# Final Project Data Scientist HarvardX Program / Drivers Churn in a ride-hailing APP

Facundo Armentano

2/22/2021

### Case of Study: Drivers churn in a ride-hailing APP

## 1. Context

The present project has the goal of studying drivers churn in a ride-hailing APP in order to know if through Machine Learning (ML from now on) there is an opportunity to reduce churn and make bonus budget more efficient.

The dataset used for the analysis, is a database of drivers working for a ride-hailing APP during a complete month in a city of latin america.

Drivers have been anonymized and there is no information of the month or year, or the country or region from where the data comes from.

The problem is about predicting whether a driver will drive drive on the current month (m) or he will churn using as predictors some business metrics of the previous month (m-1) that will be explained in further detail later.

The objective is to predict churn (YES or NO), and get a better understanding of drivers motivations, in order to set a bonus or other incentive to prevent churn.

## 2. Dataset

```r
summary(dc)
```

```
##  DriverTenure        FleetRole          DriverType              DO
##  Length:8617        Length:8617        Length:8617        Min.   :  1.0
##  Class :character   Class :character   Class :character   1st Qu.: 11.0
##  Mode  :character   Mode  :character   Mode  :character   Median : 68.0
##                                                           Mean   :107.1
##                                                           3rd Qu.:179.0
##                                                           Max.   :690.0
##   WorkingHours       AVGTicket            Cost          LoadFactor
##  Min.   :  0.24   Min.   : 45.89   Min.   :    75   Min.   :0.0140
##  1st Qu.: 10.84   1st Qu.:200.97   1st Qu.:  1897   1st Qu.:0.4962
##  Median : 51.26   Median :256.20   Median : 13678   Median :0.6700
##  Mean   : 75.64   Mean   :258.62   Mean   : 23036   Mean   :0.6061
##  3rd Qu.:123.32   3rd Qu.:295.05   3rd Qu.: 38633   3rd Qu.:0.7489
```

```
##  Max.   :472.51   Max.   :996.79   Max.   :165159   Max.   :0.9886
##  FrequentMoment       CorpProd         FrequentZone          Churn
##  Length:8617        Length:8617        Length:8617        Length:8617
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
```

```
## [1] "We see the dataset contains 8617 rows"
```

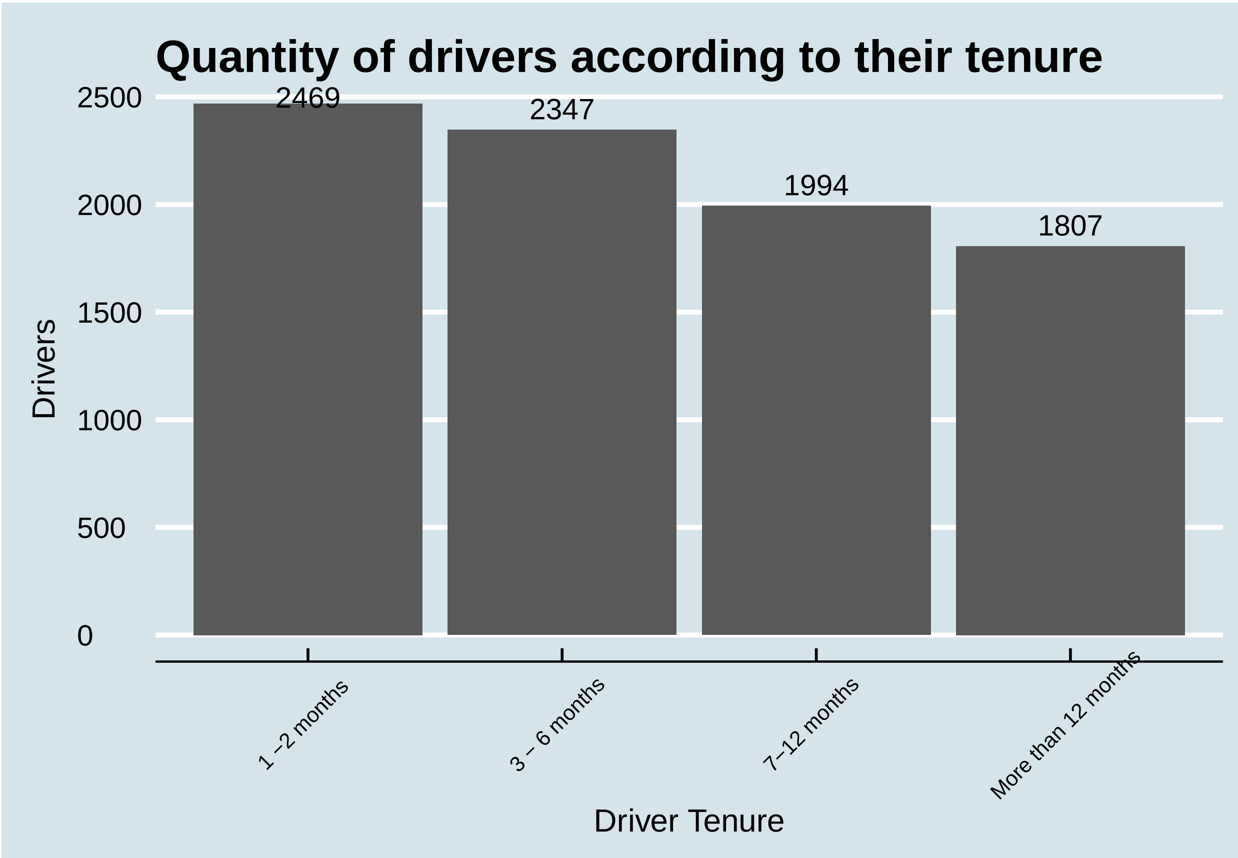
```
## [1] "And 12 columns with 11 predictors for Churn"
```

# 3. Variables description

In the present section, all the features (predictors) will be explained and shown in an intuitive way for a better understanding of the data.

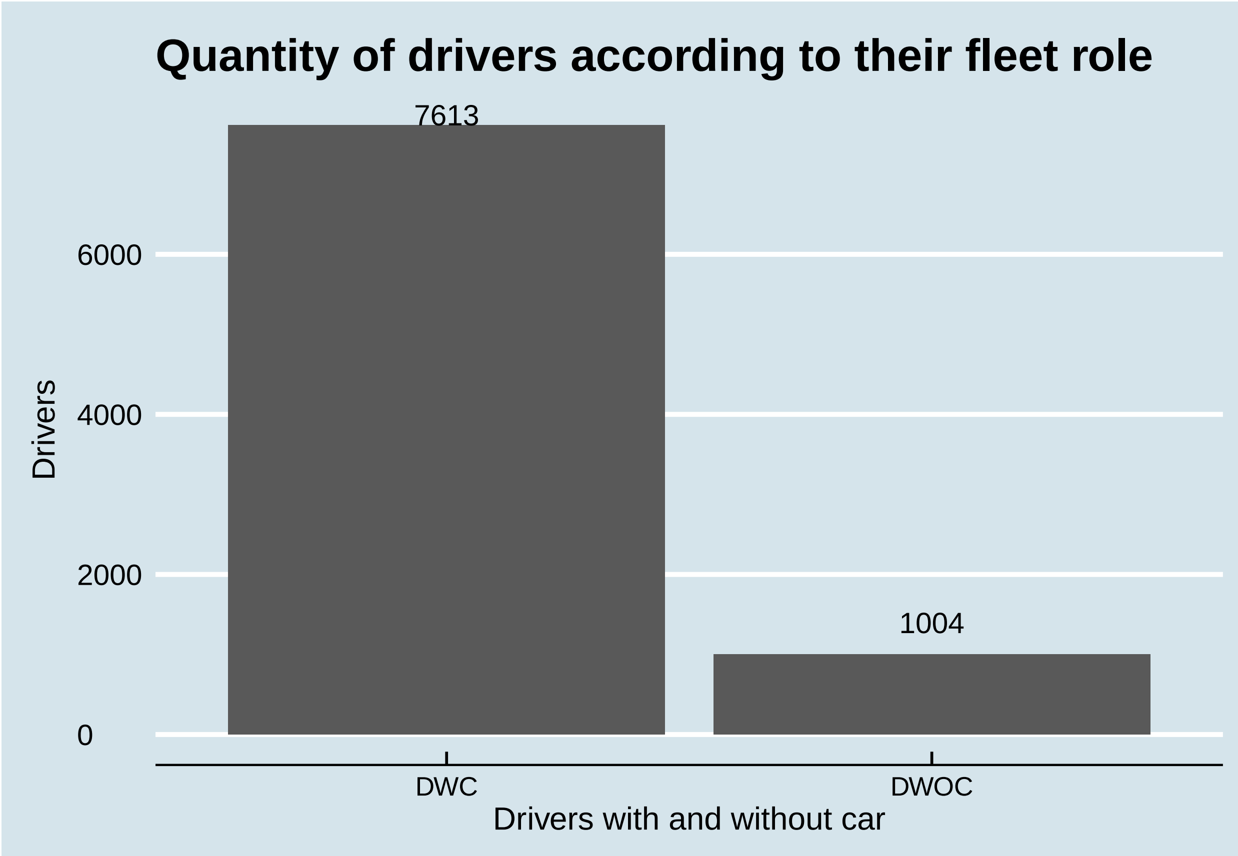
3.1. Driver Tenure makes reference to the time the driver has been working with the APP:

```
## # A tibble: 4 x 3
##   DriverTenure        Drivers  Prop
##   <chr>                 <int> <dbl>
## 1 1 -2 months            2469 0.287
## 2 3 - 6 months           2347 0.272
## 3 7-12 months            1994 0.231
## 4 More than 12 months    1807 0.210
```




We see similar proportions in the four categories, but decreasing over time.

