Capstone - Customer Churn Analysis

Duncan Welch

2024-12-10

Introduction

This is the final project for the HarvardX PH125.9x - Data Science: Capstone class.

Delivering Insights about Customers

The key to business is a happy customer. The ability to generate recurring revenue is key. We will attempt to gain some insights into why customers may leave us (aka attrition or churn). This may provide benefits like (1) how to retain them, (2) basis for better revenue forecasting, or (3) designing performance incentives on attracting more profitable customers.

In this exercise we are working towards maximizing True Negative predictions, customers that did leave, while minimizing False Positives, or customers that did leave, but we missed in our algorithm.

Data Definitions

Our data contains 10,000 observations and 14 variables.

RowNumber — Corresponds to the record (row) number and has no effect on the output.

CustomerId — Contains random values and has no effect on customer leaving.

Surname — The surname of a customer has no impact on their decision to leave.

CreditScore — Can have an effect on customer churn, since a customer with a lower credit score is more likely to leave.

Geography — A customer's location can affect their decision to leave.

Gender — Does gender play a role? Let's explore and see.

Age — Younger customers are more likely to leave their bank than older.

Tenure — The number of years that the customer has been a client. Intuitively, longer tenure means more loyal and less likely to leave.

Balance — A good indicator of customer churn, as people with a higher balance in their accounts are less likely to leave compared to those with lower balances.

NumOfProducts — The number of products that a customer has in their total relationship with the bank.

HasCrCard — Does the customer have a credit card? This column is important since people with a credit card are less likely to leave.

IsActiveMember — Active customers are less likely to leave.

EstimatedSalary — Customers with higher salaries are more likely to stay compared to those with lower salaries.

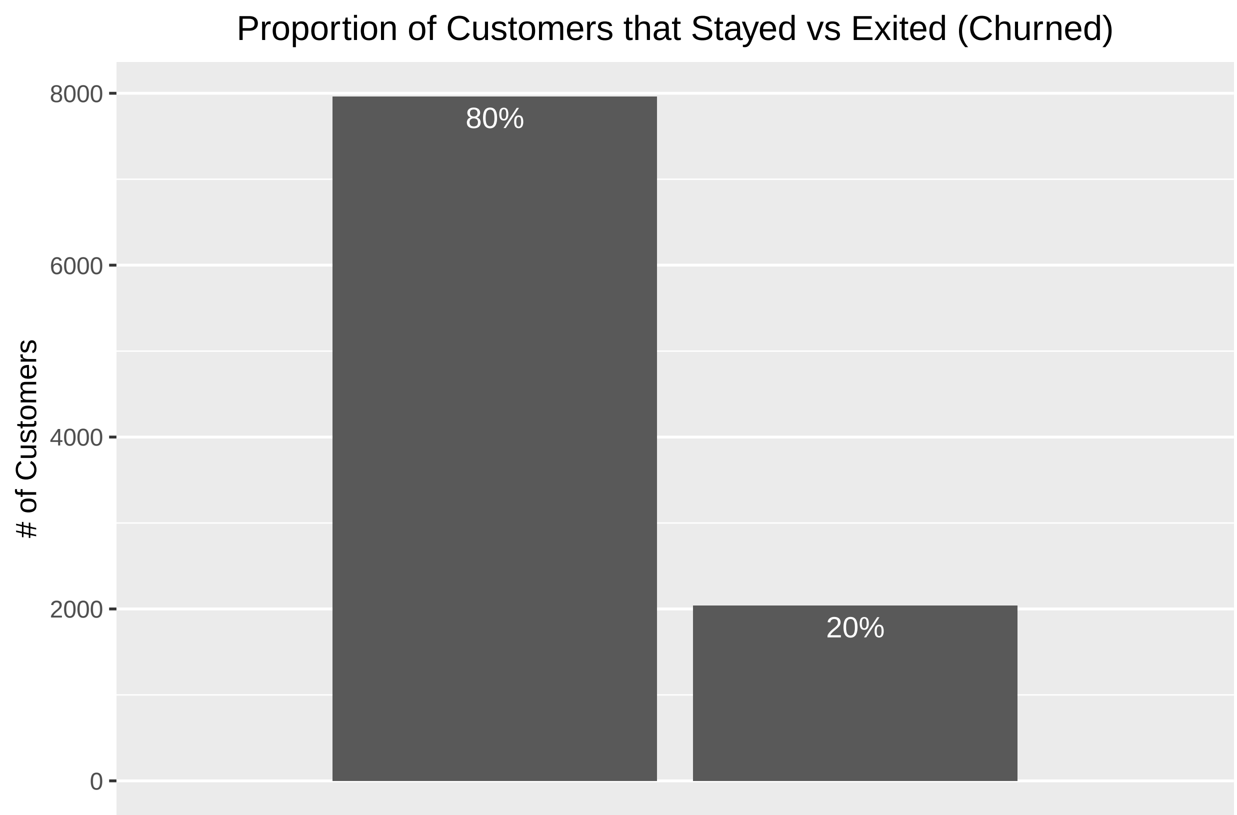
Exited — Has the customer left?

Let's take a look at a glimpse of the data together

```
## Rows: 10,000
## Columns: 14
## $ RowNumber       <dbl> 1839, 9625, 8724, 1632, 8763, 2474, 1963, 1406, 1194, ~
## $ CustomerId      <dbl> 15758813, 15668309, 15803202, 15685372, 15765173, 1567~
## $ Surname         <chr> "Campbell", "Maslow", "Onyekachi", "Azubuike", "Lin", ~
## $ CreditScore     <dbl> 350, 350, 350, 350, 350, 351, 358, 359, 363, 365, 367,~
## $ Geography       <chr> "Germany", "France", "France", "Spain", "France", "Ger~
## $ Gender          <chr> "Male", "Female", "Male", "Male", "Female", "Female", ~
## $ Age             <dbl> 39, 40, 51, 54, 60, 57, 52, 44, 28, 30, 42, 42, 29, 46~
## $ Tenure          <dbl> 0, 0, 10, 1, 3, 4, 8, 6, 6, 0, 6, 7, 4, 6, 0, 8, 1, 5,~
## $ Balance         <dbl> 109733.20, 111098.85, 0.00, 152677.48, 0.00, 163146.46~
## $ NumOfProducts   <dbl> 2, 1, 1, 1, 1, 1, 3, 1, 3, 1, 1, 1, 4, 1, 1, 1, 1, 1, ~
## $ HasCrCard       <dbl> 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, ~
## $ IsActiveMember  <dbl> 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, ~
## $ EstimatedSalary <dbl> 123602.11, 172321.21, 125823.79, 191973.49, 113796.15,~
## $ Exited          <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
```

