**Program 1:**

**Program to Implement Linear Regression using Salary Data.**

**In this Program we create a custom Data Frame consisting of Experience and Salary where Experience is an Independent Variable and Salary is a Dependent Variable and we use the Experience Data to predict the Salary of the Employees. Thus we predict Salary using the Experience. The Experience 1.1, 1.5, 3.0 means an experience of 1 Year and 1 month, 1 Year and 5 Months and 3 Years respectively*.***

*#Creating data frame with exp and salary*

exp=c(1.1,1.3,1.5,2.0,2.2,2.9,3.0,3.2,3.2,3.7)

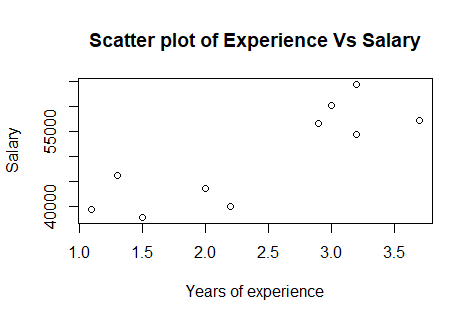
salary=c(39343.00,46205.00,37731.00,43525.00,39891.00,56642.00,60150.00,54445.00,64445.00,57189.00)

data=data.frame(exp,salary)

*#Creating scatter plot of the given dataset*

plot(data$exp, data$salary, xlab="Years of Experience”, ylab = “Salary”,main = “Scatter plot of Experience Vs Salary”)

**Output :**



*#Implementing simple linear regression*

install.packages('caTools')

library(caTools)

*#Creating training set and test set*

split=sample.split(data$salary,SplitRatio = 0.7)

trainingset=subset(data,split==TRUE)

testset=subset(data,split==FALSE)

#Printing training set and test set

Trainingset

testset

*#Creating simple linear regression model using lm() function and assigning the model to the variable LM*

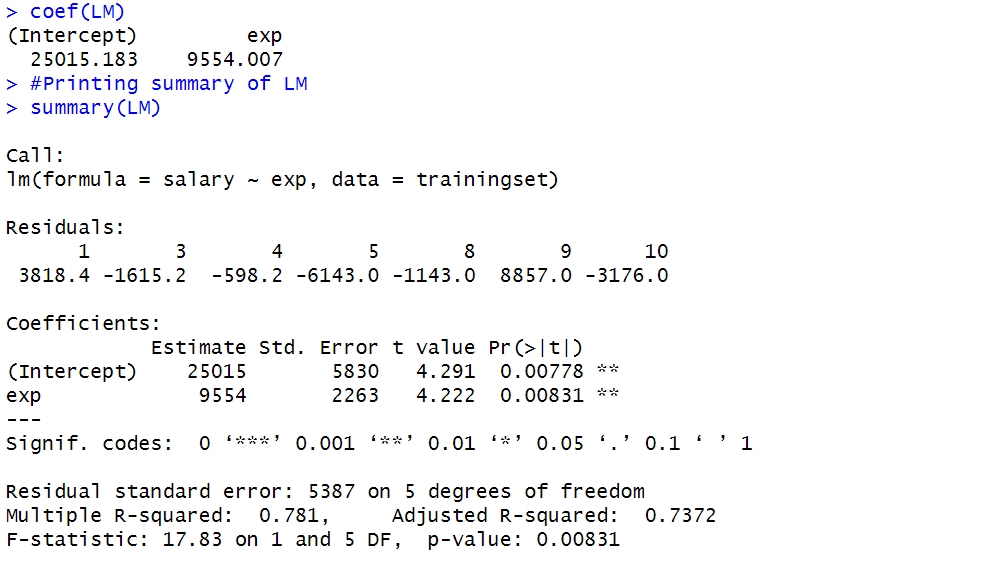
LM=lm(formula = salary~exp,data = trainingset)

*#Getting coefficient values(slope and intercept) for prediction*

coef(LM)

summary(LM)

**Output :**

****

*#Prediction using the test set*

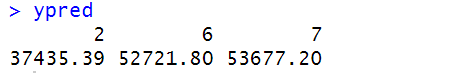
*#We train a model on one subset of the data(training set) and evaluate its performance on another unseen subset(test set),*

*#which helps assess how well the model generalizes to new, unseen data.*

ypred=predict(LM,newdata = testset)

ypred

**Output :**

****

*#Visualizing the Training set results*

install.packages("ggplot2")

library(ggplot2)

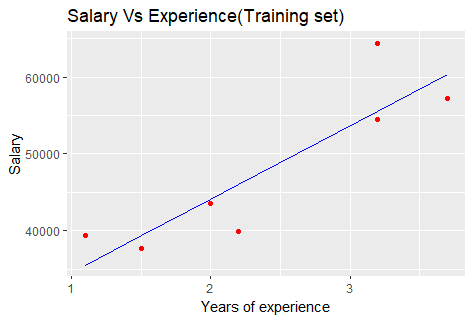
ggplot()+geom\_point(aes(x=trainingset$exp,y=trainingset$salary),color='red') +

geom\_line(aes(x=trainingset$exp,y=predict(LM,newdata=trainingset)),color='blue') + ggtitle('Salary Vs Experience(Training set)') +

xlab('Years of experience') +

ylab('Salary')

**Output :**



*#Visualizing the Test set*

ggplot()+geom\_point(aes(x=testset$exp,y=testset$salary),color='red') +

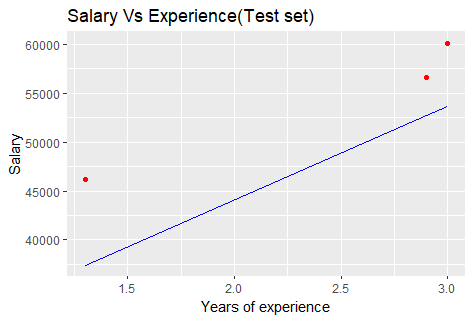
geom\_line(aes(x=testset$exp,y=predict(LM,newdata=testset)),color='blue') +

ggtitle('Salary Vs Experience(Test set)') +

xlab('Years of experience') +

ylab('Salary')

**Output :**

****

**Program 2:**

**Program to Implement Linear Regression using Age and Height Data.**

**In this program we use Age and Height Data where we have an Age as well as the Height of a particular person. Here Age is an Independent Variable and Height is an Dependent Variable which is dependent on Age. We use Linear Regression to establish a Linear Relationship between Age and Height and predict the Height using the Age.**

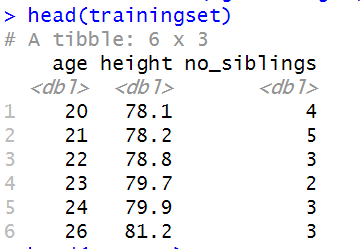
#Importing readxl library to read Excel files

library(readxl)

ageandheight<-read\_excel("ageandheight.xls")

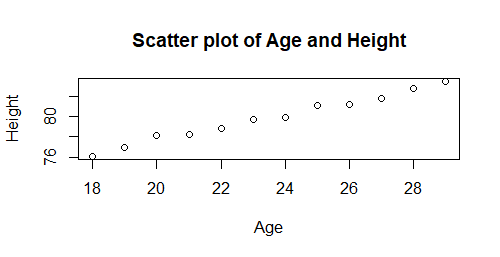
head(ageandheight)

**Output :**

****

plot(ageandheight$age,ageandheight$height,xlab= "Age", ylab = "Height",main = 'Scatter plot of Age and Height')

**Output :**

****

*#Installing caTools package for splitting the data into Training set and Test set*

install.packages('caTools')

library(caTools)

*#Spliting for training and test*

split=sample.split(ageandheight$height,SplitRatio = 0.7)

trainingset=subset(ageandheight,split==TRUE)

testset=subset(ageandheight,split==FALSE)

head(trainingset)

head(testset)

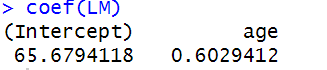
*#creating a Linear Regression Model*

LM=lm(formula=height~age,data=trainingset)

*#getting coefficient values for interpt & slope values for predition*

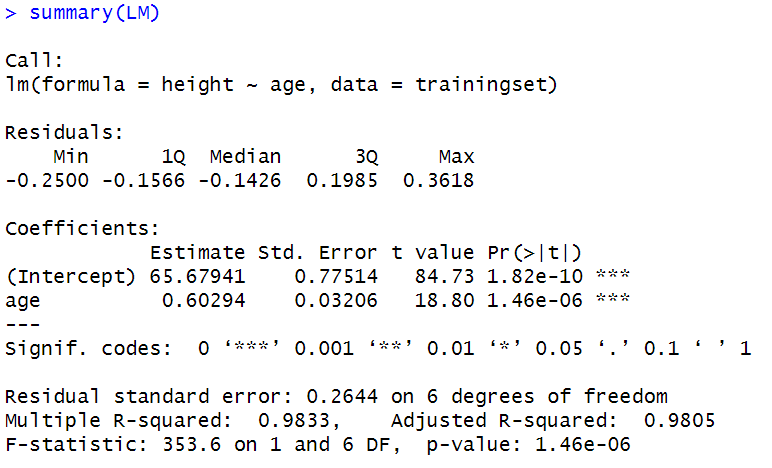
coef(LM )

**Output :**

****

summary(LM)

**Output :**

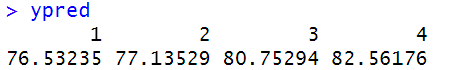
****

*#Predicting the testset result*

ypred=predict(LM,newdata=testset)

ypred

**Output :**

****

install.packages(ggplot2)

library(ggplot2)

*#ploting using ggplot*

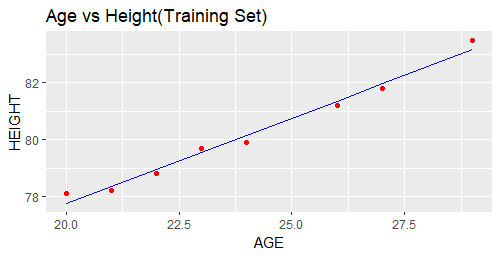
ggplot()+geom\_point(aes(x=trainingset$age,y=trainingset$height),colour='red')+geom\_line(aes(x=trainingset$age,y=predict(LM,newdata = trainingset)),colour='blue')+

ggtitle('Age vs Height(Training Set)')+

xlab('AGE')+

ylab('HEIGHT')

**Output :**

****

*#Visualising the test set results*

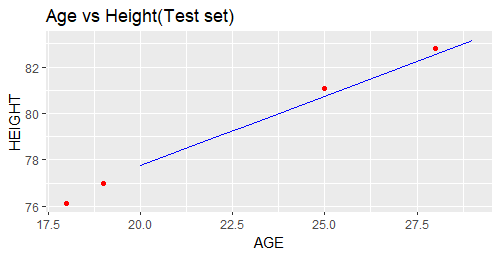
ggplot()+geom\_point(aes(x=testset$age,y=testset$height),colour='red')+

geom\_line(aes(x=trainingset$age,y=predict(LM,newdata =trainingset)),colour='blue')+

xlab('AGE')+

ylab('HEIGHT')

**Output :**

****

**Program 3:**

**Program to Implement Linear Regression using mtCars Dataset.**

**The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models). In this Data Mpg is a dependent Variable which is dependent on other variables such as Cylinders, displacement, HP, weight, etc. In this program we Hp as an Independent Variable to predict the Mpg (fuel consumption) of a Particular Model of the Car.**

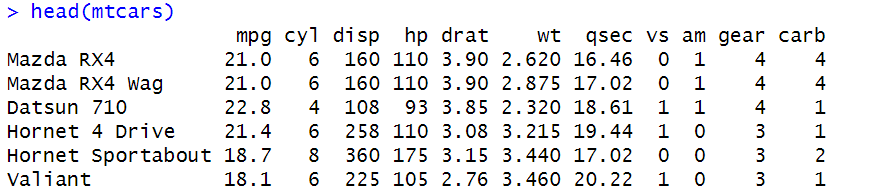
*#importing mtcars data for predicting the miles per gallons (mpg) using Horse Power (HP) of Each Cars*

data(mtcars)

*#Veiwing the first five observations of the dataset*

head(mtcars)

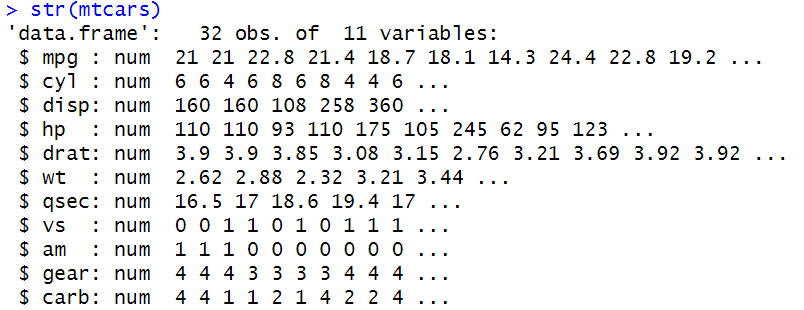
**Output :**

****

*#Observing the structure of the Data*

str(mtcars)

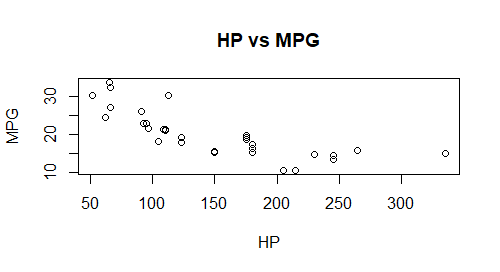
**Output :**

****

*#plotting the scatter plot of HP vs MPG*

plot(mtcars$hp, mtcars$mpg, xlab = "HP", ylab = "MPG", main = "HP vs MPG")

**Output :**

****

*#Splitting the Dataset into training and testing set*

library(caTools)

split = sample.split(mtcars$hp, SplitRatio = 0.7)

train = subset(mtcars, split == TRUE)

test = subset(mtcars, split == FALSE)

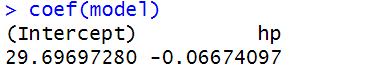
*#Creating an Linear Regression Model for Perdicting MPG using HP using the Train data*

model = lm(formula = mpg ~ hp, data = train)

*#Veiwing th Coefficient of the Model*

coef(model)

**Output :**



*#Overall Summary of the Model*

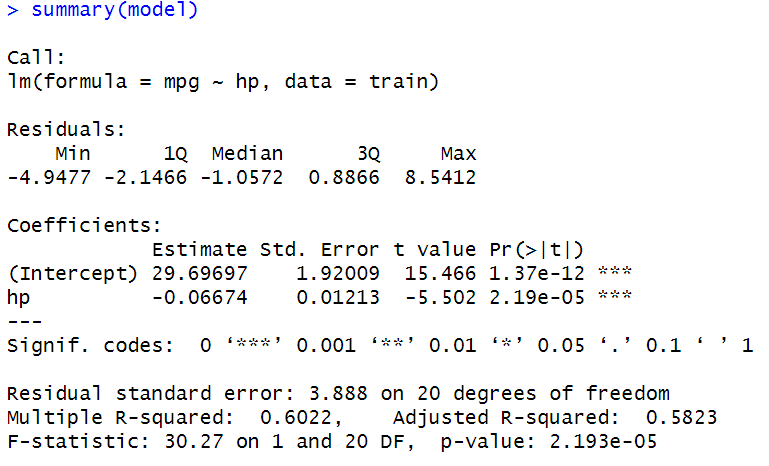
summary(model)

*#predicting the results using the Test data*

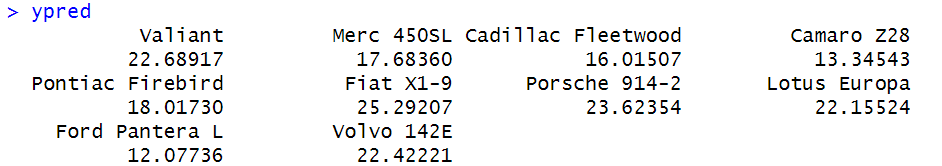
ypred = predict(model, newdata = test)

ypred

**Output :**

****

**Output :**

****

*#Importing ggplot package for visualising the result using Train as well as the Test Dataset*

library(ggplot2)

ggplot() +

geom\_point(aes(x = train$hp, y = train$mpg), colour = 'red')+

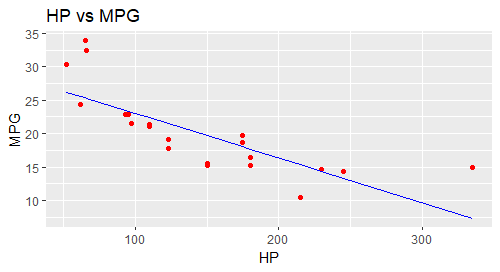
geom\_line(aes(x = train$hp,y = predict(model, newdata = train)), colour = "blue")+

xlab("HP") +

ylab("MPG")+

ggtitle("HP vs MPG")

**Output :**

****

ggplot()+geom\_point(aes(x = test$hp, y = test$mpg),colour = 'red')+

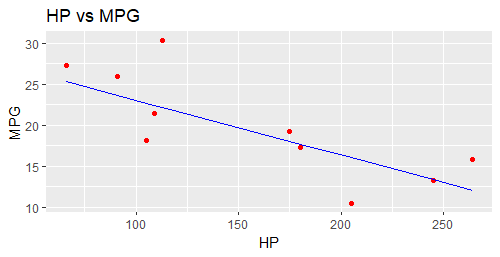
geom\_line(aes(x = test$hp, y = predict(model, newdata = test)), colour = "blue")+

xlab("HP")+

ylab("MPG")+

ggtitle("HP vs MPG")

**Output :**

****

**Program 4:**

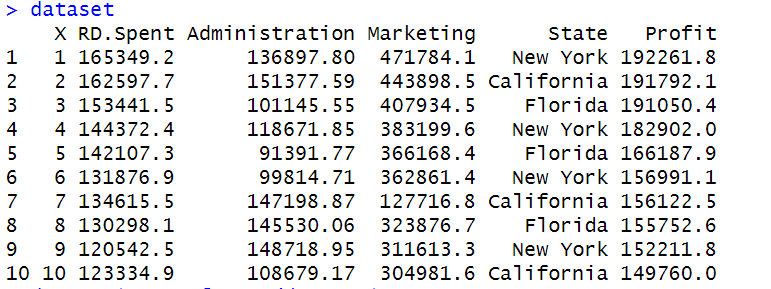
**Program to Implement Multiple Linear Regression using Company Data.**