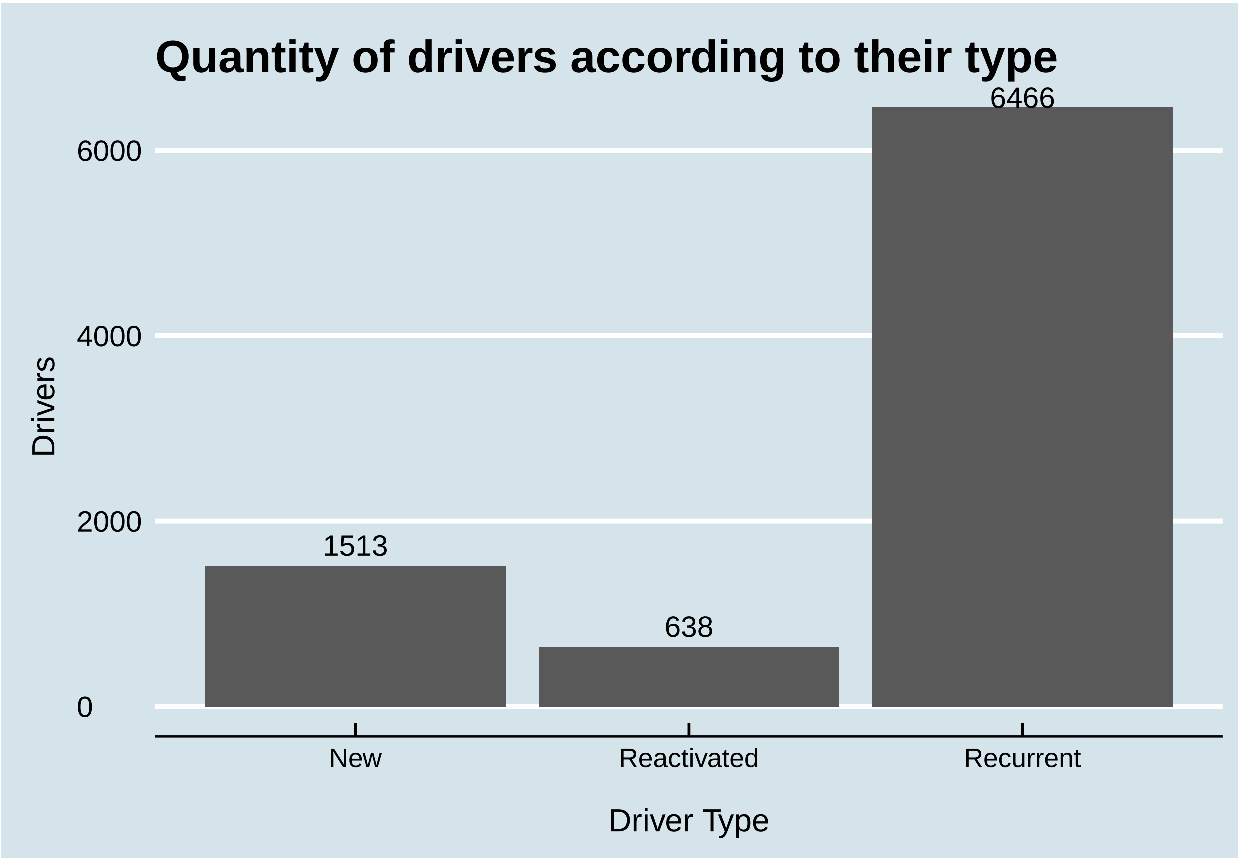
3.2. Fleet role is a binary variable in which is separated whether the driver is the owner of the car or not:




```
## # A tibble: 2 x 3
##   FleetRole Drivers  Prop
##   <chr>       <int> <dbl>
## 1 DWC          7613 0.883
## 2 DWOC         1004 0.117
```

Almost 9 out of 10 drivers own their car.

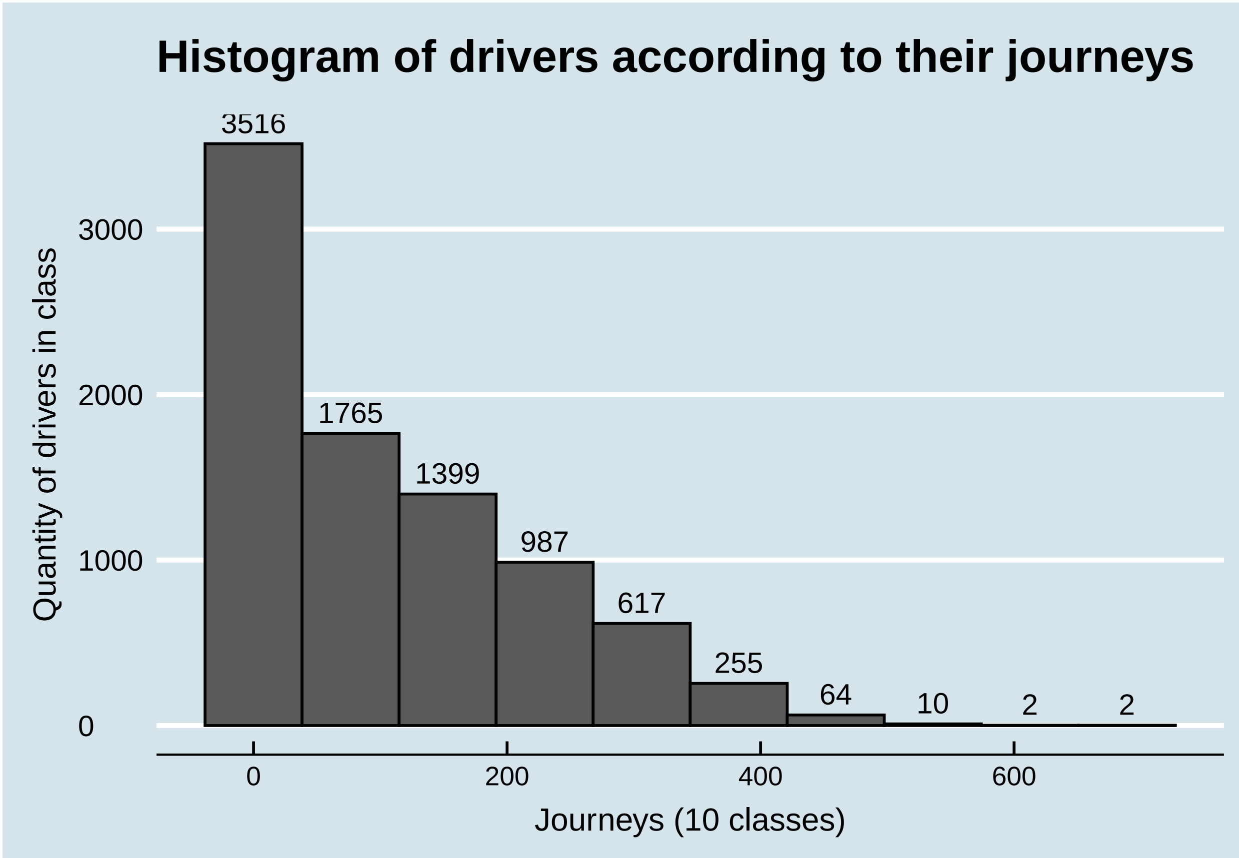
3.3. Driver type has three possible outcomes: New: The ones driving in their first month. Recurrent: Drivers with journeys in m - 1. Reactivated: Drivers without journeys in m -1 but with previous journeys.



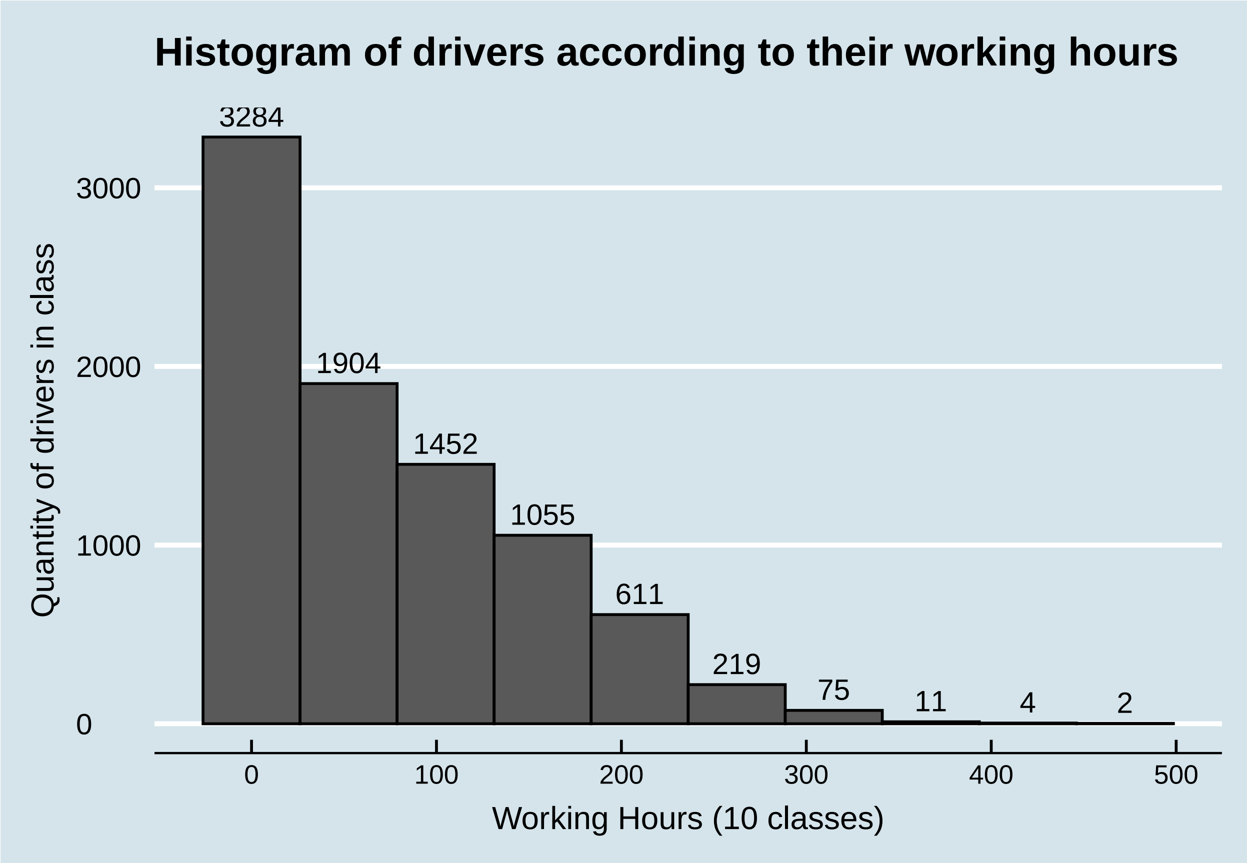

```
## # A tibble: 3 x 3
##   DriverType  Drivers   Prop
##   <chr>         <int>  <dbl>
## 1 New            1513 0.176
## 2 Reactivated     638 0.0740
## 3 Recurrent      6466 0.750
```

75% of drivers are recurrent.

