Finolex Academy of Management & Technology, Ratnagiri

Department of MCA

Course:-MCAL13 Advanced Database Management System Lab

Practical – 08

Title: - Data Analysis in R

Aim: - To perform data analysis using R programming.

Lab Objectives: -

Students will understand following R programming concepts:

I. Regression Technique
II. Market basket analysis using Apriori algorithm
III. Naïve Bayes Classification
IV. K means Clustering

Description: -

I. Linear Regression in R

Regression analysis is a very widely used statistical tool to establish a relationship model between two variables.

One of these variable is called predictor variable whose value is gathered through experiments.

The other variable is called response variable whose value is derived from the predictor variable.

Linear regression is used to predict the value of an outcome variable *Y* based on one or more input predictor variables *X*.

Mathematically a linear relationship represents a straight line when plotted as a graph.

The general mathematical equation for a linear regression is −

=b0 + b1 *

Following is the description of the parameters used −

- y is the response variable.
- x is the predictor variable.
- b1 – slope
- b0 - intercept
- Collectively, they are called *regression coefficients*.

For example, we want to predict weight (y) from height (x), the linear regression model can be represented by the following equation

Weight= b0 + b1 * height

- b1 is called slope because it defines the slope of the line or how x translates into a y i.e by how much y is affected by change in x

The goal is to find best estimates for the coefficients to minimize the error in predicting y from x

These coefficients can be solved by the method of least squares which estimates the best fitting straight line as the one that minimizes the error between the actual data & estimates of line

$$b1=\frac{\sum_{=}(\ -\bar{}\ )*(\ -\bar{}\ )}{\sum_{=}(\ -\ )}$$ **where, - mean of x1,x2,... - mean of y1,y2,...**

$$b0=\bar{}-b1*\bar{}$$

If b1 > 0, then x(predictor) and y(target) have a positive relationship.
That is increase in x will increase y.
If b1 < 0, then x(predictor) and y(target) have a negative relationship.
That is increase in x will decrease y.

II. Market Basket Analysis in R

The increasing volume of data and the growing importance of retail analytics made it easy for retailers to know their customers better.
Data can help retailers to understand customer behavior, plan and promote products, increase sales, improve customer experience, and optimize supply chain performance.
There are many algorithms and techniques used in retail that help uncover better insights and predict future events.
One of the key and widely used techniques in retail is Market Basket Analysis.
It works by searching for combinations of items that often happen in transactions together.
Market Basket Analysis is a technique that is used to discover the association between items.
In simplest terms, it allows retailers to identify a relationship between items that generally people buy together.
For instance, if one person buys ‘bread’, he/she more likely to buy ‘butter’ or ‘jam’ which is predicted as a ‘go-along’ item with the purchase
To implement this, associate rule mining is used.
Association Rule Mining is a rule-based machine learning method to find associations and relationships between large sets of items.
This rule also shows how frequently an item occurs in the itemset based on the occurrences of other items in a transaction.
Association rules are widely used to analyze basket or transaction data to discover strong rules based on the interestingness and frequency of occurrences.
Association rules can be understood as the “if this, then that” rule.
For example, if a user buys coffee and sugar, then he/she is likely to buy milk.
Multiple techniques and algorithms are being used in Market Basket Analysis.
One of the main objectives is to predict the likelihood of items being purchased together by users.
APRIORI is the by far widely-used and well-known association rule algorithm.
It finds frequent itemsets in transactions and identifies association rules between those items.

It scans the database many times which leads to increased time and reduced performance as it is a computationally expensive step because of a large database.

The association rule has primarily three measures to decide the degree of confidence, these are:

- Support
- Confidence
- Lift

Support:

- This is one of the important measures to determine how frequently an itemset occurs in the transaction as a percentage of all transactions.
- Support is the number of transactions that include both {A} and {B} parts as a percentage of the total number of transactions.

$$\textbf{Support} = \frac{(\mathbf{A} + \mathbf{B})}{\textbf{Total}}$$

Confidence:

- This rule is the ratio of the number of transactions that include items in {A} and {B} to the number of transactions that include items in {A}.
- It can be understood as to how often items in B appear in transactions that contain A only. It is a conditional probability.

$$\textbf{Confidence} = \frac{(\mathbf{A} + \mathbf{B})}{\mathbf{A}}$$

Lift:

- This third measure, lift or lift ratio is the ratio of confidence to expected confidence.
- We can say that this rule shows us how much better a rule is at predicting the result than just assuming it.
- Greater lift value tells how strong the association is.
- It shows us the rate of confidence that B will be purchased given that A was purchased.
- In other way Lift = Confidence(A=>B) / Support(B)

$$\textbf{Lift} = \left(\frac{\left(\frac{(\mathbf{A} + \mathbf{B})}{\mathbf{A}}\right)}{\left(\frac{\mathbf{B}}{\textbf{Total}}\right)}\right)$$

III. K-Means algorithm in R

Clustering is an unsupervised learning technique.

Unsupervised learning means that there is no outcome to be predicted, and the algorithm just tries to find patterns in the data.

It is the task of grouping together a set of objects in a way that objects in the same cluster are more similar to each other than to objects in other clusters.

Similarity is an amount that reflects the strength of relationship between two data objects.

Clustering is mainly used for exploratory data mining.

It is used in many fields such as machine learning, pattern recognition, image analysis, information retrieval, bio-informatics, data compression, and computer graphics.

In k means clustering, we have to the specify the number of clusters we want the data to be grouped into.

The algorithm randomly assigns each observation to a cluster, and finds the centroid of each cluster.

Then, the algorithm iterates through two steps:

- ○ Reassign data points to the cluster whose centroid is closest.
- ○ Calculate new centroid of each cluster.

These two steps are repeated till the within cluster variation cannot be reduced any further.

The within cluster variation is calculated as the sum of the euclidean distance between the data points and their respective cluster centroids.

IV. Naïve Bayes Classifier Algorithm

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.

It is mainly used in text classification that includes a high-dimensional training dataset.

Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

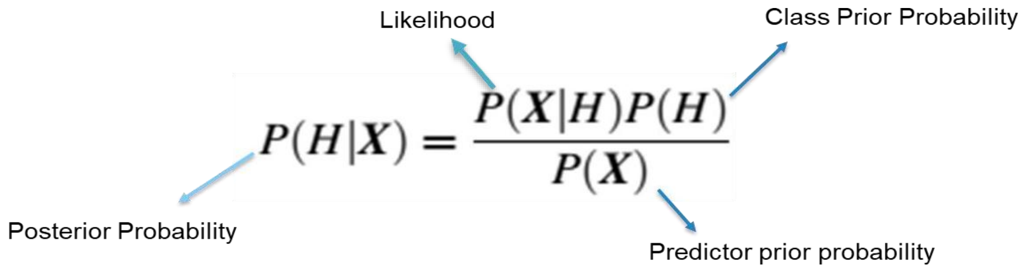
It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Bayes' Theorem:

Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

The formula for Bayes' theorem is given as:


Likelihood
Class Prior Probability
$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$
Posterior Probability
Predictor prior probability

Working of Naïve Bayes' Classifier:

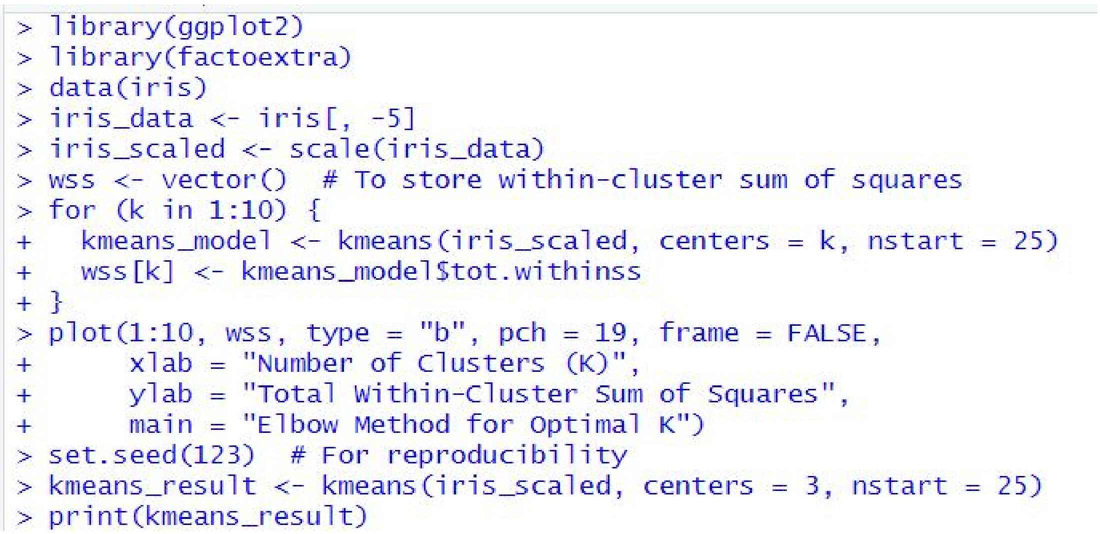
Working of Naïve Bayes' Classifier can be understood with the help of the below example: Suppose we have a dataset of weather conditions and corresponding target variable "Play". So, using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions.

