INDEX

Prac. No.	Practical	Date	Sign
1	Install, configure and run Hadoop and HDFS and explore HDFS		
2	Implement Decision tree classification techniques		
3	Implement SVM classification techniques.		
4	Implement of REGRESSION MODLE.		
5	Implement of Simple Linear Regression.		
6	Implement of Multiple Linear Regression.		
7	Implement of Logistic regression.		
8	Read a datafile grades_km_input.csv and apply k-means clustering		
9	Perform Apriori algorithm using Groceries dataset from the R rules package		

Practical 1

Aim: -Install, configure and run Hadoop and HDFS and explore HDFS on Windows

Code:

Steps to Install Hadoop :

1. Install Java JDK 1.8
2. Download Hadoop and extract and place under C drive
3. Set Path in Environment Variables
4. Config files under Hadoop directory
5. Create folder data node and name node under data directory
6. Edit HDFS and YARN files
7. Set Java Home environment in Hadoop environment
8. Setup Complete. Test by executing start-all.cmd

There are two ways to install Hadoop, i.e.

9. Single node
10. Multi node

Here, we use multi node cluster.

1. **Install Java**

11. – Java JDK Link to download (https://www.oracle.com/java/technologies/javase-jdk8-downloads.html)
12. – extract and install Java in C:\Java
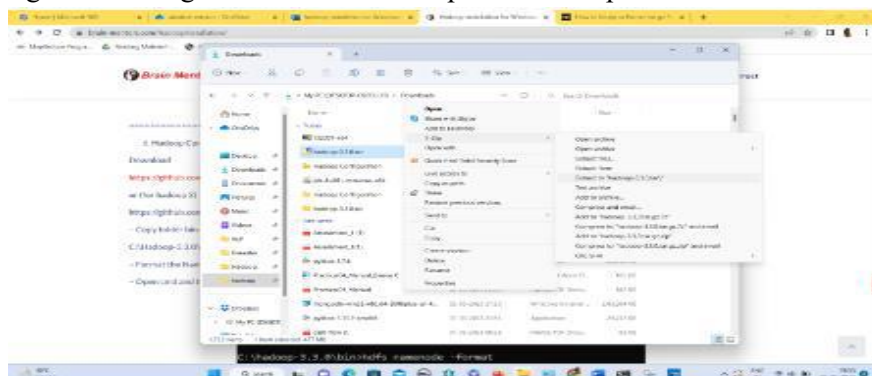13. – open cmd and type -> javac -version

```
C:\Users>cd Beena

C:\Users\Beena>java -version
java version "1.8.0_361"
Java(TM) SE Runtime Environment (build 1.8.0_361-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.361-b09, mixed mode)
```
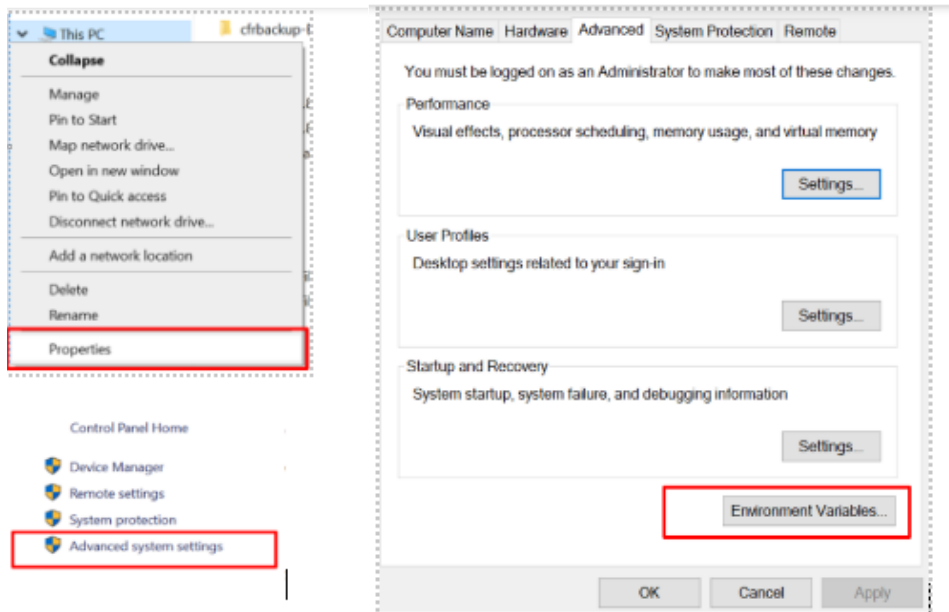
2. **Download Hadoop**

Link: https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz

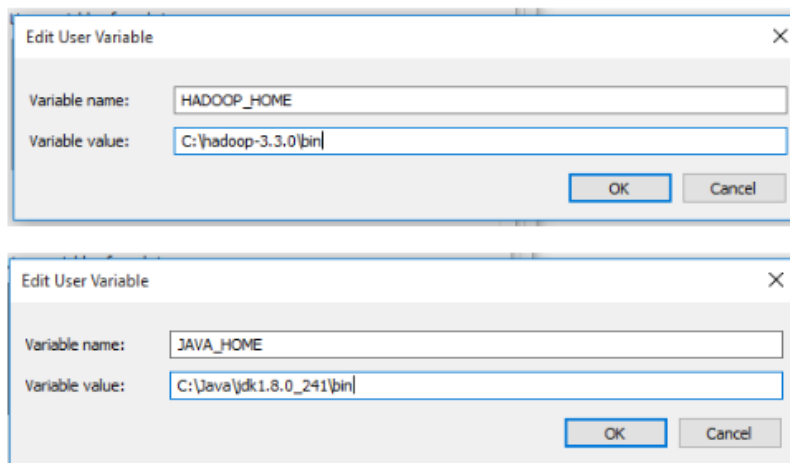- right click .rar.gz file -> show more options -> 7-zip->and extract to C:\Hadoop-3.3.0\

Mahesh Sampat More   MSc – IT(Part – I)   Seat No: 1312680

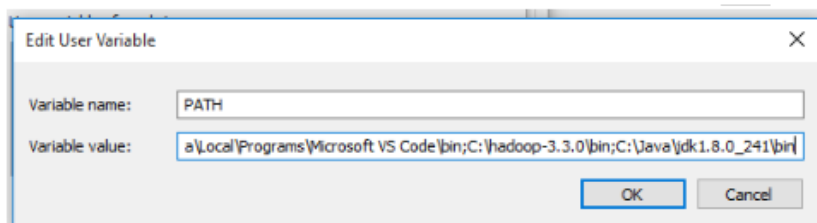3. **Set the path JAVA_HOME Environment variable**

4. **Set the path HADOOP_HOME Environment variable**

Click on New to both user variables and system variables.

Click on user variable **-> path -> edit-> add path for Hadoop and java up to 'bin'**

Click Ok, Ok, Ok.

5.**Configurations**

Edit file -> **C:/Hadoop-3.3.0/etc/hadoop/core-site.xml**

paste the xml code in folder and save

===

```
<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>
```

===

Rename "mapred-site.xml. Template" to "mapred-site.xml" and edit this file C:/Hadoop-3.3.0/etc/hadoop/mapred-site.xml, paste xml code and save this file.

===

```
<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>
```

===

Create folder "data" under "C:\Hadoop-3.3.0"

Create folder "datanode" under "C:\Hadoop-3.3.0\data"

Create folder "namenode" under "C:\Hadoop-3.3.0\data"

===

Edit file C:\Hadoop-3.3.0/etc/hadoop/hdfs-site.xml,

paste xml code and save this file.

```
<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>
```

```
<name>dfs.namenode.name.dir</name>

<value>/hadoop-3.3.0/data/namenode</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>/hadoop-3.3.0/data/datanode</value>

</property>

</configuration>
```

==

Edit file C:/Hadoop-3.3.0/etc/hadoop/yarn-site.xml,

paste xml code and save this file.

```
<configuration>

</configuration>
```

==

6. **Edit file C:/Hadoop-3.3.0/etc/hadoop/hadoop-env.cmd**

Find "JAVA_HOME=%JAVA_HOME%" and replace it as

set JAVA_HOME="C:\Java\jdk1.8.0_361"

==

7. **Download "redistributable" package**

Download and run VC_redist.x64.exe: This is a "redistributable" package of the Visual C runtime code for 64-bit applications, from Microsoft. It contains certain shared code that every application written with Visual C expects to have available on the Windows computer it runs on.

==

8. **Hadoop Configurations**

Download bin folder from

https://github.com/s911415/apache-hadoop-3.1.0-winutils

--- Copy the bin folder to c:\hadoop-3.3.0. Replace the existing bin folder.

