# Statistical Analysis of Classification Algorithms for Predicting Socioeconomics Status of Twitter Users

by

Ying Zhou

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Science

in

Probability and Statistics

Carleton University
Ottawa, Ontario

## Abstract

The purpose of this study is to compare a series of well-known statistical machine learning techniques that classify online social network (OSN) Twitter users based on their socioeconomic status (upper/middle/lower). These approaches are of difference owing to their assumptions, strengths, and weaknesses. In the experiments, five (5) classification algorithms are employed for the classification task. Logistic Regression, Support Vector Machine (SVM), Naïve Bayes (NB), k-Nearest Neighbors, and Decision Tree are applied on high-dimensional data set extracted from the users' platform-based and profile-based behavior on Twitter. These algorithms are theoretically investigated and experimentally evaluated in terms of four (4) performance measures: accuracy, precision, recall, and AUC. Then, ensemble methods i.e. Bagging and Boosting are employed to improve the performance of the aforementioned classifiers. Multivariate analysis of variance is employed to examine if performance measures of these algorithms are significantly different. And univariate analysis of variance is used to analyze the differences of our classification methods for each performance measure. The analyses indicate a significant difference among these algorithms; both SVM and NB achieve good performance on our high-dimensional OSN data set.

## Acknowledgements

I would like to express my sincere gratitude and appreciation to my supervisor Dr. Shirley E. Mills for believing in me, her invaluable guidance, patience, encouragement and support in all aspects of my study. I am lucky to have her as my supervisor and have greatly enjoyed working under her guidance.

I would like to thank all my friends and classmates who have supported and encouraged me throughout my study, particularly Lu Zhu, Shuang and Mengjie.

And I would like to thank all the faculty members and staff for their support and guidance throughout my years at Carleton University.

Finally, I would like to thank my parents, for their patience, immense support, and for taking care of me throughout my study. They gave me the strength and love that I needed to move forward.

# Table of Contents

## List of Tables

## List of Figures

# 1 Introduction

Statistical machine learning is a well-established field for data analysis. Machine learning techniques have been intensively used to analyze data from a broad spectrum of applications, such as face detection, sentiment classification, and recommendation systems. To classify the socioeconomic status (SES) of Twitter users efficiently, a series of classification algorithms are applied and their ability of classification is assessed in this thesis.

## 1.1 Background and motivation

Inferring information expressed in online social networks (OSNs) e.g. Twitter, Facebook, Myspace plays an important role in sociology, marketing, and epidemiology, and this information can be used in both predictive and descriptive problems. For example, we can improve the performance of recommendation services provided by OSNs (e.g., LinkedIn and Facebook) by estimating the relationship strength for pairs of users within a certain range (e.g., working in the same company, sharing the same hobbies, etc.). OSNs can automatically suggest new connections to users based on the relationship strength. This information can be also used to build an online member's personalized newsfeed, i.e., real-time updates about status change, activities, new posts or other stories from contacts. When building the personalized newsfeed, prioritizing updates could be more

beneficial to the user by removing or downplaying updates from acquaintance contacts rather than from friends. In addition, we can rank the search results for a certain user based on the search history of related users; this is because one's preference is more likely to coincide with those who are strongly related. Apart from the relationship strength estimation, analysis of users' impact and credibility based on online data has drawn increasing attention due to the open structure of OSNs.

This thesis aims at inferring the SES of social media users, based on a vast amount of data extracted from the platform of Twitter and users' profiles. For the period, the users' data are described by platform-based behavior (e.g. number of tweets of a user, the number of replies, mentions or retweets), platform impact, topics of interest, interactive behavior, and keywords in user's profile description.

Twitter is a large, with rising popularity, user-based microblogging platform, collecting tons of information among users all over the world. Millions of Twitter users connect and interact with their families, friends, and colleagues via posting short messages called tweets that contains a limit of 140 characters. Unlike Facebook and LinkedIn, users follow (followers) other users or are followed (being followees) un-reciprocally, which means a user can follow any user without being followed back and one can receive all the tweets from their followers, thus information is diffused rather quickly on Twitter. A retweet can be abbreviated as "RT." The RT is the message (tweet) that is reposted or forwarded with attribution to the user who first posted it. More concretely, '#' that is followed by a word stands for a hashtag (a phrase or keywords used to describe a topic),

and '@' followed by a user identifier represents a user, like "mention". Retweeting is an easy and quick way for dissemination of information. And the posting which has been retweeted thousands of times and accompanied with a proper hashtag can be seen as a trending topic. Real-world outcomes (e.g. What are the most popular topics? Which user is most influential? Which area are they most interested in?) could be predicted through analyzing the user profile information, social relations, trending topics and millions of tweets and retweets (Kwak et al., 2010; Weng et al., 2010).

The contributions of this thesis are three-fold:

1) Measuring the performances of different classification methods;
2) Balancing the trade-off between the computational complexity and the performance of models;
3) Statistical comparison of methods using variance analysis.

## 1.2 Structure of the thesis

The remainder of this thesis is organized as follows. Chapter 2 presents a literature review that includes the related work and an introduction to the field of machine learning and classification. In Chapter 3, we give the data description and discuss the methodologies including single and ensemble classification methods in detail. The experimental settings, results and evaluation of the algorithms appear in Chapter 4. Finally, Chapter 5 concludes the work and points out possible directions for future work.

# 2 Literature Review

## 2.1 Related work

Previous studies have analyzed social media from various aspects. Szabo and Huberman (2008) investigated the temporal patterns of news popularity. They reported a method to predict the long-term popularity of online social content by modeling the amassing content of views and votes gathered from YouTube and Digg. Their study indicated a statistical correlation between the prediction accuracy and updated speed of news. Benevenuto et al. (2009) studied the behaviors of OSNs users based on clickstream data from four (4) popular social websites: Orkut, MySpace, Hi5 and LinkedIn. By grouping all user activities into nine (9) categories: Search, Scrapbook, Messages, Testimonials, Videos, Photos, Profile & Friends, Communities, and Other, the clickstream model was developed to investigate the frequency, type, and relative sequence of the activities and to compute the probability of transferring from one category to another. A topological method was also applied to understand the pattern of interactions. Schneider et al. (2009) also developed a cross-system analysis of the features and their influence within Facebook, Hi5, LinkedIn, and StudyVZ. They presented an efficient customizable approach for extracting users' information and identifying OSN sessions and user behaviors. Meo et al. (2013) conducted a comparative study of users' behavior in three different social platforms: Flickr, Delicious, and StumbleUpon. The study examined if and how the features of a given system reflected users' behaviors and also the correlation

between tagging and social networking behavior of individual users. Jiang et al. (2013) highlighted the latent interactions in OSNs and the structures among online social activities. By using detailed measurement and statistics over a period of 90 days, the problems of user popularity, the mutual effect of visits and the influence of content updates upon user popularity were studied. They found that latent interactions were the vast majority of users' activities.

There exists some research focusing on Twitter data. For example, by analyzing the data extracted from Twitter, Kwak et al. (2010) found a non-power-law following distributions between followees and followers. When studying the influential on Twitter, they found a similarity between ranking individuals by the number of followers and by the PageRank algorithm; but these two rankings were different from ranking by retweets. They also studied the temporal behavior and individual's participation based on the tweets of trending topics. Asur and Huberman (2010) examined how social media could be used to predict future outcomes. They forecasted the box-office revenues for movies by analyzing the content of tweets and presented sentiment analysis to improve the power of predictions. Lampos et al. (2014) presented a study of user impact prediction on Twitter, based on non-textual and textual attributes under users' direct control. Impact score was defined for this problem which is formulated as a regression task. Linear and non-linear learning methods were applied and the model based on Gaussian Processes performed well, indicating that it is more suitable than linear methods for this problem. Another study formulated a classification task by using a variety of latent feature representations to identify the occupational class for an OSN user profile (Preoţiuc-Pietro

et al., 2015). It was conducted on an extensively labeled data set extracted from Twitter users, including platform features, the textual content of tweets, and their job titles. The experiments illustrated that both linear and non-linear learning algorithms could predict well a user's most likely job class as one of nine major groups of the Standard Occupation Classification (SOC), and the non-linear methods showed strong accuracy. Through a qualitative assessment, the textual attributes, especially the word clusters that capture both interests and behaviors of occupation classes indicated good prediction performance.

## 2.2 Machine Learning

All scientific fields aim at explaining available experimental evidence well in order to generalize it and make inferences. Statistical machine learning, as a scientific field, tries to reveal the underlying structure of data by developing models. Thus, machine learning helps us understand the nature of our data and make predictions for the future data. Machine learning is a subfield of the broader topic Artificial Intelligence, since it studies how to automatically learn to make predictions relying on past observations.

In machine learning, the data we use is essentially a collection of data instances. A data instance can be in any form or shape depending on the field of study. For example, it can be a video, text or a picture. Supervised machine learning considers data instances with their labels. In this case, data instances are pairs consisting of an input object and its associated labels. The task in supervised machine learning is to predict the labels of

future data based on existing labeled data. This is generally handled by inferring a function from the labeled data. Depending on the type of the label to be inferred, the task is usually divided further into two sub-tasks: regression and classification. In regression, the given labels are continuous variables. For example, predicting the temperature for the upcoming days is a regression task, since the temperature can take any value. In contrast to regression, in classification, the labels are a finite set of categories, i.e. classes. Thus, the goal of classification is to assign an input object to one of the classes. In this thesis, we are concerned with the classification task.

## 2.3 Classification

Classification has a broad range of applications. Diagnosing a tumor as benign or malignant by making use of X-ray mammographies that have been previously given is a classification task. Here the classes are benign and malignant. Or, consider the optical character recognition (OCR) problem. The task in OCR is to recognize the images of handwritten letters and then assign them to one of twenty-four classes. Prediction of an email as spam or ham, face detection in computer vision, fraud detection in credit card transactions and classification of microarray data in bioinformatics are well-known examples of the classification task.

Formally, the task of classification can be described as follows. Let an instance be an object that is represented by a set of features (characteristics) and a class label (categories). A learning algorithm (classifier) is essentially a mapping function from the

features to the set of class labels. In the context of supervised machine learning, the task of classification assumes the presence of a data set that consists of labeled instances. A classifier is built on the labeled data such that it automatically predicts classes and can be used for future instances. The task of classification is to search for a classifier from all possible mapping functions, subject to the constraints imposed by the labeled data set.

# 3 Methodology

To classify the SES of OSN users, we apply some methods (like NB and KNN) that are low in complexity and easy to interpret. We also try some more sophisticated or composite methods (like SVM, ensemble methods) to increase performance. The algorithms used in this thesis are:

- Logistic Regression (via Generalized Linear Model or GLM)
- Support Vector Machines
- Naïve Bayes
- K-Nearest Neighbors
- Trees and Rules: CART and C4.5
- Ensemble methods: Bagging and Boosting

To assess performance of the algorithms, four (4) performance measures (accuracy, precision, recall, and AUC) will be used, and computing time as well. Additional comparison will be run by MANOVA.

## 3.1 Data description

In this section, we give a detailed description of the data set used for this study. The data for this thesis was extracted and inferred by Lampos et al. (2015) from a set of 1,342

profiles of Twitter users and their corresponding tweets, denoted by $\mathcal{D}_1$ (2,082,651 tweets in total) for the period between February 1, 2014 to March 21, 2015. These users were selected from a pool of 100,000 Twitter users during this period of time who set their location as UK in profile descriptions. They only considered those users who have an occupation according to the Standard Occupational Classification (SOC) (Elias, Birch et al., 2010) hierarchy which provides a mapping from SOC to SES.

Lampos et al. converted this data (user profiles and corresponding tweets) to a data set with 1291-dimensional feature vectors grouped into five (5) user feature categories as follows:

*C1:* The frequent 1-gram vocabulary index presented in users' tweets, representing dimensions 1 to 560 in data_matrix.csv. As Lampos et al. (2016) defined, the1-gram ($x$) frequency for user $i$ is $f_i = \frac{|x_i|}{N_i}$, where $|x_i|$ denotes the number of appearances of $x$ in the tweets of user $i$ and $N_i$ represents the total number of tweets for $i$.

*C2:* The key words in a user's profile (1-grams and 2-grams) represent dimensions 561 to 786 and 787 to 1083 in data_matrix.csv respectively.

*C3:* The frequency distribution of 200 topics that is derived from the tweets associated with *C1* which are represented by 1-gram clusters corresponds to dimensions 1084 to

1283. The frequency of a topic ($\tau$) of a user $i$ is defined as $\tau_i = \Sigma_{f_i \in \tau} f_i$, where $f_i \in \tau$ denotes the frequency defined in *C1* which belongs the topic $\tau$ clusters.

*C4:* The platform-based behavior, from dimensions 1284 to 1287, represent the ratios, over the total number of tweets of a user $i$, of replies, mentions, retweets and unique mentions of other accounts.

*C5:* The platform impact, denoted by dimensions 1288 to 1291, expresses the number of accounts followers, followees, listing time and the impact score for each user, which are represented by the log-number of followers+1, followees+1, listing+1 and the impact score, respectively.

A SES class label for each user corresponds to each line of $\mathcal{D}_1$; there are 710 upper, 318 middle and 314 lower SES classes, mapped to users by utilising the SOC (Elias, Birch et al., 2010; Preoţiuc-Pietro et al., 2015) where jobs are hierarchically organized on the basis of required skills and job content. The SOC taxonomy consists of nine 1-digit occupational groups at the top level and each group is divided into sub-groups, breaking down to 369 4-digit groups (the fourth level).

Additionally, 159,101,560 UK tweets that were posted in the same period denoted as $\mathcal{D}_2$ were randomly sampled to compile the set of latent topics among Twitter users. To obtain 200 clusters of 1-grams vocabularies which consisted of latent topics and expressions of linguistics, Lampos et al. applied spectral clustering (Von Luxburg, 2007) on these

sampled tweets. Then they computed the frequency of each topic in the tweets selected in $\mathcal{D}_1$ presenting in category *C3* above.

**Impact score**

On Twitter, it is known that users with a large number of followers such as popular politicians, artists, and brands have a substantial impact in the real world. For any user's account, the number of followers (which represent other users interested in this account) could simply assess an account's popularity.

Nevertheless, non-popular accounts can still gain many followers via exploitation. For instance, users who follow other accounts with a purpose of expecting to be followed back. Consequently, user impact should not be quantified only by the number of followers (Cha et al., 2010). Moreover, the alternative $\Phi_{in}/\Phi_{out}$ , the ratio of followers to followees, is not a reliable measurement because it is invariable to scale. Lampos at el. (2014) defined the impact score (S) as:

$$S(\Phi_{in}, \Phi_{out}, \Phi_{\lambda}) = \ln(\frac{(\Phi_{\lambda} + 1)(\Phi_{in} + 1)^2}{\Phi_{out} + 1})$$

where $\Phi_{\lambda} \geq 0$ is an added indicator denoting the number of times that a user has been listed by others. Note that $\Phi_{in} \geq 0$ is the raw number of followers, $\Phi_{out} \geq 0$ denotes of the number of followees.

## 3.2 Single classifiers

### 3.2.1 Logistic Regression

Logistic regression was first proposed by Sir David Cox in 1958. It is somewhat similar to linear regression in that it aims to determine a good classifier by determining the value of coefficients that represent the weights for each input variable. Unlike linear regression, the output is a probability which is a non-linear function of the input variables. It is a binary classification algorithm.

Before getting into the details of logistic regression, we first introduce the following notation that will be used later. Let $X$ denote a k-dimensional feature space; $(X_1, X_2, \dots, X_k)$ be a set of explanatory variables. Let $Y$ be a class set consisting of two possible class labels $\{0, 1\}$. A training data set $T$ is a set of instances $(x_i, y_i)$ generated from the labeled space $(X, Y)$ under the independent, identically distributed (i.i.d.) assumption. Given the training data *T*, the goal of a binary classifier is to find a probability function $h$ that can predict classes accurately for future instances. A function is essentially a mapping function from the feature space *X* to the class set *Y*, where

$$Y = \begin{cases} 0 & \text{if in class 1} \\ 1 & \text{if in class 2} \end{cases}.$$

In logistic regression, the probability function of $Y = 1$ is defined as follows:

$$h(x;\theta) = g(\theta^{T}x) = \frac{1}{1 + e^{-\theta\hat{}T\,x}}$$

where θ is a vector of parameters that parameterizes the function space mapping from *X* to *Y* and $g(z) = \frac{1}{1+e^{-z}}$ is called the logistic function or the sigmoid function.

The task of logistic regression is to find a vector θ such that $h(x;\theta)$ is modelling the probability distribution of *Y*, at least for the training data *T*. To find θ, we can endow the model with a set of probabilistic assumptions and then estimate the parameter θ via maximum likelihood. More precisely, assume that the class labels and input features are related via the following function:

$$\begin{cases} P(Y=1) = p \\ P(Y=0) = q \end{cases}, \quad \text{where } p = h(x), q = 1 - p$$

The likelihood function is defined as:

$$L(\theta) = \prod_{i=1}^{m} p(y_i|x_i;\theta)$$

where $p(y_i|x_i;\theta)$ is the probability of $y_i$ given $x_i$ parameterized by θ. In this context, we should choose θ to maximize $L(\theta)$, making the data occur with as high probability as possible.

To conduct the logistic regression in this thesis, the *glm* function (generalized linear models) will be used with the argument quasibinomial (link = "logit").

### 3.2.2 SVM

Support Vector Machines (SVM), first introduced by Vapnik in the 1960s for classification, provides one of the most accurate and robust learning methods. It can be applied to data sets with varying numbers of dimensions. The efficiency of SVM models in both technique and theory is being steadily improved (Wu et al., 2008, Burbidge & Buxton, 2001). The goal of SVM is to find the "best" classification function to separate the members of different classes in the training data. Take linear separation of binary classes as an example. A linear classification function corresponding to a separating hyperplane separates the two classes (positive class $P$ and negative class $N$ for $y_i = +1, -1$ respectively) and is denoted by:

$$\mathbf{w}^T\mathbf{x}_i + b \leq -1, \text{for all } \mathbf{x}_i \in N$$

$$\mathbf{w}^T\mathbf{x}_i + b \geq +1, \text{for all } \mathbf{x}_i \in P$$

In a pair $(\boldsymbol{w}, b)$, $\boldsymbol{w}$ is the weight vector and $b$ is a constant. Hence the decision rule is given by the sign of the function: $f_{\boldsymbol{w},b}(\mathbf{x}_i) = \text{sgn}(\mathbf{w}^T\mathbf{x}_i + b)$.

Obtaining the optimal functions requires finding maximum margin hyperplanes between different classes. This gives the best generalization capability and ensures both classification performance on training data and space for correcting classification of the future data (Wu et al., 2008). An SVM classifier tries to maximize the following Lagrangian function:

$$L(\mathbf{w}, \mathrm{b}, \Lambda) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{l} \lambda_i y_i (\boldsymbol{w}^T \mathbf{x}_i + b) + \sum_{i=1}^{l} \lambda_i$$

where $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_l)^T$ are Lagrange multipliers, $l$ is the number of training examples, and the pair $(\boldsymbol{w}, b)$ defines a hyperplane.