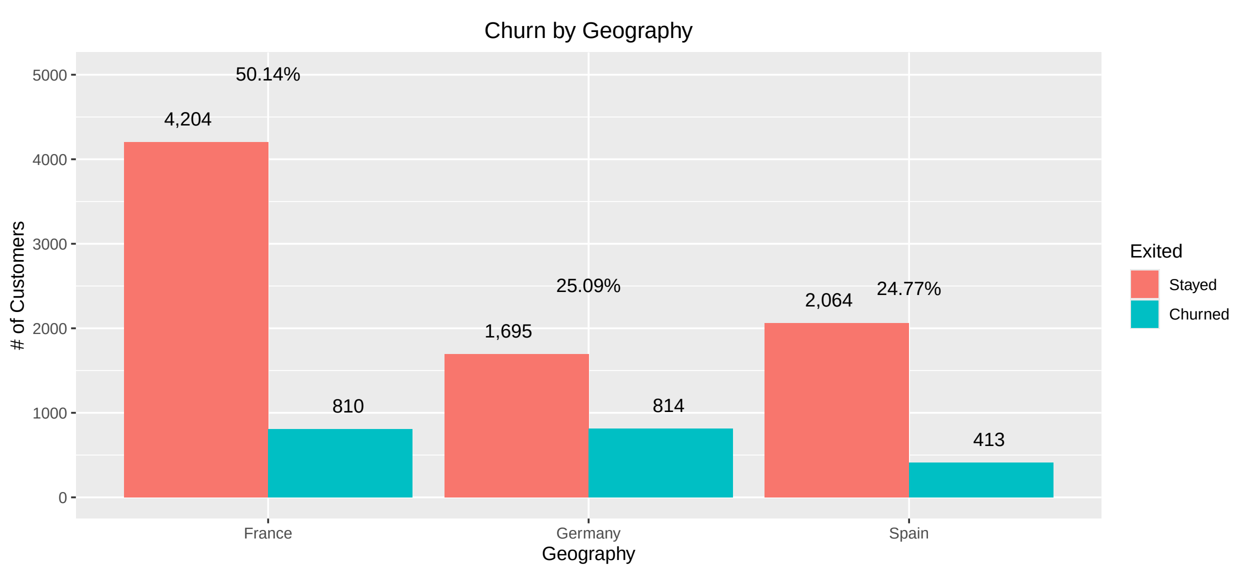
Exploring Data and Analyzing is the First Step to Understanding

The overall proportion of customers who have churned in our data set is approximately 20%. If we didn't know anything at all and just guessed that the majority of our customer's stayed with us we would be right 80% of the time!

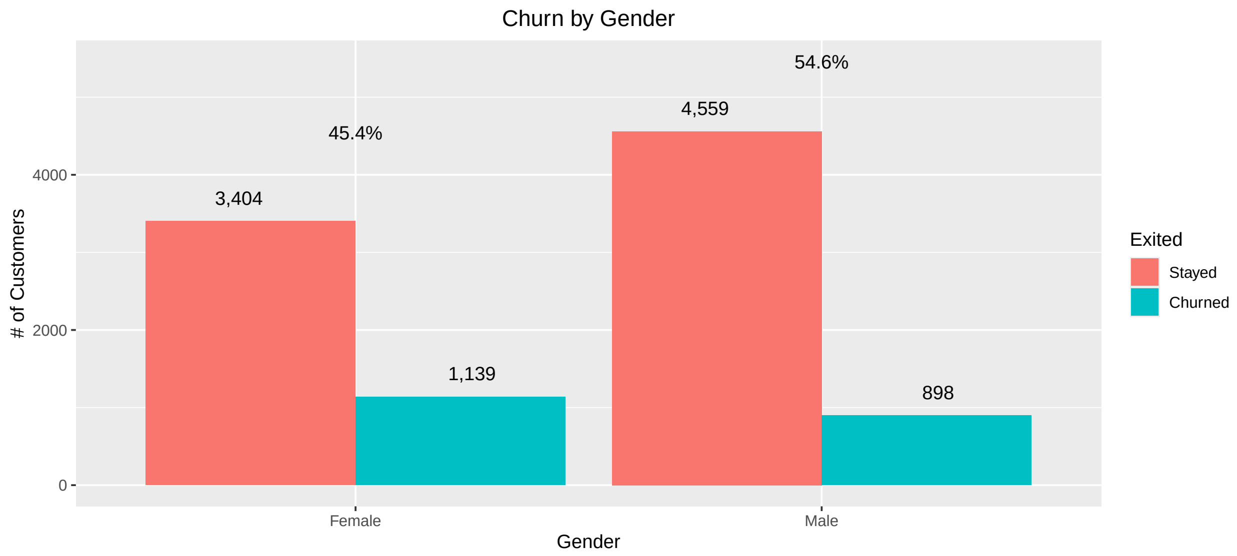


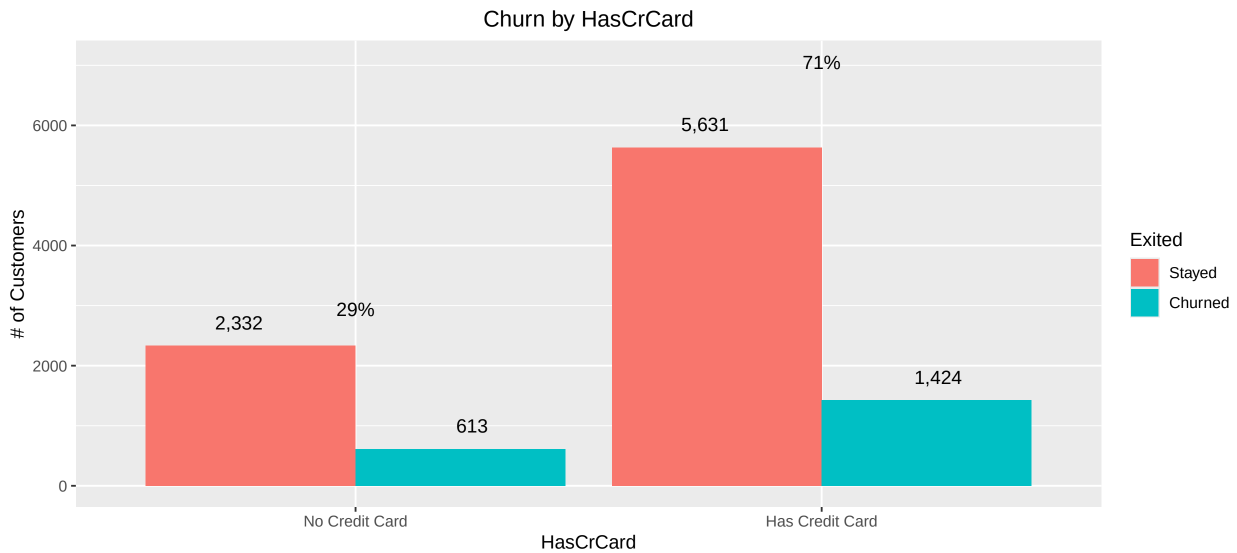
Visual Insights of our Categorical Data

The churn rate for customers in Germany are over 2x that of other countries.

