COMP 6115 KDDA1 - Final Project

Business Understanding

Customer churn is a major issue and is the single most important concern of medium and large Telecommunication companies. With data being dubbed, “The Gold of the Twenty-first Century” and the driving force behind the success of many data-driven companies; Telecommunication giants are now more than ever, turning to Data Science to help solve their business problems(e.g. Churning) and thus maximize profits.

Business Objectives

TeleCom, a telecommunications company located in the Kingston who specializes in offering call and messaging services to its clients, have noticed that several of its customers each month have stopped using their services, thus have churned. TeleCom is deeply concerned about the loss of its customers and declines in its profits, especially in light of the current global pandemic. The company aims, through a machine learning model, to identify customers who are likely to churn and offer them special deals and lower call and messaging rates, in an effort to stop at risk customers from churning.

Data Mining Goals

In light of TeleCom’s business problem, the goal of this data mining project, is to predict which of its customers will churn, given information about their previous service plans, call and messaging history. Success of the data mining project is geared towards creating a machine learning, classification model that is able to correctly predict which customers will churn with high accuracy, sensitivity, specificity, precision and good simplicity, AUC and stability.

Complete Project Plan

In order to achieve the intended data mining goals and thereby achieving the TeleCom’s business goals, a business plan is established. This entails, acquiring the company’s database used to store data about the customers’ service plans, message and call history, performing data exploration and cleaning, splitting dataset into training and test sets, creating classification models and finally evaluating their performance, to select the best model.

Data Understanding

Data Collection

The dataset used for this data mining project was, telecom_churn.csv, taken from Kaggle, an online repository for datasets and code documentation.

Data Dictionary

The dataset mostly contained columns related service usage by customers; call minutes, both local and international and call charges. The target variable field is 'Churn', with two classes, False or True, indicating which customer has churned. Other fields included are State, Area code and Account length. Based on the business understanding of the data, 19 columns were chosen to build the machine learning models.

Number	Variables	Description
1	Account.length	Length of time customer has had an account
2	Area.code	Area code of each customer
3	International.plan	Whether or not customer has an international plan
4	Voice.mail.plan	Whether or not customer has a voice mail plan
5	Number.vmail.messages	Number of voicemail messages a customer has
6	Total.day.minutes	Total number of day call minutes customer used
7	Total.day.calls	Total number of day calls made by customer
8	Total.day.charge	Total amount customer is charged for day usage of service
9	Total.eve.minutes	Total number of evening call minutes customer used
10	Total.eve.calls	Total number of evening call minutes customer used
11	Total.eve.charge	Total amount customer is charged for evening usage of service
12	Total.night.minutes	Total number of evening call minutes customer used
13	Total.night.calls	Total number of night calls made by customer
14	Total.night.charge	Total amount customer is charged for night usage of service
15	Total.intl.minutes	Total number of international call minutes customer used
16	Total.intl.calls	Total number of international calls made by customer
17	Total.intl.charge	Total amount customer is charged for international service use
19	Customer.service.calls	Number of calls made to the company's customer service
20	Churn	Classification whether or not customer have churned

First we want to load all the required packages.

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(caTools)
library(ggplot2)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
```

```
##
##     cov, smooth, var
```

```r
library(rpart)
library(rpart.plot)
```

We now want to load in the data set.

```r
data <- read.csv(file.choose())
```

We want to get some understanding of the data by looking at some statistics.

```r
#Show the dimemsion of the data
dim(data)
```

```
## [1] 3333   20
```

```r
#Shows summary statistics for the data
summary(data)
```

```
##     State           Account.length    Area.code     International.plan
##  Length:3333        Min.   :  1.0   Min.   :408.0   Length:3333       
##  Class :character   1st Qu.: 74.0   1st Qu.:408.0   Class :character  
##  Mode  :character   Median :101.0   Median :415.0   Mode  :character  
##                     Mean   :101.1   Mean   :437.2                     
##                     3rd Qu.:127.0   3rd Qu.:510.0                     
##                     Max.   :243.0   Max.   :510.0                     
##  Voice.mail.plan    Number.vmail.messages Total.day.minutes Total.day.calls
##  Length:3333        Min.   : 0.000        Min.   :  0.0     Min.   :  0.0  
##  Class :character   1st Qu.: 0.000        1st Qu.:143.7     1st Qu.: 87.0  
##  Mode  :character   Median : 0.000        Median :179.4     Median :101.0  
##                     Mean   : 8.099        Mean   :179.8     Mean   :100.4  
##                     3rd Qu.:20.000        3rd Qu.:216.4     3rd Qu.:114.0  
##                     Max.   :51.000        Max.   :350.8     Max.   :165.0  
##  Total.day.charge Total.eve.minutes Total.eve.calls Total.eve.charge
##  Min.   : 0.00    Min.   :  0.0     Min.   :  0.0   Min.   : 0.00   
##  1st Qu.:24.43    1st Qu.:166.6     1st Qu.: 87.0   1st Qu.:14.16   
##  Median :30.50    Median :201.4     Median :100.0   Median :17.12   
##  Mean   :30.56    Mean   :201.0     Mean   :100.1   Mean   :17.08   
##  3rd Qu.:36.79    3rd Qu.:235.3     3rd Qu.:114.0   3rd Qu.:20.00   
##  Max.   :59.64    Max.   :363.7     Max.   :170.0   Max.   :30.91   
##  Total.night.minutes Total.night.calls Total.night.charge Total.intl.minutes
##  Min.   : 23.2       Min.   : 33.0     Min.   : 1.040     Min.   : 0.00     
##  1st Qu.:167.0       1st Qu.: 87.0     1st Qu.: 7.520     1st Qu.: 8.50     
##  Median :201.2       Median :100.0     Median : 9.050     Median :10.30     
##  Mean   :200.9       Mean   :100.1     Mean   : 9.039     Mean   :10.24     
##  3rd Qu.:235.3       3rd Qu.:113.0     3rd Qu.:10.590     3rd Qu.:12.10     
##  Max.   :395.0       Max.   :175.0     Max.   :17.770     Max.   :20.00     
##  Total.intl.calls Total.intl.charge Customer.service.calls    Churn          
##  Min.   : 0.000   Min.   :0.000     Min.   :0.000          Length:3333       
##  1st Qu.: 3.000   1st Qu.:2.300     1st Qu.:1.000          Class :character  
##  Median : 4.000   Median :2.780     Median :1.000          Mode  :character  
##  Mean   : 4.479   Mean   :2.765     Mean   :1.563
```

```
##  3rd Qu.: 6.000   3rd Qu.:3.270     3rd Qu.:2.000
##  Max.   :20.000   Max.   :5.400     Max.   :9.000
```

```
#show the structure of the data
str(data)
```

```
## 'data.frame':    3333 obs. of  20 variables:
##  $ State                 : chr  "KS" "OH" "NJ" "OH" ...
##  $ Account.length        : int  128 107 137 84 75 118 121 147 117 141 ...
##  $ Area.code             : int  415 415 415 408 415 510 510 415 408 415 ...
##  $ International.plan    : chr  "No" "No" "No" "Yes" ...
##  $ Voice.mail.plan       : chr  "Yes" "Yes" "No" "No" ...
##  $ Number.vmail.messages : int  25 26 0 0 0 0 24 0 0 37 ...
##  $ Total.day.minutes     : num  265 162 243 299 167 ...
##  $ Total.day.calls       : int  110 123 114 71 113 98 88 79 97 84 ...
##  $ Total.day.charge      : num  45.1 27.5 41.4 50.9 28.3 ...
##  $ Total.eve.minutes     : num  197.4 195.5 121.2 61.9 148.3 ...
##  $ Total.eve.calls       : int  99 103 110 88 122 101 108 94 80 111 ...
##  $ Total.eve.charge      : num  16.78 16.62 10.3 5.26 12.61 ...
##  $ Total.night.minutes   : num  245 254 163 197 187 ...
##  $ Total.night.calls     : int  91 103 104 89 121 118 118 96 90 97 ...
##  $ Total.night.charge    : num  11.01 11.45 7.32 8.86 8.41 ...
##  $ Total.intl.minutes    : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...
##  $ Total.intl.calls      : int  3 3 5 7 3 6 7 6 4 5 ...
##  $ Total.intl.charge     : num  2.7 3.7 3.29 1.78 2.73 1.7 2.03 1.92 2.35 3.02 ...
##  $ Customer.service.calls: int  1 1 0 2 3 0 3 0 1 0 ...
##  $ Churn                 : chr  "False" "False" "False" "False" ...
```

```
# Shows the first 5 rows of data
head(data)
```

```
##   State Account.length Area.code International.plan Voice.mail.plan
## 1    KS            128       415                 No             Yes
## 2    OH            107       415                 No             Yes
## 3    NJ            137       415                 No              No
## 4    OH             84       408                Yes              No
## 5    OK             75       415                Yes              No
## 6    AL            118       510                Yes              No
##   Number.vmail.messages Total.day.minutes Total.day.calls Total.day.charge
## 1                    25             265.1             110            45.07
## 2                    26             161.6             123            27.47
## 3                     0             243.4             114            41.38
## 4                     0             299.4              71            50.90
## 5                     0             166.7             113            28.34
## 6                     0             223.4              98            37.98
##   Total.eve.minutes Total.eve.calls Total.eve.charge Total.night.minutes
## 1             197.4              99            16.78               244.7
## 2             195.5             103            16.62               254.4
## 3             121.2             110            10.30               162.6
## 4              61.9              88             5.26               196.9
## 5             148.3             122            12.61               186.9
## 6             220.6             101            18.75               203.9
##   Total.night.calls Total.night.charge Total.intl.minutes Total.intl.calls
## 1                91              11.01               10.0                3
## 2               103              11.45               13.7                3
## 3               104               7.32               12.2                5
```

```
## 4              89             8.86             6.6              7
## 5             121             8.41            10.1              3
## 6             118             9.18             6.3              6
##   Total.intl.charge Customer.service.calls Churn
## 1              2.70                      1 False
## 2              3.70                      1 False
## 3              3.29                      0 False
## 4              1.78                      2 False
## 5              2.73                      3 False
## 6              1.70                      0 False
```

```
# Displayed the number of missing values in the data set
sum(is.na(data))
```

```
## [1] 0
```

Data Preparation

In the data exploration phase of the data mining process, distribution of key attributes such as the target variable, Churn were visualized, simple queries were done to get a closer on the structure and form of the dataset, which included head, tail, summary, str, View and simple statistical analyses. Also, relationships between pairs of predictor variables and properties of significant sub-populations were explored.

We note the need for some data pre-processing on the data set. We will convert the attributes `International.plan`, `Voice.mail.plan` and `Churn` from ordinal data to numerical data types.

```
data$International.plan<-ifelse(data$International.plan=='Yes',1,0)
data$Voice.mail.plan<-ifelse(data$Voice.mail.plan=='Yes',1,0)
data$Churn<-ifelse(data$Churn=='True',1,0)
```

We will also need to classify the Churn variable as factor.

```
data$Churn <- as.factor(data$Churn)
```

We will also remove the the `State` attribute as it is not needed to create our model.

```
data$State <- NULL
```

Again looking at the structure of the the data set.

```
str(data)
```

```
## 'data.frame':    3333 obs. of  19 variables:
##  $ Account.length        : int  128 107 137 84 75 118 121 147 117 141 ...
##  $ Area.code             : int  415 415 415 408 415 510 510 415 408 415 ...
##  $ International.plan    : num  0 0 0 1 1 1 0 1 0 1 ...
##  $ Voice.mail.plan       : num  1 1 0 0 0 0 1 0 0 1 ...
##  $ Number.vmail.messages : int  25 26 0 0 0 0 24 0 0 37 ...
##  $ Total.day.minutes     : num  265 162 243 299 167 ...
##  $ Total.day.calls       : int  110 123 114 71 113 98 88 79 97 84 ...
##  $ Total.day.charge      : num  45.1 27.5 41.4 50.9 28.3 ...
##  $ Total.eve.minutes     : num  197.4 195.5 121.2 61.9 148.3 ...
##  $ Total.eve.calls       : int  99 103 110 88 122 101 108 94 80 111 ...
##  $ Total.eve.charge      : num  16.78 16.62 10.3 5.26 12.61 ...
##  $ Total.night.minutes   : num  245 254 163 197 187 ...
##  $ Total.night.calls     : int  91 103 104 89 121 118 118 96 90 97 ...
##  $ Total.night.charge    : num  11.01 11.45 7.32 8.86 8.41 ...
##  $ Total.intl.minutes    : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...
##  $ Total.intl.calls      : int  3 3 5 7 3 6 7 6 4 5 ...
##  $ Total.intl.charge     : num  2.7 3.7 3.29 1.78 2.73 1.7 2.03 1.92 2.35 3.02 ...
##  $ Customer.service.calls: int  1 1 0 2 3 0 3 0 1 0 ...
##  $ Churn                 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

Visualizing the the distribution of the target class give the following graph.

```
barplot(table(data$Churn), ylab ="Frequency",
        main = "Distribution of Target Class",
        col="lightblue")
```

From the visualization of the `Churn` target variable above, it was clearly demonstrated that there existed a class imbalance in this variable. Therefore during the stratified sampling phase of the model construction, the observations will be sampled with approximately equal proportions to achieve a better model.

After thorough examination of the dataset, the quality of the data was deemed to be excellent. This can be attributed to the data being complete. This means that it covered all the cases required. The data was correct, free of errors and no missing values were detected in the dataset.