So, to solve this problem, we need to follow the below steps:

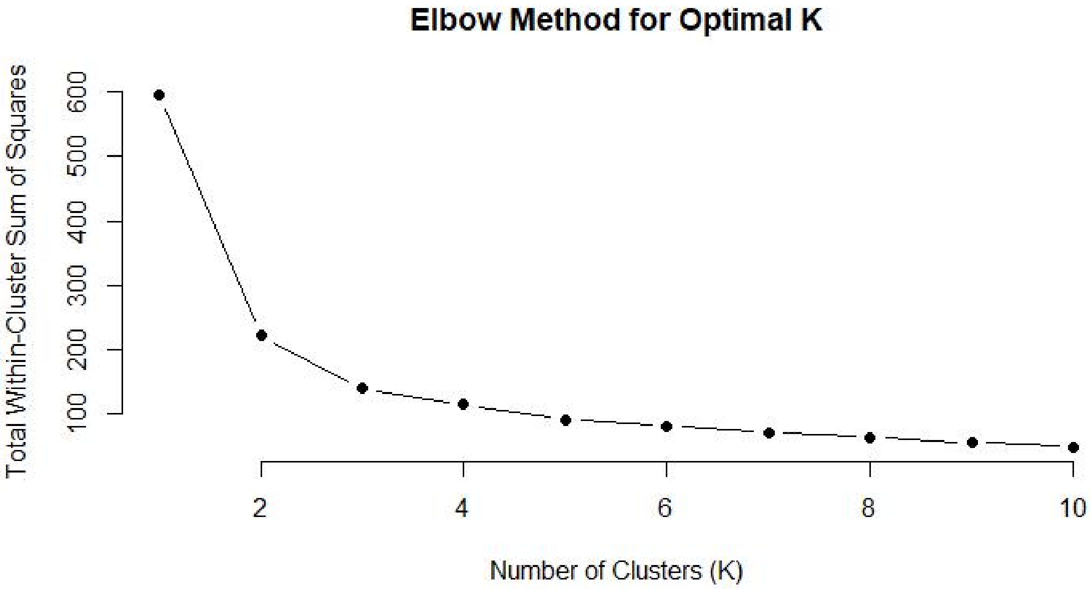
- Construct a frequency table for each attribute against the target.
- Transform the frequency tables to likelihood tables
- Finally use the Naive Bayesian equation to calculate the posterior probability for each class.
- The class with the highest posterior probability is the outcome of prediction.

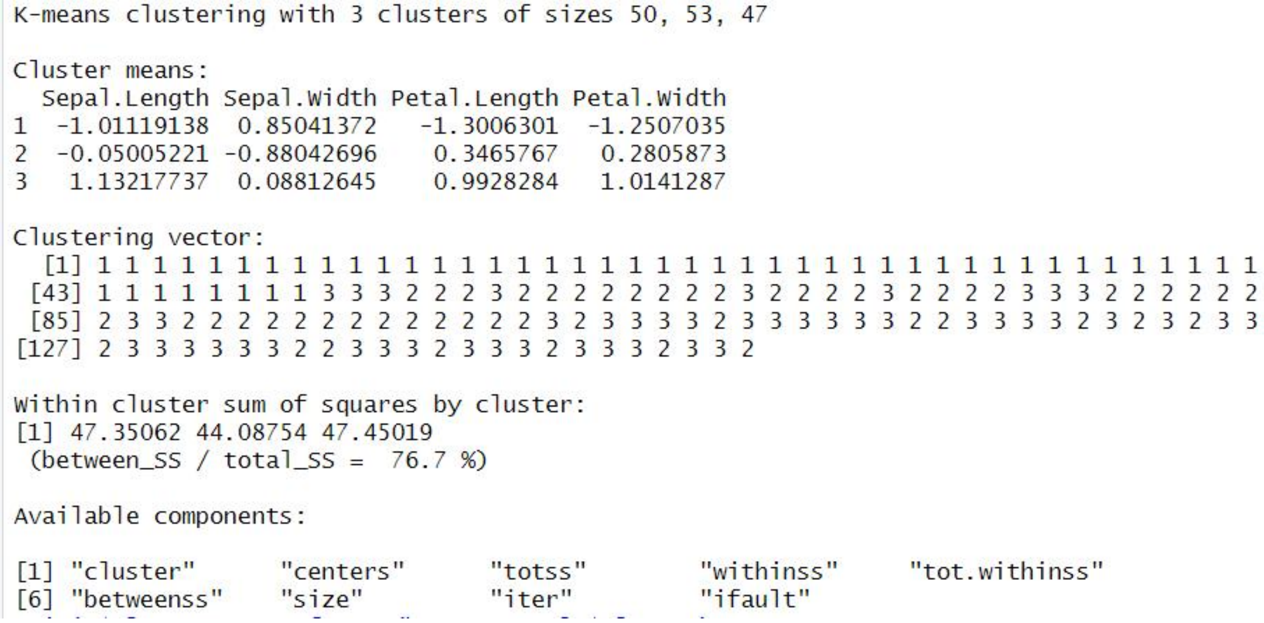
Exercises

1. **Write a program to perform k means clustering on iris dataset. Perform data pre-processing if required.**

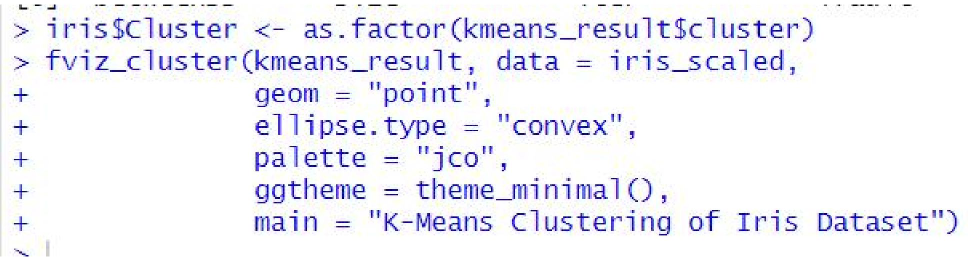


```
> library(ggplot2)
> library(factoextra)
> data(iris)
> iris_data <- iris[, -5]
> iris_scaled <- scale(iris_data)
> wss <- vector()  # To store within-cluster sum of squares
> for (k in 1:10) {
+   kmeans_model <- kmeans(iris_scaled, centers = k, nstart = 25)
+   wss[k] <- kmeans_model$tot.withinss
+ }
> plot(1:10, wss, type = "b", pch = 19, frame = FALSE,
+      xlab = "Number of Clusters (K)",
+      ylab = "Total Within-Cluster Sum of Squares",
+      main = "Elbow Method for Optimal K")
> set.seed(123)  # For reproducibility
> kmeans_result <- kmeans(iris_scaled, centers = 3, nstart = 25)
> print(kmeans_result)
```

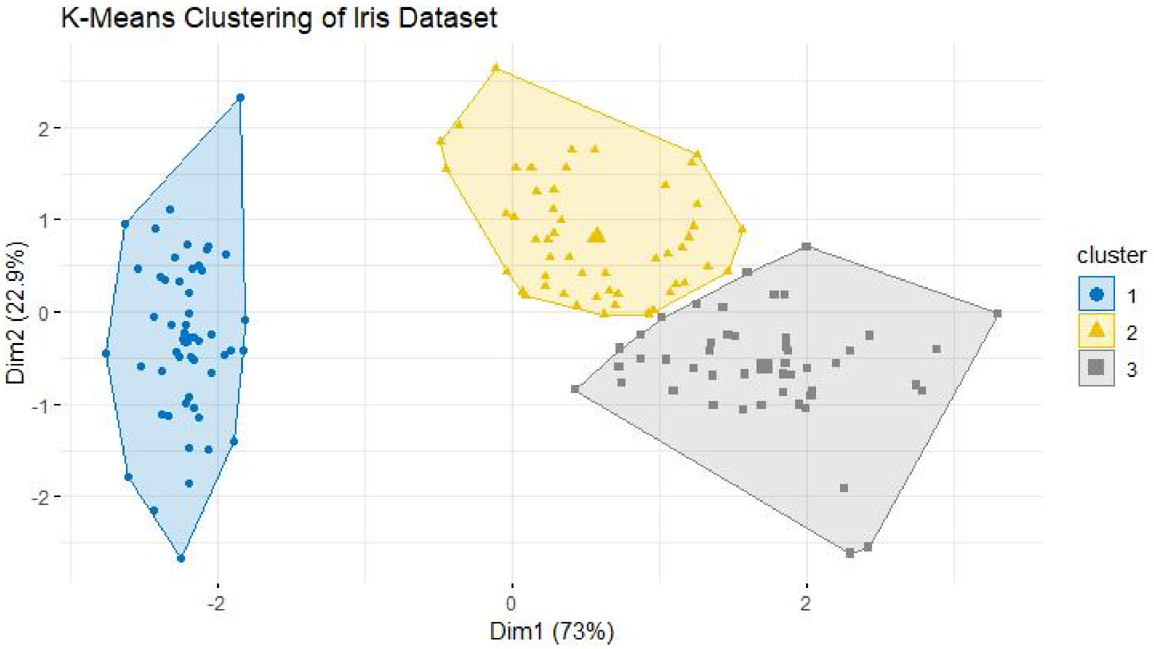




```
K-means clustering with 3 clusters of sizes 50, 53, 47

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1  -1.01119138  0.85041372   -1.3006301  -1.2507035
2  -0.05005221 -0.88042696    0.3465767   0.2805873
3   1.13217737  0.08812645    0.9928284   1.0141287

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [43] 1 1 1 1 1 1 1 1 3 3 3 2 2 2 3 2 2 2 2 2 2 2 2 3 2 2 2 2 3 2 2 2 2 3 3 3 2 2 2 2 2 2
 [85] 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3
[127] 2 3 3 3 3 3 3 2 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 2

Within cluster sum of squares by cluster:
[1] 47.35062 44.08754 47.45019
 (between_SS / total_SS =  76.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

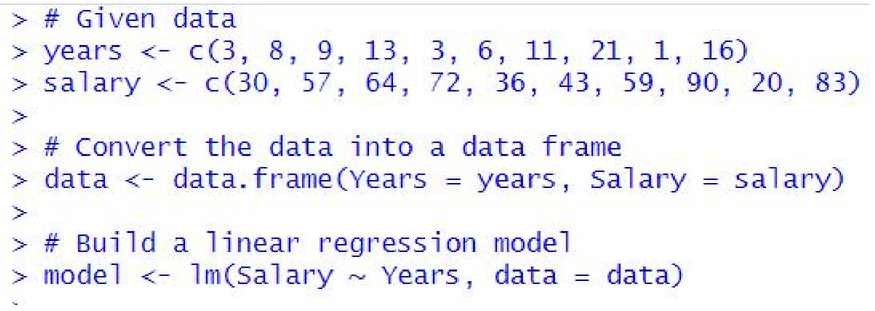


