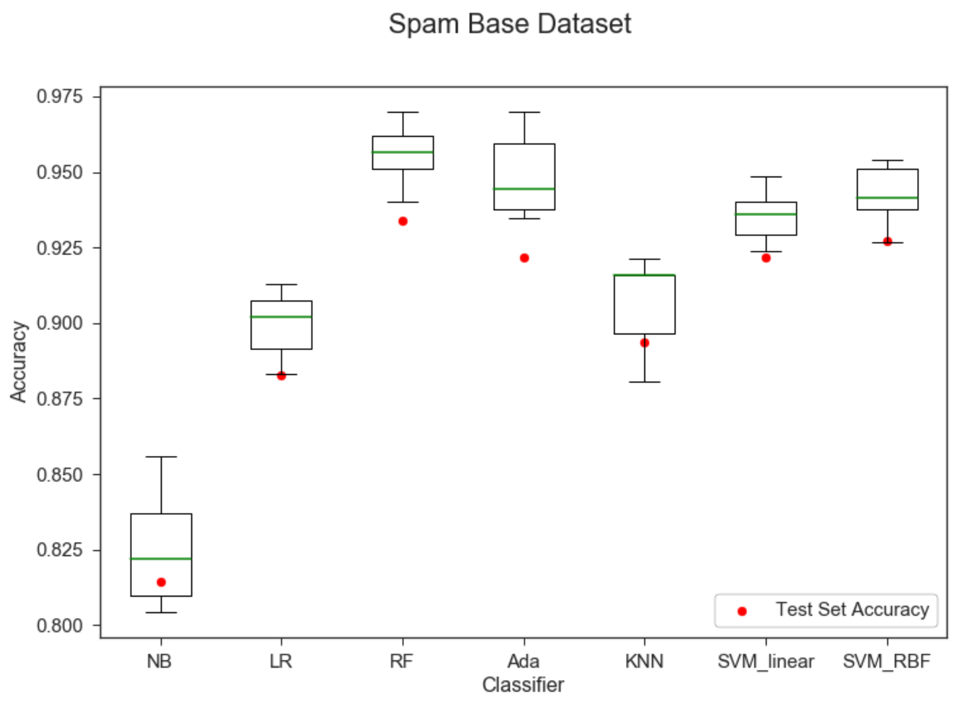


Figure 5.16: Spam base summary accuracy results showing box plots for the cross validation accuracy (median: green bar) and test set accuracy (red dots).

The summary figures 5.11-5.16 and Table 5.1 show that RBF-kernel SVM performs the best in 5 out of 6 binary classification data sets whereas naive Bayes is the least performing in 5 data sets as well. Linear SVM, KNN and RF are competing on the second performing in the same type of problems. Those three classifiers need less time for training than RBF-kernel SVM and hence better if the computational time and power are limited. Figure 5.17 summarizes the mean validation accuracy for all classifiers in all data sets.

| | **NB** | **LR** | **RF** | **Ada** | **KNN** | **SVM_L** | **SVM_RBF** |
|---|---|---|---|---|---|---|---|
| **Cancer** | 0.941 | 0.963 | 0.960 | 0.965 | 0.976 | 0.985 | 0.985 |
| **Sonar** | 0.704 | 0.782 | 0.854 | 0.781 | 0.892 | 0.784 | 0.916 |
| **Voting** | 0.968 | 0.968 | 0.975 | 0.952 | 0.919 | 0.979 | 0.979 |
| **Banknote** | 0.838 | 0.951 | 0.992 | 0.997 | 0.999 | 0.991 | 1.000 |
| **Inflam** | 0.848 | 1.000 | 1.000 | 0.997 | 1.000 | 1.000 | 1.000 |
| **Spambase** | 0.827 | 0.893 | 0.954 | 0.943 | 0.899 | 0.931 | 0.941 |

Table 5.1: Binary classification mean validation accuracy, color scale for each data set displaying red as the lowest and green as the highest accuracy.