By differentiating with respect to $\boldsymbol{w}$ and $b$, and setting the derivatives equal to 0, we obtain:

$$\frac{\partial L(\boldsymbol{w}, b, \Lambda)}{\partial \boldsymbol{w}} = w - \sum_{i=1}^{l} \lambda_i y_i \mathbf{x}_i = 0, \qquad \frac{\partial L(\boldsymbol{w}, b, \Lambda)}{\partial b} = -\sum_{i=1}^{l} \lambda_i y_i = 0.$$

Hence, the optimal function is given by:

$$f_{\boldsymbol{w}^*, b^*}(\mathbf{x}_i) = \mathrm{sgn}(\boldsymbol{w}^{*T} \mathbf{x}_i + b^*)$$

where $\boldsymbol{w}^* = \sum_{i=1}^{l} \lambda_i{}^* y_i \mathbf{x}_i$ and $b^* = y_i - \boldsymbol{w}^{*T} \mathbf{x}_i$ for any support vector $\mathbf{x}_i$.

To extend the SVM classifier from binary tasks to multiclass classification, the 'one-against-all'-approach that means repeatedly setting one class as a positive class and the rest grouped as the negative class can be used. Thus, for k category tasks (k>2), k(k-1)/2 binary classifiers are trained and the appropriate class can be found by a voting scheme.

#### 3.2.2.1 Kernel functions

Kernel functions are used to handle a variety of non-separable cases by mapping the input data into a higher-dimensional feature space. Then the new mapping is linearly-separated

(Rätsch, 2004; Scholkopf & Smola, 2001). The kernel $K(x, y) = \langle \Phi(x), \Phi(y) \rangle$ gives the definition of the mapping which is achieved by replacing the inner product (x, y). Much recent research (Amari & Wu, 1999; Afifi, 2013) has focused on using different kernels for SVM classifications. Two of the most popular kernel functions are:

- Polynomial Kernel: $\text{Kernel}(X, X') = (\xi + \gamma X^T X')^d$
- Radial Basis Function, RBF: $\text{Kernel}(X, X') = \exp(-\frac{\|X - X'\|^2}{2\sigma^2})$

The polynomial kernel allows for separating curves in the input space. When the degree of the polynomial equals to 1, it is the same as the linear kernel. For the radial basis function kernel,

$$\gamma = \frac{1}{2\sigma^2} \in (0,1)$$

which must be specified to the learning algorithm. This allows for a more complex separating region.

#### 3.2.2.2 PRBF

Notice that the classical kernels like Polynomial or Gauss functions each have good performance for some kinds of data sets. The new proposed kernel function PRBF (Afifi, 2013) that combines both Polynomial and Gauss functions can handle a high dimensional data set and perform well for nearly all data sets. The definition is given by:

$$\text{PRBF} = \left( \frac{(1 + \exp(-\text{sum}(x_{1i} - x_{2i})^2)}{PD} \right)^d$$

where *P* denotes the kernel parameter, *D* is the number of attributes (the dimension of the input data), and *d* is the polynomial degree as shown before.

By computing the limit of PRBF/RBF, their study shows that PRBF converges faster than RBF as follows:

$$\lim_{x_{1i},x_{2i}\to\infty}\left(\frac{PRBF}{RBF}\right)=\lim_{x_{1i},x_{2i}\to\infty}\left(\frac{\left(\frac{\left(1+\exp\left(-\mathrm{sum}\left(x_{1i}-x_{2i}\right)^2\right)\right)}{PD}\right)^d}{\frac{\exp\left(-\mathrm{sum}\left(x_{1i}-x_{2i}\right)^2\right)}{PD}}\right)=\lim_{x_{1i},x_{2i}\to\infty}\left(\frac{\left(\frac{1+e^{-T}}{V}\right)^d}{\frac{e^{-T}}{V}}\right)=\infty,$$

where $\mathrm{T}=\mathrm{sum}(x_{1i}-x_{2i})^2$, $\mathrm{V}=\mathrm{PD}$.

### 3.2.3 Naïve Bayes

Naïve Bayes (NB) algorithm is one of the oldest formal classification methods, with the simplest form and surprising efficiency (Wu et al., 2008). This classification algorithm is of the probabilistic type and is based on Bayes' theorem. For each class, the posterior probability is computed from training data and predicted class is the one having the highest posterior probability. NB computes the posterior $P(Y|X)$, based on estimating the likelihood $P(X|Y)$ and prior $P(Y)$, where $X =< X_1, \dots, X_k >, X_i, i \in \{1, \dots, k\}$ refers to the $i$th feature and $Y$ is the class variable. When computing $P(Y|X)$, it is assumed that each feature is mutually conditionally independent given $Y$ so as to make the estimation of $P(X|Y)$ easier. According to this assumption, $P(X|Y)$ can be readily estimated as:

$$P(X|Y)=P(X_1,\dots,X_k|Y)=\prod_{i=1}^{k}P(X_i|Y)$$

Using this, NB outputs the posterior probabilities for each class in $Y$, given by:

$$P(Y = y_l|X_1, \dots, X_k) = \frac{P(Y = y_l)\prod_{i=1}^{k} P(X_i|Y = y_l)}{\sum_j P(Y = y_j)\prod_{i=1}^{k} P(X_i|Y = y_j)}$$

In addition to the computation of the posterior probabilities for each class, the NB classification rule picks the class with the highest posterior probability, i.e.

$$Y \leftarrow argmax_i P(Y = y_l)\prod\prod_{i=1}^{k} P(X_i|Y = y_l)$$

Based on the type of features discrete-valued or real-valued probabilities $P(X_i|Y),\ i \in \{1, \dots, k\}$ are estimated in two different ways. However, in both cases, maximum likelihood estimation is usually preferred for estimating $P(X_i|Y)$ (Barber, 2012). When a feature is discrete-valued (i.e. when a feature takes a value from a finite countable set) the maximum likelihood estimate is computed as:

$$P\big(X_i = x_{ij}\big|Y = y_l\big) = \frac{\text{number of times } X_i = x_{ij} \text{ for class } y_l}{\text{number of instances in class } y_l}$$

When $X_i$ is real-valued (i.e. continuous), it is common to fit a Gaussian distribution for each class in $Y$ (Michell, 1997). In this case, the NB classifier is referred as a Gaussian Naïve Bayes Classifier. To this end, we need to estimate the mean $\mu_{il}$ and the variance $\sigma_{il}^2$ for each $y_l \in Y$. It is common to use maximum likelihood estimators to estimate $\mu_{il}$ and $\sigma_{il}^2$. Basically, an estimator of $\mu_{il}$ can be computed as:

$$\hat{\mu}_{il} = \frac{\text{sum of } X_i \text{ values for class } y_l}{\text{number of instances in class } y_l}$$

and an estimator of $\sigma_{il}^2$ can be obtained by:

$$\widehat{\sigma^2}_{il} = \frac{\text{sum of square of differences between } X_i \text{ values and } \hat{\mu}_{il}}{\text{number of instances in class } y_l}$$

Having computed $\hat{\mu}_{il}$ and $\widehat{\sigma^2}_{il}$, $P(X_i = x_{ij} | Y = y_l)$ can be estimated via the following probability density function:

$$P(X_i = x_{ij} | Y = y_l) = \frac{1}{\sqrt{2\widehat{\sigma^2}_{il}\pi}} e^{-\frac{(x_{ij} - \hat{\mu}_{il})^2}{2\widehat{\sigma^2}_{il}}}$$

Finally, we estimate the prior $P(Y = y_l)$ by means of maximum likelihood estimation, which is given by:

$$P(Y = y_l) = \frac{\text{number of instances in class } y_l}{\text{number of instances in class } y_l \text{ in the training set}}$$

Notice that with NB we can model $P(X|Y)$ and $P(Y)$. Since the joint distribution $P(X, Y)$ is equal to $P(X|Y)P(Y)$, we model the joint distribution with NB. As a result, new data instances can be sampled through NB. For this reason, NB is known as a *generative* model (Flach, 2012).

### 3.2.4 KNN

K-Nearest Neighbors (KNN) is also very simple and effective. It is a straightforward extension of Nearest Neighbor which is introduced in the works of T.M. Cover (1968). KNN is an instance-based, nonparametric and lazy-learning classification technique. It has good performance especially for scaled, no missing value and lower dimensional data. Moreover, because of a minimal explicit training phase, this algorithm is learned very fast. Unlike SVM where all the non-support vectors can be discarded, KNN, as the laziest of algorithms, makes a prediction based on the entire training data. This method makes predictions by locating the k most similar cases to a given instance. The output of the KNN classification method is calculated as the class with a majority vote from these k-most similar instances.

Given a training set $\mathcal{D}$ where the training objects $(\mathbf{x}, y) \in \mathcal{D}$ , and **x** is the data of the training object and $y$ is the corresponding class, and a test object $z = (\mathbf{x}', y')$ , the distance $d(\mathbf{x}, \mathbf{x}')$ between $z$ and each training object $(\mathbf{x}, y)$ is calculated to identify the k closest training objects $\mathcal{D}_z \subseteq \mathcal{D}$. One of the most commonly used notions is Euclidean distance:

$$d(x, x') = \sqrt{\sum_{i=1}^{k} (x_i - x'_i)^2}$$

Note that Euclidian distance treats each feature equally and its measures are only valid for continuous variables. For categorical variables, the Hamming distance is used:

$$d_H(x, x') = \sum_{i=1}^{k} |x_i - x'_i|^2, \quad x = y \Longrightarrow d = 0; x \neq y \Longrightarrow d = 1$$

Then the algorithm classifies the test object according to the majority class of its nearest neighbors. The majority voting output $y'$ is given by:

$$y' = \text{argmax}_v \sum_{(x_i, y_i) \in \mathcal{D}_z} I(v = y_i),$$

where the class label $v$ is determined by the class label for $i$th nearest neighbour $y_i$, and the indicator function $I(\cdot) = \begin{cases} 1, & \text{if its argument is true} \\ 0, & \text{otherwise} \end{cases}$. KNN is suitable for multi-class problems where an object could belong to one of several classes.

### 3.2.5 Trees and rules

#### 3.2.5.1 CART

Classification and Regression Tree (CART) is also a useful algorithm for predictive machine learning. CART was introduced by Breiman, Friedman, Olshen, and Stone (1984) and represents a crucial milestone in the fields of nonparametric statistics, data mining, Machine Learning, and Artificial Intelligence (AI). It is a powerful and simple decision tree algorithm. The basic idea here is to find binary splits on variables in a top-down manner, where each split produces a right child and a left child. The best split at each stage is chosen to be the one that reduces the total impurity of the child nodes, compared to the parent node. All the variables are tested to find the best individual split at each stage. Successive splits result in the construction of a tree. If this procedure is

repeated for all cases until there is only one value left at the final leaf, there will be overfitting and inaccurate prediction. To prevent overfitting, a stopping criterion is used based on a minimum number of leaves.

Deviance or entropy may be used to determine impurity. The impurity at a leaf is given by:

$$I_t = -2 \sum_{j\ at\ t} n_{tj} \log p_{j|t}$$

where $p_{j|t} = p_{tj}/p_t$, $p_{tj}$ is the probability of a case reaching leaf $t$ within class $j$ and $p_t$ is the probability of reaching leaf $t$. As $p_{j|t}$ is estimated by $n_{tj}\ /n_t$, where $n_{tj}$ is the number of class $j$ reaching leaf $t$ and $n_t$ is the overall number reaching leaf $t$.

Thus, the estimated impurity is:

$$\widehat{I_t} = -2 \sum_{j\ at\ t} n_{tj} \log \frac{n_{tj}}{n_t}$$

And the deviance at a node $T$ is denoted by:

$$I\ (T) = \sum_{leaves\ t \in T} I_t$$

A recursive partitioning routine in R (*rpart*) (Therneau, Atkinson and Foundation, 1997) is used in this study to construct classification trees.

#### 3.2.5.2 C4.5

Decision Tree C4.5 is one of the most widely applied learning methods for classification (Quinlan, 1993). It applies a divide-conquer approach to build decision trees.

C4.5 is different from CART in the following respects (Wu et al., 2008):

- C4.5 allows binary or multi-classes tests, whereas tasks in CART are always binary.
- C4.5 prunes trees using a single-pass method induced from binomial confidence limits; CART uses a model whose parameters are estimated by cross-validation.
- C4.5 applies information-based criteria to rank tests, while CART uses the Gini diversity index.
- When referring to unknown value attributes in a case, C4.5 distributes the case according to their probabilities, but CART searches for alternatives which approximate the outcomes.

## 3.3 Ensemble methods

### 3.3.1 Bagging

Ensemble methods are introduced to increase the accuracy of the models. Bagging (or bootstrap aggregating) is a meta technique for improving classifiers by reducing the variance component of their error (Breiman, 1996). Given the training data set $T$ of size $m$, bagging first generates $k$ new training set $T_i$ of size $m$ by sampling instances from $T$

uniformly with replacement. Then, for each training sample $T_i$ a classifier $h_i$ is trained. Bagging provides a class estimate for any instance by generating a posterior distribution of scores $s_y$ over all the possible classes $y$ such that the score for each class $y$ is the sum of scores $s_{yi}$ obtained by the classifiers $h_i$ with $i \in \{1,2,3, \dots k\}$. The class $y$ with the highest score $s_y$ is the estimated class for that instance.

### 3.3.2 Boosting

Boosting, another method of combining weak classifiers, was summarized in the works of Freund and Schapire (1999). The logic of boosting is parallel to that of bagging except for using a more sophisticated way of sampling training sets. More specifically, for bagging there is the weight for each bag while in boosting, the weight is assigned to each instance. The basic idea is to create a strong classifier using a weighted ensemble of the weak classifiers. The procedure may be described as follows (Hastie et al., 2001):

- Set the initial weights $w_i = \frac{1}{n},\ i = 1, 2, 3, \dots, n$
- For $m = 1, 2, 3, \dots, M$, by using a current weight, find a classifier $C_m(x_i)$ and find an error measure which lies in (0, 1), equal to 1 if none of the cases are correctly classified and 0 if the opposite:

$$error_m = \frac{\sum_{i=1}^{n} w_i I(y_i \neq C_m(x_i))}{\sum_{i=1}^{n} w_i}$$

- Then compute new weights sign: $w_i = w_i \exp[a_m I(y_i \neq C_m(x_i))],\ i = 1,2,3, \dots, n$, where $\alpha_m = \log((1 - error_m)/error_m)$,
- Thus, the final output class for $x_i$ is $C(x_i) = \text{sign}[\sum_{m=1}^{M} a_m C_m(x_i)]$.

## 3.4 Evaluation

In this thesis, we use two methods to evaluate the performance of a classifier: hold-out method and 10-fold cross-validation.

To estimate the performance of a classifier using the hold-out method, we need to divide the data into two sets: a training set and test set. The classifier is built on the training set. The built classifier is then used to predict the labels of instances in the test set. The error on the test set is recorded.

To estimate the accuracy using k-fold cross-validation, the data is divided randomly into 10 exclusive subsets (each subset is called a fold). Each fold is in turn used as the test set and the collection of the remaining nine (9) folds is used as the training set. A classifier is built on each training set and tested using the corresponding test data. Consequently, we end up with 10 classifiers and the average performance of them yields a cross-validation estimate of the performance accuracy.

To decide whether a predictive model is robust or good enough for our classification tasks, accuracy alone is typically not enough to make this decision. The performance of classification in this thesis is quantified using the following performance measures: accuracy, precision, recall, Area Under ROC Curve (AUC). Before defining the last three measurements, we first introduce the following concepts that form the basis for these

measurements. Assume the class set contains two labels: positive and negative. Given a classifier and an instance, there are four (4) possible outcomes:

- True positive (TP): the instance is positive, and it is classified as positive.
- False negative (FN): the instance is positive, and it is classified as negative.
- True negative (TN): the instance is negative, and it is classified as negative.
- False positive (FP): the instance is negative, and it is classified as positive.

Table 3.1 Truth table confusion matrix

| | | Actual value | |
|---|---|---|---|
| | | Positive | Negative |
| Prediction outcome | Positive | True Positive (TP) | False Positive (FP) |
| | Negative | False Negative (FN) | True Negative (TN) |

Accuracy is defined as the probability of correctly classifying a test instance:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total number of instances}}$$

Precision is called positive predictive value, and computed as:

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

Recall is also referred to true positive rate and computed as:

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

**AUC**

Given a test set, the true positive rate (tp rate) of a classifier is computed as TP/(TP+FN), and the false positive rate (fp rate) is computed as FP/(FP+TN). A ROC (Receiver Operating Characteristic curve) graph is defined as a two-dimensional graph in which the tp rate is plotted on the Y axis and the fp rate is plotted on the X axis (Fawcett, 2006). As one of the most complete performance measures for the prediction of a binary variable, ROC measures tp rate and fp rate at all possible thresholds in a binary prediction. It depicts relative trade-offs between true positives and false positives. When the classifier is a discrete classifier (e.g., decision trees, KNN), it produces one (fp rate, tp rate) pair corresponding to a single point in the ROC graph. When the classifier is a scoring or probabilistic classifier (e.g., logistic regression, SVM, Naïve Bayes), the outcome corresponds to a curve in the ROC graph.

To compare the performance of different learning methods we need to reduce the ROC performance to a single scalar representation. To this end, the notion of the area under the ROC curve (AUC) was introduced by Bradley (1997). AUC represents the probability that a randomly chosen positive class instance will be ranked higher by a scoring classifier than a randomly chosen negative class (Bradley, 1997; Fawcett, 2006). The random guessing scoring classifier corresponds to an AUC value equal to 0.5. Any realistic classifier should have an AUC higher than 0.5 and the higher the AUC is, the better the classifier performs.

# 4 Experiment and results

## 4.1 Experiment environment and setup

The classification results and their analyses were obtained using R 3.3.2 and SAS 9.4. For Logistic Regression and Naïve Bayes, we employ the Logistic classifier provided by the *glm()* and *e1071* and *klaR* package for Naïve Bayes. For Support Vector Machine, we use the *e1071* and *kernlab* packages. For the Decision Trees, we use the library *rpart* and *tree* in R and *Rweka* package for C4.5. SAS statistical routines are used for conducting the comparative analyses of results.

## 4.2 Results

The experiments are divided into two parts. In the first part, we compare the performance of five (5) algorithms: logistic regression, SVM, Naïve Bayes, KNN and C4.5. The performances of these classifiers are evaluated in terms of accuracy, precision, recall, and AUC using the hold-out method repeated 50 times. In the second part, we repeat the same experiments for two ensemble classifiers- Bagging and Boosting- in which the aforementioned five (5) single classifiers are employed as base classifiers. The performances are evaluated using 10-fold cross-validation.

### 4.2.1 Single classifier

#### 4.2.1.1 Parameter setting

In this subsection, we show the performance of single classifiers.

**Logistic regression**

As shown in the confusion matrix (Table 4.1) above, this algorithm does not perform very well on our data set, showing 45.8% mean accuracy. Here we see that users from the upper SES class are better classified than those from the other two classes. The precision and recall per class are presented in the extensions of the confusion matrix (62.7%, 29.5%, and 37.7% precision for upper, middle, and lower SES classes; 51.4%, 41.2%, and 37.9% recall for upper, middle, and lower SES classes)

Table 4.1 The cumulative confusion matrix for the classification task using LR.

