Churn Prediction for a Bank

Aim of the project

The aim of my project is to provide some consultancy to a bank, because they have noticed that a lot of customers are leaving the company. The first thing I saw in this company is that IT department is not very efficient and need a bit of restyling. Nowadays it's fundamental to manage relationships with the customers and have a market pull approach rather than a technology push, at least for a retail bank. Market pull → the need is identified by customers; Technology push → R&D drives the development of new products. This is done through Customer Relationship Management, which helps the company to ensure customer satisfaction. This tool, if apply successfully, improve the retention power (the probability that a customer will not leave). This is particularly powerful because acquiring new customers is far more costly than satisfying and retaining existing customers. The phenomenon related to the customers who leave the company it is commonly called customer churn. Typically, the customer churn rate is calculated as a relative number in percentage(churn rate) In particular, there are several reasons that lead to monitoring the churn rate. - marketing costs to acquire new customers are high;
- allows to calculate customer lifetime value; - it allows to see whether what the company is improving the customer churn.

The process for identifying the churners (those who leave the company) is called customer churn prediction. Is there any tool as good as machine learning for prediction? I don't think so. I don't consider new customers acquired during the selected period of time. This is a classification problem, and I chose to model the churn prediction problem as a standard binary classification task, labelling each customer as "churner" or "non-churner".

Import the packages

Now let's pass to the practical part. First of all I import the packages I need

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

## Loading required package: lattice

## Loading required package: ggplot2

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:rattle':
##
##     importance

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:randomForest':
##
##     combine
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Import the dataset

The dataset has 10k observations and 14 columns. The response variable is exited, 0 means the customer has stayed, 1 means the customer's left. This is not a real dataset. I've taken it from Kaggle. I don't think Kaggle needs any presentation. The dataset is already cleaned and ready to be used, that's why I didnt do a deep data cleansing nor feature engineering. By looking at some variables, such as 'balance' and 'isactivemember' I thought that even customers with a very low balance and inactive can be seen as churners. But in the end I opted for considering just the variable exited.

```
churn <- read_delim('Churn_Modelling.csv', delim = ',', col_types = cols(
  RowNumber = col_integer(),
  CustomerId = col_integer(),
  Surname = col_character(),
  CreditScore = col_integer(),
  Geography = col_character(),
  Gender = col_character(),
  Age = col_integer(),
  Tenure = col_integer(),
  Balance = col_double(),
  NumOfProducts = col_integer(),
  HasCrCard = col_integer(),
  IsActiveMember = col_integer(),
  EstimatedSalary = col_double(),
  Exited = col_integer()
))

head(churn)
```

```
## # A tibble: 6 x 14
##   RowNumber CustomerId Surname CreditScore Geography Gender   Age Tenure
##       <int>      <int> <chr>         <int> <chr>     <chr>  <int>  <int>
## 1         1   15634602 Hargra~         619 France    Female    42      2
## 2         2   15647311 Hill            608 Spain     Female    41      1
## 3         3   15619304 Onio            502 France    Female    42      8
## 4         4   15701354 Boni            699 France    Female    39      1
## 5         5   15737888 Mitche~         850 Spain     Female    43      2
## 6         6   15574012 Chu             645 Spain     Male      44      8
## # ... with 6 more variables: Balance <dbl>, NumOfProducts <int>,
## #   HasCrCard <int>, IsActiveMember <int>, EstimatedSalary <dbl>,
## #   Exited <int>
```