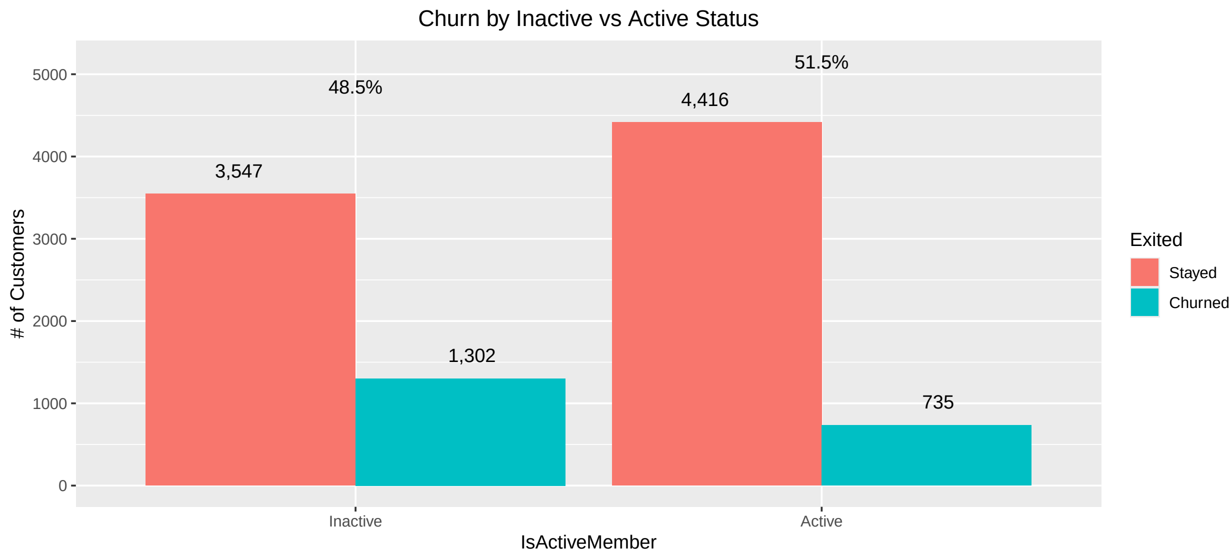


The Male category churn is ~16% (or 44% of total churn) vs the female category churn is ~25% churn (or 56% of total churn)

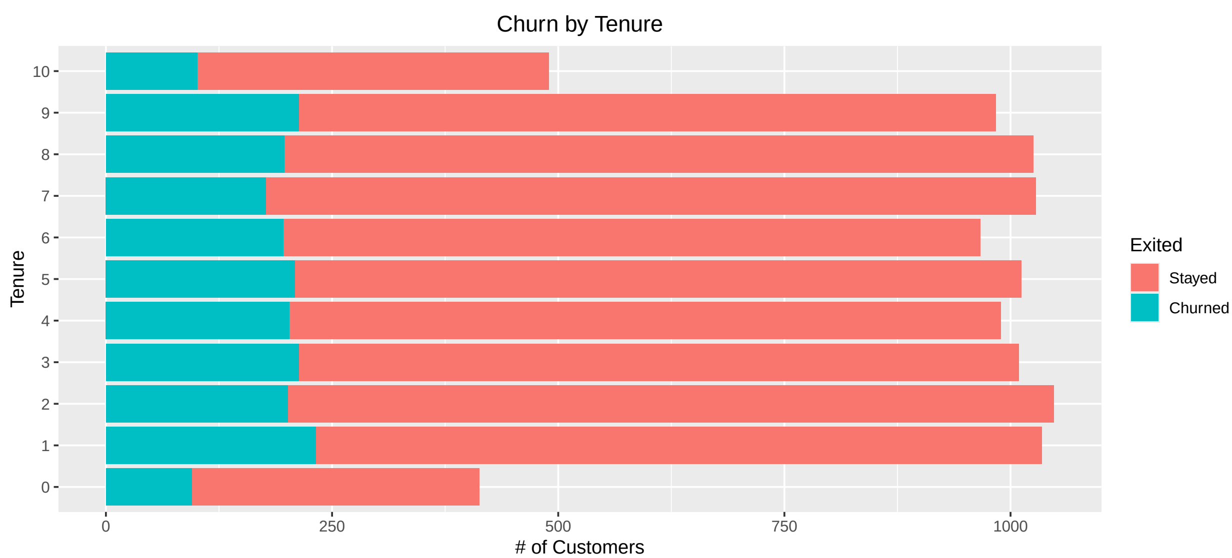




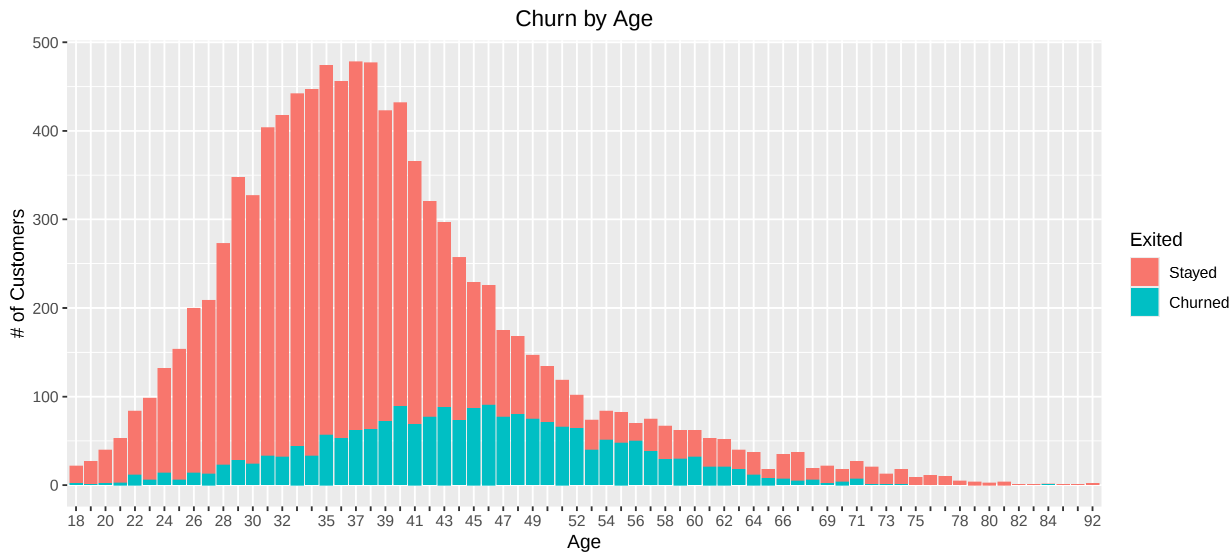
It seems that whether a specific customer has a credit card or not does not matter because the propensity to churn is proportionally even.



Inactive customers have a higher propensity to churn than Active with Active customers being slightly larger at 51.5% of the total population.

