**Exp.No.9. CONTENT BEYOND SYLLABUS**

**RANDOM FOREST APPROACH FOR CLASSIFICATION**

**Aim**

To implement Radom Forest approach for classification for iris data set.

**Theory**

Random forest approach is supervised nonlinear classification and regression algorithm. Classification is a process of classifying a group of datasets in categories or classes. As random forest approach can use classification or regression techniques depending upon the user and target or categories needed. A random forest is a collection of decision trees that specifies the categories with much higher probability. Random forest approach is used over decision trees approach as decision trees lack accuracy and decision trees also show low accuracy during the testing phase due to the process called over-fitting. In R programming, randomForest() function of randomForest package is used to create and analyze the random forest.

**Program**:

# Loading data

data(iris)

# Structure

str(iris)

#Installing package

install.packages("caTools") # For sampling the dataset install.packages("randomForest") # For implementing random forest algorithm # Loading package

library(caTools) library(randomForest)

# Splitting data in train and test data split <- sample.split(iris, SplitRatio = 0.7) train <- subset(iris, split == "TRUE")

test <- subset(iris, split == "FALSE")

# Fitting Random Forest to the train dataset

set.seed(120) # Setting seed

classifier\_RF = randomForest(x = train[-5],

y = train$Species, ntree = 500)

classifier\_RF

# Predicting the Test set results

y\_pred = predict(classifier\_RF, newdata = test[-5])

# Confusion Matrix

confusion\_mtx = table(test[, 5], y\_pred)

confusion\_mtx

# Plotting model plot(classifier\_RF) # Importance plot

importance(classifier\_RF)

# Variable importance plot

varImpPlot(classifier\_RF)

**OUTPUT**:

'data.frame': 150 obs. of 5 variables:

$ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...

$ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...

$ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...

$ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...

$ Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...

[1] TRUE TRUE FALSE TRUE FALSE

Call:

randomForest(x = train[-5], y = train$Species, ntree = 500) Type of random forest: classification

Number of trees: 500 No. of variables tried at each split: 2 OOB estimate of error rate: 5.56% Confusion matrix:

Setosa

versicolor

virginica

class.error

setosa 30

0

0

0.00000000

versicolor 0

28

2

0.06666667

virginica 0

y\_pred

setosa

3

versicolor

27

virginica

0.10000000

setosa 20

0

0

versicolor 0

19

1

virginica 0

2

18

MeanDecreaseGini Sepal.Length 6.201739

Sepal.Width

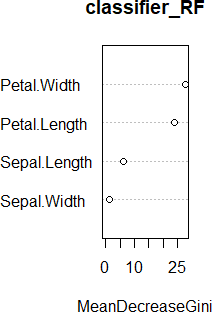
1.527756

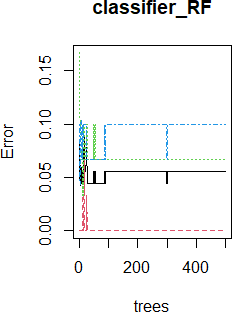
Petal.Length

23.936397

Petal.Width

27.591441





**Result**

Thus, the implementation of random forest approach for classification was executed.

K-Nearest Neighbor Classifier

Aim

To implement K-Nearest Neighbor Classifier for iris data set.

Theory

K-Nearest Neighbor or K-NN is a Supervised Non-linear classification algorithm. K-NN is a Non-parametric algorithm i.e it doesn’t make any assumption about underlying data or its distribution. It is one of the simplest and widely used algorithm which depends on it’s k value(Neighbors) and finds it’s applications in many industries like finance industry, healthcare industry etc.

In the KNN algorithm, K specifies the number of neighbors and its algorithm is as follows:

Choose the number K of neighbor.

Take the K Nearest Neighbor of unknown data point according to distance.

Among the K-neighbors, Count the number of data points in each category.

Assign the new data point to a category, where you counted the most neighbors.

For the Nearest Neighbor classifier, the distance between two points is expressed in the form of Euclidean Distance.

