STAT995 RESEARCH PROJECT

COMPARISION OF CLASSIFCATION METHODS ON A RANGE OF DATASETS

ANVAY AJIT KARAMBELKAR - 18012461
AUCKLAND UNIVERSITY OF TECHNOLOGY

1 Contents

2 Introduction

In statistics and machine learning, classification is a supervised learning technique, that predicts the instances of new data by learning the training data whose classes are already known. Few examples of classification problems are face recognition, speech recognition, spam filtering, text classification etc. All the instances of the data that is trained using classification methods are corresponding to the same set of attributes or features. The datasets may contain either two classes or more than two, which can be also called binary class and multiclass classification problem. Thus, there exist many classification methods that are suitable for binary as well as a multiclass classification problem. The performance of the classification algorithm varies based on the characteristics and limitation of the dataset. However, there is no single method that works well for all types of problems. But there are several techniques to evaluate the usefulness of the classification algorithm. In machine learning, the purpose of the evaluation methods is to figure out the effectiveness of the classification algorithms (Japkowicz, 2006).

This research paper demonstrates the end-to-end application of machine learning on classification task on various datasets. The aim of this research project is to rank the classification methods with respect to evaluation methods on a list of datasets. Furthermore, to determine the simplicity and complexity of the application of the implemented classification algorithms and evaluation techniques.

3 Literature review

This section gives a brief explanation from literature of some classification and evaluation methods used in the study.

3.1 Classification methods

3.1.1 Logistic Regression (LR)

In statistics, logistic regression uses a logistic function to train the model that has a binary dependent variable. The logistic function was first introduced by the mathematicians to study the growth of population in social science (Cramer, 2002). In predictive analytics, the logistic regression estimates the parameters of the binomial logistic model. The function is represented by the S-shaped curve that can take any number between the range 0 and 1 as shown in Fig. 1. The logistic regression equation can be given as,

$$P(y = 1) = \frac{e^{(b0+b1*x)}}{1 + e^{(b0+b1*x)}}$$

Where, y = dependent variable

x = independent variable

$b0$ = intercept

$b1$= coefficient

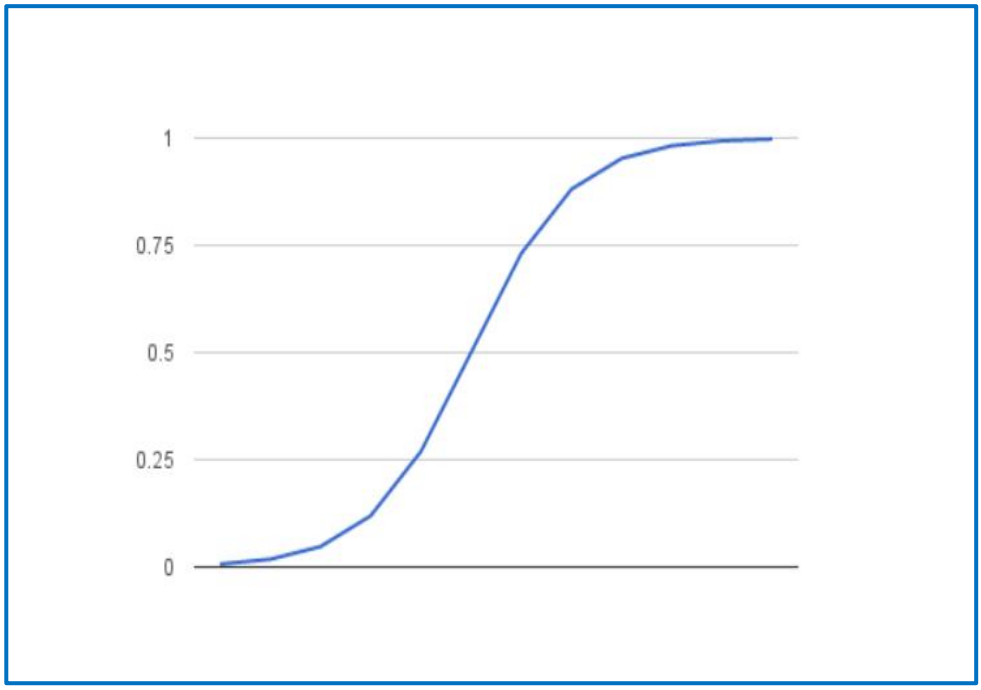

Fig. 1: Logistic model

The logistic model is much like the linear model. The major difference is that the response variable takes only two values (labelled in the form of classes) rather than numeric (James, 2013). The logistic regression algorithm estimates the values of intercept $(b0)$ and coefficient $(b1)$ from the training dataset. It applies the principles of maximum-likelihood estimation. Once $b0$ and $b1$ for all the predictors are computed then the classes are predicted by calculating the value of y. We get the value of y by substituting $b0$ and $b1$ in logistic regression equation. The threshold value for the separation of classes by default is 0.5. If value of y is less than 0.5 then it belongs to class 0 and if the value is greater than 0.5 then it belongs to class 1.

3.1.2 Linear Discriminant analysis (LDA)

In statistics, LDA is a classification method that was developed by R.A Fisher in 1936. The method discriminates the classes by identifying the linear combination of features. In machine learning, Linear Discriminant Analysis is the dimensionality reduction technique that can be implemented before training the classification model. Originally,

the method was described for only binary classes, later it was formulated for multiple classes by C. R. Rao in 1948. If the dataset contains many features, then LDA reduces the number of features to lower-dimensional space to prevent overfitting (Sorzano, 2014). LDA maintains the class discriminatory information while projecting a feature space onto a small subspace. It maximizes the component axes and separates the classes, accordingly, as shown in Fig. 2.

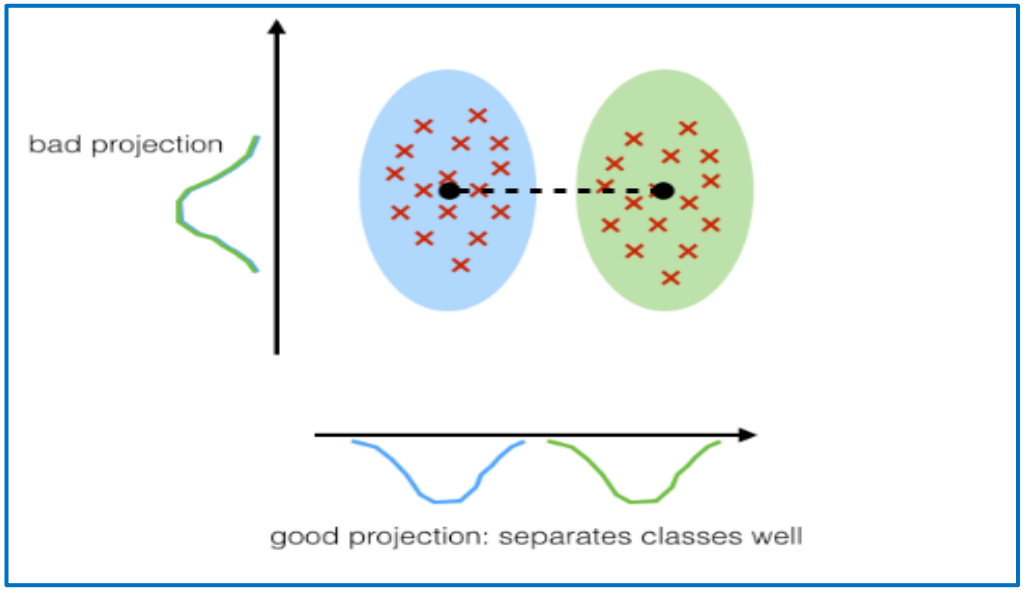

Fig. 2

Implementation of LDA is like Analysis of Variance (ANOVA) and Regression analysis, in which a response variable is linear combination features. However, in ANOVA the set of features are categorical and response variable is continuous, whereas LDA has categorical response variable with continuous features. Mathematically, it applies Bayes rule to estimates the probability that a new observation belongs to each class. The class that has the highest probability is predicted for that instance (Kuhn, 2013). Unlike Logistic Regression the key benefit of LDA model is that it can support binary as well as multiple class problem. However, there are few assumptions that need to satisfy to make good predictions. Those are mention below,

- The independent variables of the dataset must follow a multivariate normal distribution.
- Variance or Covariance across all the independent variables should be equal.
- The correlation among the predictors should very less.

3.1.3 Naïve Bayes (NB)

In machine learning and data mining the naïve Bayes is the classification technique based on the application of Bayes Theorem. It is the simple probabilistic classifier with strong conditional independence assumptions between independent variables (Zhang,

2004). It assumes that one feature in the class is not correlated with any other feature. Let's say x is a predictor and c is a class, then Bayes rule is given by

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Where, $P(c|x) =$ Posterior probability
$P(x|c) =$ Maximum Likelihood
$P(c) =$ Prior probability of a class
$P(x) =$ Prior probability of a predictor

The above equation is used to calculate posterior probability of each class. The class that has highest posterior probability is the outcome of the prediction.

3.1.4 Support Vector Machine (SVM)

The Support Vector Machine is supervised learning method that was invented by Vladimir Vapnik in 1963. In case of SVM, an instance is assumed to be a p-dimensional vector, our task is to separate such instances into the $(p-1)$ dimensional hyperplane. The hyperplane is a line or decision boundary that classify the instances by their classes. There might exist infinite hyperplanes in high dimensional space. But the best possible hyperplane is one which has the distance from the nearest instance or vector on each side for any dataset. Such decision boundary or hyperplane is called maximum-margin hyperplane (Kuhn, 2013).

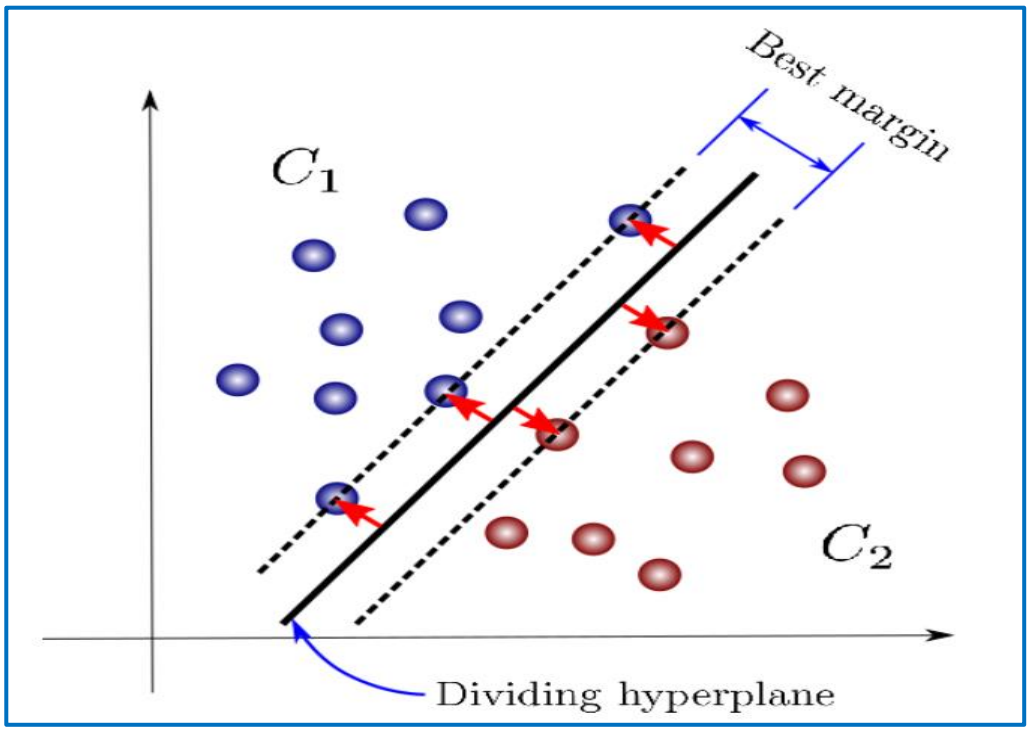

Fig.3

The Fig.3 displays a good example of a maximum-margin hyperplane that separates two classes (C_1 and C_2). The data points nearest to the line are called as support vectors. This is an example of linear SVM classifier. However, in 1992 Vapnik and other mathematicians introduced non-linear SVM classifier by implementing Kernel trick. In this study, the Gaussian radial basis Kernel function is used to train the model.

3.1.5 Random Forest (RF)

The Random Forest method for classification was first formulated by Leo Breniman. It is an ensemble learning method in supervised learning that builds multiple decision trees to produce a generalized model to reduce overfitting. This method can be applied for regression as well as a classification task. Theory of Random Forest is like bagging or bootstrap aggregation. In bagging the random samples from the original dataset are selected by sampling with replacement. Further, the features are sampled without replacement because have the same features in each tree model does not make sense. This process is called as feature bagging. The bagging method decreases the variance by creating more uncorrelated trees. In machine learning, the Random Forest algorithm constructs multiple decision trees, each tree predicts the classes to which the instances belong and makes the prediction by considering the majority vote. The output that receives many votes is chosen by the random forest as a final prediction (Kuhn, 2013). The accuracy of such a model is better than the individual decision tree model. The general implementation of the random forest algorithm is shown in Fig.4 below.