Exploration Phase

Now I want to see the differences in average score considering all the categorical variables.

```r
# Not huge differences in credit score among geographic zones
churn %>%
  group_by(Geography) %>%
  summarise(n = n(),
            avg_credit_score = mean(CreditScore),
            sd_credit_score = sd(CreditScore))
```

```
## # A tibble: 3 x 4
##   Geography     n avg_credit_score sd_credit_score
##   <chr>     <int>            <dbl>           <dbl>
## 1 France     5014             650.            97.0
## 2 Germany    2509             651.            98.2
## 3 Spain      2477             651.            94.4
```

```r
# Those who didnt exit tend to have a higher credit score
churn %>%
  group_by(Exited) %>%
  summarise(n = n(),
            avg_credit_score = mean(CreditScore),
            sd_credit_score = sd(CreditScore))
```

```
## # A tibble: 2 x 4
##   Exited     n avg_credit_score sd_credit_score
##    <int> <int>            <dbl>           <dbl>
## 1      0  7963             652.            95.7
## 2      1  2037             645.           100.
```

```r
# Doesn't seem to have an impact the number of products held by a customer
churn %>%
  group_by(NumOfProducts) %>%
  summarise(n = n(),
            avg_credit_score = mean(CreditScore),
            sd_credit_score = sd(CreditScore))
```

```
## # A tibble: 4 x 4
##   NumOfProducts     n avg_credit_score sd_credit_score
##           <int> <int>            <dbl>           <dbl>
## 1             1  5084             649.            97.2
## 2             2  4590             652.            96.1
## 3             3   266             648.            95.2
## 4             4    60             654.           101.
```

```r
# Credit score is roughly the same between Gender
churn %>%
  group_by(Gender) %>%
  summarise(n = n(),
            avg_credit_score = mean(CreditScore),
            sd_credit_score = sd(CreditScore))
```

```
## # A tibble: 2 x 4
##   Gender     n avg_credit_score sd_credit_score
##   <chr>  <int>            <dbl>           <dbl>
## 1 Female  4543             651.            96.8
## 2 Male    5457             650.            96.5
```

```r
# Active members tend to have a higher score.
churn %>%
  group_by(IsActiveMember) %>%
  summarise(n = n(),
            avg_credit_score = mean(CreditScore),
            sd_credit_score = sd(CreditScore))
```

```
## # A tibble: 2 x 4
##   IsActiveMember     n avg_credit_score sd_credit_score
##            <int> <int>            <dbl>           <dbl>
## 1              0  4849             648.            97.7
## 2              1  5151             653.            95.6
```

```
#Customers with a credit card have a slightly higher credit score, on average
churn %>%
  group_by(HasCrCard) %>%
  summarise(n = n(),
  avg_credit_score = mean(CreditScore),
  sd_credit_score = sd(CreditScore))
```

```
## # A tibble: 2 x 4
##   HasCrCard     n avg_credit_score sd_credit_score
##       <int> <int>            <dbl>           <dbl>
## 1         0  2945             651.            96.6
## 2         1  7055             650.            96.7
```

Missing values

I want to verify if there are missing values. There aren't.