**The Program uses a Company Data containing Departments , State and Profit corresponding to that State. Here Profit is an Dependent Variable which is Dependent on other variables. So we use MLR to predict the profit using all the other attributes. In MLR we use RD Spent, Administration and Marketing as the Predictor variables together to predict the Profits or the Dependent Variable.**

dataset=read.csv('data2.csv')

dataset

**Output :**

****

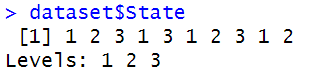
dataset$State=factor(dataset$State,

levels = c('New York','California','Florida'),

labels = c(1,2,3))

dataset$State

**Output :**

****

*#Splitting dataset into training and test sets*

library(caTools)

split=sample.split(dataset$Profit,SplitRatio = 0.8)

trainingset=subset(dataset,split==TRUE)

testset=subset(dataset,split==FALSE)

trainingset

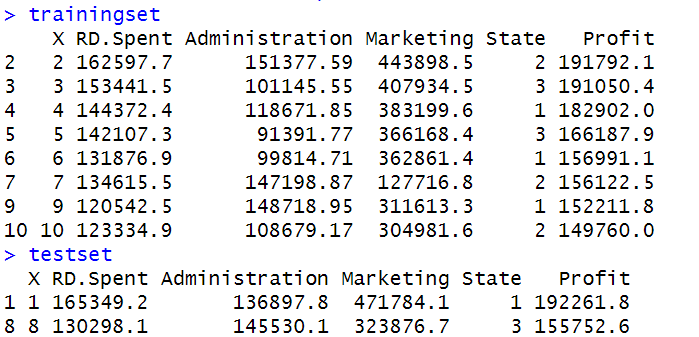
testset

*#Fitting multiple linear regression model into "regressor"*

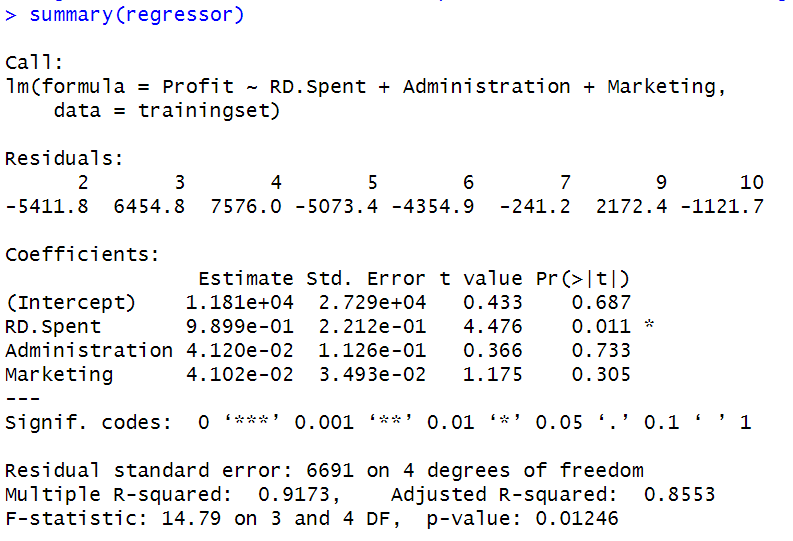
regressor=lm(formula = Profit~RD.Spent+Administration+Marketing,data=trainingset)

summary(regressor)

**Output :**

****

**Output :**

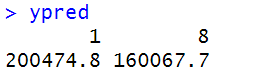
****

#Predicting on test data

ypred=predict(regressor,newdata = testset)

ypred

**Output :**

****

*#Visualizing results*

*#The best way to visualize multiple linear regression is to create a visualization for each independent variable while holding the other independent variables constant.*

*#Doing this allows us to see how each relationship between the DV(Dependent Variable) and IV(Independent Variable) looks.*

*#One way is to use Added-Variable plots. The "avPlots()" function in "car" library is used for this.*

*#plotting added variable plots*

*#install "car" package*

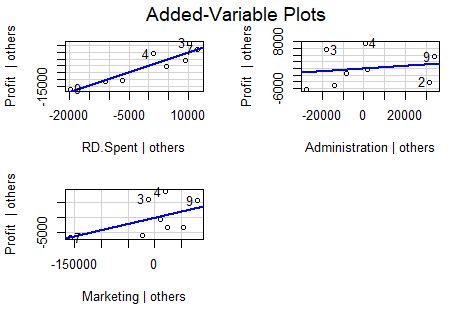
install.packages("car")

library(car)

#produce added variable plots

avPlots(regressor)

**Output :**



**Program 5:**

**Program to Implement Multiple Linear Regression using Students Performance Dataset.**

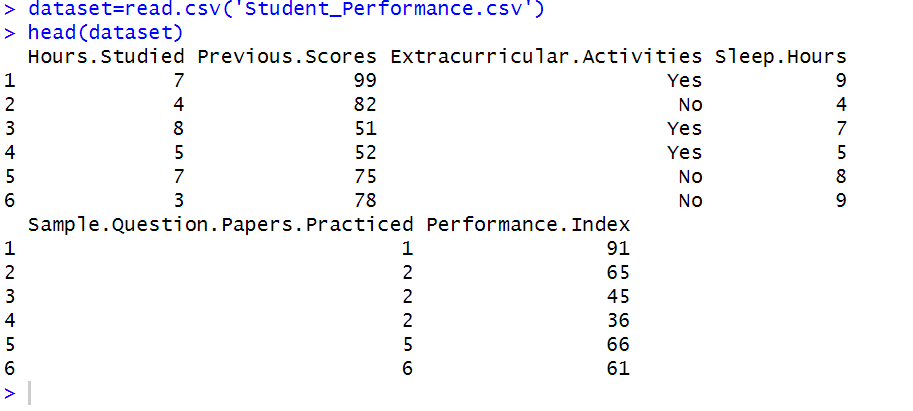
**The Program uses Students Performance Dataset to predict the Performance Index of each students based on multiple attributes such as Hours studied, Previous Scores, sleep Hours, Extracurricular activities, etc.**

**Here Performance Index is the Dependent Variables which is dependent on rest of the other Variables. So using MLR we establish an Linear relationship between Performance.Index and rest of the other variables in the Dataset to predict the performance of each students.**

dataset=read.csv('Student\_Performance.csv')

head(dataset)

**Output :**

****

View(dataset)

library(caTools)

split=sample.split(dataset$Performance.Index,SplitRatio = 0.8)

trainingset=subset(dataset,split==TRUE)

testset=subset(dataset,split==FALSE)

trainingset

testset

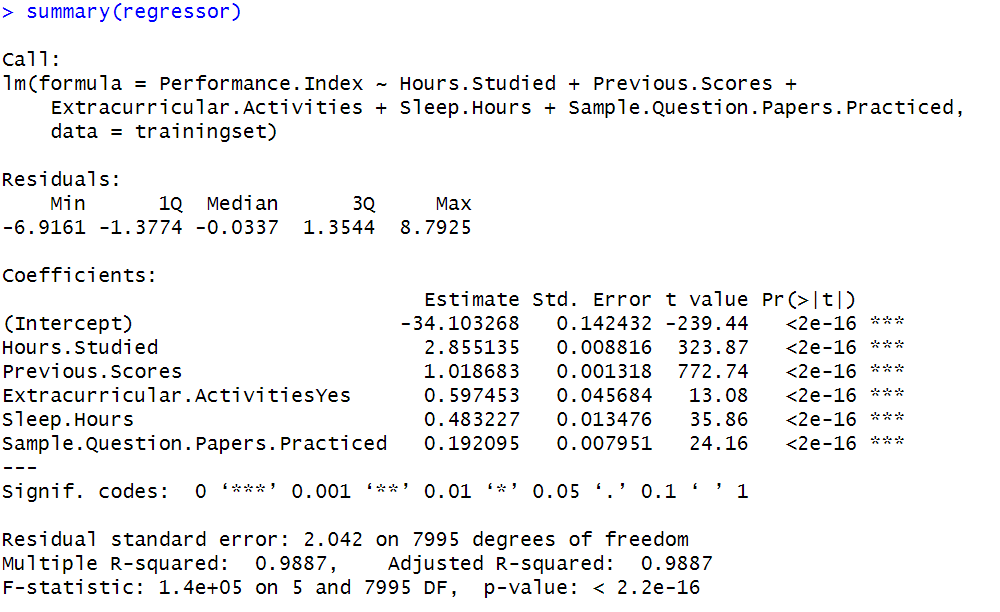
regressor = lm(formula = Performance.Index ~ Hours.Studied + Previous.Scores + Extracurricular.Activities + Sleep.Hours + Sample.Questions.Papers.Practiced, data = trainingset)

summary(regressor)

ypred=predict(regressor,newdata = testset)

ypred

**Output :**

****

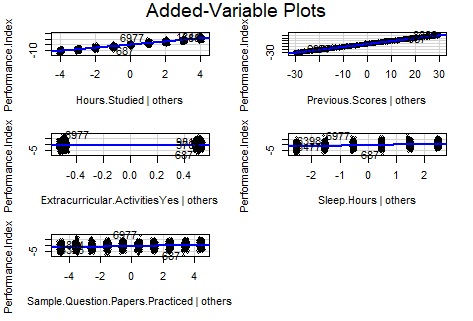
*#visualizing results*

install.packages("car")

library(car)

avPlots(regressor)

**Output :**

****

**Program 6:**

**Program to Implement Multiple Linear Regression using Iris Data.**

**This is perhaps the best-known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.**

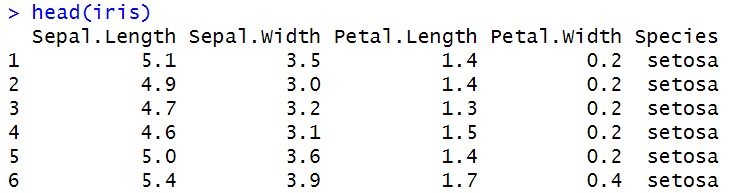
*#Importing IRIS dataset to perform Multiple Linear Regression to predict the Species of the iris*

data(iris)

*#Vewing the First 6 Observations of the Datasets*

head(iris)

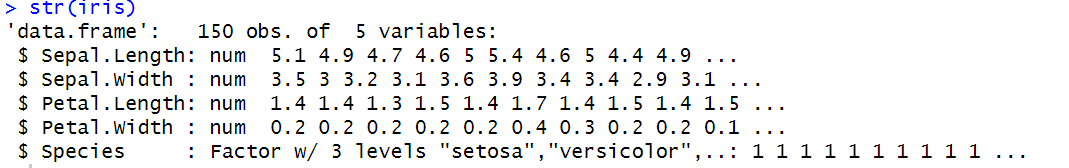
**Output :**

****

*#Vewing the Structure of the Dataset*

str(iris)

**Output :**

****

*#Converting the Species into a Numeric value i.e 1,2,3*

iris$species<- as.numeric(as.factor(iris$Species))

*#Vewing the Last 6 Observations of the species from the iris Dataset*

tail(iris$species)

**Output :**

****

*#Importing caTools package to split Dataset into Training and Testing set*

library(caTools)

set.seed(100)

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

train <- subset(iris, split == TRUE)

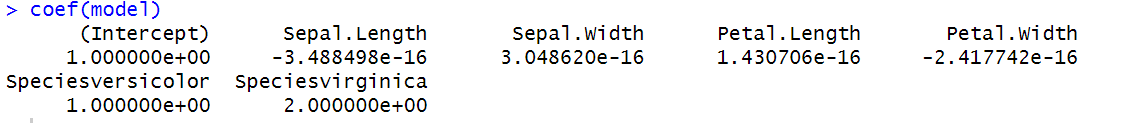
test <- subset(iris, split == FALSE)

*#Creating an MLR model to predict the species of the iris using the Train data*

model <- lm(formula = species ~ ., data = train)

coef(model)

**Output :**

****

summary(model)

*#Predicting the species using the Test Data*

ypred <- predict(model, newdata = test)

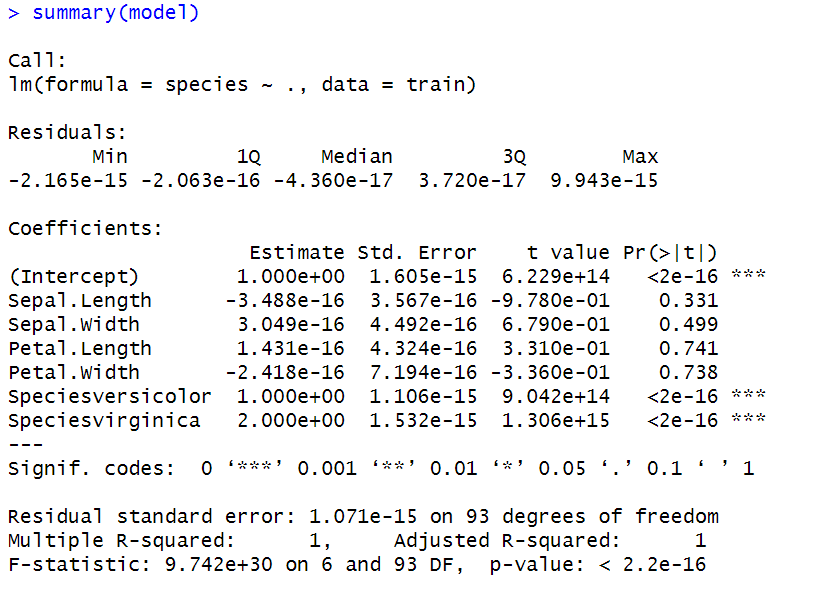
head(ypred)

*#Importing Car Package to Visualize the Results*

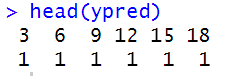
library(car)

avPlots(model)

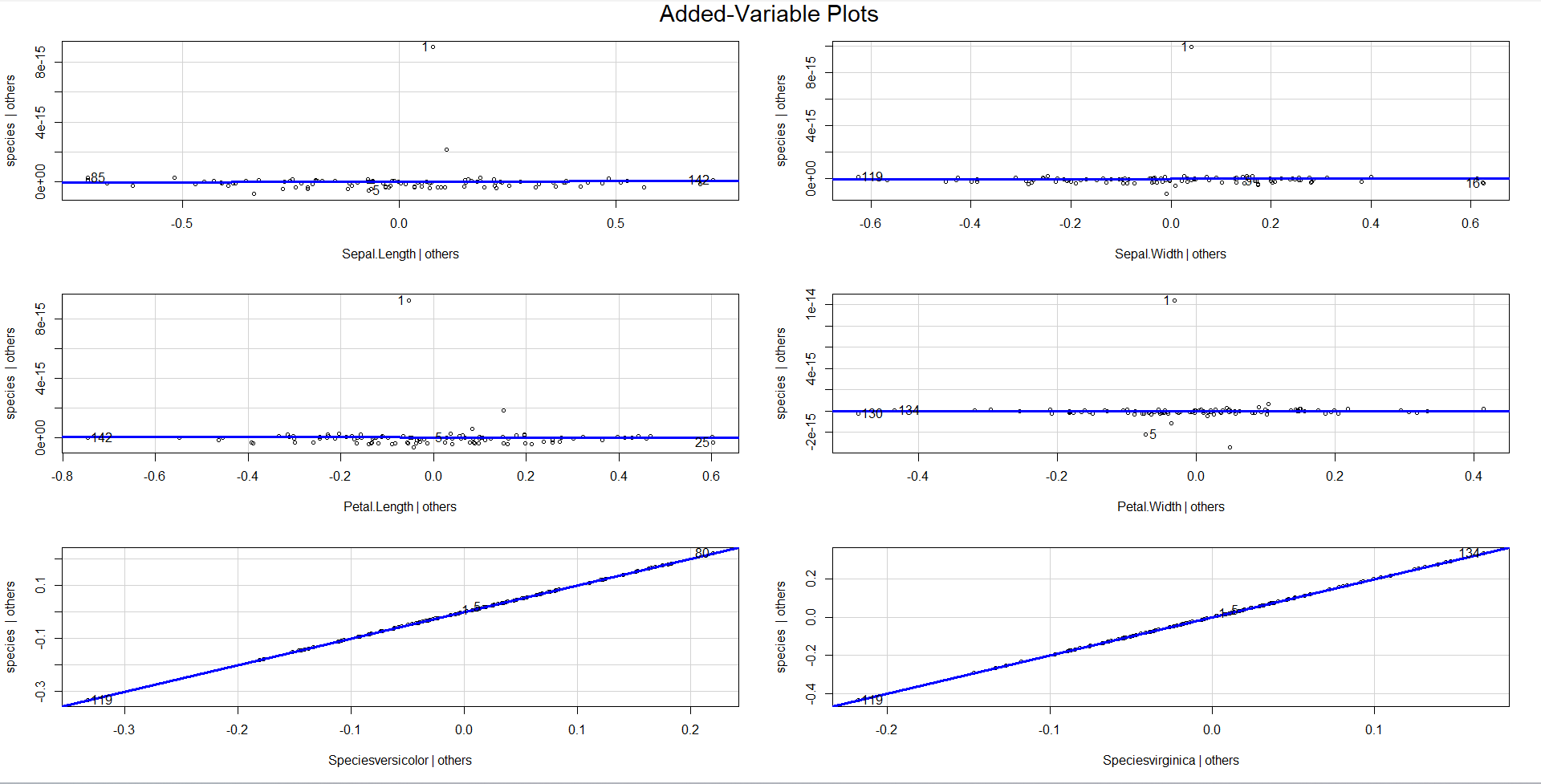
**Output :**

****

**Output :**

****

**Output :**



**Program 7:**

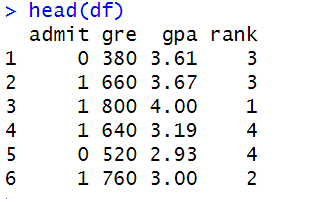
**Program to Implement Logistic Regression using Students Result Data.**

**The Program uses Students Results Dataset to predict if a student has been admitted into an Institution based on Attributes such as GRE (Graduate Record Examination) Results, GPA (Grade Point Average) result as well as the Rank of the Students. Attribute admit has only two values 0 which means “No” and 1 means “Yes”. Here admit is the Dependent Variable which is Dependent on other variables such as GRE, GPA and Rank. So we predict the Admit variable from other variables using the Logistic Regression.**

df <- read.csv("https://stats.idre.ucla.edu/stat/data/binary.csv")

head(df)

**Output :**

****

sum(is.na(df))

**Output :** 1 [0]

df$rank <- as.factor(df$rank)

logit <- glm(admit ~ gre+gpa+rank,data=df,family="binomial")

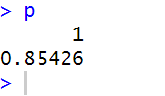
summary(logit)

x <- data.frame(gre=790,gpa=3.8,rank=as.factor(1))

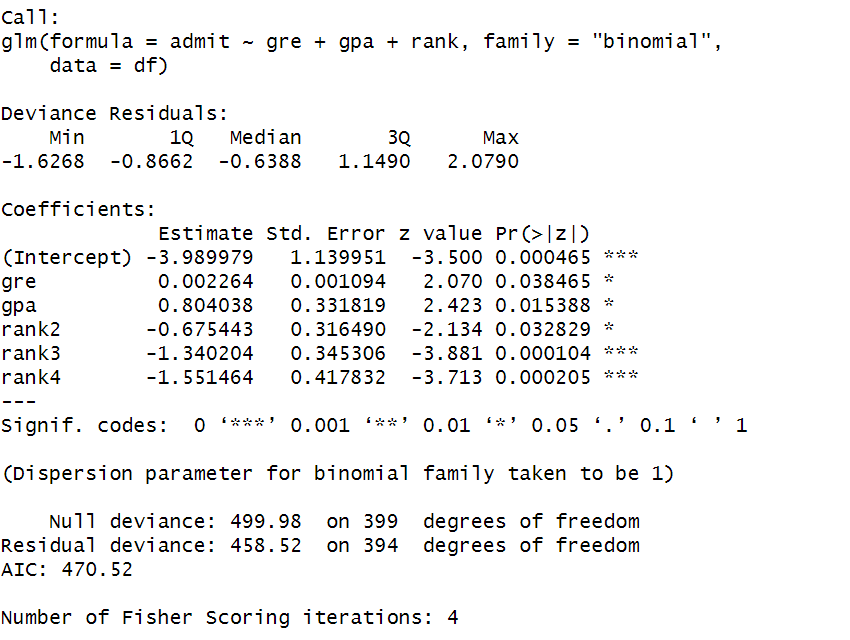
p<- predict(logit,x)

p

**Output :**

****

**Output :** summary(logit)

****

**Program 8:**

**Program to Implement Decision Tree using Reading Skills Data.**

**A toy data set illustrating the spurious correlation between reading skills and shoe size in school-children. A data frame with 200 observations on the following 4 variables.**