The columns (T1/T2/T3) represent the actual target SES class labels (upper/middle/lower) and the rows (O1/O2/O3) contain prediction outcome ones. The precision and recall for each class are shown in row and column extensions represented by P and R.

| | **T1** | **T2** | **T3** | **P** |
|---|---|---|---|---|
| **O1** | **365** | 113 | 104 | 62.7% |
| **O2** | 222 | **131** | 91 | 29.5% |
| **O3** | 123 | 74 | **119** | 37.7% |
| **R** | 51.4% | 41.2% | 37.9% | **45.8%** |

The ROC graphs for a binary classifier (one against others) are displayed in Figure 4.1 – Figure 4.3, where the X axis stands for False Positive Rate (fp rate) and the Y axis presents the True Positive Rate (tp rate). With an average AUC of 0.607, the performance of LR is better than randomly guessing (AUC = 0.5).

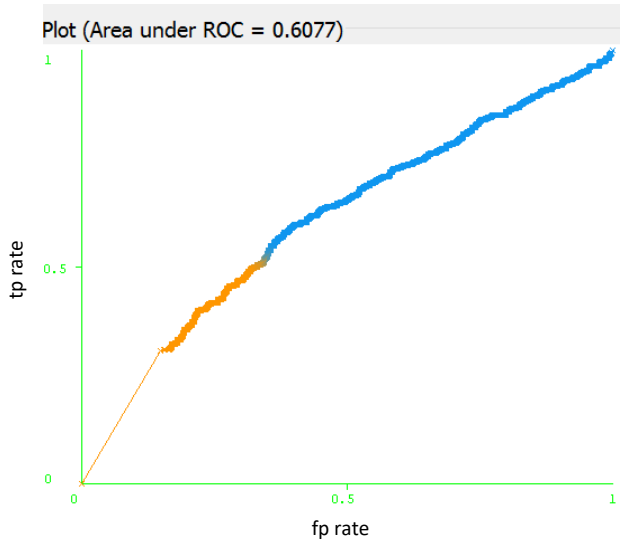


Figure 4.1 ROC plot of upper class vs. others classes for LR.

(Blue to Orange represents Threshold 0 to 1)

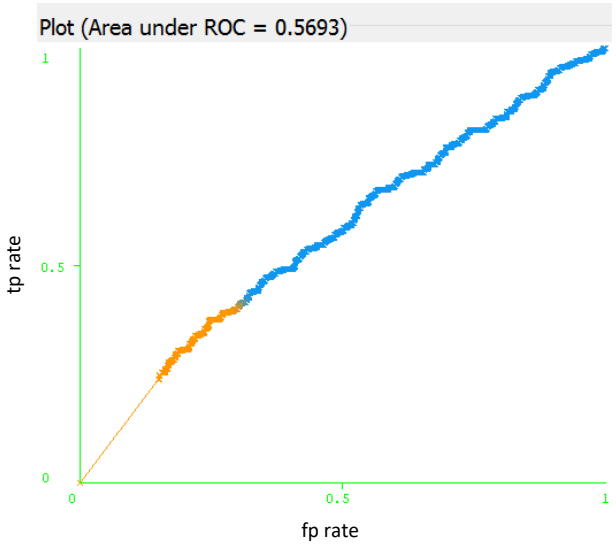


Figure 4.2 ROC plot of middle class vs. others classes for LR.

(Blue to Orange represents Threshold 0 to 1)

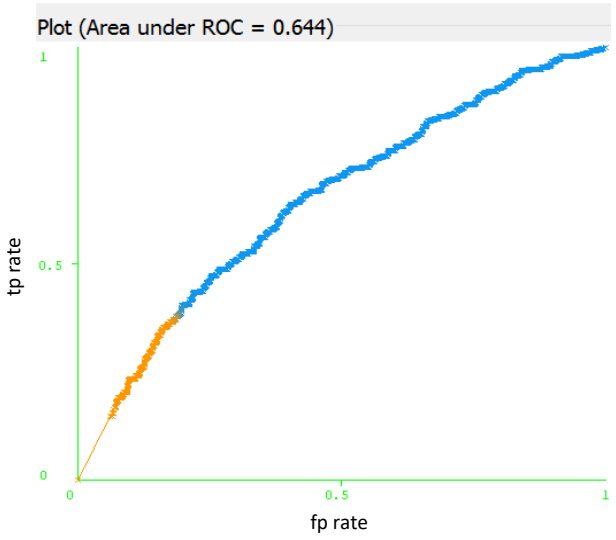


Figure 4.3 ROC plot of lower class vs. others classes for LR.

(Blue to Orange represents Threshold 0 to 1)

**SVM**

Radial basis function (RBF) is first used as the kernel of SVM for our data set, with the parameter setting cost = 100, gamma = 0.001. The confusion matrix is given in Table 4.2. Then The percentage of correctly classified instances is 70.8%. The precision for upper, middle, and lower SES class is 73.4%, 71.4%, and 63.0% respectively; the recall is 87.2%, 44.7%, and 60.2% for each class level. The AUC of this learning method is very good with an average of 0.854, as shown in the ROC graphs (Figure 4.4 – Figure 4.6).

Table 4.2 The cumulative confusion matrix for the classification task using SVM.

The columns (T1/T2/T3) represent the actual target SES class labels (upper/middle/lower) and the rows (O1/O2/O3) contain prediction outcome ones. The precision and recall for each class are shown in row and column extensions represented by P and R.

| | **T1** | **T2** | **T3** | **P** |
|---|---|---|---|---|
| **O1** | **619** | 123 | 101 | 73.4% |
| **O2** | 33 | **142** | 24 | 71.4% |
| **O3** | 58 | 53 | **189** | 63.0% |
| **R** | 87.2% | 44.7% | 60.2% | **70.8%** |

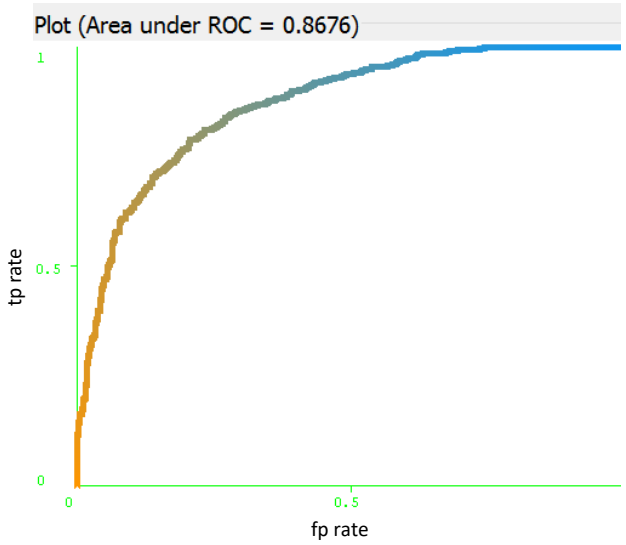


Figure 4.4 ROC plot of upper class vs. others classes for SVM.

(Blue to Orange represents Threshold 0 to 1)

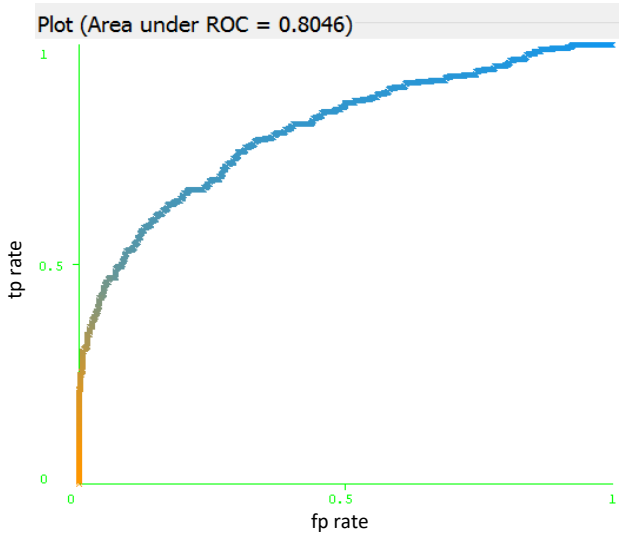


Figure 4.5 ROC plot of middle class vs. others classes for SVM.

(Blue to Orange represents Threshold 0 to 1)

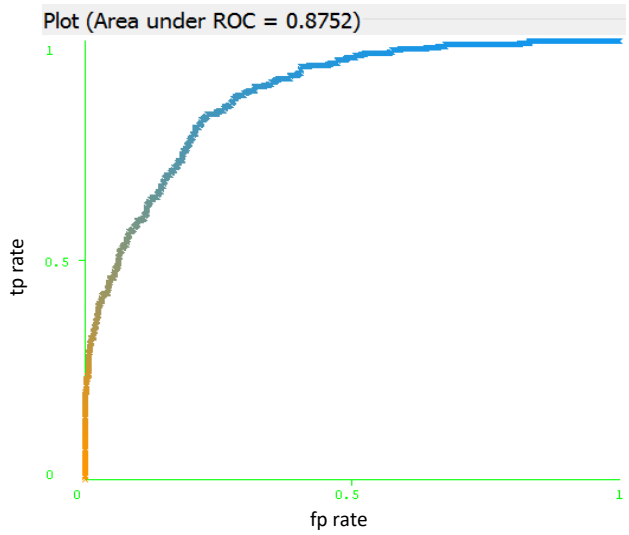


Figure 4.6 ROC plot of lower class vs. others classes for SVM.

(Blue to Orange represents Threshold 0 to 1)

Table 4.3 gives a simple comparison of the accuracy of SVMs using different kernels (RBF, Linear, Polynomial, Sigmoid, and PRBF). The RBF kernel shows the best performance on our data set because the radial kernel is very local and concentrated and can build complex regions in the feature space. The linear kernel function ranks second because of the high dimension of our data set.

Table 4.3 Accuracy of SVM with kernels

| kernels | Radial basis (gamma=0.001) | linear | Polynomial (d=3) | Sigmoid (coef0 = 0) | Polynomial radial basis |
|---|---|---|---|---|---|
| Accuracy | 71.94% | 64.07% | 54.40% | 49.01% | 55.22% |

**Naïve Bayes**

From the confusion matrix for NB (Table 4.4) (using naiveBayes method in the *e1071* package or NaiveBayes method in the *klaR* package with default setting: a normal density is estimated), the total accuracy of classification for this algorithm is 61.5%, with 77.4%, 43.6%, 51.9% precision and 67.6%, 49.1%, 60.2% recall for the upper class, middle class, and lower class respectively. While using a kernel density estimator for density estimation in this algorithm, the correctly classified instances increase to 65.1% presented in Table 4.5.

Table 4.4 The cumulative confusion matrix for the classification task using NB (default). The columns (T1/T2/T3) represent the actual target SES class labels (upper/middle/lower) and the rows (O1/O2/O3) contain prediction outcomes. The precision and recall for each class are shown in row and column extensions represented by P and R.

| | **T1** | **T2** | **T3** | **P** |
|---|---|---|---|---|
| **O1** | **480** | 83 | 57 | 77.4% |
| **O2** | 134 | **156** | 68 | 43.6% |
| **O3** | 96 | 79 | **189** | 51.9% |
| **R** | 67.6% | 49.1% | 60.2% | **61.5%** |

Table 4.5 The cumulative confusion matrix for the classification task using NB (kernel estimator). The columns (T1/T2/T3) represent the actual target SES class labels (upper/middle/lower) and the rows (O1/O2/O3) contain prediction outcome ones. The precision and recall for each class are shown in row and column extensions represented by P and R.

| | **T1** | **T2** | **T3** | **P** |
|---|---|---|---|---|
| **O1** | **556** | 119 | 86 | 73.1% |
| **O2** | 52 | **121** | 32 | 59.0% |
| **O3** | 102 | 78 | **196** | 52.1% |
| **R** | 78.3% | 38.1% | 62.4% | **65.1%** |

**KNN**

$k$ represents the number of neighbors that influence the classification. If $k = 1$, the algorithm then is called the Nearest Neighbor algorithm. A small value of $k$ means that in this case the noise will have a high impact on the result, while a large value of $k$ will lead to an expensive computation. As can be seen in Table 4.6, the accuracy rate for KNN has been improved by altering the value of $k$. We choose $k = 3$ for the following experiments due to computational efficiency. The ROC curves present the AUCs of this algorithm, with an average of 0.728 (Figure 4.7– Figure 4.9).

Table 4.6 Accuracy of KNN with different values of k

| | k=1 | k=3 | k=5 | k=7 | k=10 | k=11 | k=12 | k=13 | k=17 | k=20 | k=21 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ACC (%) | 51.9 | 56.6 | 57.1 | 57.9 | 59.5 | 60.4 | 60.3 | 59.6 | 59.5 | 60.8 | 60.5 |

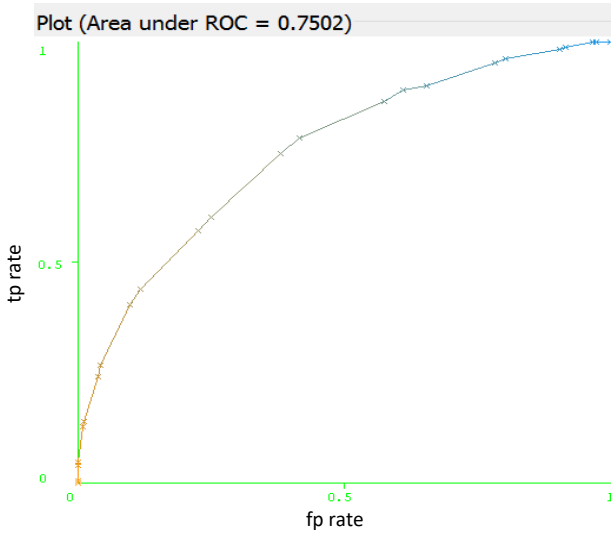


Figure 4.7 ROC plot of upper class vs. others classes for KNN.

(Blue to Orange represents Threshold 0 to 1)

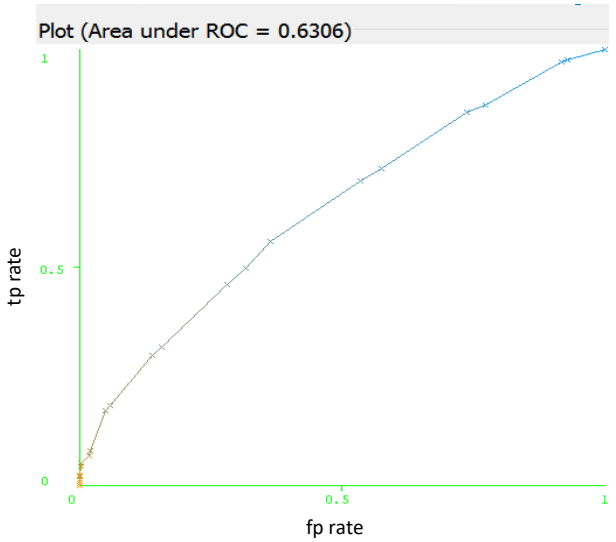


Figure 4.8 ROC plot of middle class vs. others classes for KNN.

(Blue to Orange represents Threshold 0 to 1)

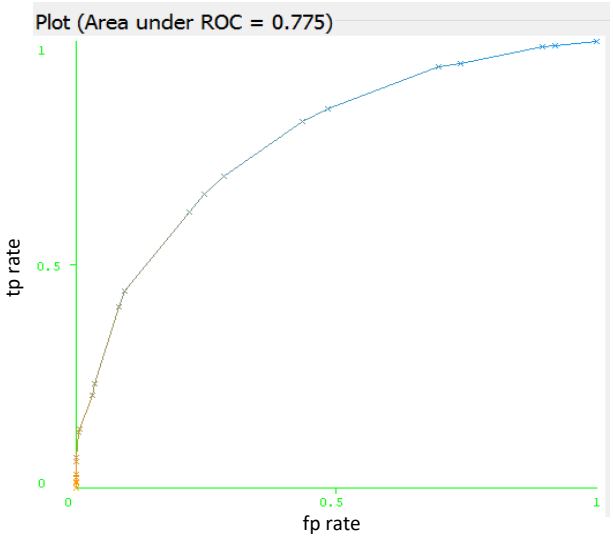


Figure 4.9 ROC plot of lower class vs. others classes for KNN.

(Blue to Orange represents Threshold 0 to 1)

## C 4.5

Table 4.7 The cumulative confusion matrix for the classification task using C4.5.

The columns (T1/T2/T3) represent the actual target SES class labels (upper/middle/lower) and the rows (O1/O2/O3) contain prediction outcome ones. The precision and recall for each class are shown in row and column extensions represented by P and R.

| | **T1** | **T2** | **T3** | **P** |
|---|---|---|---|---|
| **O1** | **483** | 109 | 111 | 68.7% |
| **O2** | 128 | **157** | 60 | 45.5% |
| **O3** | 99 | 52 | **143** | 48.6% |
| **R** | 68.0% | 49.4% | 45.5% | **58.3%** |

As can be seen in Table 4.7, the correctly classified instances for C4.5 are 58.3%. For OSN users in the upper class, classification precision of 68.0% exceeds that of 49.4% for middle SES OSN uses and 45.5% for lower SES OSN users. The similarity is found in recalls. In this study, we consider only C4.5 because it allows both binary and multi-classes classifications. It performs better on our data set than CART.

#### 4.2.1.2 Compare the performance of five single classifiers

We compare the performance of five (5) single classifiers: LR, SVM, NB, KNN, and C4.5 because each algorithm has its assumptions, strengths, and weaknesses. Experiments are conducted on the original data set and five (5) reduced-dimensional data sets. The reduced-dimensional data set are generated by applying Principal Component Analysis (PCA) on the original data. The data sets are described in Table 4.8 below.

Table 4.8 Description of data sets used in the experiments

| Data set | Variance covered | Dimension |
|---|---|---|
| Original data | 1.00 | 1291 |
| PCA (0.9) | 0.90 | 572 |
| PCA (0.8) | 0.80 | 416 |
| PCA (0.7) | 0.70 | 299 |
| PCA (0.6) | 0.60 | 225 |
| PCA (0.5) | 0.50 | 158 |

Table 4.9 Accuracy of LR, SVM, NB, KNN and C4.5

| Accuracy | LR | SVM | NB | KNN | C4.5 |
|---|---|---|---|---|---|
| Original data | 46.79% | 71.65% | 61.02% | 57.70% | 57.66% |
| PCA (0.9) | 59.82% | 67.48% | 43.63% | 52.92% | 50.05% |
| PCA (0.8) | 55.49% | 66.96% | 43.49% | 55.90% | 50.83% |
| PCA (0.7) | 59.97% | 66.74% | 44.30% | 56.29% | 50.75% |
| PCA (0.6) | 65.54% | 65.56% | 43.57% | 59.36% | 50.77% |
| PCA (0.5) | 68.07% | 64.71% | 44.90% | 61.86% | 51.23% |

Accuracy is a simple measure which is well suited for assessing multi-label classification and well-balanced data. As shown in Table 4.9, the accuracy of LR grows as the dimension of the data drops down (the accuracy grows from 46.79% to 68.07%, while the dimension drops from 1291 to 158). This is because LR is a rather simple algorithm, and thus it is more suitable for low-dimensional data. A similar pattern can be also found for KNN. It achieves the best result (an accuracy of 61.86%) when the dimension of input data reduces to only 158. It is because in a very high dimensional space the pair-wise Euclidean distances between points can be misleading.

On the contrary, the accuracies of SVM, NB, and C4.5 decrease as the dimension of the data decreases (e.g., the accuracy of SVM drops from 71.65% to 64.71% as the dimension of the data is reduced to 158). This is because of the information loss caused by dimension reduction, i.e., the original data provide more useful information to train these classifiers.