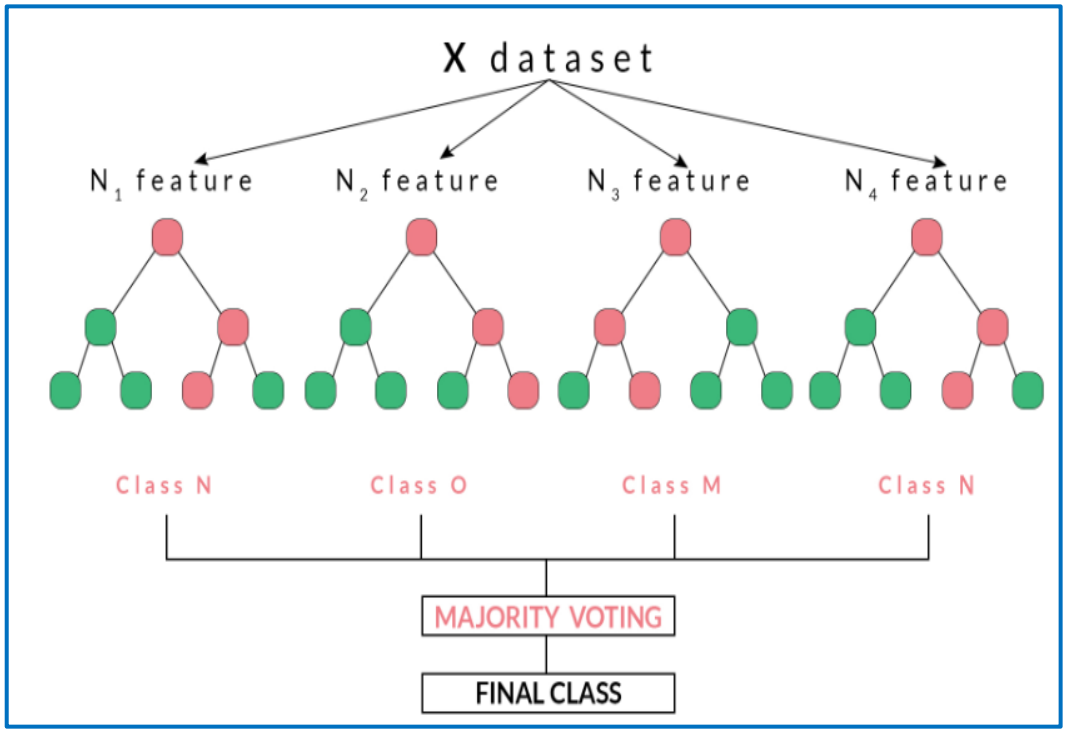

Fig.4

3.2 Evaluation Methods

3.2.1 Accuracy

The accuracy is the simplest evaluation method that is used to calculate the performance of the model. It is defined as the number of correctly classified instances divided by the total number of instances from training data. It can be calculated by the formula,

$$Accuracy = \frac{True\ Positive + True\ Negative}{Total\ Instances}$$

It is good to evaluate performance only if the dataset contains same size of sample for each class. Suppose out training dataset contains 99% instances of class A and 1% instances of class B. Then any classification algorithm may give perfect accuracy by predicting all the observation belonging to class A. It works well for two class classification problem (binary classification) than multiclass problem. This indicates that we cannot rely completely on accuracy score. In this research the datasets don't have any challenges, therefore, I have included accuracy as an evaluation measure.

3.2.2 Cohen's Kappa (Kappa statistic)

The Kappa statistic was invented by Jacob Cohen in 1960. It is designed to measure the agreement between the two raters (Kuhn, 2013). It is given by the equation,

$$K = \frac{O - E}{1 - E}$$

Where O and E refers to expected and observed accuracy respectively. Kappa takes the value between -1 and 1. If the value is 1 then the observed and expected accuracy completely agree with each other. If the value is negative or 0 the there is no agreement among them. It works like classification accuracy, but it includes the accuracy that might be generated by randomness or chance. Unlike classification accuracy, Cohen's Kappa is more effective when the classes are imbalanced.

3.2.3 Precision and Recall

In classification, precision refers to the percentage of our predictions that are relevant. Whereas, recall is the percentage of the total relevant predictions correctly classified by the model. They are given by the formula,

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Precision is also called sensitivity, and precision is also called as positive predictive value (PPV).

4 Research Methodology

The analysis is done in two parts, one with including all the classes and other with only two classes. The class that is well separated from other two was eliminated. The reason to consider only two classes is to compare logistic regression method. We know that logistic regression is only meant for two-class classification (binary classification) problem, it is not applicable for multiclass problem. All the computations were done in R software by using caret package.

4.1 Datasets

The four datasets were randomly selected from the UCI machine learning repository that are associated with classification problem. The information about the datasets is presented in Table.1 below.

Names	Features	Instances	Classes	Names of classes
Iris	4	150	3	setosa, versicolor, virginica.
Wine	13	178	3	cultivars (1, 2 and 3)
Breast Cancer Wisconsin (Diagnostic)	30	569	2	benign and malignant
Abalone	8	4177	3	Rings (1-8, 9 and 10, 11-onwards)

Table.1

i. Iris

The data has four features sepal length, sepal width, petal length and petal width this are the physical measurements of the flowers. There are three types of species (setosa, virginica and versicolor). Each of them consists of 50 samples.

ii. Wine

This data constitutes 13 chemical elements that are outcomes of the chemical analysis of the wine grown in Italy. There are three 3 classes representing 3 different types of Wine (Cultivars). It has 178 instances in total.

iii. Breast Cancer Wisconsin (Diagnostic)

Cancer data contains the attributes that are the physical measurement of the images of the breast mass. There are 30 features with 560 instances distributed into two classes (benign and malignant).

iv. Abalone

The data consist of features that are physical measurements of the abalone shell. There are 4177 observations with 8 features. The class variable is the number of rings on the abalone shell. The rings are categories into three classes (rings 1-8 in one class, 9 and 10 in second class and 11 onwards in third class). The problem is to predict the age of the abalone by calculating the number of rings on its shell. to count those rings for each abalone is a time-consuming task, but it can be solved by classifying the abalones into three age groups based on the physical measurements.

4.2 Data pre-processing

The selected datasets do not have any major challenges such as there are no missing values in any of them, neither they have class imbalance problem. So, there were only a few steps that were needed to implement.

- Dealing with categorical variables.

  There exist a few categorical features in selected datasets. These features were transformed into a dummy variable. The categories from the features were labelled as 0 and 1 i.e. in the form of binary values. Moreover, few categorical variables such as sex and ID was not considered in analysis.

- Splitting the datasets into train and test set

All the datasets were divided into train and test set. The partition was done based on 66% train-test split for Iris, Wine and Cancer. That is 66% of all the instance were considered to train the model and 34% of instances in the test set to test the model. The Abalone data was divided based on 75% train-test split. 75% of observation in training set and 25% for test set because the data has a large number of instances.

- Feature Scaling

  All the datasets are scaled by transforming the distribution to standard normal distribution where the mean of the distribution is 0 and standard deviation equal to 1. Many features have physical measurements that highly vary with each other. The features with high variance dominate the other features in the dataset therefore it important to center the range of the features around zero with unit variance, which may lead the model to make better predictions.

4.3 Selection of Classification methods

The classification methods were selected by understanding the limitation and challenges of the datasets. Moreover, by studying their diversity of representation and style of learning the data. The selected classification methods are,

- Logistic Regression (LR)
- Linear Discriminant Analysis (LDA)
- Naïve Bayes (NB)
- Support Vector Machine (SVM)
- Random Forest (RF)

4.4 Selection of Evaluation methods

The selected evaluation methods are,

- Accuracy
- Cohen's Kappa
- Precision
- Recall

This evaluation metrics were selected based on its simplicity in comparison of models.

5 Analysis and Results

The classification methods were implemented on training data of all datasets. Each method was trained 25 times by sampling with replacement (bootstrapping). The predictions were made on test data. First, we will discuss the predictions of the classification and evaluation methods by including all classes in the dataset and then we will discuss the comparison of methods by considering two classes.

5.1 Model comparison and Analysis of datasets on all classes

The classification models selected for this analysis are LDA, NB, SVM and RF. The performance of these models was evaluated based on the score of accuracy and kappa statistic. Precision and recall were not included in this analysis because of its complexity in evaluating the performance of the multiclass classification problem. The results and predication of methods on each dataset are explained as follows.

5.1.1 Iris analysis with all classes

The Accuracy and kappa scores for each classification methods are mentioned in Table.2 below.

Classification methods	Accuracy	Kappa
LDA	98.04%	0.9706
NB	92.16%	0.8824
SVM	94.12%	0.9118
RF	96.08%	0.9412

Table.2

From Table.2 it is observed that the LDA model has achieved the highest accuracy (98.04%) and Kappa (0.9706) score compared to other models.

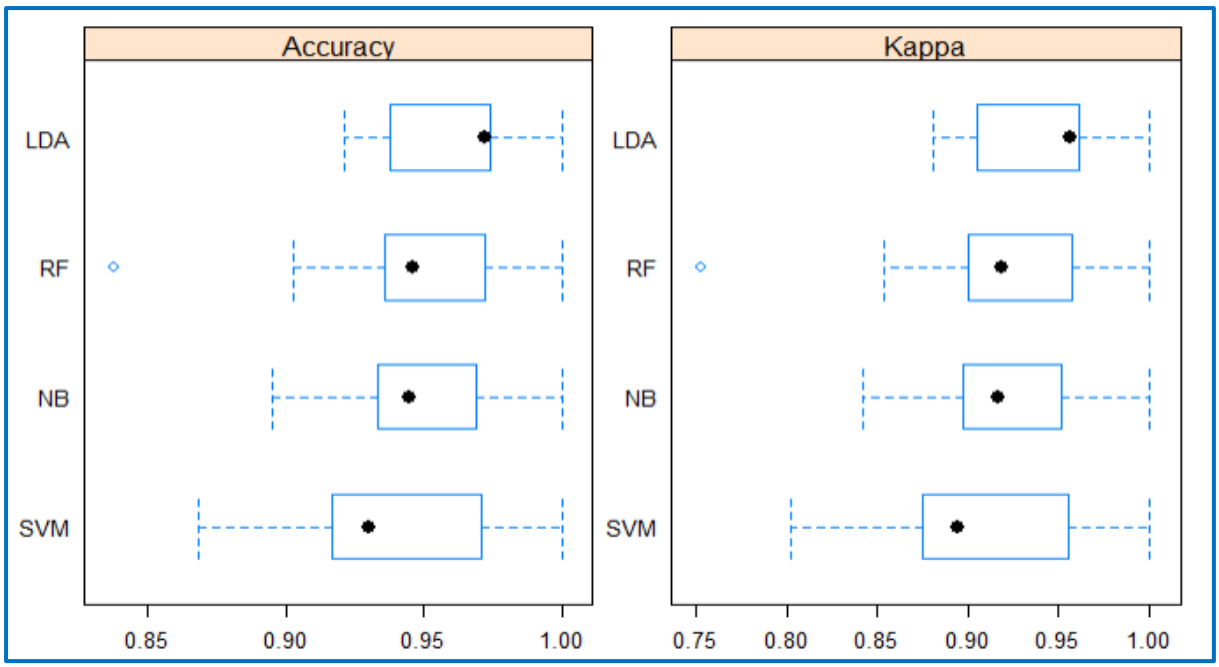

Fig.5

The box-whisker plot is used to visualize the median Accuracy and Kappa scores across all the methods as shown in Fig.5. The boxes in the plot are ranked from highest to lowest. The median values are represented by the black dots. It is clearly seen that the median accuracy and kappa for LDA are highest, whereas, that for SVM its lowest.

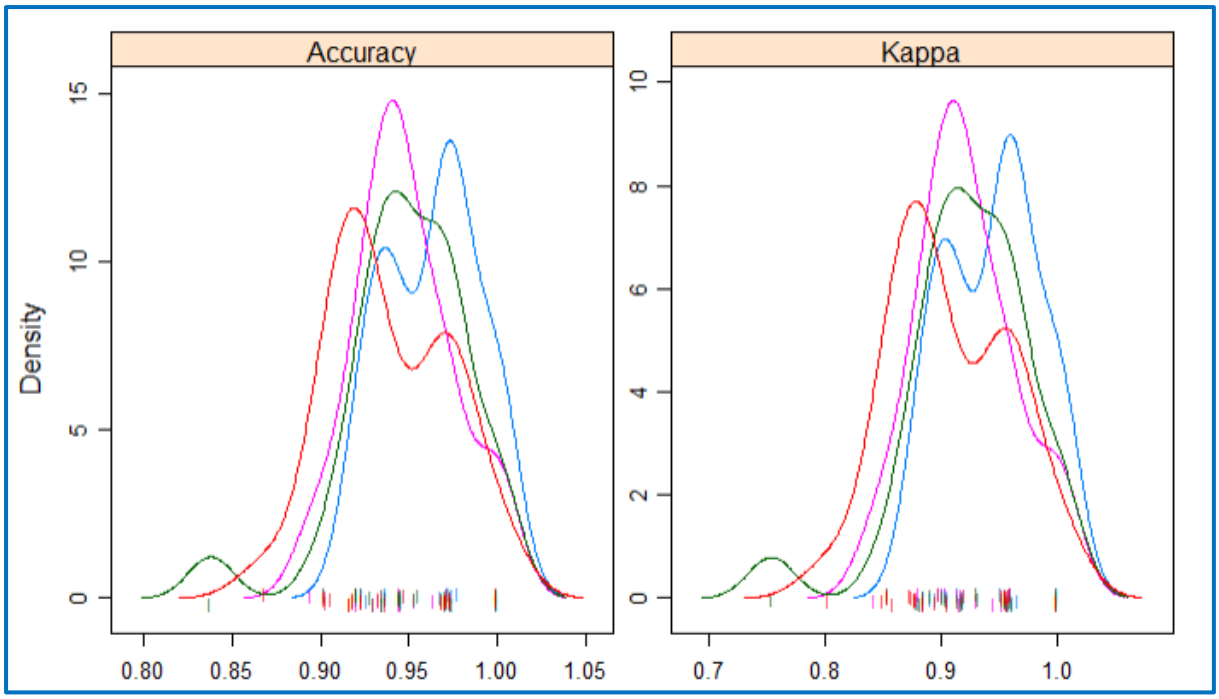

Fig.6

The Fig.6 represents the distribution of accuracy and Kappa in the form of density plots. This plot indicates the algorithms significantly overlap with each other.

5.1.2 Wine analysis with all classes

Let's have a look at the accuracy and kappa scores of methods for Wine data.

Classification methods	Accuracy	Kappa
LDA	96.67%	0.9494
NB	96.67%	0.9494
SVM	96.67%	0.9494
RF	98.33%	0.9747

Table.3

Interestingly, the Random Forest (RF) model produces the highest accuracy and kappa score, compared to other classification methods.

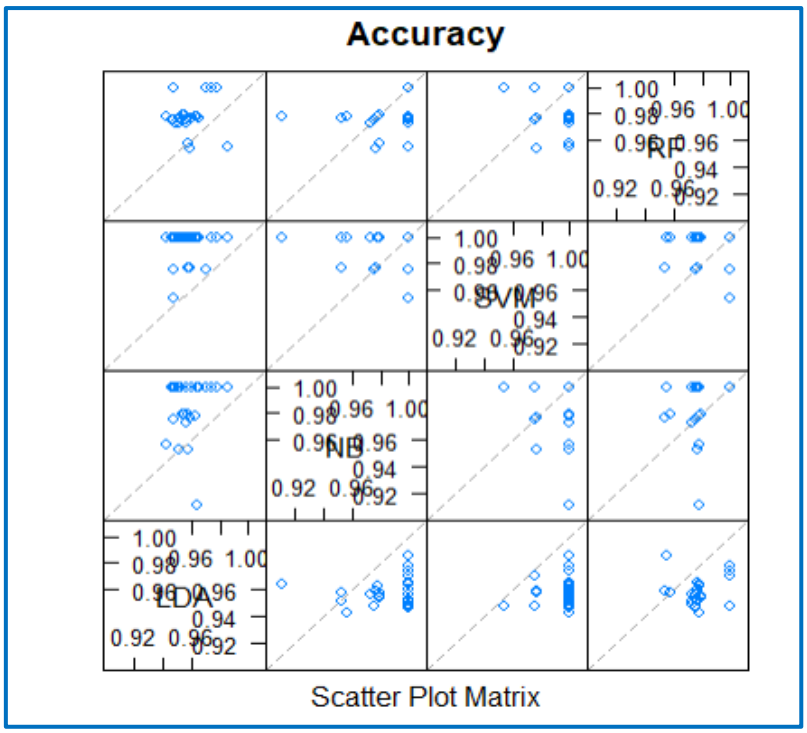

Fig.7

Let's see the correlation plot of all algorithms with respect to their accuracy in Fig.7. From the plot it looks like not a single algorithm is correlated with each other.