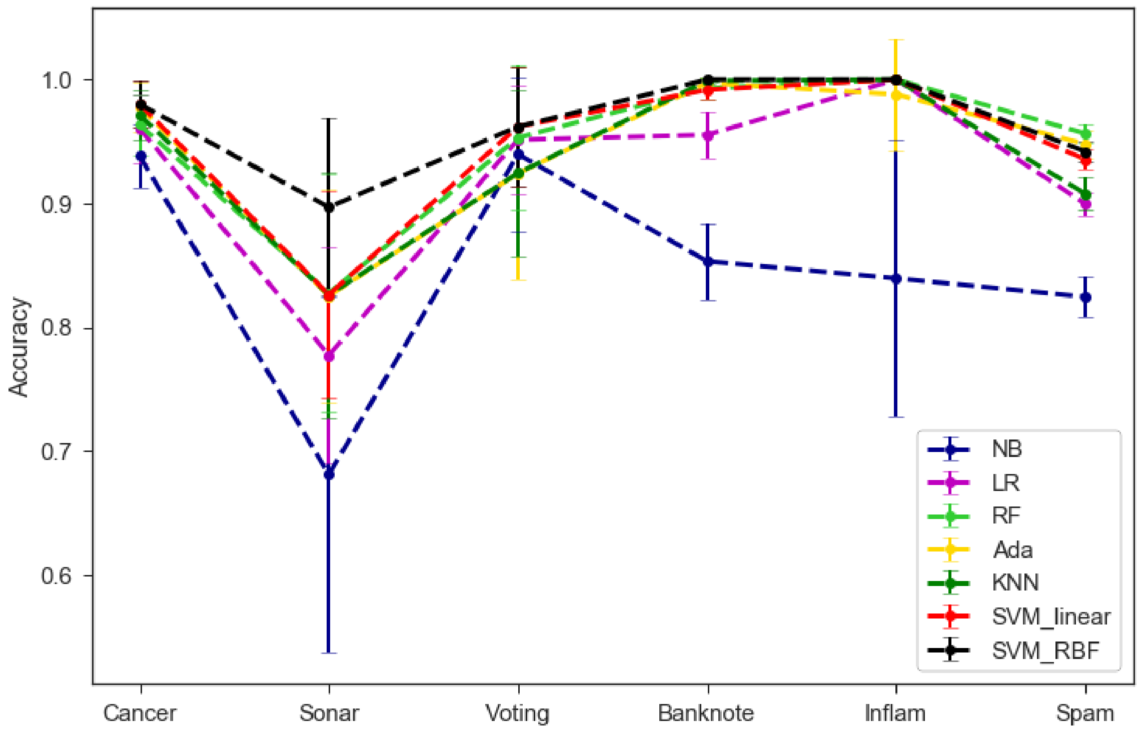


Figure 5.17: Mean validation accuracy for the binary classification problems, error bars showing the standard deviation of 10-fold cross validation with 10 repetitions for statistical confidence.

The box plots not only show the median validation accuracy but also shows how confident the classifier is of this value through the length of the box, the Inter-Quartile Range (IQR). Table 5.2 shows the robustness of linear SVM and random forests since they had the narrowest IQR in 4 out of the 6 data sets whereas RBF-kernel SVM comes right after them.

| | **NB** | **LR** | **RF** | **Ada** | **KNN** | **SVM_L** | **SVM_RBF** |
|---|---|---|---|---|---|---|---|
| **Cancer** | 0.022 | 0.045 | 0.022 | 0.022 | 0.023 | 0.043 | 0.043 |
| **Sonar** | 0.224 | 0.169 | 0.130 | 0.118 | 0.110 | 0.070 | 0.118 |
| **Voting** | 0.053 | 0.053 | 0.053 | 0.056 | 0.105 | 0.053 | 0.053 |
| **Banknote** | 0.045 | 0.036 | 0.009 | 0.009 | 0.000 | 0.000 | 0.000 |
| **Inflam** | 0.100 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **Spambase** | 0.024 | 0.019 | 0.012 | 0.014 | 0.016 | 0.016 | 0.014 |

Table 5.2: Binary classification inter-quartile range, color scale displaying red as the least confident and green as the most confident for each data set.