**NativeSpeaker : a factor with levels no and yes, where yes indicates that the child is a native speaker of the language of the reading test.**

**Age :age of the child in years.**

**shoesize : shoe size of the child in cm.**

**score : raw score on the reading test.**

library(datasets)

library(caTools)

library(party)

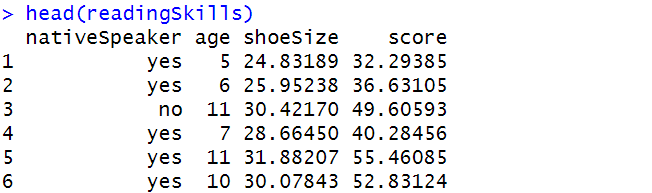
library(dplyr)

library(magrittr)

data("readingSkills")

head(readingSkills)

**Output :**

****

sample\_data=sample.split(readingSkills,SplitRatio = 0.8)

traindata=subset(readingSkills,sample\_data==TRUE)

testdata=subset(readingSkills,sample\_data==FALSE)

model=ctree(nativeSpeaker~.,traindata)

plot(model)

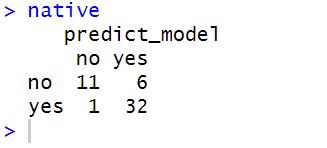
#making prediction

predict\_model=predict(model,testdata)

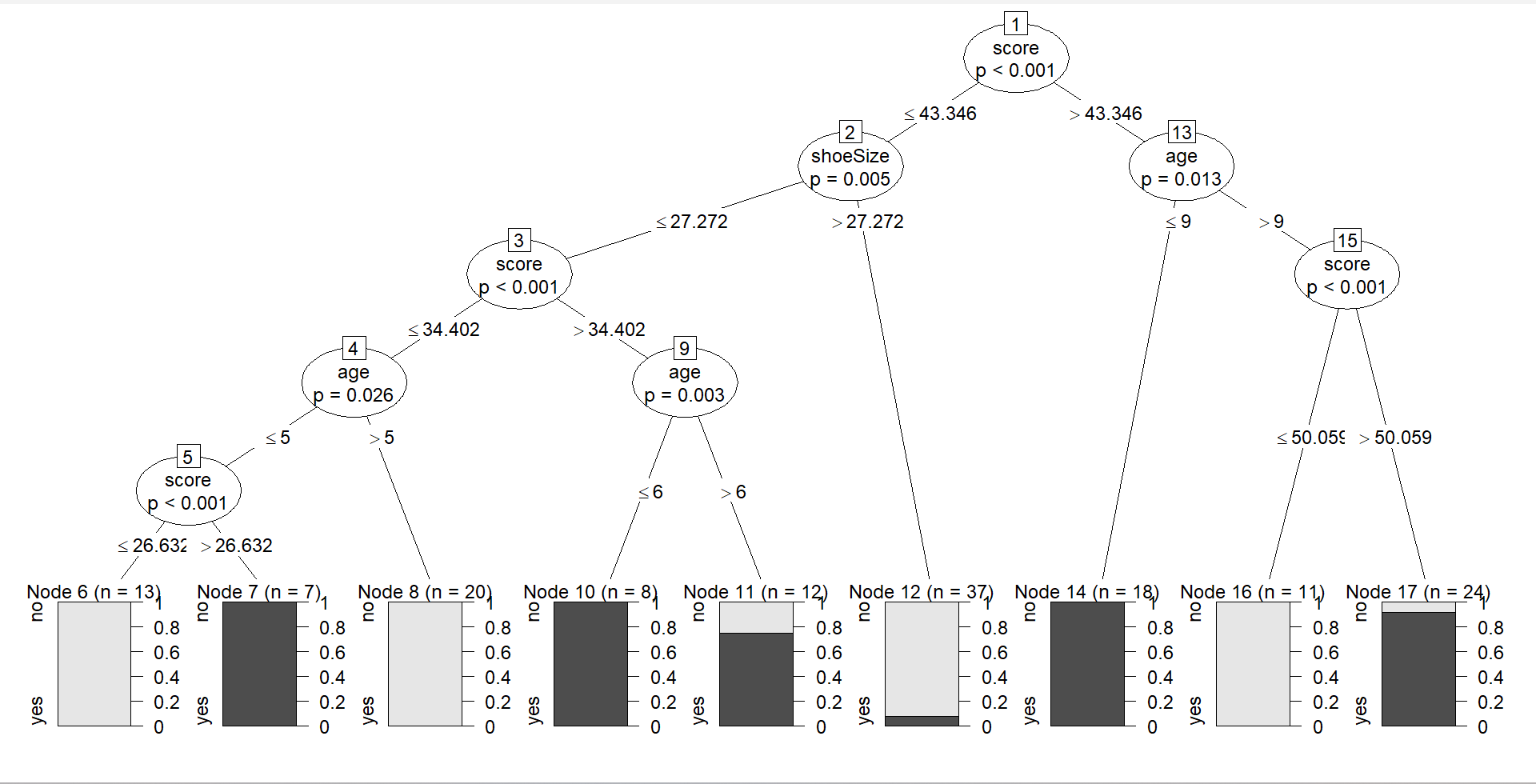
native=table(testdata$nativeSpeaker,predict\_model)

native

**Output :**

****

**Output :** plot(Model)

****

**Program 9:**

**Program to Implement Decision Tree using Raisin Data.**

**Images of Kecimen and Besni raisin varieties grown in Turkey were obtained with CVS. A total of 900 raisin grains were used, including 450 pieces from both varieties. These images were subjected to various stages of pre-processing and 7 morphological features were extracted. These features have been classified using three different artificial intelligence techniques.**

**Attribute Information:**

1. **Area: Gives the number of pixels within the boundaries of the raisin.**

**2. Perimeter: It measures the environment by calculating the distance between the boundaries of the raisin and the pixels around it.**

**3. MajorAxisLength: Gives the length of the main axis, which is the longest line that can be drawn on the raisin.**

**4. MinorAxisLength: Gives the length of the small axis, which is the shortest line that can be drawn on the raisin.**

**5. Eccentricity: It gives a measure of the eccentricity of the ellipse, which has the same moments as raisins.**

**6. ConvexArea: Gives the number of pixels of the smallest convex shell of the region formed by the raisin.**

**7. Extent: Gives the ratio of the region formed by the raisin to the total pixels in the bounding box.**

**8. Class: Kecimen and Besni raisin.**

library(rpart)

library(caret)

library(caTools)

library(data.tree)

library(dplyr)

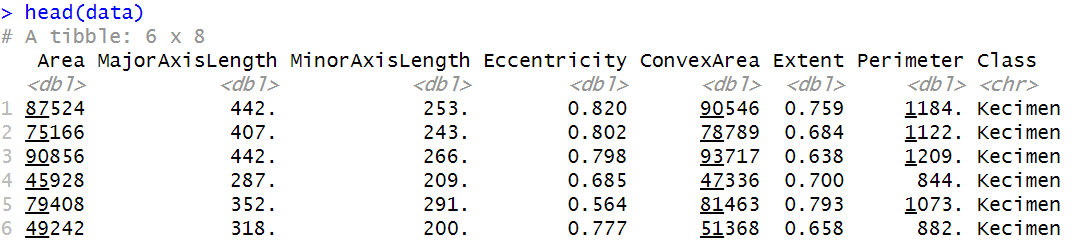
library(rpart.plot)

library(readxl)

data <- read\_excel("Raisin\_Dataset.xlsx")

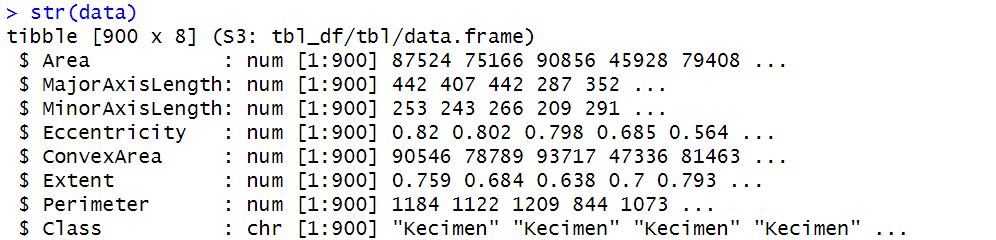
head(data)

**Output :**

****

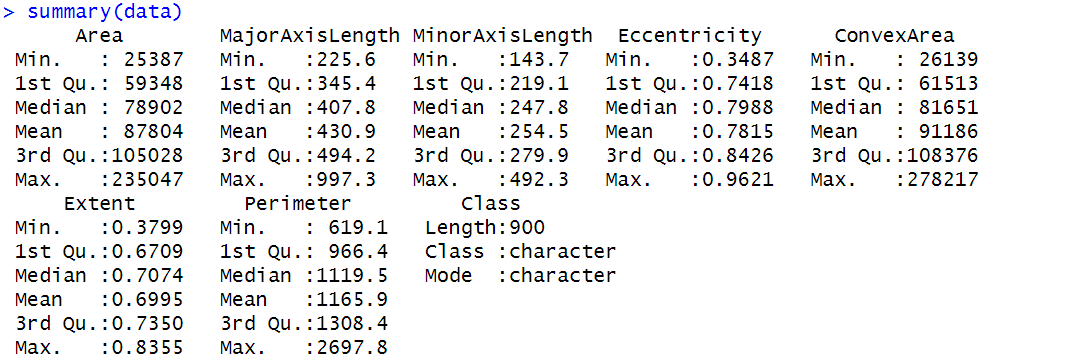
str(data)

**Output :**

****

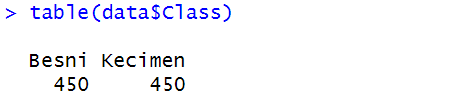
summary(data)

**Output :**

****

table(data$Class)

**Output :**

****

data <- mutate(data)

split <- sample.split(data$Class, SplitRatio = 0.7)

train <- subset(data, split == TRUE)

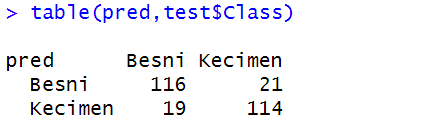
test <- subset(data, split == FALSE)

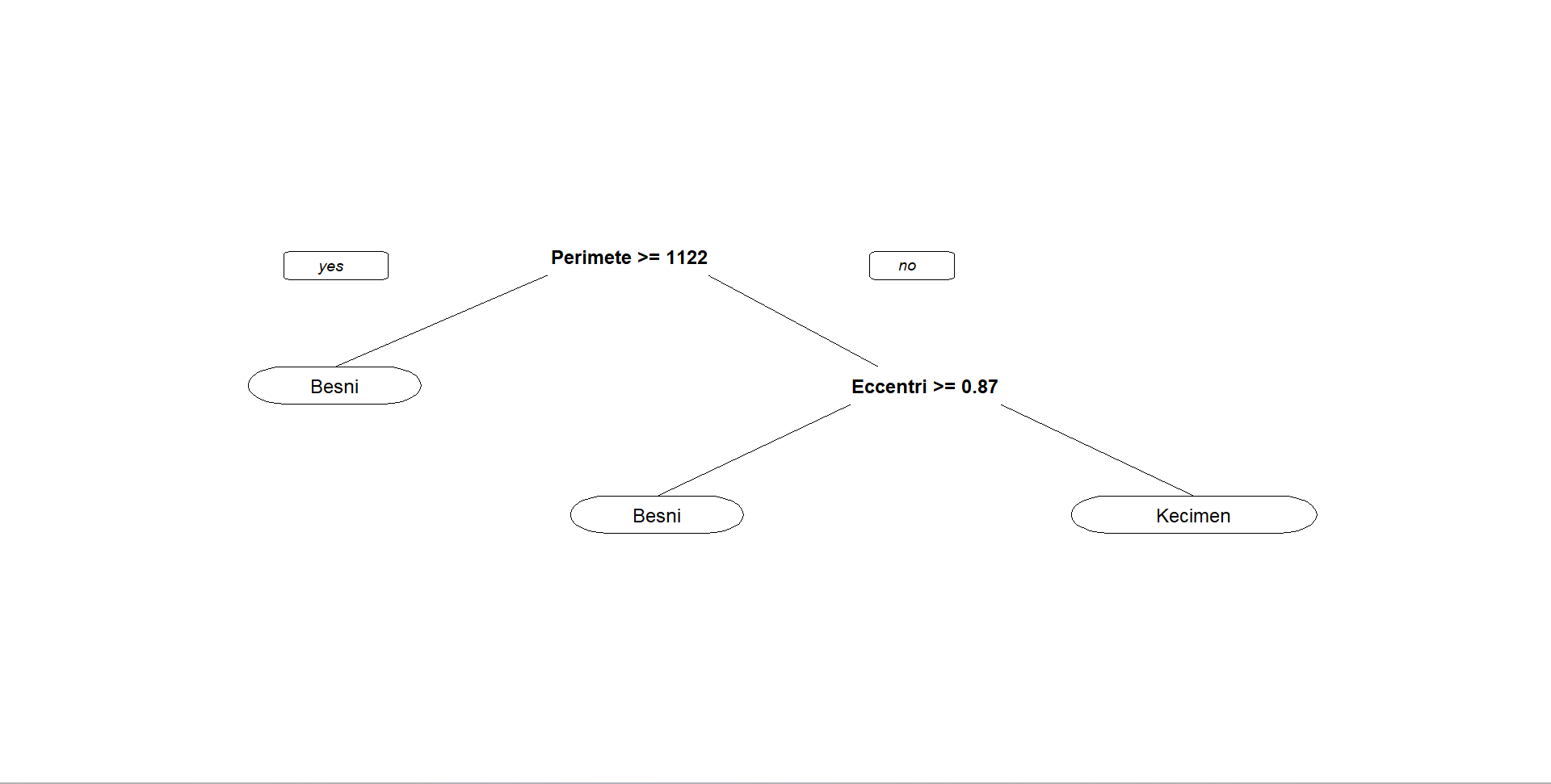
tree<-rpart(train$Class~.,data=train)

pred<- predict(tree,test,type='class')

table(pred,test$Class)

**Output :**

****

****

prp(tree)

**Output :**

**Program 10:**

**Program to Implement Decision Tree using Iris Data.**

**This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica**

**Iris is a data frame with 150 cases (rows) and 5 variables (columns) named Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, and Species.**

library(rpart)

library(rpart.plot)

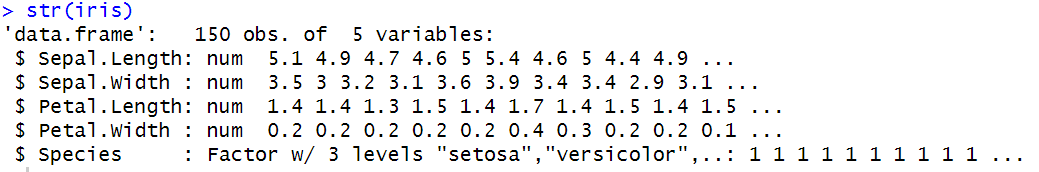
*#loading dataset*

data("iris")

#structure of iris

str(iris)

**Output :**

****

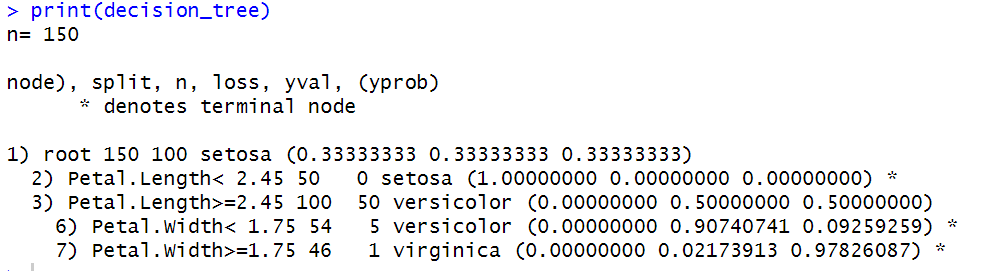
#creating decision tree model

decision\_tree=rpart(Species~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width,data = iris,method="class")

#print decision tree

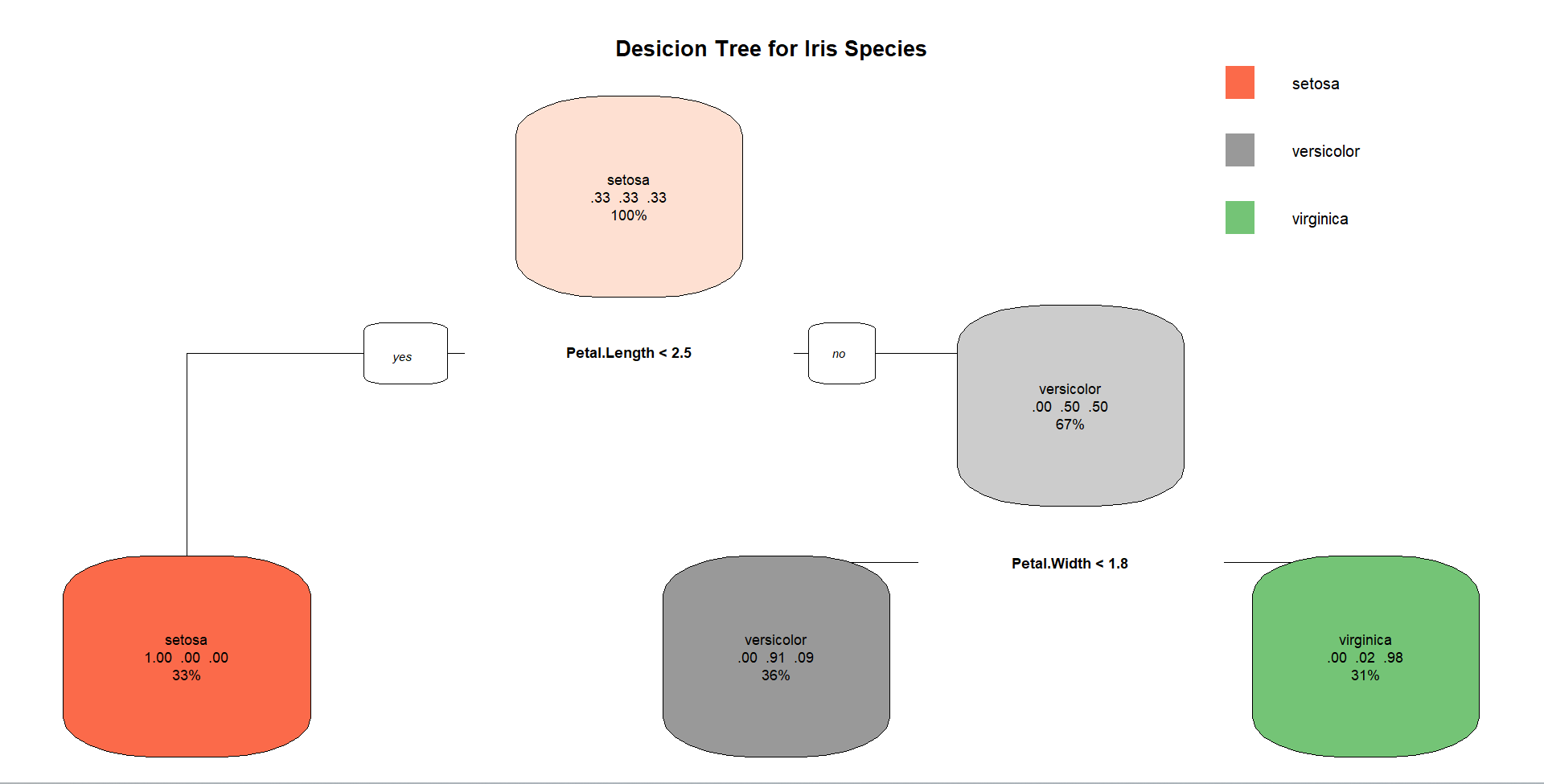
print(decision\_tree)

