R Code

```
### Installing packages and loading libraries


install.packages('factoextra')

library(factoextra)

library(readr)

library(NbClust)

library(cluster)


### Setting the working directory


setwd("D:/great learning/3. Data Mining/Extended project")

mydata = read.csv("PL_XSELL.csv", header = TRUE)

attach(mydata)

mydata_1 = mydata[,c(3,5,8,9,12,15,21,27,33,34,35,36,37)]

colSums(is.na(mydata))

str(mydata)

print(mydata)

summary(mydata)

plot(GENDER)

plot(ACC_TYPE)

plot(SCR, col = "Red")

plot(AVG_AMT_PER_ATM_TXN, col = "Yellow")


### Scale the data


mydata.scaled = scale(mydata_1)
```

```
print(mydata.scaled, digits = 3)

apply(mydata.scaled,2,mean)

apply(mydata.scaled,2,sd)


### kmeans clustering


seed = 1000

totWss = rep(0,10)

for(k in 1:10)

{

  set.seed(seed)

  cluster = kmeans(x = mydata.scaled, centers = k, nstart = 5)

  totWss[k] = cluster$tot.withinss

}

print(totWss)

plot(c(1:10), totWss, type = "b")


seed = 1000

set.seed(seed)

clust = kmeans(x=mydata.scaled, centers = 6, nstart = 5)

print(clust)

clusplot(mydata.scaled, clust$cluster, color = TRUE, shade = TRUE, label = 2, lines = 1)


mydata$cluster = clust$cluster

View(mydata)

customer.Profile = aggregate(mydata[,c(3,5,8,9,12,15,21,27,33,34,35,36,37)],list(mydata$cluster),
FUN = "mean")
```

```
print(customer.Profile)

### Splitting the data into train and test data set

library(caTools)
split = sample.split(mydata$TARGET, SplitRatio = 0.7)
trainDS = subset(mydata, split == TRUE)
testDS = subset(mydata, split == FALSE)

### CART analysis with train data set

trainDS_2 = trainDS[,c(2,3,5,8,9,12,15,21,27,33,34,35,36,37)]
plot(trainDS_2$AGE~., data = trainDS_2)
points([trainDS_2$TARGET=="0"])
library(rpart)
library(rpart.plot)
cart_tree = rpart(formula = trainDS_2$TARGET~., data = trainDS_2, method = "class", minbucket = 3, cp = 0)
cart_tree
rpart.plot(cart_tree)
printcp(cart_tree)
plotcp(cart_tree)

### Pruning

ptree = prune(cart_tree, cp = 0.00025, "CP")
printcp(ptree)
```

```
rpart.plot(ptree)

ptree


trainDS_2$prediction = predict(ptree,data = trainDS_2,type = "class")

trainDS_2$score = predict(ptree,data = trainDS_2,type = "prob")[,2]

head(trainDS_2)

View(trainDS_2)


### CART analysis with test data set


testDS_2 = testDS[,c(2,3,5,8,9,12,15,21,27,33,34,35,36,37)]

plot(testDS_2$AGE~., data = testDS_2)

points([testDS_2$TARGET=="0"])

library(rpart)

library(rpart.plot)

cart_tree.test = rpart(formula = testDS_2$TARGET~., data = testDS_2, method = "class", minbucket =
3, cp = 0)

cart_tree.test

rpart.plot(cart_tree.test)

printcp(cart_tree.test)

plotcp(cart_tree.test)


### Pruning


ptree.test = prune(cart_tree.test, cp = 0.0025, "CP")

printcp(ptree.test)

rpart.plot(ptree.test)
```

```
ptree.test


testDS_2$prediction = predict(ptree.test,data = testDS_2,type = "class")

testDS_2$score = predict(ptree.test,data = testDS_2,type = "prob")[,2]

head(testDS_2)

View(testDS_2)


### Random Forest analysis with train data set


trainDS$TARGET = as.factor(trainDS$TARGET)

trainDS_3 = trainDS

str(trainDS_3)

trainDS_3 = trainDS_3[,c(2,3,5,8,9,12,15,21,27,33,34,35,36,37)]

print(trainDS_3)

nrow(trainDS_3)

print(sum(trainDS$TARGET=="1")/nrow(trainDS))


library(randomForest)

seed=1000

set.seed(seed)

rndforest = randomForest(TARGET~., data = trainDS_3, ntree = 501, mtry = 3, nodesize = 10,
importance = TRUE)

print(rndforest)

print(rndforest$err.rate)

plot(rndforest)

importance(rndforest)
```

