**DR B.R. AMBEDKAR NATIONAL INSTITUTE OF**

**TECHNOLOGY, JALANDHAR**



**Data Mining and Data Warehousing Lab**

**CSX-425**

**Session: July-Dec 2020**

**SUBMITTED BY-**

Name – Ankit Goyal

Roll no - 17103011

Group - G-1

Branch - CSE

**SUBMITTED TO-**

Dr. Kunwar Pal

Assistant Professor

CSE Department

**Assignment – 1**

**1. For given A and B:**

1. **Find matrix – matrix multiplication (AB)**
2. **Find (AB)T and (AB)-1**
3. **Find the mean, standard deviation for each column and row for the matrices A, B, AB, (AB)-1.**

**Program:**

A <- rbind(c(3,-2,1),c(-1,4,-2))

B <- rbind(c(-7,4),c(9,5),c(2,-1))

print("Matrix A : ")

print(A)

print("Matrix B :")

print(B)

#AB

C <-A%\*%B

print("Multiplication AB :")

print(C)

#T(AB)

T <-t(C)

print("Transpose of Matrix AB :")

print(T)

#I(AB)

I <- solve(C)

print("Inverse of Matrix AB :")

print(I)

#Mean

print("Mean of Matrix A :")

#Row

mean(A[1,])

mean(A[2,])

#column

mean(A[,1])

mean(A[,2])

mean(A[,3])

print("Mean of Matrix B :")

#Row

mean(B[1,])

mean(B[2,])

mean(B[3,])

#column

mean(B[,1])

mean(B[,2])

print("Mean of Matrix AB :")

#Row

mean(C[1,])

mean(C[2,])

#column

mean(C[,1])

mean(C[,2])

print("Mean of Matrix Inverse of AB :")

#Row

mean(I[1,])

mean(I[2,])

#column

mean(I[,1])

mean(I[,2])

#Standard Deviations

print("Standard deviation of matrix A :")

sd(A,na.rm=TRUE)

print("Standard deviation of matrix B :")

sd(B,na.rm=TRUE)

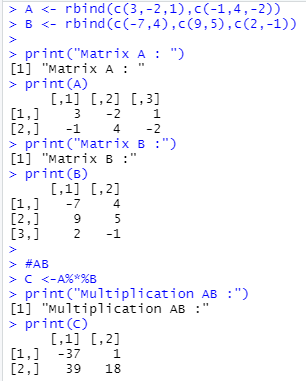
print("Standard deviation of matrix AB :")

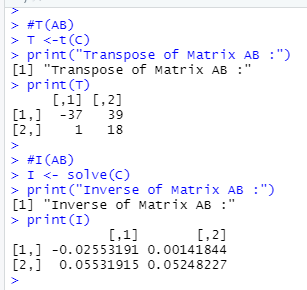
sd(C,na.rm=TRUE)

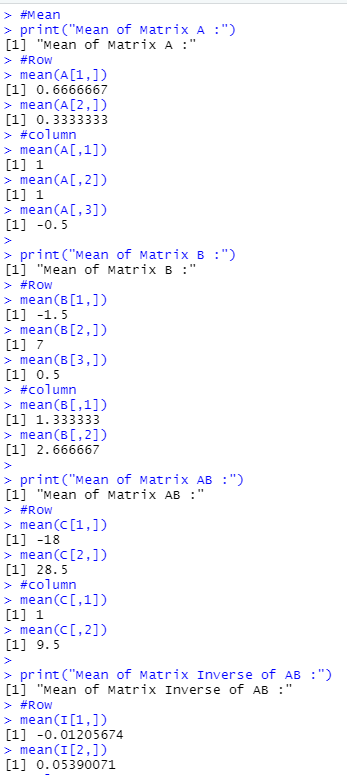
print("Standard deviation of matrix inverse of AB :")

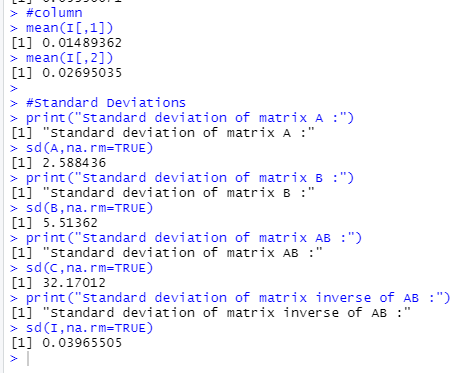
sd(I,na.rm=TRUE)

**Output:**









**2. Write a “Function” program in R to find n! . Hence Find 13! , 32! ,.Do not name the function by “Factorial”. You can initialize that 0!=1 and 1!=1.**

**Program:**

findfactorial <- function(n){

factorial <- 1

if((n==0||n==1))

factorial <- 1

else{

for(i in 1:n)

factorial <- factorial\*i

}

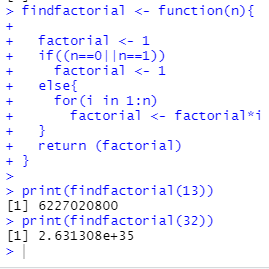
return (factorial)

}

print(findfactorial(13))

print(findfactorial(32))

**Output:**



**3. Write a “Function” program in R to find maximum and minimum from a set of numbers. Do not name the function by “max” or “min”. As an input you take (4,44.7,2,40,54,1,3,4).**

**Program:**

vector1 <- c(4,44.7,2,40,54,1,3,4)

l <- length(vector1)

min1 = 10000

max1 = -10000

for(i in 1:l){

if(min1>vector1[i]){

min1 = vector1[i];

}

if(max1<vector1[i]){

max1 = vector1[i];

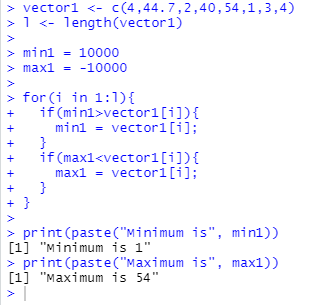
}

}

print(paste("Minimum is", min1))

print(paste("Maximum is", max1))

**Output :**



**Assignment - 2**

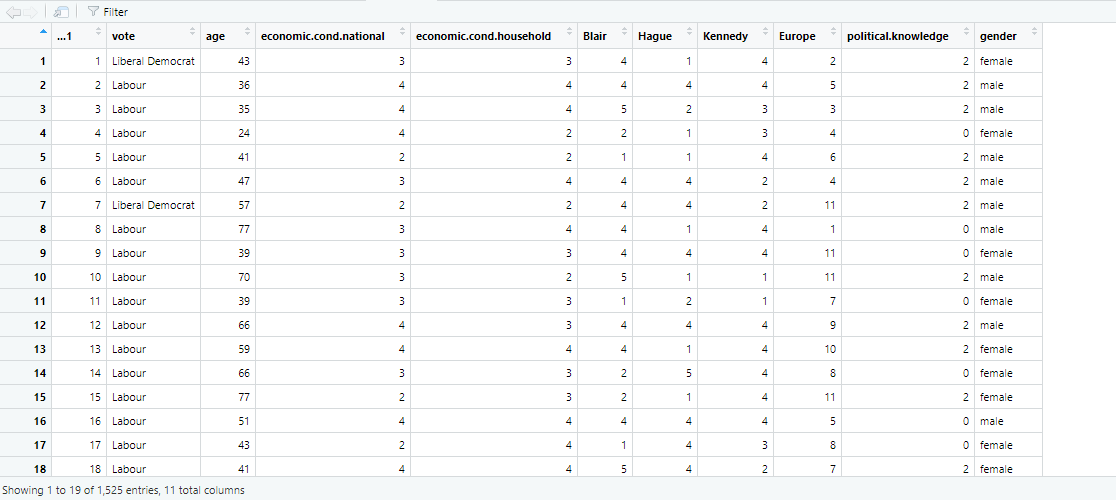
1. **How to read/write data from the dataset in R.**

**Program:**

library(readxl)

dataset=read\_excel(file.choose())

View(dataset)

**Output:**

**2. Use different function in R**

**a) Read**

**b) Head**

**c) Tail**

**d) Names**

**Program:**

data <- read\_excel("BEPSxls.xlsx")

#data-read

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

print(data)

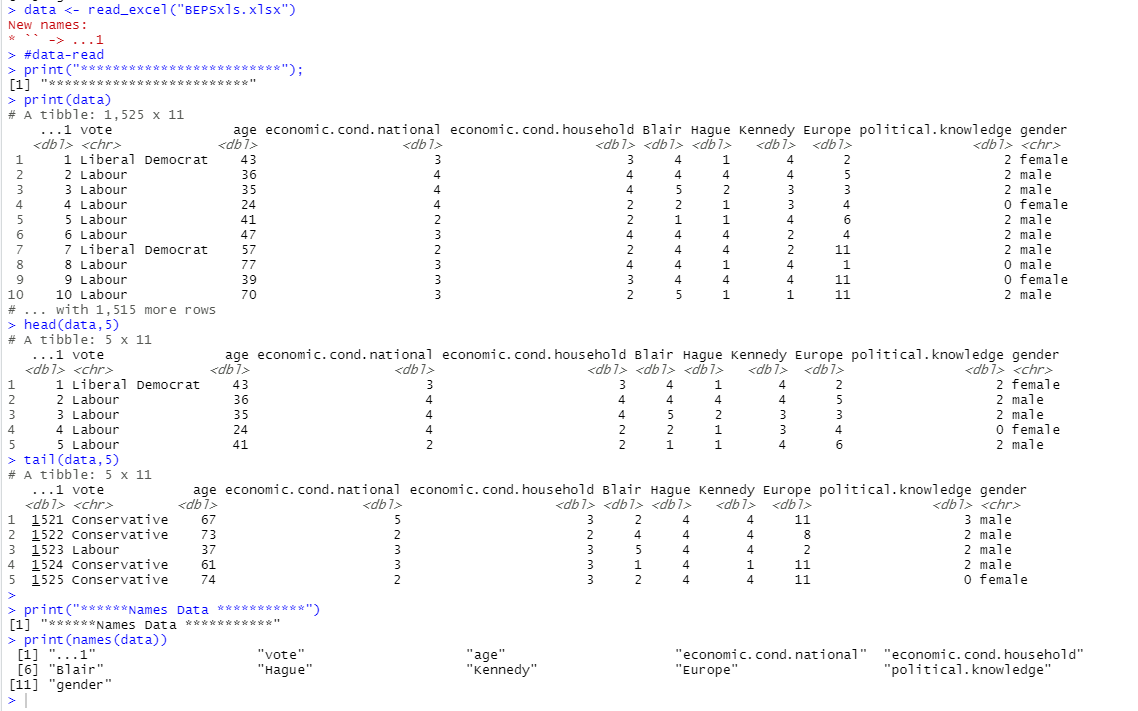
head(data,5)

tail(data,5)

print("\*\*\*\*\*\*Names Data \*\*\*\*\*\*\*\*\*\*\*")

print(names(data))