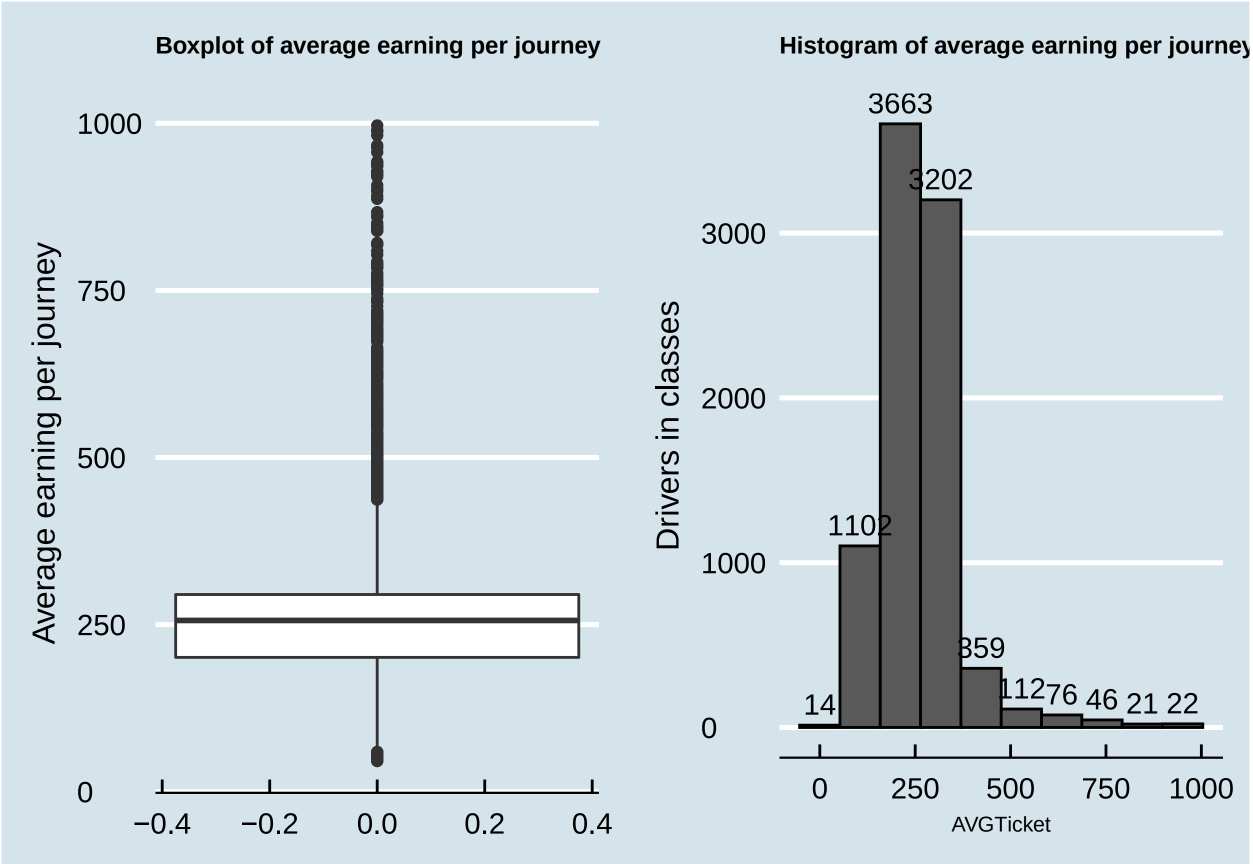
3.4. DO: is the quantity of journeys completed by the driver in m-1.