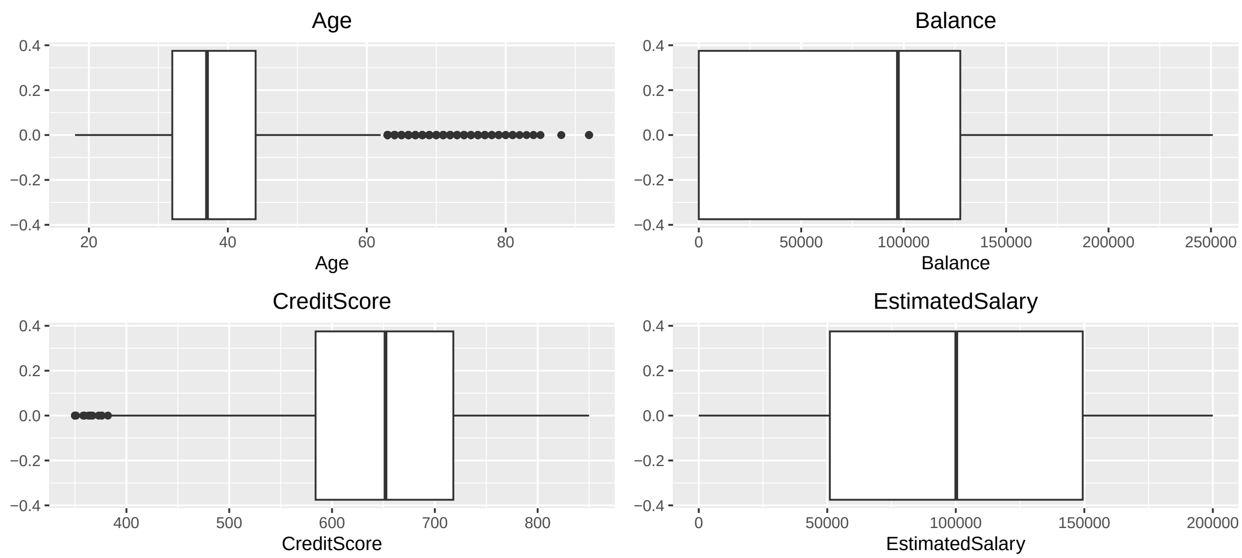


Tenure of 0 and 10 have the lowest churn, while 1 through 9 are similar and relatively flat.



Customers in the age range of 48 and older churn more as a proportion of total.

Reviewing and Identifying Potential Outliers



In reviewing Age, Credit Score, Balance, and Estimated Salary, we can identify that Age and Credit Score have outliers that in turn may impact our results.

Pre-processing of Data

Shown below is a descriptive view of our data

```
## spc_tbl_ [10,000 x 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ RowNumber      : num [1:10000] 1839 9625 8724 1632 8763 ...
##  $ CustomerId     : num [1:10000] 15758813 15668309 15803202 15685372 15765173 ...
##  $ Surname        : chr [1:10000] "Campbell" "Maslow" "Onyekachi" "Azubuike" ...
##  $ CreditScore    : num [1:10000] 350 350 350 350 350 351 358 359 363 365 ...
```

```
##  $ Geography      : chr [1:10000] "Germany" "France" "France" "Spain" ...
##  $ Gender         : chr [1:10000] "Male" "Female" "Male" "Male" ...
##  $ Age            : num [1:10000] 39 40 51 54 60 57 52 44 28 30 ...
##  $ Tenure         : num [1:10000] 0 0 10 1 3 4 8 6 6 0 ...
##  $ Balance        : num [1:10000] 109733 111099 0 152677 0 ...
##  $ NumOfProducts  : num [1:10000] 2 1 1 1 1 1 3 1 3 1 ...
##  $ HasCrCard      : num [1:10000] 0 1 1 1 0 1 1 1 1 1 ...
##  $ IsActiveMember : num [1:10000] 0 1 1 1 0 0 0 0 0 0 ...
##  $ EstimatedSalary: num [1:10000] 123602 172321 125824 191973 113796 ...
##  $ Exited         : num [1:10000] 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   RowNumber = col_double(),
##   ..   CustomerId = col_double(),
##   ..   Surname = col_character(),
##   ..   CreditScore = col_double(),
##   ..   Geography = col_character(),
##   ..   Gender = col_character(),
##   ..   Age = col_double(),
##   ..   Tenure = col_double(),
##   ..   Balance = col_double(),
##   ..   NumOfProducts = col_double(),
##   ..   HasCrCard = col_double(),
##   ..   IsActiveMember = col_double(),
##   ..   EstimatedSalary = col_double(),
##   ..   Exited = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Summary of the variables.

We need to check for data quality, nulls, N/As, blanks, and duplicates. There are no N/As or nulls in the summary so we can move forward.

```
##    RowNumber       CustomerId         Surname           CreditScore
##  Min.   :    1   Min.   :15565701   Length:10000       Min.   :350.0
##  1st Qu.: 2501   1st Qu.:15628528   Class :character   1st Qu.:584.0
##  Median : 5000   Median :15690738   Mode  :character   Median :652.0
##  Mean   : 5000   Mean   :15690941                      Mean   :650.5
##  3rd Qu.: 7500   3rd Qu.:15753234                      3rd Qu.:718.0
##  Max.   :10000   Max.   :15815690                      Max.   :850.0
##   Geography            Gender                Age            Tenure
##  Length:10000       Length:10000       Min.   :18.00   Min.   : 0.000
##  Class :character   Class :character   1st Qu.:32.00   1st Qu.: 3.000
##  Mode  :character   Mode  :character   Median :37.00   Median : 5.000
##                                        Mean   :38.92   Mean   : 5.013
##                                        3rd Qu.:44.00   3rd Qu.: 7.000
##                                        Max.   :92.00   Max.   :10.000
##     Balance       NumOfProducts    HasCrCard      IsActiveMember
##  Min.   :     0   Min.   :1.00   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:     0   1st Qu.:1.00   1st Qu.:0.0000   1st Qu.:0.0000
##  Median : 97199   Median :1.00   Median :1.0000   Median :1.0000
##  Mean   : 76486   Mean   :1.53   Mean   :0.7055   Mean   :0.5151
```

```
##  3rd Qu.:127644   3rd Qu.:2.00   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :250898   Max.   :4.00   Max.   :1.0000   Max.   :1.0000
##  EstimatedSalary         Exited
##  Min.   :    11.58   Min.   :0.0000
##  1st Qu.: 51002.11   1st Qu.:0.0000
##  Median :100193.91   Median :0.0000
##  Mean   :100090.24   Mean   :0.2037
##  3rd Qu.:149388.25   3rd Qu.:0.0000
##  Max.   :199992.48   Max.   :1.0000
```

We now check for duplicates

```
## integer(0)
```

Next, we need to check for columns that are not needed. If the count of unique values is 1, this indicates that there isn't much variation or all the values in the column are constant. These types of columns don't help in predicting the output, so we remove them from the data set. If the count is equal to the number of records then it could negatively impact our model. RowNumber and CustomerID fall in to this scenario. Also, Surname should be removed.

```
##       RowNumber      CustomerId         Surname     CreditScore       Geography
##           10000           10000            2932             460               3
##          Gender             Age          Tenure         Balance   NumOfProducts
##               2              70              11            6382               4
##       HasCrCard  IsActiveMember EstimatedSalary          Exited
##               2               2            9999               2
```

Adjusting the scale for continuous variables.

Standardization becomes important when continuous independent variables are measured at different scales. These variables do not give equal contribution when analyzing the data. The customer segmentation analysis we are performing is attempting to group customers based on their homogeneous attributes. A variable called 'transaction amount' that ranges between \$100 and \$10,000 carries more weighting than 'number of transactions' in a range between 0 and 40. Therefore, we must transform the data to comparable scales to compensate for this. The idea is to re-scale an original variable to have equal (i.e. comparable) range and/or variance.