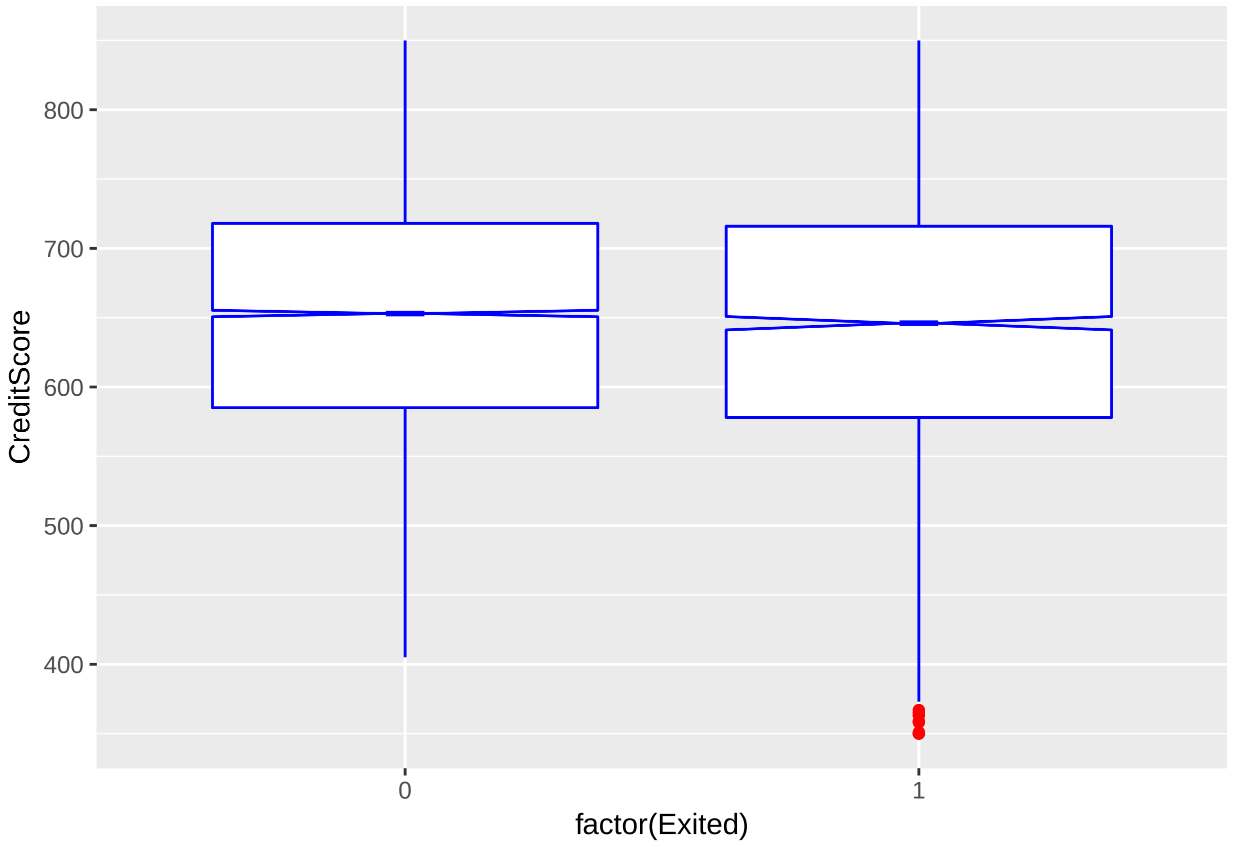
```
sum(is.na(churn))
```

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

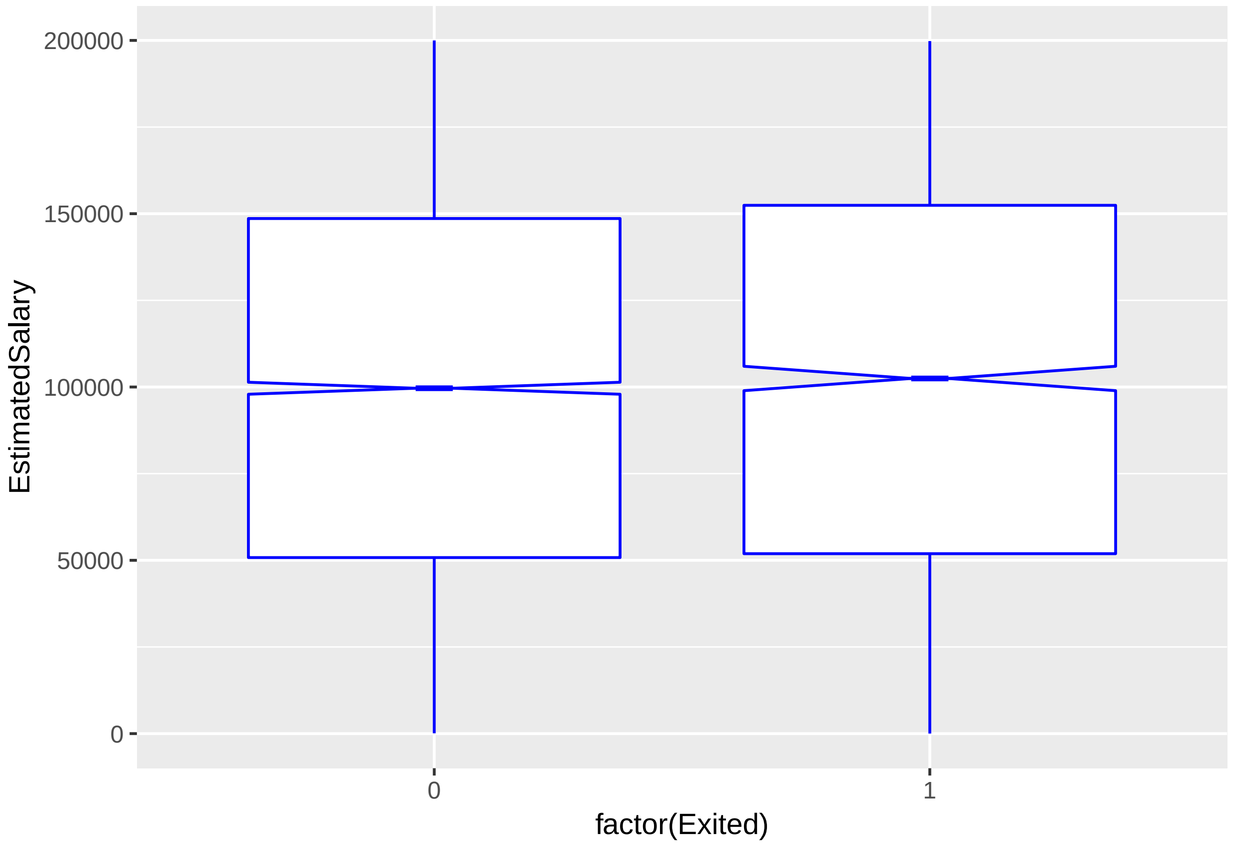
Some plot

I want to see graphically id there are important differences in credit score and in estimated salary between people who exited and people who don't. Apparently there are no important differences

```
churn %>% ggplot(aes(factor(Exited), CreditScore)) +
  geom_boxplot(color = 'blue', outlier.colour = 'red', notch = TRUE, notchwidth = 0.1)
```



```
churn %>% ggplot(aes(factor(Exited), EstimatedSalary)) +
  geom_boxplot(color = 'blue', outlier.colour = 'red', notch = TRUE, notchwidth = 0.1)
```



Statistical tests

I want to perform some t-test to see if the difference in credit score, balance and EstimatedSalary (between guys who exited and guys who did not) is significative. The difference is very likely to be different from zero just in terms of balance and creditscore but not for estimated salary.

```
t.test(CreditScore ~ Exited, churn)
```

```
##
##  Welch Two Sample t-test
##
## data:  CreditScore by Exited
## t = 2.6347, df = 3050.9, p-value = 0.008465
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   1.663067 11.340331
## sample estimates:
## mean in group 0 mean in group 1
##        651.8532        645.3515
```

```
t.test(Balance ~ Exited, churn)
```

```
##
##  Welch Two Sample t-test
##
## data:  Balance by Exited
## t = -12.471, df = 3347.8, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -21250.22 -15476.26
## sample estimates:
## mean in group 0 mean in group 1
##        72745.30        91108.54
```