**Output:**



1. **Download the given dataset and perform the following.**
   1. **Mean**
   2. **Median**
   3. **Summary**
   4. **Histogram**
   5. **Plot**

**Program:**

dataset<- read\_excel("BEPSxls.xlsx")

mean(dataset$age)

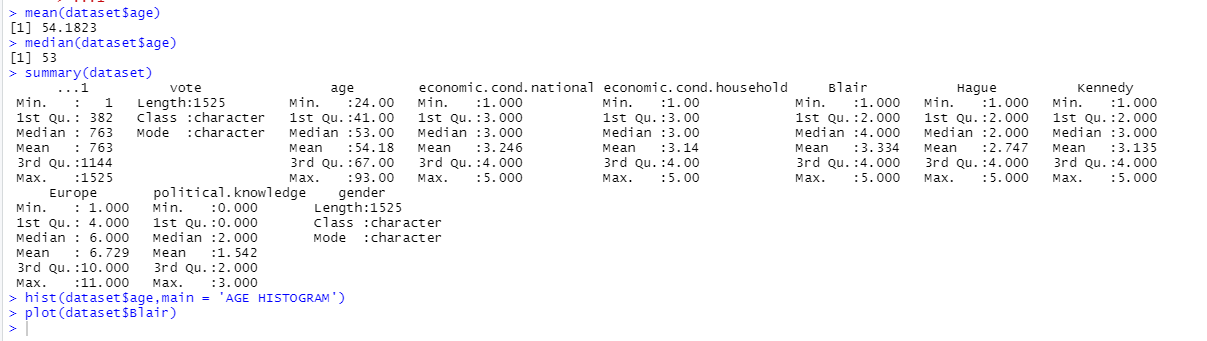
median(dataset$age)

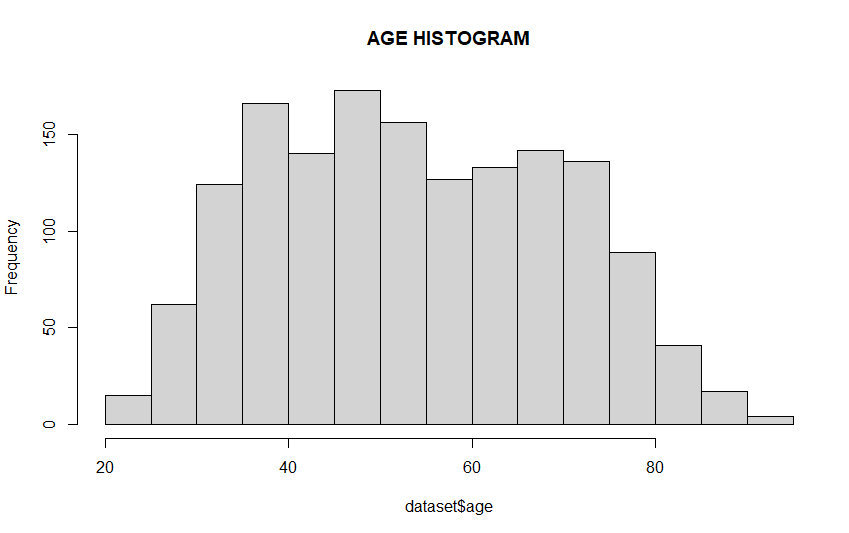
summary(dataset)

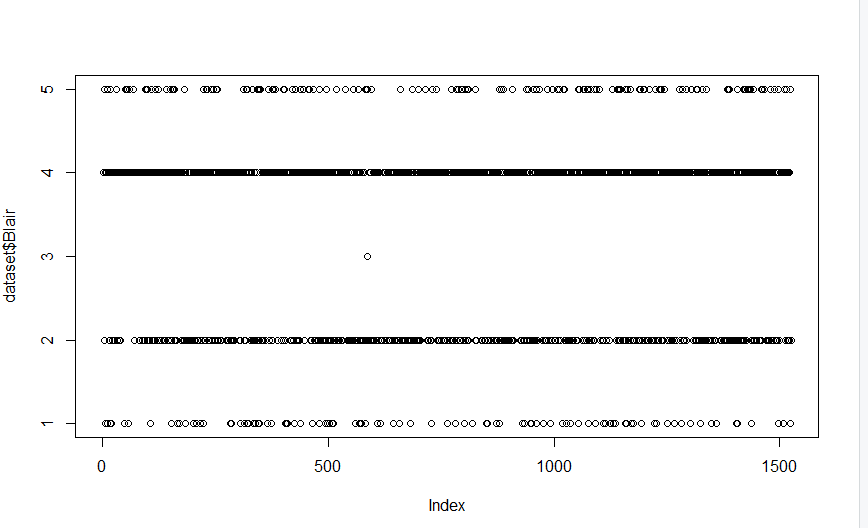
hist(dataset$age,main = 'AGE HISTOGRAM')

plot(dataset$Blair)

**Output:**







1. **Attach and Detach the dataset in R.**

**Program:**

data <- data.frame(x1 = c(9, 8, 3, 4, 8),

x2 = c(5, 4, 7, 1, 1),

x3 = c(1, 2, 3, 4, 5))

data

x1 #give error

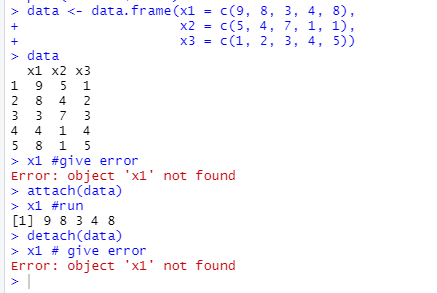
attach(data)

x1 #run

detach(data)

x1 # give error

**Output:**



**Assignment – 3**

1. **Demonstration of pre-processing on dataset mtcars. (R-studio)**

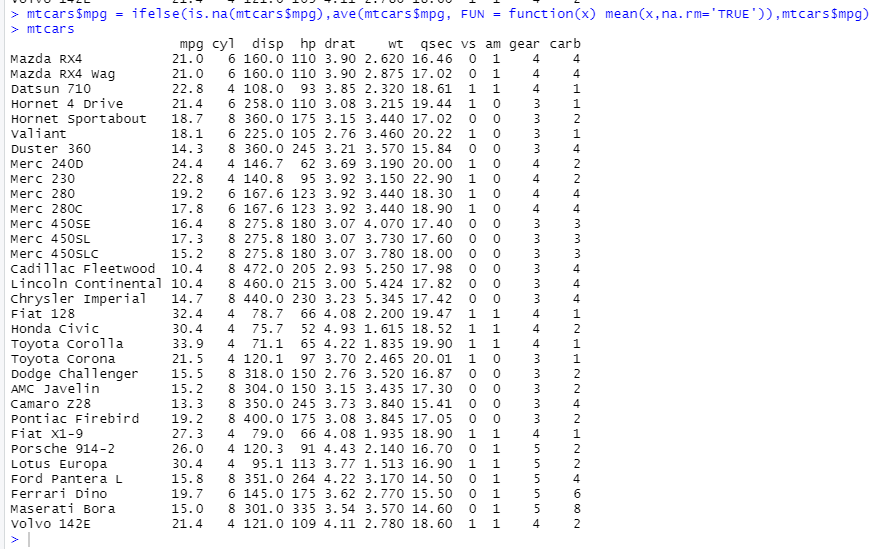
**Program:**

mtcars

mtcars$mpg = ifelse(is.na(mtcars$mpg),ave(mtcars$mpg, FUN = function(x) mean(x,na.rm='TRUE')),mtcars$mpg)

mtcars

**Output:**



1. **Demonstrate the filter function on dataset mtcars using (deplyr package)**
   1. **Show where gear attribute = 4,**
   2. **Show where disp = 160,**
   3. **Show different operations (and,or,not)**

**Program:**

library(dplyr)

#1 Show where gear attribute = 4,

gear\_4 <- filter(mtcars, gear == 4)

head(gear\_4)

#2 Show where disp = 160.

disp\_160 <- filter(mtcars, disp == 160.0)

head(disp\_160)

#3 Show different operations (and, or, not)

#AND

gear4\_and\_carb4 <- filter(mtcars, gear == 4 & carb == 4)

head(gear4\_and\_carb4)

#OR

gear4\_or\_hp110 <- filter(mtcars, gear == 4 | hp == 110)