```r
data$Age = scale(data$Age)
data$CreditScore = scale(data$CreditScore)
data$Balance = scale(data$Balance)
data$EstimatedSalary = scale(data$EstimatedSalary)
data$Tenure = scale(data$Tenure)
```

Converting categorical variables to factors is needed for our models to work by cleaning up the data and converting to usable data types.

```r
data$Geography <-factor(data$Geography)
data$Gender <-factor(data$Gender)
data$HasCrCard <-factor(data$HasCrCard)
data$IsActiveMember <-factor(data$IsActiveMember)
data$ExitedFactor <-data$Exited
data$ExitedFactor <-factor(data$ExitedFactor)
```

Convert Categorical Variables to dummys

```
data = dummy_cols(data,
           select_columns = c("Gender","Geography","HasCrCard","IsActiveMember"),
           remove_selected_columns = TRUE,
           remove_first_dummy = TRUE)
#rename y variable
#data <- data %>%  rename("Exited" = "Exited_1")
```

Check our results to make sure all pre-processing is as desired.

```
summary(data)
```

```
##   CreditScore            Age              Tenure              Balance
##  Min.   :-3.10935   Min.   :-1.9949   Min.   :-1.733229   Min.   :-1.2258
##  1st Qu.:-0.68832   1st Qu.:-0.6600   1st Qu.:-0.695947   1st Qu.:-1.2258
##  Median : 0.01522   Median :-0.1832   Median :-0.004426   Median : 0.3319
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.000000   Mean   : 0.0000
##  3rd Qu.: 0.69807   3rd Qu.: 0.4842   3rd Qu.: 0.687095   3rd Qu.: 0.8199
##  Max.   : 2.06378   Max.   : 5.0609   Max.   : 1.724377   Max.   : 2.7952
##  NumOfProducts  EstimatedSalary         Exited       ExitedFactor
##  Min.   :1.00   Min.   :-1.740181   Min.   :0.0000   0:7963
##  1st Qu.:1.00   1st Qu.:-0.853551   1st Qu.:0.0000   1:2037
##  Median :1.00   Median : 0.001803   Median :0.0000
##  Mean   :1.53   Mean   : 0.000000   Mean   :0.2037
##  3rd Qu.:2.00   3rd Qu.: 0.857200   3rd Qu.:0.0000
##  Max.   :4.00   Max.   : 1.737113   Max.   :1.0000
##   Gender_Male     Geography_Germany Geography_Spain   HasCrCard_1
##  Min.   :0.0000   Min.   :0.0000    Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000    1st Qu.:0.0000   1st Qu.:0.0000
##  Median :1.0000   Median :0.0000    Median :0.0000   Median :1.0000
##  Mean   :0.5457   Mean   :0.2509    Mean   :0.2477   Mean   :0.7055
##  3rd Qu.:1.0000   3rd Qu.:1.0000    3rd Qu.:0.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000    Max.   :1.0000   Max.   :1.0000
##  IsActiveMember_1
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :1.0000
##  Mean   :0.5151
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

```
str(data)
```

```
## tibble [10,000 x 13] (S3: tbl_df/tbl/data.frame)
##  $ CreditScore      : num [1:10000] -3.11 -3.11 -3.11 -3.11 -3.11 ...
##  $ Age              : num [1:10000] 0.00746 0.10281 1.15164 1.43769 2.00978 ...
##  $ Tenure           : num [1:10000] -1.733 -1.733 1.724 -1.387 -0.696 ...
##  $ Balance          : num [1:10000] 0.533 0.555 -1.226 1.221 -1.226 ...
##  $ NumOfProducts    : num [1:10000] 2 1 1 1 1 1 3 1 3 1 ...
##  $ EstimatedSalary  : num [1:10000] 0.409 1.256 0.447 1.598 0.238 ...
##  $ Exited           : num [1:10000] 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ ExitedFactor     : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Gender_Male      : int [1:10000] 1 0 1 1 0 0 0 0 0 1 ...
##  $ Geography_Germany: int [1:10000] 1 0 0 0 0 1 0 0 0 1 ...
##  $ Geography_Spain  : int [1:10000] 0 0 0 1 0 0 1 0 1 0 ...
##  $ HasCrCard_1      : int [1:10000] 0 1 1 1 0 1 1 1 1 1 ...
##  $ IsActiveMember_1 : int [1:10000] 0 1 1 1 0 0 0 0 0 0 ...
```

Split Data for Training and Test Purposes

Our training data set = 80% and test set = 20% of the total population.

```
trainIndex <- createDataPartition(data$Exited, p = 0.8, list = FALSE, times = 1)
train <- data[ trainIndex,]
test  <- data[-trainIndex,]
rm(trainIndex)
```

Splitting helps to smooth out our data when it is imbalanced.

```
data.frame(table(train$Exited))
```

```
##   Var1 Freq
## 1    0 6369
## 2    1 1631
```

If we try the split again but use a different technique it might help to further balance our data set. The SMOTE (Synthetic Minority Oversampling Technique) helps to correct the imbalance. The train and train_smote data sets will be used for the different models in our evaluation.

The SMOTE technique results show our Stayed(0) and Churn (1) is more balanced.

```
train_smote <- smote(ExitedFactor ~.,train, perc.over =20, perc.under =1  )
 data.frame(table(train_smote$Exited))
```

```
##   Var1  Freq
## 1    0 32620
## 2    1 34251
```

```
  data.frame(table(test$Exited))
```

```
##   Var1 Freq
## 1    0 1594
## 2    1  406
```

Modeling Approaches to Drive Better Predictions

Our analysis will take into account the performance of the following models: (1) Generalized Linear, (2) Decisions Tree, and (3) Random Forest.

Finding the balance between false negatives and false positives is key. A false negative would be customers who would not churn that our models picked up and a false positive would be customers who would churn that our models missed. This becomes extremely important when attempting to forecast the financial impact of the efforts of retaining an existing customer vs the cost of acquiring new customers. We will look at a real world example later on to demonstrate this.

Model 1: Logistic Regression Model

We chose the binomial family for GLM because it is generally best for binary data, such as 0 & 1, which is our y value.

In order to achieve the maximum sensitivity and specificity we will use a cutoff of 0.5.

```
##
## Call:
## glm(formula = ExitedFactor ~ . - Exited, family = binomial, data = train)
##
## Coefficients:
##                    Estimate Std. Error z value             Pr(>|z|)
## (Intercept)        -0.80867    0.10904  -7.416     0.00000000000012 ***
## CreditScore        -0.04707    0.03044  -1.546               0.1221
## Age                 0.79682    0.03058  26.054 < 0.0000000000000002 ***
## Tenure             -0.06231    0.03048  -2.045               0.0409 *
## Balance             0.17841    0.03602   4.954     0.00000072852037 ***
## NumOfProducts      -0.10916    0.05352  -2.040               0.0414 *
## EstimatedSalary     0.04565    0.03063   1.490               0.1361
## Gender_Male        -0.52065    0.06127  -8.498 < 0.0000000000000002 ***
## Geography_Germany  0.78358    0.07567  10.356 < 0.0000000000000002 ***
## Geography_Spain     0.01665    0.07981   0.209               0.8348
## HasCrCard_1        -0.05382    0.06652  -0.809               0.4185
## IsActiveMember_1   -1.11903    0.06515 -17.176 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8091.6  on 7999  degrees of freedom
## Residual deviance: 6778.4  on 7988  degrees of freedom
## AIC: 6802.4
##
## Number of Fisher Scoring iterations: 5
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1515  326
##          1   79   80
##
##                Accuracy : 0.7975
##                  95% CI : (0.7792, 0.8149)
##     No Information Rate : 0.797
##     P-Value [Acc > NIR] : 0.4911
##
##                   Kappa : 0.1907
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.1970
##             Specificity : 0.9504
```

```
##          Pos Pred Value : 0.5031
##          Neg Pred Value : 0.8229
##               Precision : 0.5031
##                  Recall : 0.1970
##                      F1 : 0.2832
##              Prevalence : 0.2030
##          Detection Rate : 0.0400
##    Detection Prevalence : 0.0795
##       Balanced Accuracy : 0.5737
##
##        'Positive' Class : 1
##
```

Our P Values show that not all of the variables are statistically significant.