Program

# Loading data

data(iris)

# Structure

str(iris)

# Installing Packages install.packages("e1071") install.packages("caTools") install.packages("class")

# Loading package library(e1071) library(caTools) library(class)

# Loading data data(iris) head(iris)

# Splitting data into train and test data

split <- sample.split(iris, SplitRatio = 0.7)

train\_cl <- subset(iris, split == "TRUE") test\_cl <- subset(iris, split == "FALSE") # Feature Scaling

train\_scale <- scale(train\_cl[, 1:4]) test\_scale <- scale(test\_cl[, 1:4])

# Fitting KNN Model to training dataset

classifier\_knn <- knn(train = train\_scale,

test = test\_scale,

cl = train\_cl$Species, k = 1)

classifier\_knn

# Confusiin Matrix

cm <- table(test\_cl$Species, classifier\_knn) cm

# Model Evaluation - Choosing K # Calculate out of Sample error

misClassError <- mean(classifier\_knn != test\_cl$Species) print(paste('Accuracy =', 1-misClassError))

# K = 3

classifier\_knn <- knn(train = train\_scale,

test = test\_scale,

cl = train\_cl$Species, k = 3)

misClassError <- mean(classifier\_knn != test\_cl$Species) print(paste('Accuracy =', 1-misClassError))

# K = 5

classifier\_knn <- knn(train = train\_scale,

test = test\_scale,

cl = train\_cl$Species, k = 5)

misClassError <- mean(classifier\_knn != test\_cl$Species) print(paste('Accuracy =', 1-misClassError))

# K = 7

classifier\_knn <- knn(train = train\_scale,

test = test\_scale,

cl = train\_cl$Species, k = 7)

misClassError <- mean(classifier\_knn != test\_cl$Species) print(paste('Accuracy =', 1-misClassError))

# K = 15

classifier\_knn <- knn(train = train\_scale,

test = test\_scale,

cl = train\_cl$Species, k = 15)

misClassError <- mean(classifier\_knn != test\_cl$Species) print(paste('Accuracy =', 1-misClassError))

# K = 19

classifier\_knn <- knn(train = train\_scale,test = test\_scale,cl = train\_cl$Species, k = 19)

misClassError <- mean(classifier\_knn != test\_cl$Species) print(paste('Accuracy =', 1-misClassError))

OUTPUT:

'data.frame': 150 obs. of 5 variables:

$ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...

$ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...

$ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...

$ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...

$ Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 …

Sepal.Length Sepal.Width Petal.Length Petal.Width

1 5.1 3.5 1.4 0.2

2 4.9 3.0 1.4 0.2

3 4.7 3.2 1.3 0.2

4 4.6 3.1 1.5 0.2

5 5.0 3.6 1.4 0.2

6 5.4 3.9 1.7 0.4

Species

setosa

setosa

setosa

setosa

setosa

setosa

| [1] setosa | setosa | setosa | setosa |
| --- | --- | --- | --- |
| [5] setosa | setosa | setosa | setosa |
| [9] setosa | setosa | setosa | setosa |
| [13] setosa | setosa | setosa | setosa |
| [17] setosa | setosa | setosa | setosa |

[21] versicolor versicolor versicolor versicolor

[25] versicolor versicolor versicolor versicolor

[29] virginica versicolor versicolor versicolor

[33] versicolor virginica versicolor versicolor

[37] versicolor versicolor versicolor versicolor

[41] virginica virginica virginica virginica

[45] virginica virginica virginica virginica

[49] virginica virginica virginica virginica

[53] virginica versicolor virginica virginica

[57] virginica virginica virginica virginica Levels: setosa versicolor virginica

| classifier\_knn |  | |
| --- | --- | --- |
| **setosa** | **versicolor** | **virginica** |
| **setosa** 20 | 0 | 0 |
| **versicolor** 0 | 18 | 2 |
| **virginica** 0 | 1 | 19 |

[1] "Accuracy = 0.95"

[1] "Accuracy = 0.95"

[1] "Accuracy = 0.966666666666667"

[1] "Accuracy = 0.983333333333333"

[1] "Accuracy = 0.966666666666667"

Result

Thus, the K-NN classifier using iris data set was executed.

Naive Bayes Classifier

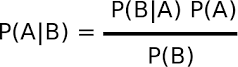
**Aim**

To implement Navie Bayes Classifier for iris dataset.

**Theory**

Naive Bayes is a Supervised Non-linear classification algorithm in [R Programming](https://www.geeksforgeeks.org/introduction-to-r-programming-language/). Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Baye’s theorem with strong(Naive) independence assumptions between the features or variables. The Naive Bayes algorithm is called “Naive” because it makes the assumption that the occurrence of a certain feature is independent of the occurrence of other features.

Naive Bayes algorithm is based on Bayes theorem. Bayes theorem gives the conditional probability of an event A given another event B has occurred.