==

9. **copy "hadoop-yarn-server-timelineservice-3.0.3.jar" from ~\hadoop-3.0.3\share\hadoop\yarn\timelineservice to ~\hadoop-3.0.3\share\hadoop\yarn folder.**

==

10. Format the NameNode

– Open cmd 'Run as Administrator' and type command "hdfs namenode –format"

```
Microsoft Windows [Version 10.0.22621.1265]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd\hadoop-3.3.0\bin

C:\hadoop-3.3.0\bin>hdfs namenode -format
```

```
2023-03-07 21:31:34,685 INFO namenode.FSImageFormatProtobuf: Saving image file C:\hadoop-3.3.0\data\namenode\current\fsi
mage.ckpt_0000000000000000000 using no compression
2023-03-07 21:31:34,844 INFO namenode.FSImageFormatProtobuf: Image file C:\hadoop-3.3.0\data\namenode\current\fsimage.ck
pt_0000000000000000000 of size 400 bytes saved in 0 seconds .
2023-03-07 21:31:34,860 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2023-03-07 21:31:34,869 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 when meet shutdown.
2023-03-07 21:31:34,870 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at DESKTOP-OL8EULH/192.168.1.19
************************************************************/
```

11. Testing

– Open cmd 'Run as Administrator' and change directory to C:\Hadoop-3.3.0\sbin

– type start-all.cmd

OR

- type start-dfs.cmd

– type start-yarn.cmd

```
C:\hadoop-3.3.0\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
The filename, directory name, or volume label syntax is incorrect.
The filename, directory name, or volume label syntax is incorrect.
starting yarn daemons
The filename, directory name, or volume label syntax is incorrect.
```

– You will get 4 more running threads for Datanode, namenode, resource manager and node manager

```
Apache Hadoop Distribution - hadoop namenode
2023-03-07 20:33:00,395 INFO ipc.Server: Starting Socket Reader #1 for port 9000
2023-03-07 20:33:00,547 INFO namenode.FSNamesystem: Registered FSNamesystemState, ReplicatedBlocksState and ECBlockGroup
sState MBeans.
2023-03-07 20:33:00,549 INFO common.Util: Assuming 'file' scheme for path /hadoop-3.3.0/data/namenode in configuration.
2023-03-07 20:33:00,554 INFO namenode.LeaseManager: Number of blocks under construction: 0
2023-03-07 20:33:00,563 INFO blockmanagement.DatanodeAdminDefaultMonitor: Initialized the Default Decommission and Maint
enance monitor
2023-03-07 20:33:00,566 INFO blockmanagement.BlockManager: initializing replication queues
2023-03-07 20:33:00,567 INFO hdfs.StateChange: STATE* Leaving safe mode after 0 secs
2023-03-07 20:33:00,567 INFO hdfs.StateChange: STATE* Network topology has 0 racks and 0 datanodes
2023-03-07 20:33:00,569 INFO hdfs.StateChange: STATE* UnderReplicatedBlocks has 0 blocks
2023-03-07 20:33:00,574 INFO blockmanagement.BlockManager: Total number of blocks            = 0
2023-03-07 20:33:00,575 INFO blockmanagement.BlockManager: Number of invalid blocks          = 0
2023-03-07 20:33:00,575 INFO blockmanagement.BlockManager: Number of under-replicated blocks = 0
2023-03-07 20:33:00,576 INFO blockmanagement.BlockManager: Number of  over-replicated blocks = 0
2023-03-07 20:33:00,576 INFO blockmanagement.BlockManager: Number of blocks being written    = 0
2023-03-07 20:33:00,576 INFO hdfs.StateChange: STATE* Replication Queue initialization scan for invalid, over- and under
 replicated blocks completed in 9 msec
2023-03-07 20:33:00,607 INFO ipc.Server: IPC Server Responder: starting
2023-03-07 20:33:00,607 INFO ipc.Server: IPC Server listener on 9000: starting
2023-03-07 20:33:00,611 INFO namenode.NameNode: NameNode RPC up at: localhost/127.0.0.1:9000
2023-03-07 20:33:00,614 INFO namenode.FSNamesystem: Starting services required for active state
2023-03-07 20:33:00,614 INFO namenode.FSDirectory: Initializing quota with 4 thread(s)
2023-03-07 20:33:00,622 INFO namenode.FSDirectory: Quota initialization completed in 7 milliseconds
name space=1
storage space=0
storage types=RAM_DISK=0, SSD=0, DISK=0, ARCHIVE=0, PROVIDED=0
2023-03-07 20:33:00,626 INFO blockmanagement.CacheReplicationMonitor: Starting CacheReplicationMonitor with interval 300
00 milliseconds
```

Output:

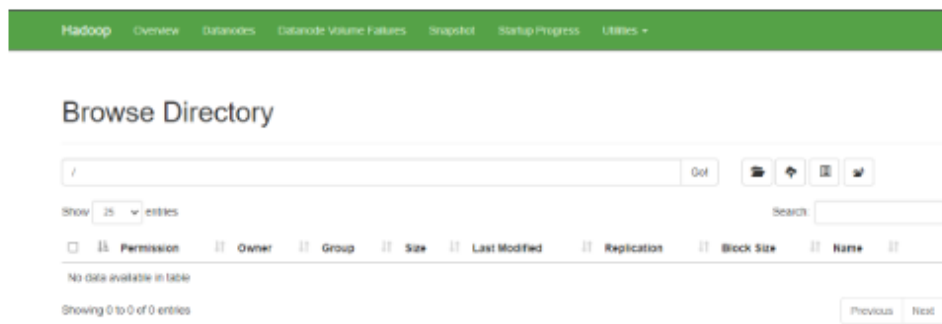**12. Type JPS command to start-all.cmd command prompt, you will get following output.**

```
C:\hadoop-3.3.0\sbin>jps
5632 Jps
7572 DataNode
3752 ResourceManager
7992 NameNode
8028 NodeManager
```

13. **Run http://localhost:9870/ from any browser or http://localhost:50070/**
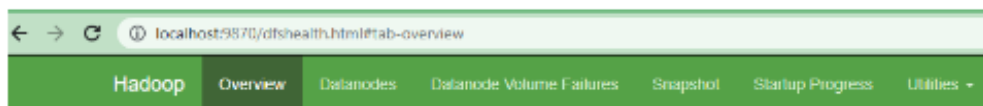
Practical 2

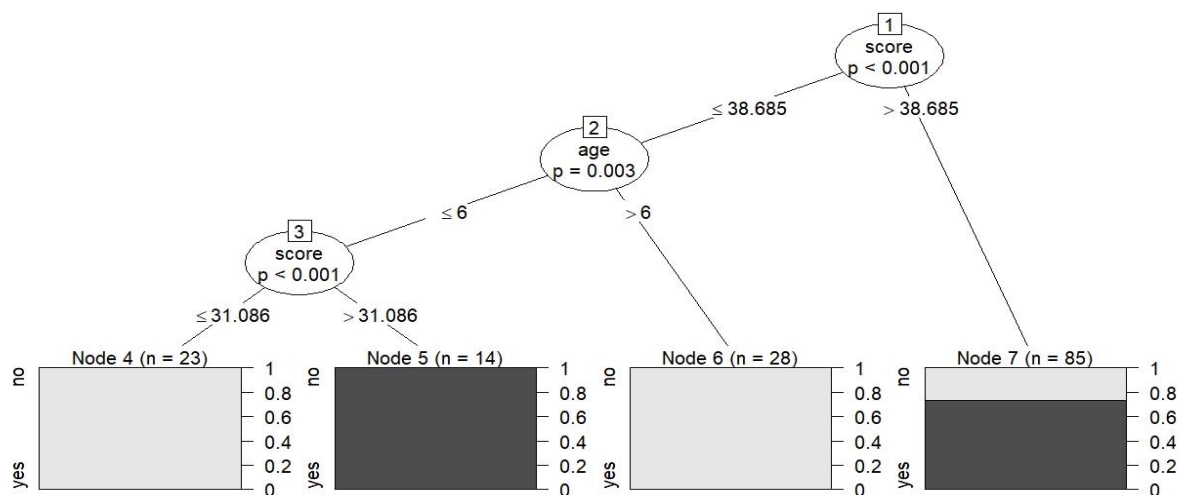**Aim: - Implement Decision tree classification techniques.**

install.packages('datasets')
install.packages('caTools')
install.packages('party')
install.packages('dplyr')
install.packages('magrittr')
library(datasets)
library(caTools)
library(party)
library(dplyr)
library(magrittr)
data("readingSkills")
head(readingSkills)
sample_data = sample.split(readingSkills, SplitRatio = 0.8)
train_data <- subset(readingSkills, sample_data == TRUE)
test_data <- subset(readingSkills, sample_data == FALSE)
model<- ctree(nativeSpeaker ~ ., train_data)
plot(model)

Output:

Practical 3

Aim: - Implement SVM classification techniques.