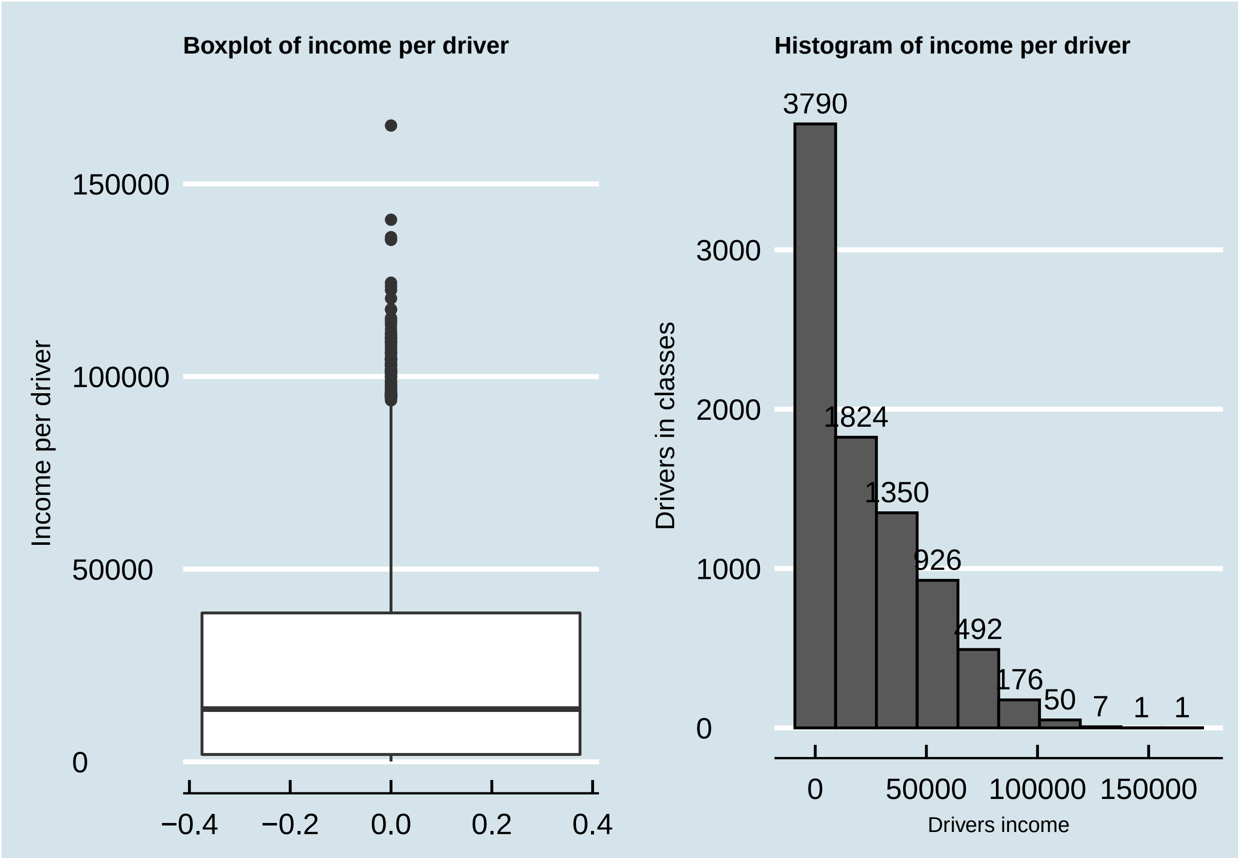

3.5. Working Hours are the amount of hours of connection the driver had in the month in m-1.



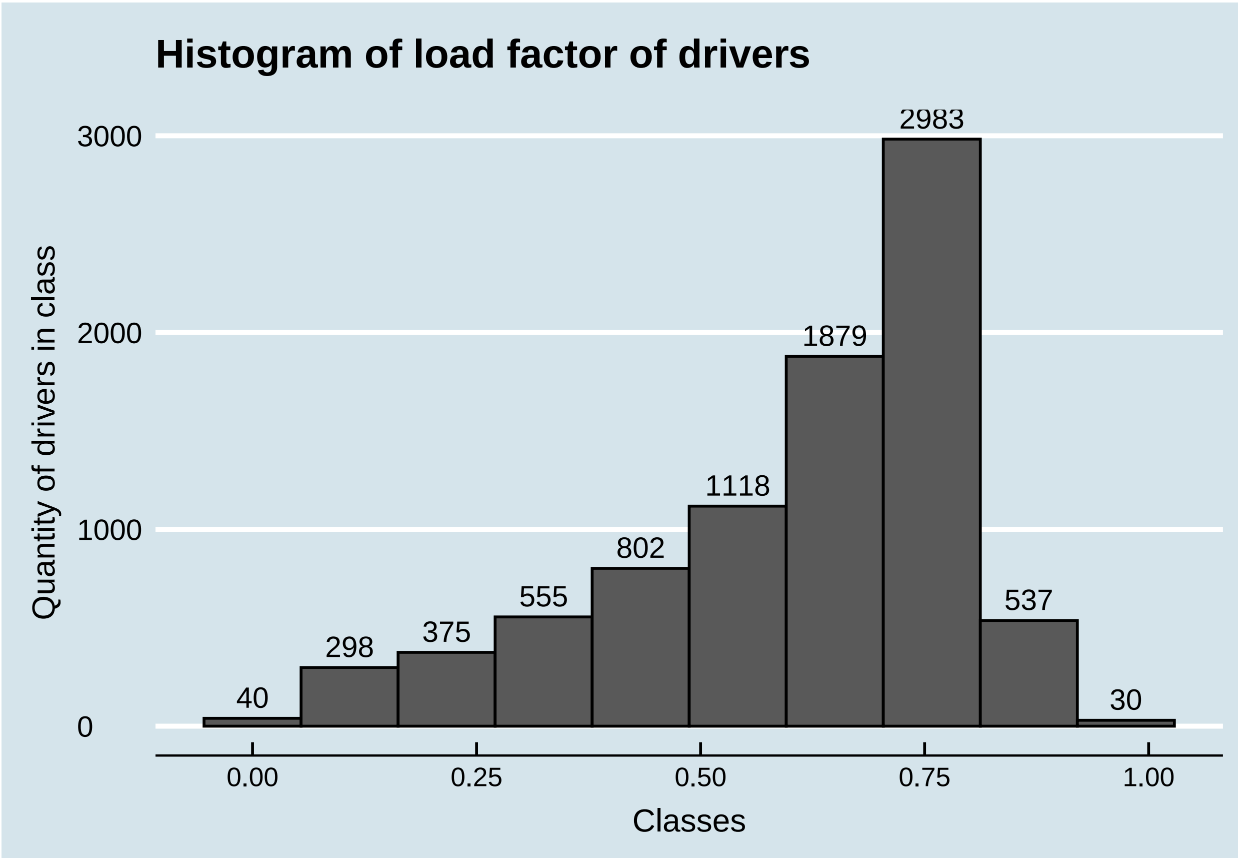

3.6. AVG Ticket refers to the mean of the income the driver got by journey in m-1.



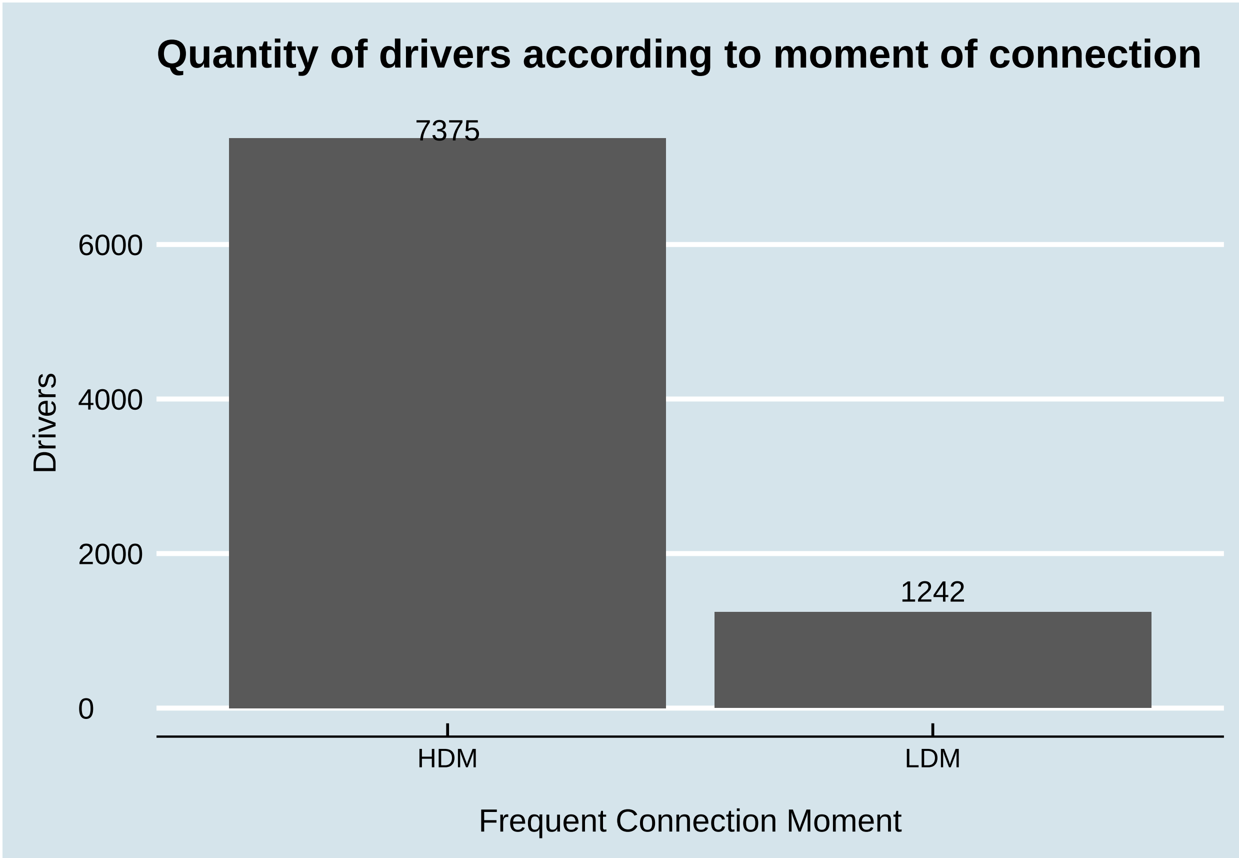

3.7. Cost refers to the total or earnings the driver had in m-1.




3.8. Load factor is a number between 0 and 1 that represents the proportion that the driver was in journey in m-1.

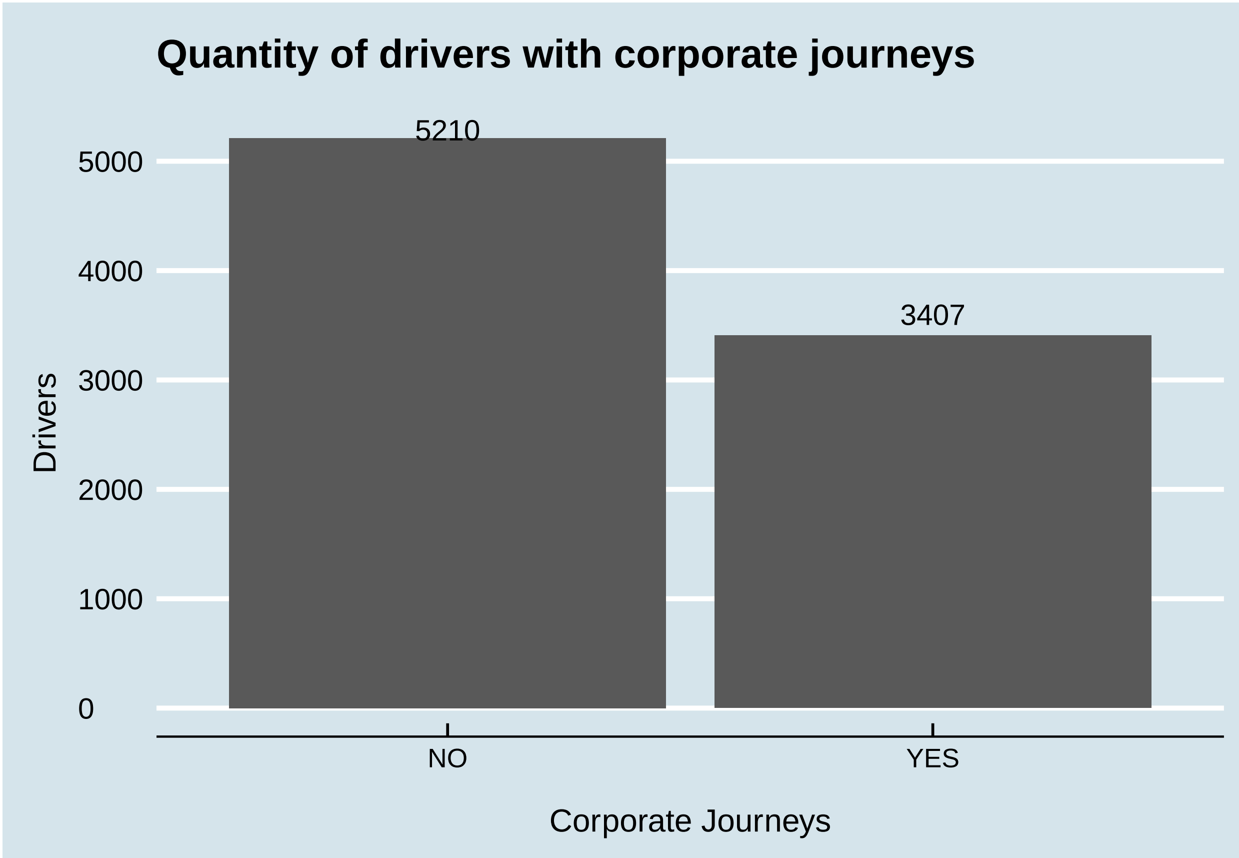



3.9. Frequent Moment is a binary variable that is HDM (High Demand Moment) if the driver was 50% or more of his connected time in High Demand moments and LDM (Low Demand Moments) in any other way.



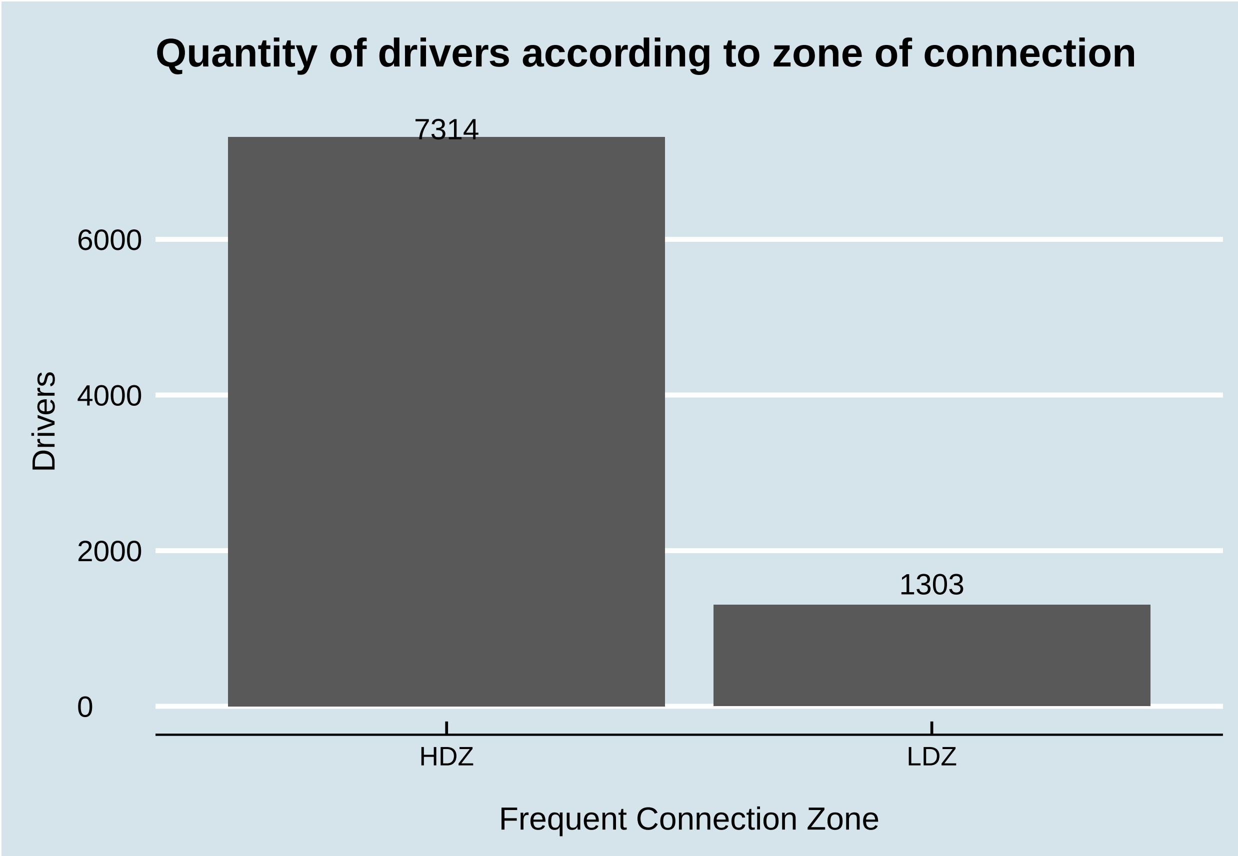

```
## # A tibble: 2 x 3
##   FrequentMoment Drivers  Prop
##   <chr>            <int> <dbl>
## 1 HDM               7375 0.856
## 2 LDM               1242 0.144
```

3.10. CorpProd is a binary variable that is YES if driver is eligible for corporate journeys.




```
## # A tibble: 2 x 3
##   CorpProd Drivers  Prop
##   <chr>      <int> <dbl>
## 1 NO          5210 0.605
## 2 YES         3407 0.395
```

