UNIVERSITY OF CENTRAL MISSOURI

LEARNING TO A GREATER DEGREE

COLLEGE OF HEALTH, SCIENCE AND TECHNOLOGY

DEPARTMENT: COMPUTER SCIENCE

PROJECT :

COMPARISION OF CLASSIFICATION MODEL

(IRIS DATASET)

SUBMITTED TO :

PROF. LIANWEN WANG

SUBMITTED BY:

RISHABH GALKAR (700693771)

JANHVI JOSHI (700687844)

MAMTA SINHA (700658976)

Contribution to the project by each group member

Rishabh:

Dataset selection and cleanup for Project

Background information for all techniques.

Decision tree R code

Powerpoint for background information and Decision Tree .

Mamta:

Dataset selection and cleanup for Project

Insight into data using boxplot and scatter plot.

Background information for all techniques.

Naïve Bayer source code.

Powerpoint for background information, insights of data using boxplot and scatter plot and Naïve Bayer technique.

Janhvi :

Dataset selection and cleanup for Project

Background information for all techniques.

Support Vector Machine R code for all three kernels types.

Powerpoint for background information and Support Vector Machine

TABLE OF CONTENT

Introduction

As we know in data mining, we can generate new information by examining large databases. i.e. It is the process of sorting large data sets to identify patterns and establish relationships to solve problems through data analysis. Data mining tools allow to predict new future trends. The machine learning algorithms are used for extracting knowledge or interesting patterns from unstructured data.

Classification is data mining technique which assigns categories to collection of data for more accurate predictions and analysis of very large datasets. Classification is used for following purpose:

The descriptive data mining tasks characterize the general properties of the data present in the database, while in contrast predictive data mining technique perform inference from the current data for making prediction. This overview briefly introduces these two most important techniques that perform data mining task as Predictive and Descriptive. Between this predictive and descriptive they consist of their own method as Classification, clustering, summarization, association, etc.,

A classification technique is a systematic approach to building classification models from an input data set. Examples include decision tree classifiers, rule-based classifiers, neural networks, support vector machines and naive Bayes classifiers.
Approach to solve classification problem is first, a training set consisting of records whose class labels are known must be provided. The training set is used to build a classification model, which is subsequently applied to the test set, which consists of records with unknown class labels.
Evaluation of the performance of a classification model is based on the counts of test records correctly and incorrectly predicted by the model. These counts are tabulated in a table known as a confusion matrix.

Decision Tree: A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node.

Tree pruning is performed in order to remove anomalies in the training data due to noise or outliers. The pruned trees are smaller and less complex.

There are two approaches to prune a tree −

Pre-pruning − The tree is pruned by halting its construction early.

Post-pruning - This approach removes a sub-tree from a fully grown tree.

Random Forest: Random Forest (RF) is a collection or ensemble model of numerous Decision Trees (DT). Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

Bagging: It is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting.

Naïve Bayes classifier: Bayesian classification is based on Bayes' Theorem. Bayesian classifiers are the statistical classifiers. Bayesian classifiers can predict class membership probabilities such as the probability that a given tuple belongs to a particular class.

According to Bayes' Theorem,

$$P(H/X) = P(X/H)\ P(H) / P(X)$$

Support Vector Machine: In machine learning, support vector machines supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

Support vector machines are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis. However, they are mostly used in classification problems.

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N-the number of features) that distinctly classifies the data points.

To separate the two classes of data points, there are many possible hyperplanes that could be chosen. the objective is to find a plane that has the maximum margin, i.e. the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

Kernel:-

- The kernel defines the similarity or a distance measure between new data and the support vectors.
- The dot product is the similarity measure used for linear SVM or a linear kernel because the distance is a linear combination of the inputs.
- Other kernels can be used that transform the input space into higher dimensions such as a Polynomial Kernel and a Radial Kernel. This is called the Kernel Trick.
- It is desirable to use more complex kernels as it allows lines to separate the classes that are curved or even more complex. This in turn can lead to more accurate classifiers.