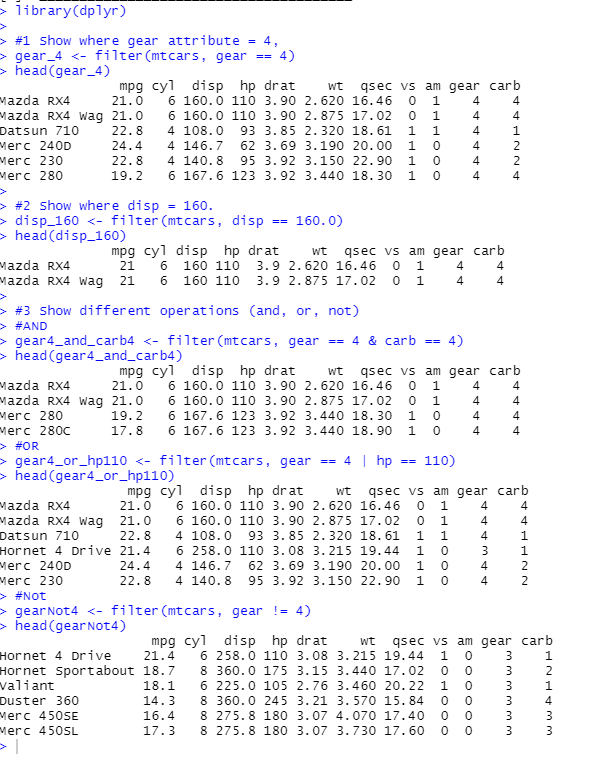
head(gear4\_or\_hp110)

#Not

gearNot4 <- filter(mtcars, gear != 4)

head(gearNot4)

**Output:**



1. **Demonstrate the different function on dataset mtcars/Titanic**
   1. **arrange**
   2. **group\_by**
   3. **summarise**
   4. **select**
   5. **intersect**
   6. **setdiff**

**Program:**

print("Arrange : ")

arrange(mtcars, desc(disp))

print("Group By : ")

group\_by(mtcars,drat)

print("Summarise : ")

summarise(mtcars,mean(disp))

print("Select : ")

select(mtcars,qsec)

print("Intersect :")

A<- subset(mtcars,disp==160)

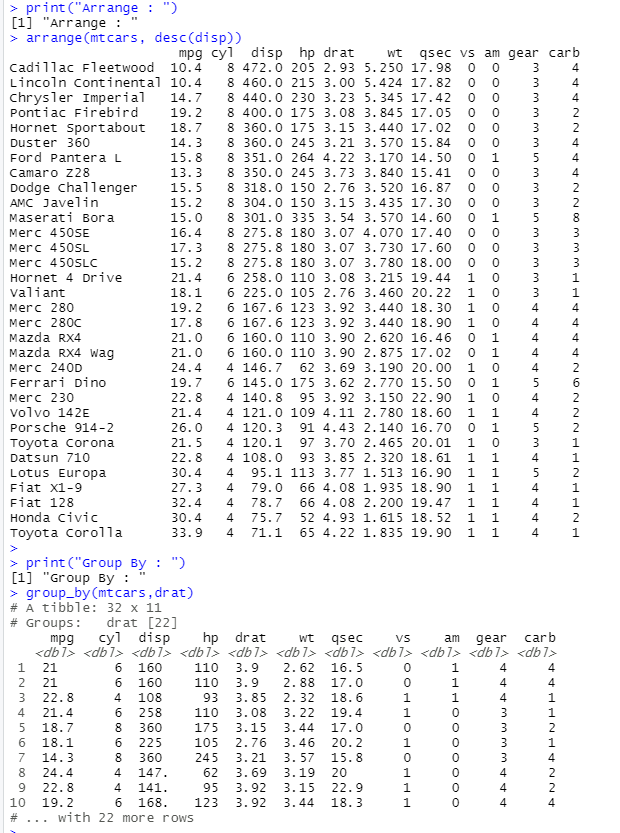
B<- subset(mtcars,cyl=100)

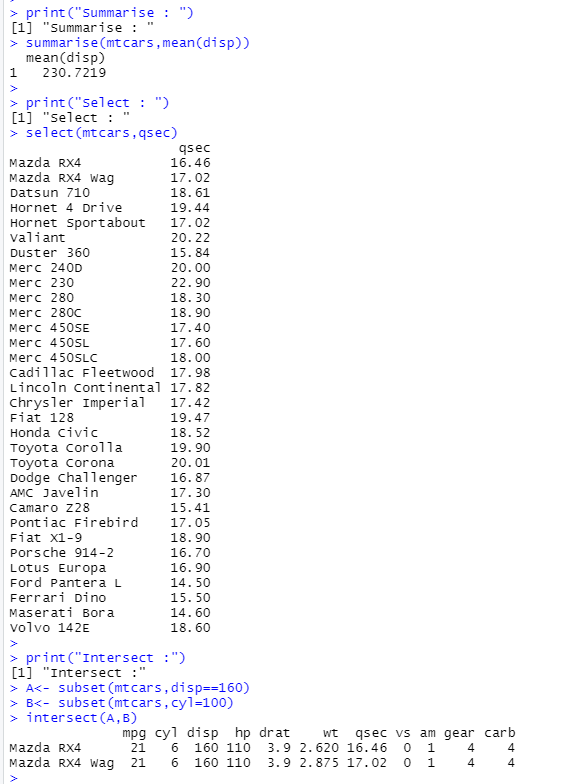
intersect(A,B)

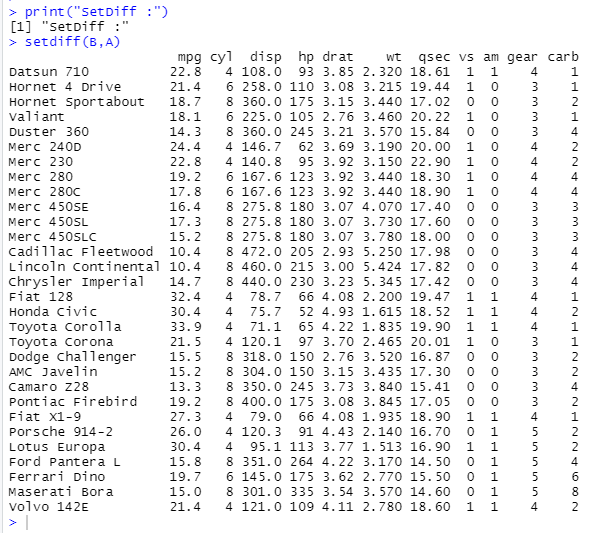
print("SetDiff :")

setdiff(B,A)

**Output:**







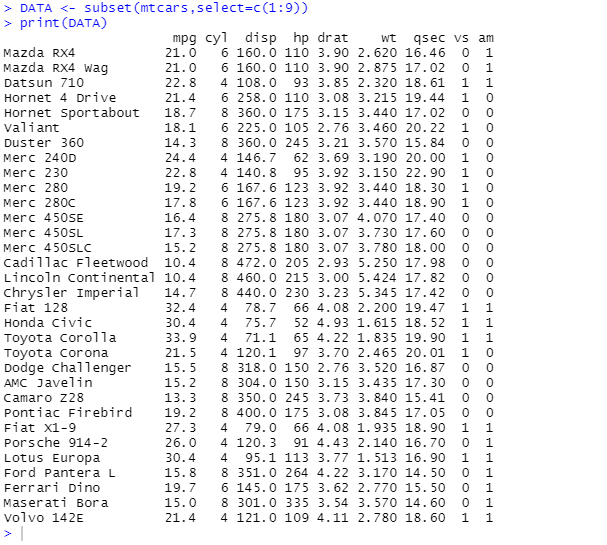
1. **Remove the not required columns from mtcars dataset.**

**Program:**

DATA <- subset(mtcars,select=c(1:9))

print(DATA)

**Output:**



1. **Show the attribute containing NA values in a column in dataset.**

**Program:**

myData <- data.frame(col1 = c(4,6,2, NA),

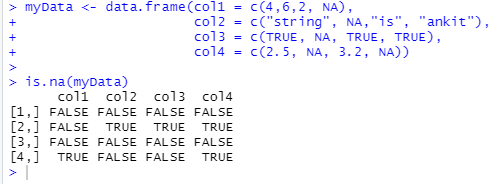
col2 = c("string", NA,"is", "ankit"),

col3 = c(TRUE, NA, TRUE, TRUE),

col4 = c(2.5, NA, 3.2, NA))

is.na(myData)

**Output:**

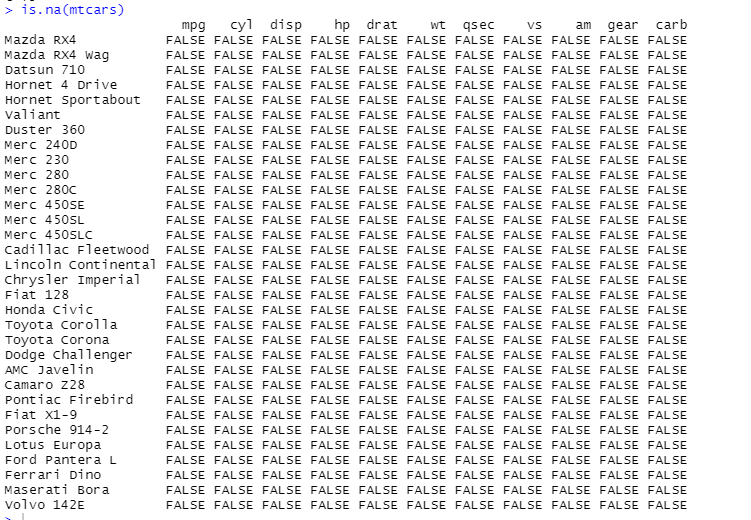


1. **Repeat the above question on downloaded dataset.**

**Program:**

Is.na(mtcars)

**Output:**



**Assignment- 4**

1. **As a crop researcher, you want to test effect of three different fertilizers mixtures on crop yield. You can use a one-way ANOVA to find out if there is a difference in crop yields between the three groups. Using the data, perform a one-way analysis of variance using alpha = .05.**
   1. **Perform a one-way analysis of variance**
   2. **Calculate test statistics**
   3. **Interpreting the results**
   4. **State conclusion**
   5. **Plot the graph for the same**

**Theory:**

The one-way analysis of variance (ANOVA), also known as one-factor ANOVA, is an extension of independent two-samples t-test for comparing means in a situation where there are more than two groups. In one-way ANOVA, the data is organized into several groups base on one single grouping variable (also called factor variable). The one-way analysis of variance (ANOVA) is used to determine whether there are any statistically significant differences between the means of three or more independent (unrelated) groups.To clarify if the data comes from the same population, you can perform a one-way analysis of variance (one-way ANOVA hereafter). This test, like any other statistical tests, gives evidence whether the H0 hypothesis can be accepted or rejected.