Data Modelling

In this stage we will train four (4) models to determine which one of them provides the most accurate prediction. Here we will use two (2) logistic regression models and two (2) decision tree models.

Splitting the Data

Before we create our models, we first need to split the data into a training set and a testing set. The training set will be used to train the model and define the optimal parameters to be used to create the models. The test data is needed to evaluate the accuracy of the trained model.

Here we will use a 75/25 split on the data.

```
set.seed(1670)
new.data <- sample.split(Y = data$Churn, SplitRatio = 0.75)
train.data <- data[new.data,]
test.data <- data[!new.data,]

print(paste('The dimension of the training data is', dim(train.data)[1], 'rows and',dim(train.data)[2],
```

```
## [1] "The dimension of the training data is 2500 rows and 19 attributes"
```

```
print(paste('The dimension of the test data is', dim(test.data)[1], 'rows and',dim(test.data)[2], 'attr
```

```
## [1] "The dimension of the test data is 833 rows and 19 attributes"
```

Model 1

We will create the first model using logistic regression utilizing all the attributes.

```
model.1 <- glm(Churn ~ .,
               data=train.data,
               family=binomial(link="logit"))

summary(model.1)
```

```
##
## Call:
## glm(formula = Churn ~ ., family = binomial(link = "logit"), data = train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0175  -0.5205  -0.3519  -0.2110   3.0625
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -7.752e+00  1.049e+00  -7.391 1.46e-13 ***
## Account.length          4.652e-04  1.599e-03   0.291  0.77107
## Area.code              -1.036e-03  1.509e-03  -0.687  0.49223
## International.plan      2.109e+00  1.656e-01  12.739  < 2e-16 ***
## Voice.mail.plan        -1.497e+00  6.277e-01  -2.384  0.01712 *
## Number.vmail.messages   1.994e-02  2.000e-02   0.997  0.31879
## Total.day.minutes       9.898e-01  3.747e+00   0.264  0.79165
## Total.day.calls         2.916e-03  3.175e-03   0.918  0.35839
## Total.day.charge       -5.752e+00  2.204e+01  -0.261  0.79412
## Total.eve.minutes       7.073e-01  1.883e+00   0.376  0.70725
## Total.eve.calls        -8.358e-05  3.194e-03  -0.026  0.97912
## Total.eve.charge       -8.242e+00  2.216e+01  -0.372  0.70991
## Total.night.minutes    -1.053e+00  1.005e+00  -1.048  0.29470
## Total.night.calls       1.324e-03  3.270e-03   0.405  0.68560
## Total.night.charge      2.348e+01  2.233e+01   1.051  0.29310
## Total.intl.minutes     -4.668e+00  6.111e+00  -0.764  0.44495
## Total.intl.calls       -7.450e-02  2.837e-02  -2.626  0.00864 **
## Total.intl.charge       1.763e+01  2.263e+01   0.779  0.43605
## Customer.service.calls  4.532e-01  4.471e-02  10.138  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2067.9  on 2499  degrees of freedom
## Residual deviance: 1651.2  on 2481  degrees of freedom
## AIC: 1689.2
##
## Number of Fisher Scoring iterations: 5
```

Model 1 - Log Likelihood

Analyzing the results we see that only `International.plan`, `Customer.service.calls`and `Total.intl.calls` are statistically significant within this model.This suggests a strong association of the these attributes with the probability of having churned. The negative coefficient for `Total.intl.calls`, suggests that all other variables being equal, the customers with a high number of international calls are less likely to have churned.

For a given model, we want to maximize the log likelihood. Here we see the log likelihood for model 1.

```
logLik(model.1)
```

```
## 'log Lik.' -825.5975 (df=19)
```

Model 1 - R-Squared

R-squared is a statistical measure of how close the data are to the fitted regression line. That is, it measures how well the model explains the variability of the response data around its mean by find how much variation is explained by the model.

```
#log-likelihood of the null model
model1.null <- model.1$null.deviance/-2

#log-likelihood of model 1
model1.proposed <- model.1$deviance/-2

#Calculating McFaddens Pseudo R squared
r_sq1 <- (model1.null  - model1.proposed)/model1.null
r_sq1
```

```
## [1] 0.2015152
```

Model 1 - P-Value

Now we want to find the associated p-value for our model 1.

```
p_value1 <- 1 - pchisq(2*(model1.proposed - model1.null),
                       df = (length(model.1$coefficients)-1))
p_value1
```

```
## [1] 0
```

Since the p-value is 0, it indicates that the explained variation is not due to chance.

Model 1 - Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model. Below is the confusion matrix for our model 1.

First we will use the test data to make prediction for the Churn probabilities. After converting all probabilities greater than or equal to 0.5 to 1 and the probabilities less than 0.5 to 0, where 1 indicates the customer has churned and 0 indicated that the customer has not churned, we will display the confusion matrix.

```
#predict the probabilities
probtest.model1 =predict(model.1, test.data, type = "response")

#Re-code probability to classifiers
```

```
predVal1 <- ifelse(probtest.model1 >= 0.5, 1, 0)
predtest.model1 <- factor(predVal1, levels = c(0,1))

# Assigning the target class to a variable
actualTest.model1 <-test.data$Churn
```

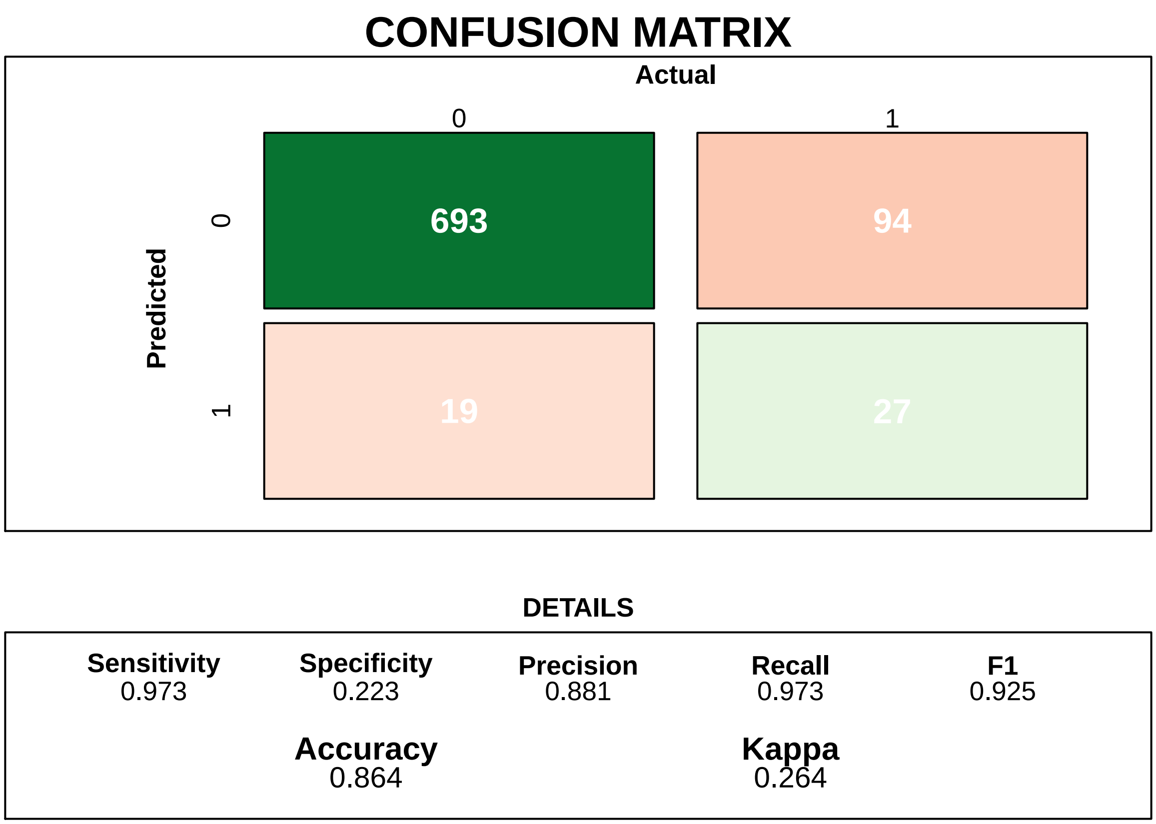
To create the confusion matrix with the associated perfomance measures we need to evaluate the model, we create the function draw_confusion_matrix.

```
draw_confusion_matrix <- function(cm) {

  total <- sum(cm$table)
  res <- as.numeric(cm$table)

  # Generate color gradients. Palettes come from RColorBrewer.
  greenPalette <- c("#F7FCF5","#E5F5E0","#C7E9C0","#A1D99B","#74C476","#41AB5D","#238B45","#006D2C","#0
  redPalette <- c("#FFF5F0","#FEE0D2","#FCBBA1","#FC9272","#FB6A4A","#EF3B2C","#CB181D","#A50F15","#670
  getColor <- function (greenOrRed = "green", amount = 0) {
    if (amount == 0)
      return("#FFFFFF")
    palette <- greenPalette
    if (greenOrRed == "red")
      palette <- redPalette
    colorRampPalette(palette)(100)[10 + ceiling(90 * amount / total)]
  }

  # set the basic layout
  layout(matrix(c(1,1,2)))
  par(mar=c(2,2,2,2))
  plot(c(100, 345), c(300, 450), type = "n", xlab="", ylab="", xaxt='n', yaxt='n')
  title('CONFUSION MATRIX', cex.main=2)

  # create the matrix
  classes = colnames(cm$table)
  rect(150, 430, 240, 370, col=getColor("green", res[1]))
  text(195, 435, classes[1], cex=1.2)
  rect(250, 430, 340, 370, col=getColor("red", res[3]))
  text(295, 435, classes[2], cex=1.2)
  text(125, 370, 'Predicted', cex=1.3, srt=90, font=2)
  text(245, 450, 'Actual', cex=1.3, font=2)
  rect(150, 305, 240, 365, col=getColor("red", res[2]))
  rect(250, 305, 340, 365, col=getColor("green", res[4]))
  text(140, 400, classes[1], cex=1.2, srt=90)
  text(140, 335, classes[2], cex=1.2, srt=90)

  # add in the cm results
  text(195, 400, res[1], cex=1.6, font=2, col='white')
  text(195, 335, res[2], cex=1.6, font=2, col='white')
  text(295, 400, res[3], cex=1.6, font=2, col='white')
  text(295, 335, res[4], cex=1.6, font=2, col='white')

  # add in the specifics
```

```
  plot(c(100, 0), c(100, 0), type = "n", xlab="", ylab="", main = "DETAILS", xaxt='n', yaxt='n')
  text(10, 85, names(cm$byClass[1]), cex=1.2, font=2)
  text(10, 70, round(as.numeric(cm$byClass[1]), 3), cex=1.2)
  text(30, 85, names(cm$byClass[2]), cex=1.2, font=2)
  text(30, 70, round(as.numeric(cm$byClass[2]), 3), cex=1.2)
  text(50, 85, names(cm$byClass[5]), cex=1.2, font=2)
  text(50, 70, round(as.numeric(cm$byClass[5]), 3), cex=1.2)
  text(70, 85, names(cm$byClass[6]), cex=1.2, font=2)
  text(70, 70, round(as.numeric(cm$byClass[6]), 3), cex=1.2)
  text(90, 85, names(cm$byClass[7]), cex=1.2, font=2)
  text(90, 70, round(as.numeric(cm$byClass[7]), 3), cex=1.2)

  # add in the accuracy information
  text(30, 35, names(cm$overall[1]), cex=1.5, font=2)
  text(30, 20, round(as.numeric(cm$overall[1]), 3), cex=1.4)
  text(70, 35, names(cm$overall[2]), cex=1.5, font=2)
  text(70, 20, round(as.numeric(cm$overall[2]), 3), cex=1.4)
}
```

See the confusion matrix for model 1 below. We will use these statistics for model performance comparison later on.

```
model1_cm <- confusionMatrix(predtest.model1,actualTest.model1)
draw_confusion_matrix(model1_cm)
```