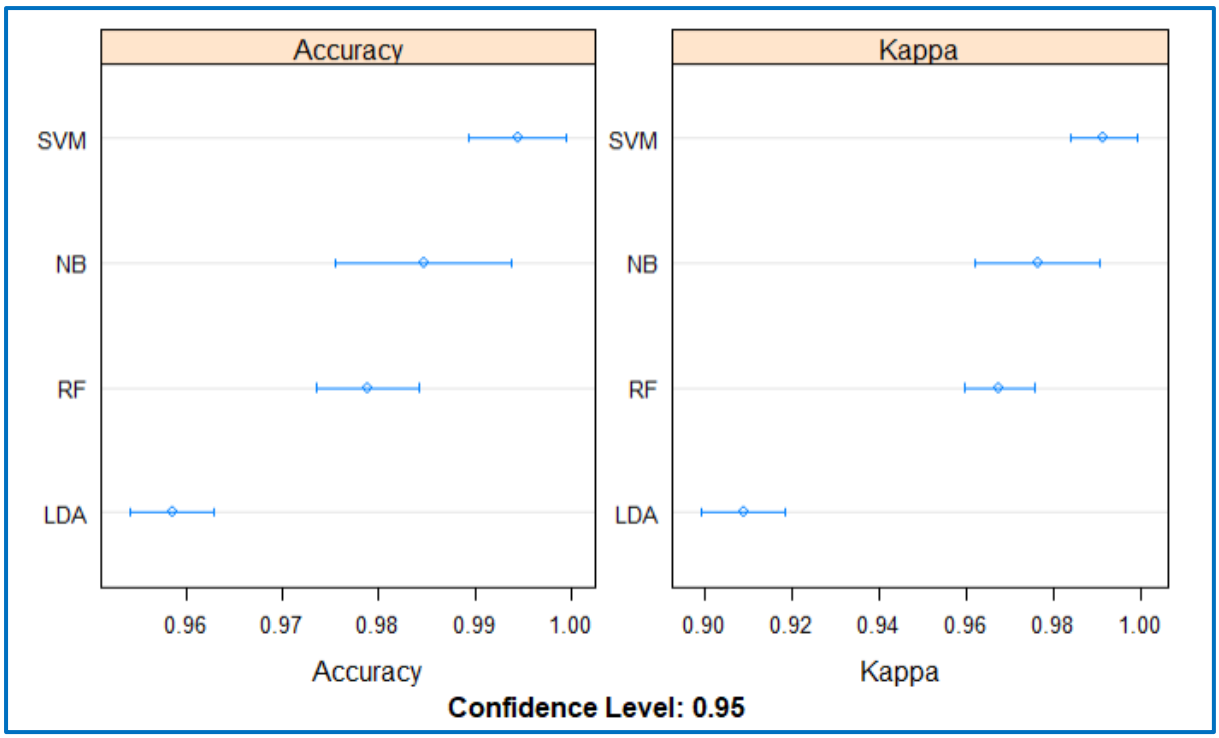

Fig.8

The dot plot in Fig.8 shows both mean estimated accuracy and kappa as well 95% confidence interval. We can say from the plot that the SVM is the best model in terms of 95% confidence level and mean estimated scores, However, LDA's Confidence interval is much lesser than other models.

5.1.3 Breast Cancer Wisconsin (Diagnostics) analysis with all classes

The Accuracy and kappa statistic of selected models for breast cancer dataset is mentioned in Table.4.

Classification methods	Accuracy	Kappa
LDA	95.79%	0.908
NB	93.68%	0.8635
SVM	94.21%	0.8753
RF	94.21%	0.8738

Table.4

We can say that LDA has the highest accuracy and kappa score compared to other classification method and Naïve Bayes has the lowest accuracy and kappa score.

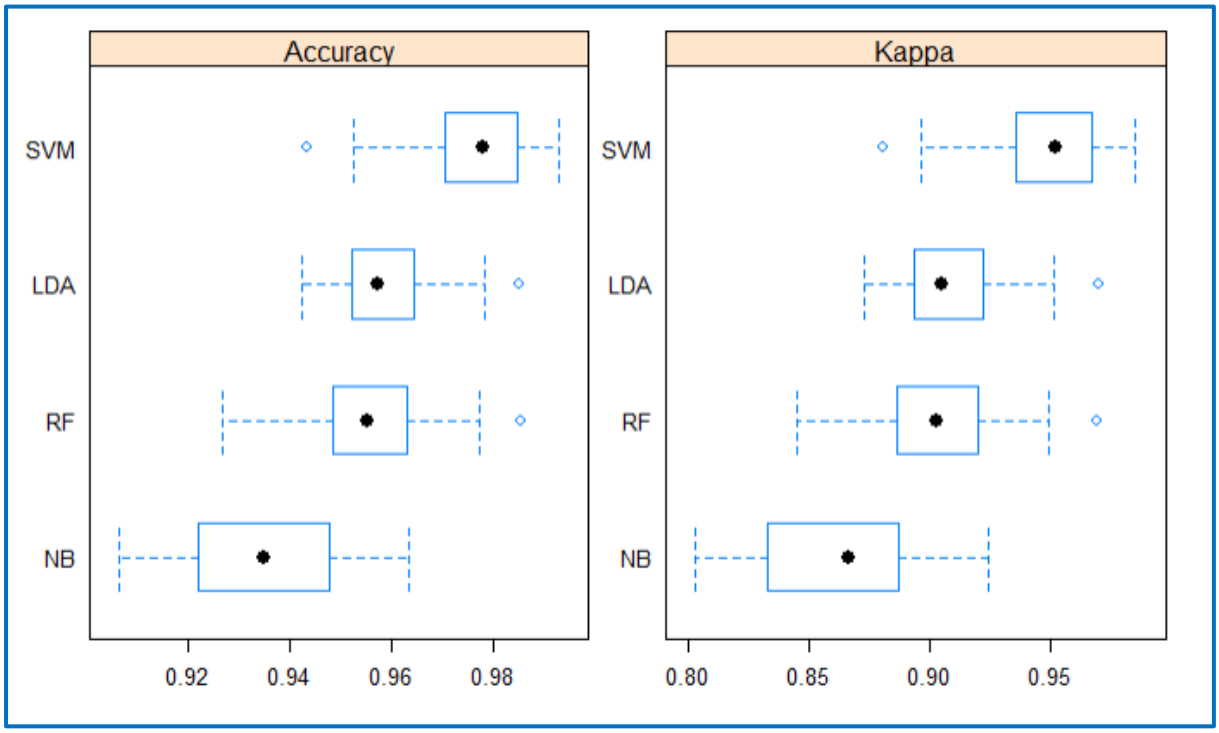

Fig.9

The Fig.9 indicates that the median accuracy and kappa scores of SVM are better than the other models. However, Naïve Bayes still gives the lowest results.

Now let's look at the 95% confidence level plot of all the algorithms in Fig.10 below.

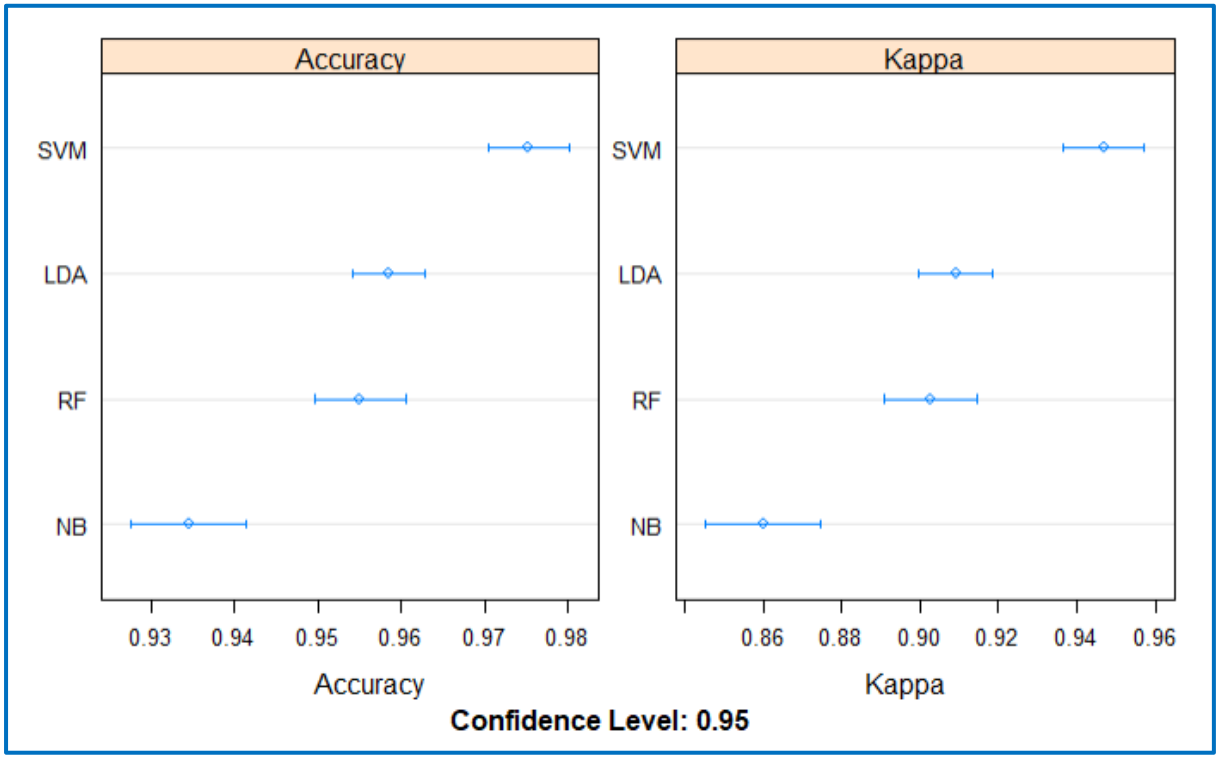

Fig.10

The outcome is same as that of previous plot, the confidence interval of Naïve Bayes is significantly less than the other models.

5.1.4 Abalone analysis with all classes

The accuracy and kappa scores for abalone data are given in Table.5.

Classification methods	Accuracy	Kappa
LDA	64.21%	0.4635
NB	57.61%	0.3647
SVM	64.21%	0.4618
RF	65.65%	0.4839

Table.5

As you can see from Table.5 that the accuracy and kappa score of the Random Forest is slightly better than other methods. However, the score of accuracy and kappa is less for Naïve Bayes algorithm.

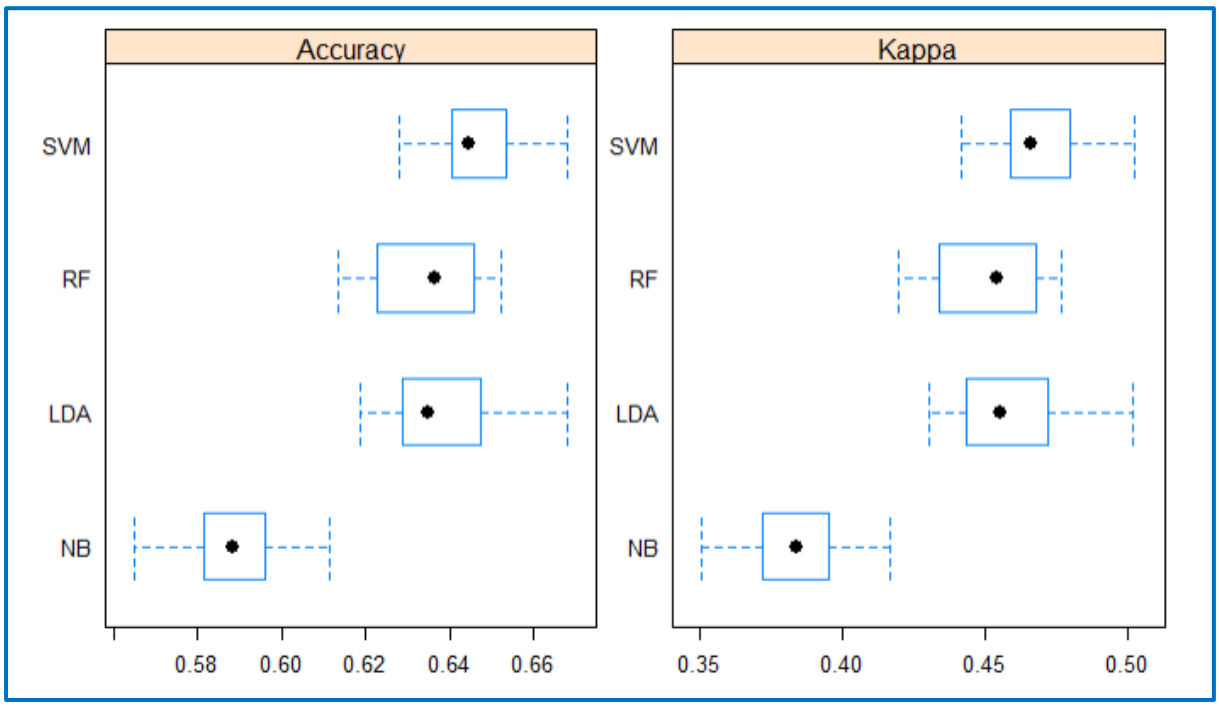

Fig.11

Now we will look at the median scores of accuracy and kappa from the box-whisker plot in Fig. 11. In this case, the median accuracy and kappa for SVM is highest and for Naïve Bayes algorithm they are lowest.