```
> iris$Cluster <- as.factor(kmeans_result$cluster)
> fviz_cluster(kmeans_result, data = iris_scaled,
+              geom = "point",
+              ellipse.type = "convex",
+              palette = "jco",
+              ggtheme = theme_minimal(),
+              main = "K-Means Clustering of Iris Dataset")
```

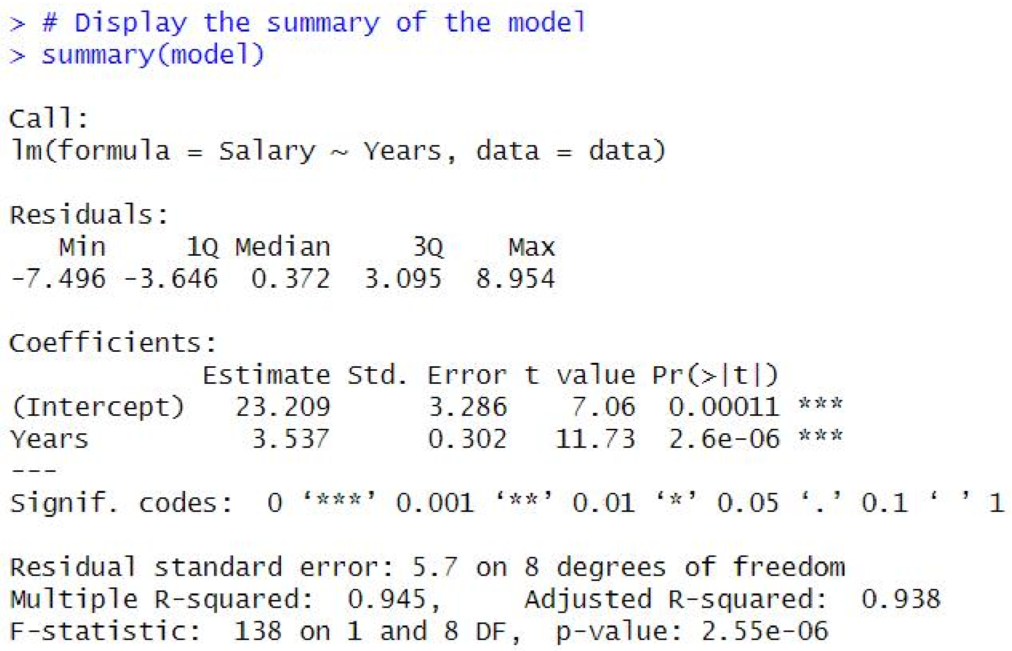


2 mplement Regression Classification for following example using R years=(3,8,9,13,3,6,11,21,1,16) salary=(30,57,64,72,36,43,59,90,20,83)

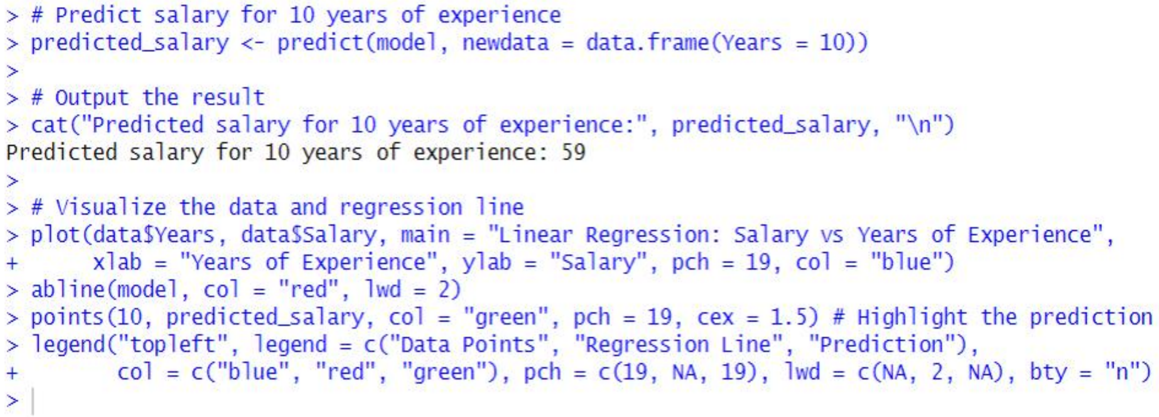
Predict salary of a person having 10 years of experience in a company.



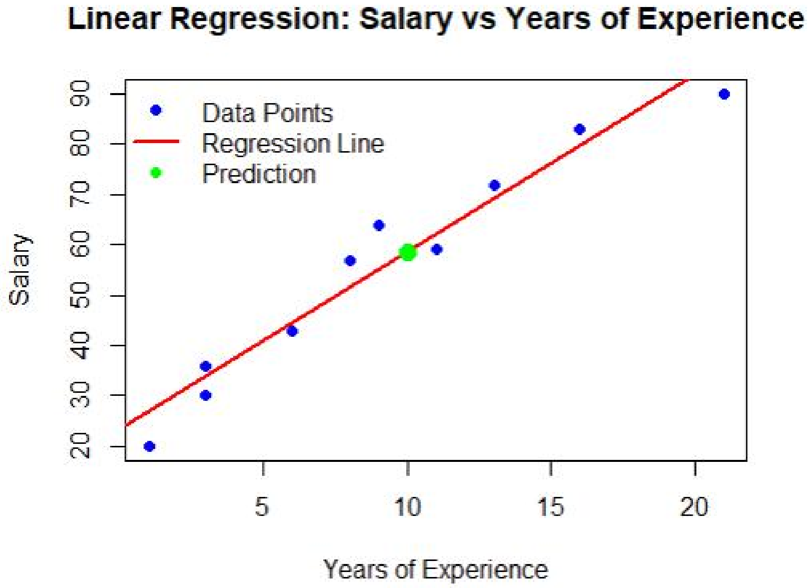
```
> # Given data
> years <- c(3, 8, 9, 13, 3, 6, 11, 21, 1, 16)
> salary <- c(30, 57, 64, 72, 36, 43, 59, 90, 20, 83)
>
> # Convert the data into a data frame
> data <- data.frame(Years = years, Salary = salary)
>
> # Build a linear regression model
> model <- lm(Salary ~ Years, data = data)
```



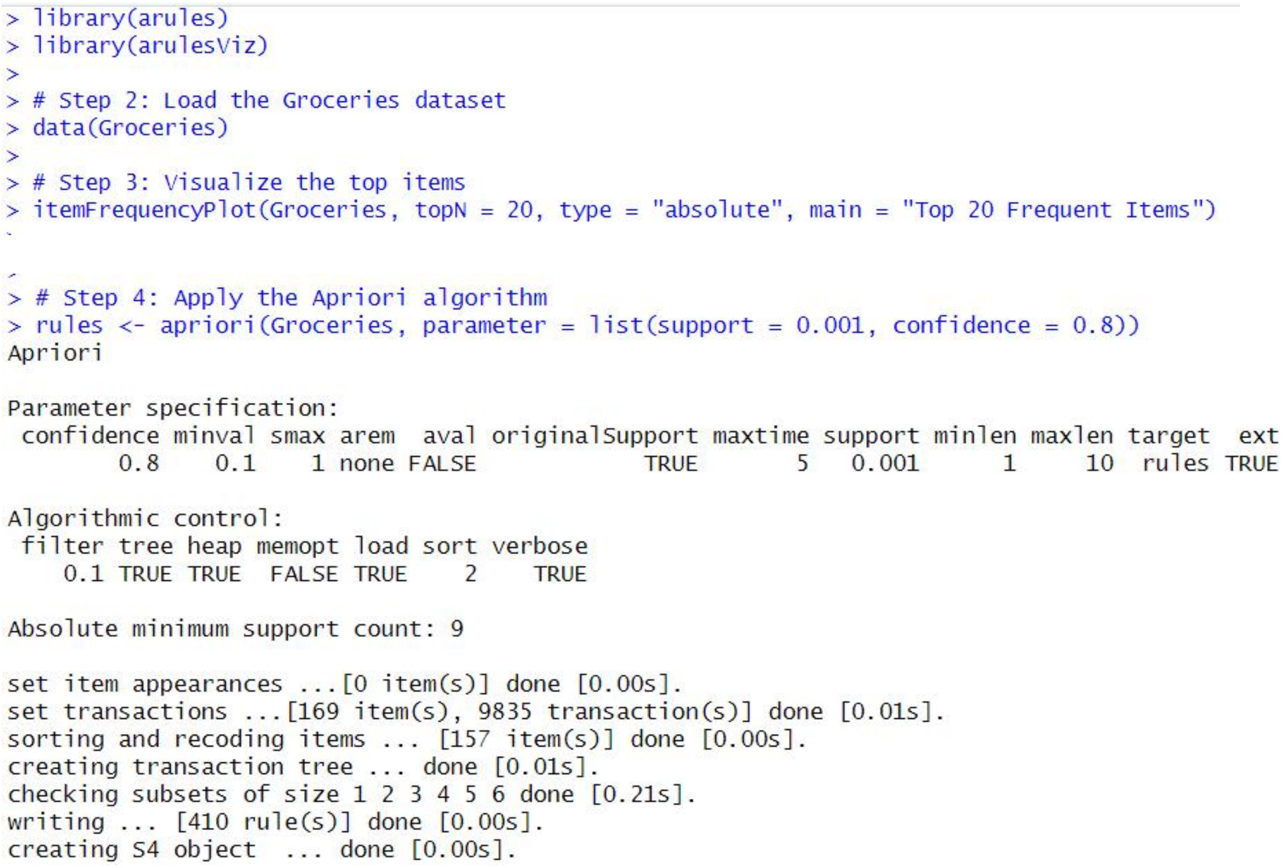
```
> # Display the summary of the model
> summary(model)