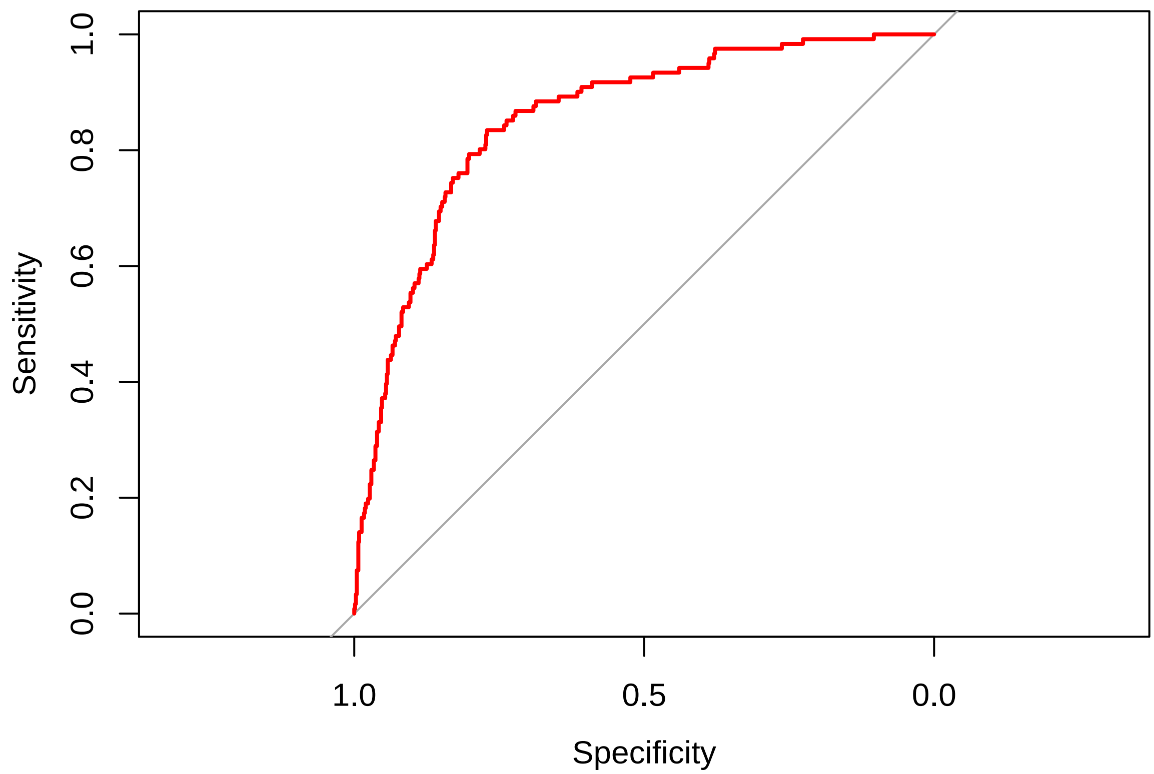
Model 1 - ROC & AUC

ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0s as 0s and 1s as 1s.

```
ROC1 <- roc(actualTest.model1, probtest.model1)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(ROC1, col="red")
```




```
#Area under the curve
AUC1 <- auc(ROC1)
AUC1
```

```
## Area under the curve: 0.8544
```

Model 1 - Stability

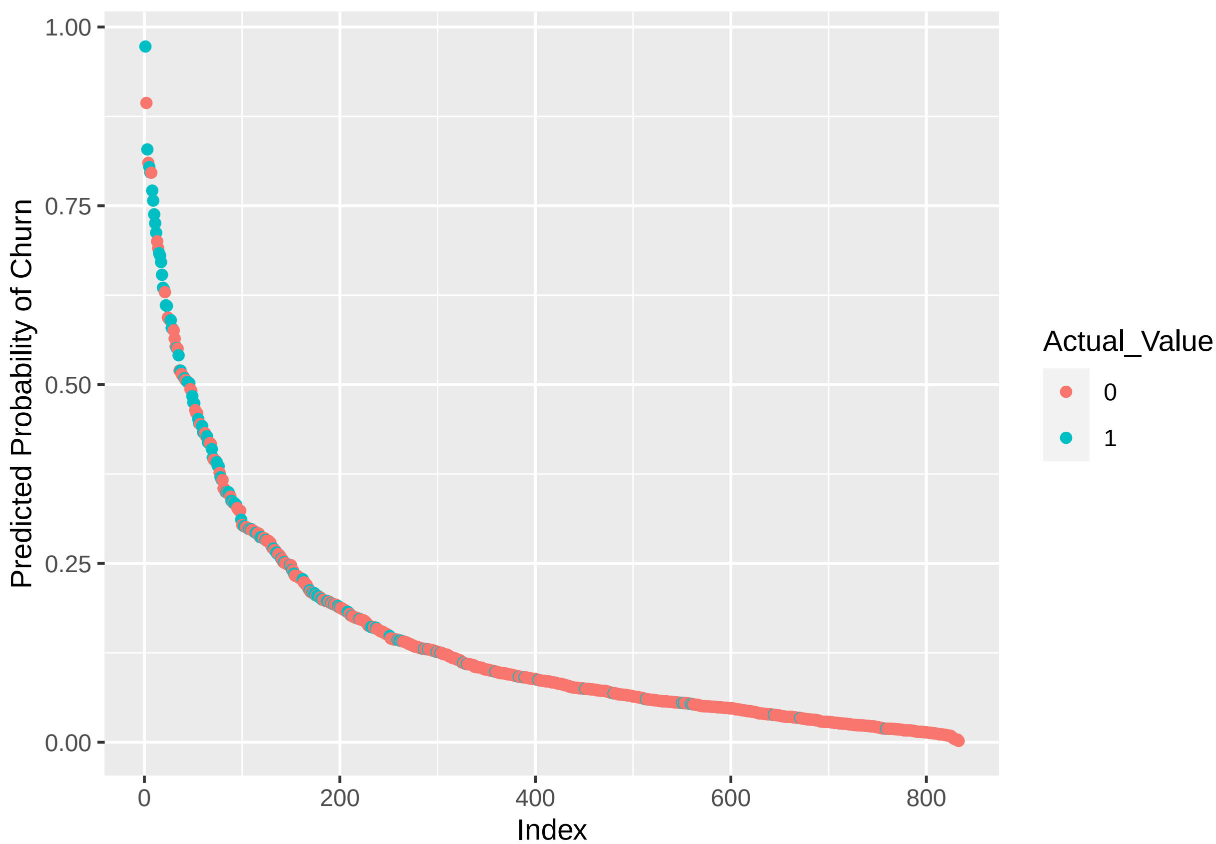
First we want to create a new data frame with the with predicted probabilities, Actual Value and Predicted Value.

```
predicted_data1 <- data.frame(Probs = probtest.model1,
                              Actual_Value= actualTest.model1,
                              Predicted_Value = predtest.model1 )

#sorting the probabilities
```

```r
predicted_data1 <- predicted_data1[order(predicted_data1$Probs,
                                         decreasing=TRUE),]

# Add rank variable
predicted_data1$Rank <- 1:nrow(predicted_data1)

head(predicted_data1)
```

```
##          Probs Actual_Value Predicted_Value Rank
## 2733 0.9727029            1               1    1
## 3310 0.8938153            0               1    2
## 2595 0.8288437            1               1    3
## 185  0.8101122            0               1    4
## 1594 0.8041893            1               1    5
## 1534 0.7966862            1               1    6
```

```r
ggplot(data=predicted_data1, aes(x=Rank, y=Probs)) +
  geom_point(aes(color = Actual_Value)) + xlab("Index") + ylab("Predicted Probability of Churn")
```



The graph above shows how the actual churn value is distributed against the models predicted probability of churn. We know that a rank/index of 1 will have a higher probability and will decrease as you move down the rank. The colour here shows the distribution of the actual values.

Next we will put the values into decile.

```r
#Creating an empty data frame
decile.model1<- data.frame(matrix(ncol=4,nrow = 0))
colnames(decile.model1) <- c("Decile", "per_correct_preds", "No_correct_Preds",
                             "cum_preds")

#Initializing the variables
num_of_deciles=10
Obs_per_decile<-nrow(predicted_data1)/num_of_deciles
decile_count=1
start=1
stop=(start-1) + Obs_per_decile
prev_cum_pred<-0
x=0

#Creating the deciles
while (x < nrow(predicted_data1)) {
  subset<-predicted_data1[c(start:stop),]
  correct_count<- ifelse(subset$Actual_Value==subset$Predicted_Value,1,0)
  no_correct_Preds<-sum(correct_count,na.rm = TRUE)
  per_correct_Preds<-(no_correct_Preds/Obs_per_decile)*100
  cum_preds<-no_correct_Preds+prev_cum_pred
  addRow<-data.frame("Decile"=decile_count,"per_correct_preds"=per_correct_Preds,"No_correct_Preds"=no_c
  decile.model1<-rbind(decile.model1,addRow)
  prev_cum_pred<-prev_cum_pred+no_correct_Preds
  start<-stop+1
  stop=(start-1) + Obs_per_decile
  x<-x+Obs_per_decile
  decile_count<-decile_count+1
}
```

See data below for the stability table

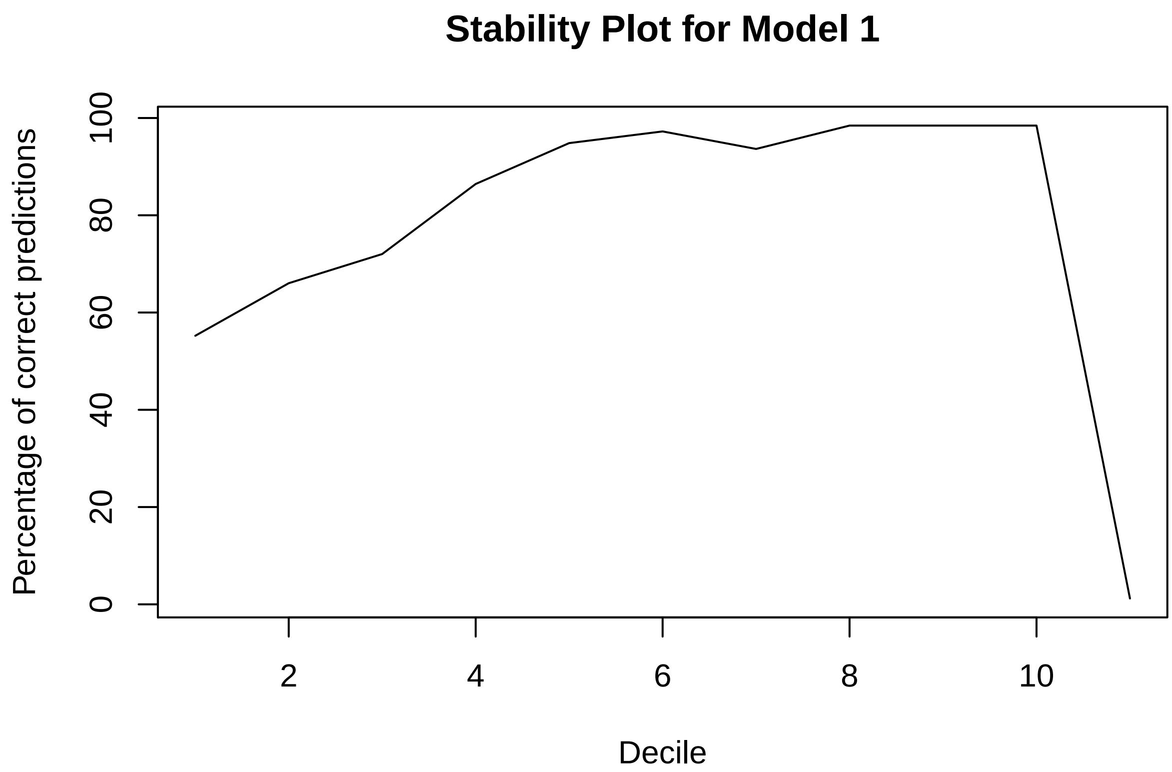
```r
decile.model1
```

```
##    Decile per_correct_preds No_correct_Preds cum_preds
## 1       1          55.22209               46        46
## 2       2          66.02641               55       101
## 3       3          72.02881               60       161
## 4       4          86.43457               72       233
## 5       5          94.83794               79       312
## 6       6          97.23890               81       393
## 7       7          93.63745               78       471
## 8       8          98.43938               82       553
## 9       9          98.43938               82       635
## 10     10          98.43938               82       717
## 11     11           1.20048                1       718
```

Plotting the stability graph for model 1 gives.

```r
plot(decile.model1$Decile,
     decile.model1$per_correct_preds,
     type = "l",
```

```
     xlab = "Decile",
     ylab = "Percentage of correct predictions",
     main="Stability Plot for Model 1")
```



Based on visual inspection of the deciles of Model1, we can conclude that the model is unstable

Model 2

For the second regression model we will start out with a blank model. The starting point here will be an intercept and no terms except the response (churn).

```
base.model <- glm(Churn ~ 1, data = train.data, family = binomial)
summary(base.model)
```

```
##
## Call:
## glm(formula = Churn ~ 1, family = binomial, data = train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.5593  -0.5593  -0.5593  -0.5593   1.9659
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.77598    0.05683  -31.25   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2067.9  on 2499  degrees of freedom
## Residual deviance: 2067.9  on 2499  degrees of freedom
## AIC: 2069.9
##
## Number of Fisher Scoring iterations: 4
```

Now that we have a blank model we will look at the effect of adding each variable in turn. The variable that has the lowest AIC value will be the ones we incorporate into our model 2.

```
add1(base.model, scope = train.data, test = 'Chisq')
```

```
## Single term additions
##
## Model:
## Churn ~ 1
##                        Df Deviance    AIC     LRT  Pr(>Chi)
## <none>                      2067.9 2069.9
## Area.code               1   2067.9 2071.9   0.005 0.9453382
## International.plan      1   1931.8 1935.8 136.112 < 2.2e-16 ***
## Voice.mail.plan         1   2045.4 2049.4  22.495 2.107e-06 ***
## Number.vmail.messages   1   2049.2 2053.2  18.739 1.499e-05 ***
## Total.day.minutes       1   1973.8 1977.8  94.156 < 2.2e-16 ***
## Total.day.calls         1   2066.9 2070.9   1.000 0.3172526
## Total.day.charge        1   1973.8 1977.8  94.154 < 2.2e-16 ***
## Total.eve.minutes       1   2048.8 2052.8  19.123 1.225e-05 ***
## Total.eve.calls         1   2067.9 2071.9   0.009 0.9261324
## Total.eve.charge        1   2048.8 2052.8  19.120 1.227e-05 ***
## Total.night.minutes     1   2064.8 2068.8   3.153 0.0757651 .
## Total.night.calls       1   2067.6 2071.6   0.337 0.5616236
## Total.night.charge      1   2064.8 2068.8   3.157 0.0755993 .
```

```
## Total.intl.minutes      1   2055.4 2059.4  12.540 0.0003983 ***
## Total.intl.calls        1   2060.9 2064.9   7.053 0.0079145 **
## Total.intl.charge       1   2055.4 2059.4  12.551 0.0003960 ***
## Customer.service.calls  1   1986.4 1990.4  81.544 < 2.2e-16 ***
## Churn                   0   2067.9 2069.9   0.000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here we want to include only variables that are significant.