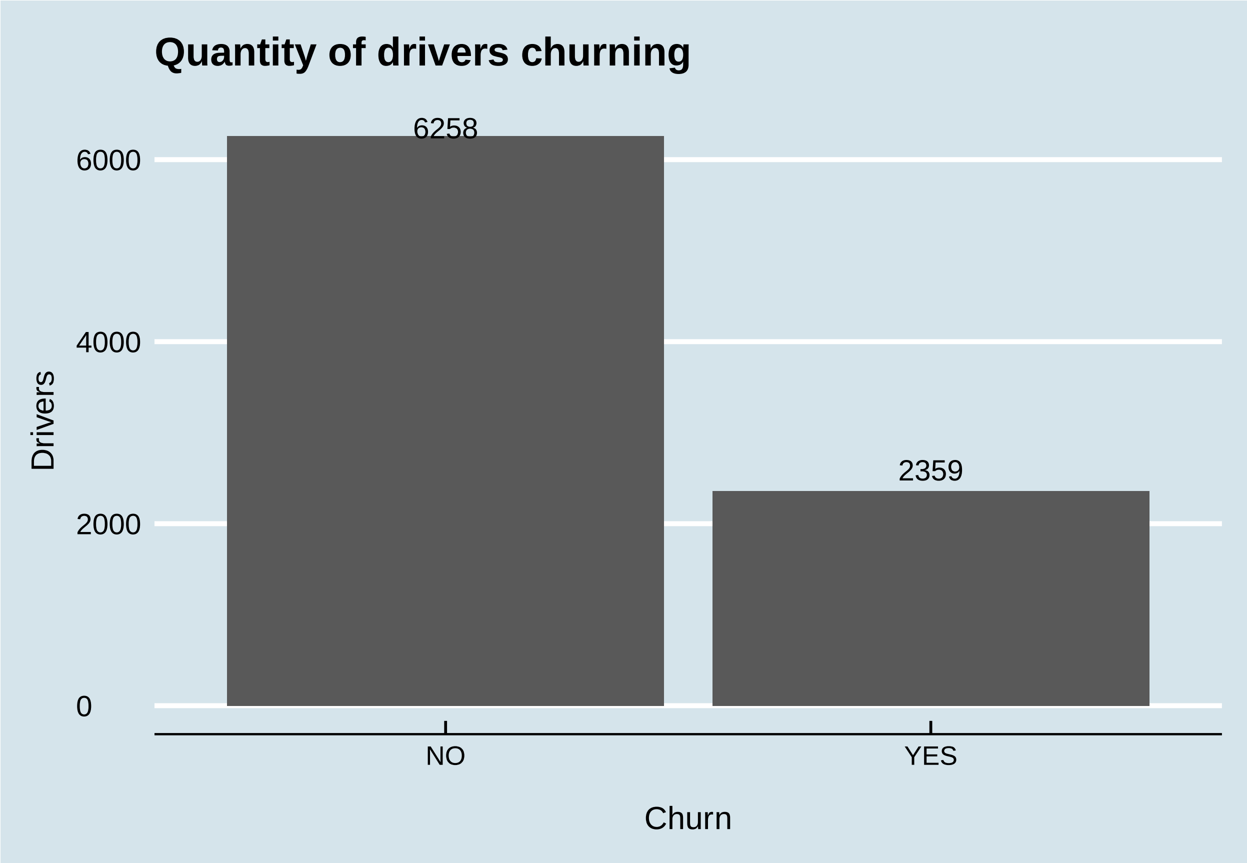
3.11. Frequent Zone is binary variable that is HDZ (High Demand Zone) if the 50% or more of the driver's connected time, the driver does it in a High Demand zone and LDZ (Low Demand Zone) in any other way.




```
## # A tibble: 2 x 3
##   FrequentZone Drivers  Prop
##   <chr>          <int> <dbl>
## 1 HDZ             7314 0.849
## 2 LDZ             1303 0.151
```

## 4. Dataset Prevalence

Studying the dataset, we see that more than 70% of drivers are not churners:



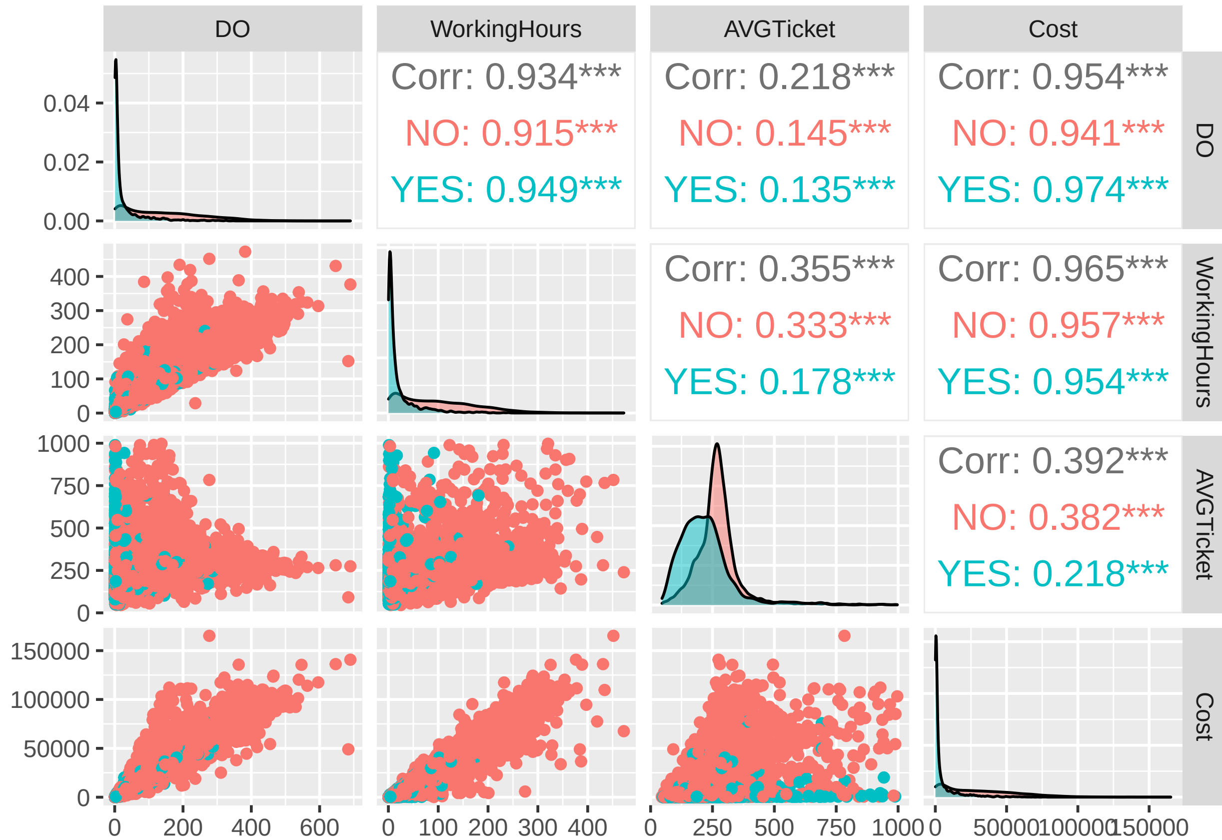

```
## # A tibble: 2 x 3
##   Churn Drivers  Prop
##   <chr>   <int> <dbl>
## 1 NO       6258 0.726
## 2 YES      2359 0.274
```

This is intuitive, but having an unbalanced dataset may bring problems to predictions.

For this reason, at the end of the project, all algorithms will run for a balanced dataset to see if there are differences in Accuracy, Sensitivity and Specificity.

## 5. Numeric variables and Churn

In a first approach, it is interesting to see if there is a correlation between numerical variables, and if there is a relationship with Churn:




Churners are in general more likely to have less DO, Working Hours, AVG Ticket and Income in m-1.

## 6. Create data partition

The next step in the analysis is to create a partition of the data to train different models. In this case only 10% will be separated for testing our models.

As a result, *we split data in two*:

```
## [1] "Training Set 7755 rows"
```

```
## [1] "Training Set 862 rows"
```

## 7. Models

The models to be trained in order to study Churn are: 1. Linear Discriminant Analysis (lda) 2. Quadratic Discriminant Analysis (qda) 3. Logistic Regression (glm) 4. k-Nearest Neighbor Classification with Bootstrap (knn_bootstrap) 5. k-Nearest Neighbor Classification with Cross-validation (knn_crosvalidation) 6. Classification Tree Model (ctm) 7. Random Forests (rf) 8. Naive Bayes (naive_bayes) 9. Support Vector Machine (svmLinear) 10. Generalized Additive Models - Locally Estimated Scatterplot Smoothing (gamLoess) 11. Multinomial Logistic Regression (multinom)

7.1. Linear Discriminant Analysis (lda)

```
## [1] "Accuracy LDA 0.7942"
```

7.2. Quadratic Discriminant Analysis (qda)

```
## [1] "Accuracy QDA 0.6913"
```

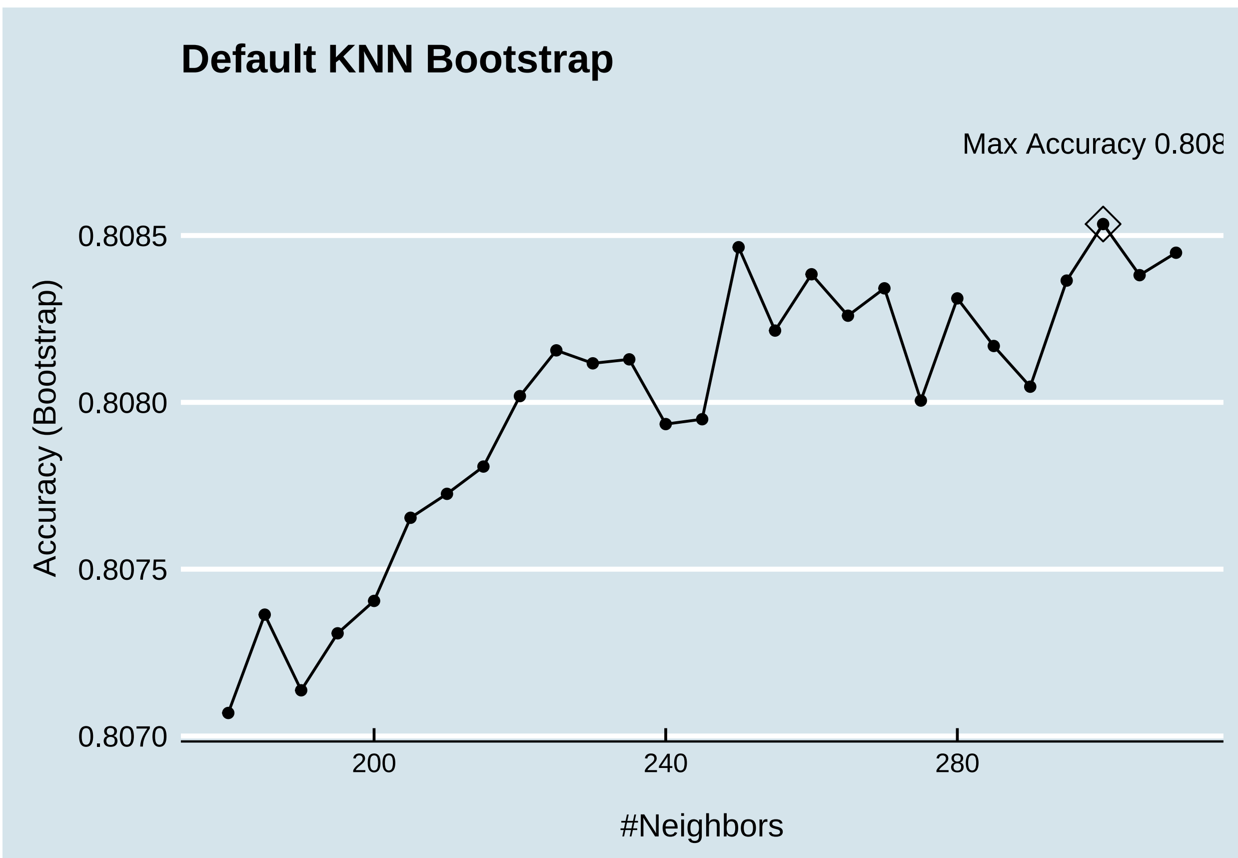
7.3. Logistic Regression (glm)

```
## [1] "Accuracy GLM 0.8039"
```