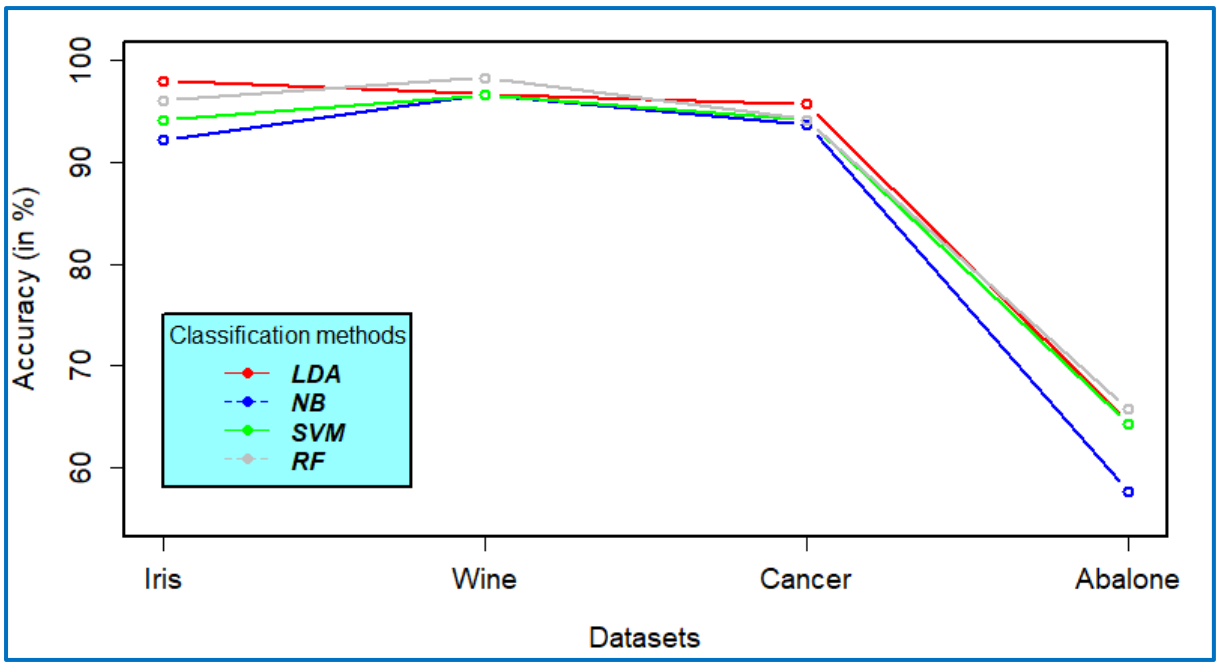

Fig.12

The Fig.12 represents the plot of the accuracy scores of classification methods against four the datasets. It is clearly observed from the plot that Linear Discriminant Analysis (LDA) performs significantly better in terms of accuracy score on all the datasets. However, the accuracy score of Naïve Bayes (NB) is comparatively less.

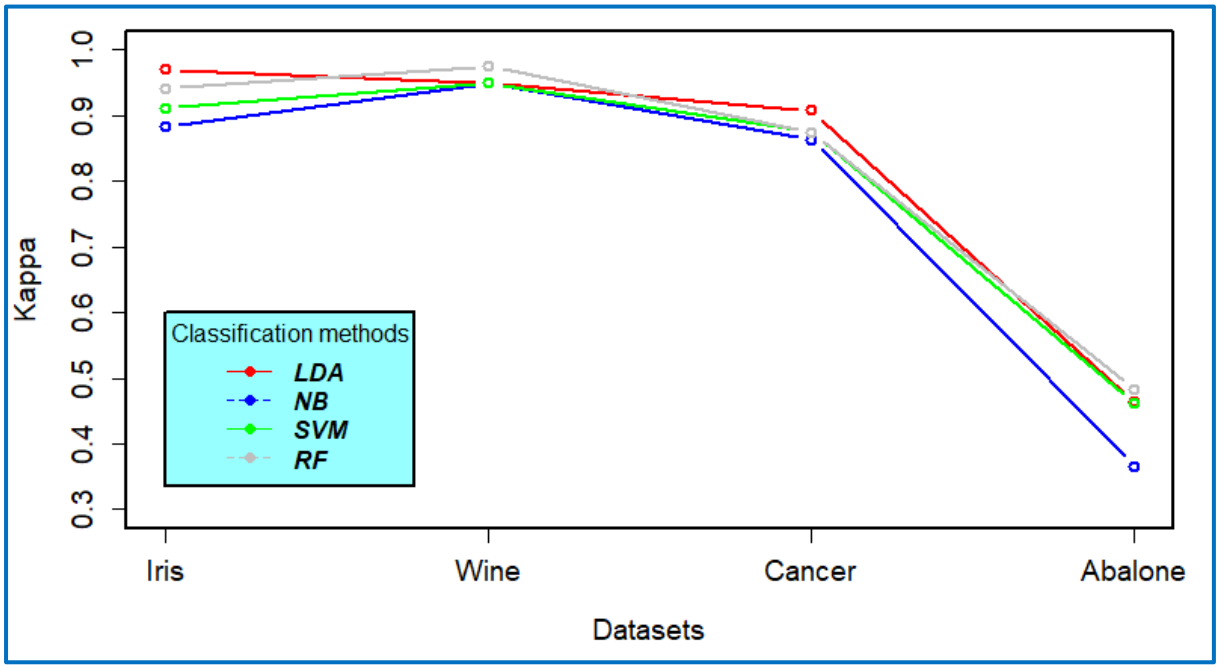

Fig.13

Fig. 13 shows the plot of the kappa scores of classification methods against four datasets. We can see that similar results are observed for kappa statistics, where LDA and RF perform significantly better than Naïve Bayes on each dataset.

5.2 Model comparison and Analysis on two classes

In the case of binary classification, five classification methods (LR, LDA, NB, SVM and RF) were selected for this analysis. The results and predication of each dataset is explained as follows,

5.2.1 Iris analysis with two classes

The class (setosa) that was more separable than the other two was eliminated and only two classes (versicolor and virginica) were considered for analysis. Therefore, the sample size of the data reduced to 100.

The accuracy, kappa, precision and recall of all five methods are mentioned in Table.6 below.

Classification methods	Accuracy	Kappa	Precision	Recall
LR	97.06%	0.9412	100%	94.12%
LDA	97.06%	0.9412	100%	94.12%
NB	94.12%	0.8824	94.12%	94.12%
SVM	91.18%	0.8235	88.89%	94.12%
RF	91.18%	0.8235	88.89%	94.12%

Table.6

It is clear from Table.6 that the LR and LDA model performs significantly well on all evaluation methods compared to other models.

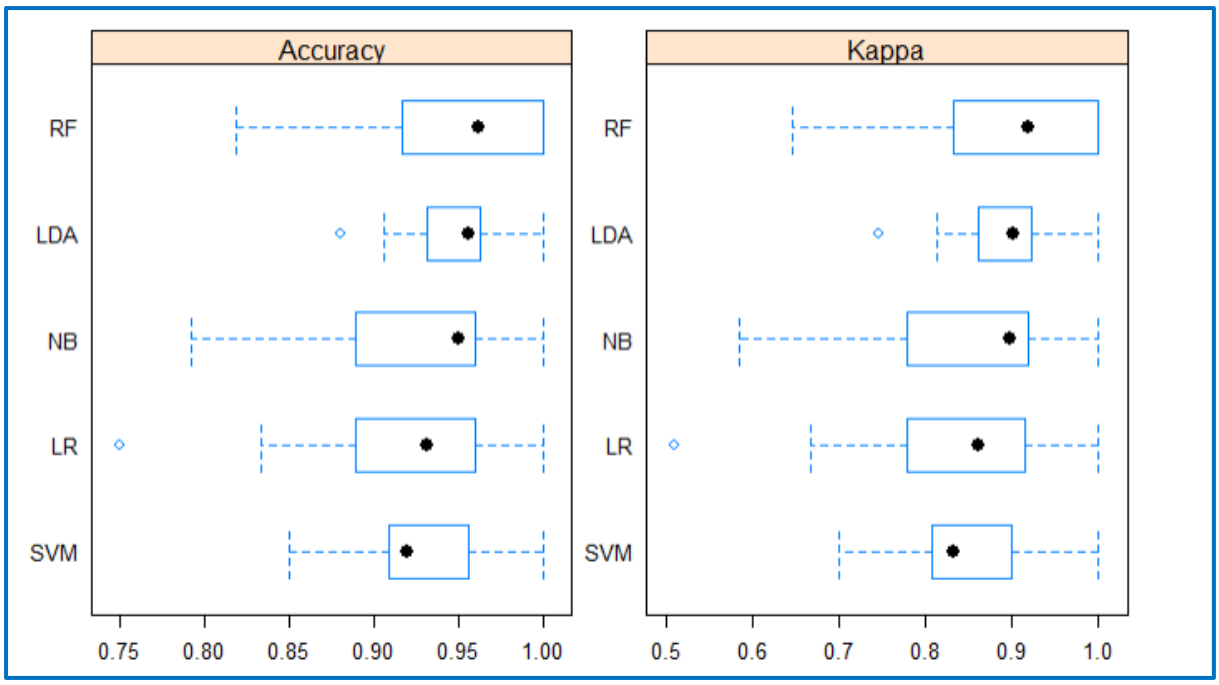

Fig.14

In Fig.14 the black dots represent the median values of accuracy and kappa. It is observed that the median accuracy and kappa score of Random Forest model are highest. Whereas, that of SVM model is lowest.

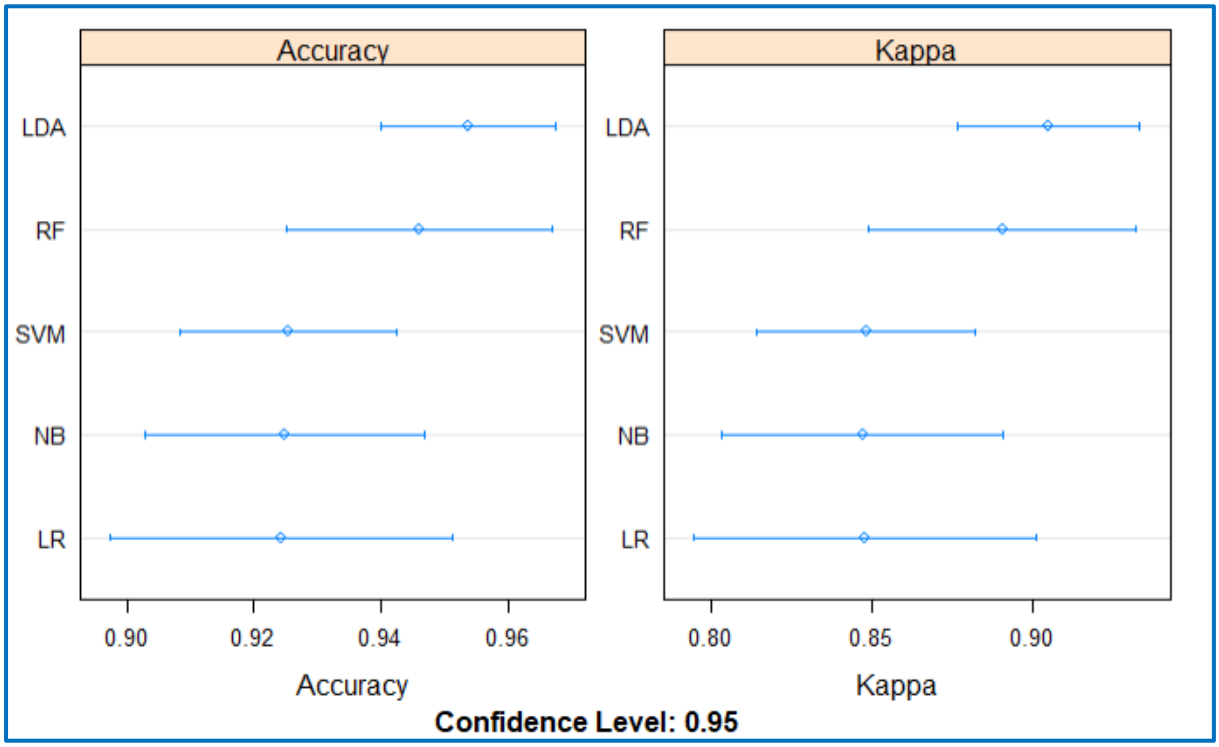

Fig.15

The Fig.15 represents the dot plot of mean accuracy and kappa score within a 95% confidence interval. Surprisingly, in terms of confidence level, the accuracy and kappa score of LR model is lowest, whereas, that of LDA model is highest.

5.2.2 Wine analysis with two classes

In this data, class 3 was more separable than the other two. So, it was removed. Now, out of 178 instances we left with 130 instances.

The accuracy and kappa score of all five methods is mentioned in Table.6 below.

Classification methods	Accuracy	Kappa	Precision	Recall
LR	97.06%	0.9412	100%	95%
LDA	97.06%	0.9412	100%	90%
NB	94.12%	0.8824	100%	95%
SVM	91.18%	0.8235	100%	95%
RF	91.18%	0.8235	100%	100%

Table.7

Table.7 indicates that LR and LDA models perform better than other methods for wine data with two classes.

Now, let's look at the density plot algorithms scores with respect to accuracy and kappa.

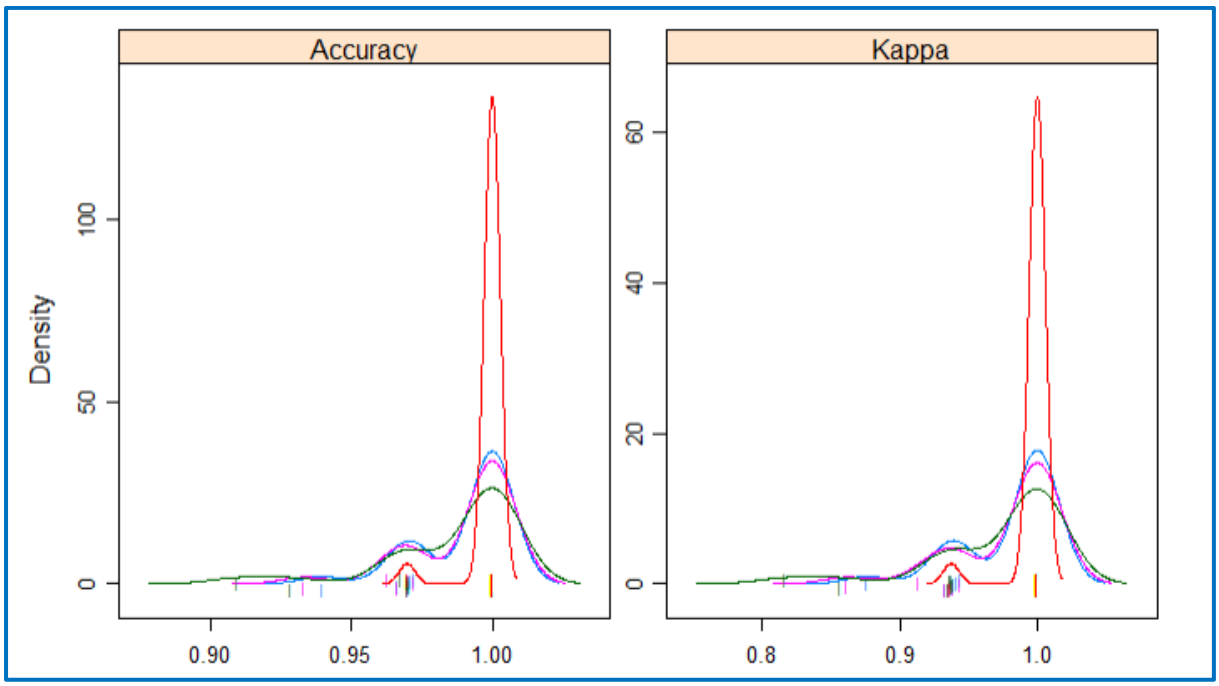

Fig.16

The density plot from Fig.16 clearly indicates the algorithms highly overlap with each other.