For the original data set with all the information included, SVM achieves the highest accuracy at 71.65%, which is 10% more than for the other four (4) classifiers. This outcome indicates that SVM is a strong classifier and is able to handle a high-dimensional data set. NB occupies the second place with an accuracy of 61.02%, followed by KNN (57.70%) and C4.5 (57.66%). The accuracy of LR is the lowest at 46.79%.

For PCA with 0.9 variance covered, the accuracy of SVM is still the highest but decreases to 67.48%. The same trend can be found in KNN and C4.5 (7% less than that trained on the original data). NB shows a significant reduction of around 20%. This is evidence that NB performs well with complete information and that the accuracy declines fast in case of information loss due to reduced dimensionality. However, for LR, the accuracy increases to 59.82%. This result indicates that LR performs better for low dimensional data sets.

When considering data with 0.6 variance covered (225 features), LR and SVM show almost the same accuracy of around 65.55% while the other three algorithms NB, KNN, C4.5 perform considerably worse, at 43.57%, 59.36%, and 50.77% respectively.

At 0.5 variance covered, the performance of LR is the best among all the classifiers, at 68.07%. NB results in the lowest accuracy, only 44.90%, followed by C4.5 with 51.23% accuracy. SVM and KNN have similar classification accuracy, at 64.71% and 61.86% respectively.

Table 4.10 Precision of LR, SVM, NB, KNN and C4.5

| Precision | LR | SVM | NB | KNN | C4.5 |
|---|---|---|---|---|---|
| Original data | 0.62 | 0.74 | 0.77 | 0.65 | 0.67 |
| PCA (0.9) | 0.76 | 0.70 | 0.60 | 0.57 | 0.63 |
| PCA (0.8) | 0.74 | 0.72 | 0.61 | 0.63 | 0.63 |
| PCA (0.7) | 0.75 | 0.73 | 0.63 | 0.67 | 0.63 |
| PCA (0.6) | 0.77 | 0.73 | 0.65 | 0.69 | 0.63 |
| PCA (0.5) | 0.77 | 0.74 | 0.68 | 0.72 | 0.64 |

As is shown in Table 4.10, the precision of LR is the lowest for the original data, but it sharply rises from 0.62 to 0.76 as the dimension of the data is reduced to half of the original dimension. Further dimension reduction seems to have little effect since it remains around 0.77. The precision of SVM appears insensitive to dimension reduction, i.e., it stays at 0.7 with small fluctuations. NB achieves the highest precision for the original data set, and drops after dimension reduction. The precision of KNN grows as the dimension decreases. C4.5 does not provide very promising precision for these data sets.

Table 4.11 Recall of LR, SVM, NB, KNN and C4.5

| Recall | LR | SVM | NB | KNN | C4.5 |
|---|---|---|---|---|---|
| Original data | 0.52 | 0.89 | 0.68 | 0.79 | 0.70 |
| PCA (0.9) | 0.67 | 0.87 | 0.51 | 0.83 | 0.62 |
| PCA (0.8) | 0.61 | 0.84 | 0.50 | 0.77 | 0.64 |
| PCA (0.7) | 0.66 | 0.82 | 0.49 | 0.73 | 0.64 |
| PCA (0.6) | 0.75 | 0.80 | 0.46 | 0.75 | 0.64 |
| PCA (0.5) | 0.78 | 0.77 | 0.46 | 0.76 | 0.64 |

According to Table 4.11, the recall of classifiers generally declines as dimensionality decreases the only exception being for LR. The recall of LR rises from the lowest level (0.52) to the highest level (0.78). SVM outperforms other algorithms most of the time. The advantage is most visible for the original data set, where SVM achieves a recall of 0.89. The recall of SVM drops slowly as the dimension decreases, and even results in fairly high recall (0.77) for the PCA (0.5) set. A similar situation happens for KNN, i.e., it has a good performance (more than 0.73 for all cases) for recall. NB and C4.5 only perform well on the original data set and both decrease markedly when applied on reduced-dimensional data sets. To summarize, recall suffers with dimension reduction. In high dimension, SVM provides the most promising recall.

Table 4.12 AUCs of LR, SVM, NB, KNN and C4.5

| AUC | LR | SVM | NB | KNN | C4.5 |
|---|---|---|---|---|---|
| Original data | 0.60 | 0.77 | 0.77 | 0.73 | 0.66 |
| PCA (0.9) | 0.77 | 0.73 | 0.61 | 0.70 | 0.60 |
| PCA (0.8) | 0.73 | 0.73 | 0.61 | 0.70 | 0.61 |
| PCA (0.7) | 0.77 | 0.74 | 0.62 | 0.74 | 0.61 |
| PCA (0.6) | 0.83 | 0.74 | 0.63 | 0.76 | 0.61 |
| PCA (0.5) | 0.84 | 0.74 | 0.65 | 0.79 | 0.61 |

Table 4.12 shows the AUCs of the five (5) classifiers. For LR and KNN, the AUCs generally grow as the dimension decreases. Especially for LR, the AUC increases steeply from 0.60 to 0.84. The AUCs for SVM, NB, and C4.5 decrease slightly at the beginning (i.e., when the number of dimensions reduces to half of the original dimension), then the AUCs level off. This result indicates that AUCs of SVM, NB, and C4.5 are not sensitive to dimension reduction.

Computational efficiency is an important criterion to evaluate a learning method. It implies the applicability of classifiers. For example, a classifier that involves a huge computational overhead can hardly be applied to big data. In the next table (Table 4.13), we compare the computing time for each of the five (5) classifiers.

Table 4.13 Computing time of LR, SVM, NB, KNN and C4.5

| Time （seconds） | LR | SVM | NB | KNN | C4.5 |
|---|---|---|---|---|---|
| Original data | 59.31 | 17.45 | 1.24 | 15.61 | 2.59 |
| PCA (0.9) | 26.15 | 16.83 | 0.55 | 9.66 | 0.97 |
| PCA (0.8) | 9.27 | 12.27 | 0.38 | 6.2 | 0.66 |
| PCA (0.7) | 2.66 | 11.87 | 0.27 | 4.79 | 0.49 |
| PCA (0.6) | 1.94 | 8.99 | 0.26 | 3.47 | 0.35 |
| PCA (0.5) | 0.99 | 5.17 | 0.16 | 2.35 | 0.24 |

When using the original data, LR takes on average 59.31 seconds due to the complexity of solving the optimization problem for such high-dimensional data. When the number of dimensions reduces, the computing time decreases significantly. SVM takes an average of 17.45 seconds for the original data set and decreases steadily when dimension reduction occurs. Similarly, KNN runs 15.61 seconds for the original data and falls to 2.35 which is considerably longer than LR, NB and C4.5. NB and C4.5 run very fast even for the original high-dimensional data and NB is the fastest for all cases.

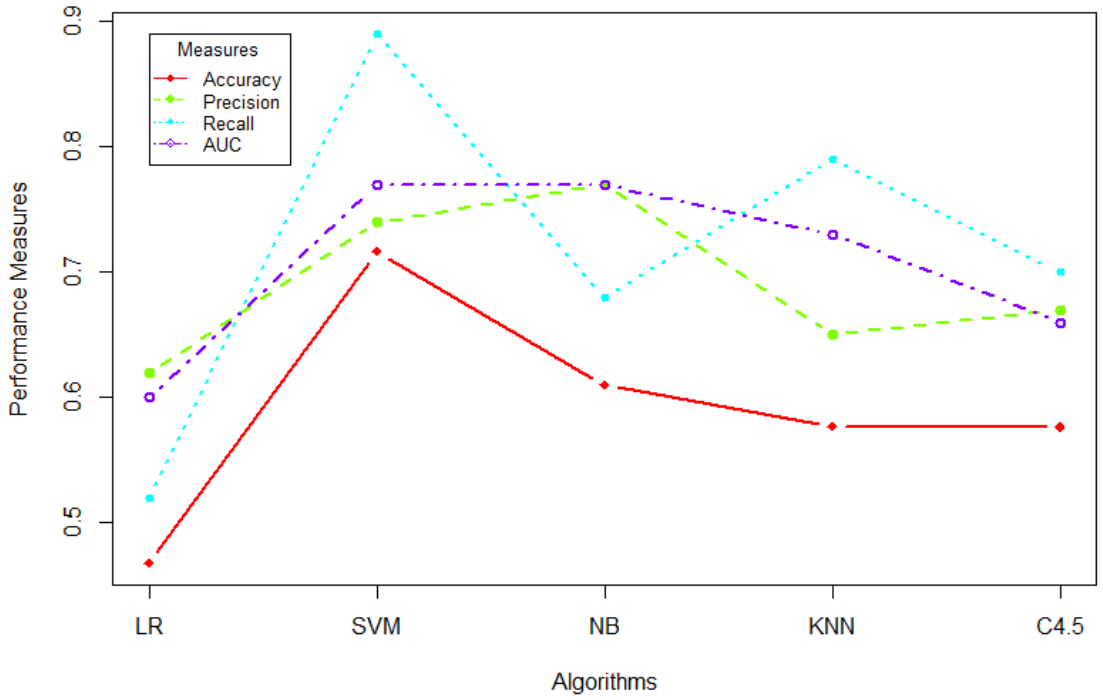


Figure 4.10 Line chart of performance measures for five algorithms on original data

Table 4.14 Comparison of performance measures by descending orders on original data

| Original data | | | | | |
|---|---|---|---|---|---|
| Accuracy | SVM | NB | KNN | C4.5 | LR |
| Precision | NB | SVM | C4.5 | KNN | LR |
| Recall | SVM | KNN | C4.5 | NB | LR |
| AUC | SVM/NB | / | KNN | C4.5 | LR |
| Time | NB | C4.5 | KNN | SVM | LR |

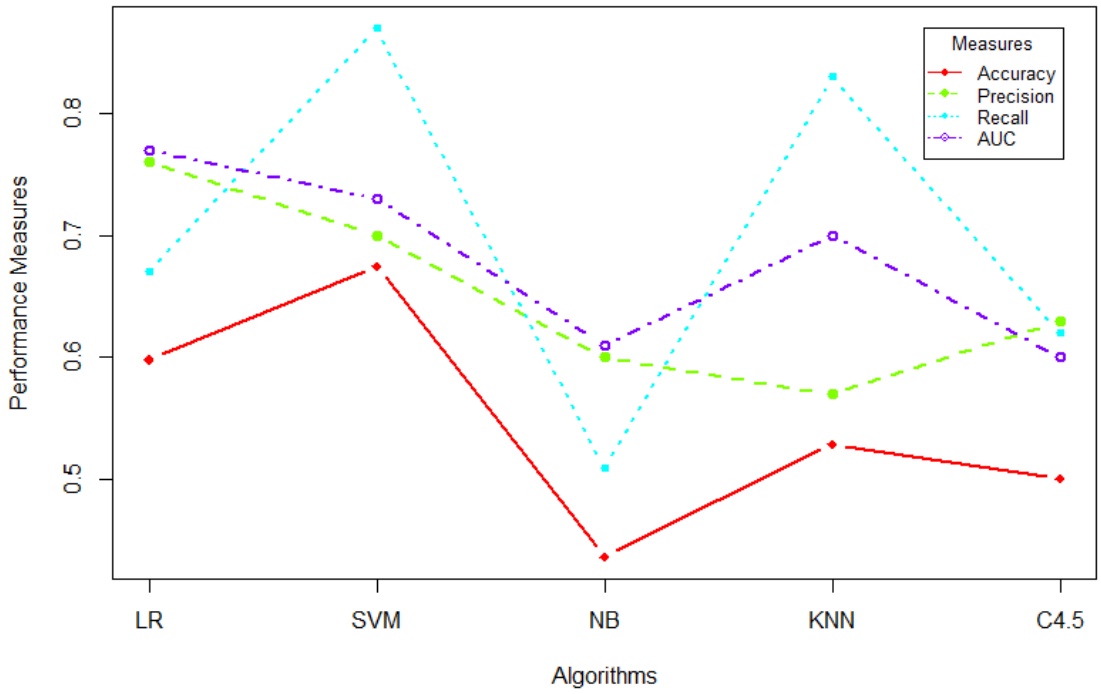


Figure 4.11 Line chart of performance measures for five algorithms on data PCA 0.9

Table 4.15 Comparison of five performance measures by descending orders on data PCA 0.9

| Data PCA (0.9) | | | | | |
|---|---|---|---|---|---|
| Accuracy | SVM | LR | KNN | C4.5 | NB |
| Precision | LR | SVM | C4.5 | NB | KNN |
| Recall | SVM | KNN | LR | C4.5 | NB |
| AUC | LR | SVM | KNN | NB | C4.5 |
| Time | NB | C4.5 | KNN | SVM | LR |

Table 4.14, Figure 4.10 and Table 4.15, Figure 4.11 sort the results of five (5) performance measures on original data and data after dimension reduction with 0.9 variance covered respectively by descending orders. As shown in the graphs above, the outcomes demonstrate that both SVM and NB have good performance on the original data set and are suitable for the high-dimensional data. For the data set with 0.9 variance covered, LR shows its advantages on lower-dimensional data and SVM still has stable performance. But considering computing time, SVM and LR still show weakness owing to their complexity. In the following section, we statistically analyze these results to determine statistical significance.

### 4.2.2 Analysis

#### 4.2.2.1 MANOVA

Multivariate analysis of variance (MANOVA) is a statistical procedure used to identify significant differences among the experimental treatments. In this thesis, in order to obtain a more accurate sense of the effect of the algorithms on the different performance measures, MANOVA is performed on the data obtained by applying the 5 ($k$) algorithms (LR, SVM, NB, KNN, C4.5) on the same data set. We use hold-out method repeated 50 ($n$) times randomly drawing a sample of 10% of the data set each time. For each of the 5 algorithms, we obtain 50 observations for each of four (4) performance measures (accuracy, precision, recall, AUC). Thus, our analysis is on a data set with a total of 250 ($nk$) observations.

In this analysis, treatments and response variables are denoted as:

$$\text{experimental treatments} = \text{algorithms (LR, SVM, NB, kNN, C4.5)}$$

$$\text{response variables} = \text{performance measures (accuracy, precision, recall, AUC)}$$

The means procedure in SAS is displayed in the following table, and shows means along with standard deviations and the values of minimum and maximum.

Table 4.16 Means procedure of response variables for each treatment

**The MEANS Procedure**

| treats | N Obs | Variable | N | Mean | Std Dev | Minimum | Maximum |
|---|---|---|---|---|---|---|---|
| tr1_C4.5 | 50 | accuracy | 50 | 1.7251730 | 0.0890481 | 1.5707963 | 1.9481064 |
| | | precision | 50 | 1.9249225 | 0.0875747 | 1.7721543 | 2.1598278 |
| | | recall | 50 | 1.9771835 | 0.1274750 | 1.6695488 | 2.2942430 |
| | | AUC | 50 | 1.9085590 | 0.0920926 | 1.7385845 | 2.1323312 |
| tr2_KNN | 50 | accuracy | 50 | 1.7257667 | 0.0728381 | 1.5558704 | 1.8583238 |
| | | precision | 50 | 1.8737831 | 0.0753015 | 1.7269866 | 2.0297809 |
| | | recall | 50 | 2.1851342 | 0.1353474 | 1.8417039 | 2.5027997 |
| | | AUC | 50 | 2.0618964 | 0.0887933 | 1.8519269 | 2.2526406 |
| tr3_NB | 50 | accuracy | 50 | 1.7939332 | 0.0861412 | 1.5930204 | 1.9581914 |
| | | precision | 50 | 2.1340553 | 0.1058941 | 1.9217037 | 2.4866847 |
| | | recall | 50 | 1.9388058 | 0.1212992 | 1.6412767 | 2.1864153 |
| | | AUC | 50 | 2.1544153 | 0.0845131 | 1.9676755 | 2.3543340 |
| tr4_SVM | 50 | accuracy | 50 | 2.0198748 | 0.0698810 | 1.8896022 | 2.1777530 |
| | | precision | 50 | 2.0693192 | 0.0769022 | 1.9265682 | 2.2725657 |
| | | recall | 50 | 2.4642836 | 0.1165440 | 2.2213587 | 2.6623058 |
| | | AUC | 50 | 2.1335802 | 0.0849245 | 1.9657355 | 2.3368129 |
| tr5_LR | 50 | accuracy | 50 | 1.5062433 | 0.0987623 | 1.3296603 | 1.7357221 |
| | | precision | 50 | 1.8213696 | 0.1007331 | 1.6449382 | 2.0943951 |
| | | recall | 50 | 1.6115827 | 0.1536954 | 1.3579243 | 2.0226345 |
| | | AUC | 50 | 1.7740481 | 0.1096502 | 1.5777264 | 2.0112347 |

The models for each performance measure (response variable) are defined as:

$$\begin{cases} Y_1 = \mu + \tau_i + \varepsilon_i, \\ Y_2 = \mu^0 + \tau_i^0 + \varepsilon_i^0, \\ Y_3 = \mu' + \tau_i' + \varepsilon_i', \\ Y_4 = \mu^* + \tau_i^* + \varepsilon_i^*, \end{cases}$$

$$i = 1,2,3,4,5$$

where $Y_1,\ Y_2,\ Y_3, Y_4$ are the vectors $(250\times1)$ representing accuracy, precision, recall and AUC, respectively, for each of the 5 algorithms. Here the parameter vector $\mu$, $\mu^0$, $\mu'$, $\mu^*$ is an overall mean, and $\tau_i$, $\tau_i^0$, $\tau_i'$, $\tau_i^*$ represent the effect of algorithm $i$ for accuracy, precision, recall and AUC and all error terms and in general, $\varepsilon_i$, $\varepsilon_i^0$, $\varepsilon_i'$, $\varepsilon_i^* \sim N(0, \Sigma)$.

For each response variable, we have 250 observations, consisting of 50 outcomes for each of 5 algorithms. Table 4.16 details the value of the mean, standard deviation, minimum and maximum for each performance measure by algorithm.