7.4. k-Nearest Neighbor Classification with Bootstrap (knn_bootstrap)

```
## [1] "Optimal NN 300"
```

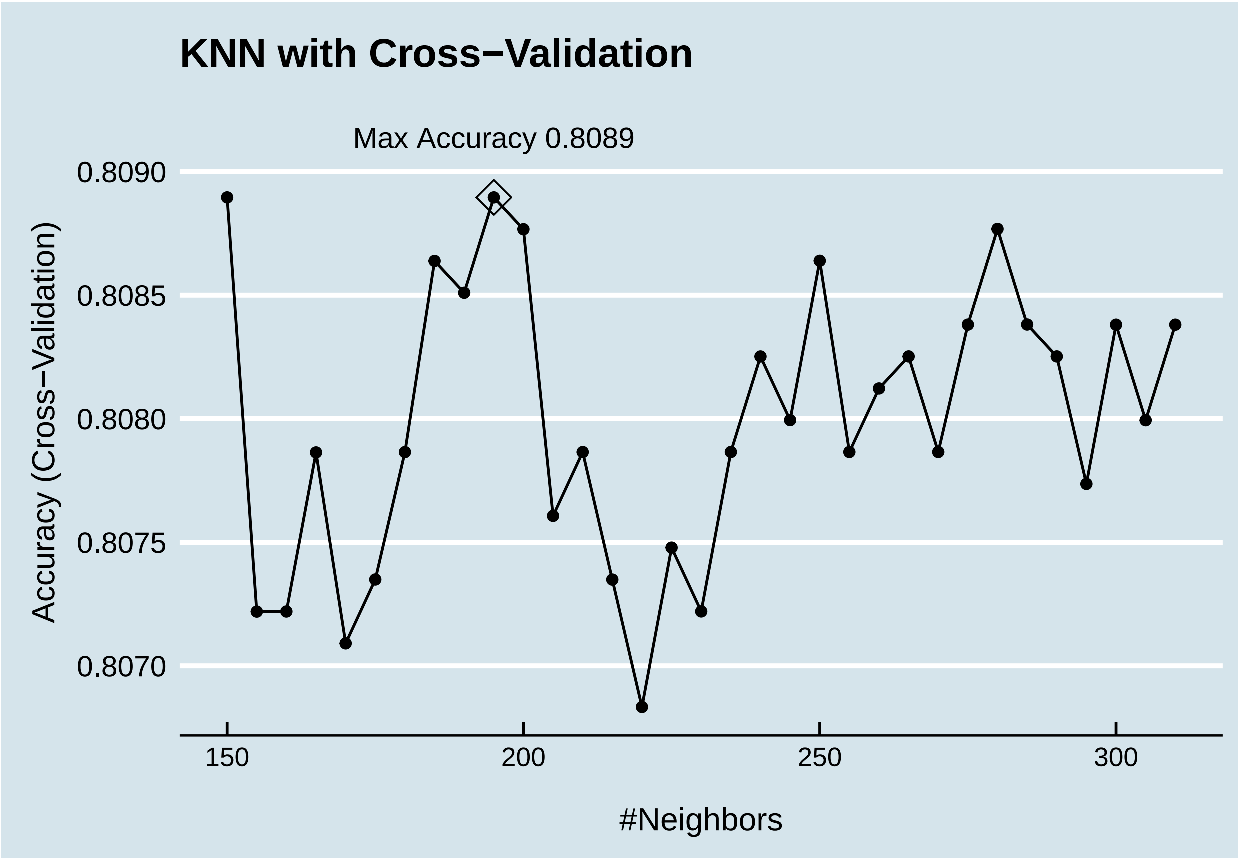
```
## [1] "Accuracy KNN_Bootstrap 0.8085"
```




7.5. k-Nearest Neighbor Classification with Cross-validation (knn_crosvalidation)

```
## [1] "Optimal NN 195"
```

```
## [1] "Accuracy KNN_crossvalidation 0.8089"
```

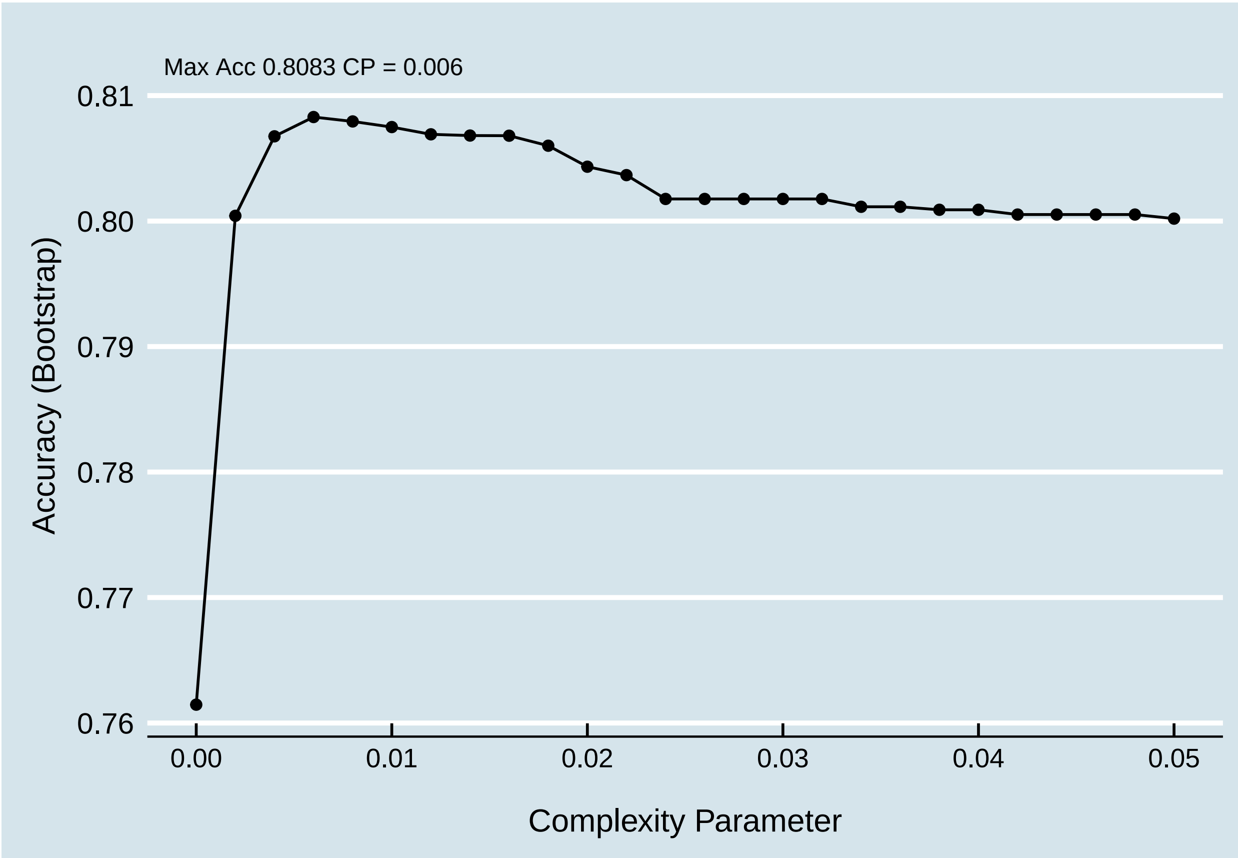


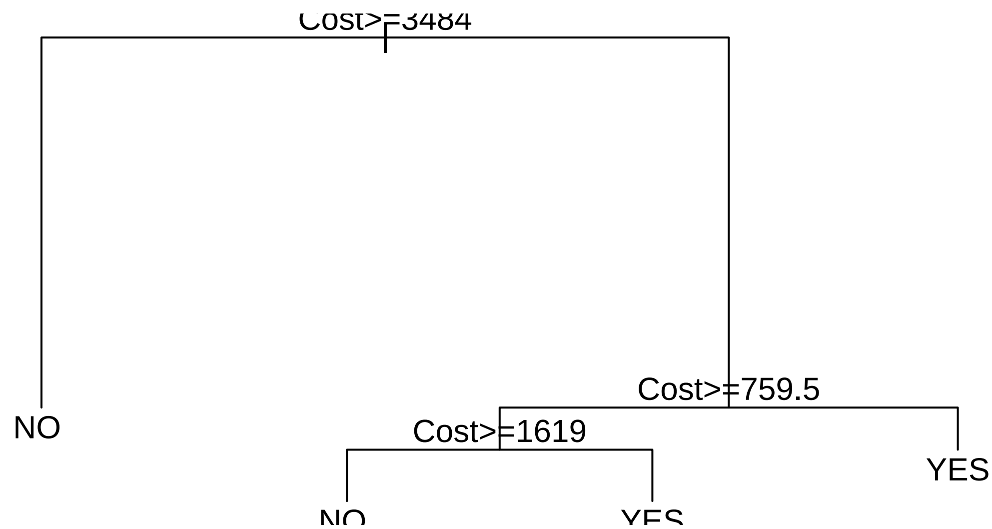

7.6. Classification Tree Model (ctm)

```
## [1] "Optimal iteration 4"
```

```
## [1] "Optimal CP 0.006"
```

```
## [1] "Accuracy CTM 0.8083"
```


Max Acc 0.8083 CP = 0.006
0.81
0.80
0.79
0.78
0.77
0.76
Accuracy (Bootstrap)
0.00
0.01
0.02
0.03
0.04
0.05
Complexity Parameter