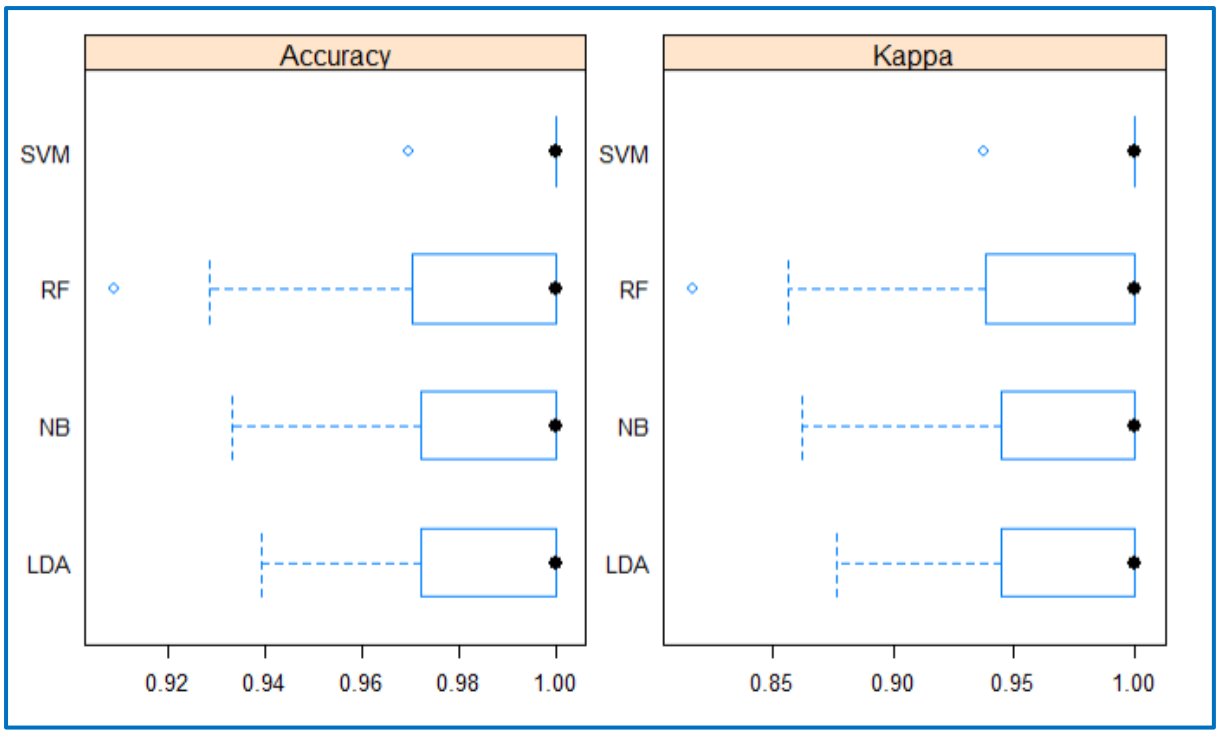

Fig.17

Surprisingly, in Fig.17 the values of median accuracy and kappa statistic are the same for all the classification methods (100% for all the methods). This means when we analyze the Wine data with two classes the median classification accuracy and kappa statistic score for all the classification methods is 100%. Hence, all classification models did well on all evaluation methods.

5.2.3 Breast Cancer Wisconsin (Diagnostic) analysis with two classes

The analysis of Breast Cancer data remains the same because there are only two classes (benign and malignant) in this dataset.

Let's have a look at values of accuracy, kappa, precision and recall from Table.8 shown below.

Classification methods	Accuracy	Kappa	Precision	Recall
LR	92.63%	0.8399	100%	95%
LDA	95.79%	0.908	100%	90%
NB	93.68%	0.8635	100%	95%
SVM	95.26%	0.8973	100%	95%
RF	95.26%	0.8973	100%	100%

Table.8

The scores of precisions and recall for all the models are equally good. However, the random forest algorithm performed slightly better in term of accuracy and kappa statistic.

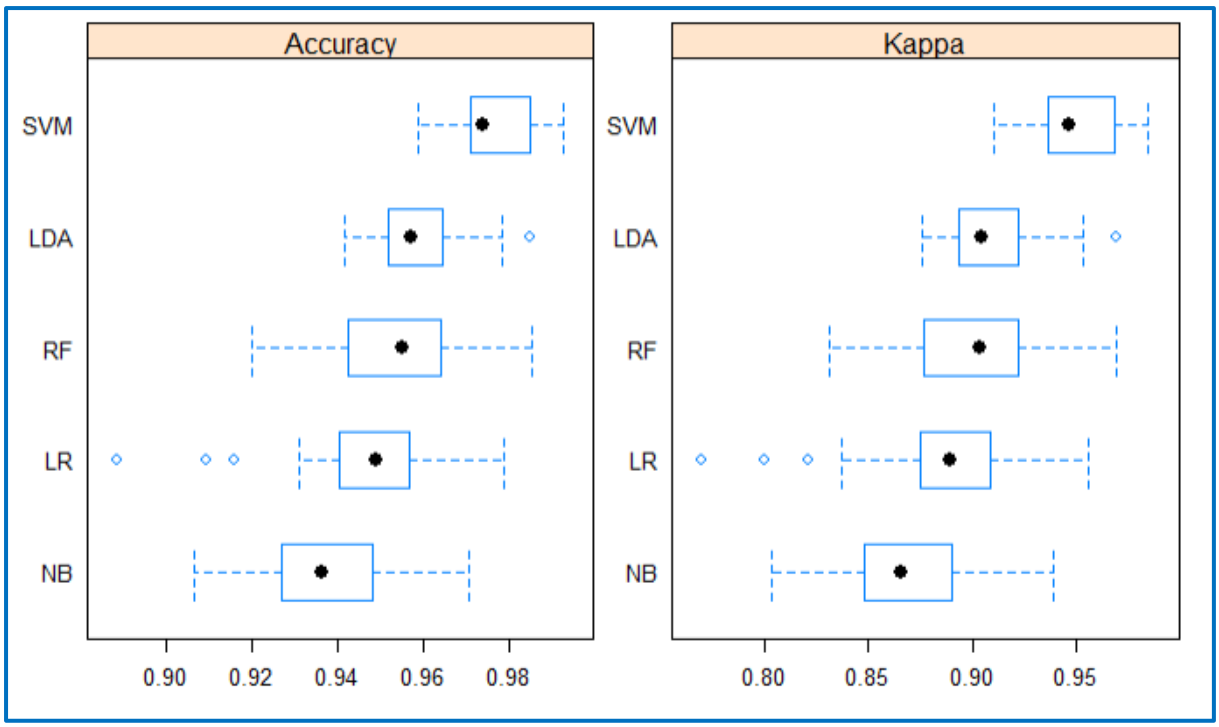

Fig.18

The median scores of accuracy and kappa for SVM model is highest, whereas, that for NB model is the lowest.

5.2.4 Abalone analysis with two classes

In the case of Abalone, the first class was removed and second and third class was considered for analysis.

The accuracy, kappa, precision and recall scores predicted by all classification methods is shown in Table.9.

Classification methods	Accuracy	Kappa	Precision	Recall
LR	68.69%	0.3742	66.19%	70.39%
LDA	68.83%	0.3775	66.11%	71.30%
NB	56.13%	0.1319	53.15%	68.88%
SVM	67.39%	0.3461	66.06%	65.26%
RF	68.98%	0.3783	67.47%	67.47%

Table.9

It is observed from the above table that compared to other classification models; the performance of Naïve Bayes is very poor.

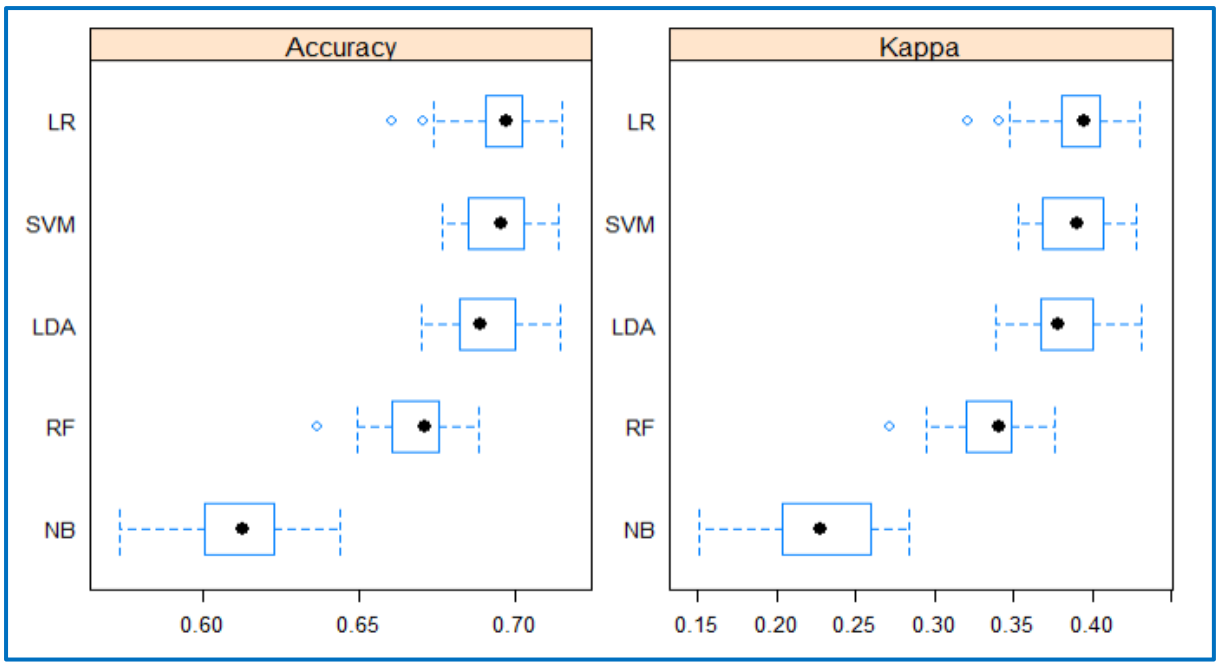

Fig.19

The Box-whisker plot in Fig.19 indicates that the median estimated accuracy and kappa score of LR is highest and that of Naïve Bayes is lowest.

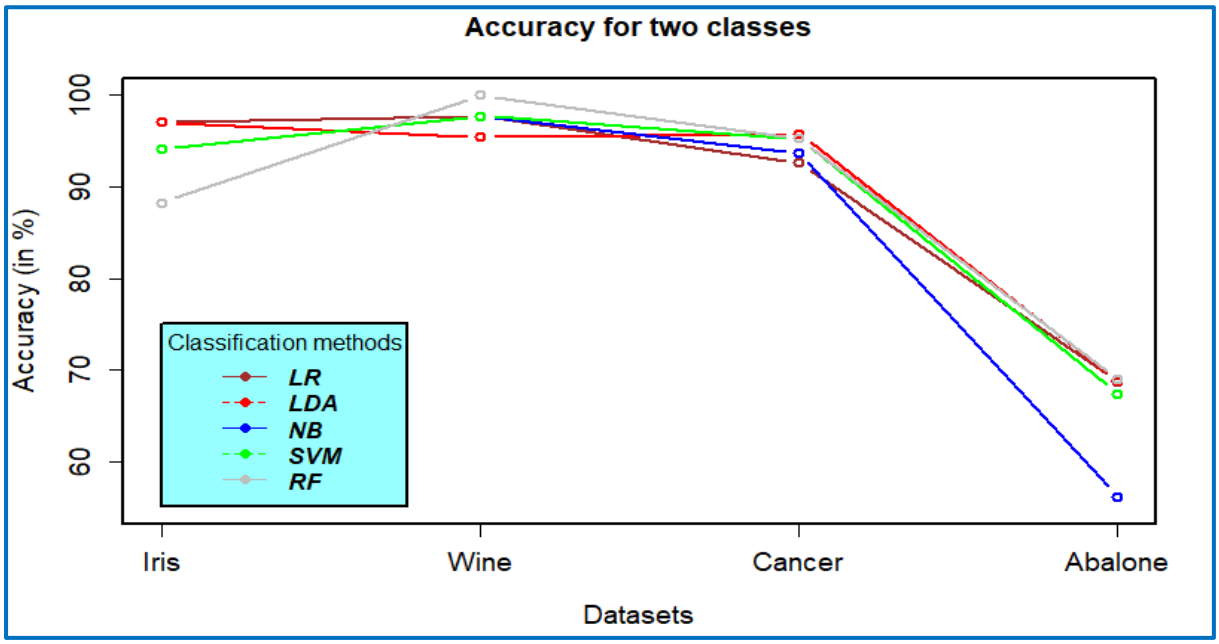

Fig.20

The Fig.20 represents the plot of the accuracy of classification methods against datasets with binary classes. When we compare classification methods in terms of accuracy and for binary class datasets the Linear discriminant analysis and Logistic regression method performed significantly well, whereas, the performance of Naïve Bayes model was very poor.