```
model.2 <-  glm(Churn ~ International.plan + Total.day.minutes +
                  Total.day.charge + Voice.mail.plan +Number.vmail.messages +
                  Total.eve.minutes + Total.eve.charge + Total.intl.charge +
                  Customer.service.calls,
                data=train.data,
                family=binomial(link="logit"))

summary(model.2)
```

```
##
## Call:
## glm(formula = Churn ~ International.plan + Total.day.minutes +
##     Total.day.charge + Voice.mail.plan + Number.vmail.messages +
##     Total.eve.minutes + Total.eve.charge + Total.intl.charge +
##     Customer.service.calls, family = binomial(link = "logit"),
##     data = train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8674  -0.5247  -0.3602  -0.2228   2.9877
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -7.24560    0.49204 -14.726  < 2e-16 ***
## International.plan       2.04515    0.16259  12.579  < 2e-16 ***
## Total.day.minutes        1.58617    3.72085   0.426   0.6699
## Total.day.charge        -9.26032   21.88745  -0.423   0.6722
## Voice.mail.plan         -1.45840    0.62365  -2.339   0.0194 *
## Number.vmail.messages    0.01881    0.01989   0.946   0.3444
## Total.eve.minutes        1.07222    1.87261   0.573   0.5669
## Total.eve.charge       -12.53862   22.03083  -0.569   0.5693
## Total.intl.charge        0.33572    0.08609   3.900 9.63e-05 ***
## Customer.service.calls   0.45180    0.04442  10.170  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2067.9  on 2499  degrees of freedom
## Residual deviance: 1669.7  on 2490  degrees of freedom
## AIC: 1689.7
##
## Number of Fisher Scoring iterations: 5
```

Model 2 - Log Likelihood

Here we calculate the log likelihood of model 2.

```
logLik(model.2)
```

```
## 'log Lik.' -834.8282 (df=10)
```

Model 2 - R-Squared

Calculating the r-squared value for model 2, gives:

```
#log-likelihood of the null model
model2.null <- model.2$null.deviance/-2

#log-likelihood of model 2
model2.proposed <- model.2$deviance/-2

#Calculating McFaddens Pseudo R squared
r_sq2 <- (model2.null  - model2.proposed)/model2.null
r_sq2
```

```
## [1] 0.1925876
```

Model 2 - P-Value

```
p_value2 <- 1 - pchisq(2*(model1.proposed - model1.null),
                      df = (length(model.1$coefficients)-1))
p_value2
```

```
## [1] 0
```

Since the p-value is 0, it indicates that the explained variation in this model is not due to chance.

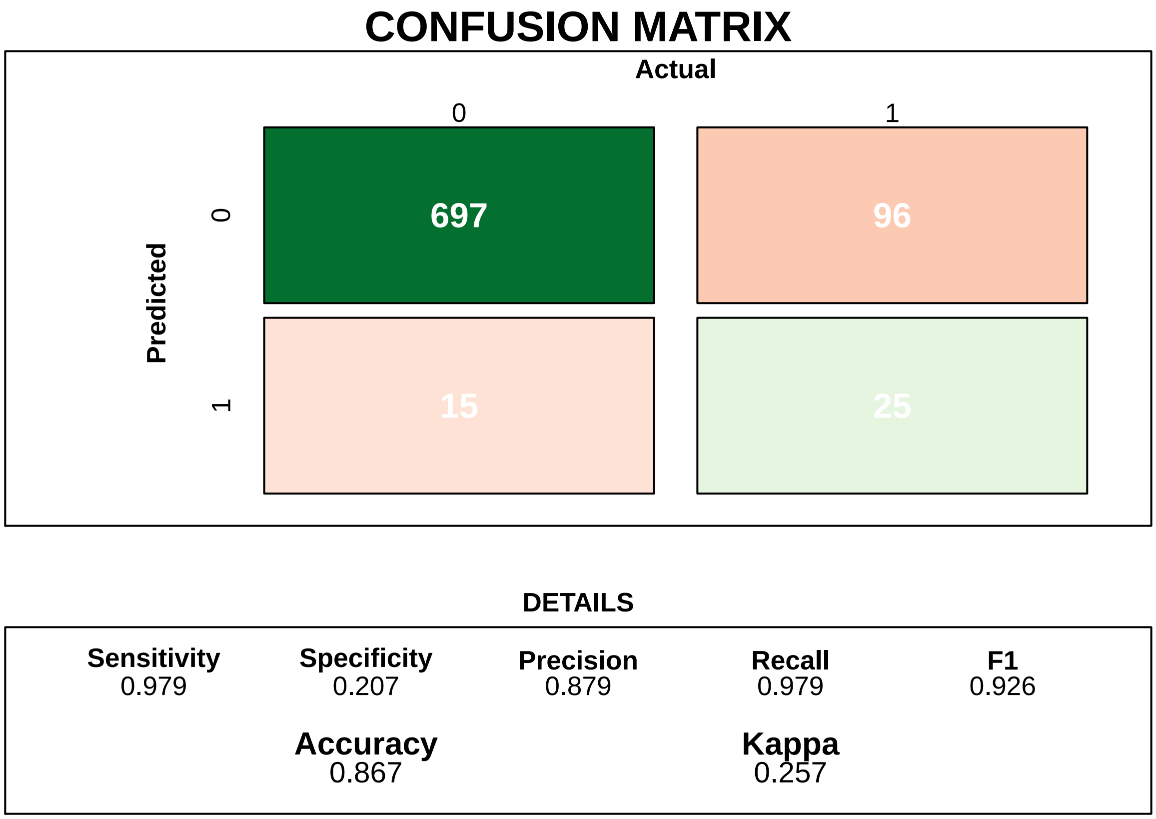
Model 2 - Confusion Matrix

Again, we use the test data to make prediction for the Churn probabilities. After converting all probabilities greater than or equal to 0.5 to 1 and the probabilities less than 0.5 to 0, where 1 indicates the customer has churned and 0 indicated that the customer has not churned, we will display the confusion matrix.

```
#predict the probabilities
probtest.model2 =predict(model.2, test.data, type = "response")

#Re-code probability to classifiers
predVal2 <- ifelse(probtest.model2 >= 0.5, 1, 0)
predtest.model2 <- factor(predVal2, levels = c(0,1))

# Assigning the target class to a variable
actualTest.model2 <-test.data$Churn
```

The confusion matrix for model 2 can be seen below

```
model2_cm <- confusionMatrix(predtest.model2,actualTest.model2)
draw_confusion_matrix(model2_cm)
```



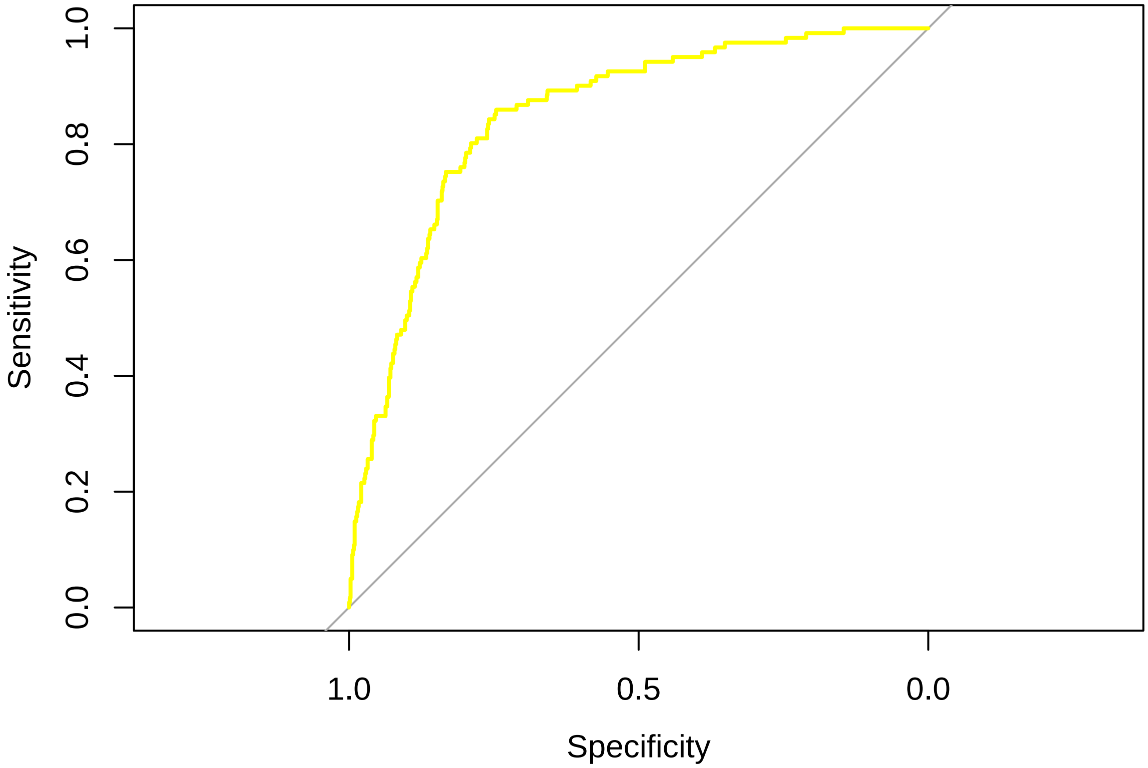
We will take a deeper look at the details later.

Model 2 - ROC & AUC

```
ROC2 <- roc(actualTest.model2, probtest.model2)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(ROC2, col="yellow")
```




```r
#Area under the curve
AUC2 <- auc(ROC2)
AUC2
```

```
## Area under the curve: 0.8497
```

Model 2 - Stability

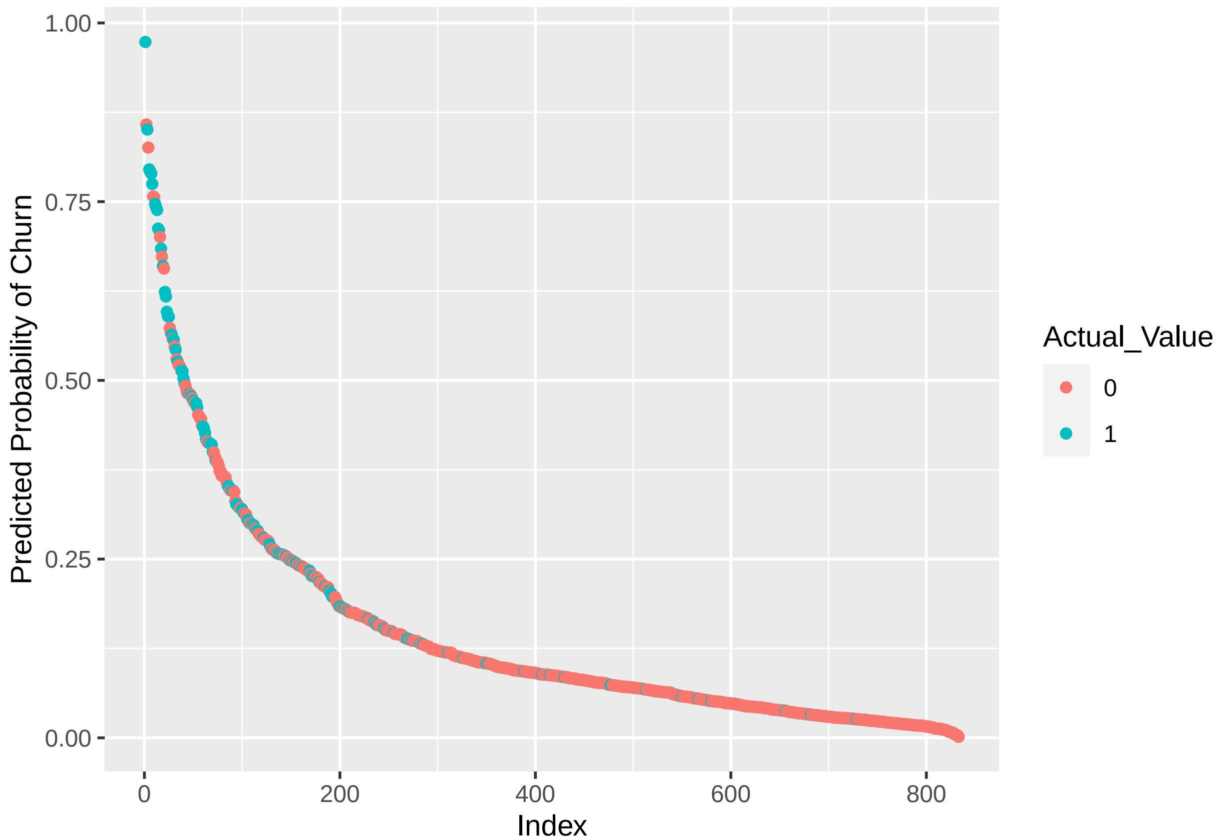
First we want to create a new data frame with the with predicted probabilities, Actual Value and Predicted Value.

