PRACTICAL

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

Code:

```
a = 5
b = 10
c = a – b
print(c)
install.packages("plyr")
install.packages("ggplot2")
install.packages("cluster")
install.packages("lattice")
install.packages("grid")
install.packages("gridExtra")

##########

library(plyr)
library(ggplot2)
library(cluster)
library(lattice)
library(grid)
library(gridExtra)
grade_input=as.data.frame(read.csv("C:\\Users\\mukad\\Downloads\\
grades_km_input.csv"))
kmdata_orig=as.matrix(grade_input[, c
("Student","English","Math","Science")])
kmdata=kmdata_orig[,2:4]
kmdata[1:10,]
```

```
wss=numeric(15)
for(k in
1:15)wss[k]=sum(kmeans(kmdata,centers=k,nstart=25)$withinss)
plot(1:15,wss,type="b",xlab="Number of Clusters",ylab="Within sum
of square")
km = kmeans(kmdata,3,nstart=25)
km
c( wss[3] , sum(km$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

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

Code:

```
install.packages("arules")
install.packages("arulesViz")
install.packages("RColorBrewer")

library(arules)
library(arulesViz)
library(RColorBrewer)
data(Groceries)
Groceries
summary(Groceries)
class(Groceries)
rules = apriori(Groceries, parameter = list(supp = 0.02, conf = 0.2))
summary (rules)
inspect(rules[1:10])
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)
inspect(itemsets[1:10])
itemsets_3 = apriori(Groceries, parameter = list(minlen=3,
maxlen=3,support=0.02, target="frequent itemsets"))
summary(itemsets_3)
inspect(itemsets_3)
```

PRACTICAL

A. Simple Linear Regression.

Code:

```
years_of_exp = c(7,5,1,3)
salary_in_lakhs = c(21,13,6,8)
employee.data = data.frame(years_of_exp, salary_in_lakhs)
employee.data
model <- lm(salary_in_lakhs ~ years_of_exp, data = employee.data)
summary(model)
plot(salary_in_lakhs ~ years_of_exp, data = employee.data)
abline(model)
```

B. Logistic Regression.

```
install.packages("ISLR")
library(ISLR)
data <- ISLR::Default
print (head(ISLR::Default))
summary(data)
nrow(data)
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)
model <- glm(default~student+balance+income, family="binomial",
data=train)
summary(model)
#install.packages("InformationValue")
library(InformationValue)
predicted <- predict(model, test, type="response")
confusionMatrix(test$default, predicted)
#install.packages("confusionMatrix")
```

PRACTICAL

Aim: DT and NBC

A. Decision Tree.

Code:

```
dataset =
read.csv("C:\\Users\\mukad\\Downloads\\Social_Network_Ads
(1).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 Decision Tree Classification to the Training set
#install.packages('rpart')
library(rpart)
classifier = rpart(formula = Purchased ~ .,
                   data = training_set)
# Predicting the Test set results
y_pred = predict(classifier, newdata = test_set[-3], type = 'class')
# Making the Confusion Matrix
cm = table(test_set[, 3], y_pred)
# Visualising the Training set results
#install.packages("ElemStatLearn")
library(ElemStatLearn)
set = training_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, type = 'class')
plot(set[, -3],
     main = 'Decision Tree Classification (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, type = 'class')
plot(set[, -3], main = 'Decision Tree Classification (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'))
# Plotting the tree
plot(classifier)
text(classifier)
```

B. Naïve Bayes Classification.

Code:

```
dataset =
read.csv("C:\\Users\\mukad\\Downloads\\Social_Network_Ads
(1).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$Purchased, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
# Feature Scaling
training_set[-3] = scale(training_set[-3])
test_set[-3] = scale(test_set[-3])
# Fitting Naive Bayes to the Training set
install.packages('e1071')
library(e1071)
classifier = naiveBayes(x = training_set[-3],
                        y = training_set$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'))
```

PRACTICAL

Aim: Text Analysis.

A. Natural Language Processing.

Code:

```
dataset_original =
read.delim("C:\\Users\\mukad\\Downloads\\Restaurant_Reviews.tsv.t
xt", 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)
```