**Hypothesis in one-way ANOVA test**:

• H0: The means between groups are identical

• H3: At least, the mean of one group is different

In other words, the H0 hypothesis implies that there is not enough evidence to prove the mean of the group (factor) are different from another.

**Program:**

library(readxl)

my\_data=read\_excel(file.choose())

#my\_data <- read\_excel('DMDW\_LAB4.xlsx')

View(my\_data)

#check and display ordered levels

my\_data$group <- ordered(my\_data$group, levels = c("Group1", "Group2", "Group3"))

#compute summary statistics by group

library(dplyr)

group\_by(my\_data, group) %>%

summarise(

count = n(),mean = mean(values, na.rm = TRUE),

sd = sd(values, na.rm = TRUE)

)

#compute one way ANOVA

#compute analysis of variance

res.aov <- aov(values ~ group, data = my\_data)

#summary of analysis

summary(res.aov)

#interpret result of ANOVA

#multiple pairwise comparison

TukeyHSD(res.aov)

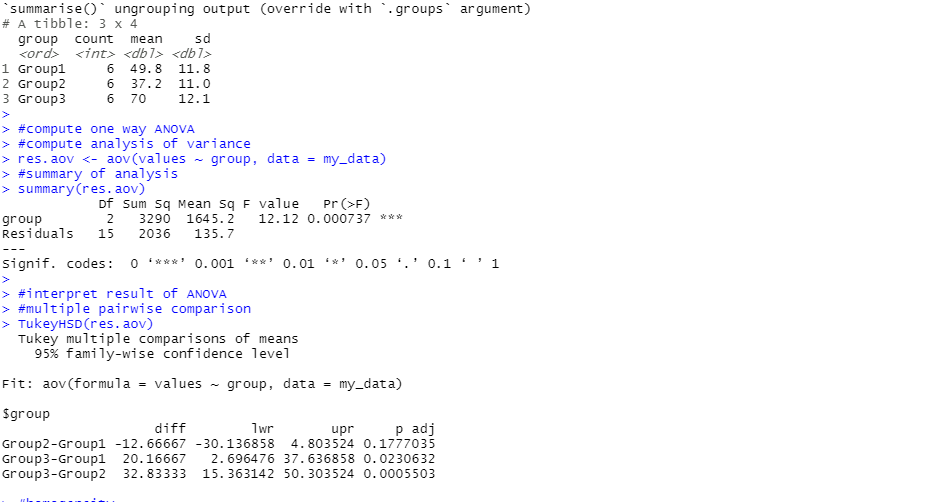
#homogeneity

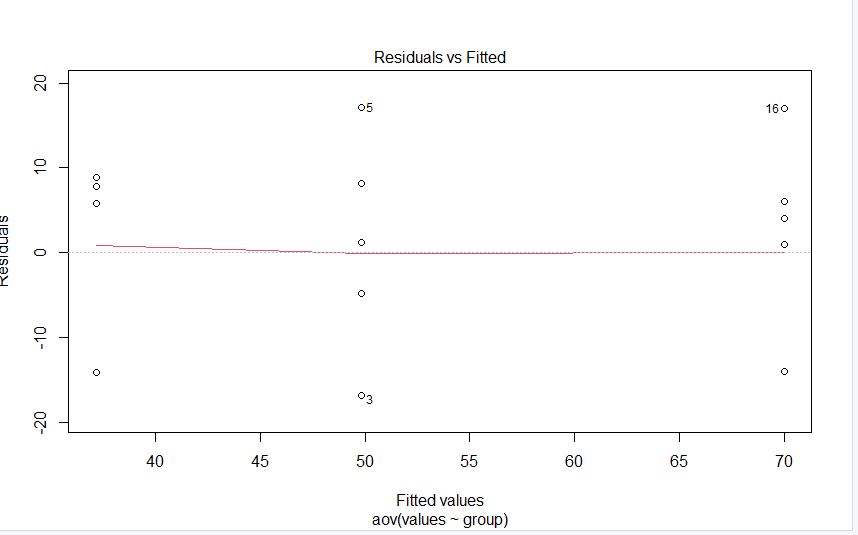
plot(res.aov,1)

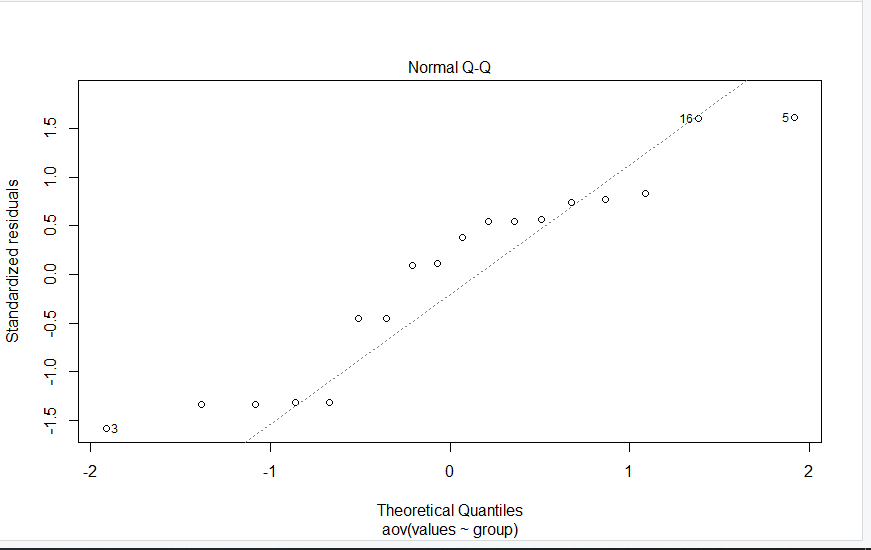
#normality

plot(res.aov,2)

**Output:**







1. **Repeat the quesion1 and perform one-way analysis of variance using inbuilt dataset in Rstudio.**

**Program:**

#build data

my\_data<- PlantGrowth

#check data and display ordered levels

sample\_n(my\_data,10)

#show levels

levels(my\_data$group)

#compute summary statistics

library(dplyr)

group\_by(my\_data, group) %>%

summarise(

count = n(),

mean = mean(weight, na.rm = TRUE),

sd = sd(weight, na.rm = TRUE)

)

#compute anova test

# Compute the analysis of variance

res.aov <- aov(weight ~ group, data = my\_data)

# Summary of the analysis

summary(res.aov)

#Interpret the result of one-way ANOVA tests

#multiple pairwise comparison

TukeyHSD(res.aov)

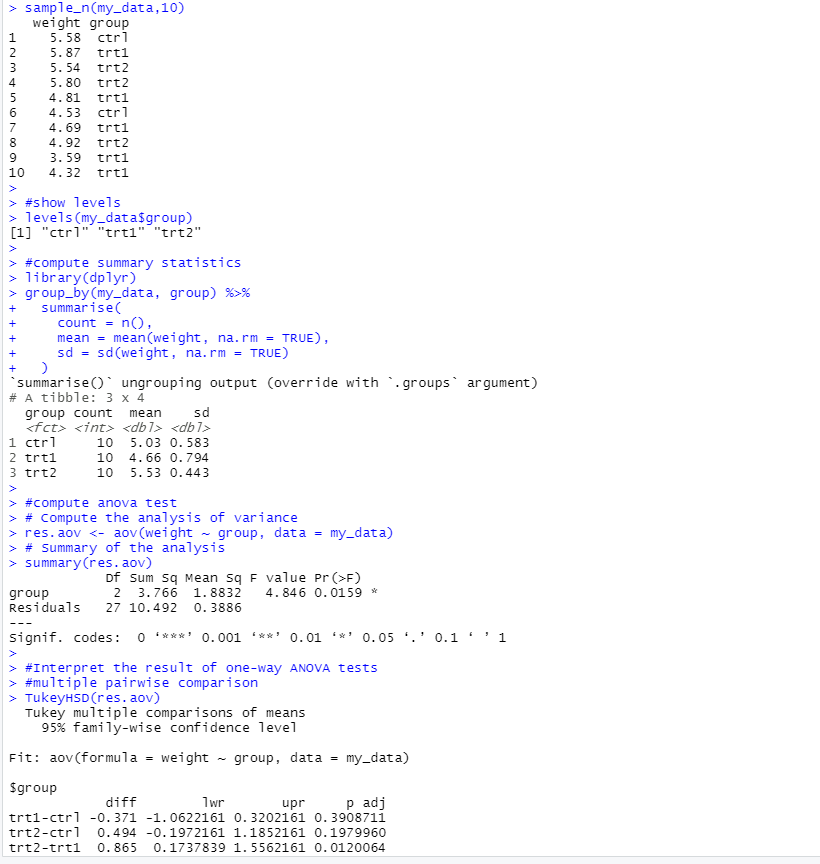
#Homogeneity of variances

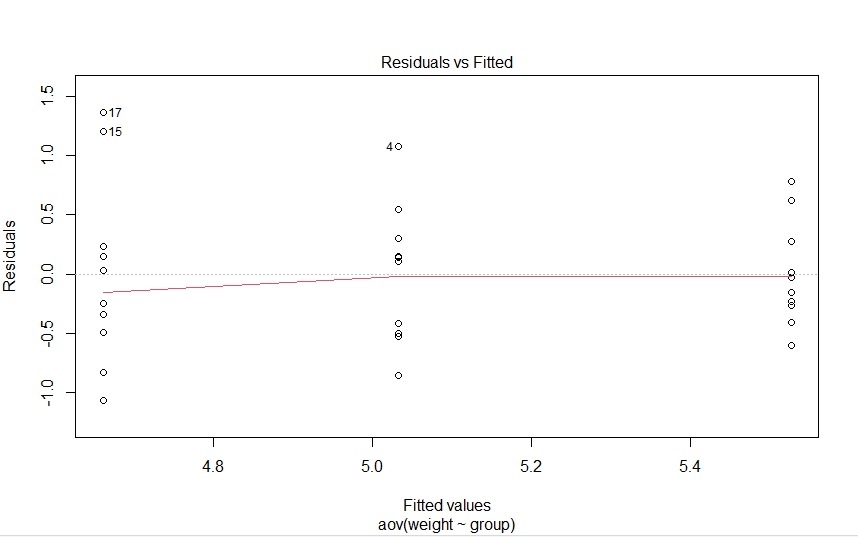
plot(res.aov, 1)