**Output :**

****

#plotting

rpart.plot(decision\_tree,main="Desicion Tree for Iris Species")

**Output :**

**Program 11:**

**Program to Implement Random Forest using Wine Quality-Red Data.**

**The dataset is related to red variant of the Portuguese "Vinho Verde" wine.**

**Attribute Information: Input variables (based on physicochemical tests):**

**1. fixed acidity**

**2. volatile acidity**

**3. citric acid**

**4. residual sugar**

**5. chlorides**

**6. free sulfur dioxide**

**7. total sulfur dioxide**

**8. density**

**9. pH**

**10. sulphates**

**11. alcohol**

**Output variable (based on sensory data):**

**12. quality (score between 0 and 10)**

library(randomForest)

library(caTools)

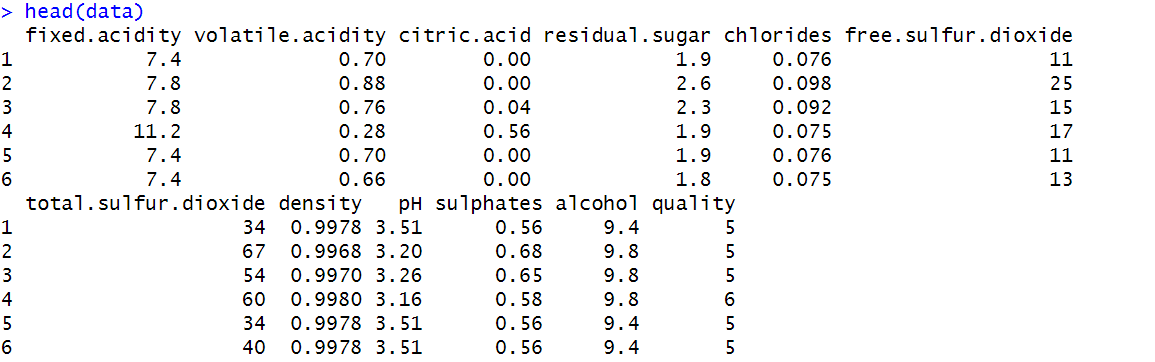
library(dplyr)

library(party)

data <- read.csv('winequality-red.csv')

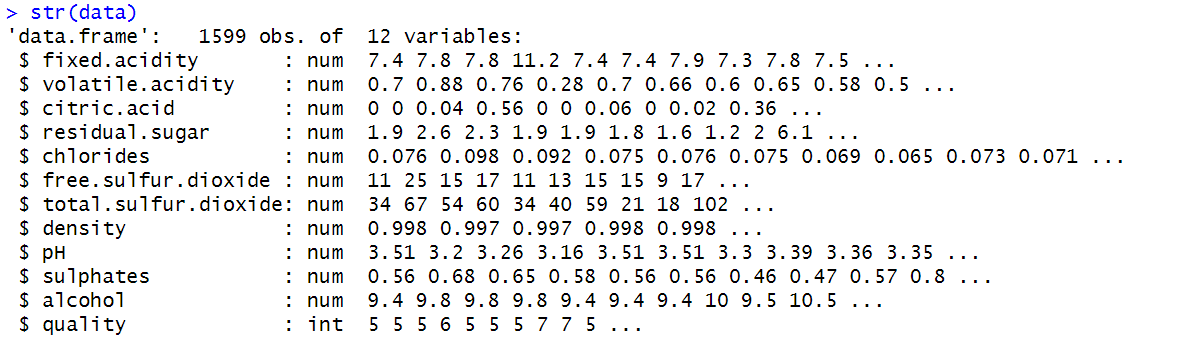
head(data)

**Output :**

****

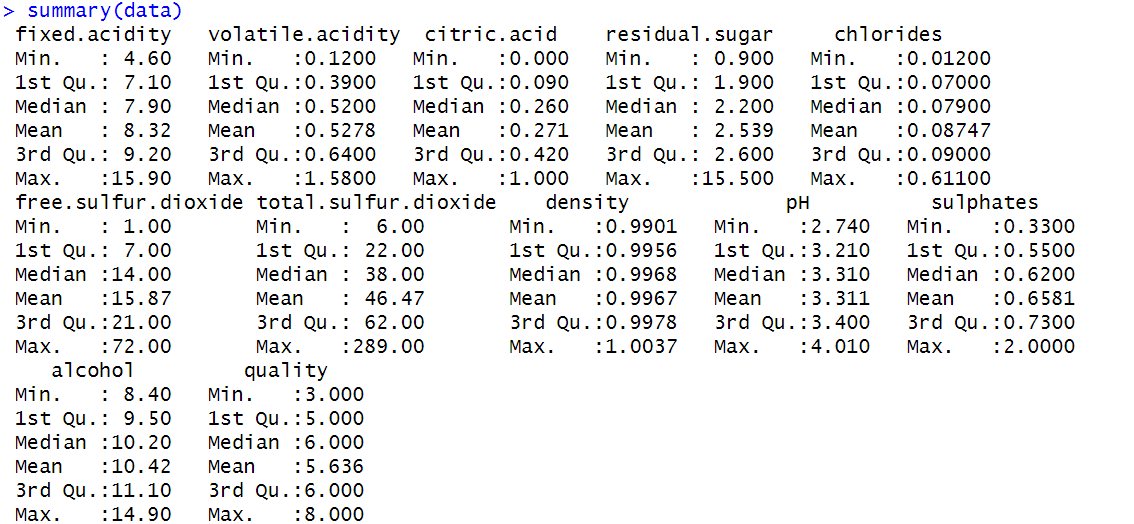
str(data)

**Output :**

****

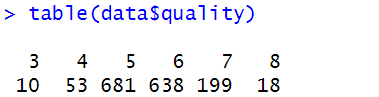
summary(data)

**Output :**

****

table(data$quality)

**Output :**

****

split <- sample.split(data$quality, SplitRatio = 0.7)

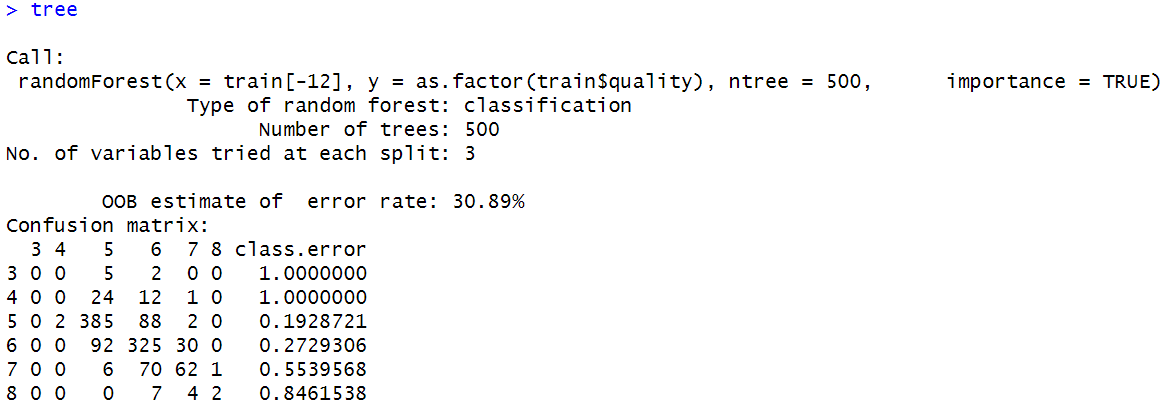
train = subset(data, split == TRUE)

test = subset(data, split == FALSE)

tree <- randomForest(x = train[-12], y = as.factor(train$quality),ntree = 500, importance = TRUE)

tree

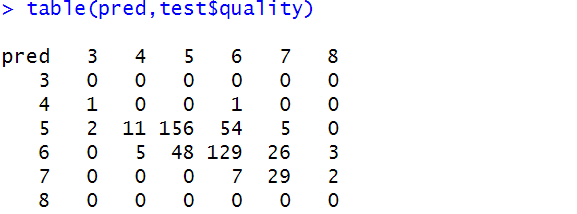
**Output :**

****

pred <- predict(tree, test, type = 'class')

table(pred,test$quality)

**Output :**

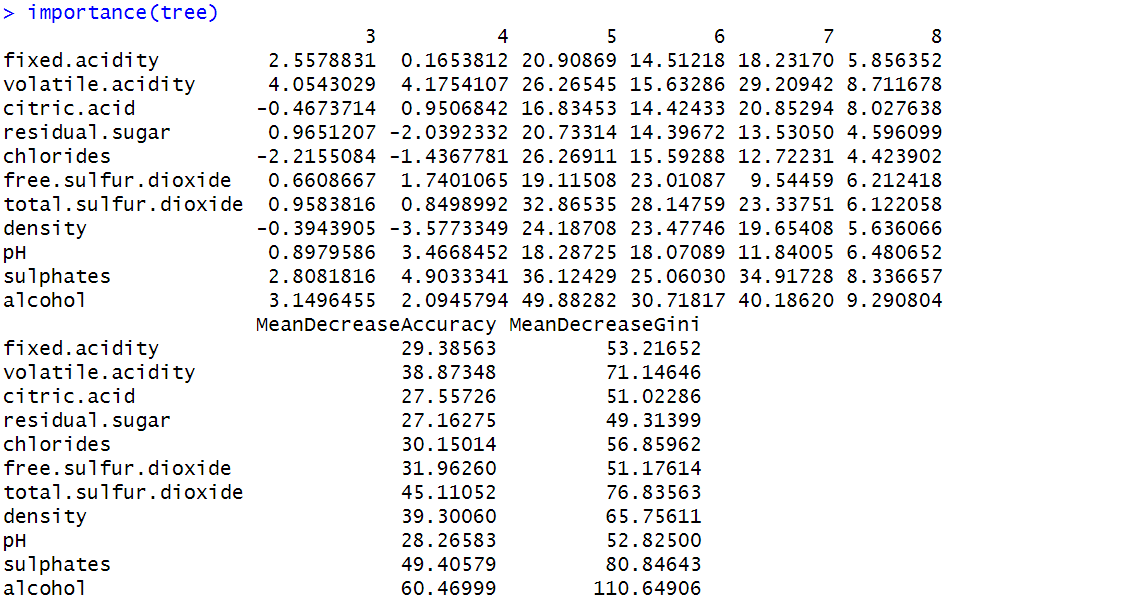
****

importance(tree)

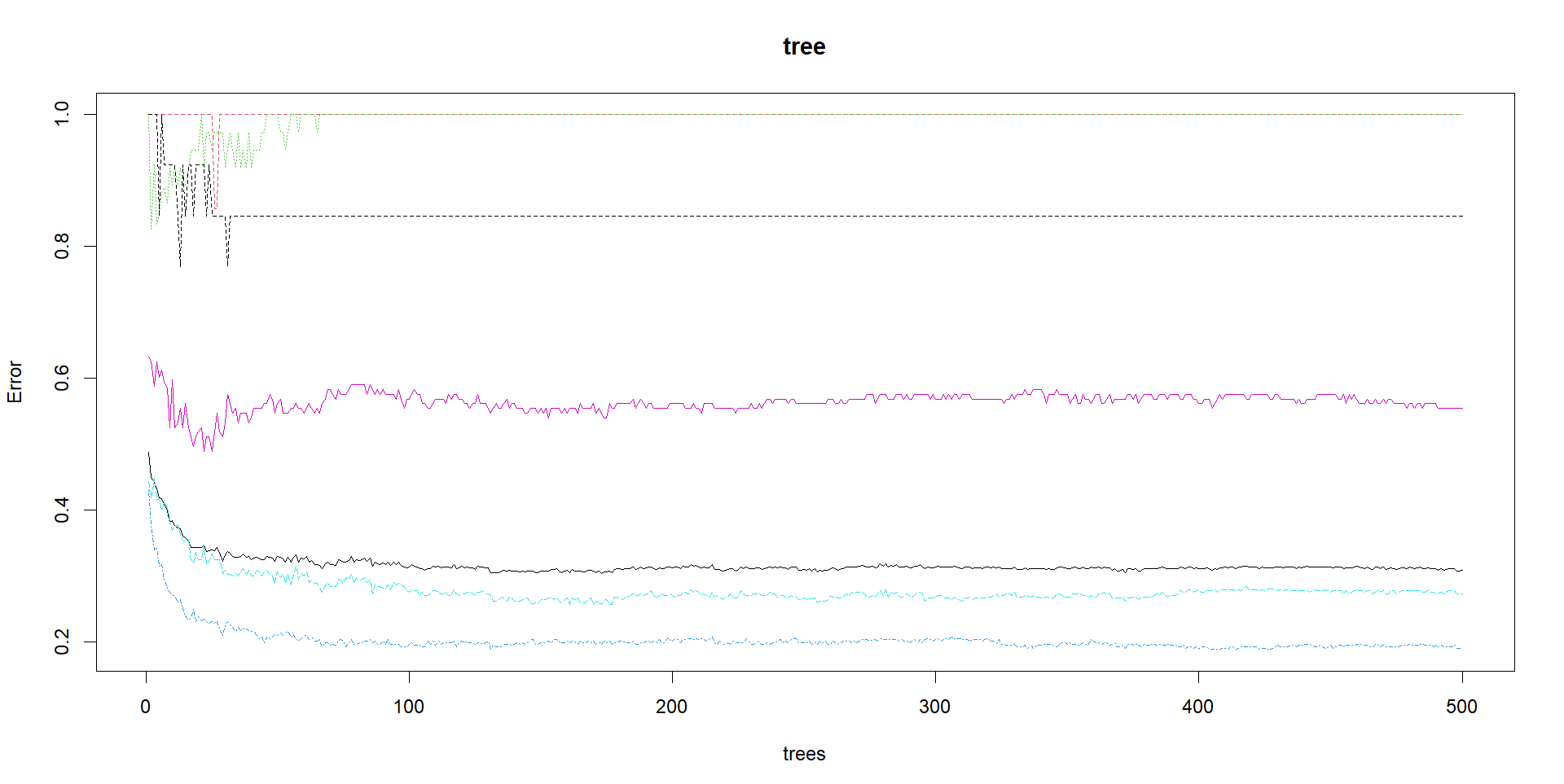
plot(tree)

varImpPlot(tree)

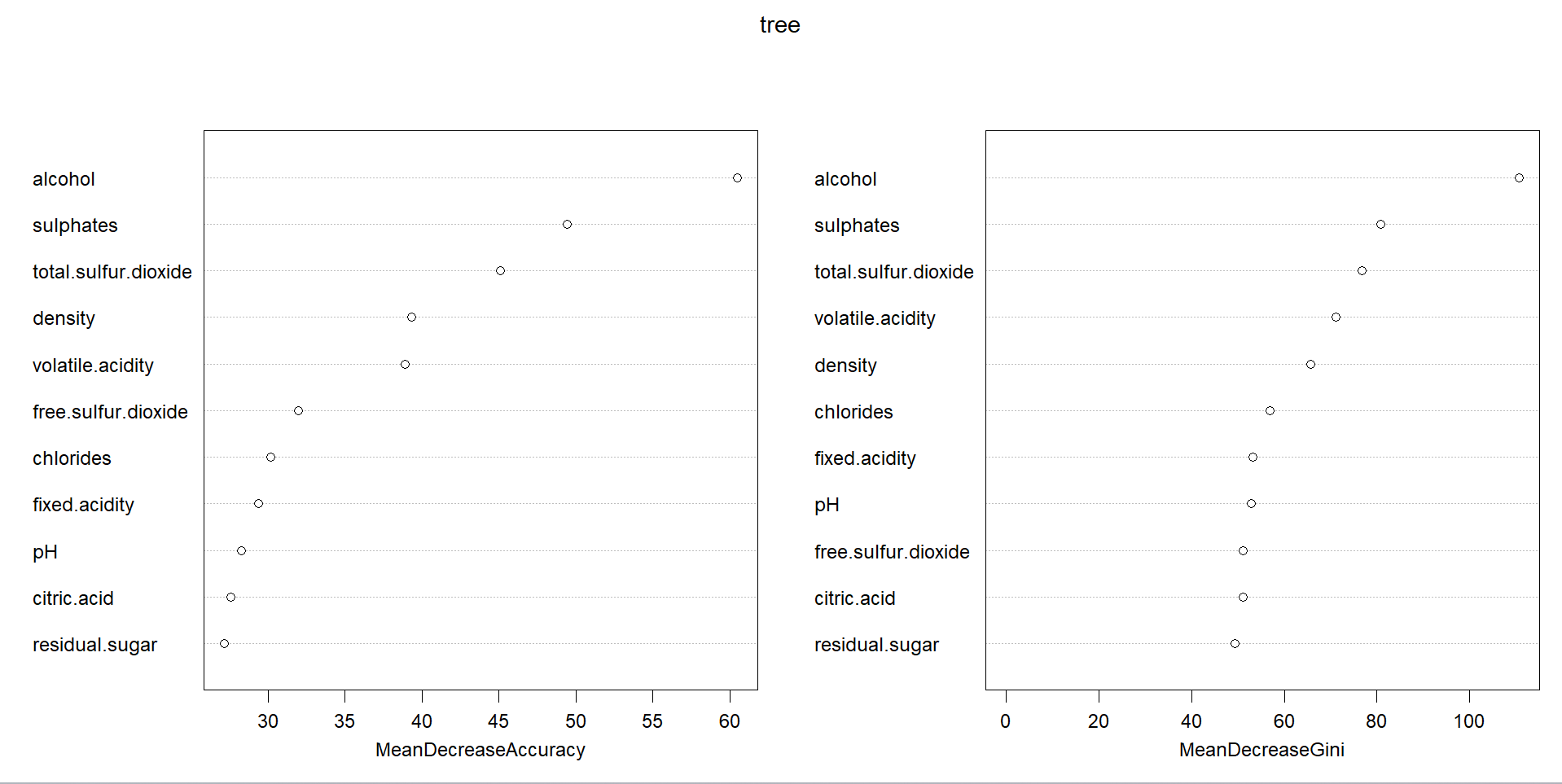
**Output :** Importance(tree)