Call:
lm(formula = Salary ~ Years, data = data)

Residuals:
   Min     1Q Median     3Q    Max 
-7.496 -3.646  0.372  3.095  8.954 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   23.209      3.286    7.06  0.00011 ***
Years          3.537      0.302   11.73  2.6e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.7 on 8 degrees of freedom
Multiple R-squared:  0.945,	Adjusted R-squared:  0.938 
F-statistic:  138 on 1 and 8 DF,  p-value: 2.55e-06
```



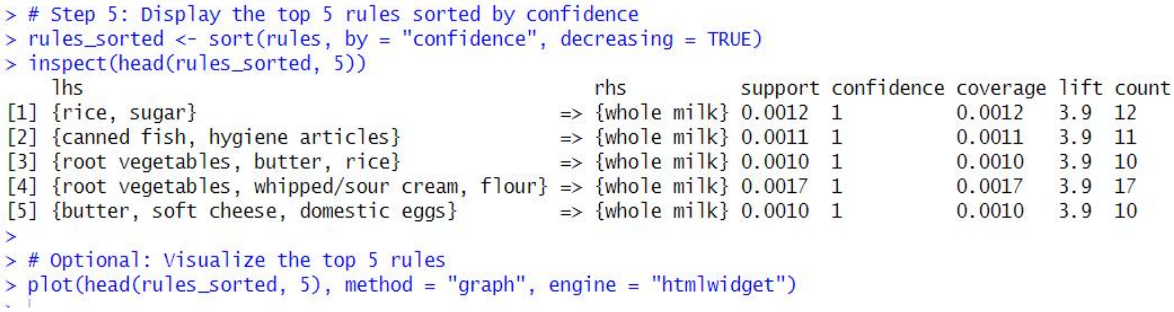
```
> # Predict salary for 10 years of experience
> predicted_salary <- predict(model, newdata = data.frame(Years = 10))
>
> # Output the result
> cat("Predicted salary for 10 years of experience:", predicted_salary, "\n")
Predicted salary for 10 years of experience: 59
>
> # Visualize the data and regression line
> plot(data$Years, data$Salary, main = "Linear Regression: Salary vs Years of Experience",
+      xlab = "Years of Experience", ylab = "Salary", pch = 19, col = "blue")
> abline(model, col = "red", lwd = 2)
> points(10, predicted_salary, col = "green", pch = 19, cex = 1.5) # Highlight the prediction
> legend("topleft", legend = c("Data Points", "Regression Line", "Prediction"),
+        col = c("blue", "red", "green"), pch = c(19, NA, 19), lwd = c(NA, 2, NA), bty = "n")
> |
```



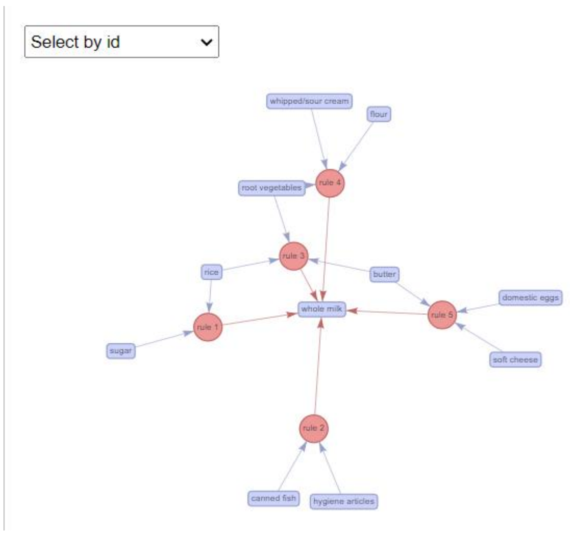
3Write a program to perform market basket analysis on Groceries dataset and display the top 5 important rules after sorting by confidence.



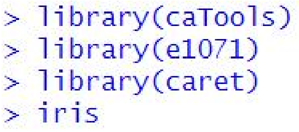
```
> library(arules)
> library(arulesViz)
>
> # Step 2: Load the Groceries dataset
> data(Groceries)
>
> # Step 3: Visualize the top items
> itemFrequencyPlot(Groceries, topN = 20, type = "absolute", main = "Top 20 Frequent Items")


> # Step 4: Apply the Apriori algorithm
> rules <- apriori(Groceries, parameter = list(support = 0.001, confidence = 0.8))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.8    0.1    1 none FALSE            TRUE       5   0.001      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 9

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 5 6 done [0.21s].
writing ... [410 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```



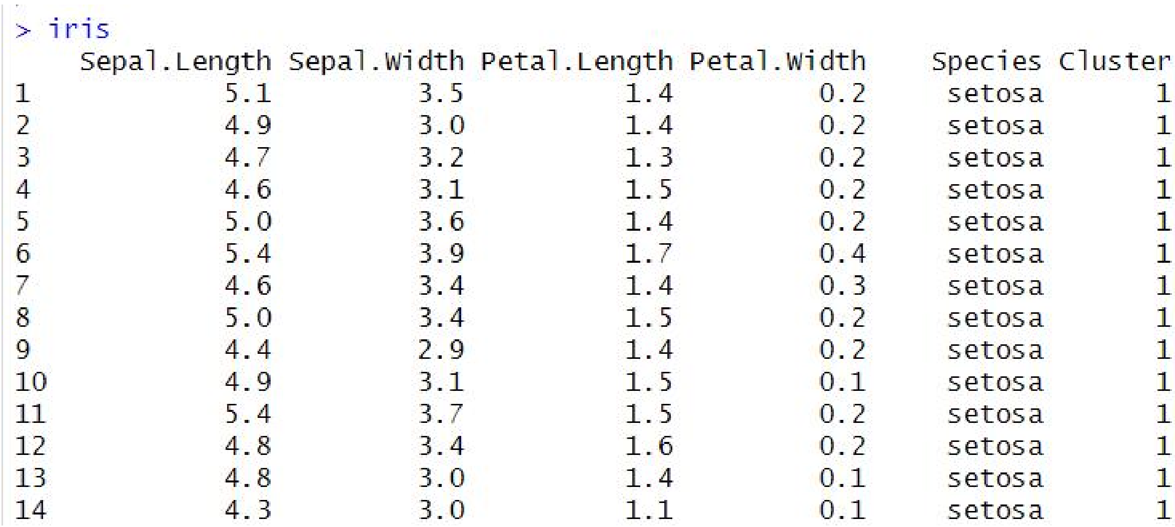
```
> # Step 5: Display the top 5 rules sorted by confidence
> rules_sorted <- sort(rules, by = "confidence", decreasing = TRUE)
> inspect(head(rules_sorted, 5))
    lhs                                            rhs          support confidence coverage lift count
[1] {rice, sugar}                               => {whole milk} 0.0012  1          0.0012   3.9  12
[2] {canned fish, hygiene articles}             => {whole milk} 0.0011  1          0.0011   3.9  11
[3] {root vegetables, butter, rice}             => {whole milk} 0.0010  1          0.0010   3.9  10
[4] {root vegetables, whipped/sour cream, flour} => {whole milk} 0.0017  1          0.0017   3.9  17
[5] {butter, soft cheese, domestic eggs}        => {whole milk} 0.0010  1          0.0010   3.9  10
>
> # Optional: Visualize the top 5 rules
> plot(head(rules_sorted, 5), method = "graph", engine = "htmlwidget")
```



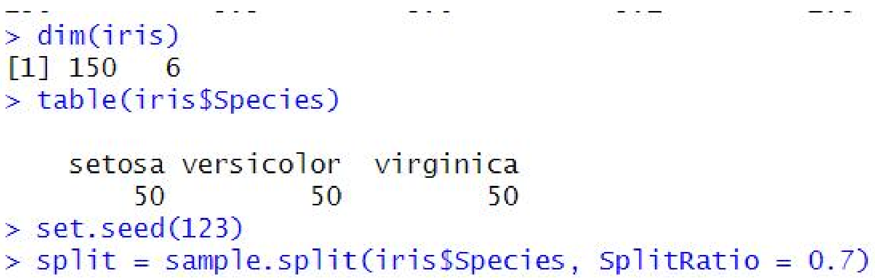
4 .Write a Program to perform naïve bayes classification on iris dataset. Perform data pre-processing if required.