#Code for installation of all necessary packages
install.packages("caret")
install.packages("ggplot2")
install.packages("GGally")
install.packages("psych")
install.packages("ggpubr")
install.packages("reshape")
# Code for importation of all necessary packages
library(caret)
library(ggplot2)
library(GGally)
library(psych)
library(ggpubr)
library(reshape)
# Code
df <- read.csv("C:/Users/DELL/Downloads/diabetes.csv")
head(df)

```
> # Code
> plot(model)
> df <- read.csv("C:/Users/DELL/Downloads/diabetes.csv")
> head(df)
  Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI DiabetesPedigreeFunction Age Outcome
1           6     148            72            35       0 33.6                    0.627  50       1
2           1      85            66            29       0 26.6                    0.351  31       0
3           8     183            64             0       0 23.3                    0.672  32       1
4           1      89            66            23      94 28.1                    0.167  21       0
5           0     137            40            35     168 43.1                    2.288  33       1
6           5     116            74             0       0 25.6                    0.201  30       0
>
```

# Code
sum(is.na(df))
# Code
dim(df)

```
> # Code
> sum(is.na(df))
[1] 0
> # Code
> dim(df)
[1] 768   9
>
```

# Code
sapply(df, class)
# Code
summary(df) # to calculate the summary of our dataset

```
> summary(df) # to calculate the summary of our dataset
  Pregnancies        Glucose      BloodPressure    SkinThickness      Insulin          BMI
 Min.   : 0.00    Min.   :  0    Min.   :  0.0    Min.   : 0.0    Min.   :  0    Min.   : 0.0
 1st Qu.: 1.00    1st Qu.: 99    1st Qu.: 62.0    1st Qu.: 0.0    1st Qu.:  0    1st Qu.:27.3
 Median : 3.00    Median :117    Median : 72.0    Median :23.0    Median : 30    Median :32.0
 Mean   : 3.85    Mean   :121    Mean   : 69.1    Mean   :20.5    Mean   : 80    Mean   :32.0
 3rd Qu.: 6.00    3rd Qu.:140    3rd Qu.: 80.0    3rd Qu.:32.0    3rd Qu.:127    3rd Qu.:36.6
 Max.   :17.00    Max.   :199    Max.   :122.0    Max.   :99.0    Max.   :846    Max.   :67.1
 DiabetesPedigreeFunction       Age           Outcome
 Min.   :0.078           Min.   :21.0    Min.   :0.000
 1st Qu.:0.244           1st Qu.:24.0    1st Qu.:0.000
 Median :0.372           Median :29.0    Median :0.000
 Mean   :0.472           Mean   :33.2    Mean   :0.349
 3rd Qu.:0.626           3rd Qu.:41.0    3rd Qu.:1.000
 Max.   :2.420           Max.   :81.0    Max.   :1.000
>
```

```
# Code
a <- ggplot(data = df, aes(x = Pregnancies)) +
  geom_histogram( color = "red", fill = "blue", alpha = 0.1) +
  geom_density()
b <- ggplot(data = df, aes(x = Glucose)) +
  geom_histogram( color = "red", fill = "blue", alpha = 0.1) +
  geom_density()
c <- ggplot(data = df, aes(x = BloodPressure)) +
  geom_histogram( color = "red", fill = "blue", alpha = 0.1) +