****

**Output :** plot(tree)

****

**Output :** varImpPlot(tree)

****

**Program 12:**

**Program to Implement Random Forest using Raisin Data.**

**Images of Kecimen and Besni raisin varieties grown in Turkey were obtained with CVS. A total of 900 raisin grains were used, including 450 pieces from both varieties. These images were subjected to various stages of pre-processing and 7 morphological features were extracted. These features have been classified using three different artificial intelligence techniques.**

**Attribute Information:**

**1. Area: Gives the number of pixels within the boundaries of the raisin.**

**2. Perimeter: It measures the environment by calculating the distance between the boundaries of the raisin and the pixels around it.**

**3. MajorAxisLength: Gives the length of the main axis, which is the longest line that can be drawn on the raisin.**

**4. MinorAxisLength: Gives the length of the small axis, which is the shortest line that can be drawn on the raisin.**

**5. Eccentricity: It gives a measure of the eccentricity of the ellipse, which has the same moments as raisins.**

**6. ConvexArea: Gives the number of pixels of the smallest convex shell of the region formed by the raisin.**

**7. Extent: Gives the ratio of the region formed by the raisin to the total pixels in the bounding box.**

**8. Class: Kecimen and Besni raisin.**

library(randomForest)

library(dplyr)

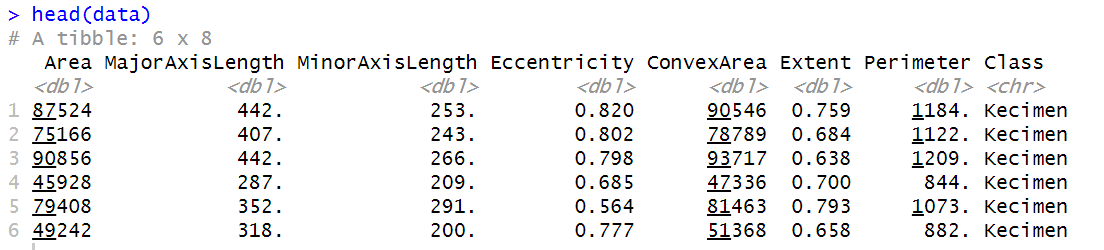
library(readxl)

*#Importing Raisin Dataset*

data <- read\_excel("Raisin\_Dataset.xlsx")

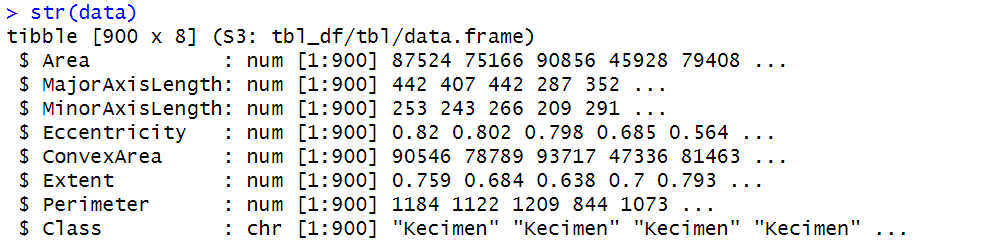
head(data)

**Output :**

****

str(data)

**Output :**

****

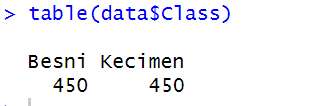
sum(is.na(data))

**Output :**

****

table(data$Class)

**Output :**

****

#using mutate function for Data manipulation

data <- mutate(data)

#Importing Package caTools for splitting the dataset into Train and Test set

library(caTools)

split <- sample.split(data$Class, SplitRatio = 0.7)

train <- subset(data, split == TRUE)

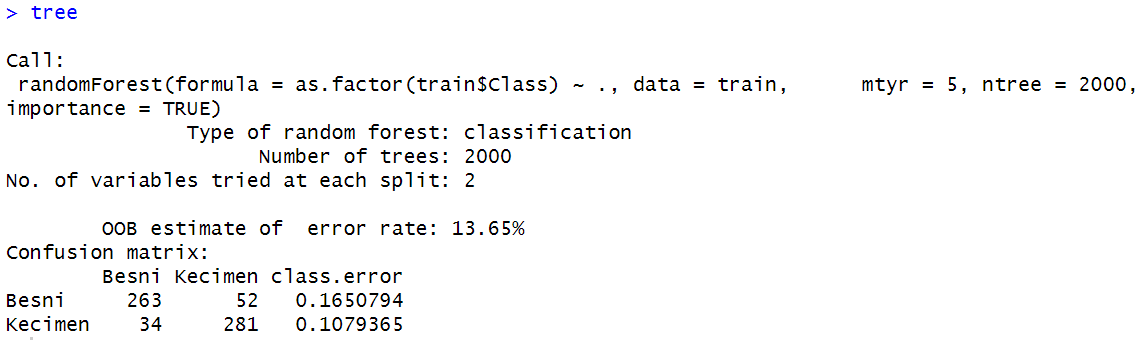
test <- subset(data, split == FALSE)

#using a Random forest function to build a random forest classifier

tree <- randomForest(as.factor(train$Class) ~ ., data = train, mtyr = 5, ntree = 2000, importance = TRUE)

tree

**Output :**

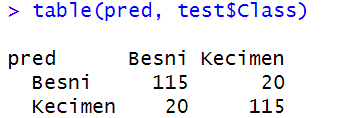
****

summary(tree)

pred <- predict(tree, newdata = test)

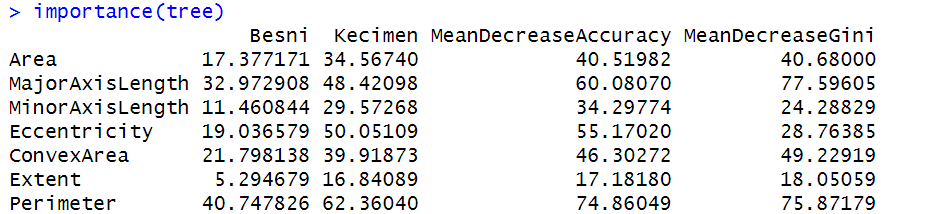
table(pred, test$Class)

**Output :**

****

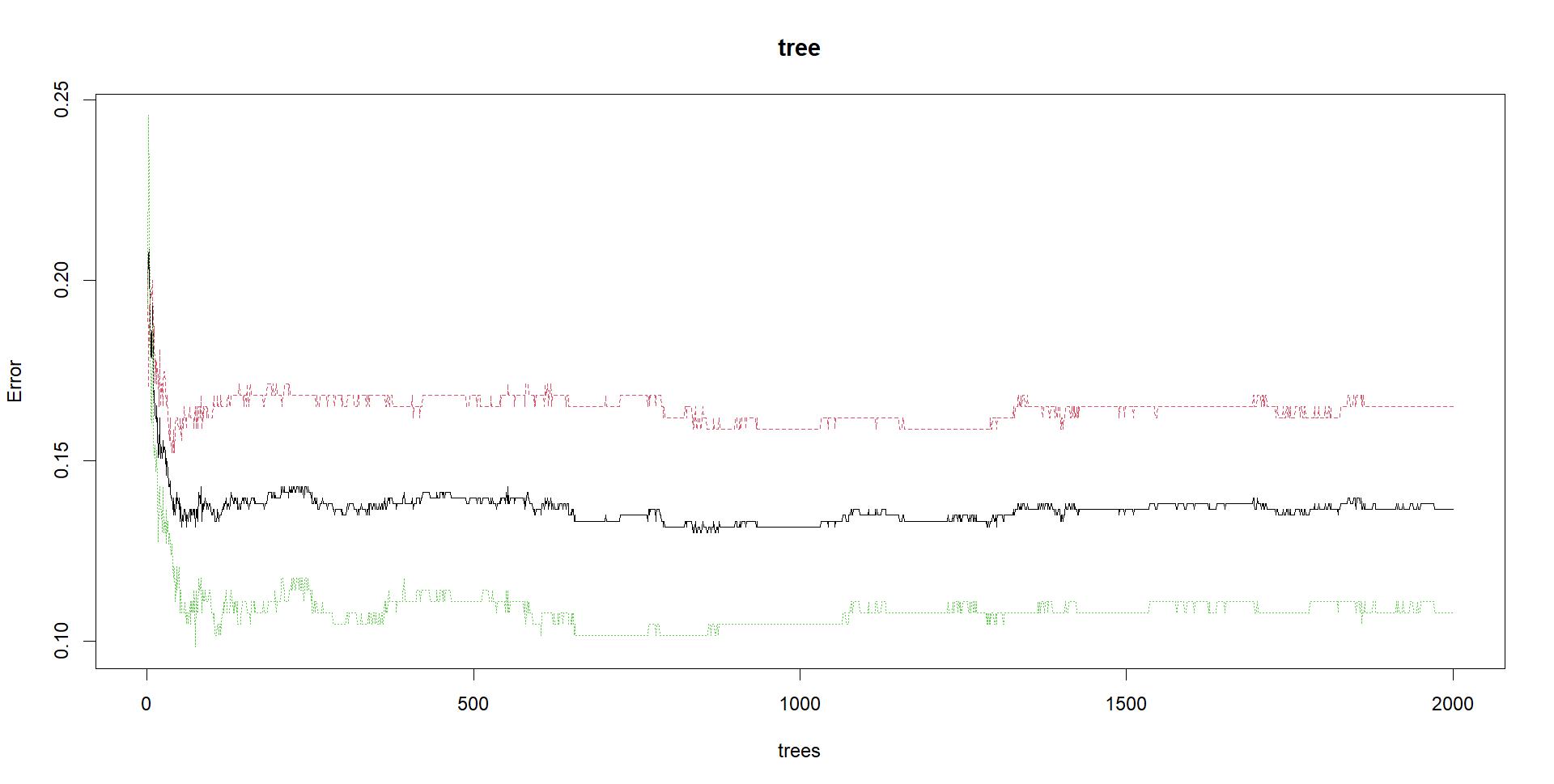
importance(tree)

**Output :**

****

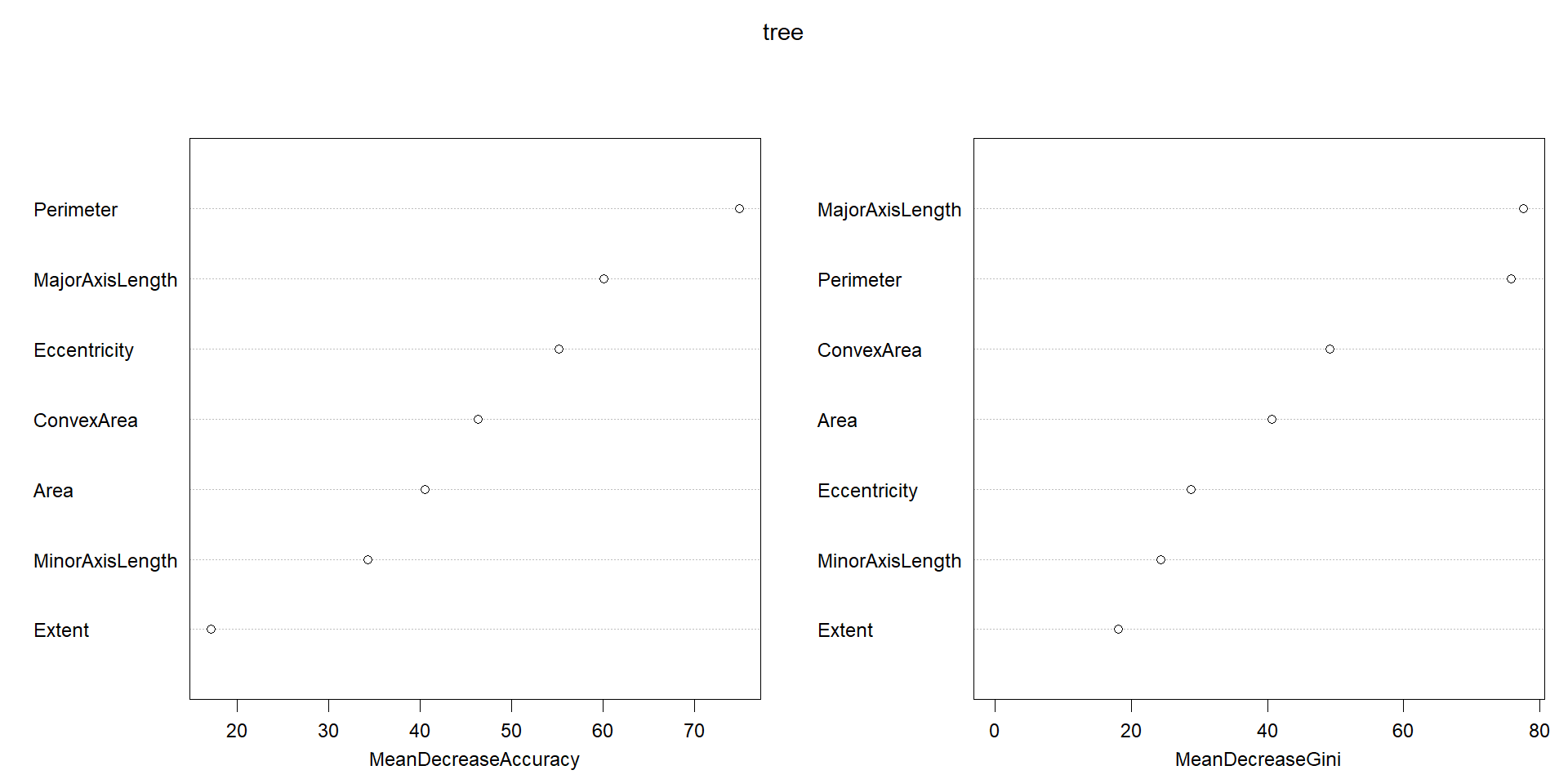
plot(tree)

**Output :**

****

varImpPlot(tree)

**Output :**

****

**Program 13:**

**Program to Implement Random Forest using Iris Data.**

**This is perhaps the best-known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.**

**Predicted attribute: class of iris plant.**

**Attribute Information:**

**1. sepal length in cm**

**2. sepal width in cm**

**3. petal length in cm**

**4. petal width in cm**

**class: - Iris Setosa, Iris Versicolour, Iris Virginica**

*#Importing Random Forest Package for Creating and Analysing a Random Forest*

library(randomForest)

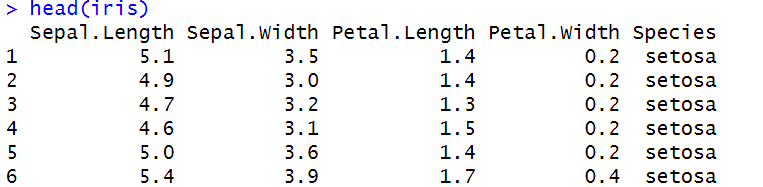
library(dplyr)

*#Importing Iris Data*

data(iris)

head(iris)

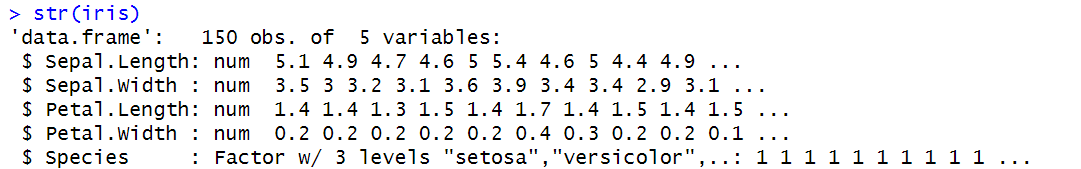
**Output :**

****

*#Structure of the Iris Data*

str(iris)

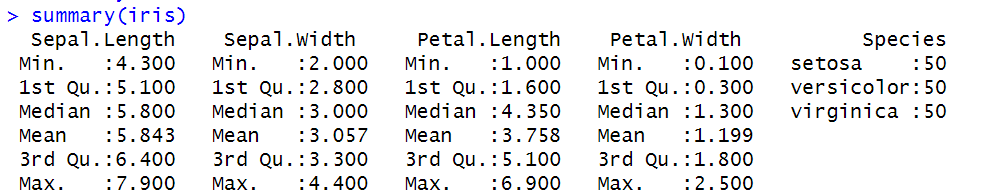
**Output :**

****

*#Summary of the Iris Data*

summary(iris)

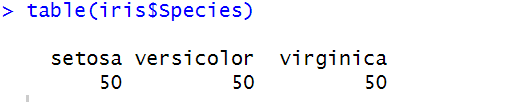
**Output :**

****

*#Checking the number of each Species of the Iris*

table(iris$Species)

**Output :**

****

*#Importing Package caTools for Splitting the data into Train and Test set*

library(caTools)

split = sample.split(iris$Species, SplitRatio = 0.7)

train <- subset(iris, split == TRUE)

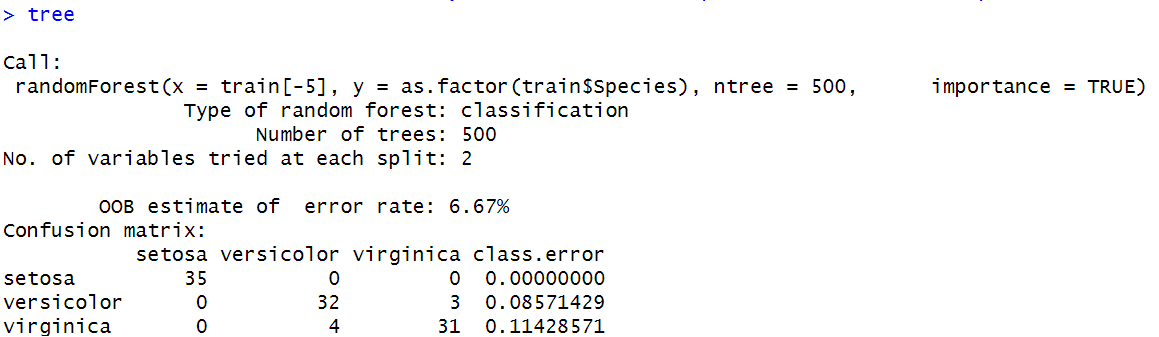
test <- subset(iris , split == FALSE)

*#Using randomForest function for creating a Classifier*

tree <- randomForest(x = train[-5],y = as.factor(train$Species),ntree = 500, importance = TRUE)

tree

**Output :**

****

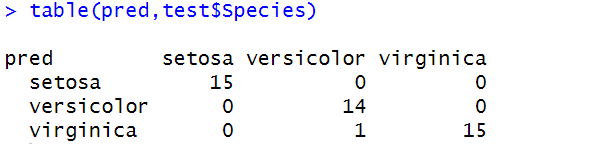
*#Making Predictions using Test Data*

pred <- predict(tree, newdata = test)

#*Creating a Confusion Matrix for checking the Accuracy*

table(pred,test$Species)