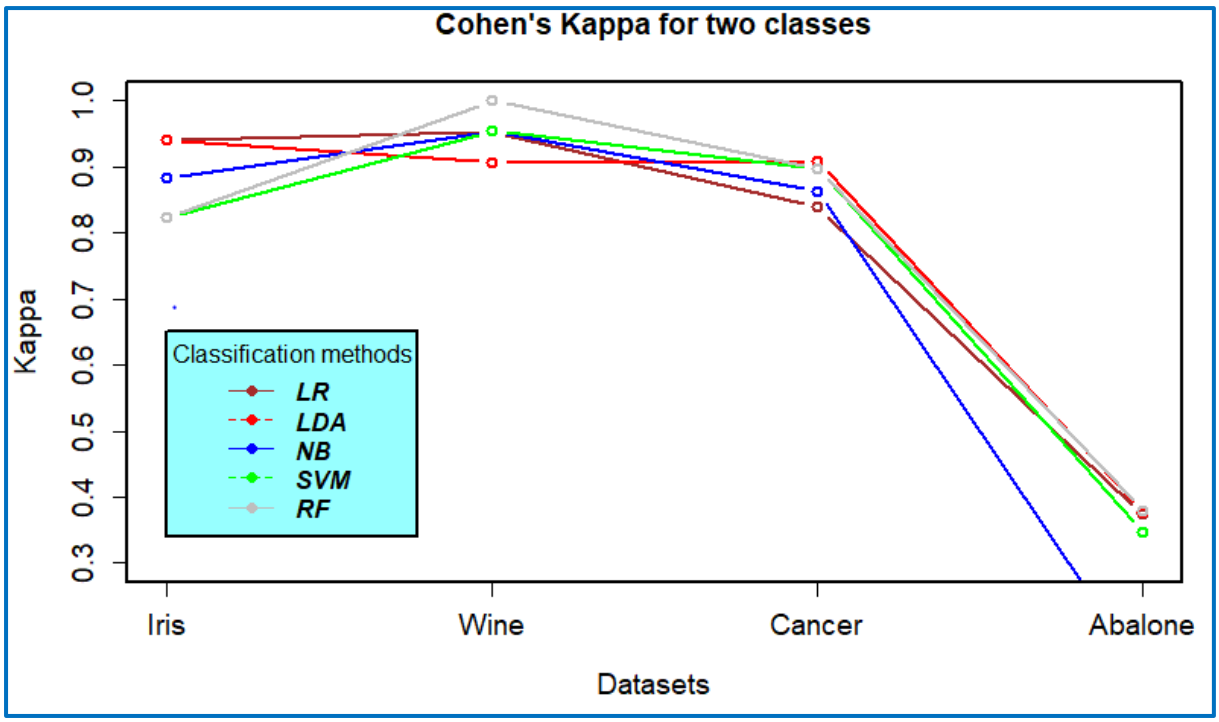

Fig.21

Cohen's Kappa plot in Fig.21 also produces the same results by giving a high score for LDA and LR model but a low score for Naïve Bayes model. The only difference is the lines are slightly more separated than the accuracy plot.

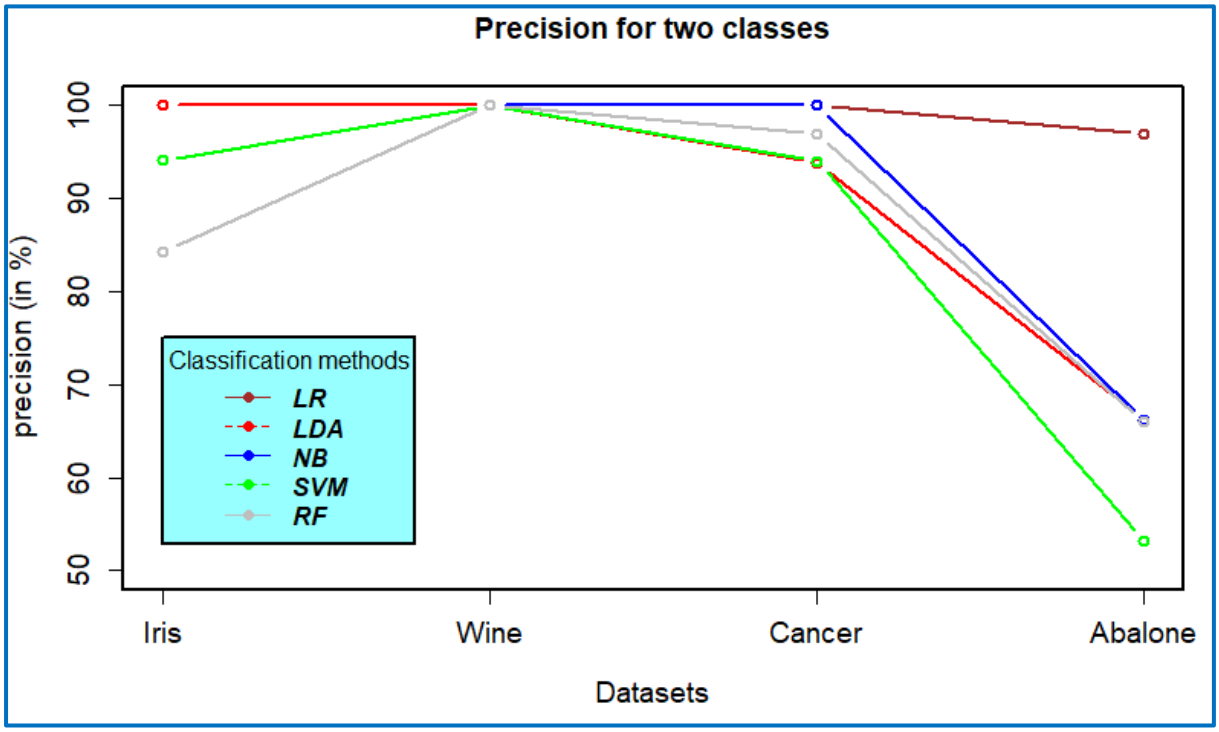

Fig.22

The plot of precision in Fig.22 tell us that the Logistic regression performs better than other methods on all datasets. Whereas, support vector machine underperforms on all datasets except Wine data.

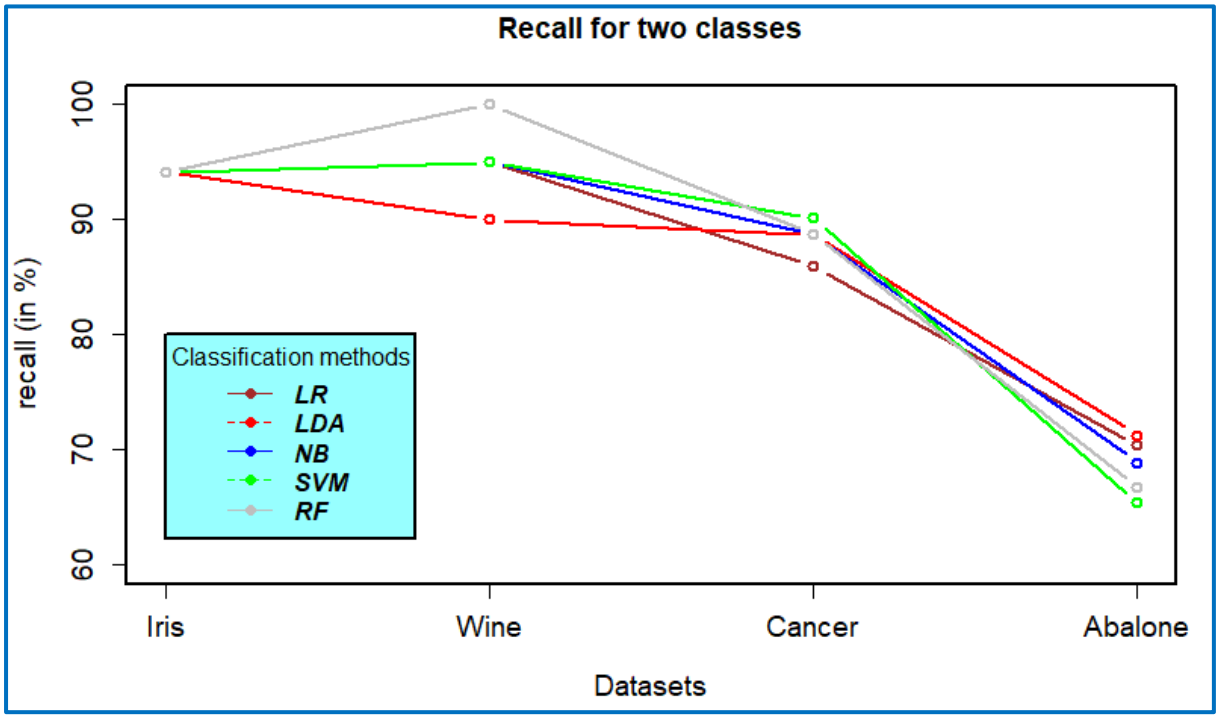

Fig.23

However, when we look at the plot of recall in Fig.23. All the classification methods produce good predictions. There is no clarity regarding the best performing model because several lines overlap with each other.

6 Conclusion

In this research project we learned to compare various classification methods with different evaluation methods on a range of datasets. The research was conducted into two parts one with multiclass datasets and other with the binary class dataset. In data analysis with more than two classes, the accuracy and kappa statistic score of LDA are highest, whereas that of Naïve Bayes is lowest. In data analysis with two classes, the accuracy and kappa scores of all classification methods were equally good. Further, the classification methods were evaluated based on precision and recall values. It was observed that Logistic regression was able to achieve better results than other classification methods, whereas, Support Vector Machine slightly underperformed in comparison with other methods. However, the order of the classification methods was not clear for recall values. There were not any increasing or decreasing trend in the plot. Moreover, the lines were overlapping with each other.

At a glance, the Wine data was simplest to classify the instances. All classification methods were able to classify more than 90% observations for Wine data. In contrast, not a single model was able to classify at least 70% of observation of Abalone data. This is because the age of the abalone is determined by counting the number of rings on its shell. But here we tried to predict the age by classifying the instance based on its physical measurements such as length, height etc. Therefore, we get fewer promising results for Abalone data.

7 References

Cramer. (2002). The origins of logistic regression .

J.S, C. (2002). The origin of logistic regression.

James, G. W. (2013). *An Introduction to statistical learning.* New York: springer.

Japkowicz, N. (2006). Why Question Machine Learning Evaluation Methods? *In AAAI workshop on evaluation methods for machine learning*, 6-11.

Kuhn, M. &. (2013). *Applied Predictive Modeling.* New York: Springer.

Sorzano, C. O. (2014). *A survey of dimensionality reduction techniques.*

Zhang, H. (2004). The Optimality of Naive Bayes. *AA*.

8 Appendix

In this section I have mentioned R code that I have used for computation.

1. **Iris data with all classes**

```
library(caret)

#Import Iris data
iris = read.csv('iris.csv')
head(iris)

#The data is pre-processed by converting three categories of categorical variable
'species' into factors. Further they are divided into train and test set.

iris$species = factor(iris$species,
                      levels = c('Iris-setosa','Iris-versicolor', 'Iris-virginica'),
                      labels = c(0, 1, 2))
head(iris)

#The data is partitioned based on 66% train-test split.
```

```
library(caTools)
set.seed(122)
split = sample.split(iris$species, SplitRatio = 2/3)
iris_train = subset(iris, split == T)
iris_test = subset(iris, split == F)


#APPLYING LDA:

library(MASS)
set.seed(122)

#Fitting LDA on training data

lda_iris = train(species ~ ., data=iris_train, method='lda')

#predicting outcomes on test set

iris.lda_pred = predict(lda_iris, iris_test)

#The model is evaluated based on the confusion matrix and other evalution metrics
in the following way.

confusionMatrix(iris.lda_pred, iris_test$species, mode = "everything")


#APPLYING NAIVE BAYES:

library(e1071)
library(klaR)
set.seed(122)

#Fitting Naïve Bayes on training data

naive_bayes.iris = train(species ~ ., data=iris_train, method='nb')
naive_bayes.iris

#The outcomes of test data are predicted in the following way.

iris.nb_pred = predict(naive_bayes.iris, iris_test)
```

The model is evaluated based on the confusion matrix and other evalution metrics as shown below.

```
confusionMatrix(iris.nb_pred, iris_test$species, mode = "everything")


#APPLYING SVM:

set.seed(122)
svm.iris = train(species ~ ., data=iris_train, method='svmRadial')

#Fitting SVM on training data

#The outcomes of test data are predicted in the following way.

iris.svm_pred = predict(svm.iris, newdata = iris_test[, 1:4])

#The model is evaluated based on the confusion matrix and other evalution metrics as shown below.

confusionMatrix(iris.svm_pred, iris_test$species, mode = "everything")


#APPLYING RANDOM FOEREST:

set.seed(122)

#Fitting random forest on training data

rf.iris = train(species ~ ., data=iris_train, method='rf')

#The outcomes of test data are predicted in the following way.

iris.rf_pred = predict(rf.iris, newdata = iris_test[, 1:4])

#The model is evaluated based on the confusion matrix and other evalution metrics as shown below.

confusionMatrix(iris.rf_pred, iris_test$species, mode = "everything")

#The four classification algorithms are compared based on their Accuracy and Kappa statistic.

iris.m1 <- resamples(list(LDA=lda_iris, NB=naive_bayes.iris, SVM=svm.iris, RF=rf.iris))
summary(iris.m1)
```

```
scales <- list(x=list(relation="free"), y=list(relation="free"))
bwplot(iris.m1, scales=scales)
densityplot(iris.m1, scales=scales, pch = "|")
dotplot(iris.m1, scales=scales)
parallelplot(iris.m1)
splom(iris.m1)
xyplot(iris.m1, models=c("SVM", "LDA"))
diffs <- diff(iris.m1)
summary(diffs)

#The table of accuracy and kappa score for each method is shown below
table = matrix(NA, nrow = 4, ncol = 3)
colnames(table) = c("Models", "Accuracy", "Kappa")
table[1, ] = c("Linear Discriminant Analysis (LDA)", "98.04%", "0.9706" )
table[2, ] = c("Naive Bayes (NB)", "92.16%", "0.8824" )
table[3, ] = c("Support Vector Machine (SVM)", "94.12%", "0.9118" )
table[4, ] = c("Random Forest (RF)", "96.08%", "0.9412" )

#install.packages("kableExtra")
library(kableExtra)
kable(table, caption = "Results of Iris data")
```

2. **Wine data with all classes**

```
#Importing Wine data.

wine = read.csv('wine.csv')
head(wine)

#Converting the class variable into factors.

wine$cultivars = as.factor(wine$cultivars)

#The data is partitioned based on 66% train-test split.

set.seed(123)
split = sample.split(wine$cultivars, SplitRatio = 2/3)
wine_train = subset(wine, split == T)
wine_test = subset(wine, split == F)

#Both train and test sets are Standardizied in the follwing way.

wine_train[-1] = scale(wine_train[-1])
wine_test[-1] = scale(wine_test[-1])
```