Gamma:-

The gamma parameter defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’. The gamma parameters be the inverse of the radius of influence of samples selected by the model as support vectors. If gamma is too large, the radius of the area of influence of the support vectors only includes the support vector itself and no amount of regularization with C will be able to prevent overfitting.

When gamma is very small, the model is too constrained and cannot capture the complexity or “shape” of the data. The region of influence of any selected support vector would include the whole training set. The resulting model will behave similarly to a linear model with a set of hyperplanes that separate the centres of high density of any pair of two classes.

Cost: - The Cost parameter tells the SVM optimization how much you want to avoid misclassifying each training. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

Introduction to Dataset:

The Iris dataset was used in R.A. Fisher's classic 1936 paper, the use of Multiple Measurements in Taxonomic Problems and can also be found in UCI Machine Learning Repository. It includes three iris species namely, Setosa, Virginica, Versicolor, with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

Attribute Information:

1. Sepal length in cm
   2. Sepal width in cm
   3. Petal length in cm
   4. Petal width in cm
   5. Class:
   - Iris Setosa
   - Iris Versicolor
   - Iris Virginica

Iris Dataset has been used for classification problem to predict the Class variable of Iris plant. The Dataset is a Multivariate with real set of attribute characteristics. It consists of 4 Variable with 150 observations for each variable and one Target variable classifying the plant into 3 different species.

Iris Dataset with first 10 Observations:-

	Sepal_len	Sepal_wid	Petal_len	Petal_wid	Class
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
5	4.6	3.4	1.4	0.3	Iris-setosa
6	5.0	3.4	1.5	0.2	Iris-setosa
7	4.4	2.9	1.4	0.2	Iris-setosa
8	4.9	3.1	1.5	0.1	Iris-setosa
9	5.4	3.7	1.5	0.2	Iris-setosa

Figure 1 Iris Dataset

Insights from the Dataset:-

Boxplot for all the attributes has been plotted comparing different species of flowers for all their features.

It is inferred from the attributes of Iris dataset that the species (class) Setosa is showing different behaviour as compared to other two classes.