Tuning

```
set.seed(seed)
trndforest = tuneRF(x=trainDS_3[,-c(1)], y=trainDS_3$TARGET, mtryStart = 3, stepFactor = 1.5, ntreeTry = 51, improve = 0.0001, trace = TRUE, plot = TRUE, doBest = TRUE, importance = TRUE)
trainDS_3$predict.class = predict(trndforest, trainDS_3, type = "class")
trainDS_3$prob = predict(trndforest, trainDS_3, type = "prob")[,"1"]
head(train)
tbl = table(trainDS_3$TARGET, trainDS_3$predict.class)
print(tbl)
print((tbl[1,2]+tbl[2,1])/14000)
qs = quantile(trainDS_3$prob, prob = seq(0,1,length = 11))
print(qs)
threshold = qs[10]
mean(trainDS_3$TARGET[trainDS_3$prob>threshold]=="1")
```

Testing the built Random Forest Model with test data set

```
testDS$TARGET = as.factor(testDS$TARGET)
print(testDS)
nrow(testDS)
testDS_3 = testDS[,c(2,3,5,8,9,12,15,21,27,33,34,35,36,37)]
testDS_3$predict.class = predict(trndforest, testDS_3, type = "class")
testDS_3$prob = predict(trndforest, testDS_3, type = "prob")[,"1"]
head(testDS_3)
tbl = table(testDS_3$TARGET, testDS_3$predict.class)
print(tbl)
```

```
print((tbl[1,2]+tbl[2,1])/6000)

mean(testDS_3$TARGET[testDS_3$prob>threshold]=="1")


### Model Performance for CART analysis


tbl.trainDS_2 = table(trainDS_2$TARGET, trainDS_2$prediction)

print(tbl.trainDS_2)

print((tbl.trainDS_2[1,2]+tbl.trainDS_2[2,1])/14000)


tbl.testDS_2 = table(testDS_2$TARGET, testDS_2$prediction)

print(tbl.testDS_2)

print((tbl.testDS_2[1,2]+tbl.testDS_2[2,1])/6000)


### Rank ordering table


probs.trainDS_2 = seq(0,1,length = 11)

print(probs.trainDS_2)

qs.trainDS_2 = unique(quantile(trainDS_2$score, probs.trainDS_2))

print(qs.trainDS_2)

trainDS_2$decile = cut(trainDS_2$score, qs.trainDS_2, include.lowest = TRUE)

library(data.table)

trainDT = data.table(trainDS_2)

ranktbl = trainDT[,list(cnt=length(TARGET),cnt_trg1 = sum(TARGET==1), cnt_trg0 =
sum(TARGET==0)),by = decile][order(-decile)]

ranktbl$rrate = round(ranktbl$cnt_trg1/ranktbl$cnt,4)*100

ranktbl$cum_resp = cumsum(ranktbl$cnt_trg1)

ranktbl$cum_nonresp = cumsum(ranktbl$cnt_trg0)
```

```
ranktbl$cum_rel_resp = round(ranktbl$cum_resp/sum(ranktbl$cnt_trg1),4)*100

ranktbl$cum_rel_nonresp = round(ranktbl$cum_nonresp/sum(ranktbl$cnt_trg0),4)*100

ranktbl$ks = abs(ranktbl$cum_rel_resp-ranktbl$cum_rel_nonresp)


print(ranktbl)


### Rank Order Table test data set


probs.test = seq(0,1,length = 11)

print(probs.test)

qs.test = quantile(testDS_2$score, probs.test)

print(qs.test)

testDS_2$decile = cut(testDS_2$score, unique(qs.test), include.lowest = TRUE)

library(data.table)

testDT = data.table(testDS_2)

ranktbl.test = testDT[,list(cnt=length(TARGET),cnt_trg1 = sum(TARGET==1), cnt_trg0 =
sum(TARGET==0)),by = decile][order(-decile)]

ranktbl.test$rrate = round(ranktbl.test$cnt_trg1/ranktbl.test$cnt,4)*100

ranktbl.test$cum_resp = cumsum(ranktbl.test$cnt_trg1)

ranktbl.test$cum_nonresp = cumsum(ranktbl.test$cnt_trg0)


ranktbl.test$cum_rel_resp = round(ranktbl.test$cum_resp/sum(ranktbl.test$cnt_trg1),4)*100

ranktbl.test$cum_rel_nonresp = round(ranktbl.test$cum_nonresp/sum(ranktbl.test$cnt_trg0),4)*100

ranktbl.test$ks = abs(ranktbl.test$cum_rel_resp-ranktbl.test$cum_rel_nonresp)


print(ranktbl.test)
```