```r
predicted_data2 <- data.frame(Probs = probtest.model2,
                              Actual_Value= actualTest.model2,
                              Predicted_Value = predtest.model2)

#sorting the probabilities
predicted_data2 <- predicted_data2[order(predicted_data2$Probs,
                                         decreasing=TRUE),]

# Add rank variable
predicted_data2$Rank <- 1:nrow(predicted_data2)

head(predicted_data2)
```

```
##          Probs Actual_Value Predicted_Value Rank
## 2733 0.9733963            1               1    1
## 3310 0.8580365            0               1    2
## 2595 0.8509580            1               1    3
```

```
## 185  0.8257149            0              1    4
## 547  0.7951517            1              1    5
## 1594 0.7916990            1              1    6
```

```
ggplot(data=predicted_data2, aes(x=Rank, y=Probs)) +
  geom_point(aes(color = Actual_Value)) + xlab("Index") + ylab("Predicted Probability of Churn")
```



We see the same graph again, this time show the distribution of the probability of churn distribution for model 2.

Next we will put the values into deciles.

```
#Creating an empty data frame
decile.model2<- data.frame(matrix(ncol=4,nrow = 0))
colnames(decile.model2) <- c("Decile", "per_correct_preds", "No_correct_Preds",
                             "cum_preds")

#Initializing the variables
num_of_deciles=10
Obs_per_decile<-nrow(predicted_data2)/num_of_deciles
decile_count=1
start=1
stop=(start-1) + Obs_per_decile
prev_cum_pred<-0
```

```r
x=0

#Creating the deciles
while (x < nrow(predicted_data2)) {
  subset<-predicted_data2[c(start:stop),]
  correct_count<- ifelse(subset$Actual_Value==subset$Predicted_Value,1,0)
  no_correct_Preds<-sum(correct_count,na.rm = TRUE)
  per_correct_Preds<-(no_correct_Preds/Obs_per_decile)*100
  cum_preds<-no_correct_Preds+prev_cum_pred
  addRow<-data.frame("Decile"=decile_count,"per_correct_preds"=per_correct_Preds,"No_correct_Preds"=no_c
  decile.model2<-rbind(decile.model2,addRow)
  prev_cum_pred<-prev_cum_pred+no_correct_Preds
  start<-stop+1
  stop=(start-1) + Obs_per_decile
  x<-x+Obs_per_decile
  decile_count<-decile_count+1
}
```

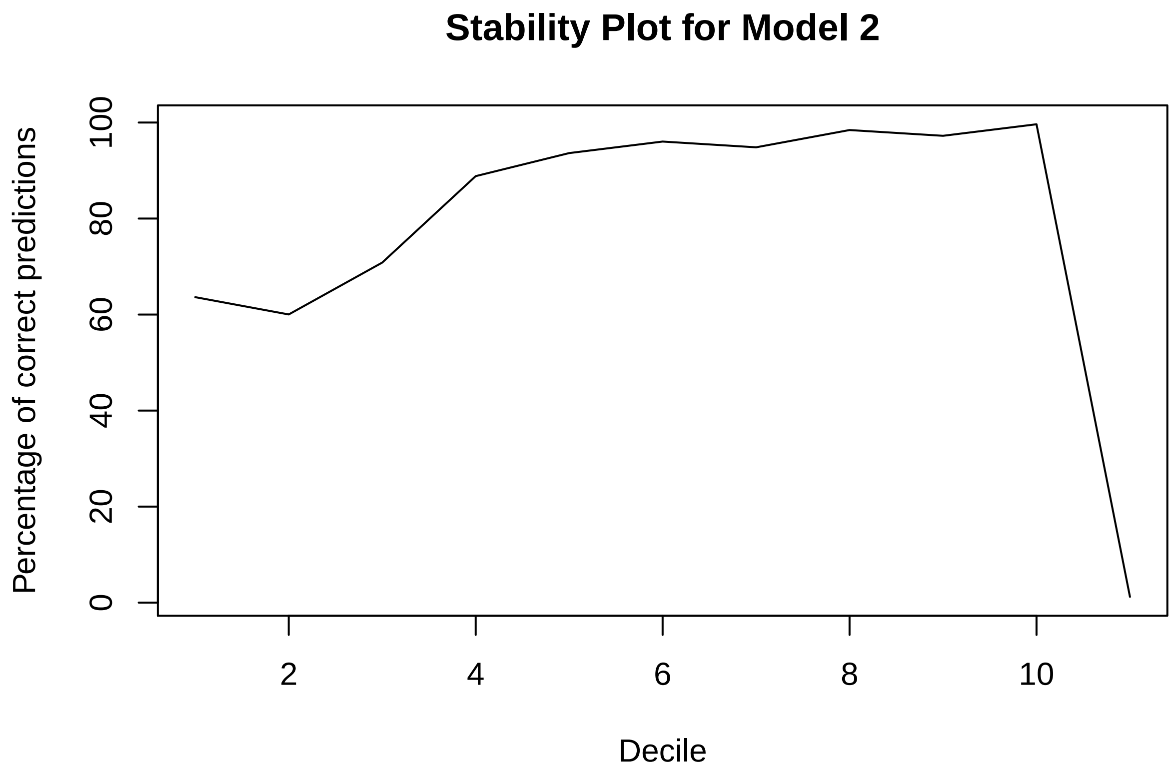
See data below for the stability table

```r
decile.model2
```

```
##    Decile per_correct_preds No_correct_Preds cum_preds
## 1       1          63.62545               53        53
## 2       2          60.02401               50       103
## 3       3          70.82833               59       162
## 4       4          88.83553               74       236
## 5       5          93.63745               78       314
## 6       6          96.03842               80       394
## 7       7          94.83794               79       473
## 8       8          98.43938               82       555
## 9       9          97.23890               81       636
## 10     10          99.63986               83       719
## 11     11           1.20048                1       720
```

Plotting the stability graph for model 1 gives.

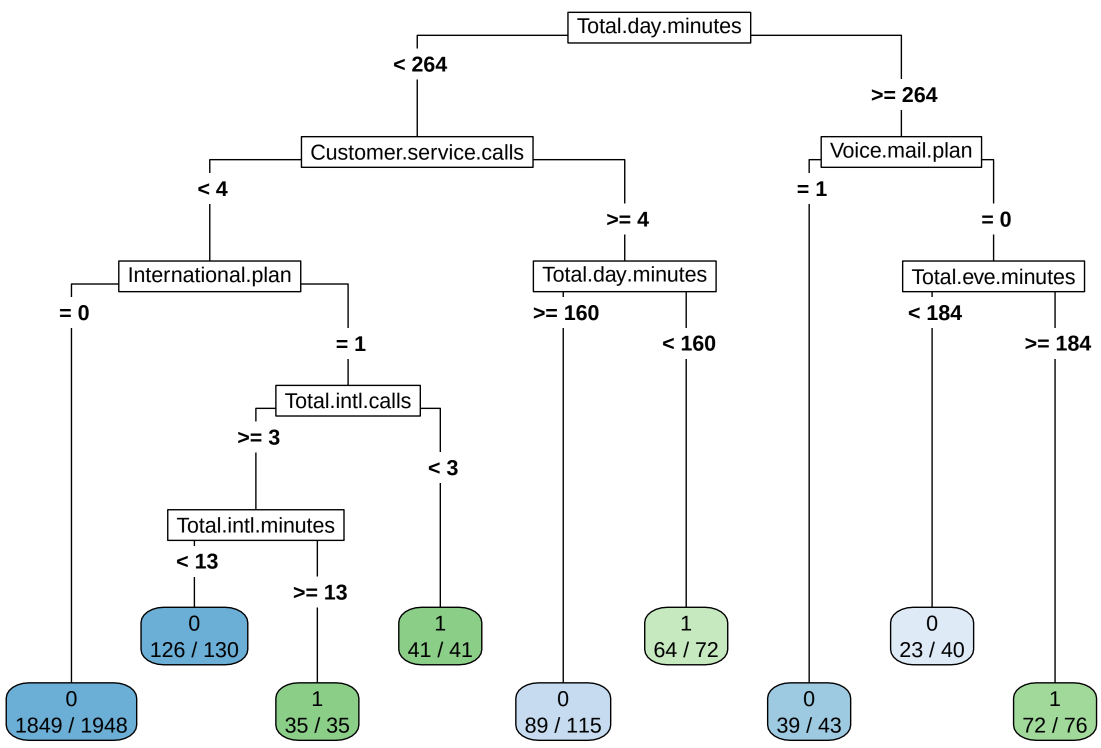
```r
plot(decile.model2$Decile,
     decile.model2$per_correct_preds,
     type = "l",
     xlab = "Decile",
     ylab = "Percentage of correct predictions",
     main="Stability Plot for Model 2")
```



Based on the visualization above we can conclude that model 2 is unstable.

Model 3

```
model.3 <- rpart(Churn ~ .,
                 method="class",
                 data=train.data,
                 parms = list (split ="information gain"),
                 control = rpart.control(minsplit = 100, maxdepth = 6))

rpart.plot(model.3, type=5, extra = 2, fallen.leaves = T, cex = 0.7)
```



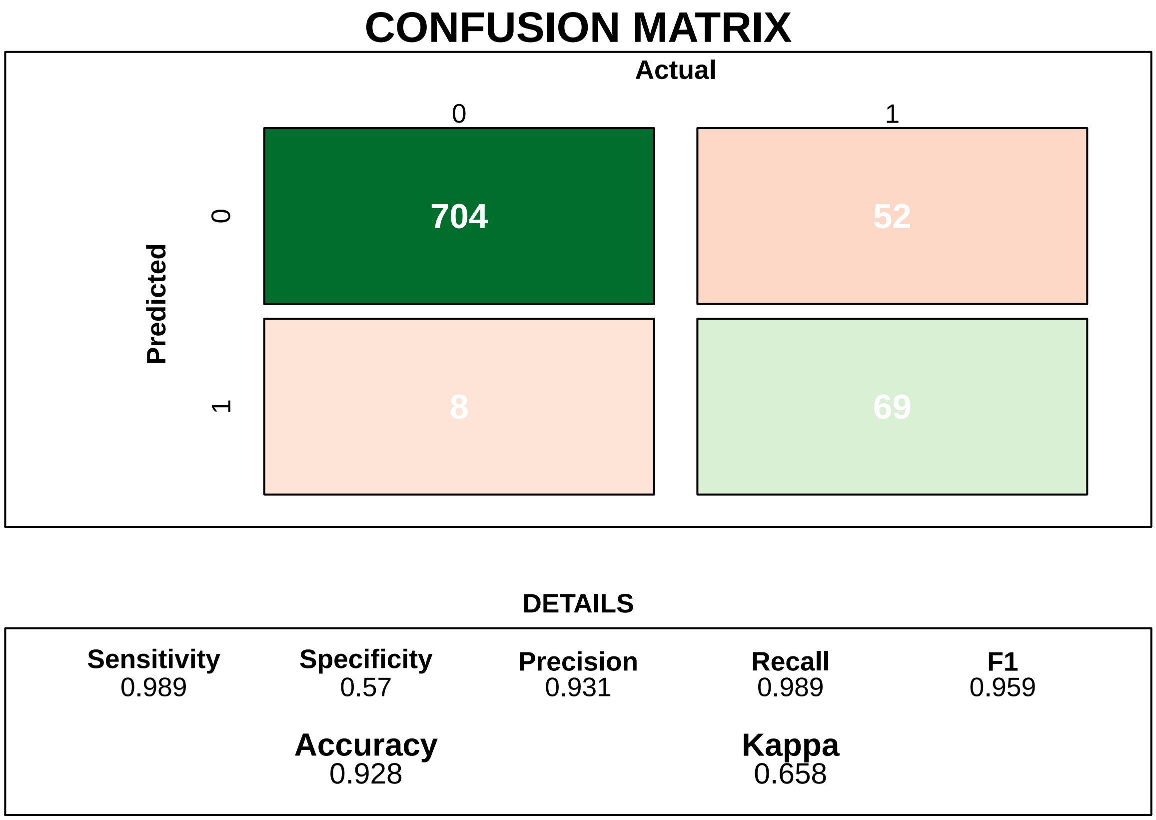
Model 3 - Confusion Matrix

Using test data to make prediction for the Churn probabilities we find the predictor class and probabilities. We will then display the confusion matrix.

```
#predicting the class
predtest.model3 <- predict(model.3, test.data, type="class")

#predicting the probabilities
probtest.model3 <- predict(model.3, test.data, type="prob")

# Assigning the target class to a variable
actualTest.model3 <-test.data$Churn
```

The confusion matrix for model 3 can be seen below.