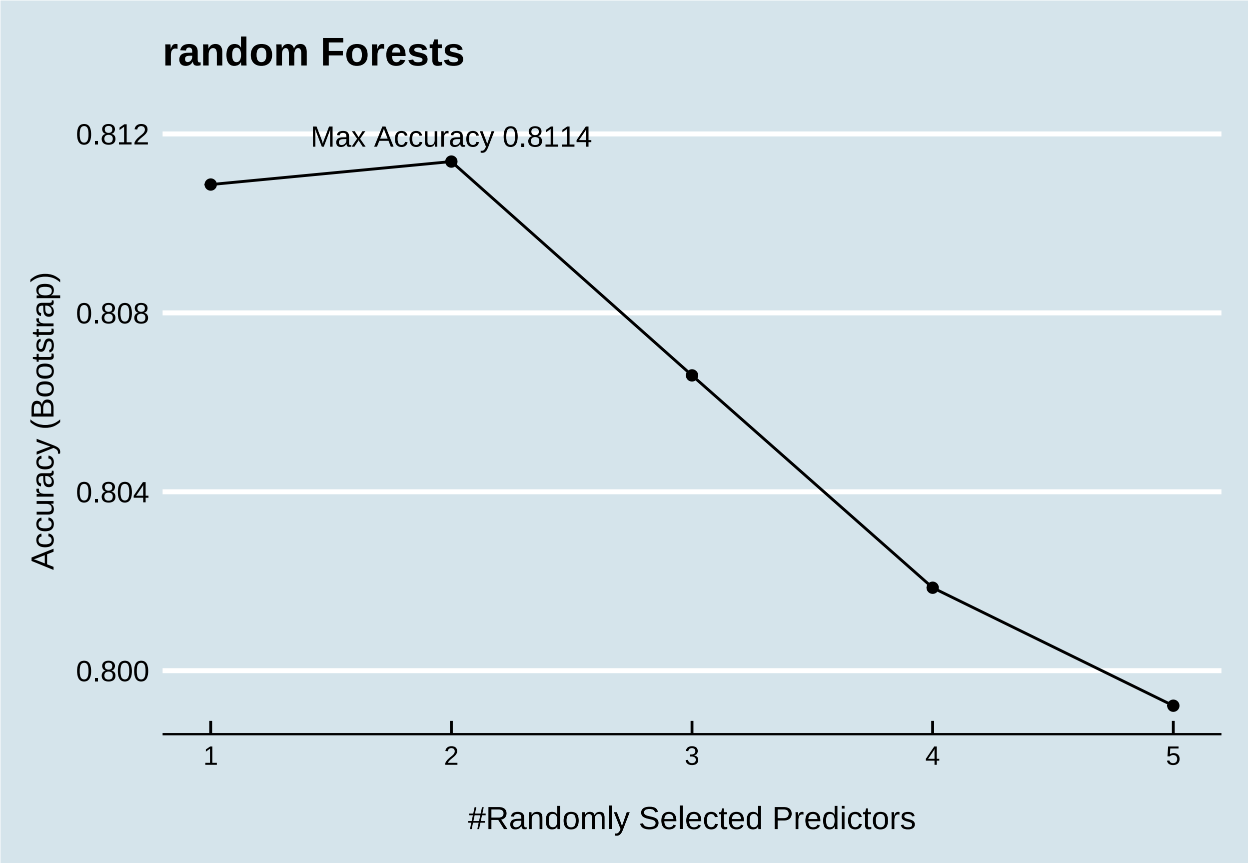



7.7. Random Forests (rf)

```
## [1] "Variable Importance"
```

```
## rf variable importance
##
##                                Overall
## Cost                           100.000
## WorkingHours                    88.530
## DO                              83.201
## AVGTicket                       46.750
## LoadFactor                      45.657
## DriverTypeRecurrent             13.841
## CorpProdYES                     11.611
## FrequentZoneLDZ                  5.981
## FleetRoleDWOC                    4.899
## DriverTenureMore than 12 months  4.170
## DriverTenure7-12 months          3.746
## FrequentMomentLDM                1.893
## DriverTenure3 - 6 months         1.398
## DriverTypeReactivated            0.000
```

```
## [1] "Accuracy RF 0.8114"
```




7.8. Naive Bayes (naive_bayes)

```
## [1] "Accuracy NB 0.8001"
```

7.9. Support Vector Machine (svmLinear)

```
## [1] "Accuracy SVM 0.7964"
```

7.10. Generalized Additive Models - Locally Estimated Scatterplot Smoothing (gamLoess)

```
## Loading required package: gam
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## Loaded gam 1.20
```

```
## [1] "Accuracy gam_loess 0.8079"
```

7. 11. Multinomial Logistic Regression (multinom)

```
## [1] "Accuracy multinom 0.8039"
```

# 8. Models' summary

Models have already been trained, so the next step is to chose the best performing ones in order to create an ensemble, and improve their individual predictive power:

```
##           model_name         models   results
## 1                lda      train_lda 0.7942476
## 2                qda      train_qda 0.6913340
## 3                glm      train_glm 0.8039009
## 4      knn_bootstrap      train_knn 0.8085343
## 5  knn_crosvalidation    train_knn_2 0.8088957
## 6                ctm    train_rpart 0.8082912
## 7                 rf       train_rf 0.8113823
## 8        naive_bayes       train_nb 0.8000557
## 9          svmLinear      train_smv 0.7964457
## 10          gamLoess train_gamLoess 0.8079177
## 11          multinom train_multinom 0.8039150
```

As accuracy of QDA is so low and configures an outlier, we will drop it to calculate the **mean accuracy**:

```
## [1] "The mean accuracy of the models is 0.8044"
```

Now there will only stay in the analysis those models giving results above the mean:

```
##               model     train_name   results
## 1     knn_bootstrap      train_knn 0.8085343
## 2 knn_crosvalidation    train_knn_2 0.8088957
## 3               ctm    train_rpart 0.8082912
## 4                rf       train_rf 0.8113823
## 5          gamLoess train_gamLoess 0.8079177
```

# 9. Accuracy on the test set

9. 1. KNN bootstrap:

```
## [1] "Acc KNN Bootstrap on Test Set 0.8144"
```

9. 2. KNN cross-validation:

```
## [1] "Acc KNN cross-validation on Test Set 0.8167"
```

9. 3. Classification Tree Model (ctm)

```
## [1] "Acc CTM on Test Set 0.8121"
```

9. 4. Random Forests:

```
## [1] "Acc CTM on Test Set 0.8237"
```

9. 5. Gam-Loess:

```
## [1] "Acc Gam-Loess on Test Set 0.8063"
```

Accuracy of models is finally gathered in the following table:

```
##               model     train_name Accuracy_Train_Set Accuracy_Test_Set
## 1     knn_bootstrap      train_knn          0.8085343         0.8143852
## 2 knn_crosvalidation    train_knn_2          0.8088957         0.8167053
## 3               ctm    train_rpart          0.8082912         0.8120650
## 4                rf       train_rf          0.8113823         0.8236659
## 5          gamLoess train_gamLoess          0.8079177         0.8062645
```

# 10. Final Predictions:

Once best models are selected over the others, the next step is to get a classification solution that takes them all into account, voting for each case.

The first ten rows of this experiment, look like this:

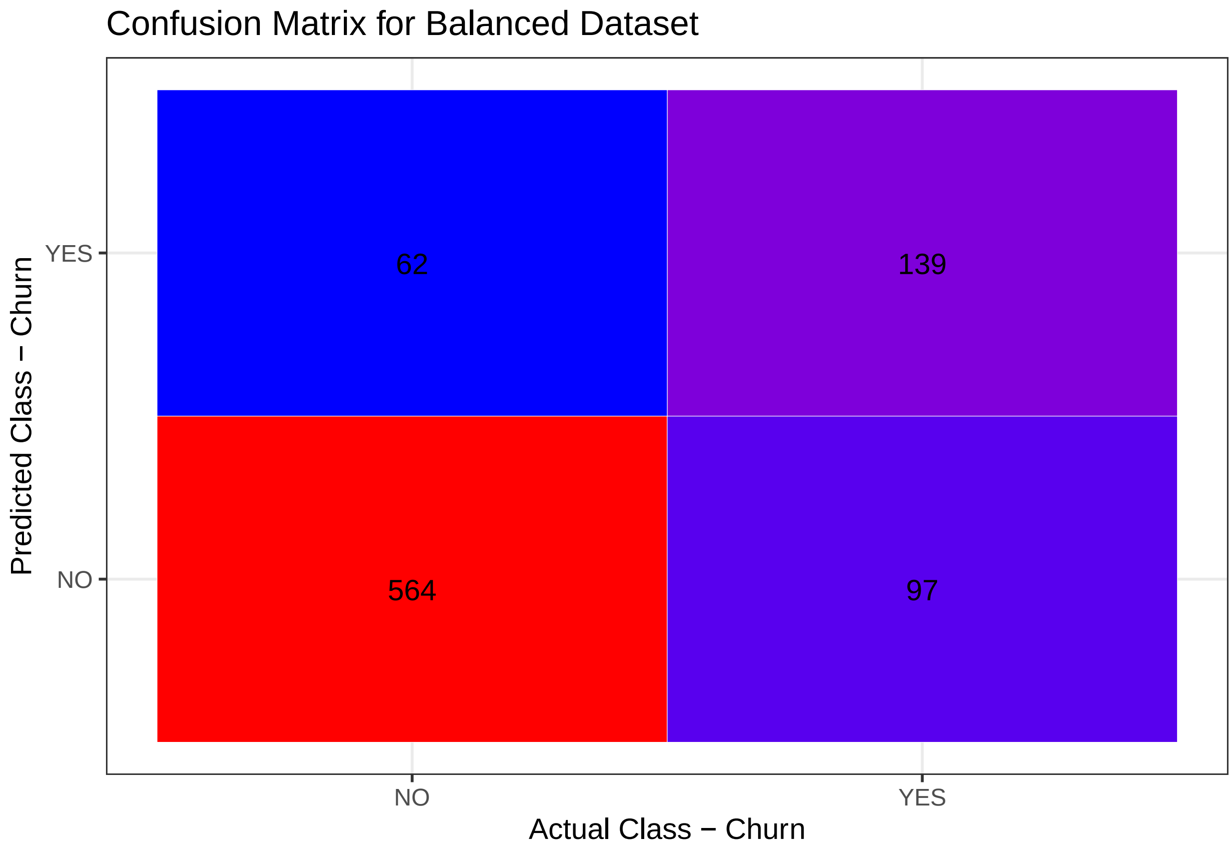
```
##    prediction_knn_bootstrap prediction_knn_cv prediction_ctm prediction_rf
## 1                         0                 0              0             0
## 2                         0                 0              0             0
## 3                         0                 0              0             0
## 4                         0                 0              0             0
## 5                         0                 0              0             0
## 6                         0                 0              0             0
## 7                         0                 0              0             0
## 8                         0                 0              0             0
## 9                         0                 0              0             0
## 10                        0                 0              0             0
##    prediction_gl Final_Decision
## 1              0             NO
## 2              0             NO
## 3              0             NO
## 4              0             NO
## 5              0             NO
## 6              0             NO
## 7              0             NO
## 8              0             NO
## 9              0             NO
## 10             0             NO
```