  geom_density()
d <- ggplot(data = df, aes(x = SkinThickness)) +
  geom_histogram( color = "red", fill = "blue", alpha = 0.1) +
  geom_density()
e <- ggplot(data = df, aes(x = Insulin)) +
  geom_histogram( color = "red", fill = "blue", alpha = 0.1) +
  geom_density()
f <- ggplot(data = df, aes(x = BMI)) +
  geom_histogram( color = "red", fill = "blue", alpha = 0.1) +
  geom_density()
g <- ggplot(data = df, aes(x = DiabetesPedigreeFunction)) +
  geom_histogram( color = "red", fill = "blue", alpha = 0.1) +
  geom_density()
h <- ggplot(data = df, aes(x = Age)) +
  geom_histogram( color = "red", fill = "blue", alpha = 0.1) +geom_density()
ggarrange(a, b, c, d,e,f,g, h + rremove("x.text"),
      labels = c("a", "b", "c", "d","e", "f", "g", "h"),
      ncol = 3, nrow = 3)
```

```
R  ▾  Error fetching R version  ·  ~/
> # Code
> a <- ggplot(data = df, aes(x = Pregnancies)) +
+   geom_histogram( color = "red", fill = "blue", alpha = 0.1) +
+   geom_density()
> b <- ggplot(data = df, aes(x = Glucose)) +
+   geom_histogram( color = "red", fill = "blue", alpha = 0.1) +
+   geom_density()
> c <- ggplot(data = df, aes(x = BloodPressure)) +
+   geom_histogram( color = "red", fill = "blue", alpha = 0.1) +
+   geom_density()
> d <- ggplot(data = df, aes(x = SkinThickness)) +
+   geom_histogram( color = "red", fill = "blue", alpha = 0.1) +
+   geom_density()
> e <- ggplot(data = df, aes(x = Insulin)) +
+   geom_histogram( color = "red", fill = "blue", alpha = 0.1) +
+   geom_density()
> f <- ggplot(data = df, aes(x = BMI)) +
+   geom_histogram( color = "red", fill = "blue", alpha = 0.1) +
+   geom_density()
> g <- ggplot(data = df, aes(x = DiabetesPedigreeFunction)) +
+   geom_histogram( color = "red", fill = "blue", alpha = 0.1) +
+   geom_density()
> h <- ggplot(data = df, aes(x = Age)) +
+   geom_histogram( color = "red", fill = "blue", alpha = 0.1) +geom_density()
> ggarrange(a, b, c, d,e,f,g, h + rremove("x.text"),
+          labels = c("a", "b", "c", "d","e", "f", "g", "h"),
+          ncol = 3, nrow = 3)
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
>
```
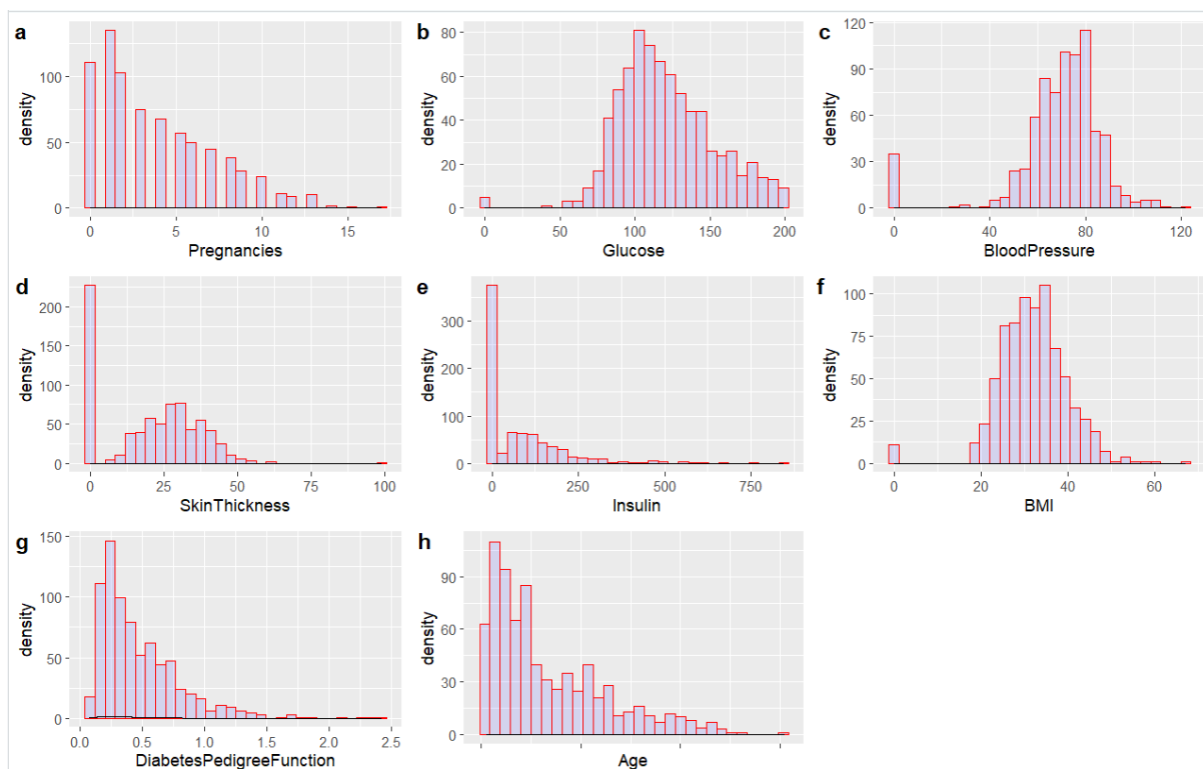
```
# Code
ggplot(data = df, aes(x =Outcome, fill = Outcome)) +
  geom_bar()
```
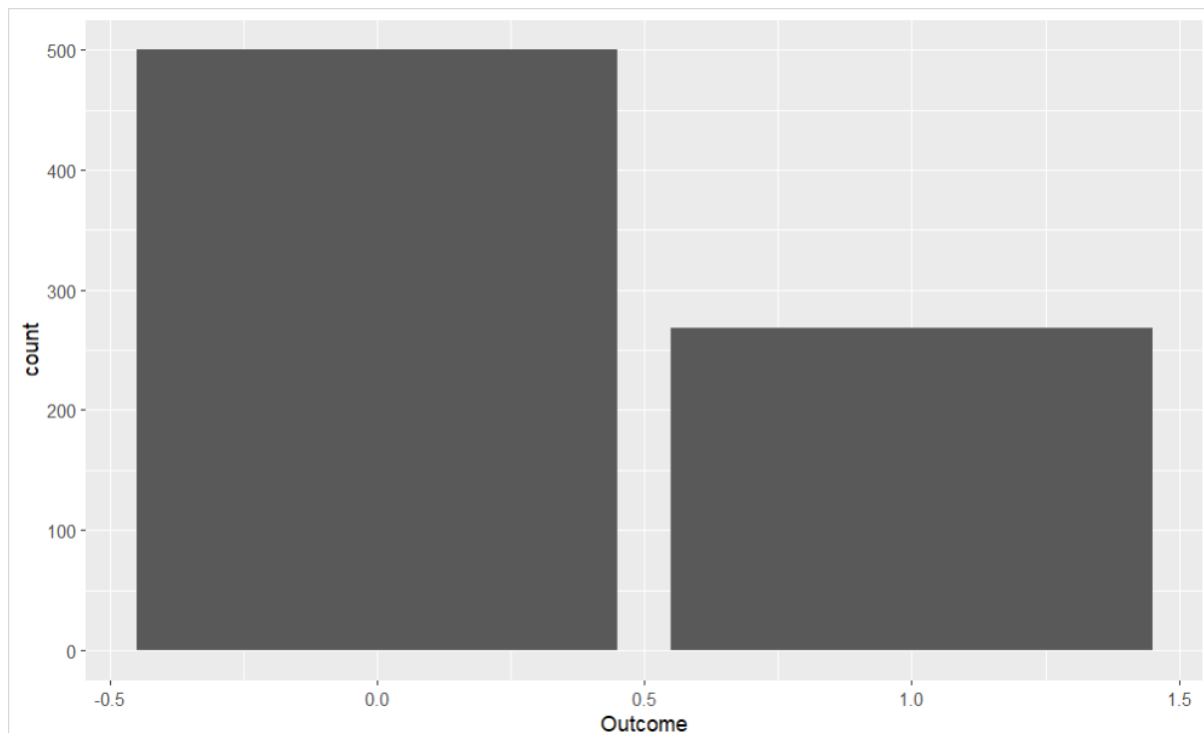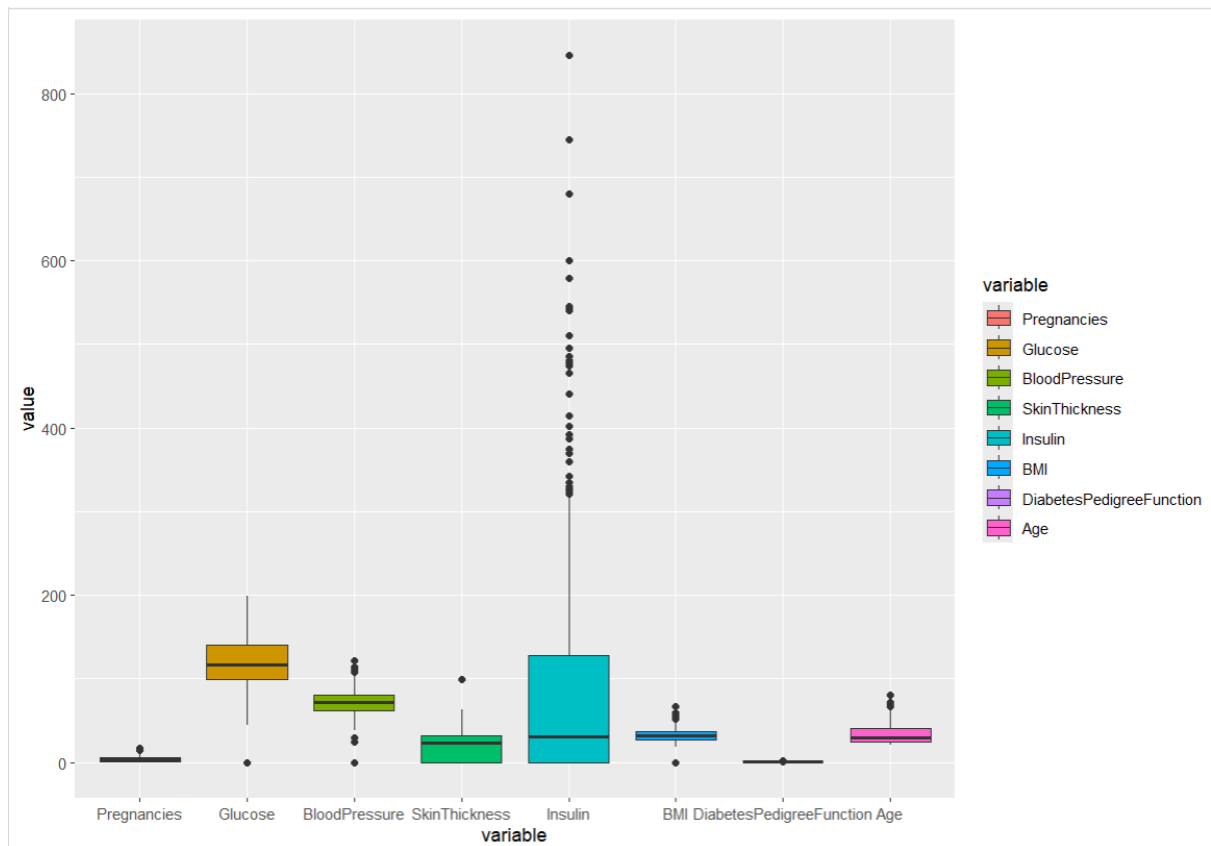


```
# Code to label our categorical variable as a factor
df$Outcome<- factor(df$Outcome,
            levels = c(0, 1),
            labels = c("Negative", "Positive"))
out <- subset(df,
        select = c(Pregnancies,Glucose,
                BloodPressure,SkinThickness,
                Insulin,BMI,
                DiabetesPedigreeFunction,Age))
# Code for boxplot
ggplot(data = melt(out),
    aes(x=variable, y=value)) +
  geom_boxplot(aes(fill=variable))
```
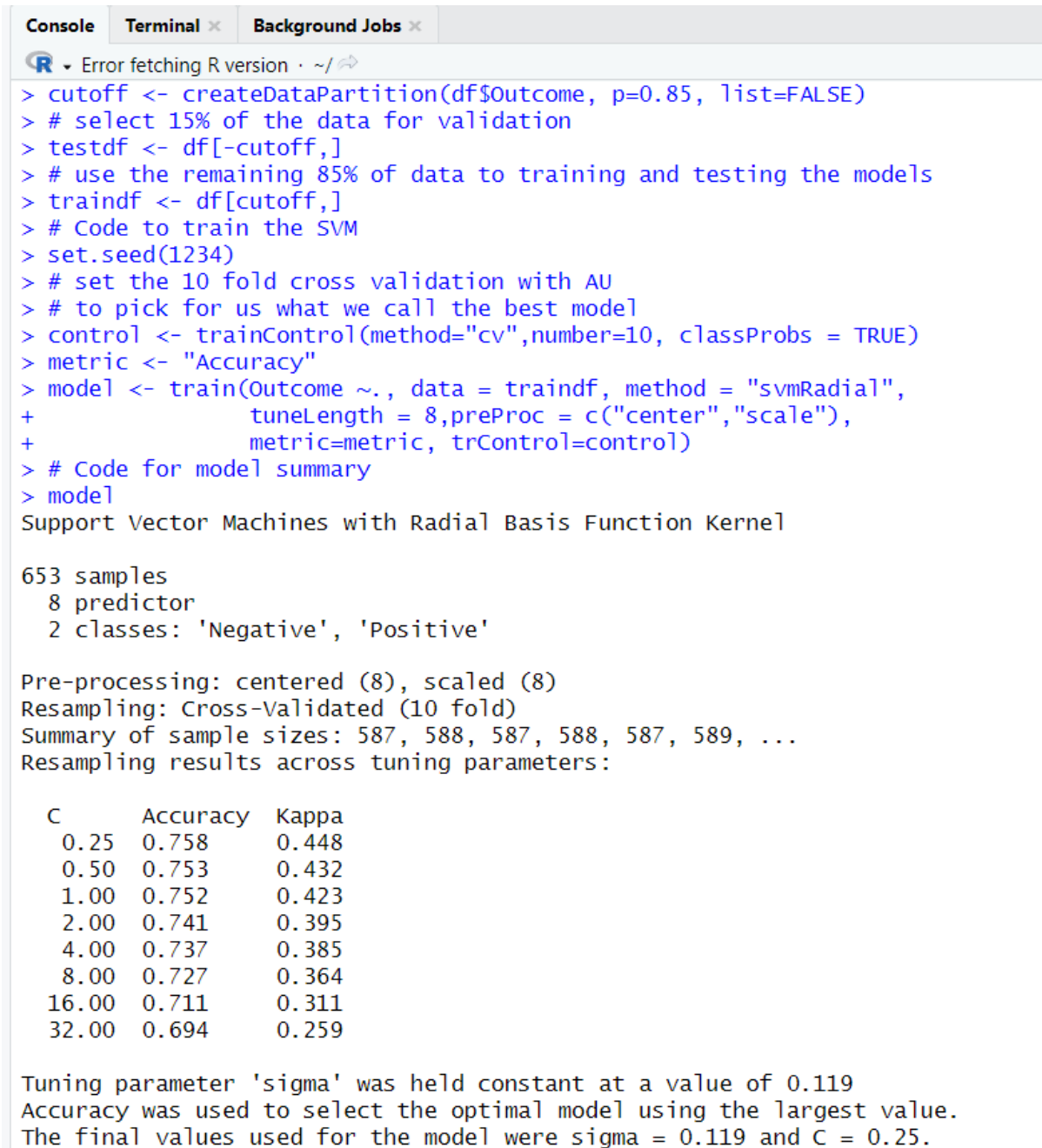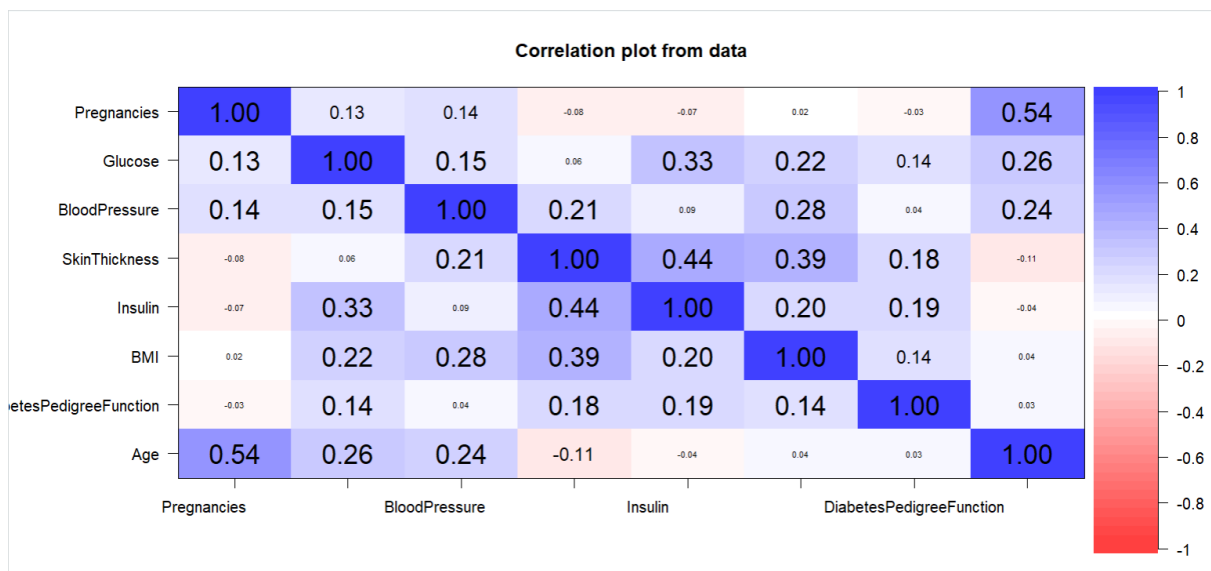
corPlot(df[, 1:8])