```
model3_cm <- confusionMatrix(predtest.model3,actualTest.model3)
draw_confusion_matrix(model3_cm)
```



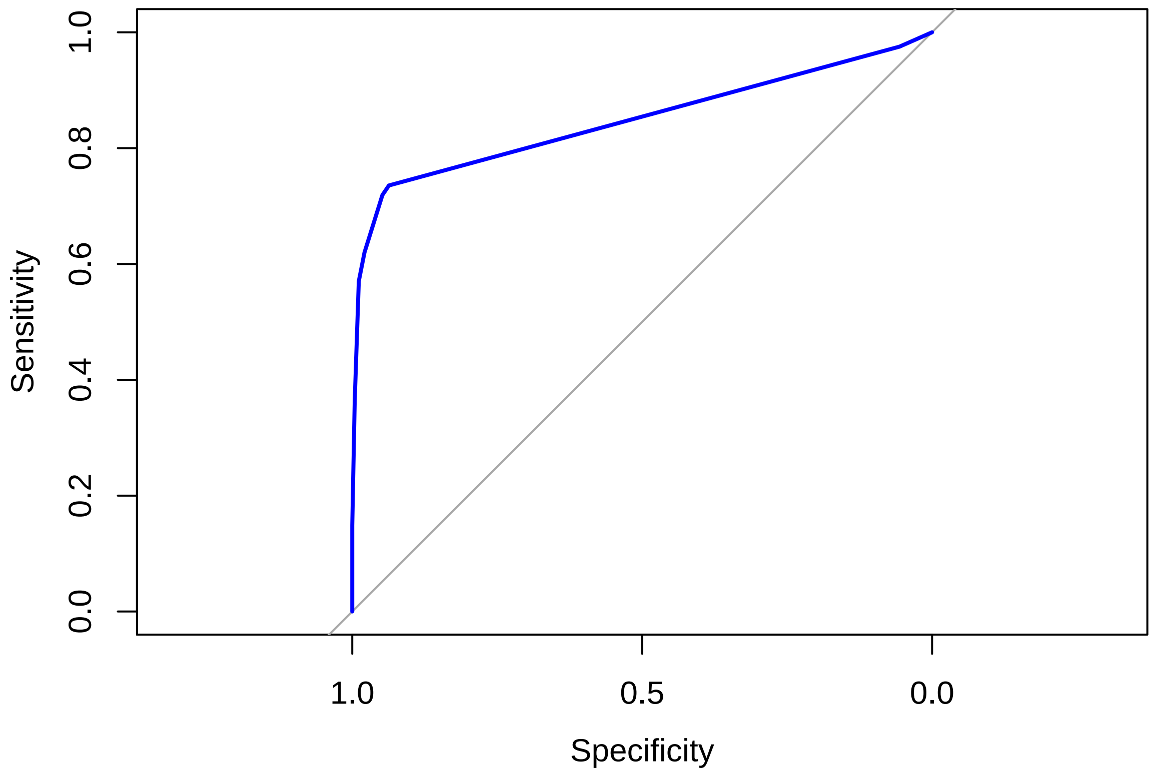
Model 3 - ROC & AUC

```
ROC3 <- roc(actualTest.model3, probtest.model3[,2])
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(ROC3, col="blue")
```




```r
#Area under the curve
AUC3 <- auc(ROC3)
AUC3
```

```
## Area under the curve: 0.8478
```

Model 3 - Stability

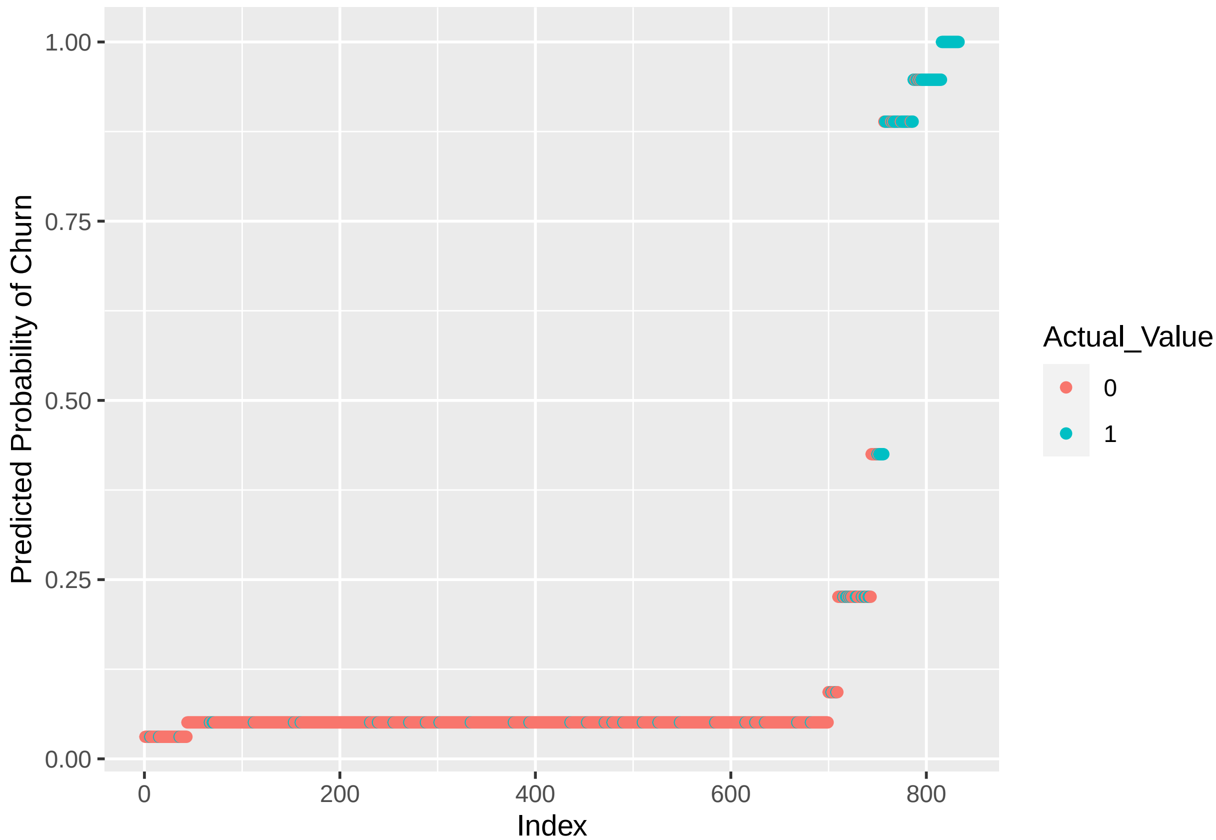
Again we want to create a new data frame with the with predicted probabilities, Actual Value and Predicted Value.

```r
predicted_data3 <- data.frame(Probs = probtest.model3,
                              Actual_Value= actualTest.model3,
                              Predicted_Value = predtest.model3 )

#sorting the probabilities
predicted_data3 <- predicted_data3[order(predicted_data3$Probs.0,
                                         decreasing=TRUE),]

# Add rank variable
predicted_data3$Rank <- 1:nrow(predicted_data3)

head(predicted_data3)
```

```
##       Probs.0    Probs.1 Actual_Value Predicted_Value Rank
## 68  0.9692308 0.03076923            0               0    1
## 82  0.9692308 0.03076923            0               0    2
## 185 0.9692308 0.03076923            0               0    3
```

```
## 212 0.9692308 0.03076923           0              0    4
## 235 0.9692308 0.03076923           0              0    5
## 361 0.9692308 0.03076923           1              0    6
```

```r
ggplot(data=predicted_data3, aes(x=Rank, y=Probs.1)) +
  geom_point(aes(color = Actual_Value)) + xlab("Index") + ylab("Predicted Probability of Churn")
```



Looking at the graph above, we can see clearly that the model has done great job a predicting 0's as 0's and 1's as ones. This is apparent through the start divide between the red data points whose actual value is 0 and the probability of churn is low versus the blue data points whose actual value is 1 and the probability of churn is high.

Next we will put the values into deciles.

```r
#Creating an empty data frame
decile.model3<- data.frame(matrix(ncol=4,nrow = 0))
colnames(decile.model3) <- c("Decile", "per_correct_preds", "No_correct_Preds",
                             "cum_preds")

#Initializing the variables
num_of_deciles=10
Obs_per_decile<-nrow(predicted_data3)/num_of_deciles
decile_count=1
```

```r
start=1
stop=(start-1) + Obs_per_decile
prev_cum_pred<-0
x=0

#Creating the deciles
while (x < nrow(predicted_data3)) {
  subset<-predicted_data3[c(start:stop),]
  correct_count<- ifelse(subset$Actual_Value==subset$Predicted_Value,1,0)
  no_correct_Preds<-sum(correct_count,na.rm = TRUE)
  per_correct_Preds<-(no_correct_Preds/Obs_per_decile)*100
  cum_preds<-no_correct_Preds+prev_cum_pred
  addRow<-data.frame("Decile"=decile_count,"per_correct_preds"=per_correct_Preds,"No_correct_Preds"=no_c
  decile.model3<-rbind(decile.model3,addRow)
  prev_cum_pred<-prev_cum_pred+no_correct_Preds
  start<-stop+1
  stop=(start-1) + Obs_per_decile
  x<-x+Obs_per_decile
  decile_count<-decile_count+1
}
```

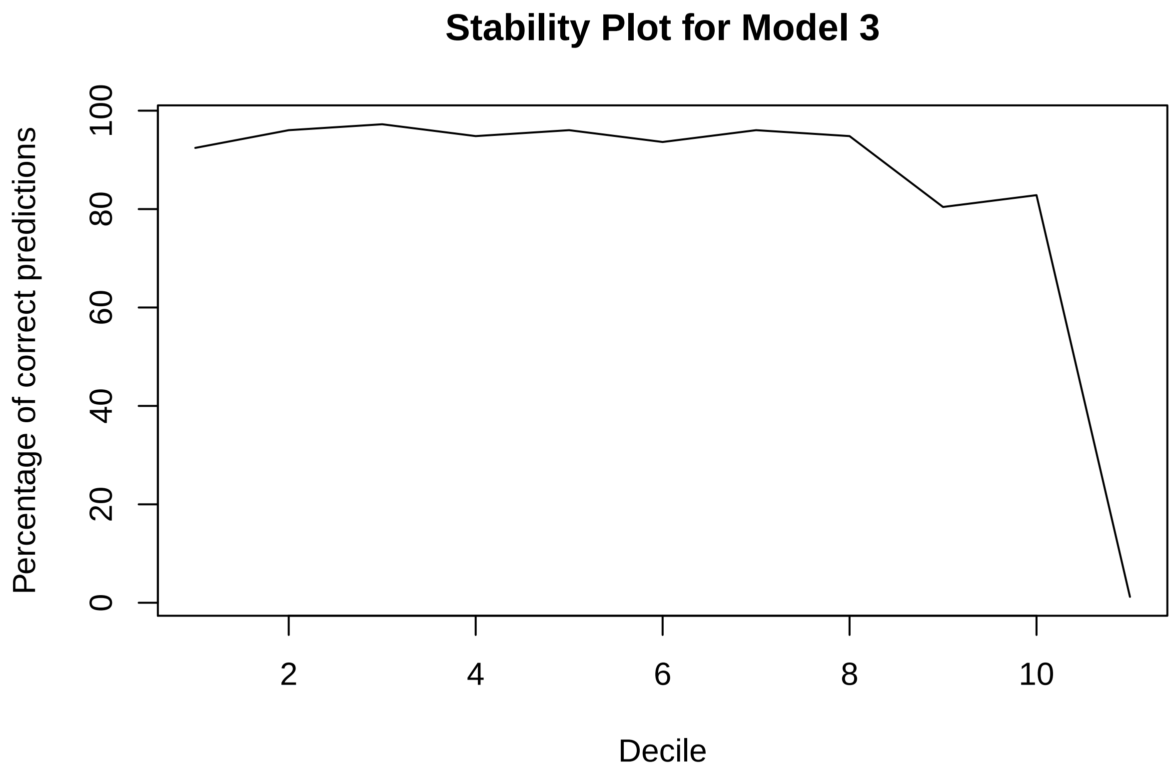
See data below for the stability table of class 0 for model 3.

