Subject:- BIG DATA ANALYTICS

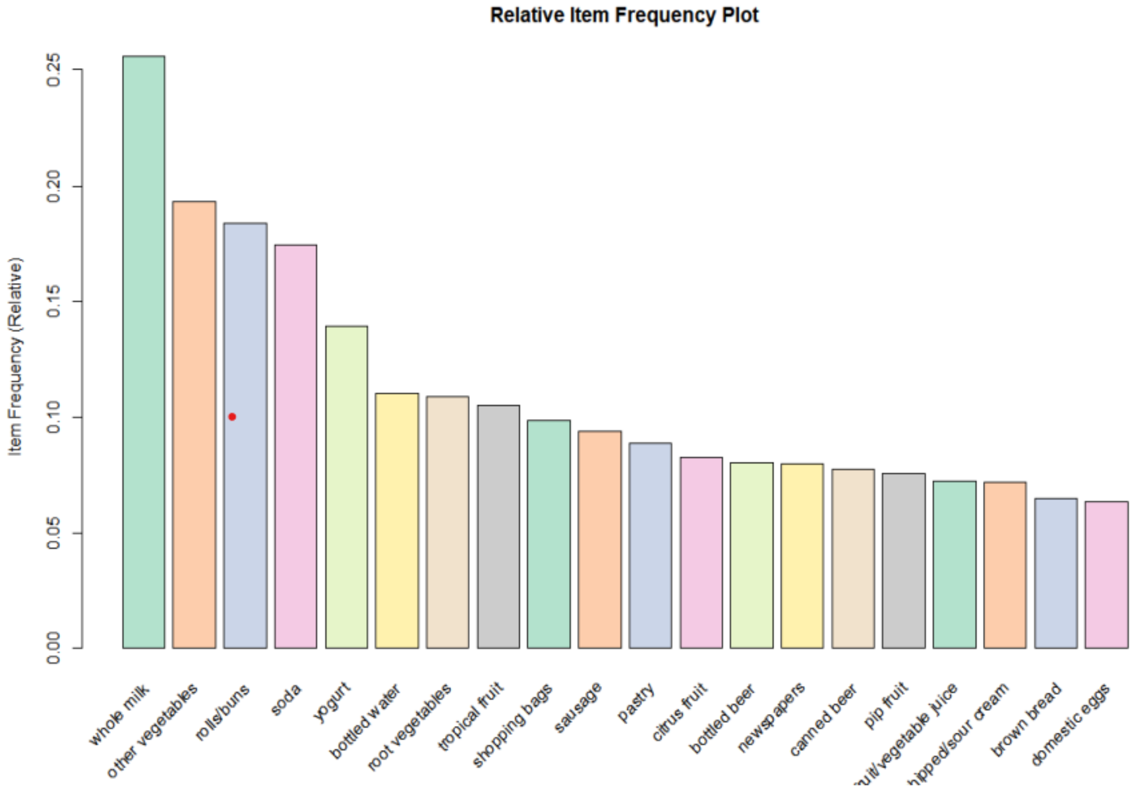
1. **Aim: Perform Apriori algorithm using Groceries dataset from the R arules package.**

   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:



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

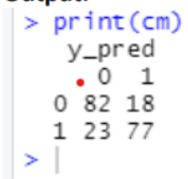
Code:

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:



```
> print(cm)
   y_pred
    0  1
0 82 18
1 23 77
>
```

3. 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'))
```