## 5.3 Classifier Comparisons in Multi-class Classification Data Sets

In multi-class classification problems, we have 3 data sets, one of them has the number of features is much larger than the number of instances, $p \gg n$.

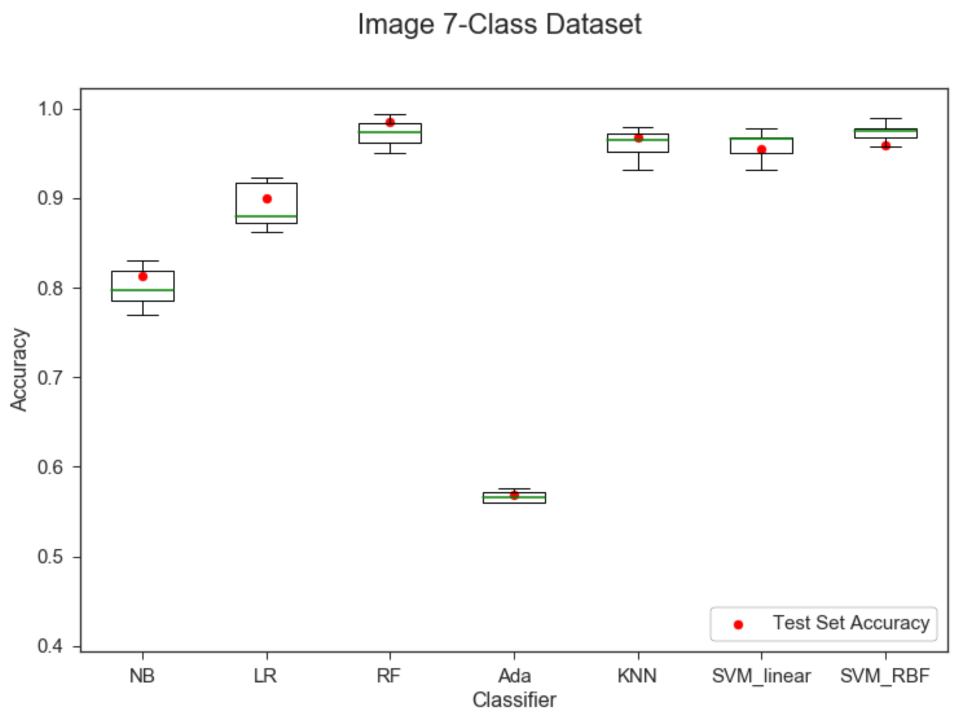


Figure 5.18: Image data set summary of accuracy results showing box plots for the cross validation accuracy (median: green bar) and test set accuracy (red dots).