#APPLYING LDA:

```
library(MASS)
set.seed(123)

#Fitting LDA on training data

lda_wine = train(cultivars ~ ., data=wine_train, method='lda')

#The observations are predicted on the test set.

wine.lda_pred = predict(lda_wine, wine_test)

#The model is evaluated based on the confusion matrix and other evalution metrics in the following way.

confusionMatrix(wine.lda_pred, wine_test$cultivars, mode = "everything")
```

#APPLYING NAIVE BAYES:

```
#install.packages('e1071')
library(e1071)
library(klaR)
set.seed(123)

#Fitting Naïve Bayes on training data

naive_bayes.wine = train(cultivars ~ ., data=wine_train, method='nb')

#The outcomes of test data are predicted in the following way.

wine.nb_pred = predict(naive_bayes.wine, wine_test)

#The model is evaluated based on the confusion matrix and other evalution metrics as shown below.

confusionMatrix(wine.nb_pred, wine_test$cultivars, mode = "everything")
```

#APPLYING SVM:

```
set.seed(123)

#Fitting SVM on training data

svm.wine = train(cultivars ~ ., data=wine_train, method='svmRadial')

#The outcomes of test data are predicted in the following way.

wine.svm_pred = predict(svm.wine, newdata = wine_test)

#The model is evaluated based on the confusion matrix and other evalution metrics as shown below.

confusionMatrix(wine.svm_pred, wine_test$cultivars, mode = "everything")


#APPLYING RANDOM FOREST:

set.seed(123)

#Fitting random forest on training data

rf.wine = train(cultivars ~ ., data=wine_train, method='rf')

#The outcomes of test data are predicted in the following way.

wine.rf_pred = predict(rf.wine, newdata = wine_test)

#The model is evaluated based on the confusion matrix and other evalution metrics as shown below.

confusionMatrix(wine.rf_pred, wine_test$cultivars, mode = "everything")

#The four classification algorithms are compared based on their Accuracy and Kappa statistic.

wine.m2 <- resamples(list(LDA=lda_wine, NB=naive_bayes.wine, SVM=svm.wine, RF=rf.wine))
summary(wine.m2)
scales <- list(x=list(relation="free"), y=list(relation="free"))
bwplot(wine.m2, scales=scales)
densityplot(wine.m2, scales=scales, pch = "|")
```

```
dotplot(wine.m2, scales=scales)
parallelplot(wine.m2)
splom(wine.m2)
xyplot(wine.m2, models=c("SVM", "LDA"))
diffs <- diff(wine.m2)
summary(diffs)

#The table of accuracy and kappa score for each method is shown below

table = matrix(NA, nrow = 4, ncol = 3)
colnames(table) = c("Models", "Accuracy", "Kappa")
table[1, ] = c("Linear Discriminant Analysis (LDA)", "96.67%", "0.9494" )
table[2, ] = c("Naive Bayes (NB)", "96.67%", "0.9494" )
table[3, ] = c("Support Vector Machine (SVM)", "96.67%", "0.9494" )
table[4, ] = c("Random Forest (RF)", "98.33%", "0.9747" )

#install.packages("kableExtra")
library(kableExtra)
kable(table, caption = "Results of Wine data")
```

3. **Breast Cancer Wisconsin (Diagnostic) with all classes**

```
#Importing cancer data.

wdbc = read.csv('wdbc.csv')

#The data is pre-processed by converting two categories of categorical variable
'diagnosis' into factors. Further they are divided into train and test set.

wdbc$diagnosis = factor(wdbc$diagnosis, levels = c('M','B'), labels = c(0,1))

#The data is partitioned based on 66% train-test split.

set.seed(24)
split = sample.split(wdbc$diagnosis, SplitRatio = 2/3)
wdbc_train = subset(wdbc, split == T)
wdbc_test = subset(wdbc, split == F)

#Both train and test sets are Standardizied in the follwing way.

wdbc_train[, 3:32] = scale(wdbc_train[, 3:32])
wdbc_test[, 3:32] = scale(wdbc_test[, 3:32])
```

#These four methods are trained on training data and then the outcomes are predicted on test data.

#APPLYING LDA:

```
library(MASS)
set.seed(24)

#Fitting LDA on training data

lda_wdbc = train(diagnosis ~ ., data=wdbc_train, method='lda')

#The observations are predicted on the test set.

wdbc.lda_pred = predict(lda_wdbc, wdbc_test)

#The model is evaluated based on the confusion matrix and other evalution metrics in the following way.

confusionMatrix(wdbc.lda_pred, wdbc_test$diagnosis, mode = "everything")
```

#APPLYING NAIVE BAYES:

```
#install.packages('e1071')
library(e1071)
library(klaR)
set.seed(24)

#Fitting Naïve Bayes on training data

naive_bayes.wdbc = train(diagnosis ~ ., data=wdbc_train, method='nb')

#The outcomes of test data are predicted in the following way.

wdbc.nb_pred = predict(naive_bayes.wdbc, wdbc_test)

#The model is evaluated based on the confusion matrix and other evalution metrics as shown below.

confusionMatrix(wdbc.nb_pred, wdbc_test$diagnosis, mode = "everything")
```

```
#APPLYING SVM:

set.seed(24)

#Fitting SVM on training data

svm.wdbc = train(diagnosis ~ ., data=wdbc_train, method='svmRadial')

#The outcomes of test data are predicted in the following way.

wdbc.svm_pred = predict(svm.wdbc, newdata = wdbc_test)

#The model is evaluated based on the confusion matrix and other evalution metrics
as shown below.

confusionMatrix(wdbc.svm_pred, wdbc_test$diagnosis, mode = "everything")


#APPLYING RANDOM FOREST:

set.seed(24)

#Fitting random forest on training data

rf.wdbc = train(diagnosis ~ ., data=wdbc_train, method='rf')

#The outcomes of test data are predicted in the following way.

wdbc.rf_pred = predict(rf.wdbc, newdata = wdbc_test)

#The model is evaluated based on the confusion matrix and other evalution metrics
as shown below.

confusionMatrix(wdbc.rf_pred, wdbc_test$diagnosis, mode = "everything")

#The four classification algorithms are compared based on their Accuracy and Kappa
statistic.

wdbc.m3 <- resamples(list(LDA=lda_wdbc, NB=naive_bayes.wdbc, SVM=svm.wdbc,
RF=rf.wdbc))

summary(wdbc.m3)
```

#The four classification algorithms are compared based on their Accuracy and Kappa statistic.

```
scales <- list(x=list(relation="free"), y=list(relation="free"))
bwplot(wdbc.m3, scales=scales)
densityplot(wdbc.m3, scales=scales, pch = "|")
dotplot(wdbc.m3, scales=scales)
parallelplot(wdbc.m3)
splom(wdbc.m3)
xyplot(wdbc.m3, models=c("SVM", "LDA"))
diffs <- diff(wdbc.m3)
summary(diffs)
```

#The table of accuracy and kappa scores for each method is shown below

```
table = matrix(NA, nrow = 4, ncol = 3)
colnames(table) = c("Models", "Accuracy", "Kappa")
table[1, ] = c("Linear Discriminant Analysis (LDA)", "95.79%", "0.908" )
table[2, ] = c("Naive Bayes (NB)", "93.68%", "0.8635" )
table[3, ] = c("Support Vector Machine (SVM)", "94.21%", "0.8753" )
table[4, ] = c("Random Forest (RF)", "94.21%", "0.8738" )

#install.packages("kableExtra")
library(kableExtra)
kable(table, caption = "Results of Breast Cancer Wisconsin data")
```

4. Abalone with all classes

#Importing abalone data.

```
abalone = read.csv('Abalone.csv')
abalone$sex = NULL
```

#The data is pre-processed by converting two categories of categorical variable 'diagnosis' into factors. Further they are divided into train and test set.

```
abalone$groups = cut(abalone$rings, breaks = c(0,8,10,max(abalone$rings)))
levels(abalone$groups)=c("1", "2", "3")
abalone$rings = NULL
```

#The data is partitioned based on 75% train-test split. i.e. 75% of the instances are considered for training set and rest of the them to test set.

```
library(caTools)
```

```
set.seed(240)
split = sample.split(abalone$groups, SplitRatio = 3/4)
abalone_train = subset(abalone, split == T)
abalone_test = subset(abalone, split == F)

#Both train and test sets are Standardizied in the follwing way.

abalone_train[, 1:7] = scale(abalone_train[, 1:7])
abalone_test[, 1:7] = scale(abalone_test[, 1:7])


#APPLYING LDA:

library(MASS)
set.seed(240)

#Fitting LDA on training data.

lda_abalone = train(groups ~. , data=abalone_train, method='lda')

#The observations are predicted on the test set.

abalone.lda_pred = predict(lda_abalone, abalone_test)

#The model is evaluated based on the confusion matrix and other evalution metrics
in the following way.

confusionMatrix(abalone.lda_pred, abalone_test$groups, mode = "everything")


#APPLYING NAIVE BAYES:

#install.packages('e1071')
library(e1071)
library(klaR)
set.seed(240)

#Fitting Naïve Bayes on training data

naive_bayes.abalone = train(groups ~. , data=abalone_train, method='nb')

#The outcomes of test data are predicted in the following way.

abalone.nb_pred = predict(naive_bayes.abalone, abalone_test)
```

#The model is evaluated based on the confusion matrix and other evalution metrics as shown below.

confusionMatrix(abalone.nb_pred, abalone_test$groups, mode = "everything")

#APPLYING SVM:

set.seed(240)

#Fitting SVM on training data

svm.abalone = train(groups ~. , data=abalone_train, method='svmRadial')

#Fitting SVM.

abalone.svm_pred = predict(svm.abalone, newdata = abalone_test)

#The model is evaluated based on the confusion matrix and other evalution metrics as shown below.

confusionMatrix(abalone.svm_pred, abalone_test$groups, mode = "everything")

#APPLYING RANDOM FOREST:

set.seed(240)

#Fitting random forest on training data

rf.abalone = train(groups ~. , data=abalone_train, method='rf')

#The outcomes of test data are predicted in the following way.

abalone.rf_pred = predict(rf.abalone, newdata = abalone_test)

The model is evaluated based on the confusion matrix and other evalution metrics as shown below.

confusionMatrix(abalone.rf_pred, abalone_test$groups, mode = "everything")

#The four classification algorithms are compared based on their Accuracy and Kappa statistic.

```
abalone.m4 <- resamples(list(LDA=lda_abalone, NB=naive_bayes.abalone,
SVM=svm.abalone, RF=rf.abalone))
summary(abalone.m4)
scales <- list(x=list(relation="free"), y=list(relation="free"))
bwplot(abalone.m4, scales=scales)
densityplot(abalone.m4, scales=scales, pch = "|")
dotplot(abalone.m4, scales=scales)
parallelplot(abalone.m4)
splom(abalone.m4)
xyplot(abalone.m4, models=c("SVM", "LDA"))
diffs <- diff(abalone.m4)
summary(diffs)

#The table of accuracy and kappa scores for each method is shown below

table = matrix(NA, nrow = 4, ncol = 3)
colnames(table) = c("Models", "Accuracy", "Kappa")
table[1, ] = c("Linear Discriminant Analysis (LDA)", "64.21%", "0.4635" )
table[2, ] = c("Naive Bayes (NB)", "57.61%", "0.3647" )
table[3, ] = c("Support Vector Machine (SVM)", "64.21%", "0.4618" )
table[4, ] = c("Random Forest (RF)", "65.65%", "0.4839" )



#install.packages("kableExtra")
library(kableExtra)
kable(table, caption = "Results of abalone data")
```

5. Iris data with binary class

```
#import iris data

iris2 = read.csv('iris2.csv')
head(iris2)

#The data is pre-processed by converting two categories of categorical variable
'species' into factors. Further they are divided into train and test set.

iris2$species = factor(iris2$species,
                       levels = c('Iris-versicolor', 'Iris-virginica'),
                       labels = c(1, 2))
head(iris2)
```

```
#The data is partitioned based on 66% train-test split. i.e. 66% of the instances are considered for training set and rest of the them to test set.

set.seed(122)
split = sample.split(iris2$species, SplitRatio = 2/3)
iris2_train = subset(iris2, split == T)
iris2_test = subset(iris2, split == F)


#APPLYING LR:

set.seed(122)

#Fitting LR on training data

glm_iris2 = train(species ~ ., data=iris2_train, method='glm')

#The observations are predicted on the test set.

iris2.glm_pred = predict(glm_iris2, iris2_test)

#The model is evaluated based on the confusion matrix and other evalution metrics in the following way.

confusionMatrix(iris2.glm_pred, iris2_test$species, mode = "everything")


#APPLYING LDA:

library(MASS)
set.seed(122)

#Fitting LDA on training data

lda_iris2 = train(species ~ ., data=iris2_train, method='lda')

#The observations are predicted on the test set.

iris2.lda_pred = predict(lda_iris2, iris2_test)

#The model is evaluated based on the confusion matrix and other evalution metrics in the following way.

confusionMatrix(iris2.lda_pred, iris2_test$species, mode = "everything")
```

```
#APPLYING NAIVE BAYES:

library(e1071)
library(klaR)
set.seed(122)

#Fitting Naive Bayes on training data

naive_bayes.iris2 = train(species ~ ., data=iris2_train, method='nb')

#The outcomes of test data are predicted in the following way.

iris2.nb_pred = predict(naive_bayes.iris2, iris2_test)

#The model is evaluated based on the confusion matrix and other evalution metrics
as shown below.

confusionMatrix(iris2.nb_pred, iris2_test$species, mode = "everything")


#APPLYING SVM:

set.seed(122)

#Fitting SVM on training data

svm.iris2 = train(species ~ ., data=iris2_train, method='svmRadial')

#The outcomes of test data are predicted in the following way.

iris2.svm_pred = predict(svm.iris2, newdata = iris2_test[, 1:4])

#The model is evaluated based on the confusion matrix and other evalution metrics
as shown below.

confusionMatrix(iris2.svm_pred, iris2_test$species, mode = "everything")


#APPLYING RANDOM FOEREST:

set.seed(122)
```

```
#Fitting random forest on training data

rf.iris2 = train(species ~ ., data=iris2_train, method='rf')
```

