Data Mining

Q.1) Implementation of Linear Regression in R.

```
# Free up memory

rm(list = ls())

# Garbage Collection

gc()

# dataset

# Model building.

# Prediction -- predict weight for a given height.

# The predictor vector.

#height : The predictor vector.

x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

#Weight : The response vector.

y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

plot(x, y, col = "blue", main = "X-Y plot for Regression",pch=8, xlab = "Height", ylab = "Weight")

# Apply the lm() function

reg <- lm(y~x)

# to get summary of the relationship

print(summary(reg))

# Plot the chart.

plot(x,y,col = "blue",main = "Height & Weight Regression", pch = 10,xlab = "Height in Kg",ylab
= "Weight in cm")

abline(reg,col="red")

# Find weight of a person with height 170.--- PREDICTION

height<-data.frame(x=70)

predicted_weight <- predict(reg,newdata = height)

print(predicted_weight)
```

Q.2) Implementation and analysis of Classification algorithms:

Implementation of Naive Bayessian Algorithm using "iris" dataset.

```
# Loading data ( you can skip this )
data(iris)
iris
# Structure
str(iris)
# Installing Packages
install.packages("e1071")
install.packages("caTools")
# Loading package
library(e1071)
library(caTools)
# Splitting data into train and test data
split <- sample.split(iris, SplitRatio = 0.8)
train_data <- subset(iris, split == "TRUE")
test_data <- subset(iris, split == "FALSE")
nrow(iris)
nrow(train_data)
nrow(test_data)
#Set seed
set.seed(120)
#creating the model
classifier_model <- naiveBayes(Species ~ ., data = train_data)
classifier_model
# Predicting on test data
y_pred <- predict(classifier_model, newdata = test_data)
y_pred
```

 # Confusion Matrix

```
cm <- table(test_data$Species, y_pred)
cm
```

#Accuracy calculation

```
accuracy = sum(diag(cm))/length(test_data$Species)
accuracy
```

Q.3) Implementation of K-Nearest Neighbor Algorithm using "iris" dataset.

Example 1 :

```
# Loading data ( you can skip this )
data(iris)
iris
# Structure
str(iris)
# training and testing data set
train = iris[1:120,-5]
test = iris[121:150,-5]
library(class)
# creating the model
model = knn(train,test,iris[1:120,5],k=6)
summary(model)
# Confusion Matrix
# table(actual,predicted)
cm = table(iris[121:150,5],model)
cm
accuracy = sum(diag(cm))/length(iris[121:150,5])
accuracy
sprintf("Accuracy: %.2f%%", accuracy*100)
```

Example 2 :

 # study a Cancer dataset and build a Machine Learning model that predicts whether a patientcan be diagnosed as Malignant or Benign.

#Logic: This problem statement can be solved using the KNN algorithm that will classify the

diagnosis into two classes:

 # 1. Malignant, M

 # 2. Benign, B

#install packages

```
install.packages("class")

library(class)
```

 # Adding the dataset

```
wdbc <- read.csv("cancer_dataset.csv", header = TRUE)
```

 # Uncomment the following line to manually choose the file

 # wdbc <- read.csv(file.choose(), header = TRUE)

#Data Cleaning

#Removing the first column ,id , which is unnecessary

```
Wdbc<-wdbc[,-1]
```

#Normalize the data

#The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values.

 # Function to normalize data

```
data_norm <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}
```

 # Normalize the dataset, excluding the first column

```
wdbc_norm <- data.frame(lapply(wdbc[,-1], data_norm))

summary(wdbc[,2:5])

summary(wdbc_norm[,1:4])
```

#Creating Training and Testing dataset

```
wdbc_train <- wdbc_norm[1:450,]

wdbc_test <- wdbc_norm[451:569,]
```

#Applying KNN model

#The knn () function needs to be used to train a model for which we need to install a package 'class'. The knn() function identifies the k-nearest neighbours using Euclidean distance where k is a user-specified number.

#The value for k is generally chosen as the square root of the number of observations.

#knn() returns a factor value of predicted labels for each of the examples in the test data set which is then assigned to the data frame.

```
wdbc_pred <- knn(wdbc_train,wdbc_test,wdbc[1:450,1],k=21)

wdbc_pred
```

#confusion matrix or frequency table

#Table function in R table(), performs categorical tabulation of data with the variable and its frequency.c

```
cm = table(wdbc[451:569,1],wdbc_pred)

cm
```

Q4) Implementation of C4.5 on the iris dataset.

```
#install the package Rweka
install.packages("RWeka")
#load the package
library(RWeka)
#loading data and splitting it into training and testing data
data_train <- iris[1:105,]
data_test <- iris[106:150,]
#fit model
#summarize the model
fit <- J48(Species ~ ., data = data_train)
summary(fit)
#make predictions
predictions <- predict(fit, data_test)
#confusion matrix
cm = table(predictions,iris[106:150,5])
cm
#accuracy
accuracy = sum(diag(cm))/length(iris[106:150,5])
accuracy
```

Q.5) Implementation and analyis of Apriori Algorithm using Market Basket Analysis.