```r
decile.model3
```

```
##    Decile per_correct_preds No_correct_Preds cum_preds
## 1       1          92.43697               77        77
## 2       2          96.03842               80       157
## 3       3          97.23890               81       238
## 4       4          94.83794               79       317
## 5       5          96.03842               80       397
## 6       6          93.63745               78       475
## 7       7          96.03842               80       555
## 8       8          94.83794               79       634
## 9       9          80.43217               67       701
## 10     10          82.83313               69       770
## 11     11           1.20048                1       771
```

Plotting the stability graph for model 2 gives.

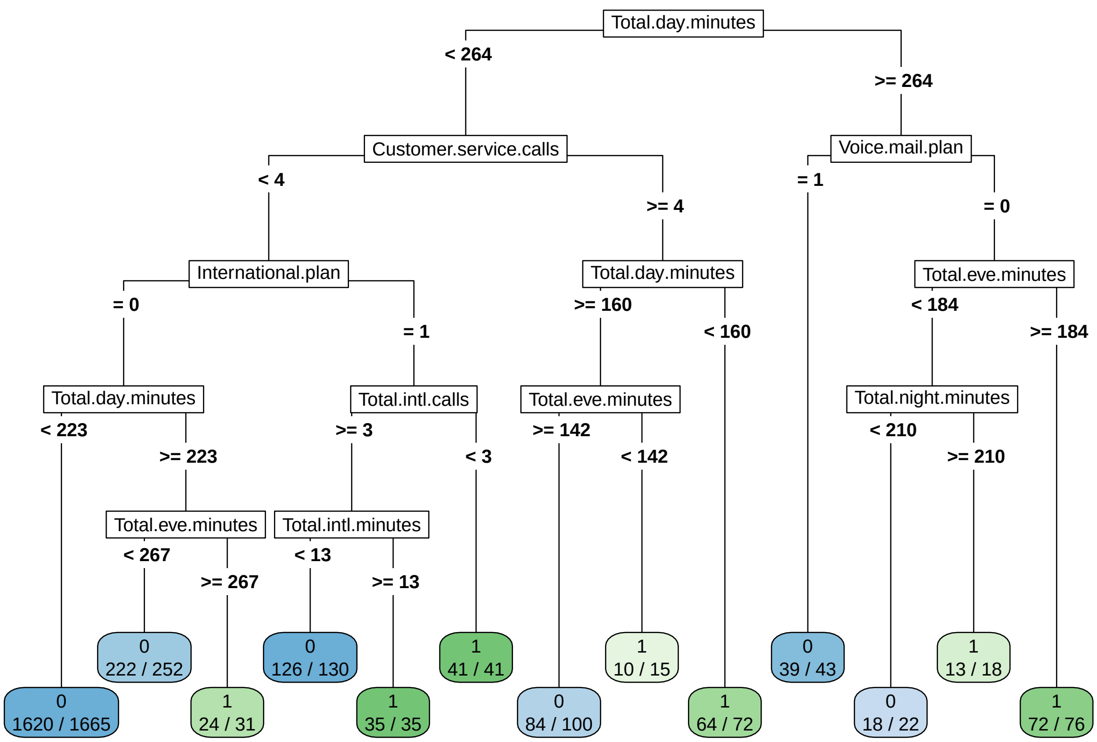
```r
plot(decile.model3$Decile,
     decile.model3$per_correct_preds,
     type = "l",
     xlab = "Decile",
     ylab = "Percentage of correct predictions",
     main="Stability Plot for Model 3")
```



Based on the visualization above we can conclude that model 3 is stable.

Model 4

```r
model.4 <- rpart(Churn ~ .,
                 method="class",
                 data=train.data,
                 parms = list (split ="gini"),
                 control = rpart.control(minsplit = 20, maxdepth = 6))

rpart.plot(model.4, type=5, extra = 2, fallen.leaves = T, cex = 0.6)
```



Model 4 - Confusion Matrix

Finally using test data to make prediction for the Churn probabilities we find the predictor class and probabilities. We will then display the confusion matrix.

```r
#predicting the class
predtest.model4 <- predict(model.4, test.data, type="class")

#predicting the probabilities
probtest.model4 <- predict(model.4, test.data, type="prob")

# Assigning the target class to a variable
actualTest.model4 <-test.data$Churn
```

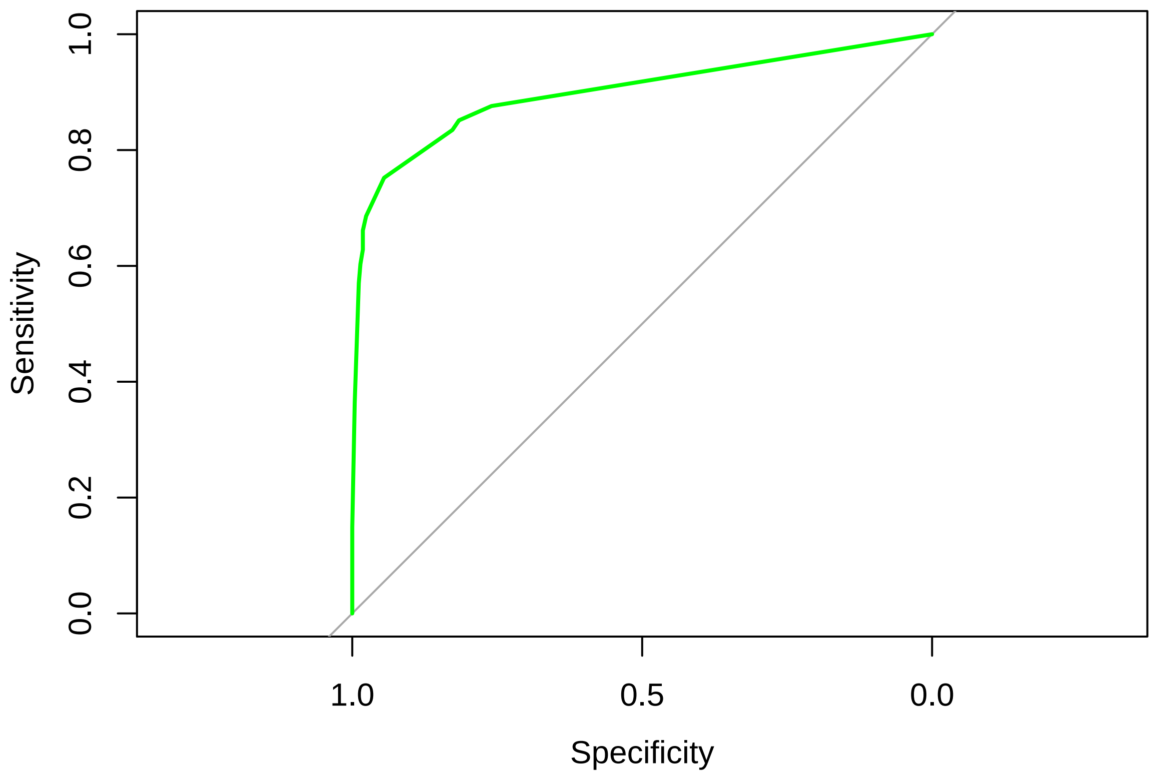
Model 4 - ROC & AUC

```
ROC4 <- roc(actualTest.model4, probtest.model4[,2])
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(ROC4, col="green")
```




```
#Area under the curve
AUC4 <- auc(ROC4)
AUC4
```

```
## Area under the curve: 0.8989
```

Model 4 - Stability

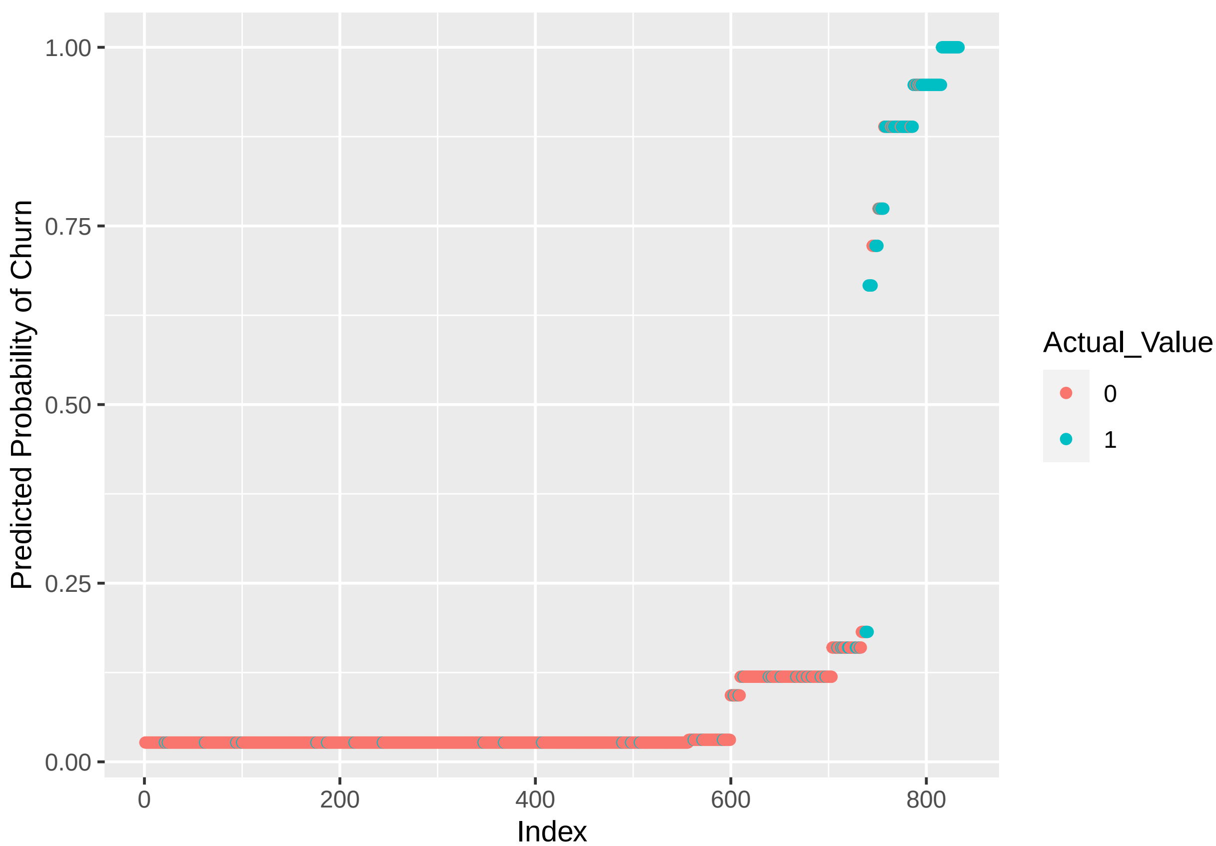
Finally, we want to create a new data frame with the with predicted probabilities, Actual Value and Predicted Value.

```
predicted_data4 <- data.frame(Probs = probtest.model4,
                              Actual_Value= actualTest.model4,
                              Predicted_Value = predtest.model4 )

#sorting the probabilities
predicted_data4 <- predicted_data4[order(predicted_data4$Probs.0,
                                         decreasing=TRUE),]
```

```r
# Add rank variable
predicted_data4$Rank <- 1:nrow(predicted_data4)

head(predicted_data4)
```

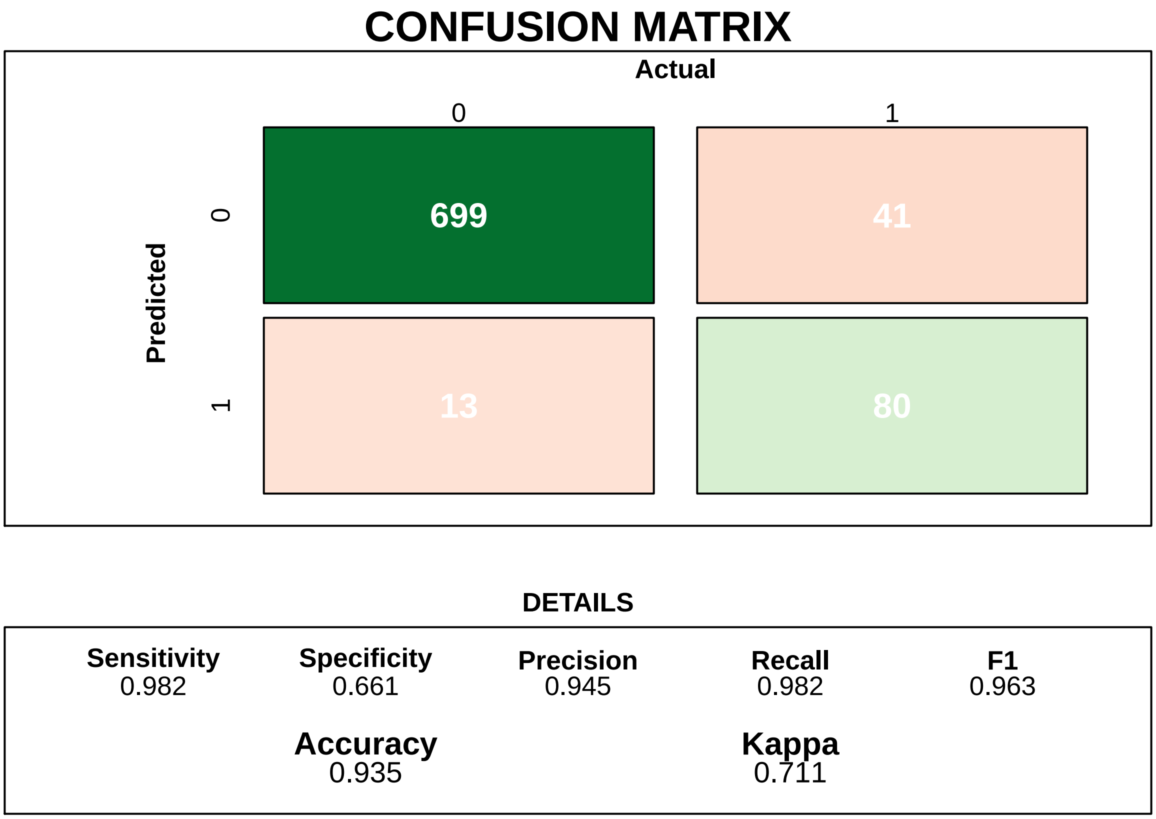
```
##      Probs.0    Probs.1 Actual_Value Predicted_Value Rank
## 17 0.972973 0.02702703            0               0    1
## 24 0.972973 0.02702703            0               0    2
## 29 0.972973 0.02702703            0               0    3
## 30 0.972973 0.02702703            0               0    4
## 33 0.972973 0.02702703            0               0    5
## 36 0.972973 0.02702703            0               0    6
```

```r
ggplot(data=predicted_data4, aes(x=Rank, y=Probs.1)) +
  geom_point(aes(color = Actual_Value)) + xlab("Index") + ylab("Predicted Probability of Churn")
```



Here again we see a plot of the predicted probabilities against the actual values. Giving the visual distinction of how well the model is performing.