Unfortunately, our results show 80% accuracy, which is what we would also achieve if we choose all customers were not to churn. We correctly classified 80 customers who will churn and missed 326 making our recall 19.7%. We will attempt to provide better results going forward.

Logistic Regression Model: Removing insignificant Variables

Let's see what the results show if we tailor our cutoff to favor recall and run the model again excluding CreditScore, Tenure, Estimate Salary, and Spain.

```
##
## Call:
## glm(formula = Exited ~ . - CreditScore - Tenure - EstimatedSalary -
##     Geography_Spain - HasCrCard_1 - ExitedFactor, family = binomial,
##     data = train)
##
## Coefficients:
##                   Estimate Std. Error z value             Pr(>|z|)
## (Intercept)       -0.83723    0.09562  -8.756 < 0.0000000000000002 ***
## Age                0.79583    0.03055  26.054 < 0.0000000000000002 ***
## Balance            0.17985    0.03597   5.000          0.000000574 ***
## NumOfProducts     -0.10962    0.05341  -2.053               0.0401 *
## Gender_Male       -0.52333    0.06120  -8.551 < 0.0000000000000002 ***
## Geography_Germany  0.77371    0.07077  10.933 < 0.0000000000000002 ***
## IsActiveMember_1  -1.11489    0.06501 -17.150 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8091.6  on 7999  degrees of freedom
## Residual deviance: 6787.9  on 7993  degrees of freedom
## AIC: 6801.9
##
## Number of Fisher Scoring iterations: 5
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction    0    1
##          0 1308  198
##          1  286  208
##
##                Accuracy : 0.758
##                  95% CI : (0.7386, 0.7766)
##     No Information Rate : 0.797
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.308
##
##  Mcnemar's Test P-Value : 0.00007668
##
##             Sensitivity : 0.5123
##             Specificity : 0.8206
##          Pos Pred Value : 0.4211
##          Neg Pred Value : 0.8685
##               Precision : 0.4211
##                  Recall : 0.5123
##                      F1 : 0.4622
##              Prevalence : 0.2030
##          Detection Rate : 0.1040
##    Detection Prevalence : 0.2470
##       Balanced Accuracy : 0.6664
##
##        'Positive' Class : 1
##
```

If you look at recall you can see that We have correctly guessed around 50% of customers that will churn. However, the trade-off is that we are incorrectly guessing non-churn customers as churn in this scenario.

Logistic Regression Model: SMOTE Data Set

Now let's run another GLM with our SMOTE data set and see the impact to recall.

```
##
## Call:
## glm(formula = ExitedFactor ~ . - Exited, family = binomial, data = train_smote)
##
## Coefficients:
##                   Estimate Std. Error z value             Pr(>|z|)
## (Intercept)       -0.50817    0.04344 -11.698 < 0.0000000000000002 ***
## CreditScore       -1.03495    0.01146 -90.347 < 0.0000000000000002 ***
## Age                1.30231    0.01350  96.448 < 0.0000000000000002 ***
## Tenure             0.04674    0.01203   3.885             0.000102 ***
## Balance            0.17144    0.01368  12.533 < 0.0000000000000002 ***
## NumOfProducts     -0.26459    0.02003 -13.207 < 0.0000000000000002 ***
## EstimatedSalary    0.33190    0.01287  25.796 < 0.0000000000000002 ***
## Gender_Male       -0.54883    0.02522 -21.763 < 0.0000000000000002 ***
## Geography_Germany  0.69522    0.03314  20.980 < 0.0000000000000002 ***
## Geography_Spain    0.29804    0.03096   9.628 < 0.0000000000000002 ***
## HasCrCard_1        0.17232    0.02801   6.152       0.000000000765 ***
## IsActiveMember_1  -1.15826    0.02594 -44.660 < 0.0000000000000002 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 92663  on 66870  degrees of freedom
## Residual deviance: 49111  on 66859  degrees of freedom
## AIC: 49135
##
## Number of Fisher Scoring iterations: 5
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1342  233
##          1  252  173
##
##                Accuracy : 0.7575
##                  95% CI : (0.7381, 0.7761)
##     No Information Rate : 0.797
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : 0.2634
##
##  Mcnemar's Test P-Value : 0.4137
##
##             Sensitivity : 0.4261
##             Specificity : 0.8419
##          Pos Pred Value : 0.4071
##          Neg Pred Value : 0.8521
##               Precision : 0.4071
##                  Recall : 0.4261
##                      F1 : 0.4164
##              Prevalence : 0.2030
##          Detection Rate : 0.0865
##    Detection Prevalence : 0.2125
##       Balanced Accuracy : 0.6340
##
##        'Positive' Class : 1
##
```

In an attempt to balance sensitivity and specificity the cutoff is 0. All variables are significant and the recall is 42%.