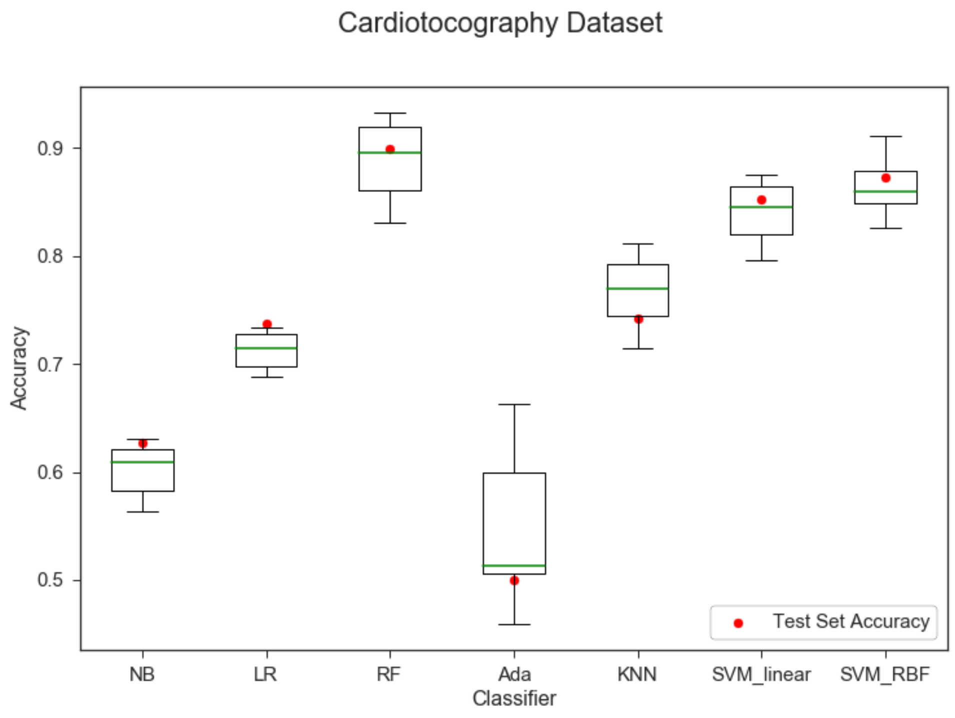


Figure 5.19: Cardiotocography data set summary of accuracy results showing box plots for the cross validation accuracy (median: green bar) and test set accuracy (red dots).

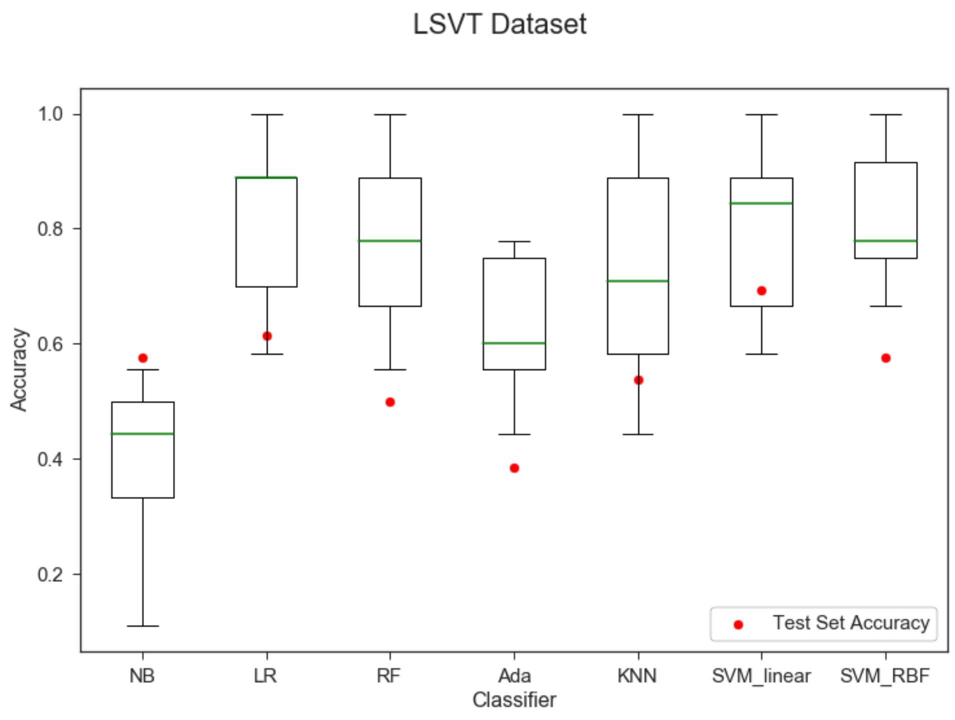


Figure 5.20: LSVT summary of accuracy results showing box plots for the cross validation accuracy (median: green bar) and test set accuracy (red dots).