ROC

```
library(ROCR)
library(ineq)
predobj.trainDS_2 = prediction(trainDS_2$score, trainDS_2$TARGET)
perf.train = performance(predobj.trainDS_2, "tpr", "fpr")
predobj.test = prediction(testDS_2$score, testDS_2$TARGET)
perf.test = performance(predobj.test, "tpr", "fpr")
plot(perf.train)
plot(perf.test)
```

KS

```
print(perf.train@y.values[[1]]-perf.train@x.values[[1]])
KS.train = max(perf.train@y.values[[1]]-perf.train@x.values[[1]])
print(KS.train)

print(perf.test@y.values[[1]]-perf.test@x.values[[1]])
KS.test = max(perf.test@y.values[[1]]-perf.test@x.values[[1]])
print(KS.test)
```

auc

```
auc.train = performance(predobj.trainDS_2,"auc")
auc.train = as.numeric(auc.train@y.values)
print(auc.train)
```

```
auc.test = performance(predobj.test,"auc")
auc.test = as.numeric(auc.test@y.values)
print(auc.test)

### GINI

gini = ineq(trainDS_2$score, "gini")
gini1 = ineq(testDS_2$score, "gini")
print(gini)
print(gini1)

### Concordance and Discordance

library(InformationValue)
Concordance(actuals = trainDS_2$TARGET, predictedScores = trainDS_2$score)
Concordance(actuals = testDS_2$TARGET, predictedScores = testDS_2$score)

### Model performance for Random Forest

tbl.trainDS_3 = table(trainDS_3$TARGET, trainDS_3$predict.class)
print(tbl.trainDS_3)
print((tbl.trainDS_3[1,2]+tbl.trainDS_3[2,1])/14000)

tbl.testDS_3 = table(testDS_3$TARGET, testDS_3$predict.class)
print(tbl.testDS_3)
print((tbl.testDS_3[1,2]+tbl.testDS_3[2,1])/6000)
```