Model 2: Decision Tree

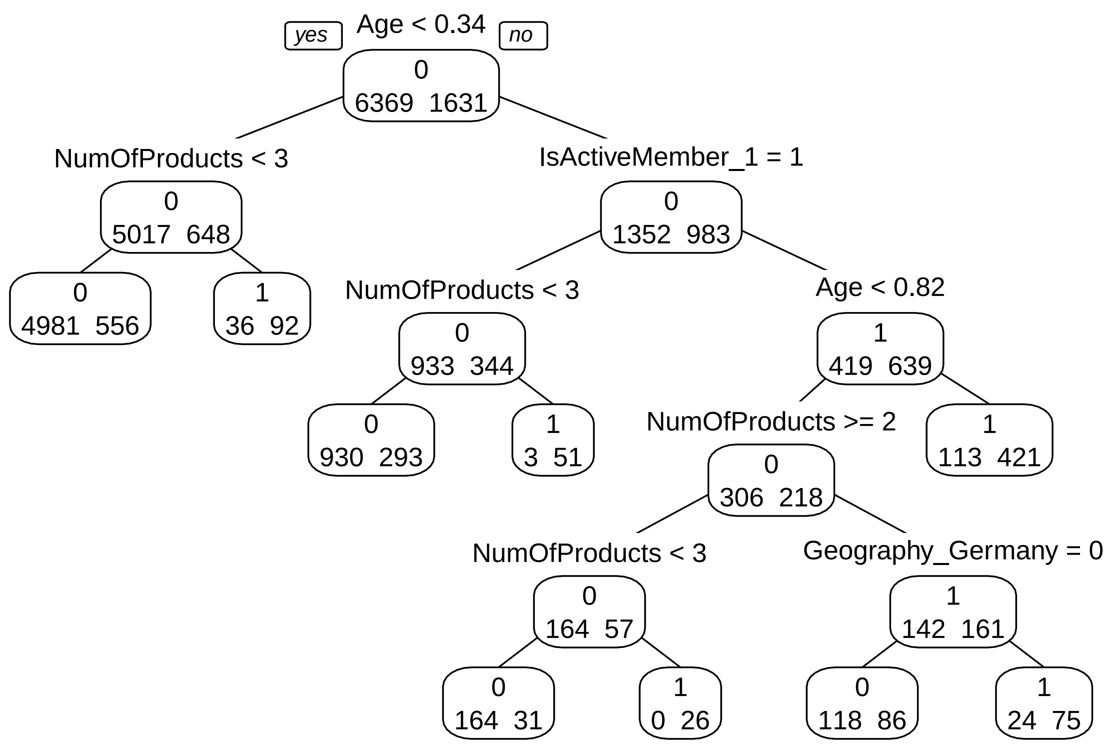
One of the main drivers in Decision Tree Models is the trade-off between tree size and error. This is referred to as Complexity Parameter (CP).

```
##
## Regression tree:
## rpart(formula = Exited ~ . - ExitedFactor, data = train)
```

```
##
## Variables actually used in tree construction:
## [1] Age              IsActiveMember_1 NumOfProducts
##
## Root node error: 1298.5/8000 = 0.16231
##
## n= 8000
##
##         CP nsplit rel error  xerror     xstd
## 1 0.119702      0   1.00000 1.00018 0.016439
## 2 0.049886      1   0.88030 0.88083 0.015198
## 3 0.036838      2   0.83041 0.83104 0.015952
## 4 0.028241      3   0.79357 0.79459 0.015731
## 5 0.019788      4   0.76533 0.76678 0.016053
## 6 0.014380      5   0.74555 0.74706 0.016181
## 7 0.010347      6   0.73117 0.73279 0.015433
## 8 0.010000      7   0.72082 0.72352 0.015459
```

```
##
## Classification tree:
## rpart(formula = Exited ~ . - ExitedFactor, data = train, method = "class")
##
## Variables actually used in tree construction:
## [1] Age               Geography_Germany IsActiveMember_1  NumOfProducts
##
## Root node error: 1631/8000 = 0.20388
##
## n= 8000
##
##         CP nsplit rel error  xerror     xstd
## 1 0.067443      0   1.00000 1.00000 0.022093
## 2 0.053955      2   0.86511 0.86695 0.020919
## 3 0.034335      3   0.81116 0.81116 0.020374
## 4 0.029430      4   0.77682 0.79031 0.020161
## 5 0.015737      5   0.74739 0.74739 0.019708
## 6 0.010000      8   0.70018 0.70018 0.019184
```

```
## [1] 9
```




```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1540  260
##          1   54  146
##
##                Accuracy : 0.843
##                  95% CI : (0.8263, 0.8587)
##     No Information Rate : 0.797
##     P-Value [Acc > NIR] : 0.00000007996
##
##                   Kappa : 0.4017
##
##  Mcnemar's Test P-Value : < 0.00000000000000022
##
##             Sensitivity : 0.3596
##             Specificity : 0.9661
##          Pos Pred Value : 0.7300
##          Neg Pred Value : 0.8556
##               Precision : 0.7300
##                  Recall : 0.3596
##                      F1 : 0.4818
##              Prevalence : 0.2030
##          Detection Rate : 0.0730
##    Detection Prevalence : 0.1000
```

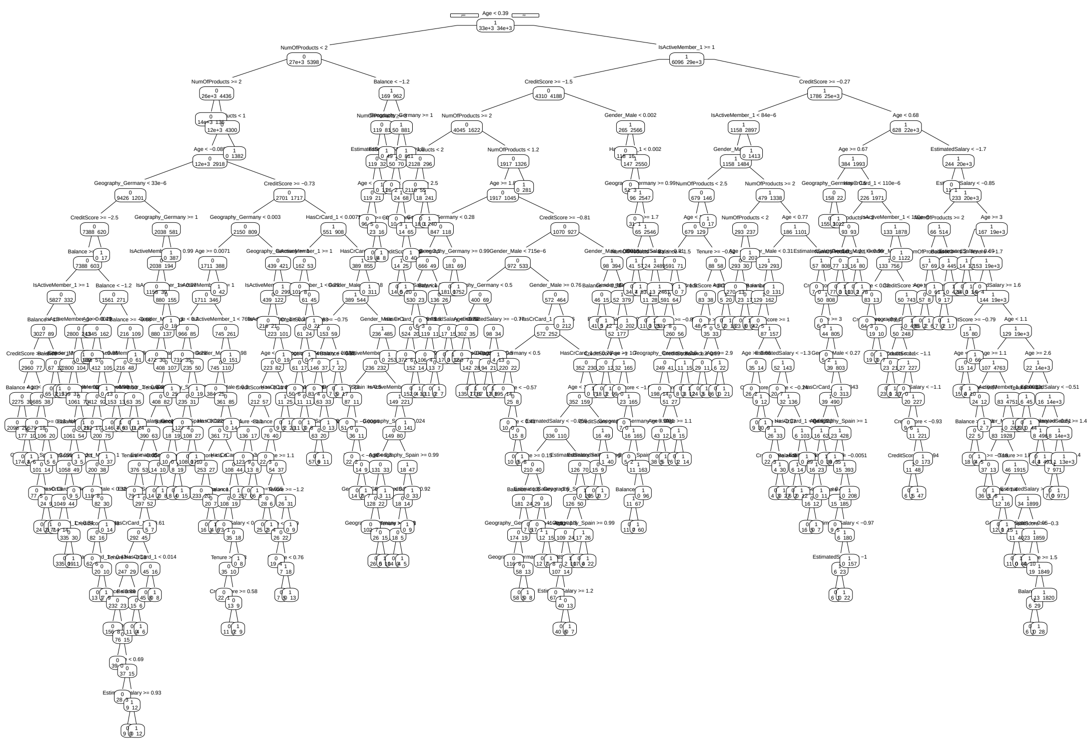
```
##       Balanced Accuracy : 0.6629
##
##        'Positive' Class : 1
##
```

The Decision Tree results show an 84% accuracy, but 35.9% recall, which is low.

Decision Tree: SMOTE Data Set

We are going to use the same approach, but with our balanced SMOTE Data Set.

```
## [1] 290
```



```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1509  256
##          1   85  150
##
##                Accuracy : 0.8295
##                  95% CI : (0.8123, 0.8457)
##     No Information Rate : 0.797
##     P-Value [Acc > NIR] : 0.0001278
```

```
##
##                  Kappa : 0.375
##
##  Mcnemar's Test P-Value : < 0.00000000000000022
##
##            Sensitivity : 0.3695
##            Specificity : 0.9467
##         Pos Pred Value : 0.6383
##         Neg Pred Value : 0.8550
##              Precision : 0.6383
##                 Recall : 0.3695
##                     F1 : 0.4680
##             Prevalence : 0.2030
##         Detection Rate : 0.0750
##   Detection Prevalence : 0.1175
##      Balanced Accuracy : 0.6581
##
##       'Positive' Class : 1
##
```

The Decision Tree - SMOTE Data Set results show an 82.9% accuracy, but 36.9% recall, which is low.