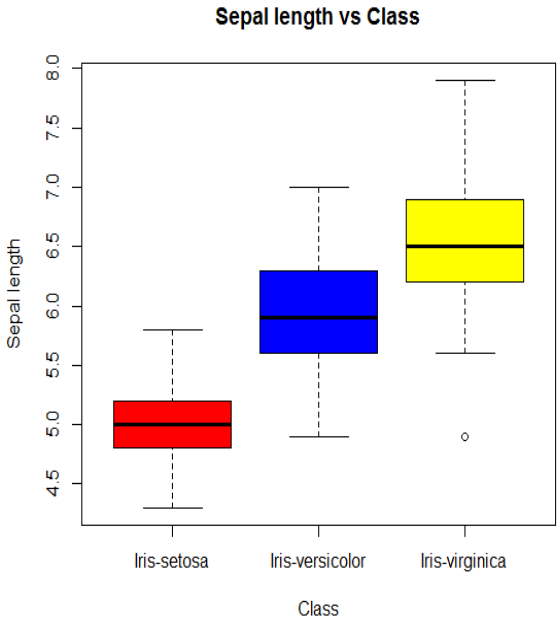


Figure 2 Boxplot Comparing Sepal Length in cm for all Species

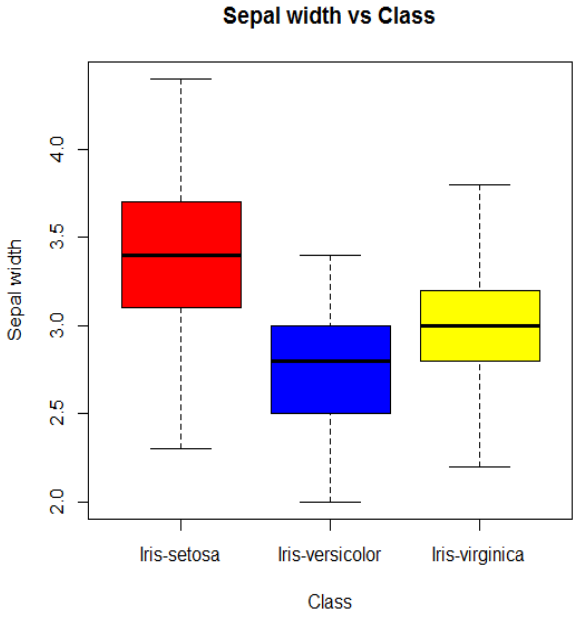
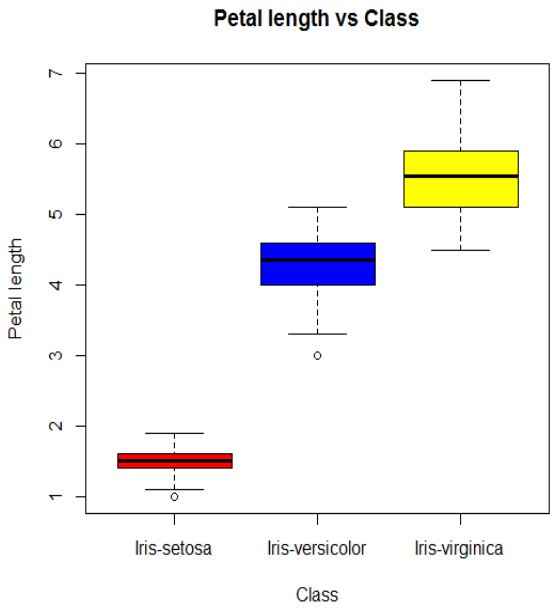


Figure 3 Boxplot Comparing Sepal Width in cm for all Species



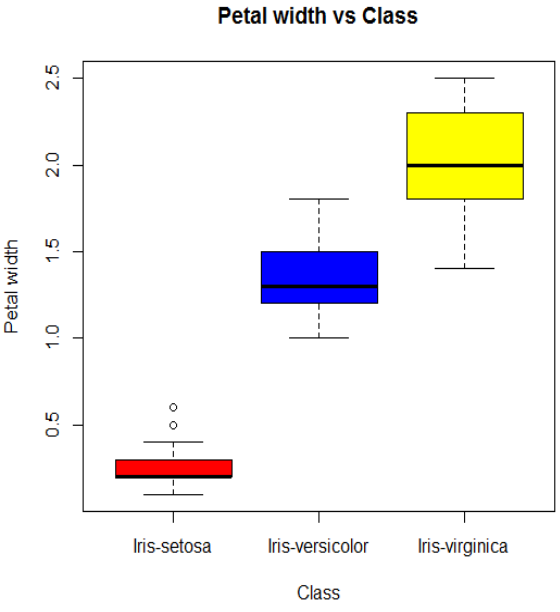


Figure 4 Boxplot Comparing Petal Length in cm for all Species *Figure 5 Boxplot Comparing Petal Width in cm for all Species*

Scatterplot has been plotted comparing Sepal Length and Sepal Width for the 3 Species. It can be inferred from the scatter plots that one flower species (Setosa) is linearly separable from the other two, but the other two are not linearly separable from each other.

