**Subject:- BIG DATA ANALYTICS**

1. Aim: Perform Apriori algorithm using Groceries dataset from the R arules package.

Code :-

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])

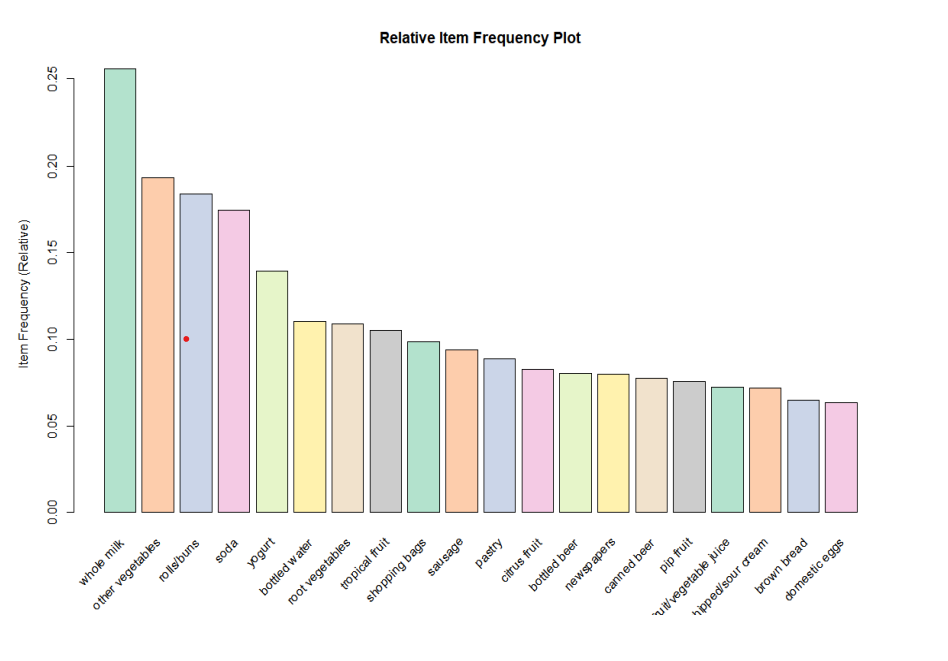
itemsets\_3 = apriori(Groceries, parameter = list(minlen=3, maxlen=3, support =0.02, target="frequent itemsets"))

summary(itemsets\_3)

# using inspect() function

inspect(itemsets\_3)

Output:



1. Aim: Write a R code for Text Analysis.

Code:

Natural Language Processing

# Importing the dataset

dataset\_original = read.delim('D:\\2020\\Big Data Analytics\\Practical\\P6 NLP\\Restaurant\_Reviews.tsv', quote = '', stringsAsFactors = FALSE)

# Cleaning the texts

install.packages('tm')

install.packages('SnowballC')

library(tm)

library(SnowballC)

corpus = VCorpus(VectorSource(dataset\_original$Review))

corpus = tm\_map(corpus, content\_transformer(tolower))

corpus = tm\_map(corpus, removeNumbers)

corpus = tm\_map(corpus, removePunctuation)

corpus = tm\_map(corpus, removeWords, stopwords())

corpus = tm\_map(corpus, stemDocument)

corpus = tm\_map(corpus, stripWhitespace)

# Creating the Bag of Words model

dtm = DocumentTermMatrix(corpus)

dtm = removeSparseTerms(dtm, 0.999)

dataset = as.data.frame(as.matrix(dtm))

dataset$Liked = dataset\_original$Liked

print(dataset$Liked)

# Encoding the target feature as factor

dataset$Liked = factor(dataset$Liked, levels = c(0, 1))

# Splitting the dataset into the Training set and Test set install.packages('caTools')

library(caTools)

set.seed(123)

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

training\_set = subset(dataset, split == TRUE)

test\_set = subset(dataset, split == FALSE)

# Fitting Random Forest Classification to the Training set install.packages('randomForest')

library(randomForest)

classifier = randomForest(x = training\_set[-692],

y = training\_set$Liked,

ntree = 10)

# Predicting the Test set results

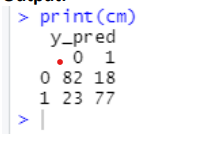
y\_pred = predict(classifier, newdata = test\_set[-692])

# Making the Confusion Matrix

cm = table(test\_set[, 692], y\_pred)

print(cm)

Output:



1. Aim: Write a R code for Naïve Bayes.

Code:

# Naive Bayes

# Importing the dataset

dataset = read.csv('D:\\2020\\Big Data Analytics\\Practical\\p4 naive bayes \\ Social\_Network\_Ads.csv')

dataset = dataset[3:5]

# Encoding the target feature as factor

dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))

# Splitting the dataset into the Training set and Test set

#install.packages('caTools')

library(caTools)

set.seed(123)

split = sample.split(dataset$Purchased, SplitRatio = 0.75)

training\_set = subset(dataset, split == TRUE)

test\_set = subset(dataset, split == FALSE)

# Feature Scaling

training\_set[-3] = scale(training\_set[-3])

test\_set[-3] = scale(test\_set[-3])

# Fitting Naive Bayes to the Training set

nstall.packages('e1071')

library(e1071)

classifier = naiveBayes(x = training\_set[-3], y = training\_set$Purchased)

# Predicting the Test set results

y\_pred = predict(classifier, newdata = test\_set[-3])

# Making the Confusion Matrix

cm = table(test\_set[, 3], y\_pred)

print(cm)

# Visualising the Training set results

install.packages("ElemStatLearn")

library(ElemStatLearn)

set = training\_set

print(set)

X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)

grid\_set = expand.grid(X1, X2)

colnames(grid\_set) = c('Age', 'EstimatedSalary')

y\_grid = predict(classifier, newdata = grid\_set)

plot(set[, -3],

main = 'Naive Bayes (Training set)',

xlab = 'Age', ylab = 'Estimated Salary',

xlim = range(X1), ylim = range(X2))

contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE) points(grid\_set, pch = '.', col = ifelse(y\_grid == 1, 'springgreen3', 'tomato'))

points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))

# Visualising the Test set results

library(ElemStatLearn)

set = test\_set

X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)

grid\_set = expand.grid(X1, X2)

colnames(grid\_set) = c('Age', 'EstimatedSalary')

y\_grid = predict(classifier, newdata = grid\_set)

plot(set[, -3], main = 'NaiveBayes (Test set)',

xlab = 'Age', ylab = 'Estimated Salary',

xlim = range(X1), ylim = range(X2))

contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE) points(grid\_set, pch = '.', col = ifelse(y\_grid == 1, 'springgreen3', 'tomato'))

points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))