Example 1 :

```
#install.packages("plyr")
library(plyr)
#install.packages("arules")
library(arules)
#install.packages("arulesViz")
library(arulesViz)
#install.packages("ggplot2")
library(ggplot2)
# Read the data
df_groceries <- read.csv("Groceries_dataset.csv")
# Data cleaning and manipulations using R
#First make sure that the Member numbers are of numeric data type and then
#sort the dataframe based on the Member_number.
df_sorted <- df_groceries[order(df_groceries$Member_number),]
View(df_sorted)
df_sorted$Member_number <- as.numeric(df_sorted$Member_number)
#convert the dataframe into transactions format such that all the items bought at the same time in one row.
#ie; convert the dataframe into basket format based on the Member_number and Date of transaction #ddply: Split data frame, apply function, and return results in a data frame.
df_itemList <- ddply(df_groceries,c("Member_number","Date"),
function(df1)paste(df1$itemDescription,collapse = ","))
View(df_itemList)
#Once we have the transactions, we no longer need the date and member numbers in our analysis. Delete those columns.
df_itemList$Member_number <- NULL
df_itemList$Date <- NULL
View(df_itemList)
#Rename column headers for ease of use.
```

```
colnames(df_itemList) <- c("ItemList")

View(df_itemList)

#Write dataframe to a csv file using write.csv()

write.csv(df_itemList,"new_Grocery_ItemList1.csv", row.names = TRUE)

#Find the association rules

#Run algorithm on Grocery_ItemList.csv to find relationships among the items

#Using the read.transactions() functions, we can read the file ItemList.csv and convert it to a transaction format

txn = read.transactions(file="new_Grocery_ItemList1.csv", rm.duplicates= TRUE,

                format="basket",sep=",",cols=1);

txn

#Run the apriori algorithm on the transactions by specifying minimum values for support and confidence.

basket_rules <- apriori(txn,parameter = list(sup = 0.01, conf = 0.01));

print(basket_rules)

#inspect() function prints the internal representation of an R object or the result of an expression.

inspect(basket_rules)

plot(basket_rules)

#Graph to display top 5 items

itemFrequencyPlot(txn, topN = 5)
```

Example 2 :

```
# Loading Libraries

library(arules)

library(arulesViz)

library(RColorBrewer)
```

```
# import dataset
data("Groceries")
# apriori() function
rules <- apriori(Groceries,parameter = list(supp = 0.01, conf = 0.2))
# using inspect() function
inspect(rules[1:10])
# itemFrequencyPlot() function
itemFrequencyPlot(Groceries, topN = 10)
```

Q.6) Implementation and analysis of clustering alogorithms :

K-Means

```
# Free up memory
rm(list = ls())
# Garbage Collection
gc()
# install and import necessary packages.
install.packages(fpc)
install.packages(cluster)
library(cluster)
library(fpc)
# Loading data ( you can skip this )
data(iris)
iris
# Structure
summary(iris)
# Remove the class variable
data1 <- iris[, -5]
View(data1)
head(data1)
plot(data1)
# K-means clustering
set.seed(789)
clust <- kmeans(data1, centers = 3, iter.max = 10)
clust
#Plotting the Clusters
plotcluster(data1, clust$cluster)
```

Q.7) Agglomerative Hierarchical Clustering

```
# Free up memory

rm(list = ls())

# Garbage Collection

gc()

# install and import necessary packages.

install.packages("cluster")

library(cluster)

data1 <- read.csv("seeds_dataset1.csv", header = TRUE)

data1

plot(data1)

# To calculate the distance provide 'method' parameter to the dist function.

# Available distance measures are Euclidean, Manhattan, Maximum, Minkowski, etc.

 # To use cosine similarity measure, use cosine() function in 'lsa' package. dist() function computes and returns the distance matrix computed by using the specified distance measure to compute the distances between the rows of a data matrix.

#Default method is "euclidean" but options include "maximum", "manhattan", #"canberra", "binary" or "minkowski".

distMat <- dist(data1,method = "euclidean")

distMat

# Use 'method' option of hclust() to provide merging criteria like, single, complete, average, etc.

Clust1 <- hclust(distMat,method="single")

Clust1

plot(Clust1)

# Convert hclust into a dendrogram and plot

dend <- as.dendrogram(Clust1)

plot(dend)
```