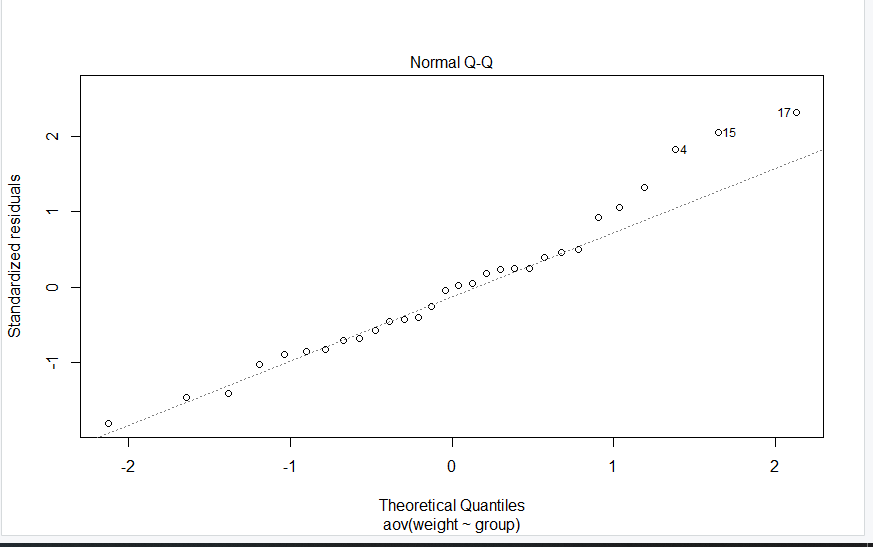
#Normality

plot(res.aov, 2)

**Output:**







**Assignment – 5**

1. **Consider dataset “Groceries” and apply apriori algorithm on it. What are the first 5 rules generated when the min support is 0.001 and min confidence is 0.9?**

**Program:**

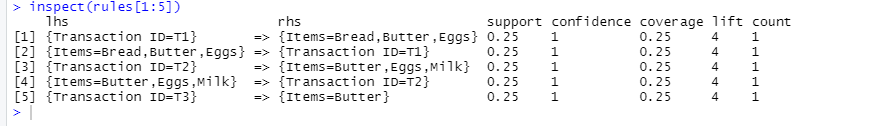
library(arules)

groceries <- read\_excel("LAB5.csv")

rules=apriori(data= groceries, parameter = list(support=0.001,confidence=0.9))

inspect(rules[1:5])

**Output:**



1. **The database has four transaction. What association rule can be found in this set, if the minimum support is 60% and minimum confidence is 80%.**

**Program:**

library(arules)

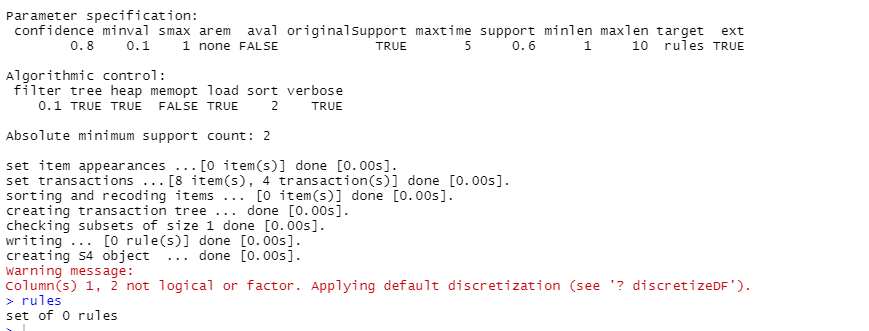
library(readr)

groceries2 <- read\_excel("LAB5-2.csv")

rules=apriori(data= groceries2,parameter= list(support=0.6,confidence=0.8))

rules

**Output:**



1. **Demonstration of association rule process on dataset titanic using apriori algorithm in rstudio.**

**Code:**

library(arules)

library(readr)

titanic <- read\_csv("titanic.csv")

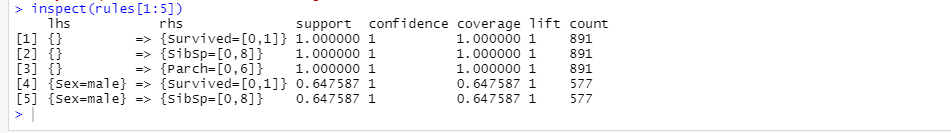
data(titanic)

rules=apriori(data= titanic, parameter = list(support=0.6,confidence=0.8))

rules

inspect(rules[1:5])

**OUTPUT:**



**Assignment - 6**

1. **Demonstrate performing linear regression on given data using R/Python.**
   1. **Plot the scattered graph**
   2. **Calculate test statistics**
   3. **Find coefficient and different performance matrix**

**Program:**

dataset <- read\_excel("LAB6.xlsx")

summary(dataset)

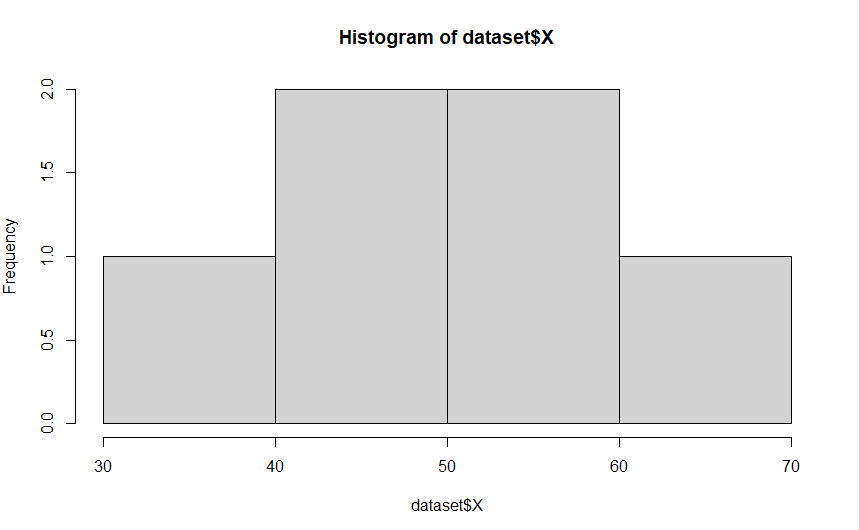
hist(dataset$X)

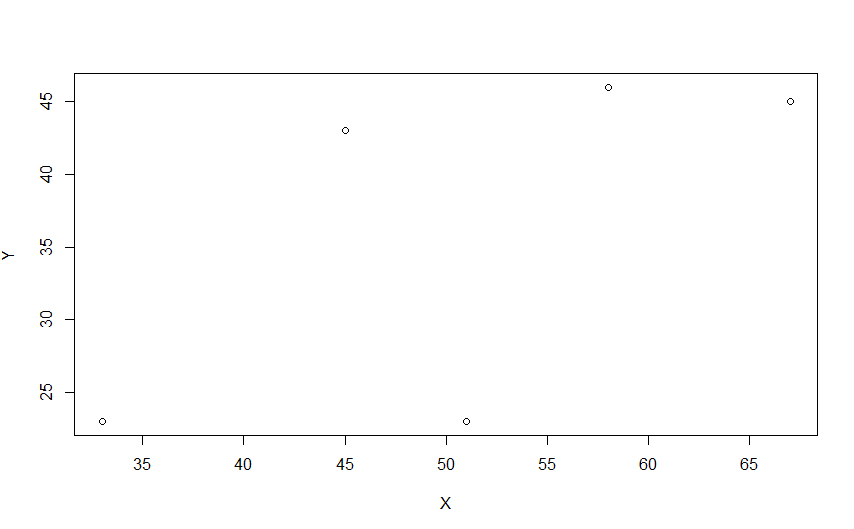
plot(Y~X, data=dataset)

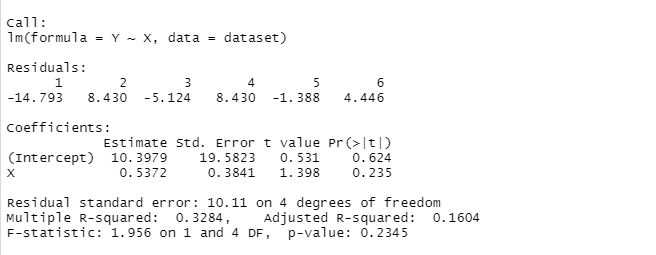
dataset.lm <- lm(Y~X, dataset)

summary(dataset.lm)