```
t.test(EstimatedSalary ~ Exited, churn)
```

```
##
##  Welch Two Sample t-test
##
## data:  EstimatedSalary by Exited
## t = -1.2034, df = 3137.4, p-value = 0.2289
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -4541.656  1087.085
## sample estimates:
## mean in group 0 mean in group 1
##        99738.39       101465.68
```

Data Preprocessing

Now i want to turn into numerical dummies the categorical features, such as Geography and Gender

```
churn$Geography <- factor(churn$Geography,
                          levels = c('France', 'Spain', 'Germany'),
                          labels = c(1,2,3))

churn$Gender <- factor(churn$Gender,
                          levels = c('Female', 'Male'),
                          labels = c(0,1))
```

Split the dataset

Now I split the dataset in train and test, with a 75:25 proportion. Then, I label the response variable.

```
churn <- churn[,4:14]
churn_train <- churn %>% sample_frac(0.75)
churn_test <- churn %>% anti_join(churn_train)
```

```
## Joining, by = c("CreditScore", "Geography", "Gender", "Age", "Tenure", "Balance", "NumOfProducts", "
```

```
churn$Exited <- factor(churn$Exited, levels = c(0,1), labels = c('FALSE', 'TRUE'))
churn_train$Exited <- factor(churn_train$Exited, levels = c(0,1), labels = c('FALSE', 'TRUE'))
churn_test$Exited <- factor(churn_test$Exited, levels = c(0,1), labels = c('FALSE', 'TRUE'))
```

Machine Learning

Now it's time to build some machine learning model to try to help with some insights the company. First of all, as it is a binary problem, we have to consider the type of errors related to the business problem.