The basic statistical model is that the values of each response variable can be written as:

$$\mathbf{Y} = \mathbf{X\beta} + \boldsymbol{\varepsilon}$$

$\mathbf{Y}$ is the $250\times4$ matrix of vectors $Y_1,\ Y_2,\ Y_3, Y_4$, and $\mathbf{X}$ is a $250\times5$ design matrix for $\boldsymbol{\beta}$ $(5\times4)$. $\boldsymbol{\varepsilon}$ denotes a $250\times4$ matrix compositing by vectors $\varepsilon_i$, $\varepsilon_i^0$, $\varepsilon_i'$, $\varepsilon_i^*$. We can also write the model as follows:

$$\begin{bmatrix} y_{1,1} & y_{2,1} & y_{3,1} & y_{4,1} \\ \vdots & \vdots & \vdots & \vdots \\ y_{1,50} & y_{2,50} & y_{3,50} & y_{4,50} \\ y_{1,51} & y_{2,51} & y_{3,51} & y_{4,51} \\ \vdots & \vdots & \vdots & \vdots \\ y_{1,100} & y_{2,100} & y_{3,100} & y_{4,100} \\ y_{1,101} & y_{2,101} & y_{3,101} & y_{4,101} \\ \vdots & \vdots & \vdots & \vdots \\ y_{1,150} & y_{2,150} & y_{3,150} & y_{4,150} \\ y_{1,151} & y_{2,151} & y_{3,151} & y_{4,151} \\ \vdots & \vdots & \vdots & \vdots \\ y_{1,200} & y_{2,200} & y_{3,200} & y_{4,200} \\ y_{1,201} & y_{2,201} & y_{3,201} & y_{4,201} \\ \vdots & \vdots & \vdots & \vdots \\ y_{1,250} & y_{2,250} & y_{3,250} & y_{4,250} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 0 & 1 \\ 1 & -1 & -1 & -1 & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & -1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} \mu & \mu^0 & \mu' & \mu^* \\ \tau_1 & \tau_1^0 & \tau_1' & \tau_1^* \\ \tau_2 & \tau_2^0 & \tau_2' & \tau_2^* \\ \tau_3 & \tau_3^0 & \tau_3' & \tau_3^* \\ \tau_4 & \tau_4^0 & \tau_4' & \tau_4^* \end{bmatrix} + \boldsymbol{\varepsilon}$$

We obtain all of the multivariate tests from the SAS GLM procedure. We test all the main effects of the experimental treatments (factors) LR, SVM, NB, KNN, C4.5. All of the multivariate test results for each effect that SAS offers are shown in the following section. The first sub-table in Table 4.17 displays the elements of the error matrix, called the Error Sums of Squares and Cross Products (SSCP) matrix. The diagonal elements of this matrix are the error sums of squares from the corresponding univariate analyses. The analysis also gives the partial correlation matrix associated with the E matrix (see Table 4.17). All measures are strongly correlated; the strongest correlation (r = 0.838) is between precision and AUC; the weakest one (r = 0.455) is between recall and precision.

Table 4.17 The GLM Procedure MANOVA_1

| E = Error SSCP Matrix | | | | |
|---|---|---|---|---|
| | accuracy | precision | recall | AUC |
| accuracy | 1.7293369579 | 1.3741921567 | 2.0191776443 | 1.4711405324 |
| precision | 1.3741921567 | 1.9901007333 | 1.3215462046 | 1.710451127 |
| recall | 2.0191776443 | 1.3215462046 | 4.2378657911 | 1.9146217899 |
| AUC | 1.4711405324 | 1.710451127 | 1.9146217899 | 2.0944112576 |

| Partial Correlation Coefficients from the Error SSCP Matrix / Prob > \|r\| | | | | |
|---|---|---|---|---|
| DF = 245 | accuracy | precision | recall | AUC |
| accuracy | 1.000000 | 0.740747<br><.0001 | 0.745867<br><.0001 | 0.773007<br><.0001 |
| precision | 0.740747<br><.0001 | 1.000000 | 0.455063<br><.0001 | 0.837803<br><.0001 |
| recall | 0.745867<br><.0001 | 0.455063<br><.0001 | 1.000000 | 0.642656<br><.0001 |
| AUC | 0.773007<br><.0001 | 0.837803<br><.0001 | 0.642656<br><.0001 | 1.000000 |

The SSCP matrix for treatments is shown in Table 4.18. It gives the Type III **H** matrix where diagonal elements are the model sums of squares for the corresponding univariate analyses.

The characteristic root is the square root of an eigenvalue. From Table 4.18, we can see the first root accounts for 57.81% of the variability in the effect of treatments which in our case are algorithms. The rows of this table describe the characteristic vector i.e. eigenvector of each root. This table shows which variables are influenced most by the algorithm choice. From this, we see that accuracy and precision have the largest magnitudes. Note that no response loading is zero which means that no variable is completely redundant.

The next portion of this table provides the test results for the algorithm (i.e. treatment) effect. SAS produces four (4) test statistics: Wilks' Lambda, Pillai's Trace, Hotelling-Lawley Trace, and Roy's Greatest Root for testing overall s effect of algorithms, based on the characteristic roots and vectors of $\mathbf{E^{-1}H}$. As a good general choice, we consider the F test based on Wilks' Lambda; it has a F value of 130.08 with 16 and 740 degrees of freedom. The p-value is <0.0001, indicating an overall highly significant difference at the level of 0.05. Hence, the $H_0$: $\boldsymbol{\beta} = \mathbf{0}$ can be rejected and the overall effects on the performance measures due to the different algorithms are not all the same.

Table 4.18 The GLM Procedure MANOVA_2

| H = Type III SSCP Matrix for treats | | | | |
|---|---|---|---|---|
| | accuracy | precision | recall | AUC |
| accuracy | 6.7647680123 | 3.690157308 | 10.631315839 | 4.9272505642 |
| precision | 3.690157308 | 3.5009080921 | 3.8979492595 | 3.5261053819 |
| recall | 10.631315839 | 3.8979492595 | 19.935107864 | 7.6364268405 |
| AUC | 4.9272505642 | 3.5261053819 | 7.6364268405 | 5.2361720696 |

| Characteristic Roots and Vectors of: E Inverse * H, where<br>H = Type III SSCP Matrix for treats<br>E = Error SSCP Matrix | | | | | |
|---|---|---|---|---|---|
| | | Characteristic Vector V'EV=1 | | | |
| Characteristic Root | Percent | accuracy | precision | recall | AUC |
| 5.55195757 | 57.82 | 0.54411406 | -0.36259214 | 0.33944940 | -0.09614452 |
| 2.27259029 | 23.67 | -0.68992511 | -0.80680836 | 0.05151314 | 1.32710071 |
| 1.77462680 | 18.48 | 0.23237379 | 0.30239208 | -0.32503150 | 0.45480950 |
| 0.00274019 | 0.03 | -1.26198024 | 1.14239190 | 0.65591314 | -0.53823954 |

| MANOVA Test Criteria and F Approximations for the Hypothesis of No Overall treats Effect<br>H = Type III SSCP Matrix for treats<br>E = Error SSCP Matrix<br><br>S=4 M=-0.5 N=120 | | | | | |
|---|---|---|---|---|---|
| Statistic | Value | F Value | Num DF | Den DF | Pr > F |
| Wilks' Lambda | 0.01676271 | 130.08 | 16 | 739.96 | <.0001 |
| Pillai's Trace | 2.18412938 | 73.67 | 16 | 980 | <.0001 |
| Hotelling-Lawley Trace | 9.60191485 | 144.63 | 16 | 478.03 | <.0001 |
| Roy's Greatest Root | 5.55195757 | 340.06 | 4 | 245 | <.0001 |
| NOTE: F Statistic for Roy's Greatest Root is an upper bound. | | | | | |

#### 4.2.2.2 ANOVA

Separate univariate ANOVAs are then employed to analyze the same data set for each of the performance measures (i.e. the dependent variables).

Each model can be written as:

$$Y_{ij} = \mu + \tau_i + \varepsilon_{ij}, \quad i = 1,2,3,4,5 \quad j = 1,2, \dots, n_i$$

where $\sum_{i=1}^{5} n_i \tau_i = 0$, and $\varepsilon_{ij}\ i.i.d \sim N(0, \sigma^2)$.

Thus, for each performance measure, we first begin by testing whether there is any difference due to the five (5) classification algorithms. i.e. the test of hypothesis becomes:

$$H_0: \tau_1 = \tau_2 = \tau_3 = \tau_4 = \tau_5$$

against

$$H_a: \text{at least one of } \tau_i \text{ is different.}$$

In matrix notation, we have:

$$Y_1 = X_1 \boldsymbol{\beta} + \boldsymbol{\varepsilon_1}$$

$$\begin{bmatrix} y_{1,1} \\ \vdots \\ y_{1,50} \\ y_{1,51} \\ \vdots \\ y_{1,100} \\ y_{1,101} \\ \vdots \\ y_{1,150} \\ y_{1,151} \\ \vdots \\ y_{1,200} \\ y_{1,201} \\ \vdots \\ y_{1,250} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 0 & 1 \\ 1 & -1 & -1 & -1 & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & -1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} \mu \\ \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \end{bmatrix} + \begin{bmatrix} \varepsilon_{1,1} \\ \varepsilon_{1,2} \\ \varepsilon_{1,3} \\ \vdots \\ \varepsilon_{1,250} \end{bmatrix}$$

Box-plots of dependent variables present the difference between different algorithms (i.e. experimental treatments) (Figure 4.12 – 4.15). As can be seen from ANOVAs in Table 4.19, Table 4.21, Table 4.23, Table 4.25, the F values and p-values ($< \alpha = 0.05$) indicate that the effects of algorithms treatments are not all the same for each performance measures (dependent variables) individually.

More specifically, for accuracy (Table 4.19), $F = MS_t/MS_e = 239.60$ (p-value < 0.0001), we reject $H_0$, thus the accuracies of the five (5) algorithms are not all the same. R square = 6.7648/8.4941 = 0.7964 means that 79.64% of the variance of accuracy can be explained by this model. And the standard error of the accuracy ($\sqrt{MS_e}$) is 0.0840. The coefficient of variance C.V = Root MSE/ accuracy Mean = 0.0840/1.7542 *100 = 4.79. Table 4.20 presents Duncan's multiple range test for the 5 classifiers for response

variable accuracy. KNN and C4.5 are not significantly different. In terms of accuracy, the best algorithm is SVM, and NB takes the second place.

For precision in Table 4.21, F value is 107.75 with p-value (Pr > F) < 0.0001 shows the average precision is not constant across the five (5) classifiers. The model here can explain 63.76% of the variance in precision with standard error 0.0901 and coefficient of variance 4.5873. From Duncan's test (Table 4.22) we can conclude that all the treatments are significantly different. Among the five (5) algorithms, NB is the best, followed by SVM.

In terms of recall, again conclude that average recall is not constant across all 5 classifiers, as F value is 288.12 and probability > F is smaller than 0.0001, shown in Table 4.23. The model in the GLM procedure can explain 82.47% of the variance in recall, with coefficient of variance 6.4616 and standard error 0.1315. Duncan's test in Table 4.24 indicates that C4.5 and NB are not significantly different. SVM ranks first again followed by KNN.

For AUC, F value of 153.13 and p-value 0.0001 presented in Table 4.25 demonstrate that this measure does not remain constant across all 5 algorithms. R square is equal to 0.7143, showing the proportion of variance in AUC that is explained by the model with a standard error of 0.0925. From Duncan's test results (Table 4.26), NB and SVM do not give significantly different AUC values.

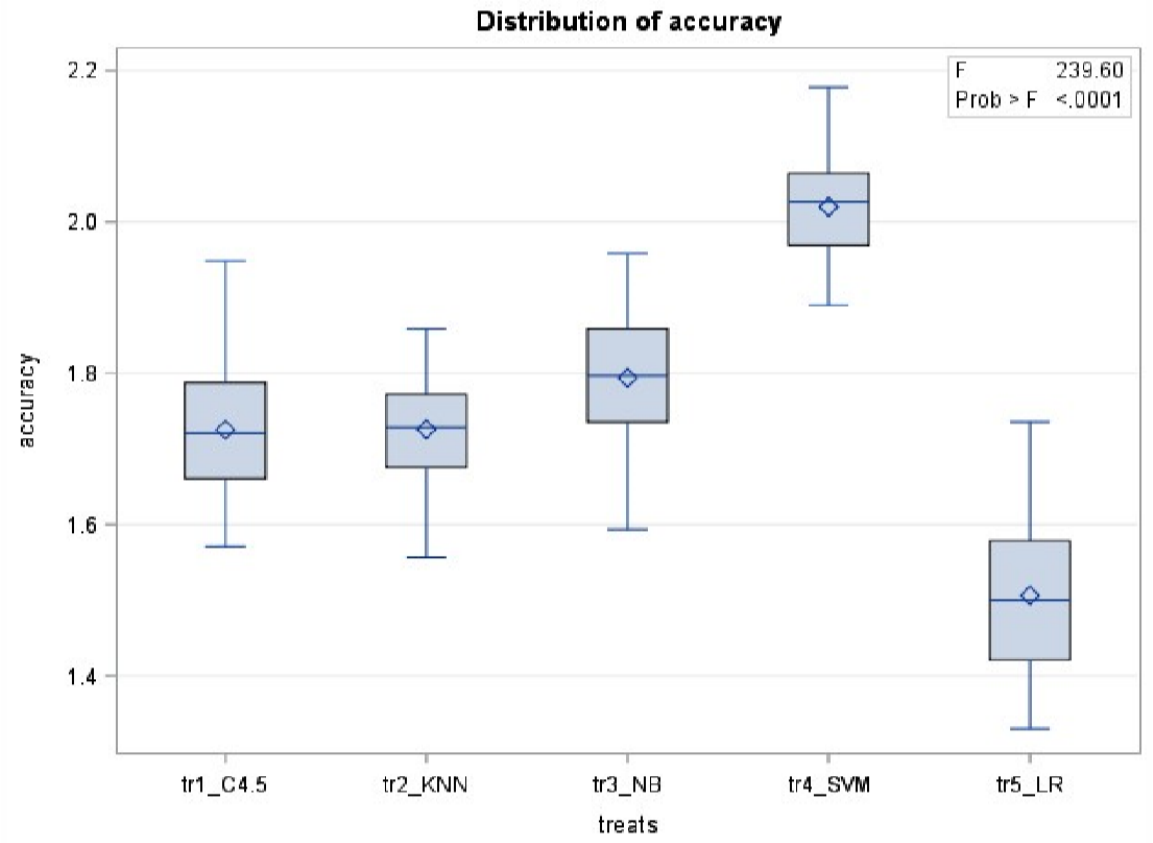


Figure 4.12 Box-plot of accuracy for each treatment

Table 4.19 ANOVA_accuracy

**Dependent Variable: accuracy**

| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| Model | 4 | 6.76476801 | 1.69119200 | 239.60 | <.0001 |
| Error | 245 | 1.72933696 | 0.00705852 | | |
| Corrected Total | 249 | 8.49410497 | | | |

| R-Square | Coeff Var | Root MSE | accuracy Mean |
|---|---|---|---|
| 0.796407 | 4.789367 | 0.084015 | 1.754198 |

| Source | DF | Type III SS | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| treats | 4 | 6.76476801 | 1.69119200 | 239.60 | <.0001 |

Table 4.20 Duncan's Multiple Range Test for accuracy

| | |
|---|---|
| **Alpha** | 0.05 |
| **Error Degrees of Freedom** | 245 |
| **Error Mean Square** | 0.007059 |

| **Number of Means** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|
| **Critical Range** | .03310 | .03484 | .03600 | .03686 |

| **Means with the same letter are not significantly different.** | | | |
|---|---|---|---|
| **Duncan Grouping** | **Mean** | **N** | **treats** |
| A | 2.01987 | 50 | tr4_SVM |
| | | | |
| B | 1.79393 | 50 | tr3_NB |
| | | | |
| C | 1.72577 | 50 | tr2_KNN |
| C | | | |
| C | 1.72517 | 50 | tr1_C4.5 |
| | | | |
| D | 1.50624 | 50 | tr5_LR |

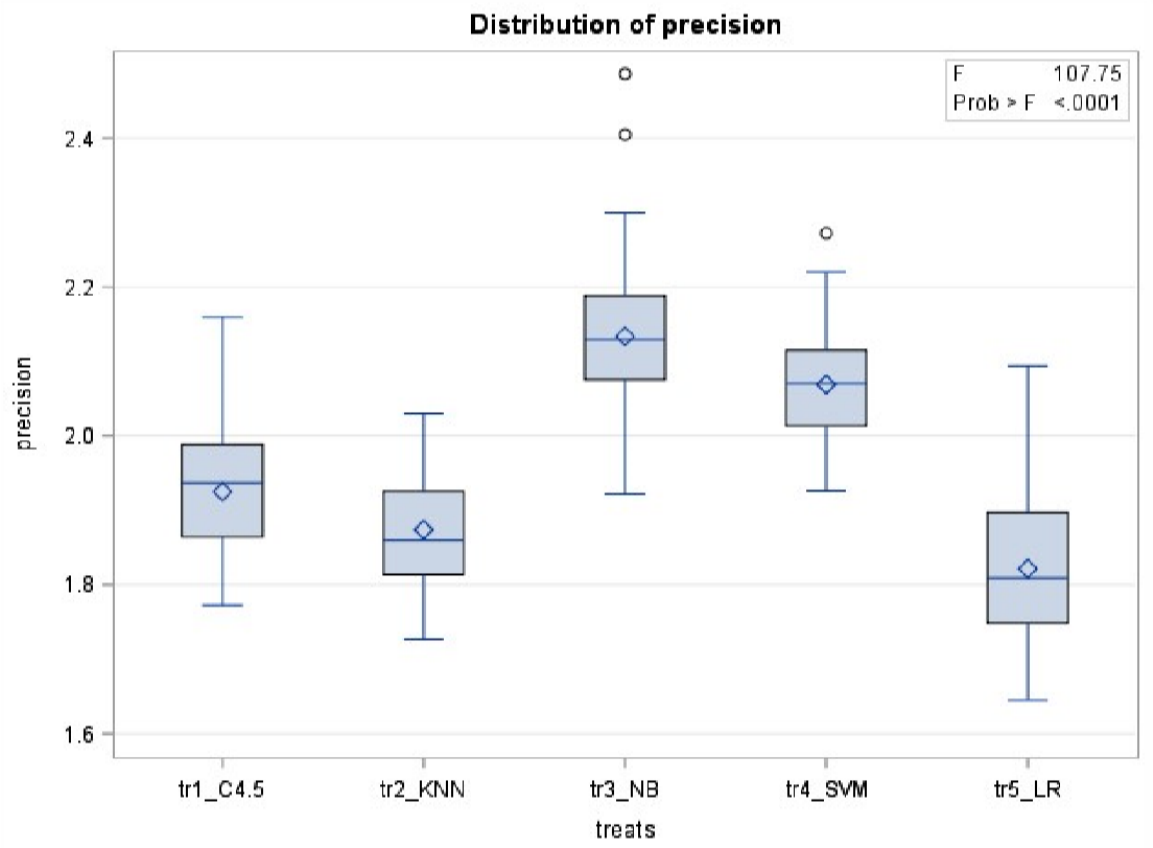


Figure 4.13 Box-plot of precision for each treatment

Table 4.21 ANOVA_precision

**Dependent Variable: precision**

| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| Model | 4 | 3.50090809 | 0.87522702 | 107.75 | <.0001 |
| Error | 245 | 1.99010073 | 0.00812286 | | |
| Corrected Total | 249 | 5.49100883 | | | |

| R-Square | Coeff Var | Root MSE | precision Mean |
|---|---|---|---|
| 0.637571 | 4.587335 | 0.090127 | 1.964690 |

| Source | DF | Type III SS | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| treats | 4 | 3.50090809 | 0.87522702 | 107.75 | <.0001 |

Table 4.22 Duncan's Multiple Range Test for precision

| | |
|---|---|
| **Alpha** | 0.05 |
| **Error Degrees of Freedom** | 245 |
| **Error Mean Square** | 0.008123 |

| **Number of Means** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|
| **Critical Range** | .03550 | .03737 | .03862 | .03954 |

| **Means with the same letter are not significantly different.** | | | |
|---|---|---|---|
| **Duncan Grouping** | **Mean** | **N** | **treats** |
| A | 2.13406 | 50 | tr3_NB |
| | | | |
| B | 2.06932 | 50 | tr4_SVM |
| | | | |
| C | 1.92492 | 50 | tr1_C4.5 |
| | | | |
| D | 1.87378 | 50 | tr2_KNN |
| | | | |
| E | 1.82137 | 50 | tr5_LR |