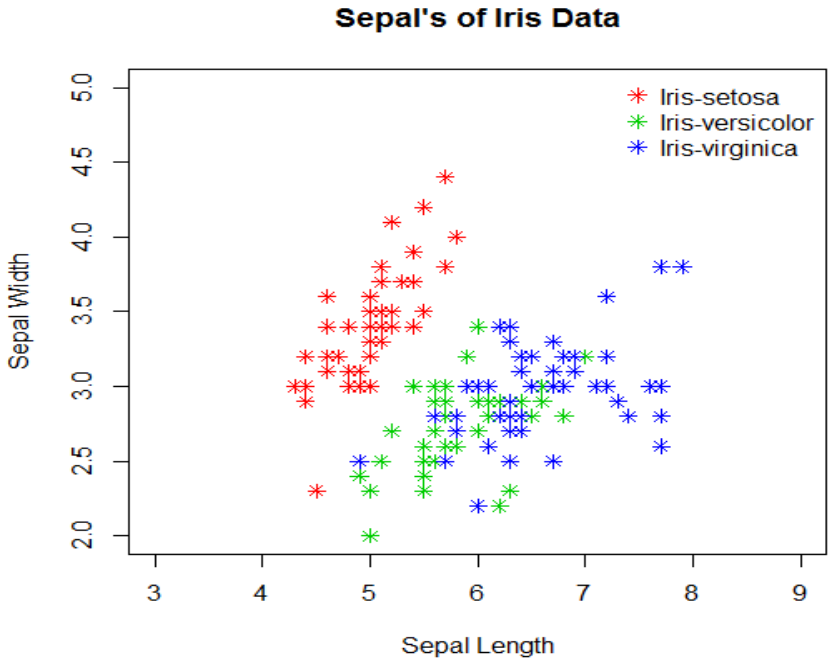


Figure 6 Scatterplot Sepal Width vs Sepal Length

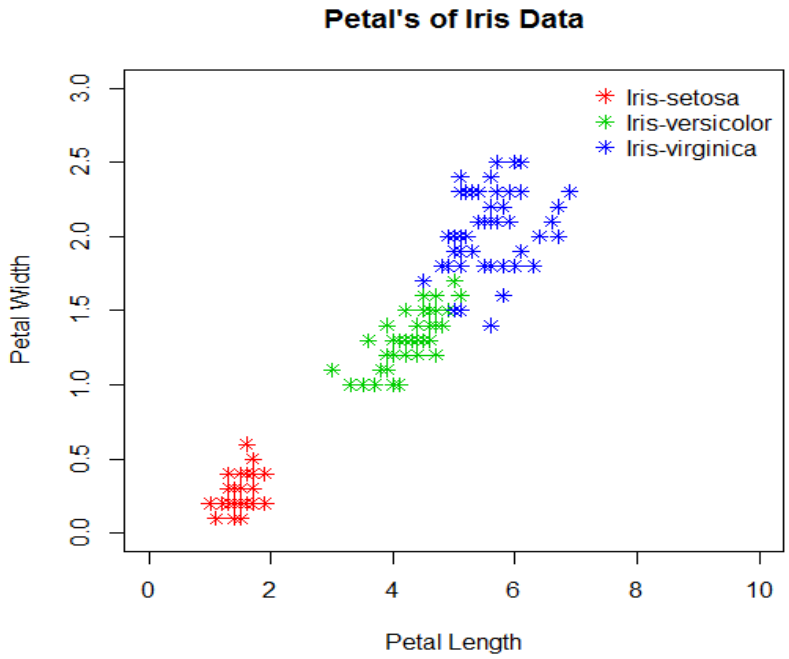


Figure 7 Scatterplot Petal Width vs Petal Length

Results

Decision Tree Using Hold-Out Method:-

We need to hold-out a validation set from the original data. following steps were followed:-

1. Hold-out some of rows of the dataset for testing; use the rest data for training.
2. Build a predictive model using only the training set.
3. Use the test set to compare predicted answers and actual answers.
4. Validate the performance of your predictive model by comparing predictions on test rows.

Table 1. Confusion Matrix for Decision Tree using Hold-Out method.

Predicted	Iris-setosa	Iris-versicolor	Iris-virginica
Actual			
Iris-setosa	13	0	0
Iris-versicolor	0	8	0
Iris-virginica	0	2	11

Calculations:-

Correct Prediction Rate:- $\frac{True\ Positive + True\ negative}{Total\ No.\ of\ Events}$

$:- \frac{13+8+11}{13+8+11+2}$

: - 0.9411

False Prediction Rate:- $\frac{False\ Psitive + False\ Negative}{Total\ No.of\ Events}$

$:- \frac{2}{13+8+11+2}$

: - 0.0588

Decision Tree using bagging: -

Decision trees are sensitive to the specific data on which they are trained. If the training data is changed (e.g. a tree is trained on a subset of the training data) the resulting decision tree can be quite different and in turn the predictions can be quite different. Following Steps were Involved:-

1. Create many random sub-samples of our dataset with replacement.
2. Train a cart model on each sample.
3. Given a new dataset, calculate the average prediction from each model.

Table 2. Confusion Matrix for Decision Tree using Bagging method.