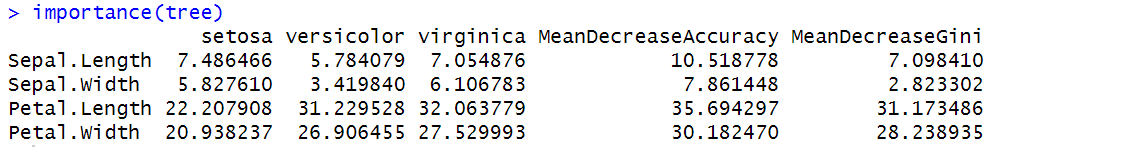
**Output :**



*#using Importance function for checking the importance of each variables in making Predictions*

importance(tree)

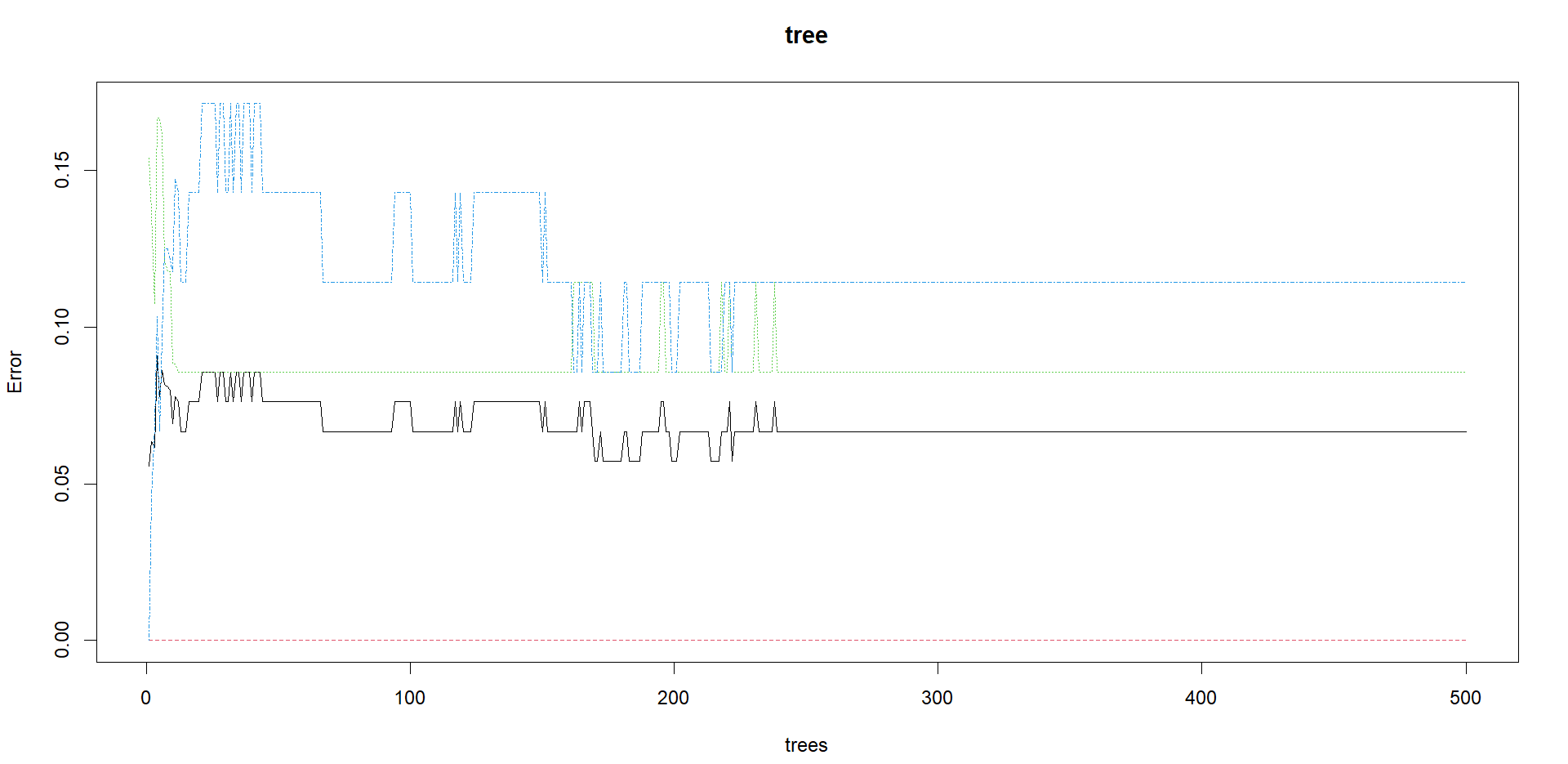
**Output :**

****

*#Visualizing the Result*

plot(tree)

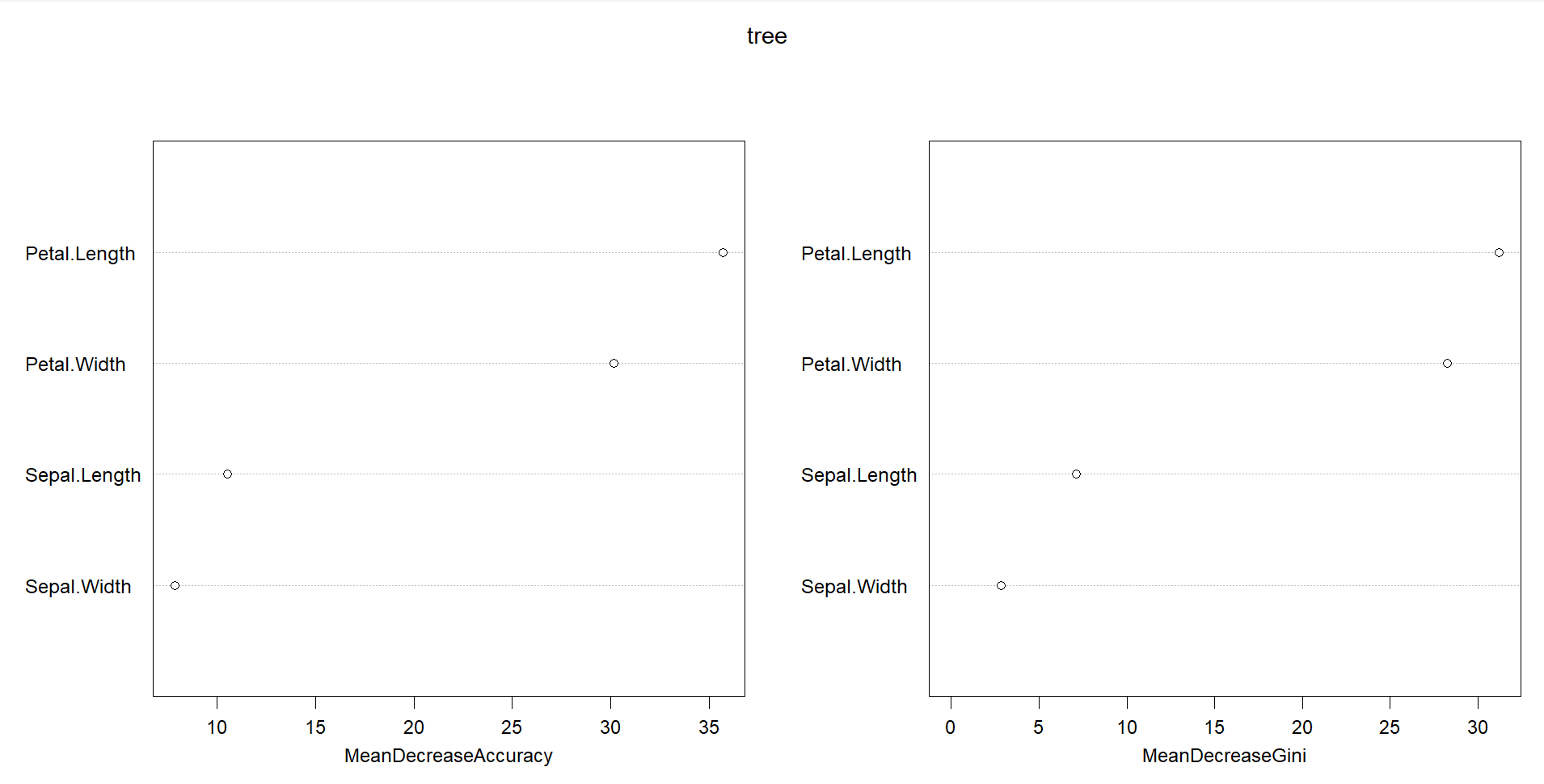
**Output :**

****

#*Plotting an Variable Importance Graph*

varImpPlot(tree)

**Output :**

****

**Program 14:**

**Program to Implement Naïve Bayesian using Iris Data.**

**This is perhaps the best-known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.**

**Predicted attribute: class of iris plant.**

**Attribute Information:**

**1. sepal length in cm**

**2. sepal width in cm**

**3. petal length in cm**

**4. petal width in cm**

**class: - Iris Setosa, Iris Versicolour, Iris Virginica**

library(e1071)

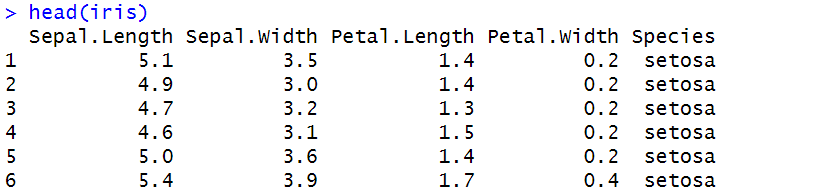
library(caTools)

library(caret)

data("iris")

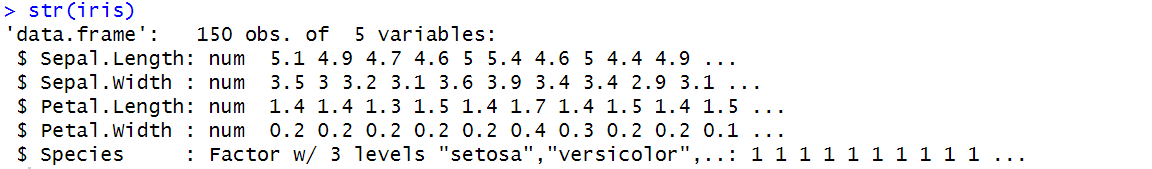
head(iris)

**Output :**

****

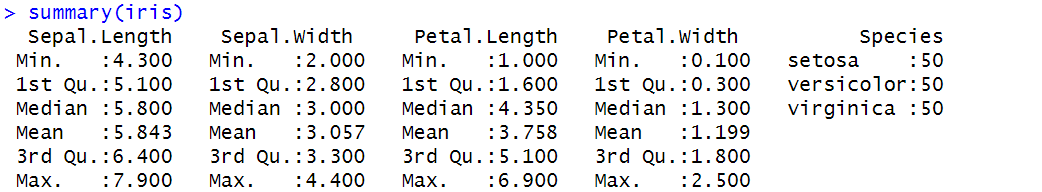
str(iris)

**Output :**

****

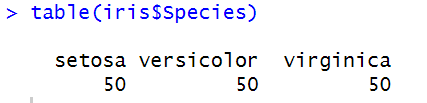
summary(iris)

**Output :**

****

table(iris$Species)

**Output :**

****

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

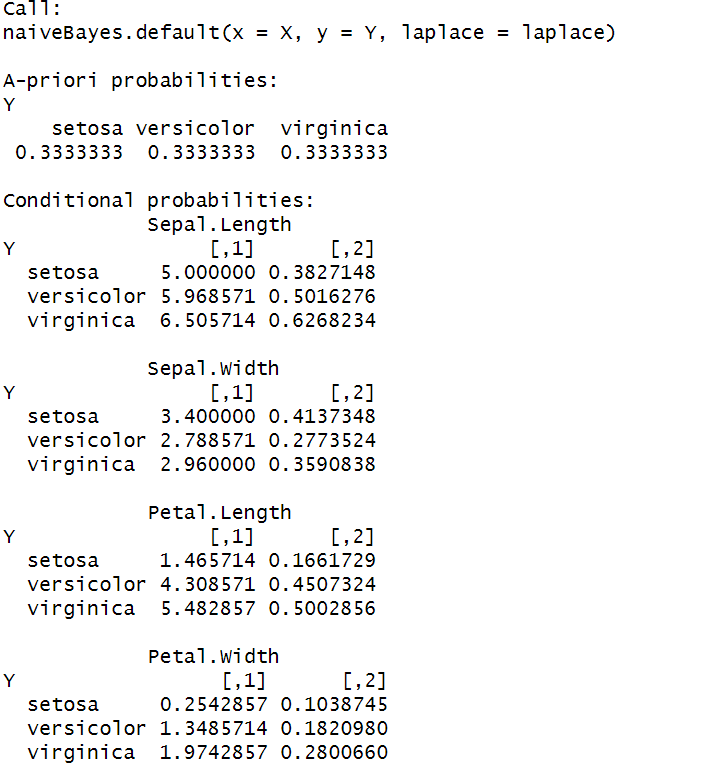
train <- subset(iris, split == TRUE)

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

model <- naiveBayes(formula = train$Species ~ ., data = train)

model

**Output :**

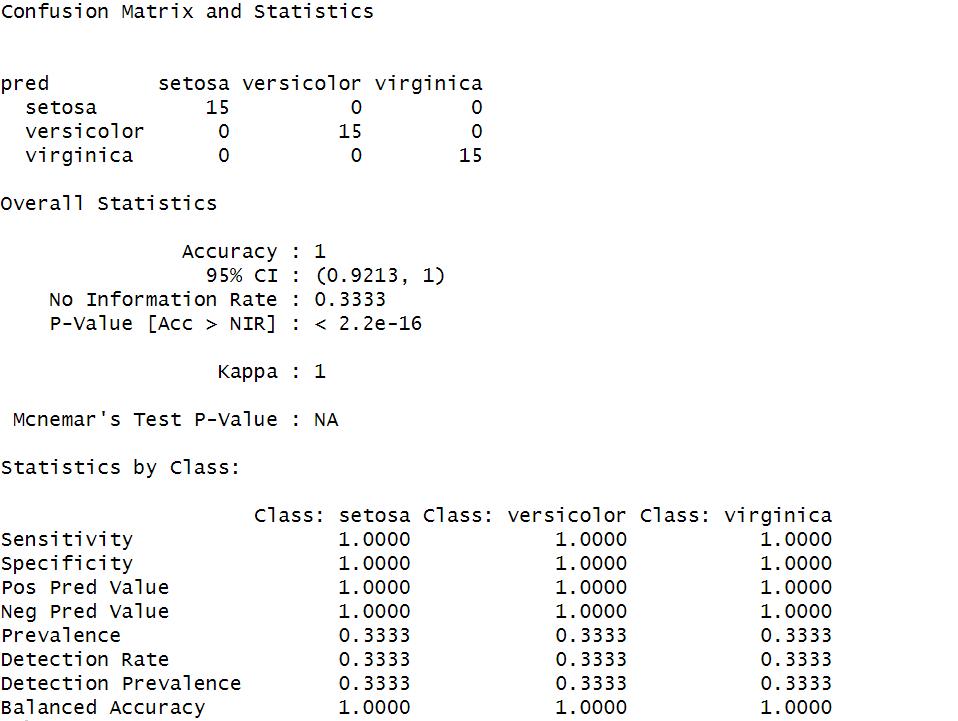
****

pred <- predict(model, newdata = test)

cm <- table(pred,test$Species)

confusionMatrix(cm)

**Output :**

****

**Program 15:**

**Program to Implement Naïve Bayesian using Wine Quality Data.**

**The dataset is related to red variant of the Portuguese "Vinho Verde" wine.**

**Attribute Information: Input variables (based on physicochemical tests):**

**1. fixed acidity**

**2. volatile acidity**

**3. citric acid**

**4. residual sugar**

**5. chlorides**

**6. free sulfur dioxide**

**7. total sulfur dioxide**

**8. density**

**9. pH**

**10. sulphates**

**11. alcohol**

**Output variable (based on sensory data):**

**12. quality (score between 0 and 10)**

library(e1071)

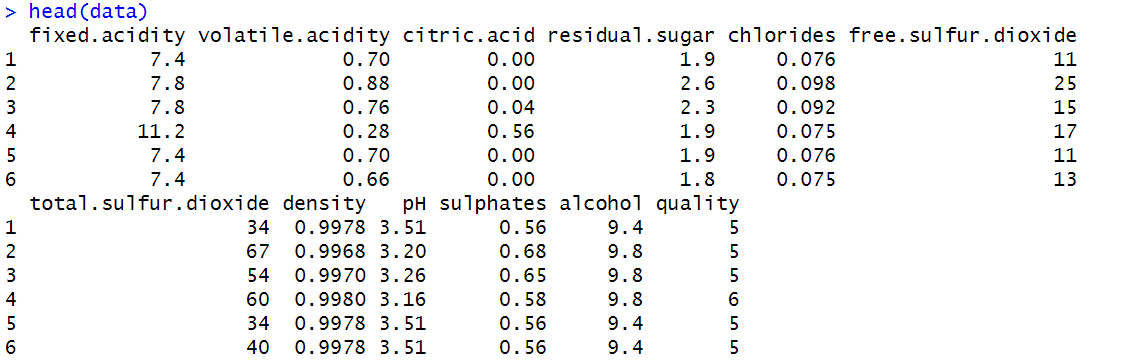
library(caTools)

library(caret)

data <- read.csv('winequality-red.csv')

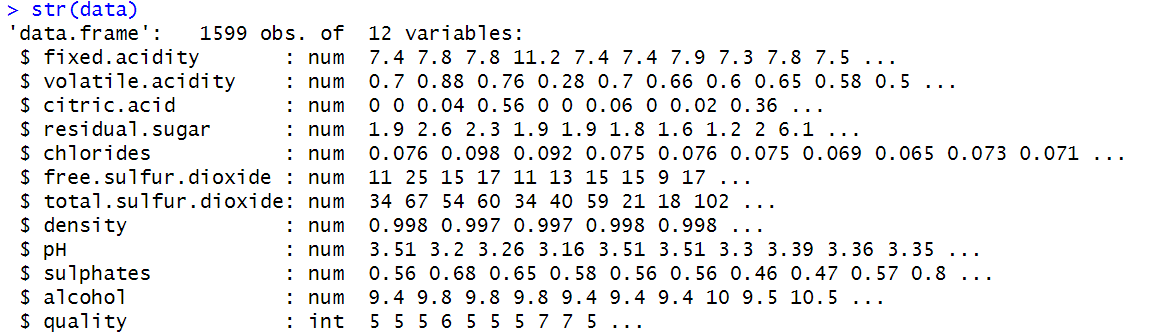
head(data)

**Output :**

****

str(data)

**Output :**

****

summary(data)