Rank ordering table

```
probs.trainDS_3 = seq(0,1,length = 11)

print(probs.trainDS_3)

qs.trainDS_3 = unique(quantile(trainDS_3$prob, probs.trainDS_3))

print(qs.trainDS_3)

trainDS_3$decile = cut(trainDS_3$prob, qs.trainDS_3, include.lowest = TRUE)

library(data.table)

trainDT_rf = data.table(trainDS_3)

ranktbl_rf = trainDT_rf[,list(cnt=length(TARGET),cnt_trg1 = sum(TARGET==1), cnt_trg0 =
sum(TARGET==0)),by = decile][order(-decile)]

ranktbl_rf$rrate = round(ranktbl_rf$cnt_trg1/ranktbl_rf$cnt,4)*100

ranktbl_rf$cum_resp = cumsum(ranktbl_rf$cnt_trg1)

ranktbl_rf$cum_nonresp = cumsum(ranktbl_rf$cnt_trg0)


ranktbl_rf$cum_rel_resp = round(ranktbl_rf$cum_resp/sum(ranktbl_rf$cnt_trg1),4)*100

ranktbl_rf$cum_rel_nonresp = round(ranktbl_rf$cum_nonresp/sum(ranktbl_rf$cnt_trg0),4)*100

ranktbl_rf$ks = abs(ranktbl_rf$cum_rel_resp-ranktbl_rf$cum_rel_nonresp)


print(ranktbl_rf)
```

Rank Order Table test data set

```
probs.testDS_3 = seq(0,1,length = 11)

print(probs.testDS_3)

qs.testDS_3 = quantile(testDS_3$prob, probs.testDS_3)

print(qs.testDS_3)
```

```
testDS_3$decile = cut(testDS_3$prob, unique(qs), include.lowest = TRUE)

library(data.table)

testDT_rf.test = data.table(testDS_3)

ranktbl_rf.test = testDT_rf.test[,list(cnt=length(TARGET),cnt_trg1 = sum(TARGET==1), cnt_trg0 =
sum(TARGET==0)),by = decile][order(-decile)]

ranktbl_rf.test$rrate = round(ranktbl_rf.test$cnt_trg1/ranktbl_rf.test$cnt,4)*100

ranktbl_rf.test$cum_resp = cumsum(ranktbl_rf.test$cnt_trg1)

ranktbl_rf.test$cum_nonresp = cumsum(ranktbl_rf.test$cnt_trg0)


ranktbl_rf.test$cum_rel_resp = round(ranktbl_rf.test$cum_resp/sum(ranktbl_rf.test$cnt_trg1),4)*100

ranktbl_rf.test$cum_rel_nonresp =
round(ranktbl_rf.test$cum_nonresp/sum(ranktbl_rf.test$cnt_trg0),4)*100

ranktbl_rf.test$ks = abs(ranktbl_rf.test$cum_rel_resp-ranktbl_rf.test$cum_rel_nonresp)


print(ranktbl_rf.test)


### ROC


library(ROCR)

library(ineq)

predobj.trainDS_3 = prediction(trainDS_3$prob, trainDS_3$TARGET)

perf.train_rf_3 = performance(predobj.trainDS_3, "tpr", "fpr")

predobj.testDS_3 = prediction(testDS_3$prob, testDS_3$TARGET)

perf.test_rf_3 = performance(predobj.testDS_3, "tpr", "fpr")

plot(perf.train_rf_3)

plot(perf.test_rf_3)
```

```
### KS

print(perf.train_rf_3@y.values[[1]]-perf.train_rf_3@x.values[[1]])
KS.train3 = max(perf.train_rf_3@y.values[[1]]-perf.train_rf_3@x.values[[1]])
print(KS.train3)

print(perf.test_rf_3@y.values[[1]]-perf.test_rf_3@x.values[[1]])
KS.test3 = max(perf.test_rf_3@y.values[[1]]-perf.test_rf_3@x.values[[1]])
print(KS.test3)

### auc

auc.train3 = performance(predobj.trainDS_3,"auc")
auc.train3 = as.numeric(auc.train3@y.values)
print(auc.train3)

auc.test3 = performance(predobj.testDS_3,"auc")
auc.test3 = as.numeric(auc.test3@y.values)
print(auc.test3)

### GINI

gini2 = ineq(trainDS_3$score, "gini")
gini3 = ineq(testDS_3$score, "gini")
print(gini2)
print(gini3)
```

Concordance and Discordance

```
library(InformationValue)

Concordance(actuals = trainDS_3$TARGET, predictedScores = trainDS_3$prob)

Concordance(actuals = testDS_3$TARGET, predictedScores = testDS_3$prob)
```