Predicted	Iris-setosa	Iris-versicolor	Iris-virginica
Actual			
Iris-setosa	13	0	0
Iris-versicolor	0	9	0
Iris-virginica	0	1	11

False Prediction Rate: - 0.02941 %

Decision Tree using Random Forest:-

Random forest changes the algorithm for the way that the sub-trees are learned so that the resulting predictions from all the subtrees have less correlation. The number of features that can be searched at each split point (m) must be specified as a parameter to the algorithm. we can try different values and tune it using cross validation.

1. For classification a good default is m = sqrt(p)

Table 3. Confusion Matrix for Decision Tree using Random Forest method.

Predicted	Iris-setosa	Iris-versicolor	Iris-virginica
Actual			
Iris-setosa	13	0	0
Iris-versicolor	0	9	0
Iris-virginica	0	1	11

False Prediction Rate: - 0.02941 %

Naïve Bayer Classifier :

Test data result:

a) For train data = 110 and test data = 39.

Predicted	Iris-setosa	Iris-versicolor	Iris-virginica
Actual			
Iris-setosa	13	0	0
Iris-versicolor	0	11	2
Iris-virginica	0	1	12

Table 4. Confusion Matrix for Naïve Bayer (train data=110)

Correct Prediction Rate = (13+11+12) / (13+11+12+1+2)

= 0.923

False Prediction Rate = (1+2) / (13+11+12+1+2)

= 0.0769

Error rate is 7.69% for Test data.

b) For train data = 100 and test data = 49.

Predicted	Iris-setosa	Iris-versicolor	Iris-virginica
Actual			
Iris-setosa	16	0	0
Iris-versicolor	0	15	2
Iris-virginica	0	1	15

Table 5. Confusion Matrix for Naïve Bayer (train data=100)

Correct Prediction Rate = (16+15+15) / (16+15+15+1+2)

= 0.9387

False Prediction Rate = (1+2) / (16+15+15+1+2)

= 0.06122

Error rate is 6.1% for Test data.

c) For train data = 115 and test data = 34.

Predicted	Iris-setosa	Iris-versicolor	Iris-virginica
Actual			
Iris-setosa	13	0	0
Iris-versicolor	0	9	0
Iris-virginica	0	1	11

Table 6. Confusion Matrix for Naïve Bayer (test data=34)

Correct Prediction Rate = (13+9+11) / (13+9+11+1)

= 0.970588

False Prediction Rate = (1) / (13+9+11+1)

= 0.0294

Error rate is 2.94% for Test data.

Predicted	Iris-setosa	Iris-versicolor	Iris-virginica
Actual			
Iris-setosa	36	0	0
Iris-versicolor	0	38	2
Iris-virginica	0	3	36

Table 7. Confusion Matrix for Naïve Bayer (train data=115)

Correct Prediction Rate = (36+38+36) / (36+38+36+3+2)

= 0.956521

False Prediction Rate = (3+2) / (36+38+36+3+2)

= 0.04347

Error rate is 4.347% for Training data.

SUPPORT VECTOR MACHINE

We have to take two data sets i.e. training data set and test data set

1. from the data set we use 34 rows for the test data and remaining for the train data.

2. Build a predictive model using only the training set with the best cost and gamma.

3. Use the test set to compare predicted answers an actual answer.

4. Validate the performance of your predictive model by comparing predictions on test rows.

By tuning our model. the best cost to train the data set is 1 and also the gamma is 1.

Table 8. Confusion Matrix for Simple vector machine using Linear kernel on Train dataset.

Predicted	Iris-setosa	Iris-versicolor	Iris-virginica
Actual			
Iris-setosa	36	0	0
Iris-versicolor	0	37	0
Iris-virginica	0	3	39

Calculations: - for train dataset

Correct Prediction Rate:- $\frac{True\ Positive + True\ negative}{Total\ No.\ of\ Events}$

:- $\frac{36+37+39}{36+37+39+3}$

: - 0.973913

False Prediction Rate:- $\frac{False\ Psitive + False\ Negative}{Total\ No.of\ Events}$

:- $\frac{3}{36+37+39+3}$

: - 0.02608696

Table 9. Confusion Matrix for Simple vector machine using linear kernel on test dataset.