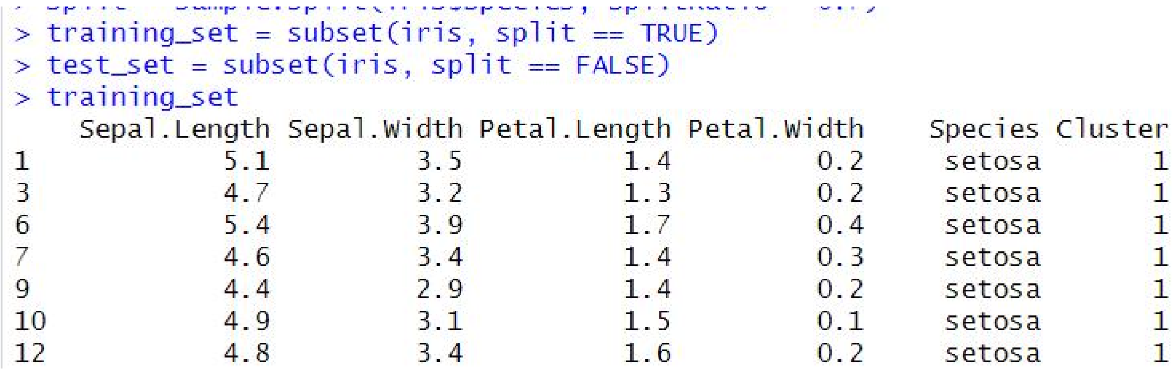
```
> library(caTools)
> library(e1071)
> library(caret)
> iris
```



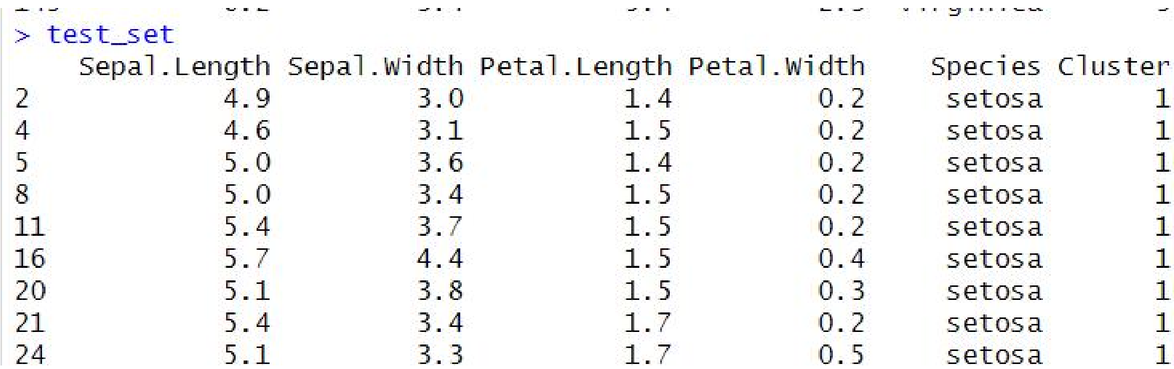
```
> iris
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species Cluster
1            5.1         3.5          1.4         0.2     setosa       1
2            4.9         3.0          1.4         0.2     setosa       1
3            4.7         3.2          1.3         0.2     setosa       1
4            4.6         3.1          1.5         0.2     setosa       1
5            5.0         3.6          1.4         0.2     setosa       1
6            5.4         3.9          1.7         0.4     setosa       1
7            4.6         3.4          1.4         0.3     setosa       1
8            5.0         3.4          1.5         0.2     setosa       1
9            4.4         2.9          1.4         0.2     setosa       1
10           4.9         3.1          1.5         0.1     setosa       1
11           5.4         3.7          1.5         0.2     setosa       1
12           4.8         3.4          1.6         0.2     setosa       1
13           4.8         3.0          1.4         0.1     setosa       1
14           4.3         3.0          1.1         0.1     setosa       1
```



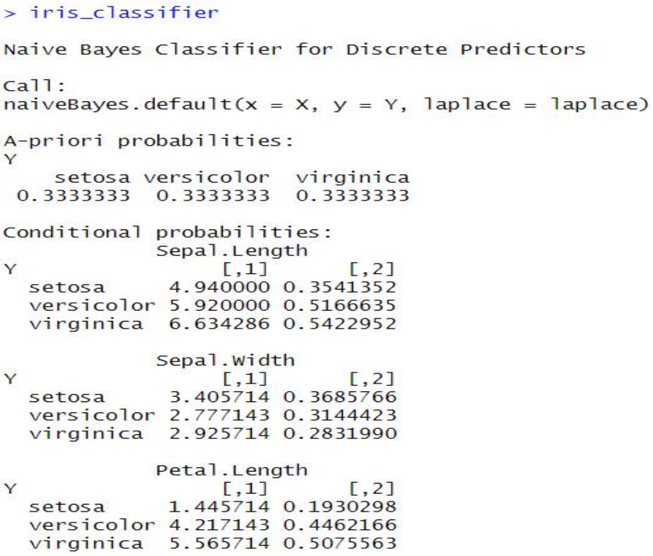
```
> dim(iris)
[1] 150   6
> table(iris$Species)

    setosa versicolor  virginica 
        50         50         50 
> set.seed(123)
> split = sample.split(iris$Species, SplitRatio = 0.7)
```



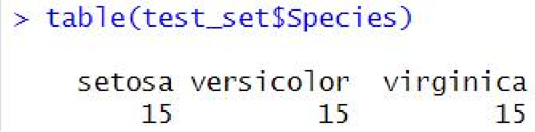
```
> training_set = subset(iris, split == TRUE)
> test_set = subset(iris, split == FALSE)
> training_set
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species Cluster
1            5.1         3.5          1.4         0.2     setosa       1
3            4.7         3.2          1.3         0.2     setosa       1
6            5.4         3.9          1.7         0.4     setosa       1
7            4.6         3.4          1.4         0.3     setosa       1
9            4.4         2.9          1.4         0.2     setosa       1
10           4.9         3.1          1.5         0.1     setosa       1
12           4.8         3.4          1.6         0.2     setosa       1
```



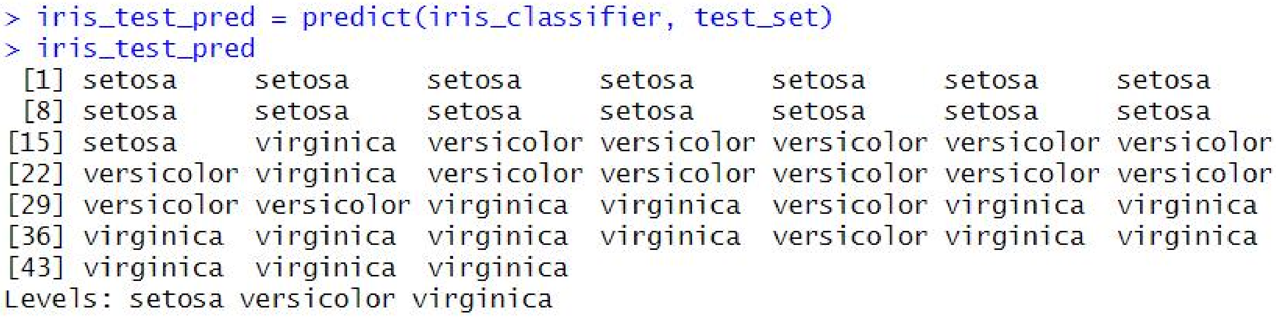
```
> test_set
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species Cluster
2            4.9         3.0          1.4         0.2     setosa       1
4            4.6         3.1          1.5         0.2     setosa       1
5            5.0         3.6          1.4         0.2     setosa       1
8            5.0         3.4          1.5         0.2     setosa       1
11           5.4         3.7          1.5         0.2     setosa       1
16           5.7         4.4          1.5         0.4     setosa       1
20           5.1         3.8          1.5         0.3     setosa       1
21           5.4         3.4          1.7         0.2     setosa       1
24           5.1         3.3          1.7         0.5     setosa       1
```



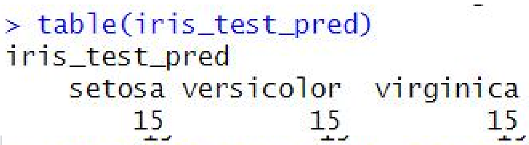
```
> iris_classifier

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     4.940000 0.3541352
  versicolor 5.920000 0.5166635
  virginica  6.634286 0.5422952

            Sepal.Width
Y                [,1]      [,2]
  setosa     3.405714 0.3685766
  versicolor 2.777143 0.3144423
  virginica  2.925714 0.2831990

            Petal.Length
Y                [,1]      [,2]
  setosa     1.445714 0.1930298
  versicolor 4.217143 0.4462166
  virginica  5.565714 0.5075563
```



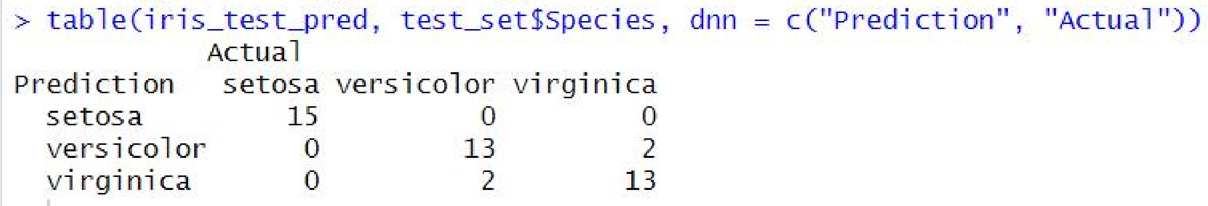
```
> table(test_set$Species)

    setosa versicolor  virginica 
        15         15         15 
```