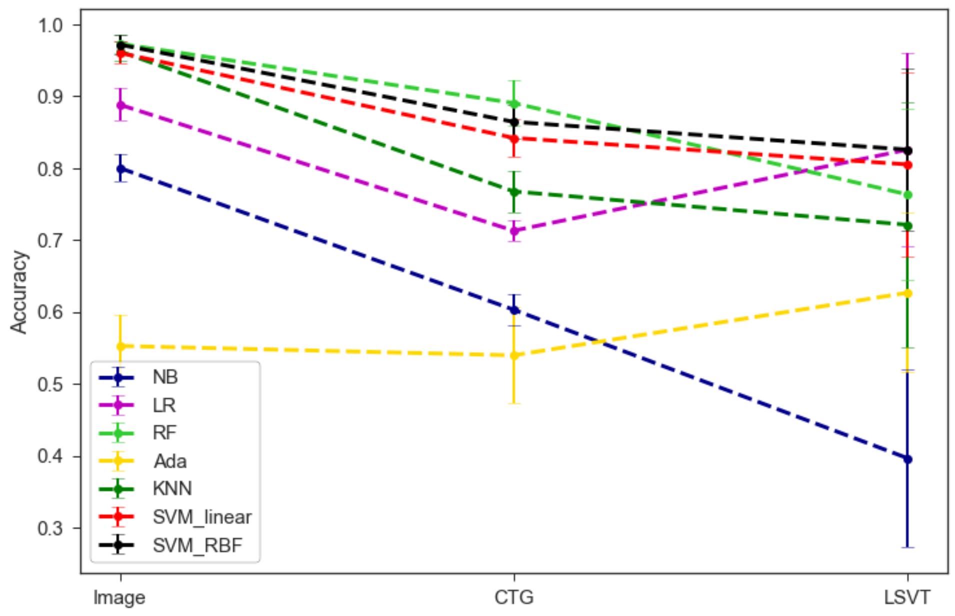


Figure 5.21: Mean validation accuracy for the multi-class classification problems, error bars showing the standard deviation of 10 repeated 10-fold cross validation.

Figure 5.21 and the summary in Figures 5.18-5.20 show that random forests classifier competes with linear SVM and RBF-kernel SVM and outperforms them in 2 out of 3 data sets. Logistic regression was surprisingly the best in the LSVT data set. RBF-kernel SVM showed confidence in its accuracy by having the narrowest IQR. The mean validation accuracy results are also summarized in Table 5.3 and IQR in Table 5.4.

| | **NB** | **LR** | **RF** | **Ada** | **KNN** | **SVM_L** | **SVM_RBF** |
|---|---|---|---|---|---|---|---|
| **Image** | 0.797 | 0.885 | 0.981 | 0.554 | 0.970 | 0.965 | 0.950 |
| **Cardio** | 0.575 | 0.710 | 0.890 | 0.512 | 0.749 | 0.827 | 0.853 |
| **LSVT** | 0.495 | 0.779 | 0.688 | 0.566 | 0.668 | 0.729 | 0.738 |

Table 5.3: Multi-class classification mean validation accuracy, color scale for each data set displaying red as the lowest and green as the highest accuracy.

| | **NB** | **LR** | **RF** | **Ada** | **KNN** | **SVM_L** | **SVM_RBF** |
|---|---|---|---|---|---|---|---|
| **Image** | 0.033 | 0.045 | 0.022 | 0.011 | 0.020 | 0.018 | 0.010 |
| **Cardio** | 0.038 | 0.031 | 0.059 | 0.094 | 0.048 | 0.044 | 0.030 |
| **LSVT** | 0.139 | 0.233 | 0.167 | 0.222 | 0.389 | 0.333 | 0.167 |

Table 5.4: Multi-class classification inter-quartile range, color scale displaying red as the least confident and green as the most confident for each data set.

# 6 Discussion and Conclusions

In this chapter, we discuss our key results, rules of thumb in classification, limitation of our study and the potential future work.

## 6.1 Summary of Key Results

It has been clearly shown that the assumptions of naive Bayes put the classifier back when compared to other classifiers, specially in the binary classification setting. However, it outperformed $K$NN and AdaBoost in the voting data set whose features were only binary categorical. Although it was the least performing in the cancer diagnostics data set, it scored more than 94% which can be enough in some cases given its fast implementation. In multi-class data sets, the only classifier that performed worse than naive Bayes was AdaBoost. The low performance of Adaboost is justified since we do not optimize its hyperparameters. Random forests outperformed the other classifiers in 2 out of the 3 multi-class data sets. In the third data set, LSVT, logistic regression and RBF-kernel SVM and linear SVM had the highest accuracy respectively. LSVT data set is different in nature than the other two data sets since it has very small number of instances ($n = 100$) where the number of features is very large ($p = 310$).