Predicted	Iris-setosa	Iris-versicolor	Iris-virginica
Actual			
Iris-setosa	13	0	0
Iris-versicolor	0	9	0
Iris-virginica	0	1	11

Calculations: -

Correct Prediction Rate:- $\frac{True\ Positive + True\ negative}{Total\ No.\ of\ Events}$

:- $\frac{13+9+11}{13+9+1+11}$

: - 0.970588

False Prediction Rate:- $\frac{False\ Psitive + False\ Negative}{Total\ No.of\ Events}$

:- $\frac{1}{13+9+1+11}$

: - 0.02941176

Table 10. Confusion Matrix for Simple vector machine using Radial kernel on train data.

Predicted	Iris-setosa	Iris-versicolor	Iris-virginica
Actual			
Iris-setosa	36	0	0
Iris-versicolor	0	39	0
Iris-virginica	0	1	39

Calculations:-

Correct Prediction Rate:- $\frac{True\ Positive + True\ negative}{Total\ No.\ of\ Events}$

:- $\frac{36+39+39}{36+39+39+1}$

: - 0.991304

False Prediction Rate:- $\frac{False\ Psitive + False\ Negative}{Total\ No.of\ Events}$

:- $\frac{1}{36+39+39+1}$

: - 0.008695652

Table 11. Confusion Matrix for Simple vector machine using Radial kernel on test data.

Predicted	Iris-setosa	Iris-versicolor	Iris-virginica
Actual			
Iris-setosa	12	0	0
Iris-versicolor	0	9	0
Iris-virginica	1	1	11

Calculations:-

Correct Prediction Rate:- $\frac{True\ Positive + True\ negative}{Total\ No.\ of\ Events}$

:- $\frac{12+9+11}{12+1+9+1+11}$

: - 0.941176

False Prediction Rate:- $\frac{False\ Psitive + False\ Negative}{Total\ No.of\ Events}$

:- $\frac{2}{12+1+9+1+11}$

: - 0.0588235

Simple vector machine using Polynomial kernel When degree=3 which gives the minimum error.

Table 12. Confusion Matrix for Simple vector machine using Polynomial kernel on train dataset.

Predicted	Iris-setosa	Iris-versicolor	Iris-virginica
Actual			
Iris-setosa	36	0	0
Iris-versicolor	0	39	0

Iris-virginica	0	1	39

Calculations:-

Correct Prediction Rate:- $\frac{True\ Positive + True\ negative}{Total\ No.\ of\ Events}$

:- $\frac{36+39+39}{36+39+39+1}$

: - 0.991304

False Prediction Rate:- $\frac{False\ Psitive + False\ Negative}{Total\ No.of\ Events}$

:- $\frac{1}{36+39+39+1}$

: - 0.008695652

Table 13. Confusion Matrix for Simple vector machine using Polynomial kernel on test dataset.

Predicted	Iris-setosa	Iris-versicolor	Iris-virginica
Actual			
Iris-setosa	13	0	0
Iris-versicolor	0	9	1
Iris-virginica	0	1	10

Calculations:-

Correct Prediction Rate:- $\frac{True\ Positive + True\ negative}{Total\ No.\ of\ Events}$

:- $\frac{13+9+10}{13+1+9+1+10}$

: - 0.941176

False Prediction Rate:- $\frac{False\ Psitive + False\ Negative}{Total\ No.of\ Events}$

$$:- \frac{2}{13+1+9+1+10}$$

: - 0.0588235

According to the error rate simple vector machine with linear kernel gives the minimum error for train data which is 0.869%.

According to the error rate simple vector machine with linear kernel gives the minimum error for test data which is 2.9%.

Comparison of multiple classification techniques

- We have chosen train data = 115 and test data = 34 here.
- Decision Tree : In decision tree technique, we are getting minimum error (test data) for bagging and random forest which is 2.9%.
- Naïve Bayer : We are getting error rate of 2.9% in Naïve Bayer technique on test data.
- SVM : In SVM on test data among linear, radial and polynomial kernel , we are getting minimum error rate of 2.9 % .
- On comparing error rates of all three techniques, we are getting same test error rate for all three techniques which is 2.9% for Iris dataset.