Method 3: Random Forest

The final model we are evaluating is Random Forest. Let's set our cutoff to favor recall and view the results.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1258  105
##          1  336  301
##
##               Accuracy : 0.7795
##                 95% CI : (0.7607, 0.7975)
##    No Information Rate : 0.797
##    P-Value [Acc > NIR] : 0.9749
##
##                  Kappa : 0.4378
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##            Sensitivity : 0.7414
##            Specificity : 0.7892
##         Pos Pred Value : 0.4725
##         Neg Pred Value : 0.9230
##              Precision : 0.4725
##                 Recall : 0.7414
##                     F1 : 0.5772
##             Prevalence : 0.2030
##         Detection Rate : 0.1505
##   Detection Prevalence : 0.3185
##      Balanced Accuracy : 0.7653
```

```
##
##        'Positive' Class : 1
##
```

We achieved a recall of 74%.

Random Forest: SMOTE

Finally, we are going to use our balanced SMOTE Data Set and run the same model.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1326  135
##          1  268  271
##
##                Accuracy : 0.7985
##                  95% CI : (0.7802, 0.8159)
##     No Information Rate : 0.797
##     P-Value [Acc > NIR] : 0.4469
##
##                   Kappa : 0.445
##
##  Mcnemar's Test P-Value : 0.00000000004853
##
##             Sensitivity : 0.6675
##             Specificity : 0.8319
##          Pos Pred Value : 0.5028
##          Neg Pred Value : 0.9076
##               Precision : 0.5028
##                  Recall : 0.6675
##                      F1 : 0.5735
##              Prevalence : 0.2030
##          Detection Rate : 0.1355
##    Detection Prevalence : 0.2695
##       Balanced Accuracy : 0.7497
##
##        'Positive' Class : 1
##
```

At 66.7%, the recall is lower than our previous run.

Conclusion - What Insights Have We Learned?

```
res.cm <- data.frame(RF_CM_smote,RF_CM,DT_CM_smote,DT_CM,LR_CM_Smote,LR_CM_Slim,LR_CM)
res <- data.frame(t(res.cm))
rownames(res) <- colnames(res.cm)
colnames(res) <- rownames(res.cm)
res[,c(7,5,6,2,11,4)] %>%
  arrange(desc(F1))
```

```
##                                 F1 Precision    Recall Specificity
## RandOm_Forest            0.5771812 0.4725275 0.7413793   0.7892095
## RandOm_Forest_Smote      0.5735450 0.5027829 0.6674877   0.8318695
## DecisionTree_Pruned      0.4818482 0.7300000 0.3596059   0.9661230
## DecisionTreeSmote_Pruned 0.4680187 0.6382979 0.3694581   0.9466750
## Logistic_slim            0.4622222 0.4210526 0.5123153   0.8205772
## Logistic_Smote           0.4163658 0.4070588 0.4261084   0.8419072
## Logistic                 0.2831858 0.5031447 0.1970443   0.9504391
##                          Balanced Accuracy Neg Pred Value
## RandOm_Forest                    0.7652944      0.9229640
## RandOm_Forest_Smote              0.7496786      0.9075975
## DecisionTree_Pruned              0.6628644      0.8555556
## DecisionTreeSmote_Pruned         0.6580666      0.8549575
## Logistic_slim                    0.6664462      0.8685259
## Logistic_Smote                   0.6340078      0.8520635
## Logistic                         0.5737417      0.8229223
```

```
#specificity = True Neg / True Neg + F
```

The Random Forest Model gives the highest recall of 74%. This means that we accurately predicted 74% of customers that ultimately churned. If we are willing to believe our predictions and the driving indicators, then we could proactively reach out and attempt to save some of those that ordinary end up leaving.

In Practice

Let's assume the bank is trying to make a decision on which, if any, specific customers to target to improve their churn rate of 20%. The retention cost is $100/customer, while acquiring new customers is 5x times that, or $500). One option would be to launch a campaign for all customers in our test set which would cost the bank $200,000 (2,000 customers x $100 retention per customer). The figure below compares each model with acquisition cost for the false positive and retention cost for the true negatives and false negatives.

Confusion matrix for reference.

		Actual	
		Stay (0)	**Churn (1)**
Prediction	**Stay (0)**	**TRUE POSITIVE** Correctly predicted Stay	**FALSE POSITIVE** Predicted Stay, but they actually Churned
	Churn (1)	**FALSE NEGATIVE** Predicted Churn, but they actually Stayed	**TRUE NEGATIVE** Correctly predicted Churn

Logistic Regression Model

		Actual			
		Stay (0)	Churn (1)		
Prediction	Stay (0)	1515	326	Acq Cost (326 * $500)	$163,000
	Churn (1)	79	80	Ret Cost (159 * $500)	$15,900
					$178,900

Logistic Regression Model: Removing insignificant Variables

		Actual			
		Stay (0)	Churn (1)		
Prediction	Stay (0)	1308	198	Acq Cost (198 * $500)	$99,000
	Churn (1)	286	208	Ret Cost (494 * $500)	$49,400
					$148,400

Logistic Regression Model: SMOTE

		Actual			
		Stay (0)	Churn (1)		
Prediction	Stay (0)	1342	233	Acq Cost (233 * $500)	$116,500
	Churn (1)	252	173	Ret Cost (425 * $500)	$42,500
					$159,000

Decision Tree

		Actual			
		Stay (0)	Churn (1)		
Prediction	Stay (0)	1540	260	Acq Cost (260 * $500)	$130,000
	Churn (1)	54	146	Ret Cost (200 * $500)	$20,000
					$150,000

Decision Tree: SMOTE

		Actual			
		Stay (0)	Churn (1)		
Prediction	Stay (0)	1509	256	Acq Cost (256 * $500)	$128,000
	Churn (1)	85	150	Ret Cost (235 * $500)	$23,500
					$151,500

Random Forest

		Actual			
		Stay (0)	Churn (1)		
Prediction	Stay (0)	1258	105	Acq Cost (105 * $500)	$52,500
	Churn (1)	336	301	Ret Cost (637 * $500)	$63,700
					$116,200

Random Forest: SMOTE

		Actual			
		Stay (0)	Churn (1)		
Prediction	Stay (0)	1326	135	Acq Cost (135 * $500)	$67,500
	Churn (1)	268	271	Ret Cost (539 * $500)	$53,900
					$121,400

The most cost effective approach, which also gave us the best recall is the Random Forest. It would be advisable to use the Random forest and even offers 42% savings compared to marketing to the entire customer base. The retention and acquisition costs are made up, but the ratio of 5 to 1 is generally the spread on the two.

In a real world scenario we could be taking about millions of dollars and hundred of thousands of customers which would make the 42% savings a win.