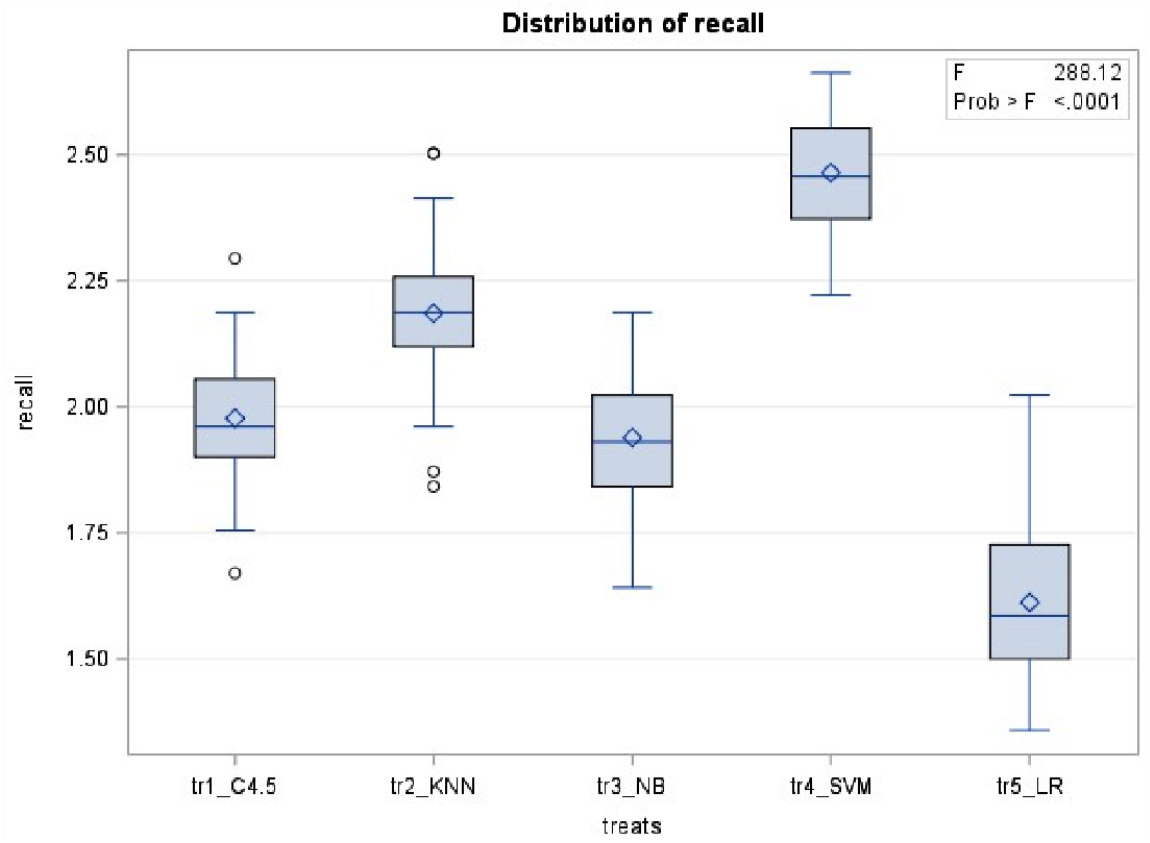


Figure 4.14 Box-plot of recall for each treatment

Table 4.23 ANOVA_recall

**Dependent Variable: recall**

| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| Model | 4 | 19.93510786 | 4.98377697 | 288.12 | <.0001 |
| Error | 245 | 4.23786579 | 0.01729741 | | |
| Corrected Total | 249 | 24.17297365 | | | |

| R-Square | Coeff Var | Root MSE | recall Mean |
|---|---|---|---|
| 0.824686 | 6.461617 | 0.131520 | 2.035398 |

| Source | DF | Type III SS | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| treats | 4 | 19.93510786 | 4.98377697 | 288.12 | <.0001 |

Table 4.24 Duncan's Multiple Range Test for recall

| Alpha | 0.05 |
|---|---|
| Error Degrees of Freedom | 245 |
| Error Mean Square | 0.017297 |

| Number of Means | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| Critical Range | .05181 | .05454 | .05636 | .05771 |

| Means with the same letter are not significantly different. | | | |
|---|---|---|---|
| Duncan Grouping | Mean | N | treats |
| A | 2.46428 | 50 | tr4_SVM |
| | | | |
| B | 2.18513 | 50 | tr2_KNN |
| | | | |
| C | 1.97718 | 50 | tr1_C4.5 |
| C | | | |
| C | 1.93881 | 50 | tr3_NB |
| | | | |
| D | 1.61158 | 50 | tr5_LR |

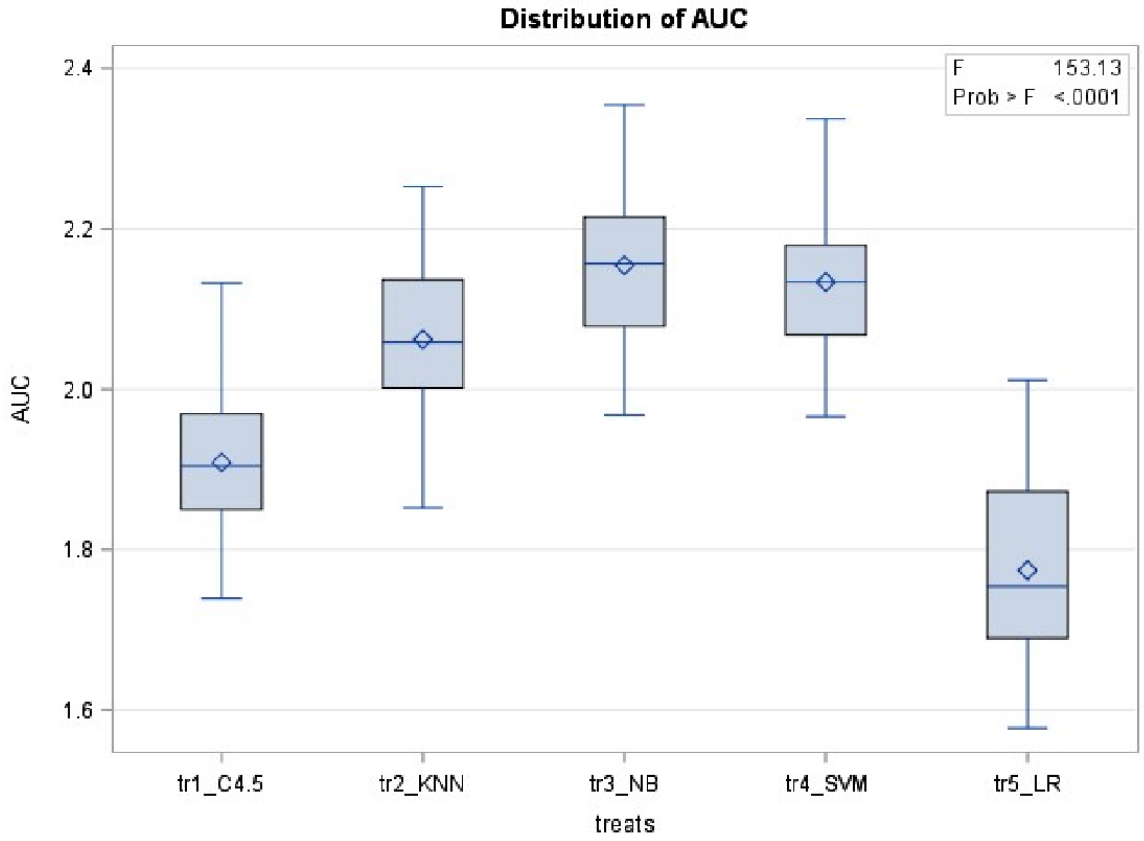


Figure 4.15 Box-plot of AUC for each treatment

Table 4.25 ANOVA_AUC

**Dependent Variable: AUC**

| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| Model | 4 | 5.23617207 | 1.30904302 | 153.13 | <.0001 |
| Error | 245 | 2.09441126 | 0.00854862 | | |
| Corrected Total | 249 | 7.33058333 | | | |

| R-Square | Coeff Var | Root MSE | AUC Mean |
|---|---|---|---|
| 0.714291 | 4.607961 | 0.092459 | 2.006500 |

| Source | DF | Type III SS | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| treats | 4 | 5.23617207 | 1.30904302 | 153.13 | <.0001 |

Table 4.26 Duncan's Multiple Range Test for AUC

| Alpha | 0.05 |
|---|---|
| Error Degrees of Freedom | 245 |
| Error Mean Square | 0.008549 |

| Number of Means | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| Critical Range | .03642 | .03834 | .03962 | .04057 |

| Means with the same letter are not significantly different. | | | |
|---|---|---|---|
| Duncan Grouping | Mean | N | treats |
| A | 2.15442 | 50 | tr3_NB |
| A | | | |
| A | 2.13358 | 50 | tr4_SVM |
| | | | |
| B | 2.06190 | 50 | tr2_KNN |
| | | | |
| C | 1.90856 | 50 | tr1_C4.5 |
| | | | |
| D | 1.77405 | 50 | tr5_LR |

#### 4.2.2.3 Summary

The aforementioned statistical analyses reinforce conclusions discussed in Chapter 4.2.1. The five (5) algorithms for different performance measures show statistically significantly differences. In general, when computing time is not considered, both SVM and NB are the best choices for our high-dimensional Twitter data, and achieve the highest accuracy, precision, recall and AUC. LR is more suitable for low-dimensional data, similarly with KNN.

The estimates of each algorithm effect for accuracy, precision, recall, and AUC are presented in Table 4.27 – Table 4.30, by setting effect of LR as the criterion. For all the models, every algorithm has a positive effect when compared to LR. For accuracy, precision and AUC, estimates based on SVM and NB suggest these two algorithms are considered to be a better choice when doing classification of SES of Twitter users. Only when talking about recall does KNN occupies NB's place but SVM is still the best choice. Thus, it can be concluded that both SVM and NB have a good performance on our data set while SVM is more stable and powerful.

Table 4.27 Parameter estimates for accuracy

| Parameter | Estimate | | Standard Error | t Value | Pr > \|t\| |
|---|---|---|---|---|---|
| Intercept | 1.506243323 | B | 0.01188151 | 126.77 | <.0001 |
| treats tr1_C4.5 | 0.218929649 | B | 0.01680300 | 13.03 | <.0001 |
| treats tr2_KNN | 0.219523426 | B | 0.01680300 | 13.06 | <.0001 |
| treats tr3_NB | 0.287689863 | B | 0.01680300 | 17.12 | <.0001 |
| treats tr4_SVM | 0.513631483 | B | 0.01680300 | 30.57 | <.0001 |
| treats tr5_LR | 0.000000000 | B | . | . | . |

Table 4.28 Parameter estimates for precision

| Parameter | Estimate | | Standard Error | t Value | Pr > \|t\| |
|---|---|---|---|---|---|
| Intercept | 1.821369575 | B | 0.01274587 | 142.90 | <.0001 |
| treats tr1_C4.5 | 0.103552936 | B | 0.01802538 | 5.74 | <.0001 |
| treats tr2_KNN | 0.052413527 | B | 0.01802538 | 2.91 | 0.0040 |
| treats tr3_NB | 0.312685774 | B | 0.01802538 | 17.35 | <.0001 |
| treats tr4_SVM | 0.247949670 | B | 0.01802538 | 13.76 | <.0001 |
| treats tr5_LR | 0.000000000 | B | . | . | . |

Table 4.29 Parameter estimates for recall

| Parameter | Estimate | | Standard Error | t Value | Pr > \|t\| |
|---|---|---|---|---|---|
| Intercept | 1.611582720 | B | 0.01859968 | 86.65 | <.0001 |
| treats tr1_C4.5 | 0.365600785 | B | 0.02630392 | 13.90 | <.0001 |
| treats tr2_KNN | 0.573551525 | B | 0.02630392 | 21.80 | <.0001 |
| treats tr3_NB | 0.327223129 | B | 0.02630392 | 12.44 | <.0001 |
| treats tr4_SVM | 0.852700873 | B | 0.02630392 | 32.42 | <.0001 |
| treats tr5_LR | 0.000000000 | B | . | . | . |

Table 4.30 Parameter estimates for AUC

| Parameter | Estimate | | Standard Error | t Value | Pr > \|t\| |
|---|---|---|---|---|---|
| Intercept | 1.774048083 | B | 0.01307564 | 135.68 | <.0001 |
| treats tr1_C4.5 | 0.134510877 | B | 0.01849175 | 7.27 | <.0001 |
| treats tr2_KNN | 0.287848310 | B | 0.01849175 | 15.57 | <.0001 |
| treats tr3_NB | 0.380367265 | B | 0.01849175 | 20.57 | <.0001 |
| treats tr4_SVM | 0.359532166 | B | 0.01849175 | 19.44 | <.0001 |
| treats tr5_LR | 0.000000000 | B | . | . | . |

### 4.2.3 Ensemble methods

We first repeat the same experiments for Bagging and Boosting, where the aforementioned five (5) single classifiers are treated as base classifiers. It is demonstrated that by using the ensemble methods (i.e. Bagging and Adaboost), the accuracy of the algorithms that have weaker performance (i.e. when LR and C4.5 are applied on high-dimensional data set in Table 4.31 and Table 4.35) are enhanced much more than the other classifiers that have better performances when applied separately. The accuracy of those stronger classifiers has a slight increase or even a slight decrease when using ensemble methods. However, the enhancements are not as evident with dimension reduction employed. Similar patterns for precision, recall, and AUC are displayed in Table 4.32-4.34, and 4.36-4.38.

Table 4.31 The Accuracy of Bagging compared with single classifiers.

Each cell contains accuracy value of Bagging/single classifier.

| Accuracy (%) | Bagging (LR) | Bagging (SVM) | Bagging (NB) | Bagging (KNN) | Bagging (C4.5) |
|---|---|---|---|---|---|
| Original data | 62.14/46.79 | 70.12/71.65 | 64.83/61.02 | 57.83/57.70 | 69.08/57.66 |
| PCA (0.9) | 64.82/59.82 | 68.48/67.48 | 44.49/43.63 | 54.99/52.92 | 60.13/50.05 |
| PCA (0.8) | 65.65/55.49 | 69.82/66.96 | 43.44/43.49 | 58.42/55.90 | 59.83/50.83 |
| PCA (0.7) | 63.27/59.97 | 68.18/66.74 | 44.48/44.30 | 59.76/56.29 | 61.70/50.75 |
| PCA (0.6) | 64.16/65.54 | 67.88/65.56 | 43.74/43.57 | 59.32/59.36 | 62.67/50.77 |
| PCA (0.5) | 67.44/68.07 | 66.02/64.71 | 44.26/44.90 | 58.86/61.86 | 62.36/51.23 |

Table 4.32 The Precision of Bagging compared with single classifiers.

Each cell contains precision value of Bagging/single classifier.

| Precision | Bagging (LR) | Bagging (SVM) | Bagging (NB) | Bagging (KNN) | Bagging (C4.5) |
|---|---|---|---|---|---|
| Original data | 0.72/0.62 | 0.72/0.74 | 0.77/0.77 | 0.67/0.65 | 0.70/0.67 |
| PCA (0.9) | 0.75/0.76 | 0.69/0.70 | 0.60/0.60 | 0.58/0.57 | 0.64/0.63 |
| PCA (0.8) | 0.78/0.74 | 0.72/0.72 | 0.60/0.61 | 0.67/0.63 | 0.64/0.63 |
| PCA (0.7) | 0.76/0.75 | 0.72/0.73 | 0.62/0.63 | 0.70/0.67 | 0.65/0.63 |
| PCA (0.6) | 0.77/0.77 | 0.72/0.73 | 0.64/0.65 | 0.71/0.69 | 0.66/0.63 |
| PCA (0.5) | 0.78/0.77 | 0.73/0.74 | 0.67/0.68 | 0.72/0.72 | 0.66/0.64 |

Table 4.33 The Recall of Bagging compared with single classifiers.

Each cell contains recall value of Bagging/single classifier.

| Recall | Bagging (LR) | Bagging (SVM) | Bagging (NB) | Bagging (KNN) | Bagging (C4.5) |
|---|---|---|---|---|---|
| Original data | 0.74/0.52 | 0.88/0.89 | 0.73/0.68 | 0.76/0.79 | 0.90/0.70 |
| PCA (0.9) | 0.77/0.67 | 0.91/0.87 | 0.54/0.51 | 0.85/0.83 | 0.86/0.62 |
| PCA (0.8) | 0.75/0.61 | 0.90/0.84 | 0.51/0.50 | 0.76/0.77 | 0.86/0.64 |
| PCA (0.7) | 0.72/0.66 | 0.86/0.82 | 0.50/0.49 | 0.71/0.73 | 0.87/0.64 |
| PCA (0.6) | 0.73/0.75 | 0.85/0.80 | 0.47/0.46 | 0.69/0.75 | 0.87/0.64 |
| PCA (0.5) | 0.78/0.78 | 0.81/0.77 | 0.46/0.46 | 0.69/0.76 | 0.85/0.64 |

Table 4.34 The AUC of Bagging compared with single classifiers.

Each cell contains AUC value of Bagging/single classifier.

| AUC | Bagging (LR) | Bagging (SVM) | Bagging (NB) | Bagging (KNN) | Bagging (C4.5) |
|---|---|---|---|---|---|
| Original data | 0.77/0.60 | 0.83/0.77 | 0.81/0.77 | 0.73/0.73 | 0.83/0.66 |
| PCA (0.9) | 0.81/0.77 | 0.81/0.73 | 0.62/0.61 | 0.68/0.70 | 0.76/0.60 |
| PCA (0.8) | 0.83/0.73 | 0.83/0.73 | 0.63/0.61 | 0.73/0.70 | 0.75/0.61 |
| PCA (0.7) | 0.82/0.77 | 0.83/0.74 | 0.64/0.62 | 0.76/0.74 | 0.76/0.61 |
| PCA (0.6) | 0.83/0.83 | 0.83/0.74 | 0.65/0.63 | 0.77/0.76 | 0.77/0.61 |
| PCA (0.5) | 0.85/0.84 | 0.83/0.74 | 0.66/0.65 | 0.78/0.79 | 0.78/0.61 |

Table 4.35 The Accuracy of Boosting compared with single classifiers.

Each cell contains accuracy value of Boosting/single classifier.

| Accuracy (%) | Boosting (LR) | Boosting (SVM) | Boosting (NB) | Boosting (KNN) | Boosting (C4.5) |
|---|---|---|---|---|---|
| Original data | 45.74/46.79 | 69.37/71.65 | 61.77/61.02 | 54.40/57.70 | 65.12/57.66 |
| PCA (0.9) | 60.13/59.82 | 66.98/67.48 | 54.03/43.63 | 50.67/52.92 | 58.94/50.05 |
| PCA (0.8) | 57.00/55.49 | 66.69/66.96 | 50.53/43.49 | 49.71/55.90 | 58.49/50.83 |
| PCA (0.7) | 59.02/59.97 | 67.14/66.74 | 45.16/44.30 | 53.88/56.29 | 57.82/50.75 |
| PCA (0.6) | 66.84/65.54 | 64.16/65.56 | 45.60/43.57 | 51.64/59.36 | 60.13/50.77 |
| PCA (0.5) | 68.71/68.07 | 64.90/64.71 | 45.91/44.90 | 54.33/61.86 | 60.81/51.23 |

Table 4.36 The Precision of Boosting compared with single classifiers.