Conclusion of project

- On comparing error rates of all three techniques, we are getting same error rate for test data for all three techniques which is 2.9% for Iris dataset.
- With all three different techniques of classification, we are getting same error rates.

Appendix of R source codes

```
install.packages("Hmisc")

library(Hmisc)


data=read.csv("D:DataMiningAss/project/iris.data",header=T, na.strings ="")

fix(data)

cleandata <- data[complete.cases(data),]

##Insights into Iris Dataset

##Renaming column name for Iris dataset

library("plyr")

dataset = rename(cleandata, c("X5.1"="sepal_len", "X3.5"="sepal_wid", "X1.4"="petal_len", "X0.2"="petal_wid", "Iris.setosa"="class"  ))

fix(dataset)

##Boxplot for Sepal Length

boxplot(dataset$sepal_len ~ dataset$class, xlab = "Class", ylab = "Sepal length", main = "Sepal length vs Class", col=c("red","blue","yellow"), border= "black" )

##Boxplot for Sepal Width

boxplot(dataset$sepal_wid ~ dataset$class, xlab = "Class", ylab = "Sepal width", main = "Sepal width vs Class", col=c("red","blue","yellow"), border= "black" )

##Boxplot for Petal Length

boxplot(dataset$petal_len ~ dataset$class, xlab = "Class", ylab = "Petal length", main = "Petal length vs Class", col=c("red","blue","yellow"), border= "black" )
```

```
##Boxplot for Petal Width

boxplot(dataset$petal_wid ~ dataset$class, xlab = "Class", ylab = "Petal width", main =
"Petal width vs Class", col=c("red","blue","yellow"), border= "black" )

## Scatter plot for Sepal length and width

plot(dataset$sepal_len , dataset$petal_wid , main = "Sepal's of Iris Data",

xlab = "Sepal Length", ylab = "Sepal Width")

## fill colour into sepal length and width scatter plot

mycolor = c("red","green3","blue")[as.factor(dataset$class)]

## Scatter plot for Sepal length and width

plot(dataset$sepal_len, dataset$sepal_wid , pch = 8, col = mycolor, main = "Sepal's of Iris
Data", xlab = "Sepal Length", ylab = "Sepal Width", xlim = c(3,9), ylim= c(2,5))

legend('topright', legend = unique(dataset$class), col = c("red","green3","blue"),

pch = 8, bty = 'n')

## Scatter plot for Petal length and width

plot(dataset$petal_len, dataset$petal_wid , pch = 8, col = mycolor, main = "Petal's of Iris
Data", xlab = "Petal Length", ylab = "Petal Width", xlim = c(0,10), ylim= c(0,3))

legend('topright', legend = unique(dataset$class), col = c("red","green3","blue"),

pch = 8, bty = 'n')

library(tree)

attach(dataset)

set.seed(123)

datatrain=sample(1:nrow(dataset),115)

train=dataset[datatrain,]

test=dataset[-datatrain,]
```

```
library(e1071)

## DECISION TREE :

#To make a decision tree that classify the Target attribute based on all other attributes

tree.mydata=tree(class~.,dataset)

summary(tree.mydata)

plot(tree.mydata)

text(tree.mydata,pretty=0)

tree.mydata

#The predict() function is used to test the model. The argument

#type="class" instructs R to return the actual class prediction

tree.pred=predict(tree.mydata,type="class")

#Create the confusion matrix

table(tree.pred,class)

#Correct prediction rate

mean(tree.pred==class)

#Error prediction rate

mean(tree.pred!=class)

#Random select a sample of 115 observations of the data set as a training set and the rest

#of the data set as a test set.

set.seed(123)

train=sample(1:nrow(dataset), 115)

mydata.test=dataset[-train,]
```

```
class.test=class[-train]
tree.mydata=tree(class~.,dataset,subset=train)
tree.pred=predict(tree.mydata,mydata.test,type="class")
table(tree.pred,class.test)
mean(tree.pred!=class.test)

#Bagging and Random Forests
install.packages('randomForest')
library(randomForest)