**where,**

P(A|B) = Conditional probability of A given B. P(B|A) = Conditional probability of B given A. P(A) = Probability of event A.

P(B) = Probability of event B.

**Program**

#### **# Loading data**

data(iris)

#### **# Structure**

str(iris)

**# Installing Packages** install.packages("e1071") install.packages("caTools") install.packages("caret")

**# Loading package** library(e1071)

library(caTools)

library(caret)

**# Splitting data into train and test data** split <- sample.split(iris, SplitRatio = 0.7) train\_cl <- subset(iris, split == "TRUE") test\_cl <- subset(iris, split == "FALSE") **# Feature Scaling**

train\_scale <- scale(train\_cl[, 1:4])

test\_scale <- scale(test\_cl[, 1:4])

#### **# Fitting Naive Bayes Model to training dataset**

set.seed(120) # Setting Seed

classifier\_cl <- naiveBayes(Species ~ ., data = train\_cl) classifier\_cl

#### **# Predicting on test data'**

y\_pred <- predict(classifier\_cl, newdata = test\_cl)

#### **# Confusion Matrix**

cm <- table(test\_cl$Species, y\_pred) cm

#### **# Model Evaluation**

confusionMatrix(cm)

### **OUTPUT:**

**'data.frame':** 150 obs. of 5 variables:

**$ Sepal.Length**: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...

**$ Sepal.Width** : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...

**$ Petal.Length:** num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...

**$ Petal.Width** : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...

**$ Species :** Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...

#### **Naive Bayes Classifier for Discrete Predictors Call:**

naiveBayes.default(x = X, y = Y, laplace = laplace)

#### **A-priori probabilities: Y**

**setosa versicolor virginica**

0.3333333 0.3333333 0.3333333

**Conditional probabilities: Sepal.Length**

### **Y [,1] [,2]**

**setosa** 4.943333 0.3766306

**versicolor** 6.000000 0.5051459

**virginica** 6.500000 0.6817827

**Sepal.Width**

### **Y [,1] [,2]**

**setosa** 3.400000 0.3859605

**versicolor** 2.746667 0.3104317

**virginica** 2.926667 0.3362402

**Petal.Length**

### **Y [,1] [,2]**

**setosa** 1.426667 0.1552158

**versicolor** 4.306667 0.5172429

**virginica** 5.486667 0.6179685

**Petal.Width**

### **Y [,1] [,2]**

**setosa** 0.250000 0.1196259

**versicolor** 1.330000 0.2019730

**virginica** 1.976667 0.2160513

#### **y\_pred**

**setosa versicolor virginica**

**setosa** 20 0 0

**versicolor** 0 19 1

**virginica** 0 2 18

#### **Confusion Matrix and Statistics y\_pred**

| **setosa** | **setosa**  20 | **versicolor**  0 | **virginica**  0 | |
| --- | --- | --- | --- | --- |
| **versicolor** 0 | | 19 | 1 |  |
| **virginica** 0 | | 2 |  | 18 |

**Overall Statistic:**

Accuracy : 0.95

95% CI : (0.8608, 0.9896)

No Information Rate : 0.35

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.925 Mcnemar's Test P-Value : NA

**Statistics by Class:**

**Class: setosa Class: versicolor**

| **Sensitivity** | 1.0000 | | 0.9048 | |
| --- | --- | --- | --- | --- |
| **Specificity** | 1.0000 | | 0.9744 | |
| **Pos Pred Value** | 1.0000 | | 0.9500 | |
| **Neg Pred Value** | 1.0000 | | 0.9500 | |
| **Prevalence** | 0.3333 | | 0.3500 | |
| **Detection Rate** | 0.3333 | | 0.3167 | |
| **Detection Prevalence** | | 0.3333 | | 0.3333 |
| **Balanced Accuracy**  **Sensitivity** | | 1.0000  **Class: virginica**  0.9474 | | 0.9396 |
| **Specificity** | | 0.9512 | |  |
| **Pos Pred Value** | | 0.9000 | |  |
| **Neg Pred Value** | | 0.9750 | |  |
| **Prevalence** | | 0.3167 | |  |
| **Detection Rate** | | 0.3000 | |  |
| **Detection Prevalence** | | 0.3333 | |  |

**Balanced Accuracy** 0

**Result**

Thus, the implementation of Navie Bayes theorem for iris data set was executed.