**Output:**







1. **Demonstrate performing linear regression on Lung capacity dataset using R/Python.**

**Program:**

dataset <- read\_excel("Lung Capacity.xls")

summary(dataset)

cor(dataset$Height, dataset$LungCapacity)

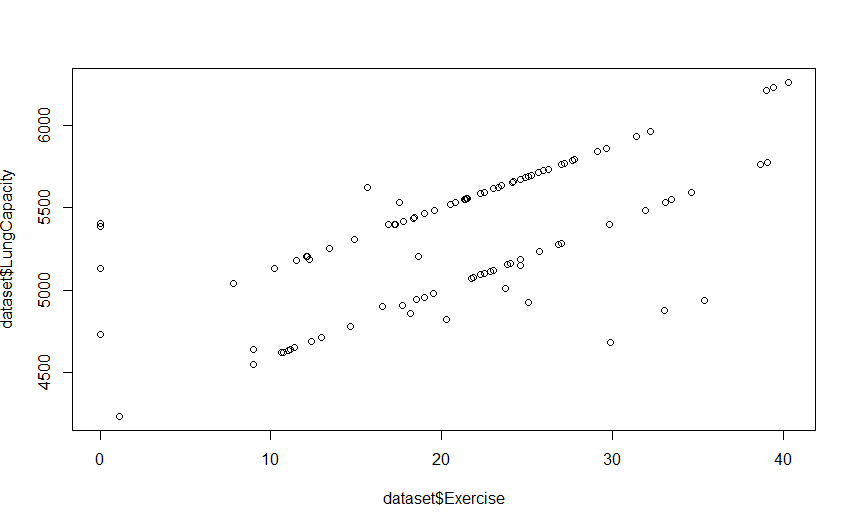
cor(dataset$Age, dataset$LungCapacity)

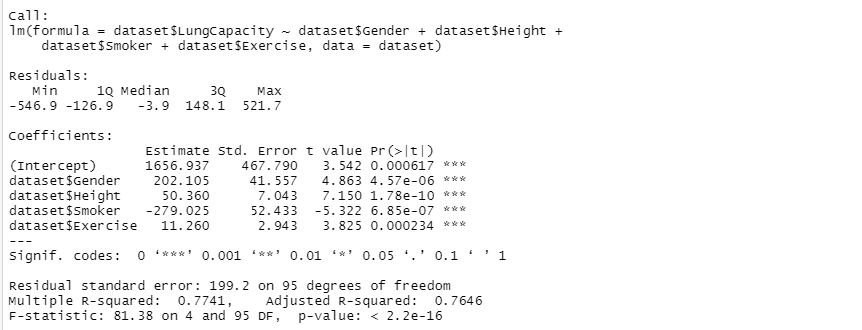
plot(dataset$Exercise, dataset$LungCapacity,data = dataset)

dataset.lm <- lm(dataset$LungCapacity ~ dataset$Gender + dataset$Height + dataset$Smoker +dataset$Exercise, data= dataset)

summary(dataset.lm)

**Output:**





**Assignment - 7**

1. **To construct Decision tree for weather data and classify it.**

**Program:**

library(rpart.plot)

library(rpart)

dataset <- read\_csv("austin\_weather.csv")

head(dataset)

shuffle\_index<-sample(1:nrow(dataset))

dataset <- dataset[shuffle\_index,]

ls(dataset)

sum(is.na(dataset$Events))

dim(dataset)

sum(is.na(dataset$DewPointAvgF))

summary(dataset$TempHighF)

dataset = subset(dataset, select = -c(Date,Events,TempAvgF, DewPointAvgF, HumidityAvgPercent,SeaLevelPressureAvgInches, VisibilityAvgMiles, WindAvgMPH ))

str(dataset)

dataset[] <- lapply(dataset, as.numeric)

dataset <- dataset %>%

mutate(TempHighF = case\_when(

TempHighF < 40 ~ "<40",

TempHighF >= 40 & TempHighF < 60 ~ "40-60",

TempHighF >= 60 & TempHighF < 80 ~ "60-80",

TempHighF >= 80 & TempHighF < 100 ~ "80-100",

TempHighF >= 100 ~ ">100",

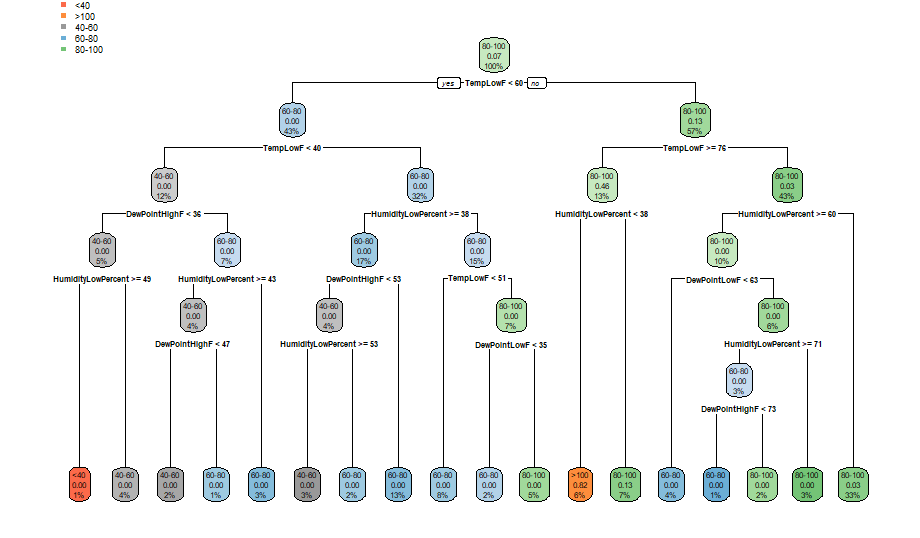
TRUE ~ "NA"

))

fit <- rpart(TempHighF~., data = dataset, method = 'class')

rpart.plot(fit, extra = 106)

**Output:**



1. **To construct Decision tree for customer data and classify it.**

**Program:**

dataset <- read\_csv("WA\_Fn-UseC\_-Telco-Customer-Churn.csv")

dim(dataset)

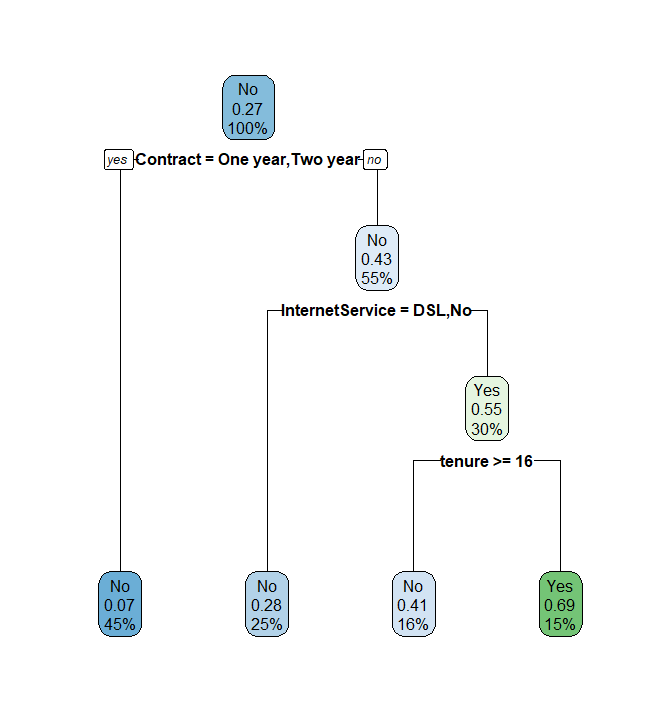
ls(dataset)

dataset = subset(dataset, select = -c(customerID ))

fit <- rpart(Churn~., data = dataset, method = 'class')

rpart.plot(fit, extra = 106)

**Output:**



**Assignment - 8**

1. **Write a procedure for clustering customer data using Simple KMeans Algorithm**

* Step 1: Choose groups in the feature plan randomly.
* Step 2: Minimize the distance between the cluster center and the different observations (**centroid**). It results in groups with observations.
* Step 3: Shift the initial centroid to the mean of the coordinates within a group.
* Step 4: Minimize the distance according to the new centroids. New boundaries are created. Thus, observations will move from one group to another.
* Repeat until no observation changes groups.