FALSE NEGATIVE COST: COST OF INCORRECTLY IDENTIFYING A CUSTOMER AS HIGH RISK OF SWITCHING (wasted marketing cost) → churners as non churners FALSE POSITIVE COST: COST OF FAILING TO IDENTIFY A HIGH RISK CUSTOMER (LOST REVENUE). → non churners as churners (more costly)

Said that, we can start by creating our models.

Logistic regression

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary) as it is in this case. First I fit the model with train data using glm function, then I predict the test data. I build a confusion matrix setting the threshold at 0.5, meaning that all the values below 0.5 are considered to be FALSE and viceversa all the values above 0.5 are considered to be TRUE.

In this case FALSE can be misleading because represents the customers who don't leave the company.

This model has an accuracy of 0.80. Not bad, but I guess we can do better.

```r
set.seed(12345)
mod1 <- glm(Exited ~ ., data = churn_train, family = binomial())
summary(mod1)
```

```
##
## Call:
## glm(formula = Exited ~ ., family = binomial(), data = churn_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2557  -0.6610  -0.4641  -0.2766   2.9056
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -3.434e+00  2.829e-01 -12.137  < 2e-16 ***
## CreditScore     -4.638e-04  3.227e-04  -1.437   0.1507
## Geography2       2.711e-02  8.086e-02   0.335   0.7374
## Geography3       7.230e-01  7.803e-02   9.265  < 2e-16 ***
## Gender1         -5.230e-01  6.275e-02  -8.335  < 2e-16 ***
## Age              6.899e-02  2.934e-03  23.516  < 2e-16 ***
## Tenure          -1.631e-02  1.080e-02  -1.510   0.1309
## Balance          2.982e-06  5.932e-07   5.027 4.98e-07 ***
## NumOfProducts   -1.022e-01  5.433e-02  -1.881   0.0599 .
## HasCrCard        2.532e-03  6.868e-02   0.037   0.9706
## IsActiveMember  -1.027e+00  6.605e-02 -15.552  < 2e-16 ***
## EstimatedSalary  3.993e-07  5.461e-07   0.731   0.4647
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7558.4  on 7499  degrees of freedom
## Residual deviance: 6461.2  on 7488  degrees of freedom
## AIC: 6485.2
##
## Number of Fisher Scoring iterations: 5
```

```r
pr1 <- predict(mod1, type = 'response', newdata = churn_test)

response_hat_insample <- pr1 > 0.5 ## Predict 1 if pr1>0.5
cm  <- confusionMatrix(factor(response_hat_insample), factor(churn_test$Exited))
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  1933  427
##      TRUE     49   91
##
##                Accuracy : 0.8096
##                  95% CI : (0.7937, 0.8248)
##     No Information Rate : 0.7928
##     P-Value [Acc > NIR] : 0.01951
##
##                   Kappa : 0.2066
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.9753
##             Specificity : 0.1757
##          Pos Pred Value : 0.8191
##          Neg Pred Value : 0.6500
##              Prevalence : 0.7928
##          Detection Rate : 0.7732
##    Detection Prevalence : 0.9440
##       Balanced Accuracy : 0.5755
##
##        'Positive' Class : FALSE
##
```

Decision tree

The next model I'm going to build is a decision tree. In this case this model could be very useful also to double check the feature importance model I've built before. We got a better accuracy than logit. The main drawback of a decision tree is the overfitting problem. So the next natural step is to build a random forest.