Correlation plot from data

cutoff <- createDataPartition(df$Outcome, p=0.85, list=FALSE)
# select 15% of the data for validation
testdf <- df[-cutoff,]
# use the remaining 85% of data to training and testing the models
traindf <- df[cutoff,]
# Code to train the SVM
set.seed(1234)

# set the 10 fold cross validation with AU
# to pick for us what we call the best model
control <- trainControl(method="cv",number=10, classProbs = TRUE)
metric <- "Accuracy"
model <- train(Outcome ~., data = traindf, method = "svmRadial",
        tuneLength = 8,preProc = c("center","scale"),
        metric=metric, trControl=control)
# Code for model summary
model

```
Console   Terminal ×   Background Jobs ×

R ▾ Error fetching R version · ~/ ⇖
> cutoff <- createDataPartition(df$Outcome, p=0.85, list=FALSE)
> # select 15% of the data for validation
> testdf <- df[-cutoff,]
> # use the remaining 85% of data to training and testing the models
> traindf <- df[cutoff,]
> # Code to train the SVM
> set.seed(1234)
> # set the 10 fold cross validation with AU
> # to pick for us what we call the best model
> control <- trainControl(method="cv",number=10, classProbs = TRUE)
> metric <- "Accuracy"
> model <- train(Outcome ~., data = traindf, method = "svmRadial",
+                tuneLength = 8,preProc = c("center","scale"),
+                metric=metric, trControl=control)
> # Code for model summary
> model
Support Vector Machines with Radial Basis Function Kernel

653 samples
  8 predictor
  2 classes: 'Negative', 'Positive'

Pre-processing: centered (8), scaled (8)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 587, 588, 587, 588, 587, 589, ...
Resampling results across tuning parameters:

  C       Accuracy  Kappa
   0.25   0.758     0.448
   0.50   0.753     0.432
   1.00   0.752     0.423
   2.00   0.741     0.395
   4.00   0.737     0.385
   8.00   0.727     0.364
  16.00   0.711     0.311
  32.00   0.694     0.259

Tuning parameter 'sigma' was held constant at a value of 0.119
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.119 and C = 0.25.
```
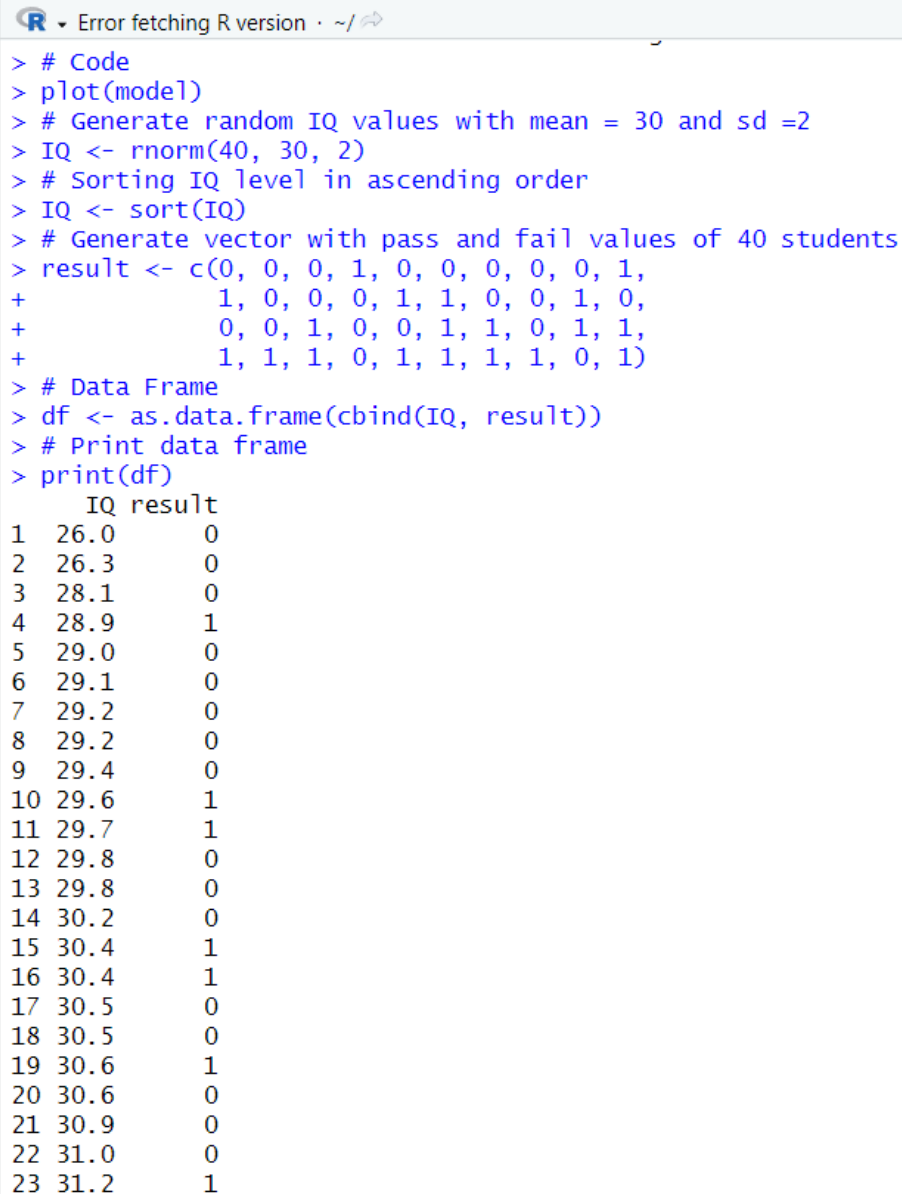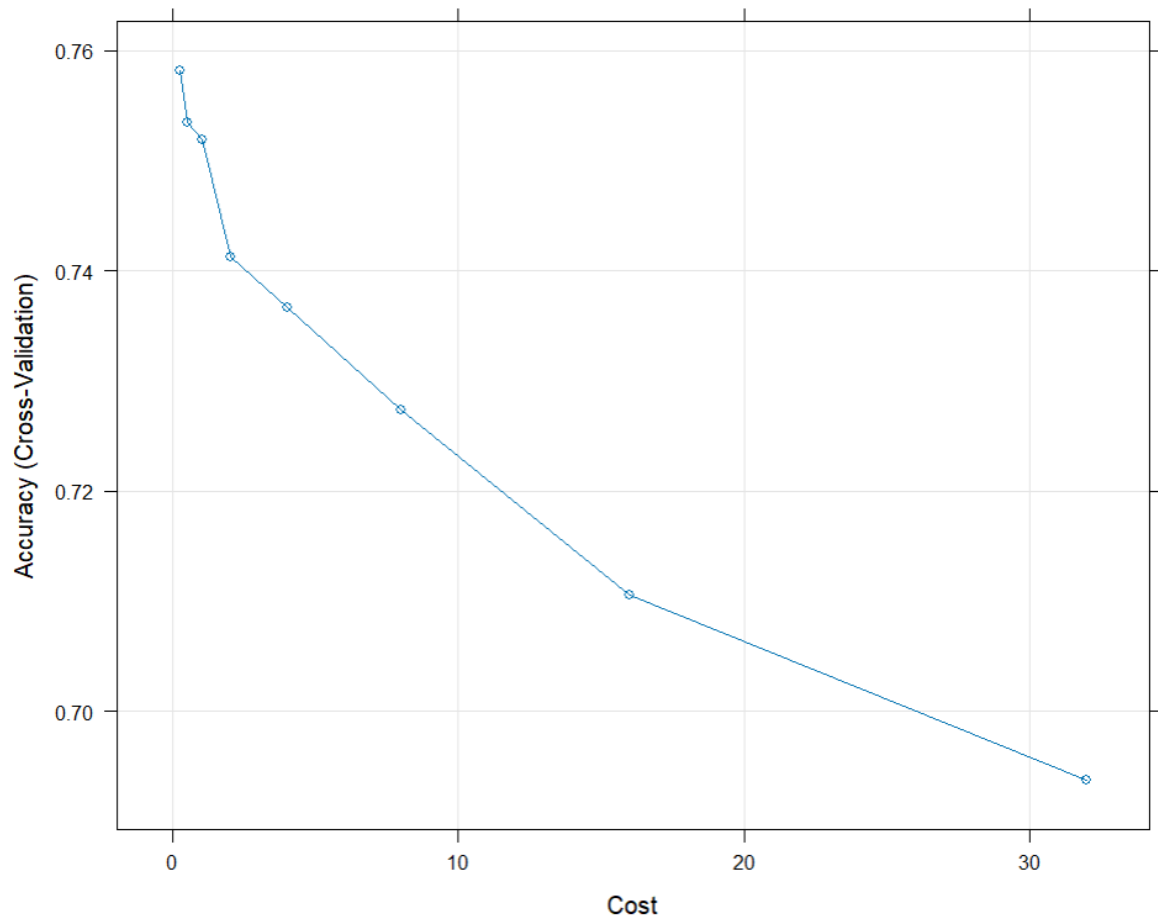
# Code
plot(model)

Practical 4

Aim: - Implement of REGRESSION MODLE.

```
# Generate random IQ values with mean = 30 and sd =2
IQ <- rnorm(40, 30, 2)
