Installing packages and loading library

```
library(readxl)
```

Setting up the working directory

```
setwd("D:/great learning/4. Predictive Modeling/Project-4")
mydata = read_xlsx("Cellphone.xlsx", sheet = "Data")
View(mydata)
str(mydata)
summary(mydata)
```

Check for multi co-linearity

```
library(corrplot)
mydata_1 = mydata[-c(1)]
mydata_1
mydata_corr = cor(mydata)
corrplot(mydata_corr, method = "number")
plot(mydata$DataPlan~mydata$DataUsage, col = "blue")
plot(mydata$DataPlan~mydata$MonthlyCharge, col = "green")
plot(mydata$MonthlyCharge~mydata$DataUsage, col = "red")
plot(mydata$Churn~mydata$DataUsage, col = "orange")
plot(mydata$Churn~mydata$MonthlyCharge)
```

Treatment of multi co-linearity (Factor analysis)

```
#install.packages("nFactors")

library(nFactors)

mydata1 = mydata[c(4,5,9)]          # Considering only correlated variables DataPlan, DataUsage, MonthlyCharge

names(mydata1)

mydata1corr = cor(mydata1)

mydata1corr

ev = eigen(cor(mydata1))

ev

EigenValue = ev$values

EigenValue

solution = fa(r = mydata1, nfactors = 1, rotate = "none", fm = "pa")

solution

solution1 = fa(r = mydata1, nfactors = 1, rotate = "varimax", fm = "pa")

solution1

solution1$scores


### Binding factor with other attributes


mydata_bind = cbind(mydata[c(1,2,3,6,7,8,10,11)],solution1$scores)

names(mydata_bind) = c("Churn","AccountWeeks", "ContractRenewal", "CustServCalls", "DayMins", "DayCalls", "OverageFee", "RoamMins", "ServiceUsage")

names(mydata_bind)

attach(mydata_bind)


### Building Logistic Regression Model


# Splitting the PCA/FA treatement data into train data set and test data set
```

```
library(caTools)

Sample = sample.split(mydata_bind,SplitRatio = 0.7)

train_mydata_bind = subset(mydata_bind, Sample ==T)

test_mydata_bind = subset(mydata_bind, Sample ==F)

dim(train_mydata_bind)

dim(test_mydata_bind)


# Step-1 Log likelyhood ratio test


install.packages("lmtest")

library(lmtest)

logit =
glm(Churn~AccountWeeks+ContractRenewal+CustServCalls+DayMins+DayCalls+OverageFee+Ro
amMins+ServiceUsage, data = train_mydata_bind, family = binomial)

lrtest(logit)


# Step-2 Mcfaden R square computation


install.packages("pscl")

library(pscl)

pR2(logit)


# Step-3 Test of individual coefficient


summary(logit)


# Step-4 Odds ration
```

```
odds = exp(coef(logit))

odds

Probability = odds/(1+odds)

Probability


# Step-5 Confusion matrix for measurement of predictive accuracy


predict(logit, type = "response")

pred = fitted(logit)

data.frame(train_mydata_bind$Churn, pred)

gg1 = floor(pred + 0.5)

gg1

table(Actual = train_mydata_bind$Churn, prediction = gg1)


# Step-6 ROC plot


#install.packages("Deducer")

#library(Deducer)

#rocplot(logit)

library(ROCR)

pred1 = prediction(gg1, train_mydata_bind$Churn)

pred1

perf1 = performance(pred1, "tpr", "fpr")

#plot(perf1)

plot(perf1, col = "Blue", main = "ROC Plot")

abline(0, 1, lty =8, col = "red")
```

```
auc = performance(pred1, "auc")
auc = auc@y.values
auc

### Building KNN model

# Exploratory data analysis

norm = function(x){(x - min(x))/(max(x) - min(x))}
norm.data = as.data.frame(lapply(mydata_bind[,-1], norm))
View(data)
View(norm.data)

# Data partitioning

usable.data = cbind(Churn,norm.data)
names(usable.data)
View(usable.data)
Sample1 = sample.split(usable.data,SplitRatio = 0.7)
train = subset(usable.data, Sample1 == T)
test = subset(usable.data, Sample1 == F)
dim(train)
dim(test)

### Use KNN classifier

library(class)
```

```
pred1 = knn(train[,-1],test[,-1],train[,1], k=19)
table.knn = table(test[,1],pred1)
table.knn
sum(diag(table.knn))/sum(table.knn)
pred1 = knn(train[,-1],test[,-1],train[,1], k=17)
table.knn = table(test[,1],pred1)
table.knn
sum(diag(table.knn))/sum(table.knn)
pred1 = knn(train[,-1],test[,-1],train[,1], k=15)
table.knn = table(test[,1],pred1)
table.knn
sum(diag(table.knn))/sum(table.knn)
pred1 = knn(train[,-1],test[,-1],train[,1], k=13)
table.knn = table(test[,1],pred1)
table.knn
sum(diag(table.knn))/sum(table.knn)
pred1 = knn(train[,-1],test[,-1],train[,1], k=11)
table.knn = table(test[,1],pred1)
table.knn
sum(diag(table.knn))/sum(table.knn)
pred1 = knn(train[,-1],test[,-1],train[,1], k=9)
table.knn = table(test[,1],pred1)
table.knn
sum(diag(table.knn))/sum(table.knn)
pred1 = knn(train[,-1],test[,-1],train[,1], k=7)
table.knn = table(test[,1],pred1)
table.knn
```

```
sum(diag(table.knn))/sum(table.knn)
pred1 = knn(train[,-1],test[,-1],train[,1], k=5)
table.knn = table(test[,1],pred1)
table.knn
sum(diag(table.knn))/sum(table.knn)
pred1 = knn(train[,-1],test[,-1],train[,1], k=3)
table.knn = table(test[,1],pred1)
table.knn
sum(diag(table.knn))/sum(table.knn)
pred1 = knn(train[,-1],test[,-1],train[,1], k=1)
table.knn = table(test[,1],pred1)
table.knn

### Building Naive Bayes Model
# Naive Bayes Classifier

library(e1071)
View(train)
NB = naiveBayes(Churn~., data = train)
NB
predNB = predict(NB,test,type = "class")
predNB
table.NB = table(test[,1],predNB)
table.NB
sum(diag(table.NB))/sum(table.NB)

### Fold validation and model comparison
```

```
install.packages("caret")
install.packages("klaR")
library(caret)
library(klaR)

### Folds are created on the basis of target variable

folds = createFolds(factor(train$Churn, K=10, list = FALSE))
folds

### Define train control for k fold cross validation

train_control = trainControl(method = "cv", number = 10)

### Fit KNN

modelKNN = train('mydata_bind[,1]'~., data = train, trcontrol = train_control, method = "knn")
summary(modelKNN)

### Fit Naive Bayes Model

modelNB = train('mydata_bind[,1]'~., data = train, trcontrol = train_control, method = "nb")
summary(modelNB)

### Collect resamples
```

```
results = resamples(list(KNN = modelKNN, NB = modelNB))

summary(results)

bwplot(results)
```