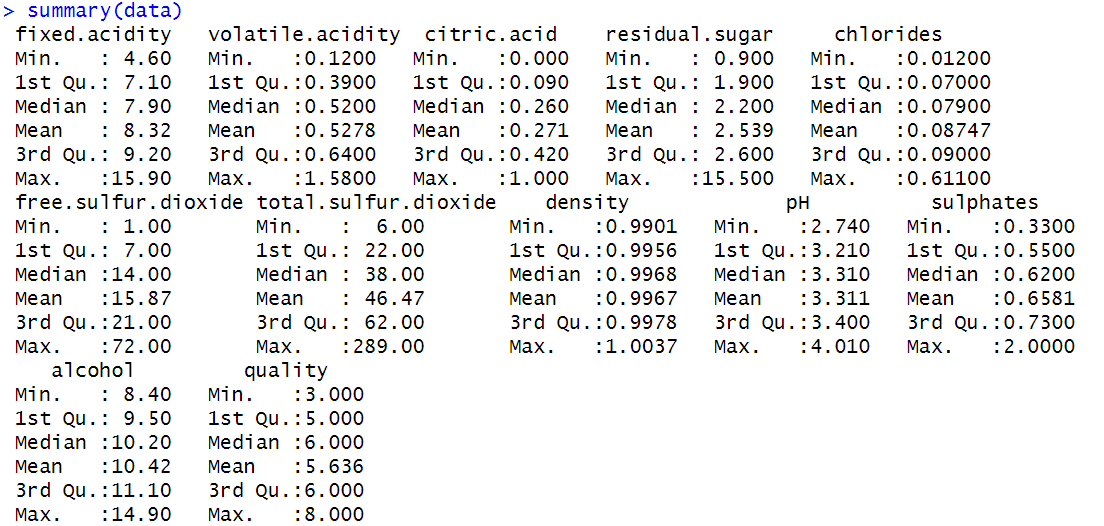
table(data$quality)

split <- sample.split(data$quality, SplitRatio = 0.7)

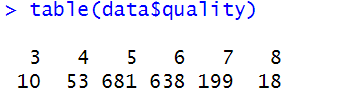
train <- subset(data, split == TRUE)

test <- subset(data, split == FALSE)

**Output :**

****

**Output :**

****

model <- naiveBayes(formula = train$quality ~ ., data = train)

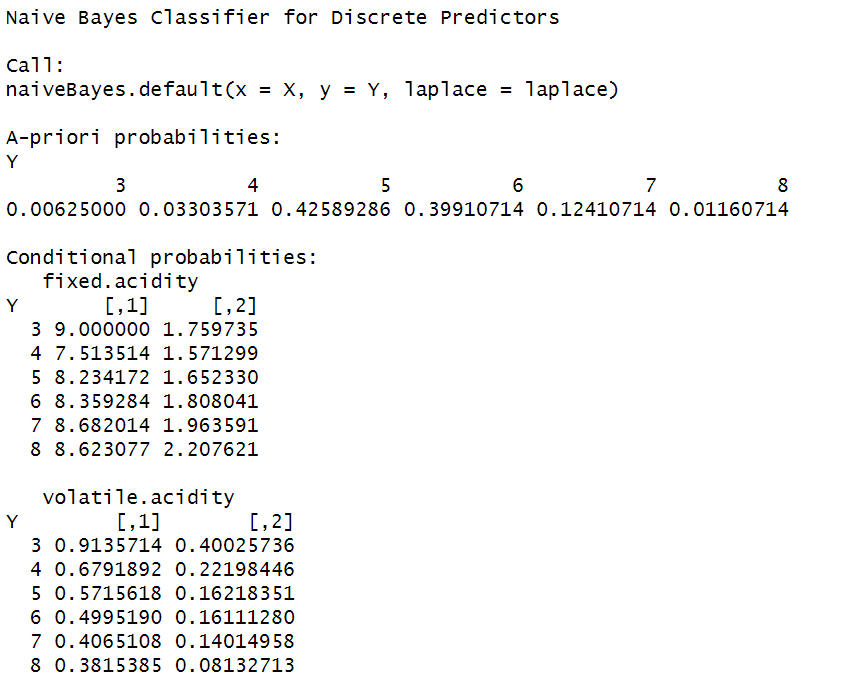
model

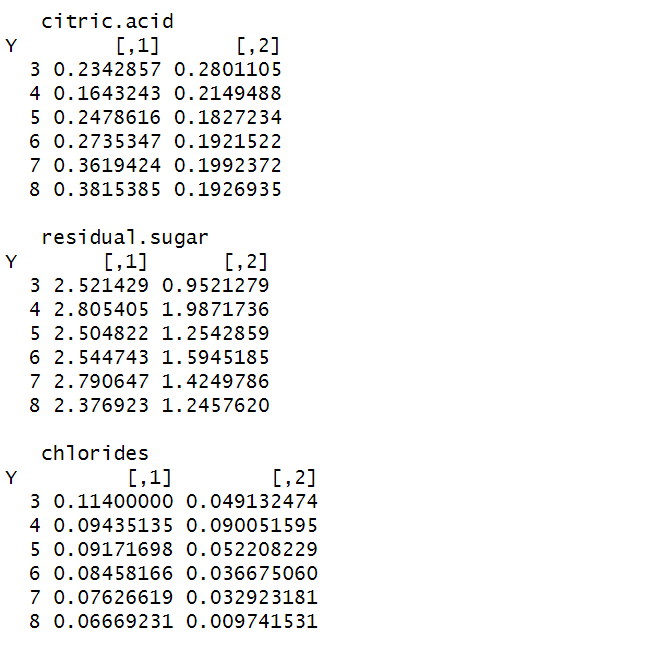
pred <- predict(model, newdata = test)

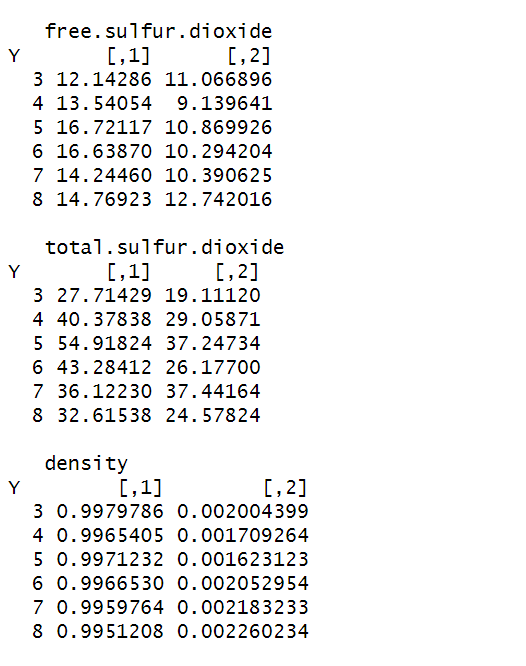
cm <- table(pred, test$quality)

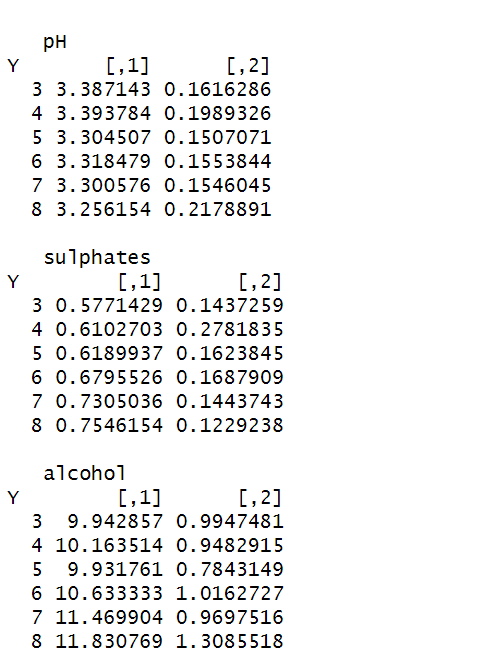
confusionMatrix(cm)

**Output :** Model

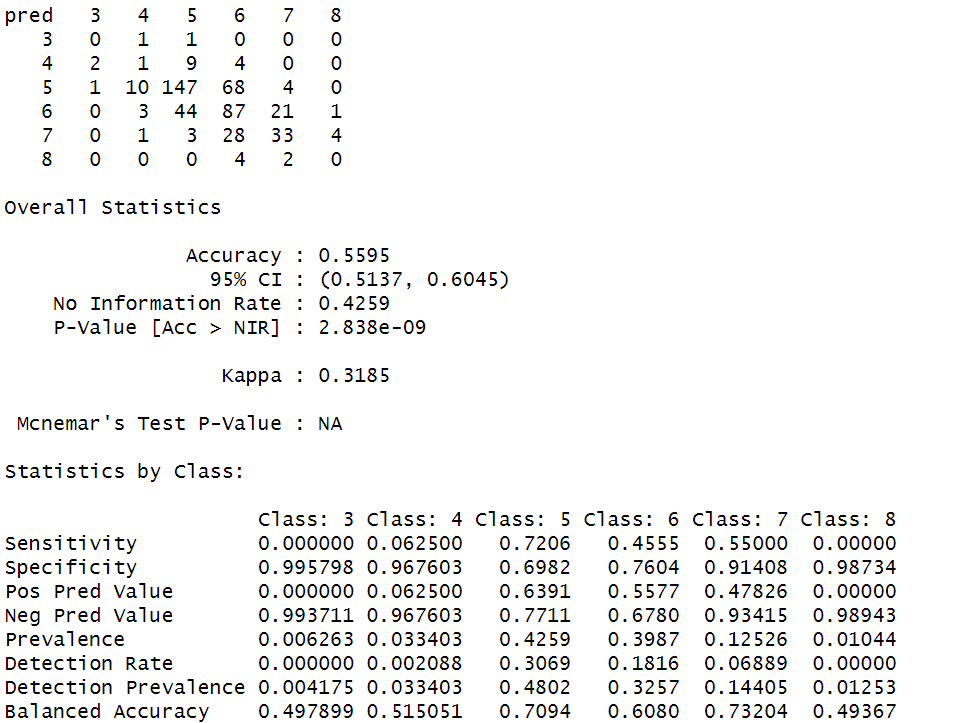
****

****

****

****

**Output :** Confusion Matrix

****

**Program 16:**

**Program to Implement Naïve Bayesian using Raisin Data.**

**Images of Kecimen and Besni raisin varieties grown in Turkey were obtained with CVS. A total of 900 raisin grains were used, including 450 pieces from both varieties. These images were subjected to various stages of pre-processing and 7 morphological features were extracted. These features have been classified using three different artificial intelligence techniques.**

**Attribute Information:**

**1. Area: Gives the number of pixels within the boundaries of the raisin.**

**2. Perimeter: It measures the environment by calculating the distance between the boundaries of the raisin and the pixels around it.**

**3. MajorAxisLength: Gives the length of the main axis, which is the longest line that can be drawn on the raisin.**

**4. MinorAxisLength: Gives the length of the small axis, which is the shortest line that can be drawn on the raisin.**

**5. Eccentricity: It gives a measure of the eccentricity of the ellipse, which has the same moments as raisins.**

**6. ConvexArea: Gives the number of pixels of the smallest convex shell of the region formed by the raisin.**

**7. Extent: Gives the ratio of the region formed by the raisin to the total pixels in the bounding box.**

**8. Class: Kecimen and Besni raisin.**

library(caTools)

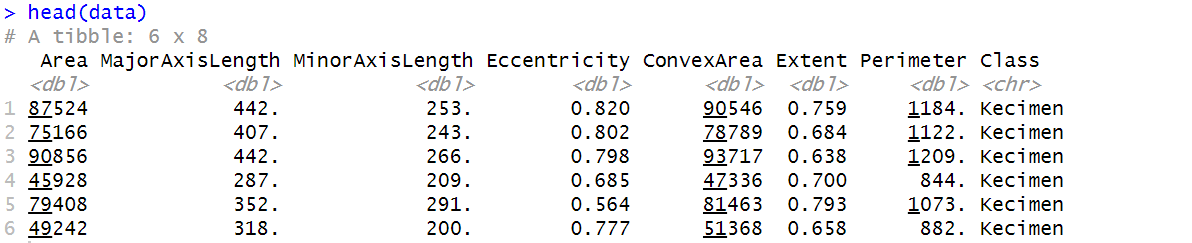
library(caret)

library(readxl)

data <- read\_excel("Raisin\_Dataset.xlsx")

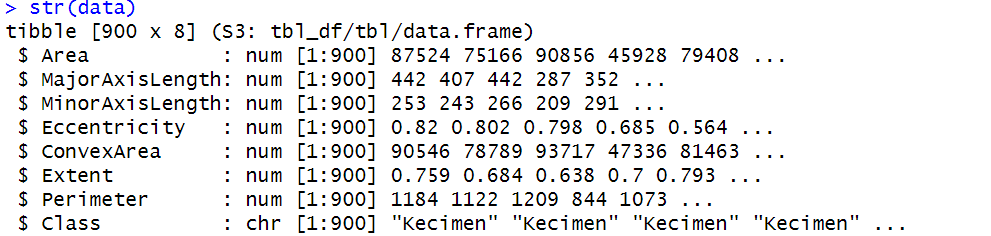
head(data)

**Output :**

****

str(data)

**Output :**

****

summary(data)

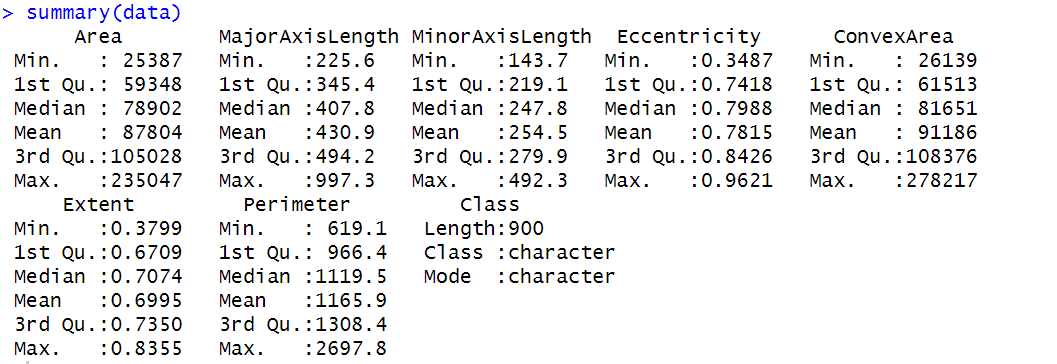
table(data$Class)

split <- sample.split(data$Class, SplitRatio = 0.7)

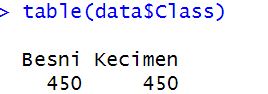
train <- subset(data, split == TRUE)

test <- subset(data, split == FALSE)

**Output :**

****

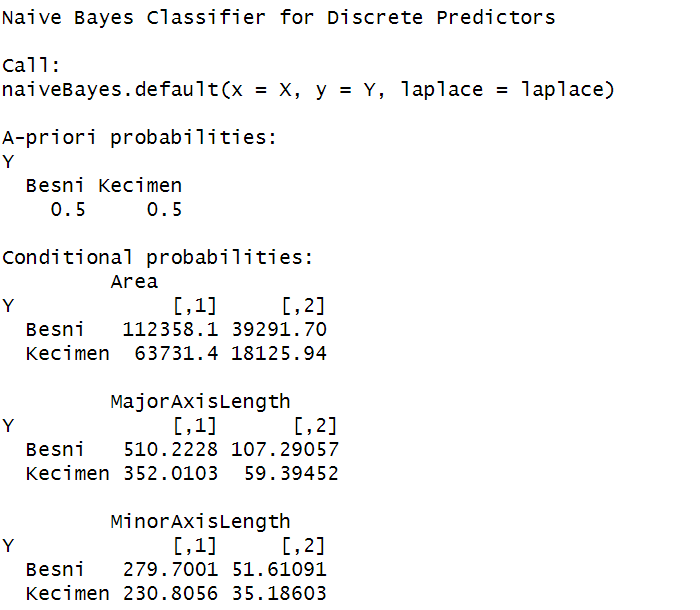
**Output :**

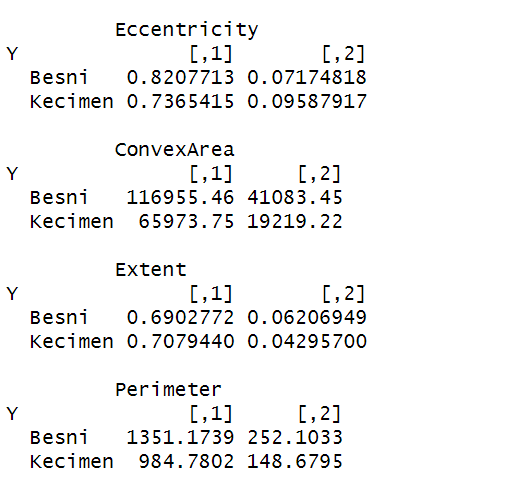
****

model <- naiveBayes(formula = train$Class ~., data = train)

model

**Output :** Model

****

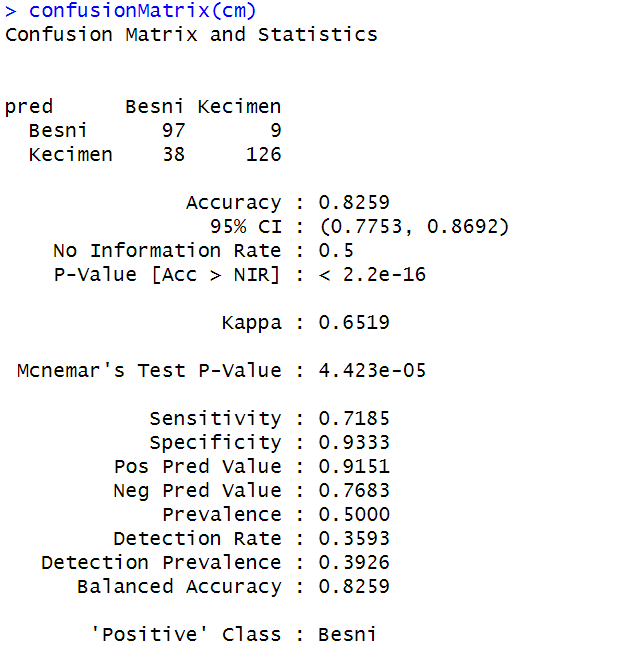
****

pred <- predict(model, newdata = test)

cm <- table(pred,test$Class)

confusionMatrix(cm)

**Output :** Confusion Matrix

****

**Program 17:**

**Program to Implement K-Nearest Neighbour using Iris Data.**

**This is perhaps the best-known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.**

**Predicted attribute: class of iris plant.**

**Attribute Information:**

**1. sepal length in cm**

**2. sepal width in cm**

**3. petal length in cm**

**4. petal width in cm**

**class: - Iris Setosa, Iris Versicolour, Iris Virginica**

library(e1071)

library(caTools)

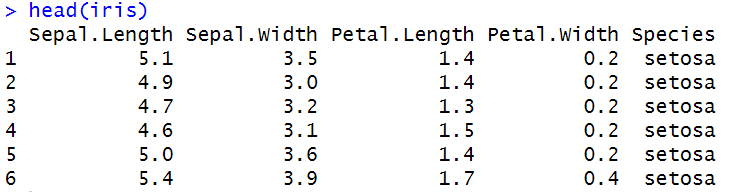
library(class)

library(caret)

data(iris)