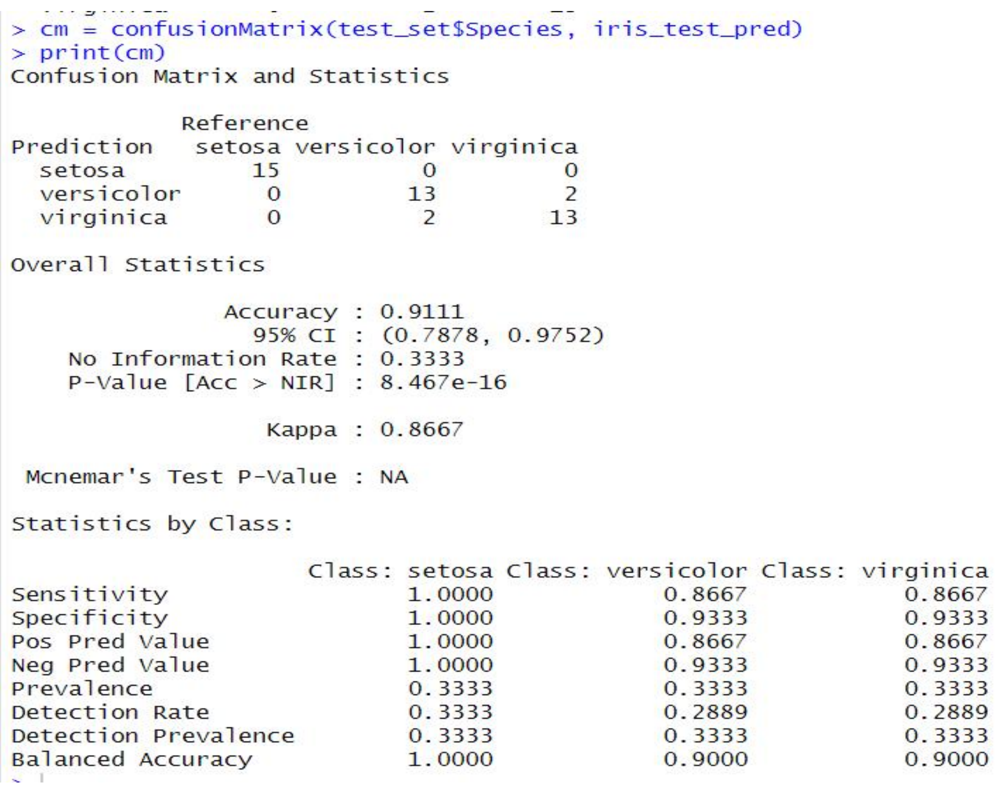
```
> iris_test_pred = predict(iris_classifier, test_set)
> iris_test_pred
 [1] setosa     setosa     setosa     setosa     setosa     setosa     setosa    
 [8] setosa     setosa     setosa     setosa     setosa     setosa     setosa    
[15] setosa     virginica  versicolor versicolor versicolor versicolor versicolor
[22] versicolor virginica  versicolor versicolor versicolor versicolor versicolor
[29] versicolor versicolor virginica  virginica  versicolor virginica  virginica 
[36] virginica  virginica  virginica  virginica  versicolor virginica  virginica 
[43] virginica  virginica  virginica 
Levels: setosa versicolor virginica
```



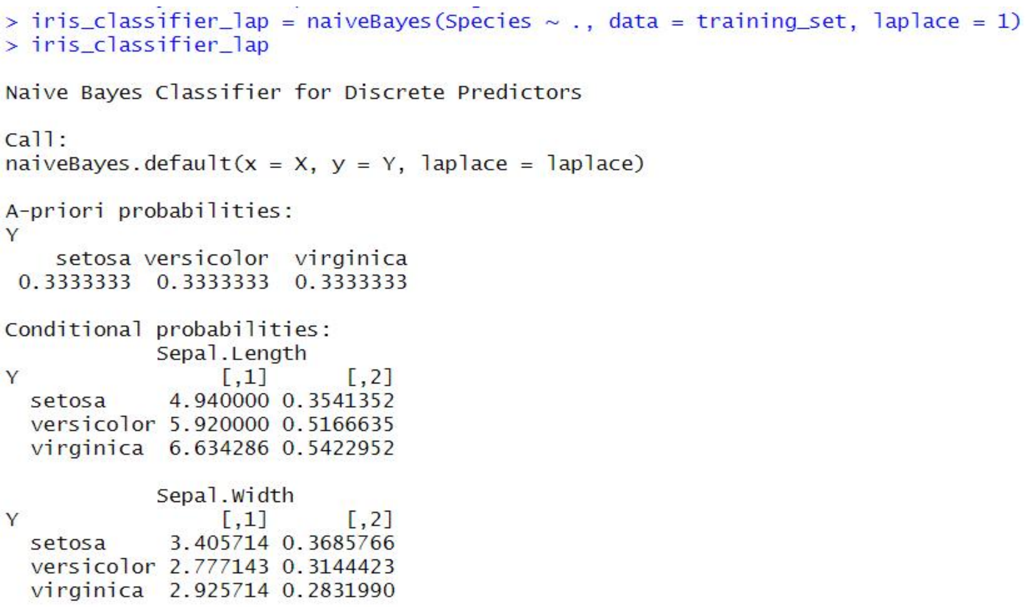
```
> table(iris_test_pred)
iris_test_pred
    setosa versicolor  virginica 
        15         15         15 
```



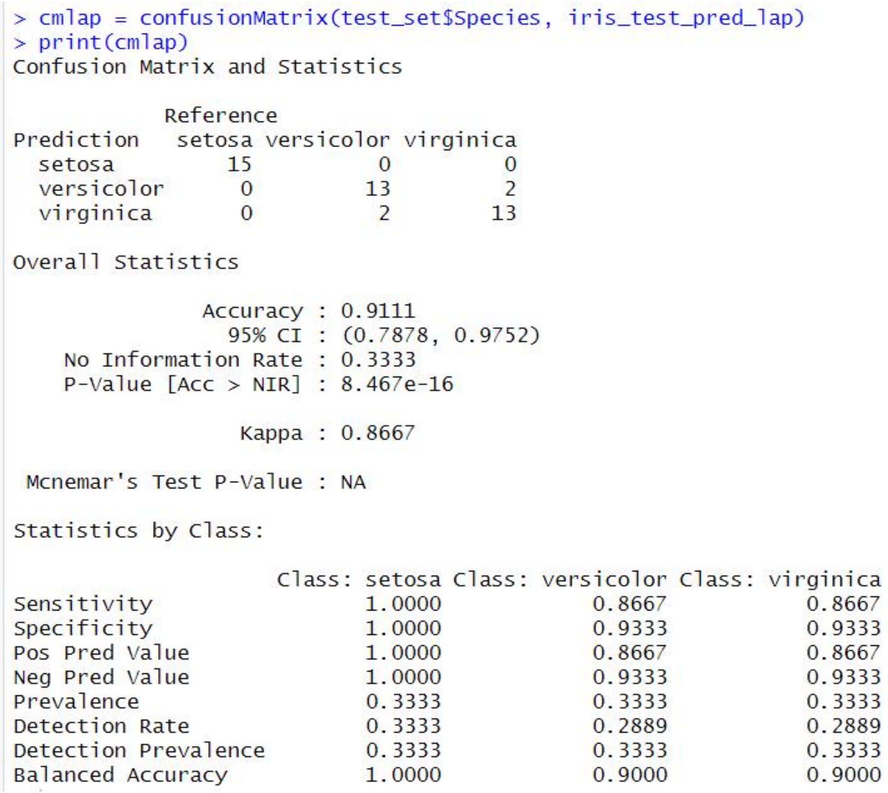
```
> table(iris_test_pred, test_set$Species, dnn = c("Prediction", "Actual"))
            Actual
Prediction   setosa versicolor virginica
  setosa         15          0         0
  versicolor      0         13         2
  virginica       0          2        13
```



```
> cm = confusionMatrix(test_set$Species, iris_test_pred)
> print(cm)
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         15          0         0
  versicolor      0         13         2
  virginica       0          2        13

Overall Statistics
                                          
               Accuracy : 0.9111          
                 95% CI : (0.7878, 0.9752)
    No Information Rate : 0.3333          
    P-Value [Acc > NIR] : 8.467e-16       
                                          
                  Kappa : 0.8667          
                                          
 Mcnemar's Test P-Value : NA              

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            0.8667           0.8667
Specificity                 1.0000            0.9333           0.9333
Pos Pred Value              1.0000            0.8667           0.8667
Neg Pred Value              1.0000            0.9333           0.9333
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.2889           0.2889
Detection Prevalence        0.3333            0.3333           0.3333
Balanced Accuracy           1.0000            0.9000           0.9000
```

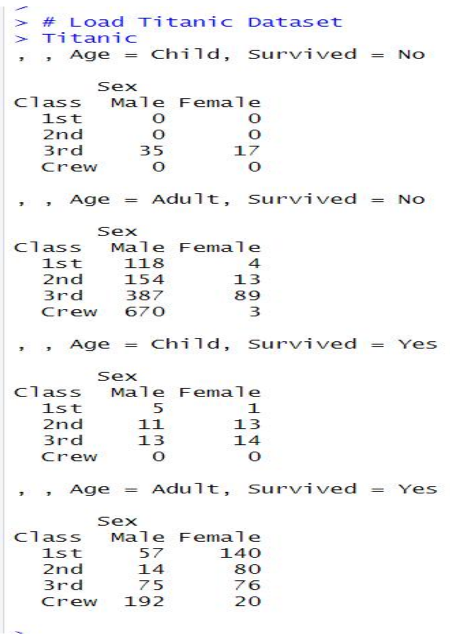


```
> iris_classifier_lap = naiveBayes(Species ~ ., data = training_set, laplace = 1)
> iris_classifier_lap

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     4.940000 0.3541352
  versicolor 5.920000 0.5166635
  virginica  6.634286 0.5422952

            Sepal.Width
Y                [,1]      [,2]
  setosa     3.405714 0.3685766
  versicolor 2.777143 0.3144423
  virginica  2.925714 0.2831990
```

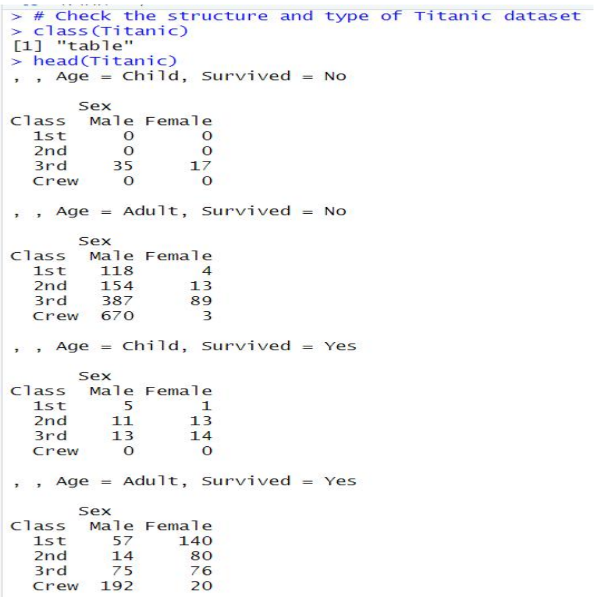


```
> cmlap = confusionMatrix(test_set$Species, iris_test_pred_lap)
> print(cmlap)
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         15          0         0
  versicolor      0         13         2
  virginica       0          2        13

Overall Statistics
                                          
               Accuracy : 0.9111          
                 95% CI : (0.7878, 0.9752)
    No Information Rate : 0.3333          
    P-Value [Acc > NIR] : 8.467e-16       
                                          
                  Kappa : 0.8667          
                                          
 Mcnemar's Test P-Value : NA              

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            0.8667           0.8667
Specificity                 1.0000            0.9333           0.9333
Pos Pred Value              1.0000            0.8667           0.8667
Neg Pred Value              1.0000            0.9333           0.9333
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.2889           0.2889
Detection Prevalence        0.3333            0.3333           0.3333
Balanced Accuracy           1.0000            0.9000           0.9000
```

