**#Random Forest**

dataset = read.csv('Churn\_Modelling.csv')

dataset = dataset[c(4,7,8,9,10,11,12,13,14)]

library(randomForest)

dataset$Exited <- factor(dataset$Exited)

set.seed(50)

trainIndex <-createDataPartition(dataset$Exited, p=0.8, list = FALSE, times=1 )

trainset <- dataset[trainIndex,]

testset <- dataset[-trainIndex,]

# Feature Scaling

trainset[-9] = scale(trainset[-9])

testset[-9] = scale(testset[-9])

Ran<-randomForest(x = trainset[-9],

y = trainset$Exited,

ntree=10)

varImpPlot(Ran)

forest\_predict<-predict(Ran, testset)

forest\_predict

table(forest\_predict, testset$Exited)

#accruacy is 85.3%

**#Random Forest after using PCA**

dataset = read.csv('Churn\_Modelling.csv')

dataset = dataset[c(4,7,8,9,10,11,12,13,14)]

library(randomForest)

dataset$Exited <- factor(dataset$Exited)

set.seed(50)

trainIndex <-createDataPartition(dataset$Exited, p=0.8, list = FALSE, times=1 )

trainset <- dataset[trainIndex,]

testset <- dataset[-trainIndex,]

# Feature Scaling

trainset[-9] = scale(trainset[-9])

testset[-9] = scale(testset[-9])

# Applying PCA

# install.packages('caret')

library(caret)

# install.packages('e1071')

library(e1071)

pca = preProcess(x = trainset[-9], method = 'pca', pcaComp = 2)

trainset = predict(pca, trainset)

trainset = trainset[c(2, 3, 1)]

testset = predict(pca, testset)

testset = testset[c(2, 3, 1)]

Ran<-randomForest(x = trainset[-9],

y = trainset$Exited,

ntree=10)

# Predicting the Test set results

y\_pred = predict(Ran, newdata = testset[-9])

# Making the Confusion Matrix

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

cm

y\_pred

0 1

0 1592 0

1. 0 407

#The accuracy is 100%

#Decision Tree

library(rpart)

library(rpart.plot)

tree<-rpart(Exited~.,trainset,method="class")

rpart.plot(tree,type=4,cex=0.6)

A close up of a map

Description automatically generated

#After prune

printcp(tree)

ptree<-prune(tree,cp=tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"])

rpart.plot(ptree,type=4,cex=0.5)

A close up of a map

Description automatically generated

predict\_tree<-predict(ptree,testset,type="class")

table(testset$Exited,predict\_tree)

predict\_tree

0 1

0 1518 74

1 212 195

#accruacy is 85.8%

#KNN

# Feature Scaling

trainset[-9] = scale(trainset[-9])

testset[-9] = scale(testset[-9])

# Applying PCA

# install.packages('caret')

library(caret)

# install.packages('e1071')

library(e1071)

pca = preProcess(x = trainset[-9], method = 'pca', pcaComp = 2)

trainset = predict(pca, trainset)

trainset = trainset[c(2, 3, 1)]

testset = predict(pca, testset)

testset = testset[c(2, 3, 1)]

fit <- train(trainset[,-9], trainset$Exited,

preProcess = c("center","scale"),

method = "knn")

plot(fit)

A close up of a map

Description automatically generated

predictknn<-predict(fit,testset)

predictknn

KNNtest<-cbind(testset,predictknn)

cm = table(KNNtest$Exited, KNNtest$predictknn)

cm

0 1

0 1592 0

1 0 407

#accruacy is (1592 + 407)/1999 =100%

#svm

s1<-svm(Exited~., data=trainset)

PredictSVM<-predict(s1, newdata=testset, type="response");

cm = table(PredictSVM, testset$Exited)

cm

PredictSVM 0 1

0 1558 259

1 34 148

#The accruacy is 85.3%