# Sorting IQ level in ascending order
IQ <- sort(IQ)
# Generate vector with pass and fail values of 40 students
result <- c(0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
        1, 0, 0, 0, 1, 1, 0, 0, 1, 0,
        0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
        1, 1, 1, 0, 1, 1, 1, 1, 0, 1)
# Data Frame
df <- as.data.frame(cbind(IQ, result))
# Print data frame
print(df)
```

```
R ▾ Error fetching R version · ~/
> # Code
> plot(model)
> # Generate random IQ values with mean = 30 and sd =2
> IQ <- rnorm(40, 30, 2)
> # Sorting IQ level in ascending order
> IQ <- sort(IQ)
> # Generate vector with pass and fail values of 40 students
> result <- c(0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
+             1, 0, 0, 0, 1, 1, 0, 0, 1, 0,
+             0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
+             1, 1, 1, 0, 1, 1, 1, 1, 0, 1)
> # Data Frame
> df <- as.data.frame(cbind(IQ, result))
> # Print data frame
> print(df)
     IQ result
1  26.0      0
2  26.3      0
3  28.1      0
4  28.9      1
5  29.0      0
6  29.1      0
7  29.2      0
8  29.2      0
9  29.4      0
10 29.6      1
11 29.7      1
12 29.8      0
13 29.8      0
14 30.2      0
15 30.4      1
16 30.4      1
17 30.5      0
18 30.5      0
19 30.6      1
20 30.6      0
21 30.9      0
22 31.0      0
23 31.2      1
```

```
# Plotting IQ on x-axis and result on y-axis
plot(IQ, result, xlab = "IQ Level",
     ylab = "Probability of Passing")
# Create a logistic model
g = glm(result~IQ, family=binomial, df)
# Create a curve based on prediction using the regression model
curve(predict(g, data.frame(IQ=x), type="resp"), add=TRUE)
# Based on fit to the regression model
points(IQ, fitted(g), pch=30)
```
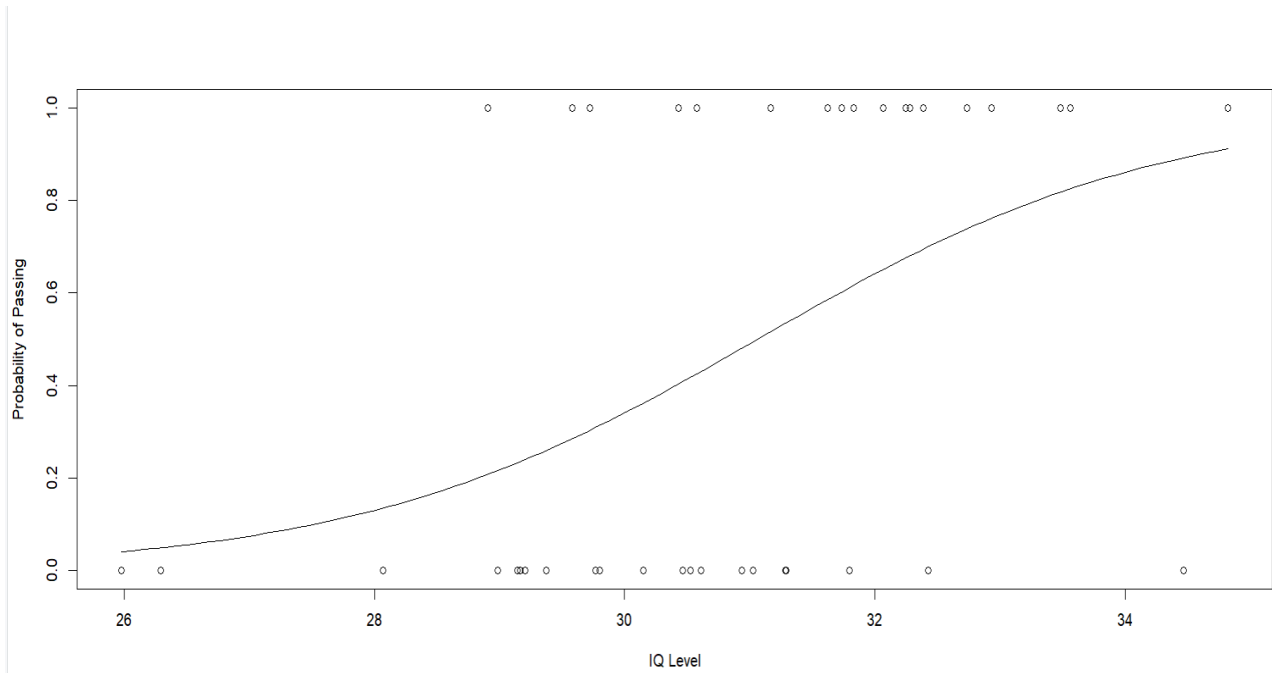


```
# Summary of the regression model
summary(g)
```

```
> summary(g)

Call:
glm(formula = result ~ IQ, family = binomial, data = df)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -19.252      7.624   -2.53    0.012 *
IQ             0.620      0.246    2.52    0.012 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 55.352  on 39  degrees of freedom
Residual deviance: 46.336  on 38  degrees of freedom
AIC: 50.34

Number of Fisher Scoring iterations: 4
```