1. **Demonstration of clustering rule process on dataset using simple k-means.**

**Program:**

library(readr)

dataset = read\_csv("Mall\_Customers.csv")

dataset = dataset[4:5]

set.seed(6)

wcss = vector()

for (i in 1:10) wcss[i] = sum(kmeans(dataset, i)$withinss)

plot(1:10,

wcss,

type = 'b',

main = paste('The Elbow Method'),

xlab = 'Number of clusters',

ylab = 'WCSS')

kmeans = kmeans(x = dataset, centers = 5)

y\_kmeans = kmeans$cluster

# Visualising the clusters

library(cluster)

clusplot(dataset,

y\_kmeans,

lines = 0,

shade = TRUE,

color = TRUE,

labels = 2,

plotchar = FALSE,

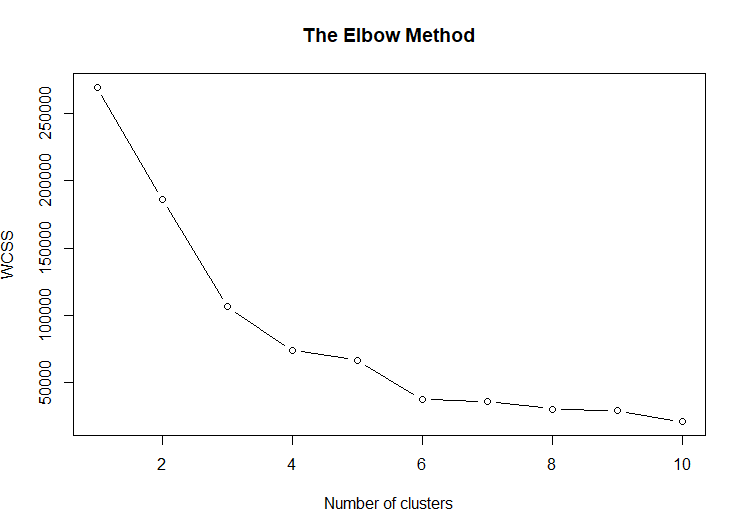
span = TRUE,

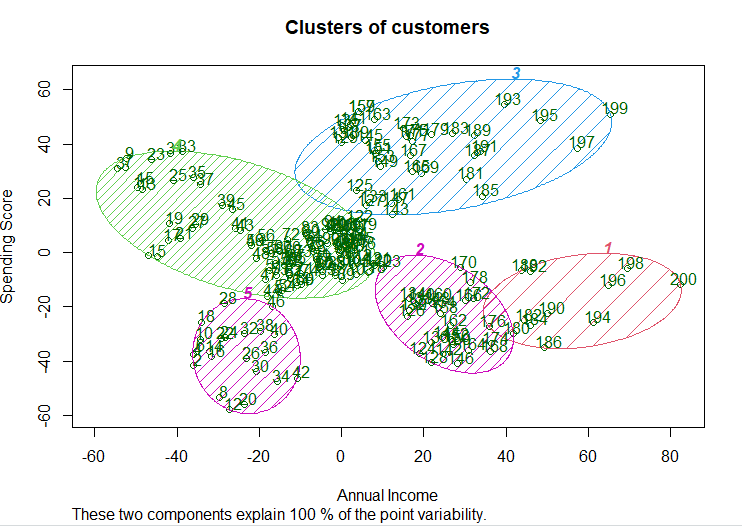
main = paste('Clusters of customers'),

xlab = 'Annual Income',

ylab = 'Spending Score')

**Output:**





**Assignment - 9**

1. **Demonstration of classification rule process on dataset using naïve bayes algorithm.**

**Program:**

# Installing Packages

install.packages("e1071")

install.packages("caTools")

install.packages("caret")

# Loading package

library(e1071)

library(caTools)

library(caret)

library(dplyr)

dataset = read\_csv("Mall\_Customers.csv")

dataset$Gender <- factor(dataset$Gender, levels = c("Male", "Female"), labels = c(0,1))

dataset <- dataset %>%

mutate(Age = case\_when(

Age < 30 ~ "<30",

Age >= 30 & Age < 45 ~ "30-45",

Age >= 45 & Age < 60 ~ "45-60",

Age >= 60 ~ ">60",

TRUE ~ "NA"

))

dataset <- dataset %>%

mutate(Income = case\_when(

Income <40 ~ "<40",

Income >= 40 & Income < 60 ~ "40-60",

Income >= 60 ~ ">60",

TRUE ~ "NA"

))

dataset <- dataset %>%

mutate(Score = case\_when(

Score < 30 ~ "<20",

Score >= 20 & Score < 40 ~ "20-40",

Score >= 40 & Score < 60 ~ "40-60",

Score >= 60 & Score < 80 ~ "60-80",

Score >= 80 ~ ">80",

TRUE ~ "NA"

))

trainIndex <- createDataPartition(dataset$Score, p = .7,

list = FALSE,

times = 1)

Train <- dataset[ trainIndex,]

Valid <- dataset[-trainIndex,]

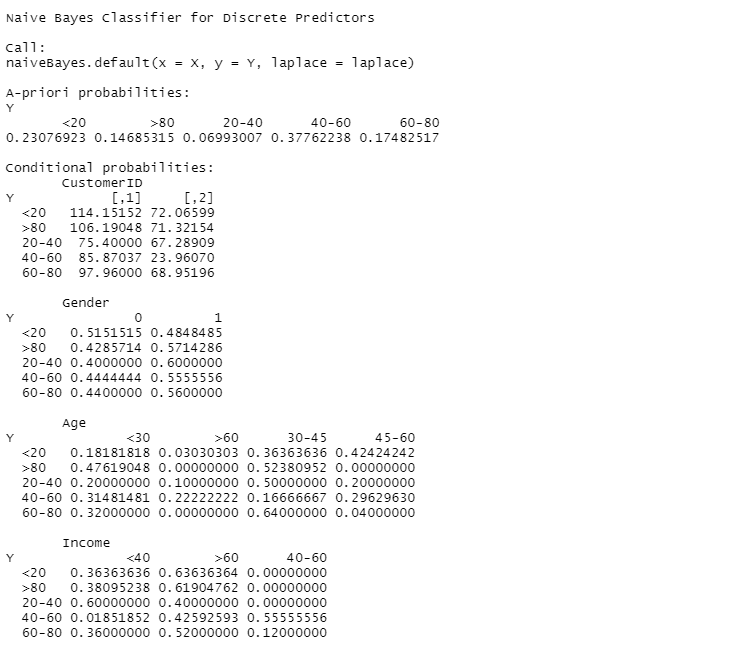
# Fitting Naive Bayes Model

# to training dataset

classifier\_cl <- naiveBayes(Score ~ ., data = Train)

classifier\_cl

**Output:**



1. **Demonstration of clustering rule process on dataset using EM algorithm.**

**Program:**

install.packages("mixtools")

dataset = read\_csv("Mall\_Customers.csv")

summary(dataset$Score)

x <- dataset$Score

plot(density(x))

mem <- kmeans(x,2)$cluster

mu1 <- mean(x[mem==1])

mu2 <- mean(x[mem==2])

sigma1 <- sd(x[mem==1])

sigma2 <- sd(x[mem==2])

pi1 <- sum(mem==1)/length(mem)

pi2 <- sum(mem==2)/length(mem)

# modified sum only considers finite values

sum.finite <- function(x) {

sum(x[is.finite(x)])

}

Q <- 0

# starting value of expected value of the log likelihood

Q[2] <- sum.finite(log(pi1)+log(dnorm(x, mu1, sigma1))) + sum.finite(log(pi2)+log(dnorm(x, mu2, sigma2)))

k <- 2

while (abs(Q[k]-Q[k-1])>=1e-6) {

# E step

comp1 <- pi1 \* dnorm(x, mu1, sigma1)

comp2 <- pi2 \* dnorm(x, mu2, sigma2)

comp.sum <- comp1 + comp2

p1 <- comp1/comp.sum

p2 <- comp2/comp.sum

# M step

pi1 <- sum.finite(p1) / length(x)

pi2 <- sum.finite(p2) / length(x)

mu1 <- sum.finite(p1 \* x) / sum.finite(p1)

mu2 <- sum.finite(p2 \* x) / sum.finite(p2)

sigma1 <- sqrt(sum.finite(p1 \* (x-mu1)^2) / sum.finite(p1))

sigma2 <- sqrt(sum.finite(p2 \* (x-mu2)^2) / sum.finite(p2))

p1 <- pi1

p2 <- pi2

k <- k + 1

Q[k] <- sum(log(comp.sum))

}

library(mixtools)

gm<-normalmixEM(x,k=2,lambda=c(0.9,0.1),mu=c(0.4,0.3),sigma=c(0.05,0.02))

gm$mu

gm$sigma

gm$lambda

hist(x, prob=T, breaks=32, xlim=c(range(x)[1], range(x)[2]), main='')

lines(density(x), col="green", lwd=2)

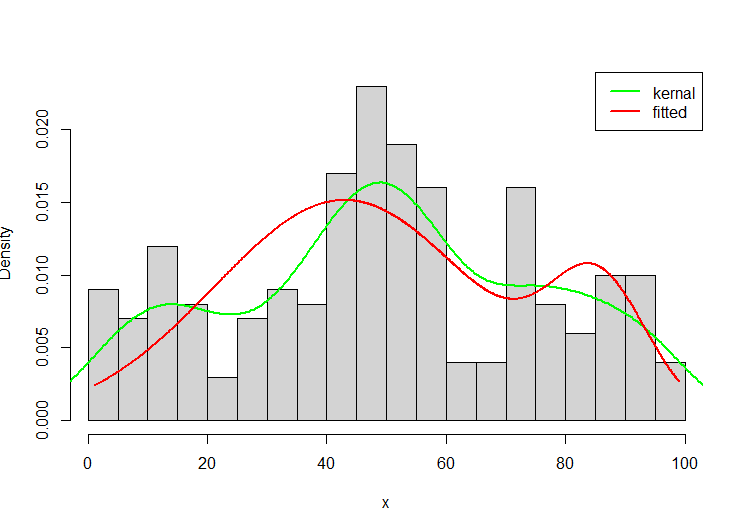
x1 <- seq(from=range(x)[1], to=range(x)[2], length.out=1000)

y <- pi1 \* dnorm(x1, mean=mu1, sd=sigma1) + pi2 \* dnorm(x1, mean=mu2, sd=sigma2)

lines(x1, y, col="red", lwd=2)

legend('topright', col=c("green", 'red'), lwd=2, legend=c("kernal", "fitted"))