SVM, both linear and RBF-kernel, and random forests are the most robust and provide a narrow confidence interval relative to the other classifiers on binary classification data sets. However, only RBF-kernel SVM managed to keep this robustness in the multi-class setting. Searching for the optimized hyperparameters for SVMs and KNN before training the model make those classifiers computationally expensive. The exact time is reported in the notebook on Github repository.

## 6.2 Rules of Thumb

Support vector machines and random forests are, on the basis of performed comparisons, the first learners to try on binary and multi-class classification problems. When using naive Bayes, it is advisable to use probability distributions that best describe each feature in the data. However, the independence of features assumption will always limit the performance.

Support Vector Machines and K-Nearest Neighbors need optimization first and this should be considered before application if time is of concern. AdaBoost also need hyperparamter optimization as well, particularly in the multi-class classification settings.

Generally, the classifier is more confident when more data is available and that was clearly observed in the IQR results of the banknote and spam base data sets. This general rule of thumb can be violated when the data set characteristics are

## 6.3 Limitations

Since we use real-world data sets, the study is limited to the open source data sets with its limited choice of size, data types and number of features. For stronger statements, a wide range of data size $n$, number of features $p$ and data types should be included. The hardware used in the experiments has limited the study to small and medium data sets. Due to time limitations, only 9 data sets have been analyzed and the missing data were just deleted.

Fitting time is one of the important metrics to consider in comparison but the reported time in the Github notebooks is not reliable to compare since the laptop used in the experiments was being used for other tasks at the same time. This would give inaccurate results for the time needed for each classifier.

## 6.4 Future Work

For further analysis, more popular classifiers should be included in the study such as *Gradient Boosting Machines* (GBM). Widening the range of data sets in terms of size, number of features and data types is crucial to strengthen the conclusions about the classifiers. In this study, we did not include categorical features of more than two categories. Future work will take this in consideration and pick data sets with differently distributed features. For faster implementation and for tracking accurate modeling time, a dedicated powerful machine shall be used. The experiments will also be repeated for statistical confidence about time results. We also advise to add to the binary and multi-class settings the *micro-array* settings due its complexity. The micro-array type of data is where the number of features are much larger than the number of instances ($p \gg n$), such as the LSVT data. Those data sets are complex in nature and need to be carefully and separately studied.

## References

[1] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer series in statistics, Springer, 2009.

[2] D. R. Amancio, C. H. Comin, D. Casanova, G. Travieso, O. M. Bruno, F. A. Rodrigues, and L. Da Fontoura Costa, "A systematic comparison of supervised classifiers," *PLoS ONE*, vol. 9, no. 4, 2014.

[3] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition (The Morgan Kaufmann Series in Data Management Systems).* 2011.

[4] R. Entezari-Maleki, A. Rezaei, and B. Minaei-Bidgoli, "Comparison of Classification Methods Based on the Type of Attributes and Sample Size," *Journal of Convergence Information Technology*, vol. 4, no. 3, pp. 94–102, 2009.

[5] L. Tjen-Sien, L. Wei-Yin, and Y.-S. Shih, "A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms," *Machine Learning*, vol. 229, pp. 203–229, 1992.

[6] B. Carnahan, G. Meyer, and L.-A. Kuntz, "Comparing Statistical and Machine Learning Classifiers: Alternatives for Predictive Modeling in Human Factors Research," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 45, no. 3, pp. 408–423, 2004.

[7] A. M. Fred Agarap, "On Breast Cancer Detection: An Application of Machine Learning Algorithms on the Wisconsin Diagnostic Dataset," *CoRR*, 2017.

[8] V. Chaurasia, S. Pal, and B. B. Tiwari, "Prediction of benign and malignant breast cancer using data mining techniques," *Journal of Algorithms and Computational Technology*, vol. 12, pp. 119–126, jun 2018.