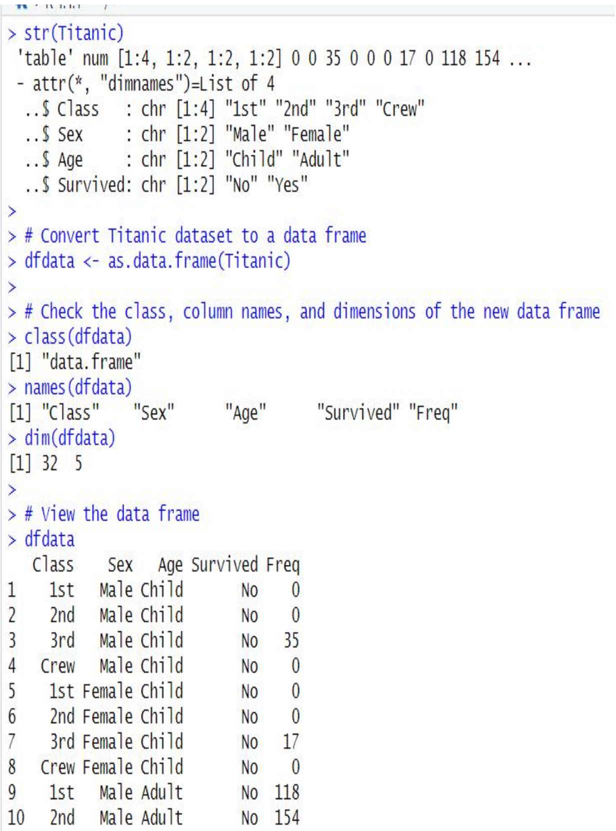
5.Write a Program to perform naïve bayes classification on Titanic dataset. Perform data pre-processing if required.



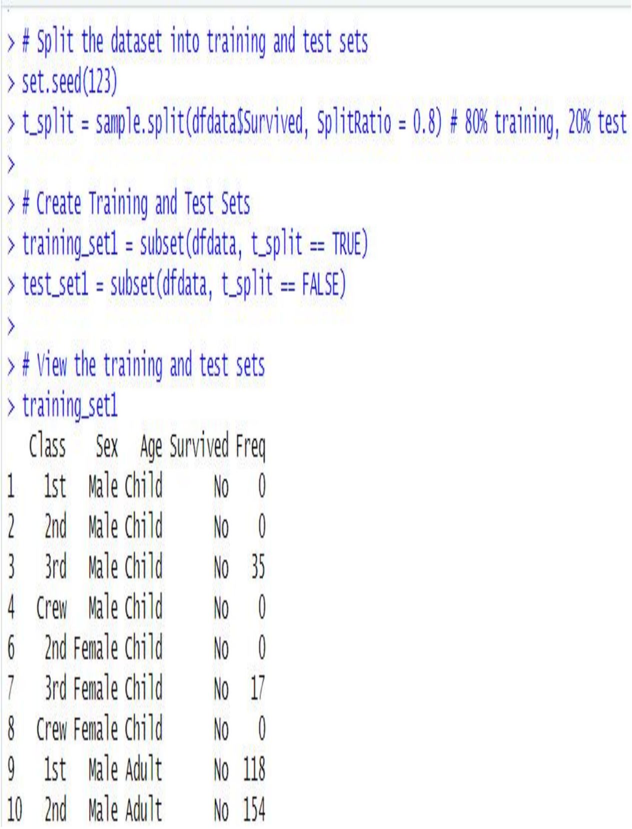
```
> # Load Titanic Dataset
> Titanic
, , Age = Child, Survived = No

      Sex
Class  Male Female
  1st     0      0
  2nd     0      0
  3rd    35     17
  Crew    0      0

, , Age = Adult, Survived = No

      Sex
Class  Male Female
  1st   118      4
  2nd   154     13
  3rd   387     89
  Crew  670      3

, , Age = Child, Survived = Yes

      Sex
Class  Male Female
  1st     5      1
  2nd    11     13
  3rd    13     14
  Crew    0      0

, , Age = Adult, Survived = Yes

      Sex
Class  Male Female
  1st    57    140
  2nd    14     80
  3rd    75     76
  Crew  192     20
```



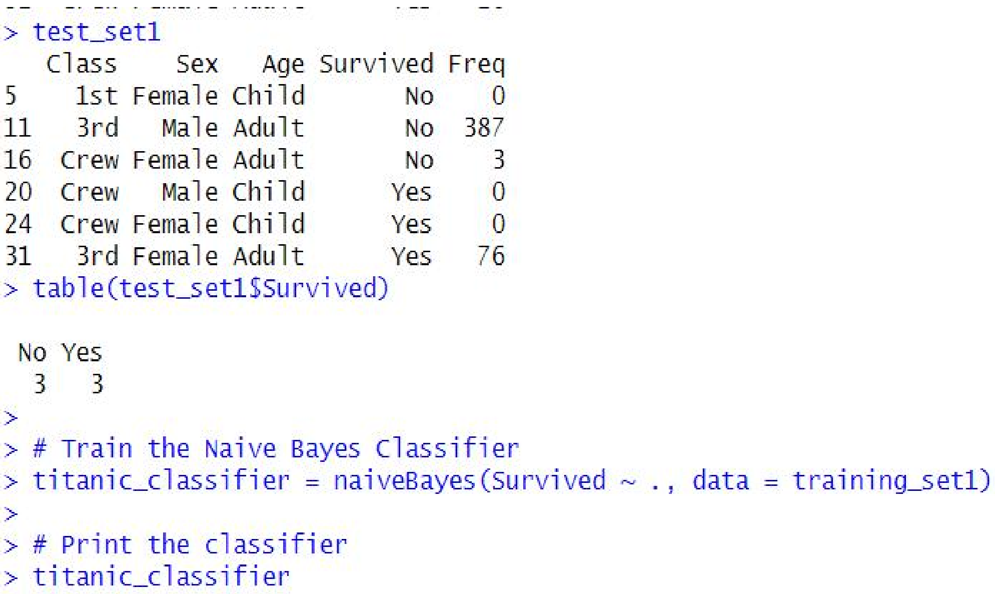
```
> # Check the structure and type of Titanic dataset
> class(Titanic)
[1] "table"
> head(Titanic)
, , Age = Child, Survived = No

      Sex
Class  Male Female
  1st     0      0
  2nd     0      0
  3rd    35     17
  Crew    0      0

, , Age = Adult, Survived = No

      Sex
Class  Male Female
  1st   118      4
  2nd   154     13
  3rd   387     89
  Crew  670      3

, , Age = Child, Survived = Yes

      Sex
Class  Male Female
  1st     5      1
  2nd    11     13
  3rd    13     14
  Crew    0      0

, , Age = Adult, Survived = Yes

      Sex
Class  Male Female
  1st    57    140
  2nd    14     80
  3rd    75     76
  Crew  192     20
```



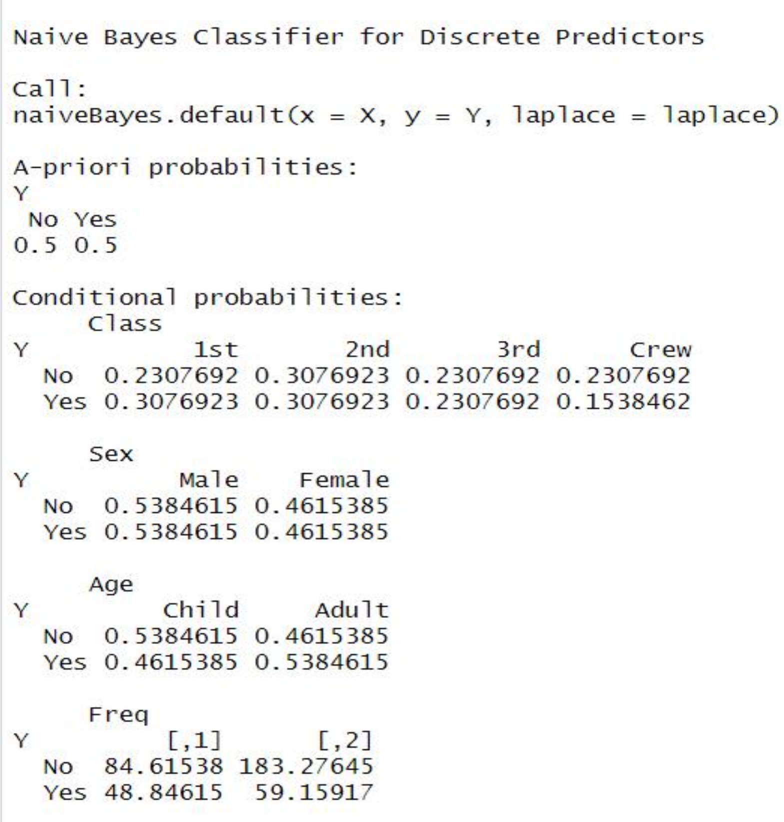
```
> str(Titanic)
 'table' num [1:4, 1:2, 1:2, 1:2] 0 0 35 0 0 0 17 0 118 154 ...
 - attr(*, "dimnames")=List of 4
  ..$ Class   : chr [1:4] "1st" "2nd" "3rd" "Crew"
  ..$ Sex     : chr [1:2] "Male" "Female"
  ..$ Age     : chr [1:2] "Child" "Adult"
  ..$ Survived: chr [1:2] "No" "Yes"
>
> # Convert Titanic dataset to a data frame
> dfdata <- as.data.frame(Titanic)
>
> # Check the class, column names, and dimensions of the new data frame
> class(dfdata)
[1] "data.frame"
> names(dfdata)
[1] "Class"    "Sex"      "Age"      "Survived" "Freq"
> dim(dfdata)
[1] 32  5
>
> # View the data frame
> dfdata
   Class    Sex   Age Survived Freq
1    1st   Male Child       No    0
2    2nd   Male Child       No    0
3    3rd   Male Child       No   35
4   Crew   Male Child       No    0
5    1st Female Child       No    0
6    2nd Female Child       No    0
7    3rd Female Child       No   17
8   Crew Female Child       No    0
9    1st   Male Adult       No  118
10   2nd   Male Adult       No  154
```