**Output:**



**Assignment - 10**

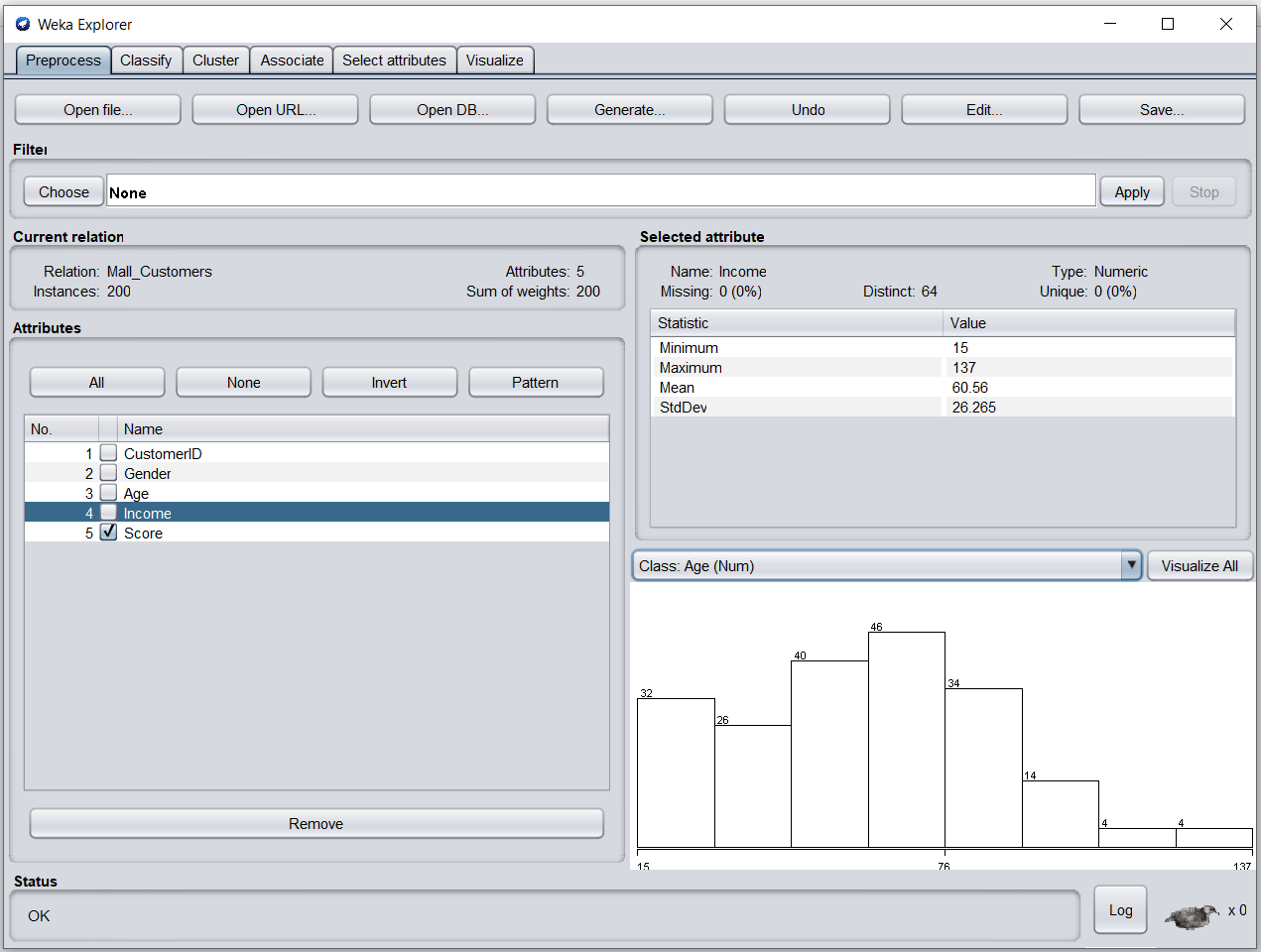
1. **Build Data Warehouse, install and Explore WEKA.**

Weka is a data mining visualization tool which contains collection of machine learning algorithms for data mining tasks.

**Features of Weka:**

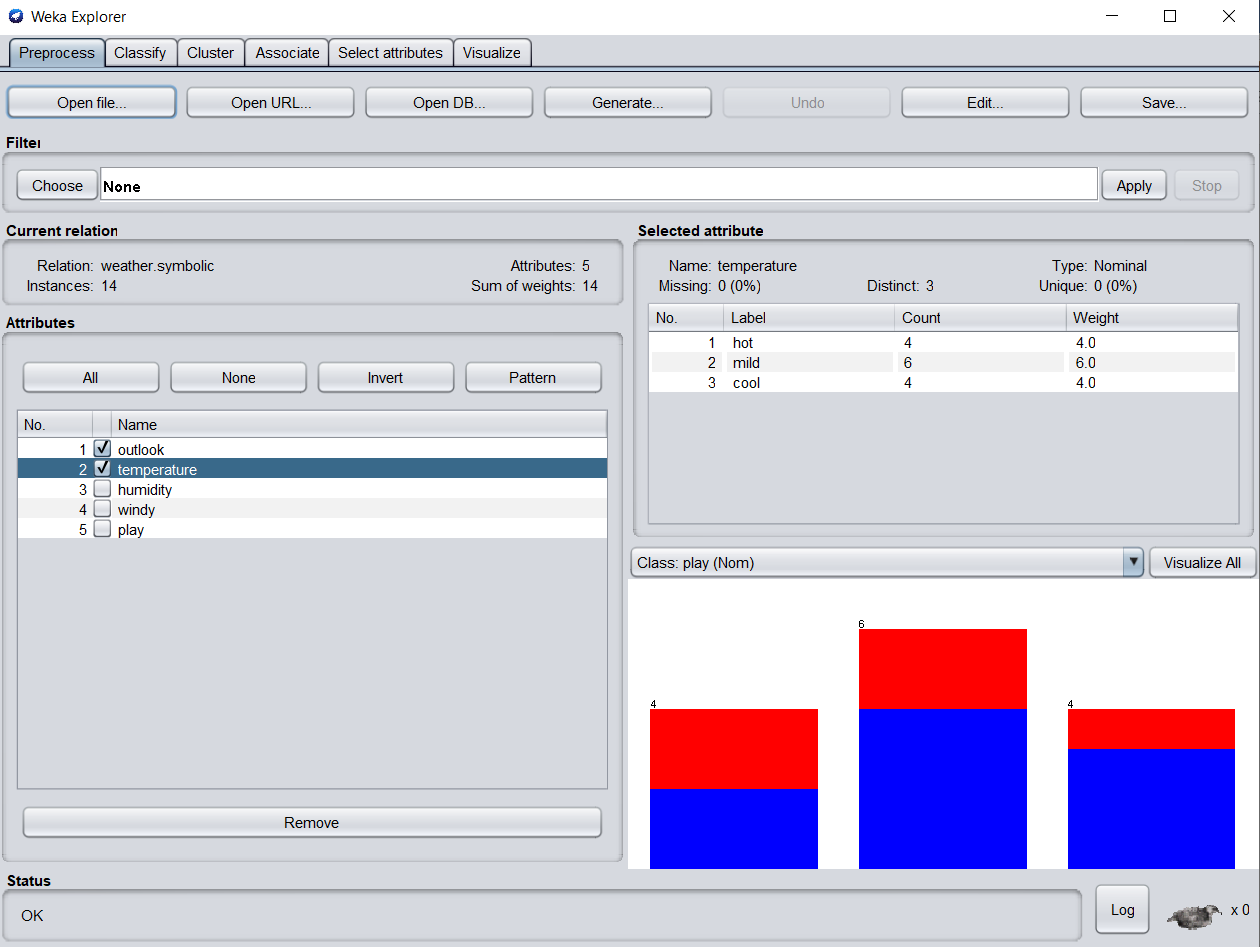
* **Data Preprocessing:** It is cleaning of data while data gathering and selection phase. It removes/adds default value to missing fields and resolve conflicts.
* **Data Classification and Prediction:** It classifies data based on relationship between things and predicts data label. For e.g., A Bank, based on available data of loan, classifies and predicts customer label ‘risky’ or ‘safe’.
* **Clustering:** Group of related data into cluster, used to discover distinct group. For e.g., We have data of weather and based on that we want to decide whether to play outside or not, in such case, using Weka tool we can visualize overall data and can make decision according to the charts.

Below is a simple example of preprocess using Weka.



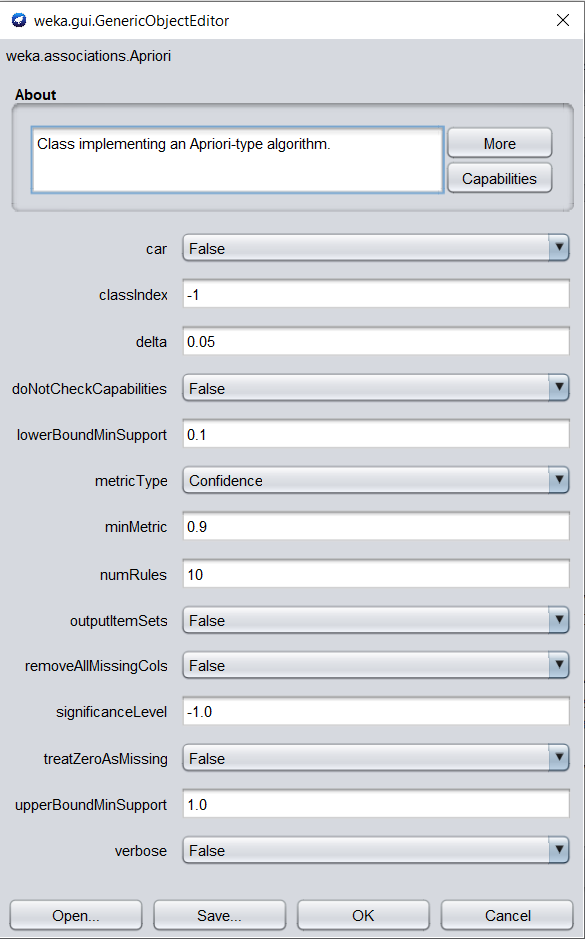
1. **Perform data pre-processing tasks and Demonstrate performing association rule mining on datasets using WEKA.**

**Preprocessing:**



**Association:**

We can set the parameters using the below interface.



Generating association rules-