#Some arguments of function randomForest():
#mtry: the number of variables randomly sampled as candidates at each split.
#ntree: the number of trees to grow.

#ntree=500 indicates that 500 trees are generated by bagging
#mtry=10 indicates that all 10 variables are used at each split.
set.seed(123)
tree.mydata=randomForest(class~.,dataset,subset=train, ntree=500,mtry=4)
tree.pred=predict(tree.mydata,mydata.test,type="class")
table(tree.pred,class.test)
mean(tree.pred!=class.test)
#Check with 1500 trees
set.seed(123)
tree.mydata=randomForest(class~.,dataset,subset=train, ntree=1500,mtry=4)
```

```
tree.pred=predict(tree.mydata,mydata.test,type="class")

table(tree.pred,class.test)

mean(tree.pred!=class.test)

#By default, randomForest() uses about sqrt(p) variables when building a random forest of

classification trees. sqrt(4)=2.

set.seed(123)

tree.mydata=randomForest(class~.,dataset,subset=train, ntree=500,mtry=2)

tree.pred=predict(tree.mydata,mydata.test,type="class")

table(tree.pred,class.test)

mean(tree.pred!=class.test)

set.seed(123)

tree.mydata=randomForest(class~.,dataset,subset=train, ntree=500,mtry=3)

tree.pred=predict(tree.mydata,mydata.test,type="class")

table(tree.pred,class.test)

mean(tree.pred!=class.test)

##NAIVE BAYER :

library(e1071)

set.seed(123)

datatrain=sample(1:nrow(dataset),115)

train=dataset[datatrain,]

test=dataset[-datatrain,]

test.label = class[-datatrain]

NB_model = naiveBayes(class~. , data = train)
```

```
NB_Prediction = predict( NB_model , test)

table(NB_Prediction, test.label)

mean(NB_Prediction!=test.label)

##for train data

train.pred = predict(NB_model, train)

table(train$class, train.pred)

train.label = class[datatrain]

mean( train.pred != train.label )

##SUPPORT VECTOR MACHINE :

set.seed(123)

datatrain=sample(1:nrow(dataset),115)

train=dataset[datatrain,]

test=dataset[-datatrain,]

cost=10^(-3:5)

gamma=(1:5)

tune.out=tune(svm,class~.,data=train,kernel="linear",ranges=list(gamma=gamma,cost=cost))

tune.out$best.model

summary(tune.out)

summary(tune.out$best.model)

##Linear kernel

set.seed(123)

svm.linear=svm(class~.,data=train, kernel="linear",gamma=tune.out$best.parameter$gamma,
cost=tune.out$best.parameter$cost)
```

```
summary(svm.linear)

##train

train.pred = predict(svm.linear,train)

train.table = table(Predict = train.pred,Truth = train$class)

print(train.table)

mean(train.pred!=train$class)

##test

test.pred = predict(svm.linear,test)

test.table = table(Predict = test.pred,Truth = test$class)

print(test.table)

mean(test.pred!=test$class)

##radial

set.seed(123)

svm.radial=svm(class~.,data=train, kernel="radial",gamma=tune.out$best.parameter$gamma,
cost=tune.out$best.parameter$cost)

summary(svm.radial)

##train

train.pred = predict(svm.radial,train)

train.table = table(Predict = train.pred,Truth = train$class)

print(train.table)

mean(train.pred!=train$class)

##test

test.pred = predict(svm.radial,test)
```

```
test.table = table(Predict = test.pred, Truth = test$class)

print(test.table)

mean(test.pred!=test$class)

##poly

set.seed(123)

degree=(1:10)

svm.poly = svm(class~., data =train, kernel =
"polynomial",ranges=list(degree=degree),gamma=tune.out$best.parameter$gamma,cost=tune
.out$best.parameter$cost)

summary(svm.poly)

##train

train.pred = predict(svm.poly,train)

train.table = table(Predict = train.pred,Truth = train$class)

print(train.table)

mean(train.pred!=train$class)

##test

test.pred = predict(svm.poly,test)

test.table = table(Predict = test.pred, Truth = test$class)

print(test.table)

mean(test.pred!=test$class)
```