Each cell contains precision value of Boosting/single classifier.

| Precision | Boosting (LR) | Boosting (SVM) | Boosting (NB) | Boosting (KNN) | Boosting (C4.5) |
|---|---|---|---|---|---|
| Original data | 0.63/0.62 | 0.81/0.74 | 0.78/0.77 | 0.66/0.65 | 0.69/0.67 |
| PCA (0.9) | 0.75/0.76 | 0.74/0.70 | 0.64/0.60 | 0.61/0.57 | 0.66/0.63 |
| PCA (0.8) | 0.74/0.74 | 0.75/0.72 | 0.63/0.61 | 0.65/0.63 | 0.65/0.63 |
| PCA (0.7) | 0.75/0.75 | 0.75/0.73 | 0.61/0.63 | 0.71/0.67 | 0.65/0.63 |
| PCA (0.6) | 0.79/0.77 | 0.73/0.73 | 0.65/0.65 | 0.70/0.69 | 0.66/0.63 |
| PCA (0.5) | 0.78/0.77 | 0.74/0.74 | 0.68/0.68 | 0.70/0.72 | 0.67/0.64 |

Table 4.37 The Recall of Boosting compared with single classifiers.

Each cell contains recall value of Boosting/single classifier.

| Recall | Boosting (LR) | Boosting (SVM) | Boosting (NB) | Boosting (KNN) | Boosting (C4.5) |
|---|---|---|---|---|---|
| Original data | 0.51/0.52 | 0.78/0.89 | 0.68/0.68 | 0.68/0.79 | 0.82/0.70 |
| PCA (0.9) | 0.68/0.67 | 0.82/0.87 | 0.69/0.51 | 0.66/0.83 | 0.79/0.62 |
| PCA (0.8) | 0.64/0.61 | 0.81/0.84 | 0.61/0.50 | 0.58/0.77 | 0.80/0.64 |
| PCA (0.7) | 0.65/0.66 | 0.81/0.82 | 0.55/0.49 | 0.59/0.73 | 0.78/0.64 |
| PCA (0.6) | 0.76/0.75 | 0.77/0.80 | 0.50/0.46 | 0.57/0.75 | 0.81/0.64 |
| PCA (0.5) | 0.79/0.78 | 0.77/0.77 | 0.47/0.46 | 0.63/0.76 | 0.81/0.64 |

Table 4.38 The AUC of Boosting compared with single classifiers.

Each cell contains AUC value of Boosting/single classifier

| AUC | Boosting (LR) | Boosting (SVM) | Boosting (NB) | Boosting (KNN) | Boosting (C4.5) |
|---|---|---|---|---|---|
| Original data | 0.60/0.60 | 0.87/0.77 | 0.76/0.77 | 0.66/0.73 | 0.79/0.66 |
| PCA (0.9) | 0.77/0.77 | 0.81/0.73 | 0.67/0.61 | 0.62/0.70 | 0.73/0.60 |
| PCA (0.8) | 0.73/0.73 | 0.81/0.73 | 0.64/0.61 | 0.66/0.70 | 0.73/0.61 |
| PCA (0.7) | 0.77/0.77 | 0.80/0.74 | 0.61/0.62 | 0.70/0.74 | 0.73/0.61 |
| PCA (0.6) | 0.79/0.83 | 0.80/0.74 | 0.63/0.63 | 0.70/0.76 | 0.74/0.61 |
| PCA (0.5) | 0.78/0.84 | 0.81/0.74 | 0.64/0.65 | 0.72/0.79 | 0.75/0.61 |

#### 4.2.3.1 MANOVA

We employ MANOVA to measure differences in the four (4) performance measures (accuracy, precision, recall, AUC) with a total of 50 observations generated by applying the five (5) ensemble algorithms (LR, SVM, NB, KNN, C4.5). For each ensemble algorithm (treatment), 10-fold cross-validation is used for generating 10 observations for each performance measure (response variable), as shown in Table 4.39, along with means, standard deviations and the values of minimum and maximum.

Table 4.39 Means procedure of response variables for each treatment (Bagging)

| treats | N Obs | Variable | N | Mean | Std Dev | Minimum | Maximum |
|---|---|---|---|---|---|---|---|
| Bg1_C4.5 | 10 | accuracy | 10 | 1.9633990 | 0.0752848 | 1.8583238 | 2.1204454 |
| | | precision | 10 | 1.9733871 | 0.0707606 | 1.8653596 | 2.1135361 |
| | | recall | 10 | 2.5047674 | 0.1357340 | 2.3324788 | 2.8043240 |
| | | AUC | 10 | 2.2999378 | 0.0736077 | 2.1869642 | 2.4450206 |
| Bg2_KNN | 10 | accuracy | 10 | 1.7285625 | 0.0842148 | 1.5930204 | 1.8427945 |
| | | precision | 10 | 1.9261302 | 0.0983125 | 1.7770761 | 2.1367630 |
| | | recall | 10 | 2.1200476 | 0.1617587 | 1.9608975 | 2.4135496 |
| | | AUC | 10 | 2.0575866 | 0.0936897 | 1.9325062 | 2.1921791 |
| Bg3_NB | 10 | accuracy | 10 | 1.8732210 | 0.0983173 | 1.7055320 | 2.0184325 |
| | | precision | 10 | 2.1508784 | 0.0759845 | 2.0005710 | 2.2644412 |
| | | recall | 10 | 2.0514783 | 0.1304974 | 1.7549331 | 2.2213587 |
| | | AUC | 10 | 2.2476170 | 0.0795257 | 2.1077422 | 2.3306288 |
| Bg4_SVM | 10 | accuracy | 10 | 1.9860560 | 0.0741694 | 1.8739248 | 2.1030339 |
| | | precision | 10 | 2.0278410 | 0.0875816 | 1.9031999 | 2.1531606 |
| | | recall | 10 | 2.4297170 | 0.0807484 | 2.2942430 | 2.5516741 |
| | | AUC | 10 | 2.3037273 | 0.0693495 | 2.1782250 | 2.4055244 |
| Bg5_LR | 10 | accuracy | 10 | 1.8168828 | 0.0842212 | 1.6454926 | 1.9371452 |
| | | precision | 10 | 2.0304662 | 0.0653940 | 1.9006917 | 2.1344119 |
| | | recall | 10 | 2.0796448 | 0.1112556 | 1.9006917 | 2.2213587 |
| | | AUC | 10 | 2.1509005 | 0.0970549 | 1.9177558 | 2.2394651 |

The MANOVA output is displayed in the following tables. The first sub-table in Table 4.40 presents the SSCP matrix for the ensemble algorithms. The rows of second sub-table describe the characteristic vector of each root. The first and second roots account for almost all the variability in the effect of ensemble algorithms (69.39% and 29.36% respectively). The next part of this table provides the four (4) test statistics of the ensemble algorithm effect. For Wilks' Lambda, we obtain an F value of 10.01 with a p-value <0.0001 which means that ensemble algorithms do not have the same effect for all performance measures; other three test statistics that offer the same proof of significant difference.

Table 4.40 The GLM Procedure MANOVA (Bagging)

| H = Type III SSCP Matrix for treats | | | | |
|---|---|---|---|---|
| | accuracy | precision | recall | AUC |
| accuracy | 0.449630937 | 0.0966719627 | 0.7167498233 | 0.4405965817 |
| precision | 0.0966719627 | 0.2826933241 | -0.259208478 | 0.1513737199 |
| recall | 0.7167498233 | -0.259208478 | 1.8169598828 | 0.6228994733 |
| AUC | 0.4405965817 | 0.1513737199 | 0.6228994733 | 0.4499216227 |

| Characteristic Roots and Vectors of: E Inverse * H, where H = Type III SSCP Matrix for treats E = Error SSCP Matrix | | | | | |
|---|---|---|---|---|---|
| | | Characteristic Vector V'EV=1 | | | |
| Characteristic Root | Percent | accuracy | precision | recall | AUC |
| 3.48200962 | 69.39 | 0.64624193 | -1.49351640 | 0.77408803 | 0.35812089 |
| 1.47344818 | 29.36 | 1.28987628 | -0.46298805 | -1.06851864 | 1.86293412 |
| 0.05368556 | 1.07 | -3.03174059 | 1.12193697 | 0.88392139 | 1.39543931 |
| 0.00873460 | 0.17 | -0.11800508 | 2.42495867 | 1.20617144 | -2.36819546 |

| MANOVA Test Criteria and F Approximations for the Hypothesis of No Overall treats Effect H = Type III SSCP Matrix for treats E = Error SSCP Matrix S=4 M=-0.5 N=20 | | | | | |
|---|---|---|---|---|---|
| Statistic | Value | F Value | Num DF | Den DF | Pr > F |
| Wilks' Lambda | 0.08486653 | 10.01 | 16 | 128.95 | <.0001 |
| Pillai's Trace | 1.43220113 | 6.27 | 16 | 180 | <.0001 |
| Hotelling-Lawley Trace | 5.01787796 | 12.87 | 16 | 78.19 | <.0001 |
| Roy's Greatest Root | 3.48200962 | 39.17 | 4 | 45 | <.0001 |
| NOTE: F Statistic for Roy's Greatest Root is an upper bound. | | | | | |

#### 4.2.3.2 ANOVA

We now consider the univariate analyses for each of the performance measures (dependent variables). From the Box-plots in Figure 4.16 – 4.19, we find that the methods of Bagging based upon five (5) algorithms perform differently for each of the response variables. More accurately, the p-values of the four (4) tests presented in Table 4.41, 4.43, 4.45, 4.47 that are $< 0.0001$ demonstrate that these five (5) classifiers have a significantly different impact on each of these performance measures.

Duncan's test (Table 4.42) shows that the means for accuracy of Bagging based on C4.5 and SVM are not significantly different; this also happens for NB and LR. When considering precision (Table 4.44), Bagging based on NB is the best while there is no obvious difference among LR, SVM, and C4.5; the same is found when comparing C4.5 with KNN. As shown in Table 4.46, ensembles with C4.5 and SVM are not significantly different and the other three have a similar effect as well. Finally, for AUC, SVM, C4.5, NB have the best performance and no demonstrated difference (Table 4.48).

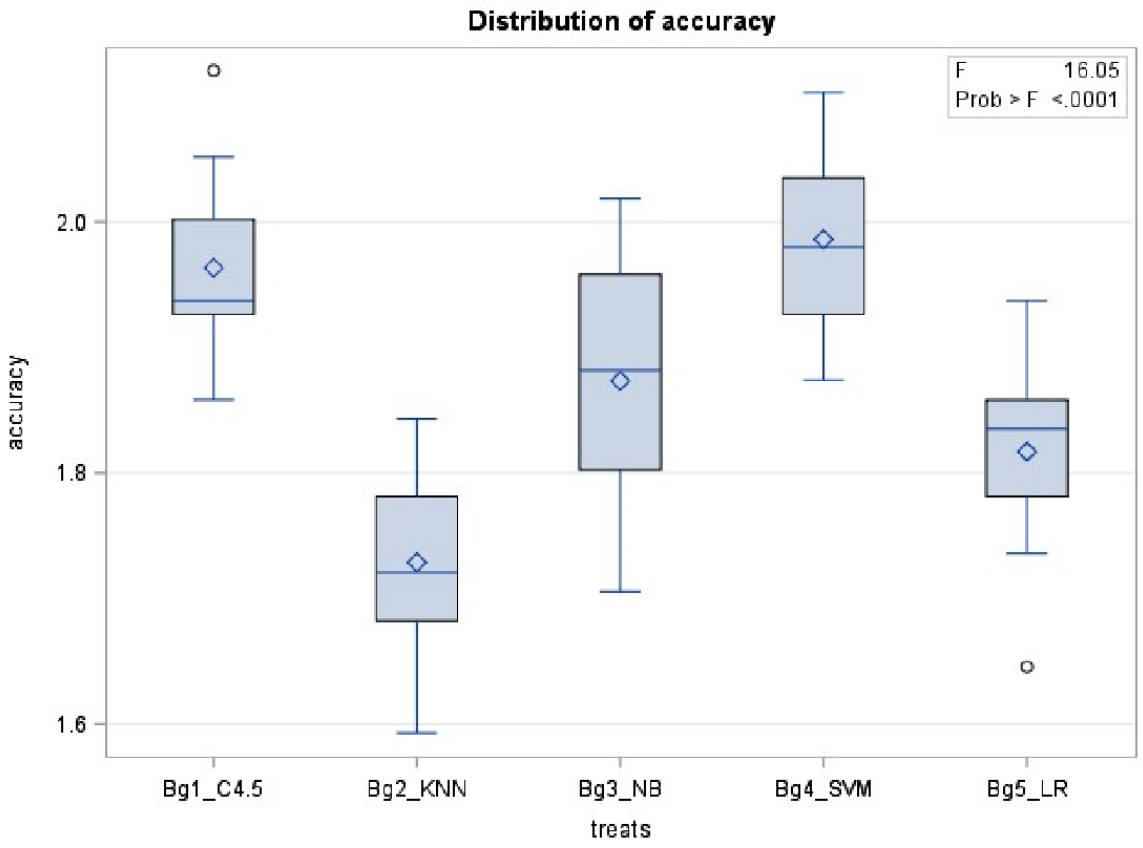


Figure 4.16 Box-plot of accuracy for each treatment (Bagging)

Table 4.41 ANOVA_accuracy (Bagging)

**Dependent Variable: accuracy**

| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| Model | 4 | 0.44963094 | 0.11240773 | 16.05 | <.0001 |
| Error | 45 | 0.31518483 | 0.00700411 | | |
| Corrected Total | 49 | 0.76481577 | | | |

| R-Square | Coeff Var | Root MSE | accuracy Mean |
|---|---|---|---|
| 0.587894 | 4.466773 | 0.083691 | 1.873624 |

| Source | DF | Type III SS | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| treats | 4 | 0.44963094 | 0.11240773 | 16.05 | <.0001 |

Table 4.42 Duncan's Multiple Range Test for accuracy (Bagging)

| | |
|---|---|
| **Alpha** | 0.05 |
| **Error Degrees of Freedom** | 45 |
| **Error Mean Square** | 0.007004 |

| **Number of Means** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|
| **Critical Range** | .07538 | .07928 | .08183 | .08367 |

| **Means with the same letter are not significantly different.** | | | |
|---|---|---|---|
| **Duncan Grouping** | **Mean** | **N** | **treats** |
| A | 1.98606 | 10 | Bg4_SVM |
| A | | | |
| A | 1.96340 | 10 | Bg1_C4.5 |
| | | | |
| B | 1.87322 | 10 | Bg3_NB |
| B | | | |
| B | 1.81688 | 10 | Bg5_LR |
| | | | |
| C | 1.72856 | 10 | Bg2_KNN |

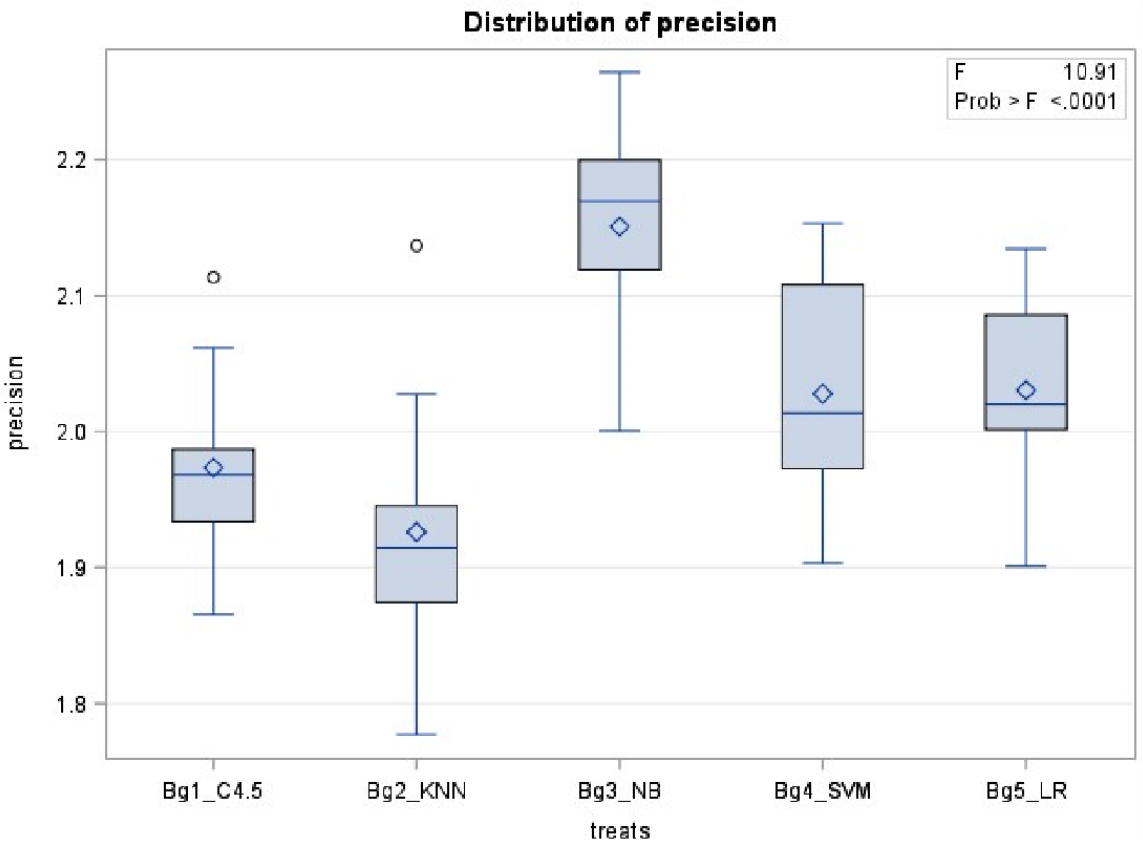


Figure 4.17 Box-plot of precision for each treatment (Bagging)

Table 4.43 ANOVA_precision (Bagging)

**Dependent Variable: precision**

| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| Model | 4 | 0.28269332 | 0.07067333 | 10.91 | <.0001 |
| Error | 45 | 0.29153661 | 0.00647859 | | |
| Corrected Total | 49 | 0.57422993 | | | |

| R-Square | Coeff Var | Root MSE | precision Mean |
|---|---|---|---|
| 0.492300 | 3.981208 | 0.080490 | 2.021741 |

| Source | DF | Type III SS | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| treats | 4 | 0.28269332 | 0.07067333 | 10.91 | <.0001 |

Table 4.44 Duncan's Multiple Range Test for precision (Bagging)

| | |
|---|---|
| **Alpha** | 0.05 |
| **Error Degrees of Freedom** | 45 |
| **Error Mean Square** | 0.006479 |

| **Number of Means** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|
| **Critical Range** | .07250 | .07624 | .07870 | .08047 |

| **Means with the same letter are not significantly different.** | | | | |
|---|---|---|---|---|
| **Duncan Grouping** | | **Mean** | **N** | **treats** |
| | A | 2.15088 | 10 | Bg3_NB |
| | | | | |
| | B | 2.03047 | 10 | Bg5_LR |
| | B | | | |
| | B | 2.02784 | 10 | Bg4_SVM |
| | B | | | |
| C | B | 1.97339 | 10 | Bg1_C4.5 |
| C | | | | |
| C | | 1.92613 | 10 | Bg2_KNN |