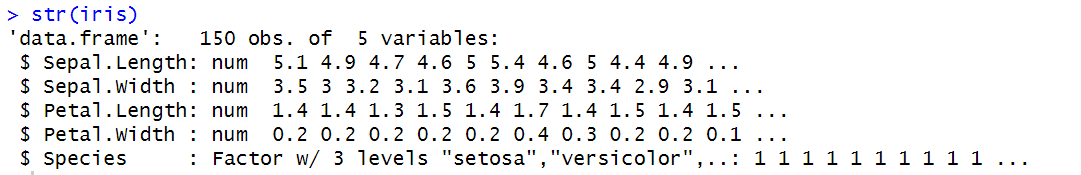
head(iris)

**Output :**

****

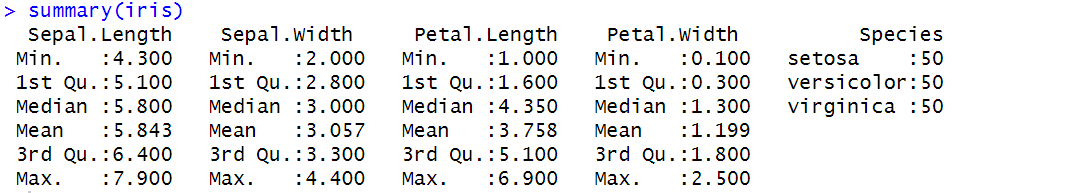
str(iris)

**Output :**

****

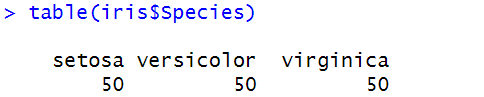
summary(iris)

**Output :**

****

table(iris$Species)

**Output :**

****

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

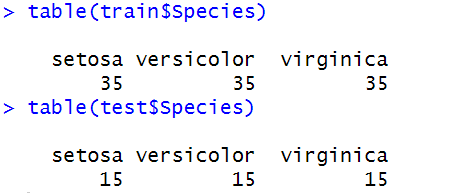
train <- subset(iris, split == TRUE)

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

table(train$Species)

table(test$Species)

**Output :**

****

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

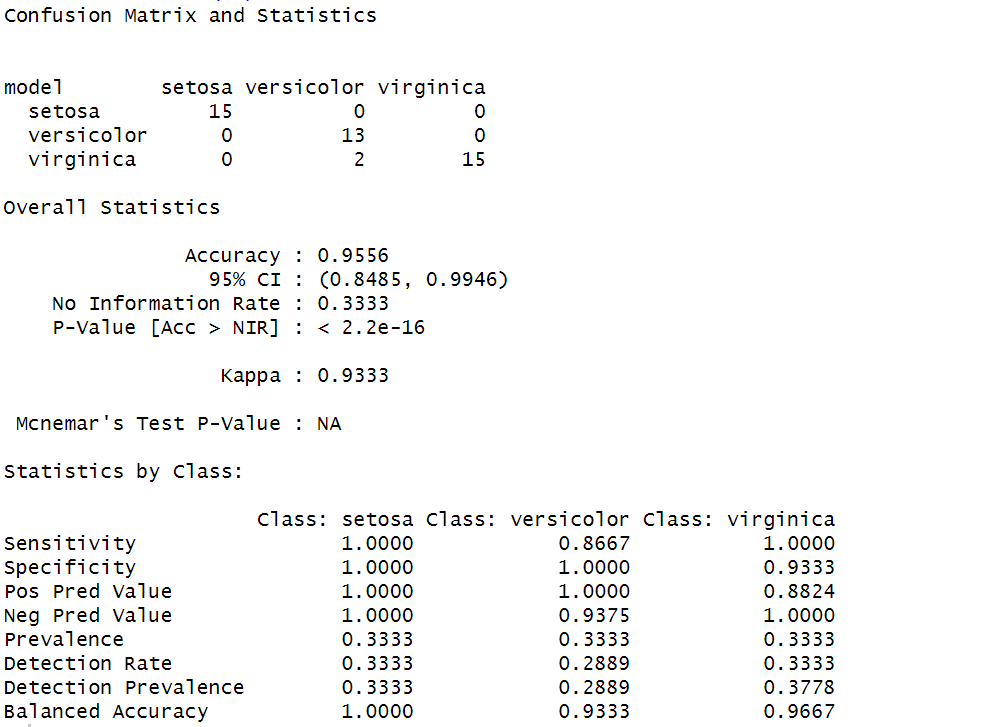
test\_scale <- scale(test[, 1:4])

model <- knn(train = train\_scale, test = test\_scale, cl = train$Species, k = 3)

cm <- table(model, test$Species)

confusionMatrix(cm)

**Output :**

****

**Program 18:**

**Program to Implement K-Nearest Neighbour using Raisin Data.**

**Images of Kecimen and Besni raisin varieties grown in Turkey were obtained with CVS. A total of 900 raisin grains were used, including 450 pieces from both varieties. These images were subjected to various stages of pre-processing and 7 morphological features were extracted. These features have been classified using three different artificial intelligence techniques.**

**Attribute Information:**

**1. Area: Gives the number of pixels within the boundaries of the raisin.**

**2. Perimeter: It measures the environment by calculating the distance between the boundaries of the raisin and the pixels around it.**

**3. MajorAxisLength: Gives the length of the main axis, which is the longest line that can be drawn on the raisin.**

**4. MinorAxisLength: Gives the length of the small axis, which is the shortest line that can be drawn on the raisin.**

**5. Eccentricity: It gives a measure of the eccentricity of the ellipse, which has the same moments as raisins.**

**6. ConvexArea: Gives the number of pixels of the smallest convex shell of the region formed by the raisin.**

**7. Extent: Gives the ratio of the region formed by the raisin to the total pixels in the bounding box.**

**8. Class: Kecimen and Besni raisin.**

library(e1071)

library(caTools)

library(class)

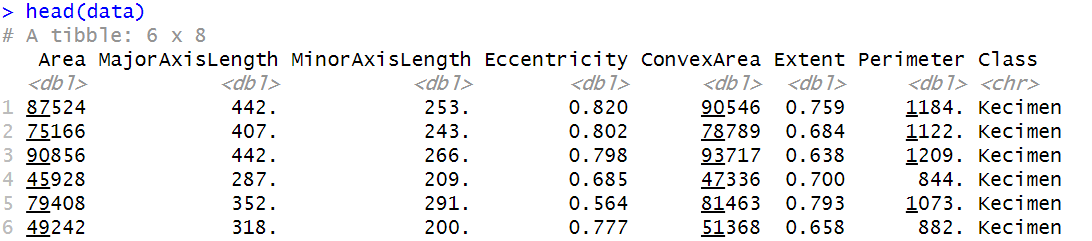
library(caret)

library(readxl)

data <- read\_excel('Raisin\_Dataset.xlsx')

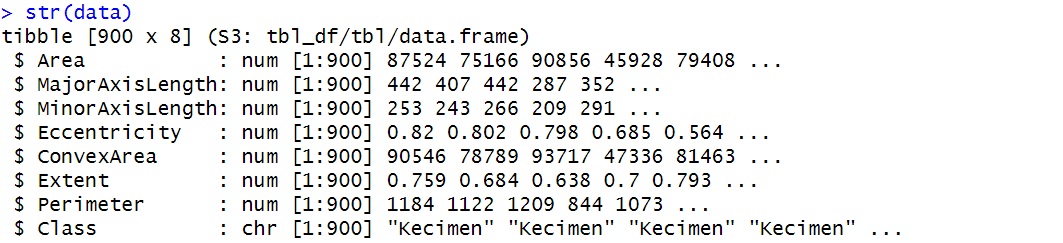
head(data)

**Output :**



str(data)

**Output :**

****

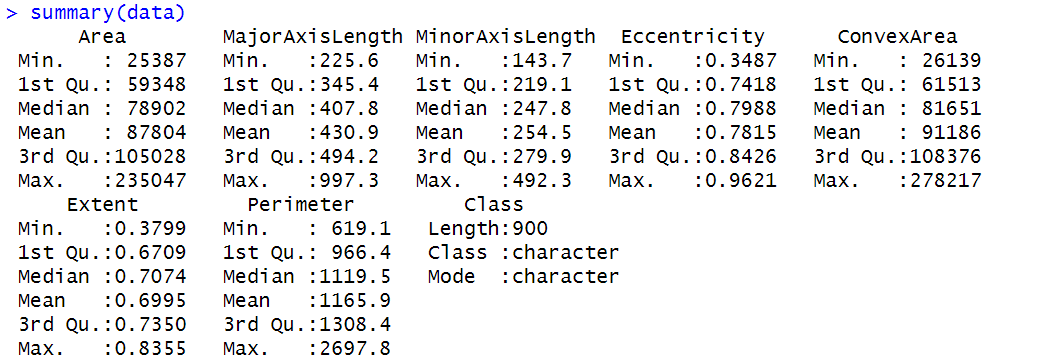
summary(data)

split <- sample.split(data$Class, SplitRatio = 0.7)

train <- subset(data, split == TRUE)

test <- subset(data, split == FALSE)

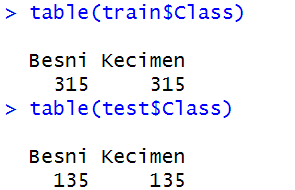
**Output :**

****

table(train$Class)

table(test$Class)

**Output :**

****

train\_scale <- scale(train[,1:7])

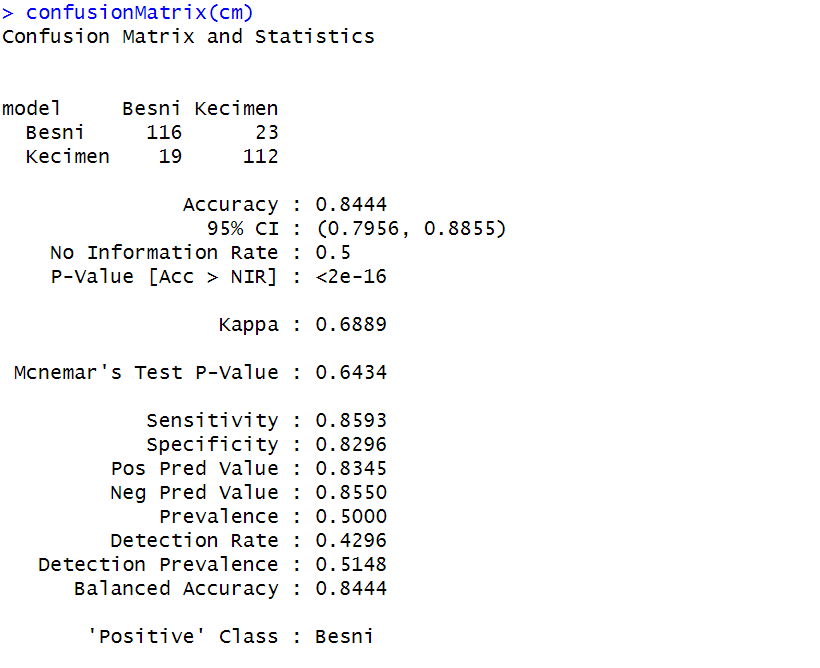
test\_scale <- scale(test[,1:7])

model <- knn(train = train\_scale, test = test\_scale, cl = train$Class, k = 3)

cm = table(model, test$Class)

confusionMatrix(cm)

**Output :** Confusion Matrix

****

**Program 19:**

**Program to Implement K-Nearest Neighbour using Wine Quality Data.**

**The dataset is related to red variant of the Portuguese "Vinho Verde" wine.**

**Attribute Information: Input variables (based on physicochemical tests):**

**1. fixed acidity**

**2. volatile acidity**

**3. citric acid**

**4. residual sugar**

**5. chlorides**

**6. free sulfur dioxide**

**7. total sulfur dioxide**

**8. density**

**9. pH**

**10. sulphates**

**11. alcohol**

**Output variable (based on sensory data):**

**12. quality (score between 0 and 10)**

library(e1071)

library(caTools)

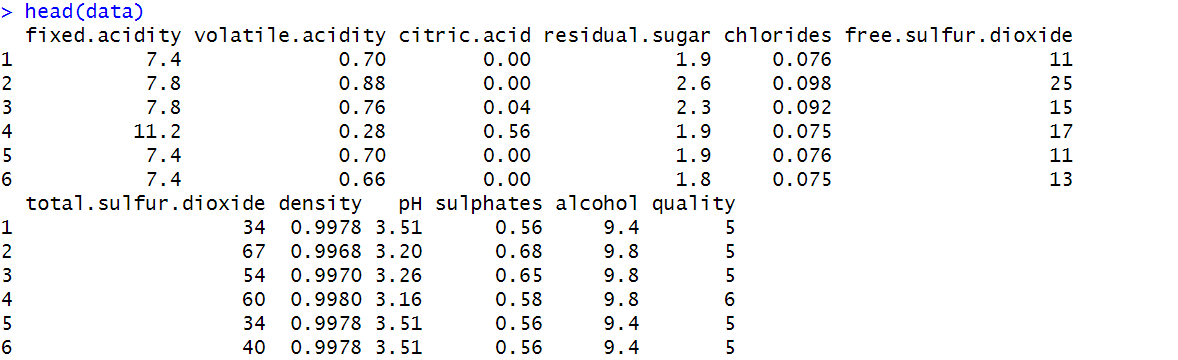
library(class)

library(caret)

data <- read.csv('winequality-red.csv')

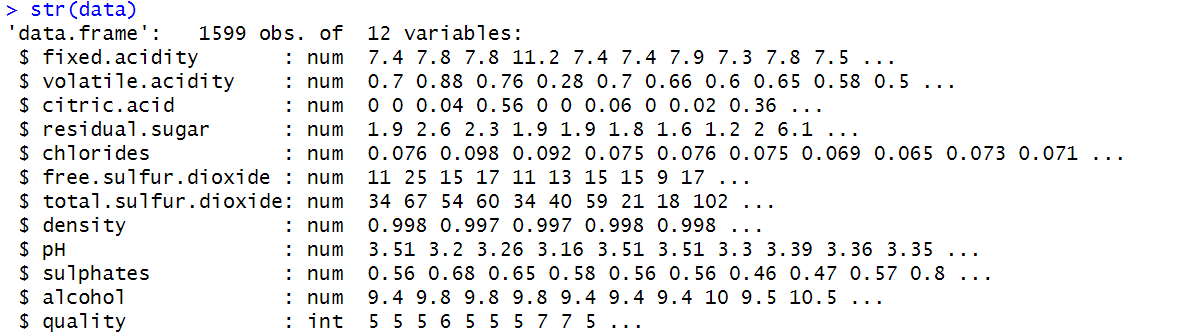
head(data)

**Output :**



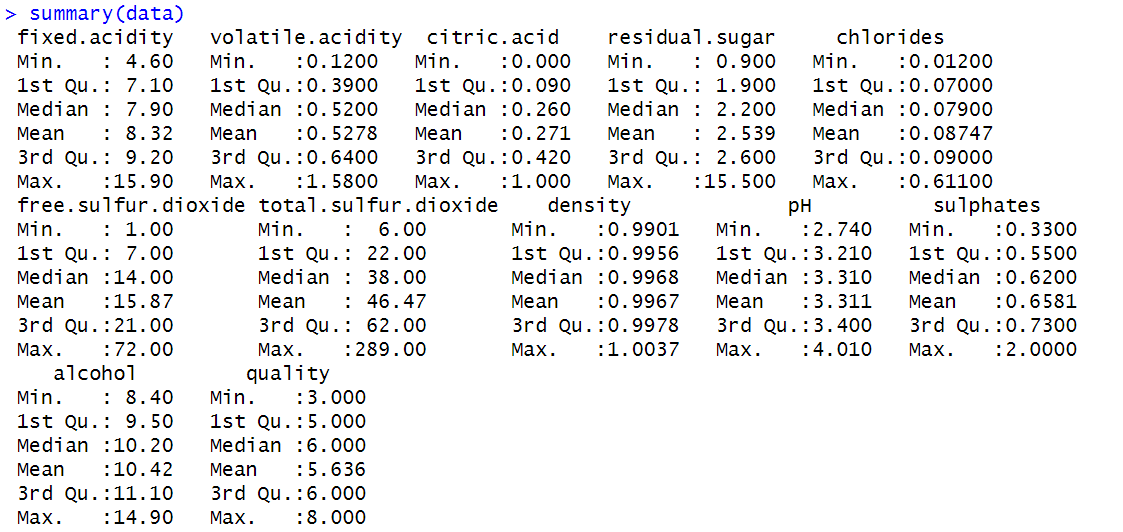
str(data)

**Output :**

****

summary(data)

**Output :**

****

split <- sample.split(data$quality, SplitRatio = 0.7)

train <- subset(data, split == TRUE)

test <- subset(data, split == FALSE)

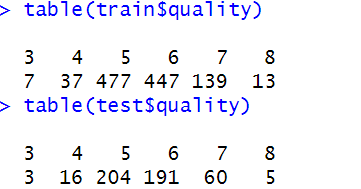
table(train$quality)

table(test$quality)

train\_scale <- scale(train[,1:11])

test\_scale <- scale(test[,1:11])

**Output :**

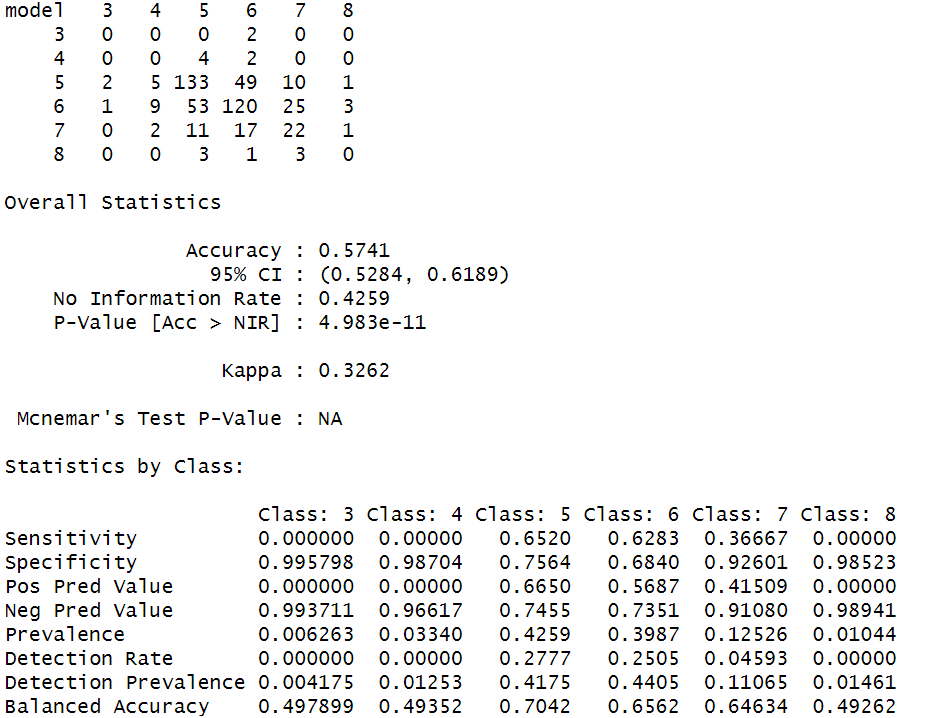
****

model <- knn(train = train\_scale, test = test\_scale, cl = train$quality, k = 3)

cm = table(model, test$quality)

confusionMatrix(cm)

**Output :** Confusion Matrix

****