```r
model4_cm <- confusionMatrix(predtest.model4,actualTest.model4)
draw_confusion_matrix(model4_cm)
```



Next we will put the values into deciles.

```
#Creating an empty data frame
decile.model4<- data.frame(matrix(ncol=4,nrow = 0))
colnames(decile.model4) <- c("Decile", "per_correct_preds", "No_correct_Preds",
                             "cum_preds")

#Initializing the variables
num_of_deciles=10
Obs_per_decile<-nrow(predicted_data4)/num_of_deciles
decile_count=1
start=1
stop=(start-1) + Obs_per_decile
prev_cum_pred<-0
x=0

#Creating the deciles
while (x < nrow(predicted_data4)) {
  subset<-predicted_data4[c(start:stop),]
  correct_count<- ifelse(subset$Actual_Value==subset$Predicted_Value,1,0)
  no_correct_Preds<-sum(correct_count,na.rm = TRUE)
  per_correct_Preds<-(no_correct_Preds/Obs_per_decile)*100
  cum_preds<-no_correct_Preds+prev_cum_pred
  addRow<-data.frame("Decile"=decile_count,"per_correct_preds"=per_correct_Preds,"No_correct_Preds"=no_
  decile.model4<-rbind(decile.model4,addRow)
  prev_cum_pred<-prev_cum_pred+no_correct_Preds
```

```
  start<-stop+1
  stop=(start-1) + Obs_per_decile
  x<-x+Obs_per_decile
  decile_count<-decile_count+1
}
```

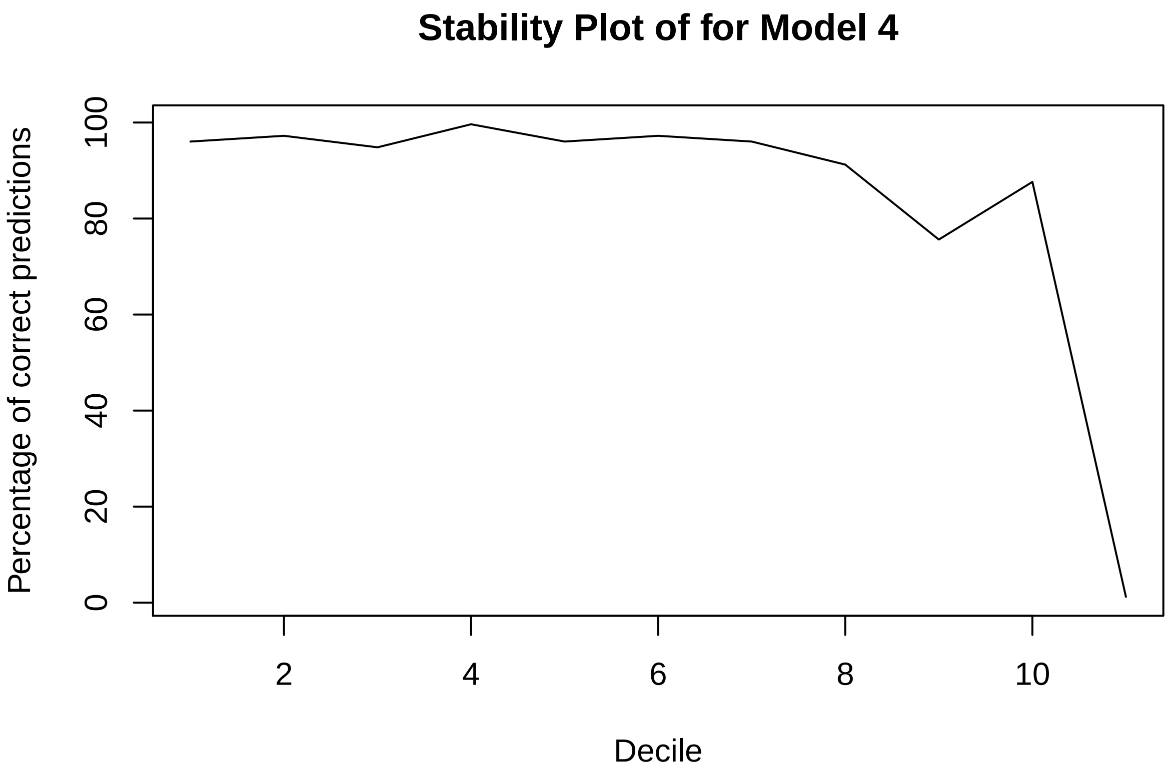
See data below for the stability table of class 0 for model 4.

```
decile.model4
```

```
##    Decile per_correct_preds No_correct_Preds cum_preds
## 1       1          96.03842               80        80
## 2       2          97.23890               81       161
## 3       3          94.83794               79       240
## 4       4          99.63986               83       323
## 5       5          96.03842               80       403
## 6       6          97.23890               81       484
## 7       7          96.03842               80       564
## 8       8          91.23649               76       640
## 9       9          75.63025               63       703
## 10     10          87.63505               73       776
## 11     11           1.20048                1       777
```

Plotting the stability graph for model 2 gives.

```
plot(decile.model4$Decile,
     decile.model4$per_correct_preds,
     type = "l",
     xlab = "Decile",
     ylab = "Percentage of correct predictions",
     main="Stability Plot of for Model 4")
```



Based on the visualization above we can conclude that model 4 is stable.

Evaluation

To evaluate the performance of out four (4) models, we will use accuracy, simplicity, AUC and stability.

From the information above we can summaries these performance metrics.

Performance Evaluation measure for all models can be seen in the table below.

Measure	Description
Simplicity	Number of Significant variable/leaves
AUC	Area Under the Curve
Accuracy	Measures how often the model correctly classifies a customer
Stability	Visual inspection of graph
Sensitivity/Recall	The models ability to correctly classify an customer as churned
Specificity	The model's its ability to designate an customer who has not churned correctly
Precision	Proportion of predicted churned customers that actually churned
F1	Covers the imbalance between precision and recall
Kappa	How better your classifier is performing over a classifier that guesses at random

Models	Accuracy	Specificity	Precision	Sensitivity	F1	Kappa
Model 1	0.864	0.223	0.881	0.973	0.925	0.264
Model 2	0.867	0.207	0.879	0.979	0.926	0.257
Model 3	0.928	0.570	0.931	0.989	0.959	0.658
Model 4	0.935	0.661	0.945	0.982	0.963	0.711

Measure	Value Function	Weight	Threshold
Accuracy	None	0.50	>0.80
Simplicity	See graph below	0.10	>0.75
AUC	None	0.25	>0.80
Stability	Binary:1 for a stable tree;0 otherwise	0.15	>0.60

Simplicity Value Function	Criteria
0	if NoOfLeaves <= 5 or NoOfLeaves >= 25
(NoOfLeaves - 5)/(10 - 5)	if 6 <= NoOfLeaves <= 9
1	if 10 <= NoOfLeaves <= 15
(25 - NoOfLeaves)/(25 -15)	if 16 <= NoOfLeaves<= 24

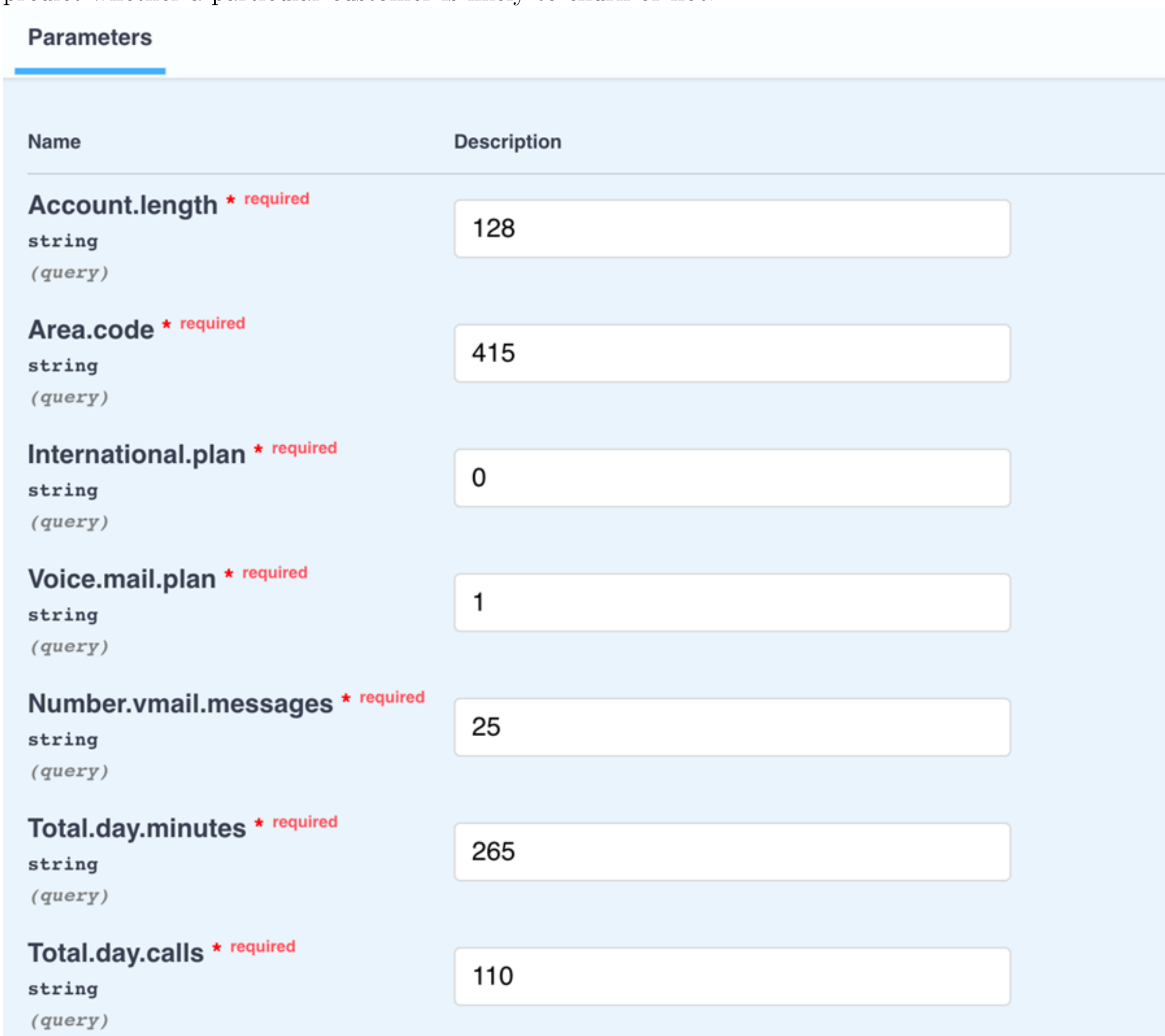
Model	Accuracy	# of leaves/Attributes	Simplicity Score	AUC	Stability	Overall
Model 1	0.864	18	0.7	0.8544	0	0.7156
Model 2	0.867	9	0.8	0.8497	0	0.7259
Model 3	0.928	9	0.80	0.8478	1	0.9060
Model 4	0.935	13	1	0.8989	1	0.9422

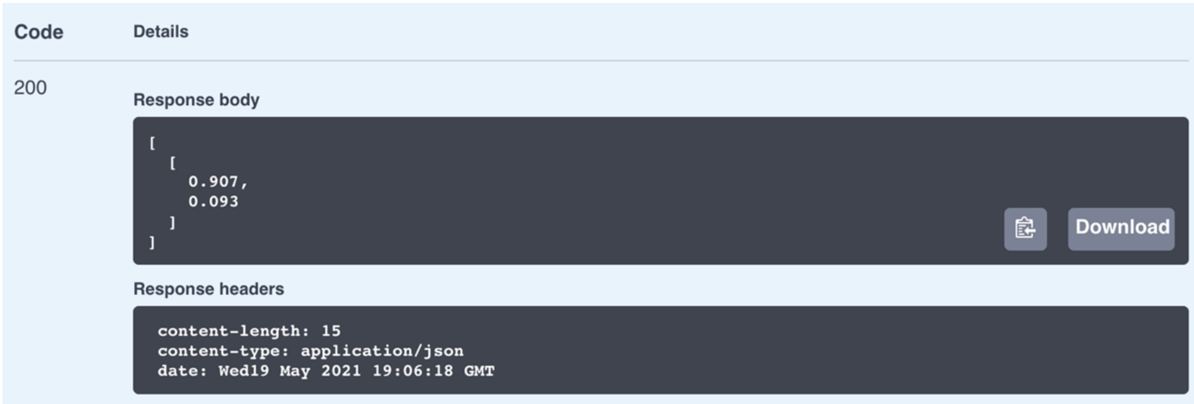
Given that Model 1 simplicity measure falls below the threshold, it has been eliminated for consideration. Since model 4 has the highest overall performance score, this is the model that will be used to fulfill the company's requirements.

Deployment

After selecting the best prediction model through evaluation using different performance measures, the model will be deployed into TeleCom's production environment which will help them to predict customers who will churn in a given month.

The deployment strategy involves saving the machine learning model as an RDS object in R. By using the Plumber package, we can create an HTTP API to be hosted on the company's server that contains a prediction calculator that accepts the full list of parameters, or variable attributes that the model uses to predict whether a particular customer is likely to churn or not.





The output, as represented in the Response body, is equivalent to the output of using the predict function in R with the same model, and identical arguments. In this example, the model predicts that this customer has a 9.3% chance of churning, and a 90.7% chance of remaining with the company.