Practical 5

Aim:- Implement of Simple Linear regression.

years_of_exp = c(7,5,1,3)
salary_in_lakhs = c(21,13,6,8)
#employee.data = data.frame(satisfaction_score, years_of_exp, salary_in_lakhs)
employee.data = data.frame(years_of_exp, salary_in_lakhs)
employee.data
Estimation of the salary of an employee, based on his year of experience and satisfaction score in
#his company.
model <- lm(salary_in_lakhs ~ years_of_exp, data = employee.data)
summary(model)
# The formula of Regression becomes
# Y = 2 + 2.5*year_of_Exp
# Visualization of Regression
plot(salary_in_lakhs ~ years_of_exp, data = employee.data)
abline(model)

```
> years_of_exp = c(7,5,1,3)
> salary_in_lakhs = c(21,13,6,8)
> #employee.data = data.frame(satisfaction_score, years_of_exp, salary_in_lakhs)
> employee.data = data.frame(years_of_exp, salary_in_lakhs)
> employee.data
  years_of_exp salary_in_lakhs
1            7              21
2            5              13
3            1               6
4            3               8
> # Estimation of the salary of an employee, based on his year of experience and satis
faction score in his company.
> model <- lm(salary_in_lakhs ~ years_of_exp, data = employee.data)
> summary(model)

Call:
lm(formula = salary_in_lakhs ~ years_of_exp, data = employee.data)

Residuals:
   1    2    3    4
 1.5 -1.5  1.5 -1.5

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)     2.000      2.174    0.92    0.455
years_of_exp    2.500      0.474    5.27    0.034 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.12 on 2 degrees of freedom
Multiple R-squared:  0.933,    Adjusted R-squared:  0.899
F-statistic: 27.8 on 1 and 2 DF,  p-value: 0.0342

> # The formula of Regression becomes
> # Y = 2 + 2.5*year_of_Exp
> # Visualization of Regression
> plot(salary_in_lakhs ~ years_of_exp, data = employee.data)
> abline(model)
```
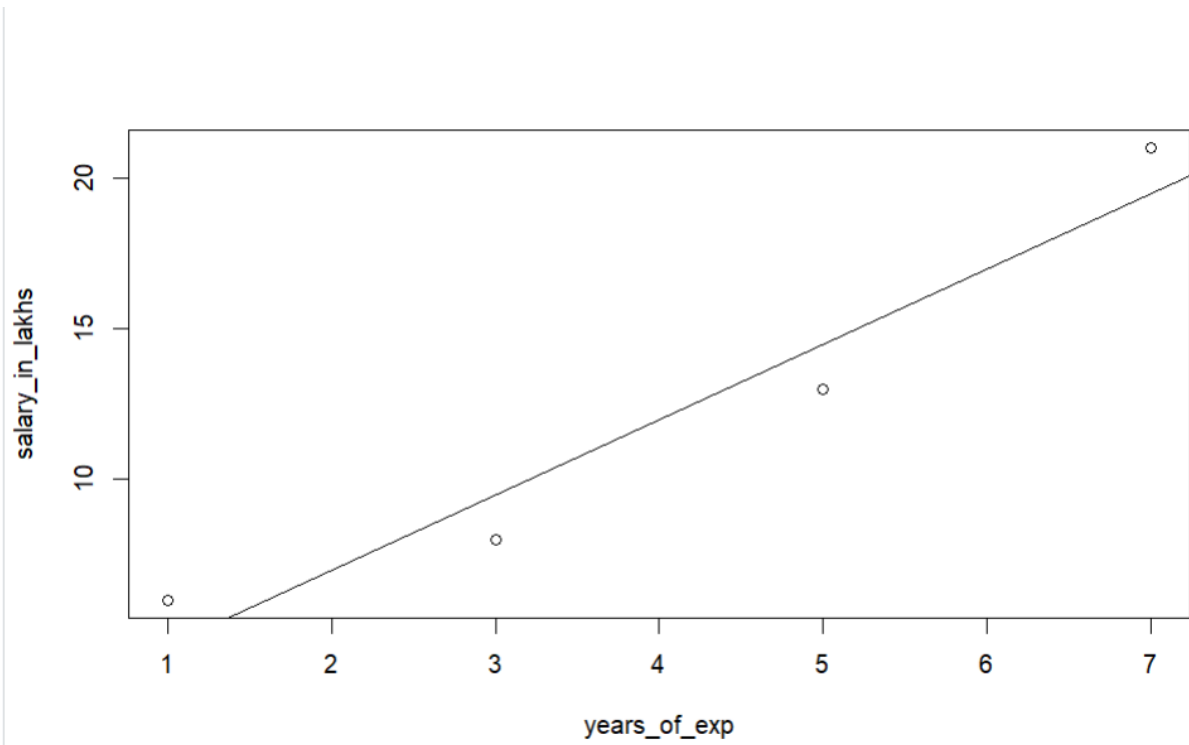
Practical 6

Aim:- Implement of Multiple Linear Regression.

```
# Importing the dataset
dataset = read.csv('C:/Users/DELL/Downloads/1000_Companies.csv')
# Encoding categorical data
dataset$State = factor(dataset$State,
                levels = c('New York', 'California', 'Florida'),
                labels = c(1, 2, 3))
dataset$State
# Splitting the dataset into the Training set and Test set
install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Profit, SplitRatio = 0.8)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
# Feature Scaling
# training_set = scale(training_set)
# test_set = scale(test_set)
# Fitting Multiple Linear Regression to the Training set
regressor = lm(formula = Profit ~ .,
          data = training_set)
# Predicting the Test set results
y_pred = predict(regressor, newdata = test_set)
regressor
```

```
> library(caTools)
> set.seed(123)
> split = sample.split(dataset$Profit, SplitRatio = 0.8)
> training_set = subset(dataset, split == TRUE)
> test_set = subset(dataset, split == FALSE)
> # Feature Scaling
> # training_set = scale(training_set)
> # test_set = scale(test_set)
> # Fitting Multiple Linear Regression to the Training set
> regressor = lm(formula = Profit ~ .,
+               data = training_set)
> # Predicting the Test set results
> y_pred = predict(regressor, newdata = test_set)
> regressor

Call:
lm(formula = Profit ~ ., data = training_set)

Coefficients:
    (Intercept)          R.D.Spend     Administration   Marketing.Spend
       -7.12e+04           5.95e-01           1.06e+00          5.61e-02
          State2             State3
       -1.66e+02          -9.73e+02

>
```

Practical 7

Aim:- Implement of Logistic regression.