The final accuracy is:

```
## [1] "Accuracy of Ensemble 0.8155"
```

And the **confusion matrix** looks like this:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NO YES
##        NO  564  97
##        YES  62 139
##
##                Accuracy : 0.8155
##                  95% CI : (0.788, 0.8409)
##     No Information Rate : 0.7262
##     P-Value [Acc > NIR] : 6.127e-10
##
##                   Kappa : 0.5137
##
##  Mcnemar's Test P-Value : 0.00701
##
##             Sensitivity : 0.9010
##             Specificity : 0.5890
##          Pos Pred Value : 0.8533
##          Neg Pred Value : 0.6915
##              Prevalence : 0.7262
##          Detection Rate : 0.6543
##    Detection Prevalence : 0.7668
##       Balanced Accuracy : 0.7450
##
##        'Positive' Class : NO
##
```




Being **NO** the positive class, the final results show a great predicting power of **SENSITIVITY**, that means that no churners will be indeed detected.

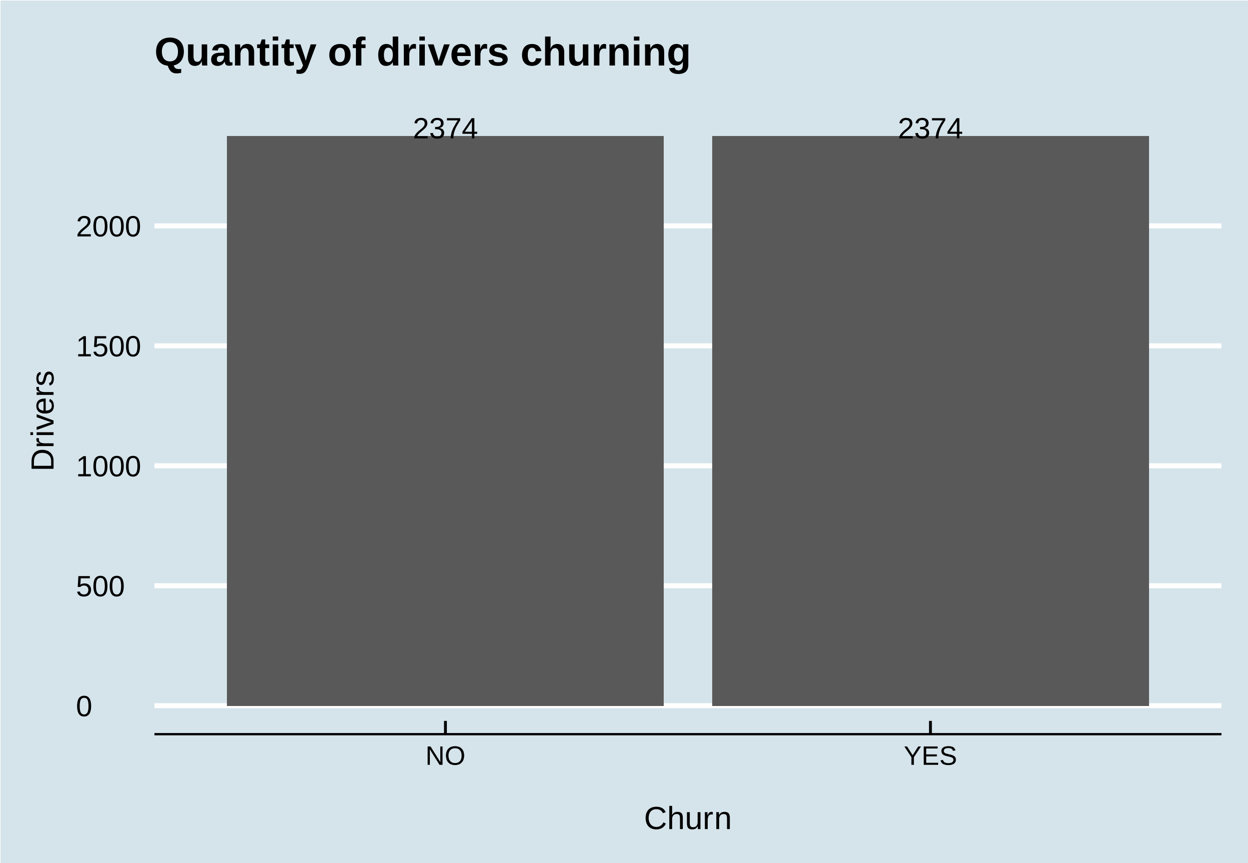
On the other hand results on **SPECIFICITY** are quite bad, meaning that a significant proportion churners are not actually being detected (False Positive - Type I Error).

As seen in the Variable Importance of Random Forests, income is the most important variable to determine Churn, so a bonus could be a great incentive to motivate those who are potential churners to keep driving with the APP.

With these considerations, we conclude that this model is good in a mature operation in which *making bonus budget efficient* is a top priority, but in an initial step of the business retaining drivers may be critical.

## 11. Comments and alternatives.

One interesting change in the study, is to balance datasets in order to train models with equal proportions of churners and not churners.




```
## # A tibble: 2 x 3
##   Churn Drivers  Prop
##   <chr>   <int> <dbl>
## 1 NO       2374   0.5
## 2 YES      2374   0.5
```

Now there is no prevalence in datasets, 50% for each churn and no churn classes.

```
## [1] "Accuracy lda 0.776"
```

```
## [1] "Accuracy qda 0.7381"
```

```
## [1] "Accuracy glm 0.7845"
```

```
## [1] "Accuracy knn_bootstrap 0.7768"
```

```
## [1] "Accuracy knn_cv 0.7781"
```

```
## [1] "Accuracy ctm 0.7813"
```

```
## [1] "Accuracy rf 0.7886"
```

```
## [1] "Accuracy nb 0.7763"
```

```
## [1] "Accuracy svm 0.7784"
```

```
## [1] "Accuracy gam_loess 0.7776"
```

```
## [1] "Accuracy multinom 0.7845"
```

The results of the models in terms of **Accuracy** can be compiled as follows:

```
##         model_name_2       models_2 results_2
## 1                lda      train_lda 0.7759527
## 2                qda      train_qda 0.7380999
## 3                glm      train_glm 0.7844839
## 4      knn_bootstrap      train_knn 0.7768416
## 5  knn_crosvalidation    train_knn_2 0.7780918
## 6                ctm    train_rpart 0.7812864
## 7                 rf       train_rf 0.7885957
## 8        naive_bayes       train_nb 0.7762744
## 9          svmLinear      train_smv 0.7783755
## 10          gamLoess train_gamLoess 0.7775667
## 11          multinom train_multinom 0.7845081
```

As done before, the best models will be selected as the ones above the mean but excluding the outlier of *qda*:

```
## [1] "The mean accuracy of the models is 0.7802"
```

The final models are the ones shown below:

```
##      model     train_name results_2
## 1      glm      train_glm 0.7844839
## 2      ctm    train_rpart 0.7812864
## 3       rf       train_rf 0.7885957
## 4 multinom train_multinom 0.7845081
```

Now it is time to measure **Accuracy** in the Test Set:

```
##      model     train_name Accuracy_Train_Set_2 Accuracy_Test_Set_2
## 1      glm      train_glm            0.7844839           0.7836134
## 2      ctm    train_rpart            0.7812864           0.7941176
## 3       rf       train_rf            0.7885957           0.7878151
## 4 multinom train_multinom            0.7845081           0.7857143
```

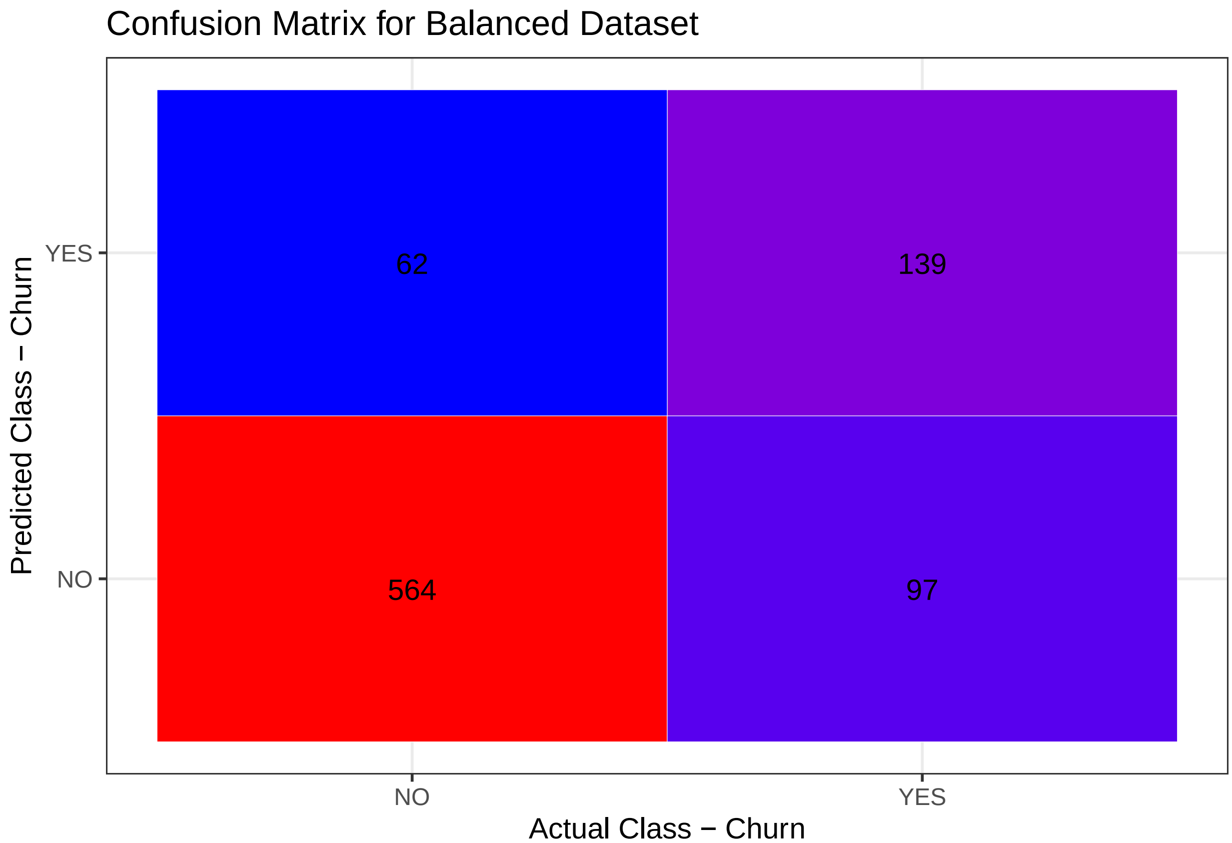
Accuracy in both training and test sets is similar, demonstrating there was no overtraining.

The ensemble is formed againt with the new best models:

```
## [1] "Accuracy of Ensemble 0.7836"
```

And the **confusion matrix** looks like this:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NO YES
##        NO  178  43
##        YES  60 195
##
##                Accuracy : 0.7836
##                  95% CI : (0.7439, 0.8198)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5672
##
##  Mcnemar's Test P-Value : 0.1149
##
##             Sensitivity : 0.7479
##             Specificity : 0.8193
##          Pos Pred Value : 0.8054
##          Neg Pred Value : 0.7647
##              Prevalence : 0.5000
##          Detection Rate : 0.3739
##    Detection Prevalence : 0.4643
##       Balanced Accuracy : 0.7836
##
##        'Positive' Class : NO
##
```




Being **NO** the positive class, the results show a lesser accuracy, losing predicting power on **SENSITIVITY**, that means being less capable of detecting actual no churners.

But as a great win, **SPECIFICITY** rises and more actual churners are detected and with a more aggressive *bonus budget allocation* retention can improve significantly, even if given bonuses to not churners.

The model to put in practice in real life will depend on the maturity of the business and other variables related to competition that are beyond this publication.