The fancyRpartplot is very insightful in my opinion. It allows to visually and explicitly represent decisions and decision making. In this case we can spot which are the 'clusters' more prone to leave. For example, we can see that people ranging from 51 years old and 45 are the ones who have a higher probability to leave.

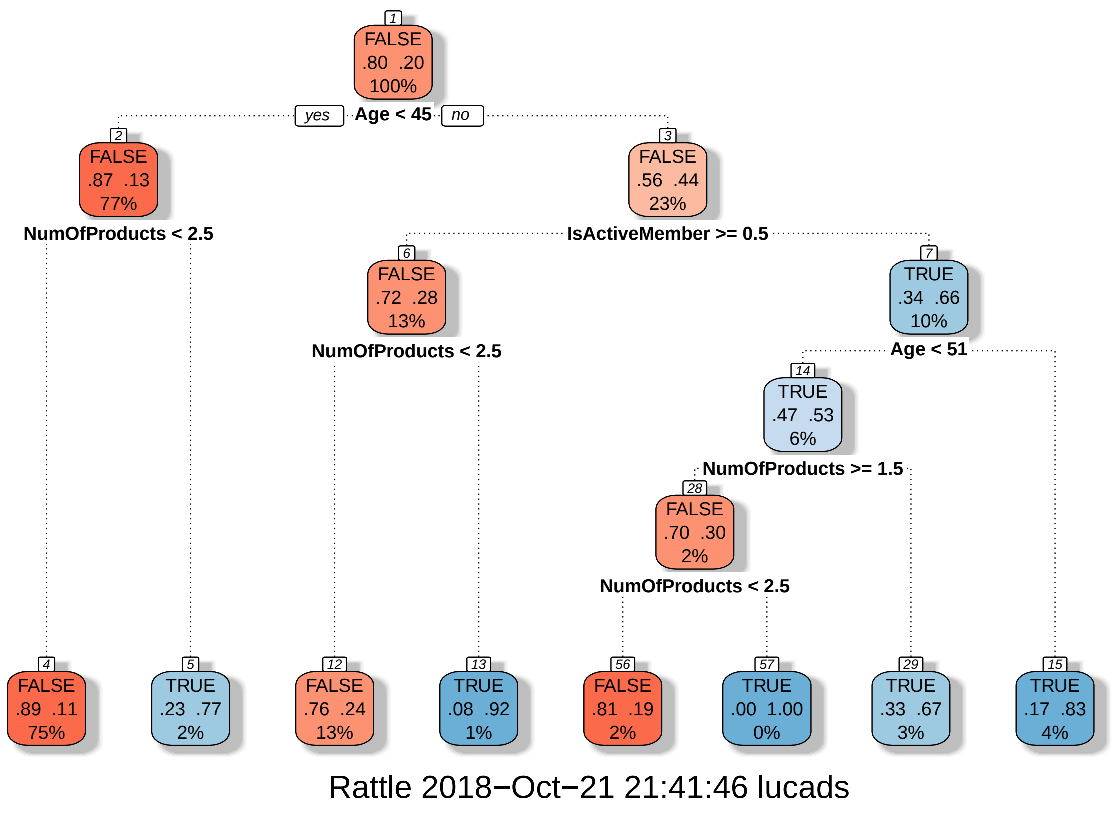
```r
set.seed(12345)
mod2 <- rpart(factor(Exited) ~., data = churn_train)

pr2 <- predict(mod2, newdata = churn_test[-11], type = 'class')
summary(pr2)
```

```
## FALSE  TRUE
##  2228   272
```

```r
cm2  <- confusionMatrix(pr2, factor(churn_test$Exited))

fancyRpartPlot(mod2, palettes = c('Reds'))
```