[9] A. Mert, N. Kiliç, E. Bilgili, and A. Akan, "Breast cancer detection with reduced feature set," *Computational and Mathematical Methods in Medicine*, vol. 2015, 2015.

[10] E. Zafiropoulos, I. Maglogiannis, and I. Anagnostopoulos, "A support vector machine approach to breast cancer diagnosis and prognosis," *IFIP International Federation for Information Processing*, vol. 204, pp. 500–507, 2006.

[11] R. D. Nindrea, T. Aryandono, L. Lazuardi, and I. Dwiprahasto, "Diagnostic Accuracy of Different Machine Learning Algorithms for Breast Cancer Risk Calculation: a Meta-Analysis," *Asian Pacific journal of cancer prevention : APJCP*, vol. 19, pp. 1747–1752, jul 2018.

[12] H. Zhang, "The Optimality of Naive Bayes," in *Proceedings of the Seventeenth International Florida Artificial Intelligence Research*, (Florida), 2004.

[13] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[14] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R.* Springer Publishing Company, Incorporated, 2014.

[15] T. M. Mitchell, *Machine learning.* McGraw-Hill Science/Engineering/Math; (March 1, 1997), 1997.

[16] L. Breiman, "Random Forests," *Machine learning*, pp. 5–32, 2001.

[17] L. Breiman, "Bagging Predictors," *Machine learning*, vol. 140, pp. 123–140, 1996.

[18] T. K. Ho, “The Random Subspace Method for Constructing Decision Forests,” *IEEE Transactions On Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, 1998.

[19] A. Gron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* O’Reilly Media, Inc., 1st ed., 2017.

[20] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, “The NumPy array: A structure for efficient numerical computation,” *Computing in Science and Engineering*, vol. 13, pp. 22–30, mar 2011.

[21] W. McKinney *et al.*, “Data structures for statistical computing in python,” in *Proceedings of the 9th Python in Science Conference*, vol. 445, pp. 51–56, Austin, TX, 2010.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Computing in Science and Engineering*, vol. 9, pp. 99–104, may 2007.

[24] M. Waskom, O. Botvinnik, D. O’Kane, P. Hobson, J. Ostblom, S. Lukauskas, D. C. Gemperline, T. Augspurger, Y. Halchenko, J. B. Cole, J. Warmenhoven, J. de Ruiter, C. Pye, S. Hoyer, J. Vanderplas, S. Villalba, G. Kunter, E. Quintero, P. Bachant, M. Martin, K. Meyer, A. Miles, Y. Ram, T. Brunner, T. Yarkoni, M. L. Williams, C. Evans, C. Fitzgerald, Brian, and A. Qalieh, “mwaskom/seaborn: v0.9.0 (july 2018),” July 2018.

[25] S. Raschka, *Python Machine Learning.* Packt Publishing, 2015.

[26] A. N. Baraldi and C. K. Enders, “An introduction to modern missing data analyses,” *Journal of School Psychology*, vol. 48, no. 1, pp. 5–37, 2010.

[27] J. Grus, *Data Science from Scratch: First Principles with Python.* O’Reilly Media, Inc., 1st ed., 2015.

[28] S. Aksoy and R. M. Haralick, “Feature normalization and likelihood-based similarity measures for image retrieval,” *Pattern Recognition Letters*, vol. 22, pp. 563–582, apr 2001.

[29] C.-C. Chang and C.-J. Lin, “LIBSVM: A Library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–39, 2011.

[30] C.-W. C. Hsu, C.-C. Chang, and C.-J. Lin, “A practical guide to support vector classication,” 2008.

[31] D. Dua and C. Graff, “UCI machine learning repository.” University of California, Irvine, School of Information and Computer Sciences, 2017. `http://archive.ics.uci.edu/ml`.

[32] A. Tsanas, M. A. Little, C. Fox, and L. O. Ramig, “Objective Automatic Assessment of Rehabilitative Speech Treatment in Parkinson’s Disease,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, pp. 181–190, jan 2014.