install.packages("ISLR")
library(ISLR)
#load dataset
data <- ISLR::Default
print (head(ISLR::Default))
#view summary of dataset
summary(data)
#find total observations in dataset
nrow(data)
#Create Training and Test Samples
#split the dataset into a training set to train the model on and a testing set to test the model
set.seed(1)
#Use 70% of dataset as training set and remaining 30% as testing set
sample <- sample(c(TRUE, FALSE), nrow(data), replace=TRUE, prob=c(0.7,0.3))
print (sample)
train <- data[sample,]
test <- data[!sample,]
nrow(train)
nrow(test)
# Fit the Logistic Regression Model
# use the glm (general linear model) function and specify family="binomial"
#so that R fits a logistic regression model to the dataset
model <- glm(default~student+balance+income, family="binomial", data=train)
#view model summary
summary(model)
#Model Diagnostics
install.packages("InformationValue")
library(InformationValue)
predicted <- predict(model, test, type="response")
confusionMatrix(test$default, predicted)

```
> predicted <- predict(model, test, type="response")
> confusionMatrix(test$default, predicted)
    No Yes
0 2912  64
1   21  39
> plotROC(test$default, predicted)
```
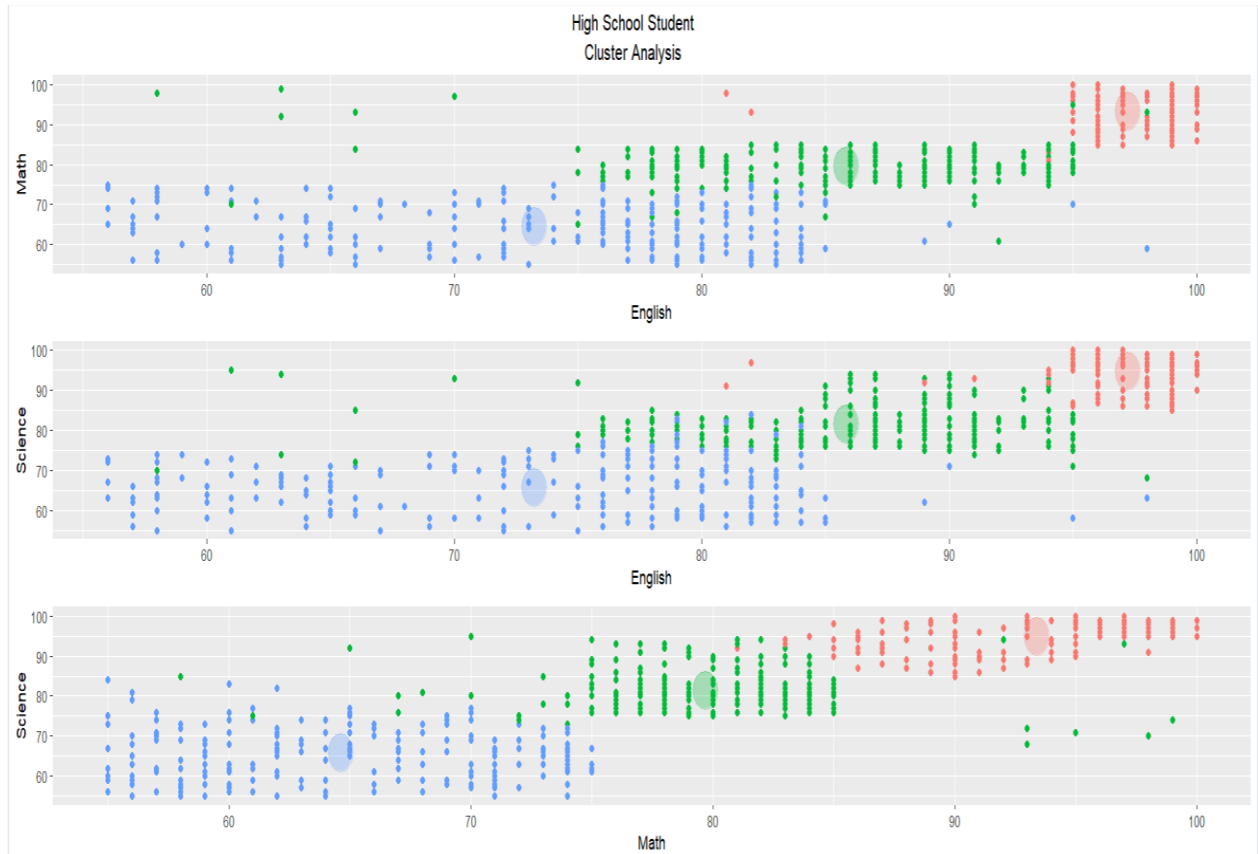
Practical 8

Aim:- Read a datafile grades_km_input.csv and apply k-means clustering.

```r
# install required packages
install.packages("plyr")
install.packages("ggplot2")
install.packages("cluster")
install.packages("lattice")
install.packages("grid")
install.packages("gridExtra")
# Load the package
library(plyr)
library(ggplot2)
library(cluster)
library(lattice)
library(grid)
library(gridExtra)
# A data frame is a two-dimensional array-like structure in which each column contains values of one
variable and each row contains one set of values from each column.
grade_input=as.data.frame(read.csv("C:/Users/DELL/Downloads/grades_km_input.csv"))
kmdata_orig=as.matrix(grade_input[, c ("Student","English","Math","Science")])
kmdata=kmdata_orig[,2:4]
kmdata[1:10,]
# the k-means algorithm is used to identify clusters for k = 1, 2, .. , 15. For each value of k, the WSS
is calculated.
wss=numeric(15)
# the option n start=25 specifies that the k-means algorithm will be repeated 25 times, each starting
with k random initial centroids
for(k in 1:15)wss[k]=sum(kmeans(kmdata,centers=k,nstart=25)$withinss)
plot(1:15,wss,type="b",xlab="Number of Clusters",ylab="Within sum of square")
#As can be seen, the WSS is greatly reduced when k increases from one to two. Another substantial
#reduction in WSS occurs at k = 3. However, the improvement in WSS is fairly linear fork > 3.
km = kmeans(kmdata,3,nstart=25)
km
c( wss[3] , sum(km$withinss))
df=as.data.frame(kmdata_orig[,2:4])
df$cluster=factor(km$cluster)
centers=as.data.frame(km$centers)
g1=ggplot(data=df, aes(x=English, y=Math, color=cluster )) +
  geom_point() + theme(legend.position="right") +
  geom_point(data=centers,aes(x=English,y=Math, color=as.factor(c(1,2,3))),size=10, alpha=.3,
        show.legend =FALSE)
g2=ggplot(data=df, aes(x=English, y=Science, color=cluster )) +
  geom_point () +geom_point(data=centers,aes(x=English,y=Science,
                        color=as.factor(c(1,2,3))),size=10, alpha=.3, show.legend=FALSE)
g3 = ggplot(data=df, aes(x=Math, y=Science, color=cluster )) +
  geom_point () + geom_point(data=centers,aes(x=Math,y=Science,
```

```
                    color=as.factor(c(1,2,3))),size=10, alpha=.3, show.legend=FALSE)
tmp=ggplot_gtable(ggplot_build(g1))
grid.arrange(arrangeGrob(g1 + theme(legend.position="none"),g2 +
                theme(legend.position="none"),g3 + theme(legend.position="none"),top ="High
School Student
Cluster Analysis" ,ncol=1))
```

Practical 9

Aim:- Perform Apriori algorithm using Groceries dataset from the R rules package.

```
install.packages("arules")
install.packages("arulesViz")
install.packages("RColorBrewer")
# Loading Libraries
library(arules)
library(arulesViz)
library(RColorBrewer)
# import dataset
data(Groceries)
Groceries
summary(Groceries)
class(Groceries)
# using apriori() function
rules = apriori(Groceries, parameter = list(supp = 0.02, conf = 0.2))
summary (rules)
# using inspect() function
inspect(rules[1:10])
# using itemFrequencyPlot() function
arules::itemFrequencyPlot(Groceries, topN = 20,
                col = brewer.pal(8, 'Pastel2'),
                main = 'Relative Item Frequency Plot',
                type = "relative",
                ylab = "Item Frequency (Relative)")
itemsets = apriori(Groceries, parameter = list(minlen=2, maxlen=2,support=0.02, target="frequent
itemsets"))
summary(itemsets)
# using inspect() function
inspect(itemsets[1:10])
itemsets_3 = apriori(Groceries, parameter = list(minlen=3, maxlen=3,support=0.02, target="frequent
itemsets"))
summary(itemsets_3)
# using inspect() function
inspect(itemsets_3)
```

```
R ▾ R 4.4.3 · ~/ ⇱
> summary(itemsets_3)
set of 2 itemsets

most frequent items:
other vegetables        whole milk  root vegetables              yogurt
              2                  2                1                   1
     frankfurter             (Other)
              0                  0

element (itemset/transaction) length distribution:sizes
3
2

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      3       3       3       3       3       3

summary of quality measures:
    support              count
 Min.   :0.0223   Min.    :219
 1st Qu.:0.0225   1st Qu.:221
 Median :0.0227   Median :224
 Mean   :0.0227   Mean    :224
 3rd Qu.:0.0230   3rd Qu.:226
 Max.   :0.0232   Max.    :228

includes transaction ID lists: FALSE

mining info:
     data ntransactions support confidence
 Groceries         9835    0.02          1
                                                                            call
 apriori(data = Groceries, parameter = list(minlen = 3, maxlen = 3, support = 0.02, target = "frequent itemsets"))
> # using inspect() function
> inspect(itemsets_3)
    items                                          support count
[1] {root vegetables, other vegetables, whole milk} 0.0232  228
[2] {other vegetables, whole milk, yogurt}          0.0223  219
```

Relative Item Frequency Plot