Random forest

The random forest model is an ensemble technique focused just on decision trees. After the ensemble of trees (the forest) is generated, the model uses a vote to combine the trees' predictions. It allows to overcome the overfitting problem mentioned before. I use 200 trees for building the model.

```
set.seed(12345)
mod3 <- randomForest(Exited ~., data = churn_train, ntree = 200)

pr3 <- predict(mod3, newdata = churn_test[-11])
summary(pr3)
```

```
## FALSE  TRUE
##  2213   287
```

```
cm3  <- confusionMatrix(pr3, factor(churn_test$Exited))
cm3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  1920  293
##      TRUE     62  225
##
##                Accuracy : 0.858
##                  95% CI : (0.8437, 0.8715)
##     No Information Rate : 0.7928
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4826
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9687
##             Specificity : 0.4344
##          Pos Pred Value : 0.8676
##          Neg Pred Value : 0.7840
##              Prevalence : 0.7928
##          Detection Rate : 0.7680
##    Detection Prevalence : 0.8852
##       Balanced Accuracy : 0.7015
##
##        'Positive' Class : FALSE
##
```

Boosting

Finally I want to use the boosting. This technique boosts the performance of weak learners to attain the performance of stronger learners. Boosting uses ensembles of models trained of resampled data and a vote to determine the final prediction. I will be using C5.0 algorithm, a very powerful one. It allows us to assign a penalty to different types of errors, in order to discourage a tree from making more costly mistakes. The penalties are designated in a cost matrix, which specifies how much costlier each error is, relative to any other prediction.

```r
set.seed(123)
churn_train$Geography <- as.integer(churn_train$Geography)
churn_train$Gender <- as.integer(churn_train$Gender)
churn_test$Geography <- as.integer(churn_test$Geography)
churn_test$Gender <- as.integer(churn_test$Gender)
matrix_dimensions <- list(c('FALSE', 'TRUE'), c('FALSE', 'TRUE'))
names(matrix_dimensions) <- c('predicted', 'actual')
error_cost <- matrix(c(0,4,1,0), ncol = 2, dimnames = matrix_dimensions)
churn_cost <- C5.0(churn_train[-11], churn_train$Exited, costs = error_cost)
churn_cost_pred <- predict(churn_cost,churn_test[-11])
cm4 <- confusionMatrix(factor(churn_cost_pred), churn_test$Exited)
cm4
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  1970  418
##      TRUE     12  100
##
##                Accuracy : 0.828
##                  95% CI : (0.8126, 0.8426)
##     No Information Rate : 0.7928
##     P-Value [Acc > NIR] : 5.154e-06
##
##                   Kappa : 0.2632
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
##             Sensitivity : 0.9939
##             Specificity : 0.1931
##          Pos Pred Value : 0.8250
##          Neg Pred Value : 0.8929
##              Prevalence : 0.7928
##          Detection Rate : 0.7880
##    Detection Prevalence : 0.9552
##       Balanced Accuracy : 0.5935
##
##        'Positive' Class : FALSE
##
```

CONCLUSIONS

So the objective of my project was to predict in advance which customers are likely to leave a bank with quite good precision, avoiding costs related to false positive errors (non churners classified as churners).

The ability to execute upon an insight, the so called actionability, is nearly always the most important metric to judge model success. Model exists to help us create more desiderable outcomes, that is to maximise profitability. For a problem like that, I think that the C50 algorithm is one of the best in terms of actionability even if a random forest model is better in term of accuracy.

I think that further development in the field must be done by considering the temporal characteristic, which increases the overall analysis complexity. Data mining and machine Learning tools can definitely help bank to understand their customers' behavior, confirming that further studies may be worth considering.

Age and number of products are the most relevant features. We can say that younger clients which have between 2 and 4 products are more prone to exit the bank. This can mean that these people are attracted by new products. Thanks to Machine Learning we are able to predict it and we are able to tackle the problem by proposing and release new type of products. I think is very powerful but at the same time simple the last algorithm I have used because in the real world, and in this case as well, the error is not symmetric.

Possible solutions:

- Use the retained earning to do some strategical acquisitions. Acquire those fintech startups who have a specific know-how and offer those kind of products that attract the segment we noticed left the company with a higher probability, allows the bank to tackle this problem and to conceal this high churn rate
- Another solution is to improve the customer relationship management system by asking for some consultancy (e.g. salesforce) and try to win the competition represented mainly by fintech startups.