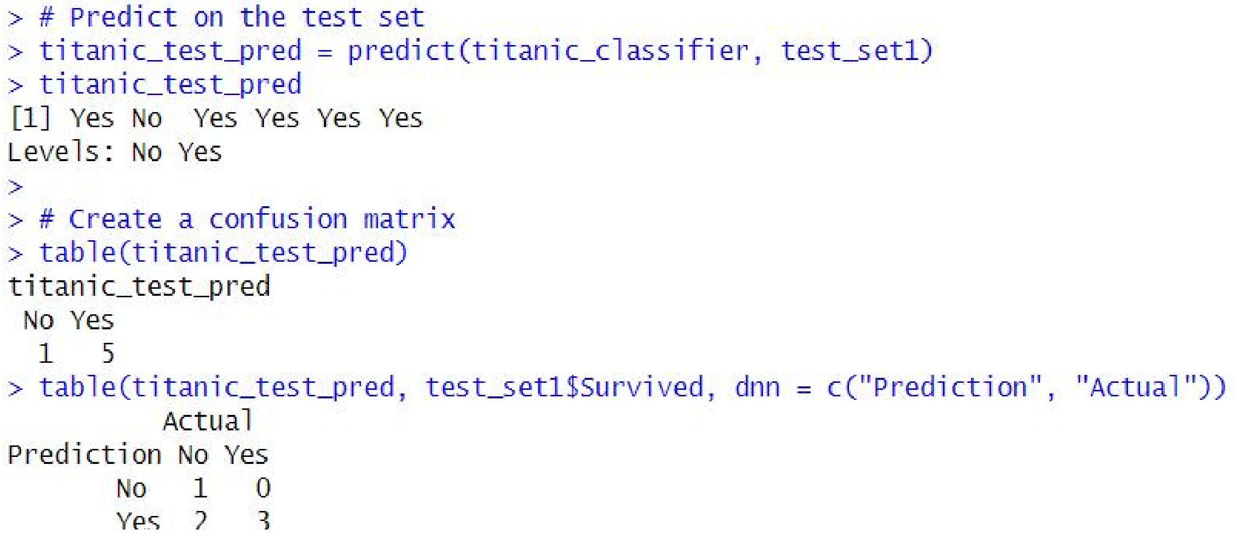
```
> # Split the dataset into training and test sets
> set.seed(123)
> t_split = sample.split(dfdata$Survived, SplitRatio = 0.8) # 80% training, 20% test
>
> # Create Training and Test Sets
> training_set1 = subset(dfdata, t_split == TRUE)
> test_set1 = subset(dfdata, t_split == FALSE)
>
> # View the training and test sets
> training_set1
   Class    Sex   Age Survived Freq
1    1st   Male Child       No    0
2    2nd   Male Child       No    0
3    3rd   Male Child       No   35
4   Crew   Male Child       No    0
6    2nd Female Child       No    0
7    3rd Female Child       No   17
8   Crew Female Child       No    0
9    1st   Male Adult       No  118
10   2nd   Male Adult       No  154
```



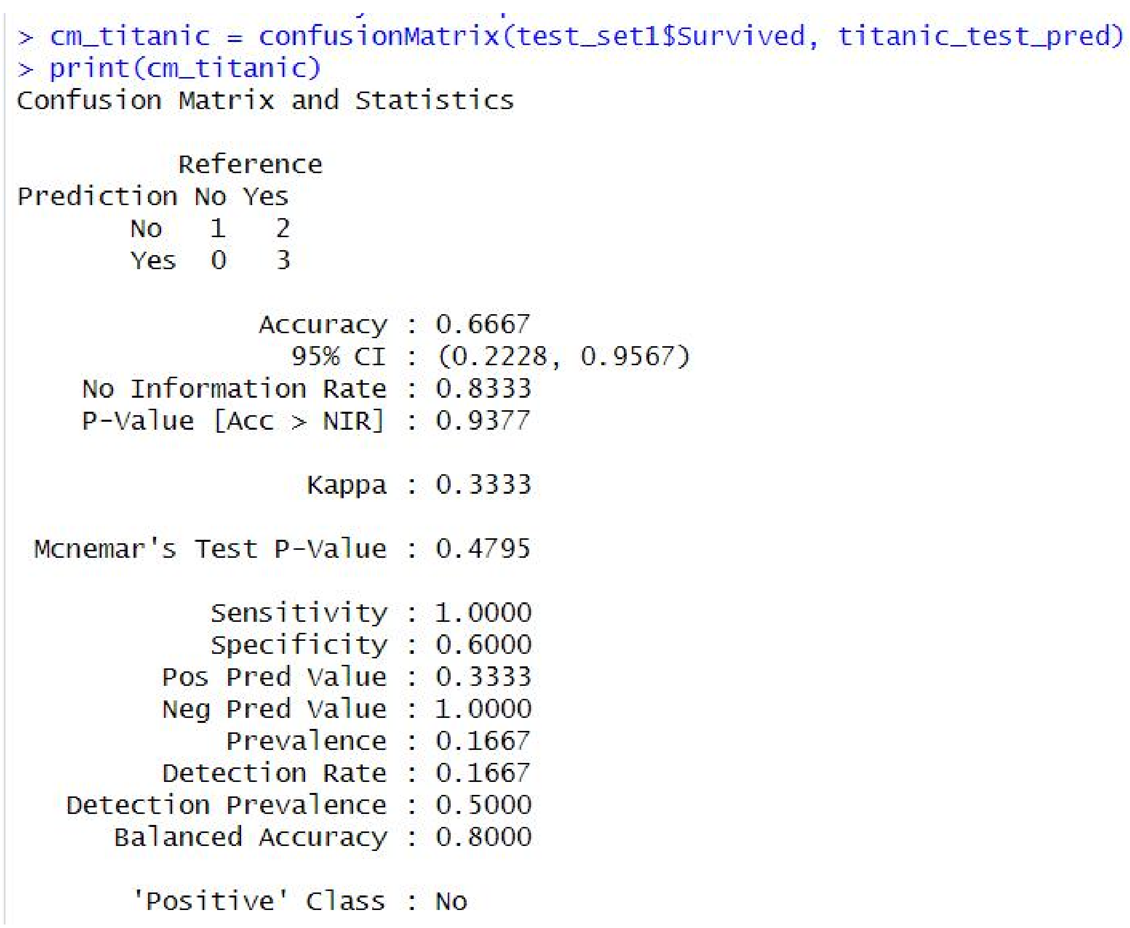
```
> test_set1
   Class    Sex   Age Survived Freq
5    1st Female Child       No    0
11   3rd   Male Adult       No  387
16  Crew Female Adult       No    3
20  Crew   Male Child      Yes    0
24  Crew Female Child      Yes    0
31   3rd Female Adult      Yes   76
> table(test_set1$Survived)

 No Yes 
  3   3 
> 
> # Train the Naive Bayes Classifier
> titanic_classifier = naiveBayes(Survived ~ ., data = training_set1)
> 
> # Print the classifier
> titanic_classifier
```



```
Naive Bayes Classifier for Discrete Predictors

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

A-priori probabilities:
Y
 No Yes 
0.5 0.5 

Conditional probabilities:
     Class
Y           1st       2nd       3rd      Crew
  No  0.2307692 0.3076923 0.2307692 0.2307692
  Yes 0.3076923 0.3076923 0.2307692 0.1538462

     Sex
Y          Male    Female
  No  0.5384615 0.4615385
  Yes 0.5384615 0.4615385

     Age
Y         Child     Adult
  No  0.5384615 0.4615385
  Yes 0.4615385 0.5384615

     Freq
Y         [,1]      [,2]
  No  84.61538 183.27645
  Yes 48.84615  59.15917
```



```
> # Predict on the test set
> titanic_test_pred = predict(titanic_classifier, test_set1)
> titanic_test_pred
[1] Yes No  Yes Yes Yes Yes
Levels: No Yes
>
> # Create a confusion matrix
> table(titanic_test_pred)
titanic_test_pred
 No Yes
  1   5
> table(titanic_test_pred, test_set1$Survived, dnn = c("Prediction", "Actual"))
          Actual
Prediction No Yes
       No   1   0
       Yes  2   3
```



```
> cm_titanic = confusionMatrix(test_set1$Survived, titanic_test_pred)
> print(cm_titanic)
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No   1   2
       Yes  0   3

               Accuracy : 0.6667
                 95% CI : (0.2228, 0.9567)
    No Information Rate : 0.8333
    P-Value [Acc > NIR] : 0.9377

                  Kappa : 0.3333

 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 1.0000
            Specificity : 0.6000
         Pos Pred Value : 0.3333
         Neg Pred Value : 1.0000
             Prevalence : 0.1667
         Detection Rate : 0.1667
   Detection Prevalence : 0.5000
      Balanced Accuracy : 0.8000

       'Positive' Class : No
```