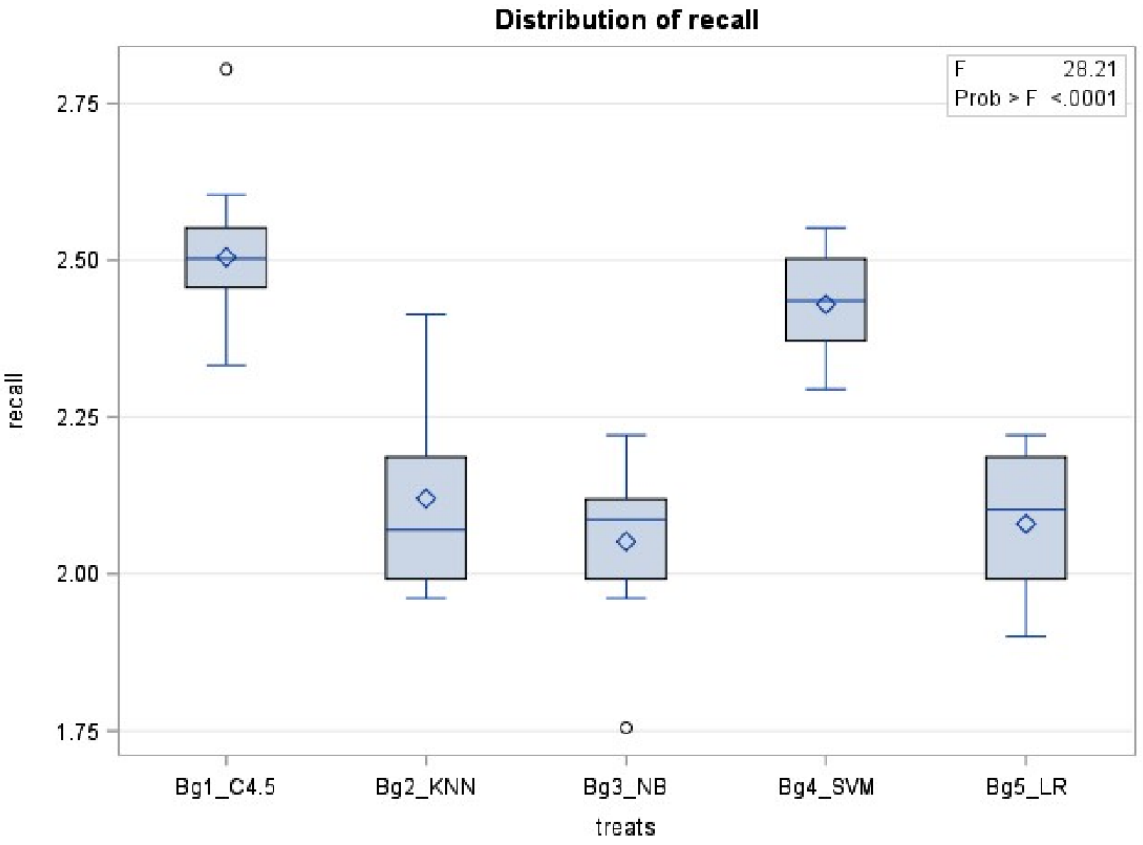


Figure 4.18 Box-plot of recall for each treatment (Bagging)

Table 4.45 ANOVA_recall (Bagging)

**Dependent Variable: recall**

| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| Model | 4 | 1.81695988 | 0.45423997 | 28.21 | <.0001 |
| Error | 45 | 0.72465567 | 0.01610346 | | |
| Corrected Total | 49 | 2.54161555 | | | |

| R-Square | Coeff Var | Root MSE | recall Mean |
|---|---|---|---|
| 0.714884 | 5.672417 | 0.126899 | 2.237131 |

| Source | DF | Type III SS | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| treats | 4 | 1.81695988 | 0.45423997 | 28.21 | <.0001 |

Table 4.46 Duncan's Multiple Range Test for recall (Bagging)

| | |
|---|---|
| **Alpha** | 0.05 |
| **Error Degrees of Freedom** | 45 |
| **Error Mean Square** | 0.016103 |

| **Number of Means** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|
| **Critical Range** | .1143 | .1202 | .1241 | .1269 |

| **Means with the same letter are not significantly different.** | | | |
|---|---|---|---|
| **Duncan Grouping** | **Mean** | **N** | **treats** |
| A | 2.50477 | 10 | Bg1_C4.5 |
| A | | | |
| A | 2.42972 | 10 | Bg4_SVM |
| | | | |
| B | 2.12005 | 10 | Bg2_KNN |
| B | | | |
| B | 2.07964 | 10 | Bg5_LR |
| B | | | |
| B | 2.05148 | 10 | Bg3_NB |

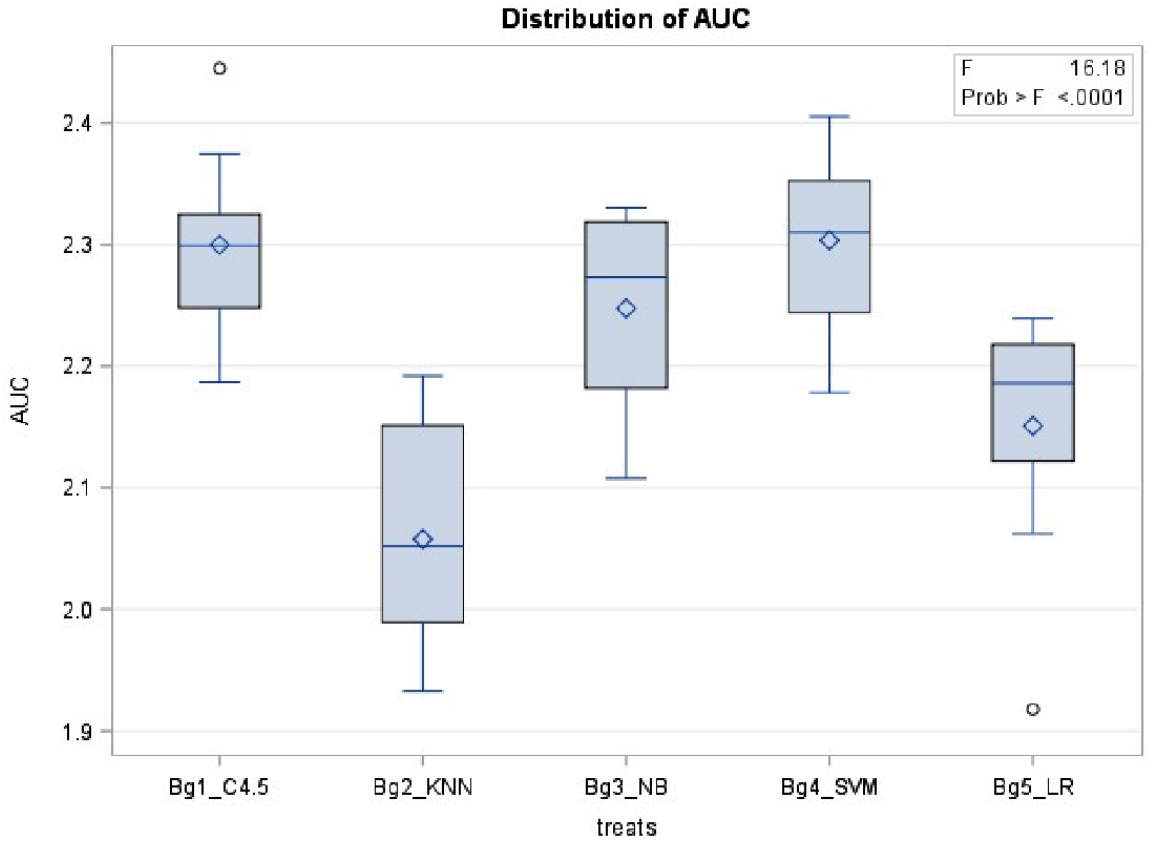


Figure 4.19 Box-plot of AUC for each treatment (Bagging)

Table 4.47 ANOVA_AUC (Bagging)

**Dependent Variable: AUC**

| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| Model | 4 | 0.44992162 | 0.11248041 | 16.18 | <.0001 |
| Error | 45 | 0.31274280 | 0.00694984 | | |
| Corrected Total | 49 | 0.76266442 | | | |

| R-Square | Coeff Var | Root MSE | AUC Mean |
|---|---|---|---|
| 0.589934 | 3.768872 | 0.083366 | 2.211954 |

| Source | DF | Type III SS | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| treats | 4 | 0.44992162 | 0.11248041 | 16.18 | <.0001 |

Table 4.48 Duncan's Multiple Range Test for AUC (Bagging)

| Alpha | 0.05 |
|---|---|
| Error Degrees of Freedom | 45 |
| Error Mean Square | 0.00695 |

| Number of Means | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| Critical Range | .07509 | .07897 | .08151 | .08335 |

| Means with the same letter are not significantly different. | | | |
|---|---|---|---|
| Duncan Grouping | Mean | N | treats |
| A | 2.30373 | 10 | Bg4_SVM |
| A | | | |
| A | 2.29994 | 10 | Bg1_C4.5 |
| A | | | |
| A | 2.24762 | 10 | Bg3_NB |
| | | | |
| B | 2.15090 | 10 | Bg5_LR |
| | | | |
| C | 2.05759 | 10 | Bg2_KNN |

#### 4.2.3.3 Summary

To sum up, we have confirmed statistically that, for different performance measures, Bagging based on the five (5) algorithms does not give uniform results. As can be seen in Table 4.49 – Table 4.52, unlike with the analyses of single classifiers in Chapter 4.2.2, these ensemble classifiers are not all significantly different when compared to the baseline ensemble LR (Bagging).

More specifically, when compared to ensemble LR, for accuracy and AUC (Table 4.49 and Table 4.52), Bagging KNN has a negative effect while the other three (Bagging C4.5, Bagging NB, Bagging SVM) all give a positive impact. Among results for accuracy and recall, only Bagging with SVM and C4.5 are significant with profound effects, see Table 4.49 and Table 4.51. On the contrary, as seen in Table 4.50, for precision only Bagging NB shows a significant and positive effect compared to Bagging LR. Therefore, without considering precision, Bagging SVM and Bagging C4.5 have a good performance on our Twitter data and that of Bagging C4.5 has a huge increase in ensemble learning.

Table 4.49 Parameter estimates for accuracy (Bagging)

| Parameter | Estimate | | Standard Error | t Value | Pr > \|t\| |
|---|---|---|---|---|---|
| Intercept | 1.816882759 | B | 0.02646527 | 68.65 | <.0001 |
| treats Bg1_C4.5 | 0.146516290 | B | 0.03742755 | 3.91 | 0.0003 |
| treats Bg2_KNN | -0.088320289 | B | 0.03742755 | -2.36 | 0.0227 |
| treats Bg3_NB | 0.056338266 | B | 0.03742755 | 1.51 | 0.1392 |
| treats Bg4_SVM | 0.169173247 | B | 0.03742755 | 4.52 | <.0001 |
| treats Bg5_LR | 0.000000000 | B | . | . | . |

Table 4.50 Parameter estimates for precision (Bagging)

| Parameter | Estimate | | Standard Error | t Value | Pr > \|t\| |
|---|---|---|---|---|---|
| Intercept | 2.030466174 | B | 0.02545308 | 79.77 | <.0001 |
| treats Bg1_C4.5 | -0.057079042 | B | 0.03599609 | -1.59 | 0.1198 |
| treats Bg2_KNN | -0.104335943 | B | 0.03599609 | -2.90 | 0.0058 |
| treats Bg3_NB | 0.120412263 | B | 0.03599609 | 3.35 | 0.0017 |
| treats Bg4_SVM | -0.002625140 | B | 0.03599609 | -0.07 | 0.9422 |
| treats Bg5_LR | 0.000000000 | B | . | . | . |

Table 4.51 Parameter estimates for recall (Bagging)

| Parameter | Estimate | | Standard Error | t Value | Pr > \|t\| |
|---|---|---|---|---|---|
| Intercept | 2.079644828 | B | 0.04012912 | 51.82 | <.0001 |
| treats Bg1_C4.5 | 0.425122612 | B | 0.05675114 | 7.49 | <.0001 |
| treats Bg2_KNN | 0.040402746 | B | 0.05675114 | 0.71 | 0.4802 |
| treats Bg3_NB | -0.028166530 | B | 0.05675114 | -0.50 | 0.6221 |
| treats Bg4_SVM | 0.350072193 | B | 0.05675114 | 6.17 | <.0001 |
| treats Bg5_LR | 0.000000000 | B | . | . | . |

Table 4.52 Parameter estimates for AUC (Bagging)

| Parameter | Estimate | | Standard Error | t Value | Pr > \|t\| |
|---|---|---|---|---|---|
| Intercept | 2.150900460 | B | 0.02636255 | 81.59 | <.0001 |
| treats Bg1_C4.5 | 0.149037351 | B | 0.03728227 | 4.00 | 0.0002 |
| treats Bg2_KNN | -0.093313837 | B | 0.03728227 | -2.50 | 0.0160 |
| treats Bg3_NB | 0.096716540 | B | 0.03728227 | 2.59 | 0.0128 |
| treats Bg4_SVM | 0.152826847 | B | 0.03728227 | 4.10 | 0.0002 |
| treats Bg5_LR | 0.000000000 | B | . | . | . |

# 5 Conclusion and future work

In this thesis, we focus on predicting OSN users' SES based on the data generated from Twitter. To start with, we tailor the parameters of a series of well-known statistical machine learning approaches for the classification task, then compare and evaluate these algorithms with regards to four (4) performance measures: accuracy, precision, recall, and AUC, as one of the main points of our study.

Based on this empirical study, we have obtained comprehensive results demonstrating that for our high-dimensional Twitter data set: SVM can be seen as one of the best choices as a classifier, as it is robust and performs significantly better than other classifiers in almost all the cases. In addition, NB, as another choice for our data set, can produce similar results to SVM. Moreover, it has a good balance between performance and computing time. We employed ensemble methods with basic single classifiers to improve the performance of weaker classifiers and found a notable enhancement, thereby suggesting ensemble C4.5 as an optimal choice of ensemble method. In addition, these results might provide guidance if we need to analyze similar data (i.e. ONS data) for SES.

This study has drawn a comparison of different classification algorithms applied to OSN data. To evaluate the performance of these algorithms on OSN data, a one-way MANOVA is run on the original data set. However, based on the results from dimension

reduction, taking reduction of dimension into account (i.e. each block is one set of data – original or reduced-dimensional) and applying a two-way MANOVA will be undertaken to provide a more comprehensive assessment of the learning methods on OSN data.

## References


Afifi, A. (2013). Improving the classification accuracy using support vector machines (SVMS) with new kernel. *Journal of global research in computer science, 4*(2), 1-7.

Amari, S. I., & Wu, S. (1999). Improving support vector machine classifiers by modifying kernel functions. *Neural Networks, 12*(6), 783-789.

Asur, S., & Huberman, B. A. (2010, August). Predicting the future with social media. *In Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on* (Vol. 1, pp. 492-499). IEEE.

Barber, D. (2012) Bayesian Reasoning and Machine Learning, *Cambridge University Press 1st edition.*

Benevenuto, Fabr, Rodrigues, Tiago, Cha, & Meeyoung, et al. (2009). Characterizing user behavior in online social networks.

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition, 30*(7), 1145-1159.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Breiman, L. (1996). Bagging predictors. *Machine learning, 24*(2), 123-140.

Burbidge, R., & Buxton, B. (2001). An introduction to support vector machines for data mining. *Keynote papers, young OR12*, 3-15.

Cover, T. M. (1968). Rates of convergence for nearest neighbor procedures. *Hawaii International Conference on System Sciences*.Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, 215-242.

Elias, P., & Birch, M. (2010). SOC2010: revision of the Standard Occupational Classification. *Economic & Labour Market Review, 4*(7), 48.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters, 27*(8), 861-874.

Flach, P. (2012). *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press

Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence, 14*(771-780), 1612.

Hastie, T. and R. Tibshirani, and J.H.Friedman (2001), The Elements of Statistical Learning: Data Mining, Inference, and Prediction, *New York: Springer-Verlag*

Jiang, J., Wilson, C., Wang, X., Sha, W., Huang, P., Dai, Y., & Zhao, B. Y. (2013). Understanding latent interactions in online social networks. *ACM Transactions on the Web (TWEB), 7*(4), 18.

Kwak, H., Lee, C., Park, H., & Moon, S. (2010, April). What is Twitter, a social network or a news media?. In *Proceedings of the 19th international conference on World wide web* (pp. 591-600). ACM.

Lampos, V., Aletras, N., Preoţiuc-Pietro, D., & Cohn, T. (2014, January). Predicting and characterising user impact on Twitter. In *14th Conference of the European Chapter of the Association for Computational Linguistics 2014, EACL 2014* (pp. 405-413).

Lampos, V.; Aletras, N.; Geyti, J.; Zou, B.; Cox, I. (2015): Socioeconomic status classification of social media users. figshare. https://doi.org/10.6084/m9.figshare.1619703.v2

Lampos, V., Aletras, N., Geyti, J. K., Zou, B., & Cox, I. J. (2016, March). Inferring the socioeconomic status of social media users based on behaviour and language. In *European Conference on Information Retrieval* (pp. 689-695). Springer International Publishing.

Meo, P. D., Ferrara, E., Abel, F., Aroyo, L., & Houben, G. J. (2013). Analyzing user behavior across social sharing environments. *Computer Science*, *5*(1), 1-31.

Michell, T.M. (1997) Machine Learning*, McGraw Hill, 1st edition*.

Olshen, L. B. J. F. R., & Stone, C. J. (1984). Classification and regression trees. *Wadsworth International Group*, *93*(99), 101.

Preoţiuc-Pietro, D., Lampos, V., & Aletras, N. (2015). An analysis of the user occupational class through Twitter content. The Association for Computational Linguistics.

Quinlan, J. (1993). C4. 5: Programs for Machine Learning. C4. 5-programs for machine learning/J. Ross Quinlan.

Rätsch, G. (2004). A brief introduction into machine learning*. Friedrich Miescher Laboratory of the Max Planck Society*.

Schneider, F., Feldmann, A., Krishnamurthy, B., & Willinger, W. (2009). Understanding online social network usage from a network perspective. *ACM SIGCOMM Conference on Internet Measurement 2009, Chicago, Illinois, Usa, November* (Vol.106, pp.35-48). DBLP.

Scholkopf, B., & Smola, A. J. (2001). Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. *MIT Press*.

Szabo, G., & Huberman, B. A. (2008). Predicting the popularity of online content. *Communications of the Acm, 53*(8), 80--88.

Therneau, T. M., & Atkinson, E. J. (1997). An introduction to recursive partitioning using the rpart routines (Tech. Rep. No. 61). *Scottsdale, AZ: Mayo Foundation*.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing, 17*(4), 395-416.

Weng, J., Lim, E. P., Jiang, J., & He, Q. (2010, February). Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining* (pp. 261-270). ACM.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., ... & Zhou, Z. H. (2008). Top 10 algorithms in data mining. *Knowledge and information systems*, *14*(1), 1-37.

## Bibliography

Huberty, C. J., & Olejnik, S. (2006). *Applied MANOVA and discriminant analysis* (Vol. 498). John Wiley & Sons.

Johnson, R. A., & Wichern, D. W. (2002). *Applied multivariate statistical analysis* (Vol. 5, No. 8). Upper Saddle River, NJ: Prentice hall.