The outcomes of test data are predicted in the following way.

```
iris2.rf_pred = predict(rf.iris2, newdata = iris2_test[, 1:4])

#The model is evaluated based on the confusion matrix and other evalution metrics as shown below.

confusionMatrix(iris2.rf_pred, iris2_test$species, mode = "everything")


#The five classification algorithms are compared based on their Accuracy and Kappa statistic.

iris2.m5 <- resamples(list(LR=glm_iris2, LDA=lda_iris2, NB=naive_bayes.iris2,
SVM=svm.iris2, RF=rf.iris2))

summary(iris2.m5)
scales <- list(x=list(relation="free"), y=list(relation="free"))
bwplot(iris2.m5, scales=scales)
densityplot(iris2.m5, scales=scales, pch = "|")
dotplot(iris2.m5, scales=scales)
parallelplot(iris2.m5)
splom(iris2.m5)
xyplot(iris2.m5, models=c("SVM", "LDA"))
diffs <- diff(iris2.m5)
summary(diffs)

table = matrix(NA, nrow = 5, ncol = 3)
colnames(table) = c("Models", "Accuracy", "Kappa")
table[1, ] = c("Logistic Regression (LR)", "97.06%", "0.9412" )
table[2, ] = c("Linear Discriminant Analysis (LDA)", "97.06%", "0.9412" )
table[3, ] = c("Naive Bayes (NB)", "94.12%", "0.8824" )
table[4, ] = c("Support Vector Machine (SVM)", "91.18%", "0.8235" )
table[5, ] = c("Random Forest (RF)", "91.18%", "0.8235" )

#install.packages("kableExtra")
library(kableExtra)
kable(table, caption = "Results of iris2 data")
```

6. **Wine data with binary class**

```
library(caret)

#Importing wine data.

wine2 = read.csv('wine2.csv')
head(wine2)

#Converting the class variable into factors.

wine2$cultivars = as.factor(wine2$cultivars)

#The data is partitioned based on 66% train-test split. i.e. 66% of the instances are considered for training set and rest of the them to test set.

set.seed(123)
split = sample.split(wine2$cultivars, SplitRatio = 2/3)
wine2_train = subset(wine2, split == T)
wine2_test = subset(wine2, split == F)

#Both train and test sets are Standardizied in the follwing way.

wine2_train[-1] = scale(wine2_train[-1])
wine2_test[-1] = scale(wine2_test[-1])


#APPLYING LR:

set.seed(123)

#Fitting LR on training data
glm_wine2 = train(cultivars ~ ., data=wine2_train, method='glm')

#The observations are predicted on the test set.

wine2.glm_pred = predict(glm_wine2, wine2_test)

#The model is evaluated based on the confusion matrix and other evalution metrics in the following way.

confusionMatrix(wine2.glm_pred, wine2_test$cultivars, mode = "everything")
```

#APPLYING LDA:

```
library(MASS)
set.seed(123)

#Fitting LDA on training data

lda_wine2 = train(cultivars ~ ., data=wine2_train, method='lda')

#The observations are predicted on the test set.

wine2.lda_pred = predict(lda_wine2, wine2_test)

#The model is evaluated based on the confusion matrix and other evalution metrics in the following way.

confusionMatrix(wine2.lda_pred, wine2_test$cultivars, mode = "everything")
```

#APPLYING NAIVE BAYES:

```
#install.packages('e1071')
library(e1071)
library(klaR)
set.seed(123)

#Fitting Naive Bayes on training data

naive_bayes.wine2 = train(cultivars ~ ., data=wine2_train, method='nb')

#The outcomes of test data are predicted in the following way.

wine2.nb_pred = predict(naive_bayes.wine2, wine2_test)

#The model is evaluated based on the confusion matrix and other evalution metrics as shown below.

confusionMatrix(wine2.nb_pred, wine2_test$cultivars, mode = "everything")
```

#APPLYING SVM:

```
set.seed(123)
```

```
#Fitting SVM on training data

svm.wine2 = train(cultivars ~ ., data=wine2_train, method='svmRadial')

#The outcomes of test data are predicted in the following way.

wine2.svm_pred = predict(svm.wine2, newdata = wine2_test)

#The model is evaluated based on the confusion matrix and other evalution metrics
as shown below.

confusionMatrix(wine2.svm_pred, wine2_test$cultivars, mode = "everything")


#APPLYING RANDOM FOREST:

set.seed(123)

#Fitting random forest on training data

rf.wine2 = train(cultivars ~ ., data=wine2_train, method='rf')

#The outcomes of test data are predicted in the following way.

wine2.rf_pred = predict(rf.wine2, newdata = wine2_test)

#The model is evaluated based on the confusion matrix and other evalution metrics
as shown below.

confusionMatrix(wine2.rf_pred, wine2_test$cultivars, mode = "everything")

#The four classification algorithms are compared based on their Accuracy and Kappa
statistic.

wine2.m6 <- resamples(list(LDA=lda_wine2, NB=naive_bayes.wine2,
SVM=svm.wine2, RF=rf.wine2))

summary(wine2.m6)
scales <- list(x=list(relation="free"), y=list(relation="free"))
bwplot(wine2.m6, scales=scales)
densityplot(wine2.m6, scales=scales, pch = "|")
dotplot(wine2.m6, scales=scales)
parallelplot(wine2.m6)
splom(wine2.m6)
```

```
xyplot(wine2.m6, models=c("SVM", "LDA"))
diffs <- diff(wine2.m6)
summary(diffs)

table = matrix(NA, nrow = 5, ncol = 3)
colnames(table) = c("Models", "Accuracy", "Kappa")
table[1, ] = c("Logistic Regression (LR)", "97.73%", "0.954" )
table[2, ] = c("Linear Discriminant Analysis (LDA)", "95.45%", "0.9076" )
table[3, ] = c("Naive Bayes (NB)", "97.73%", "0.954" )
table[4, ] = c("Support Vector Machine (SVM)", "97.73%", "0.954" )
table[5, ] = c("Random Forest (RF)", "100.00%", "1" )


#install.packages("kableExtra")
library(kableExtra)
kable(table, caption = "Results of wine2 data")
```

7. Breast Cancer data with binary class

```
#Importing cancer data.

wdbc2 = read.csv('wdbc2.csv')

#The data is pre-processed by converting two categories of categorical variable
'diagnosis' into factors. Further they are divided into train and test set.

wdbc2$diagnosis = factor(wdbc2$diagnosis, levels = c('M','B'), labels = c(0,1))

#The data is partitioned based on 66% train-test split. i.e. 66% of the instances are
considered for training set and rest of the them to test set.

set.seed(24)
split = sample.split(wdbc2$diagnosis, SplitRatio = 2/3)
wdbc2_train = subset(wdbc2, split == T)
wdbc2_test = subset(wdbc2, split == F)

#Both train and test sets are Standardizied in the follwing way.

wdbc2_train[, 2:31] = scale(wdbc2_train[, 2:31])
wdbc2_test[, 2:31] = scale(wdbc2_test[, 2:31])


#APPLYING LR:
```

```
set.seed(24)

#Fitting Logistic Regression on training data

glm_wdbc2 = train(diagnosis ~ ., data=wdbc2_train, method='glm')

#The observations are predicted on the test set.

wdbc2.glm_pred = predict(glm_wdbc2, wdbc2_test)

#The model is evaluated based on the confusion matrix and other evalution metrics in the following way.

confusionMatrix(wdbc2.glm_pred, wdbc2_test$diagnosis, mode = "everything")


#APPLYING LDA:

library(MASS)
set.seed(24)

#Fitting LDA on training data

lda_wdbc2 = train(diagnosis ~ ., data=wdbc2_train, method='lda')

#The observations are predicted on the test set.

wdbc2.lda_pred = predict(lda_wdbc2, wdbc2_test)

#The model is evaluated based on the confusion matrix and other evalution metrics in the following way.

confusionMatrix(wdbc2.lda_pred, wdbc2_test$diagnosis, mode = "everything")


#APPLYING NAIVE BAYES:

#install.packages('e1071')
library(e1071)
library(klaR)
set.seed(24)

#Fitting Naive Bayes on training data
```

```
naive_bayes.wdbc2 = train(diagnosis ~ ., data=wdbc2_train, method='nb')

#The outcomes of test data are predicted in the following way.

wdbc2.nb_pred = predict(naive_bayes.wdbc2, wdbc2_test)

#The model is evaluated based on the confusion matrix and other evalution metrics as shown below.

confusionMatrix(wdbc2.nb_pred, wdbc2_test$diagnosis, mode = "everything")


#APPLYING SVM:

set.seed(24)

#Fitting SVM on training data

svm.wdbc2 = train(diagnosis ~ ., data=wdbc2_train, method='svmRadial')

#The outcomes of test data are predicted in the following way.

wdbc2.svm_pred = predict(svm.wdbc2, newdata = wdbc2_test)

#The model is evaluated based on the confusion matrix and other evalution metrics as shown below.

confusionMatrix(wdbc2.svm_pred, wdbc2_test$diagnosis, mode = "everything")


#APPLYING RANDOM FOREST:

set.seed(24)

#Fitting random forest on training data

rf.wdbc2 = train(diagnosis ~ ., data=wdbc2_train, method='rf')

#The outcomes of test data are predicted in the following way.

wdbc2.rf_pred = predict(rf.wdbc2, newdata = wdbc2_test)

#The model is evaluated based on the confusion matrix and other evalution metrics as shown below.
```

```
confusionMatrix(wdbc2.rf_pred, wdbc2_test$diagnosis, mode = "everything")

#The four classification algorithms are compared based on their Accuracy and Kappa statistic.

wdbc2.m7 <- resamples(list(LR=glm_wdbc2, LDA=lda_wdbc2,
NB=naive_bayes.wdbc2, SVM=svm.wdbc2, RF=rf.wdbc2))

summary(wdbc2.m7)

scales <- list(x=list(relation="free"), y=list(relation="free"))
bwplot(wdbc2.m7, scales=scales)
densityplot(wdbc2.m7, scales=scales, pch = "|")
dotplot(wdbc2.m7, scales=scales)
parallelplot(wdbc2.m7)
splom(wdbc2.m7)
xyplot(wdbc2.m7, models=c("SVM", "LDA"))
diffs <- diff(wdbc2.m7)
summary(diffs)

table = matrix(NA, nrow = 5, ncol = 3)
colnames(table) = c("Models", "Accuracy", "Kappa")
table[1, ] = c("Logistic Regression (LR)", "92.63%", "0.8399" )
table[2, ] = c("Linear Discriminant Analysis (LDA)", "95.79%", "0.908" )
table[3, ] = c("Naive Bayes (NB)", "93.68%", "0.8635" )
table[4, ] = c("Support Vector Machine (SVM)", "95.26%", "0.8973" )
table[5, ] = c("Random Forest (RF)", "95.26%", "0.8973" )

#install.packages("kableExtra")
library(kableExtra)
kable(table, caption = "Results of Breast Cancer Wisconsin data")
```

8. Abalone data with binary class

```
#Importing abalone data.

abalone2 = read.csv('Abalone2.csv')
abalone2$sex = NULL
abalone2$rings = NULL
abalone2$groups = factor(abalone2$groups)

#The data is partitioned based on 75% train-test split. i.e. 75% of the instances are considered for training set and rest of the them to test set.
```

```
set.seed(240)
split = sample.split(abalone2$groups, SplitRatio = 3/4)
abalone2_train = subset(abalone2, split == T)
abalone2_test = subset(abalone2, split == F)

#Both train and test sets are Standardizied in the follwing way.

abalone2_train[, 1:7] = scale(abalone2_train[, 1:7])
abalone2_test[, 1:7] = scale(abalone2_test[, 1:7])


#APPLYING LR:

set.seed(240)

#Fitting Logistic Regression on training data

glm_abalone2 = train(groups ~. , data=abalone2_train, method='glm')

#The observations are predicted on the test set.

abalone2.glm_pred = predict(glm_abalone2, abalone2_test)

#The model is evaluated based on the confusion matrix and other evalution metrics
in the following way.

confusionMatrix(abalone2.glm_pred, abalone2_test$groups, mode = "everything")


#APPLYING LDA:

library(MASS)
set.seed(240)

#Fitting LDA on training data

lda_abalone2 = train(groups ~. , data=abalone2_train, method='lda')

#The observations are predicted on the test set.

abalone2.lda_pred = predict(lda_abalone2, abalone2_test)
```

#The model is evaluated based on the confusion matrix and other evalution metrics in the following way.

confusionMatrix(abalone2.lda_pred, abalone2_test$groups, mode = "everything")

#APPLYING NAIVE BAYES:

library(e1071)
library(klaR)
set.seed(240)

#Fitting Naive Bayes on training data

naive_bayes.abalone2 = train(groups ~. , data=abalone2_train, method='nb')

#The outcomes of test data are predicted in the following way.

abalone2.nb_pred = predict(naive_bayes.abalone2, abalone2_test)

#The model is evaluated based on the confusion matrix and other evalution metrics as shown below.

confusionMatrix(abalone2.nb_pred, abalone2_test$groups, mode = "everything")

#APPLYING SVM:

set.seed(240)
svm.abalone2 = train(groups ~. , data=abalone2_train, method='svmRadial')

#The outcomes of test data are predicted in the following way.

abalone2.svm_pred = predict(svm.abalone2, newdata = abalone2_test)

#The model is evaluated based on the confusion matrix and other evalution metrics as shown below.

confusionMatrix(abalone2.svm_pred, abalone2_test$groups, mode = "everything")

#APPLYING RANDOM FOEREST:

set.seed(240)

```
#Fitting random forest on training data

rf.abalone2 = train(groups ~. , data=abalone2_train, method='rf')

#The outcomes of test data are predicted in the following way.

abalone2.rf_pred = predict(rf.abalone2, newdata = abalone2_test)

#The model is evaluated based on the confusion matrix and other evalution metrics
as shown below.

confusionMatrix(abalone2.rf_pred, abalone2_test$groups, mode = "everything")

#The five classification algorithms are compared based on their Accuracy and Kappa
statistic.

abalone2.m8 <- resamples(list(LR=glm_abalone2, LDA=lda_abalone2,
NB=naive_bayes.abalone2, SVM=svm.abalone2, RF=rf.abalone2))
summary(abalone2.m8)

scales <- list(x=list(relation="free"), y=list(relation="free"))
bwplot(abalone2.m8, scales=scales)
densityplot(abalone2.m8, scales=scales, pch = "|")
dotplot(abalone2.m8, scales=scales)
parallelplot(abalone2.m8)
splom(abalone2.m8)
xyplot(abalone2.m8, models=c("SVM", "LDA"))
diffs <- diff(abalone2.m8)
summary(diffs)

table = matrix(NA, nrow = 5, ncol = 3)
colnames(table) = c("Models", "Accuracy", "Kappa")
table[1, ] = c("Logistic Regression (LR)", "68.69%", "0.3742" )
table[2, ] = c("Linear Discriminant Analysis (LDA)", "68.83%", "0.3775" )
table[3, ] = c("Naive Bayes (NB)", "56.13%", "0.1319" )
table[4, ] = c("Support Vector Machine (SVM)", "67.39%", "0.3461" )
table[5, ] = c("Random Forest (RF)", "68.98%", "0.3783" )

#install.packages("kableExtra")
library(kableExtra)
kable(table, caption = "Results of abalone2 data")
```