Customer Churn Analysis

Rafal Mokrzycki

Churn analysis is the evaluation of a company's customer loss rate in order to reduce it. It is one of the most important and challenging problems for businesses such as credit card and telecommunication companies. The full cost of churn includes both lost revenue and the marketing costs involved with replacing those customers with new ones. Statistics show that acquiring new customers can cost five times more than retaining existing customers.

This customers data set is from a credit card company, where it is possible to review customer attributes such as gender, age, tenure, balance, number of products they are subscribed to, their estimated salary and if they left the company or not. In this analysis tree methods will be used to predict, which customer groups have the highest risk of churn.

1. Dataset import and preparation

The first step is to import required libraries, as well as the data set itself. The following libraries will be used:

```
library(tibble)
library(tree)
library(rpart)
library(rpart.plot)
library(caret)
library(tidyverse)
library(randomForest)
library(ipred)
library(gbm)
library(plyr)
```

The dataset derives from Kaggle and can can be found here.

```
df <- read.csv("churn.csv")
head(df)
```

```
##   RowNumber CustomerId  Surname CreditScore Geography Gender Age Tenure
## 1         1   15634602 Hargrave         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 Mitchell         850     Spain Female  43      2
## 6         6   15574012      Chu         645     Spain   Male  44      8
##     Balance NumOfProducts HasCrCard IsActiveMember EstimatedSalary Exited
## 1      0.00             1         1              1       101348.88      1
## 2  83807.86             1         0              1       112542.58      0
## 3 159660.80             3         1              0       113931.57      1
```

```
## 4      0.00             2          0              0         93826.63       0
## 5 125510.82             1          1              1         79084.10       0
## 6 113755.78             2          1              0        149756.71       1
```

As one can see, in a raw dataset there are 11 variables. The depentent variable is *Exited*, which has two values: 1 when the customer exited and 0 otherwise. Independent variables can be described as follows:

- *RowNumber* - The number of the row (unique)
- *CustomerId* - The customer id (unique)
- *Surname* - Customer's surname (unique)
- *CreditScore* - Customer's credit score

- *Geography* - Which Country the customer belongs to (France, Spain or Germany)
- *Gender* - Customer's Gender

- *Age* - Customer's Age

- *Tenure* - The time of bond with company (in years)
- *Balance* - The amount left with the customer
- *NumOfProducts* - The products the customer owns

- *HasCrCard* - Whether the customer has a credit card (1) or not (0)

- *IsActiveMember* - Whether the customer is an active member (1) or not (0)

- *EstimatedSalary* - Customer's estimated salary

First, non-informative columns will be deleted from the dataset (i.e. *RowNumber*, *Surname* and *CustomerId*).

```
df <- df[-c(1:3)]
```

In order to see, how the variables are encoded and what their basic statistics are, functions *glimpse()* and *summary()* are used.

```
glimpse(df)
```

```
## Rows: 10,000
## Columns: 11
## $ CreditScore     <int> 619, 608, 502, 699, 850, 645, 822, 376, 501, 684, 5...
## $ Geography       <chr> "France", "Spain", "France", "France", "Spain", "Sp...
## $ Gender          <chr> "Female", "Female", "Female", "Female", "Female", "...
## $ Age             <int> 42, 41, 42, 39, 43, 44, 50, 29, 44, 27, 31, 24, 34,...
## $ Tenure          <int> 2, 1, 8, 1, 2, 8, 7, 4, 4, 2, 6, 3, 10, 5, 7, 3, 1,...
## $ Balance         <dbl> 0.00, 83807.86, 159660.80, 0.00, 125510.82, 113755....
## $ NumOfProducts   <int> 1, 1, 3, 2, 1, 2, 2, 4, 2, 1, 2, 2, 2, 2, 2, 2, 1, ...
## $ HasCrCard       <int> 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, ...
## $ IsActiveMember  <int> 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, ...
## $ EstimatedSalary <dbl> 101348.88, 112542.58, 113931.57, 93826.63, 79084.10...
## $ Exited          <int> 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, ...
```

```r
summary(df)
```

```
##   CreditScore     Geography            Gender               Age       
##  Min.   :350.0   Length:10000       Length:10000       Min.   :18.00  
##  1st Qu.:584.0   Class :character   Class :character   1st Qu.:32.00  
##  Median :652.0   Mode  :character   Mode  :character   Median :37.00  
##  Mean   :650.5                                         Mean   :38.92  
##  3rd Qu.:718.0                                         3rd Qu.:44.00  
##  Max.   :850.0                                         Max.   :92.00  
##      Tenure          Balance       NumOfProducts    HasCrCard     
##  Min.   : 0.000   Min.   :     0   Min.   :1.00   Min.   :0.0000  
##  1st Qu.: 3.000   1st Qu.:     0   1st Qu.:1.00   1st Qu.:0.0000  
##  Median : 5.000   Median : 97199   Median :1.00   Median :1.0000  
##  Mean   : 5.013   Mean   : 76486   Mean   :1.53   Mean   :0.7055  
##  3rd Qu.: 7.000   3rd Qu.:127644   3rd Qu.:2.00   3rd Qu.:1.0000  
##  Max.   :10.000   Max.   :250898   Max.   :4.00   Max.   :1.0000  
##  IsActiveMember   EstimatedSalary         Exited      
##  Min.   :0.0000   Min.   :    11.58   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.: 51002.11   1st Qu.:0.0000  
##  Median :1.0000   Median :100193.91   Median :0.0000  
##  Mean   :0.5151   Mean   :100090.24   Mean   :0.2037  
##  3rd Qu.:1.0000   3rd Qu.:149388.25   3rd Qu.:0.0000  
##  Max.   :1.0000   Max.   :199992.48   Max.   :1.0000  
```

There is no doubt that the categorical variables *Geography*, *Gender HasCrCard*, *IsActiveMember* and *Exited* should be encoded as factors. In terms of variables *Tenure* and *NumOfProducts* the same procedure will be applied, because these variables can take only a few, small numbers (<0,10> in case of *Tenure* and <1,4> in case of *NumOfProducts*), which can be encoded as categories.

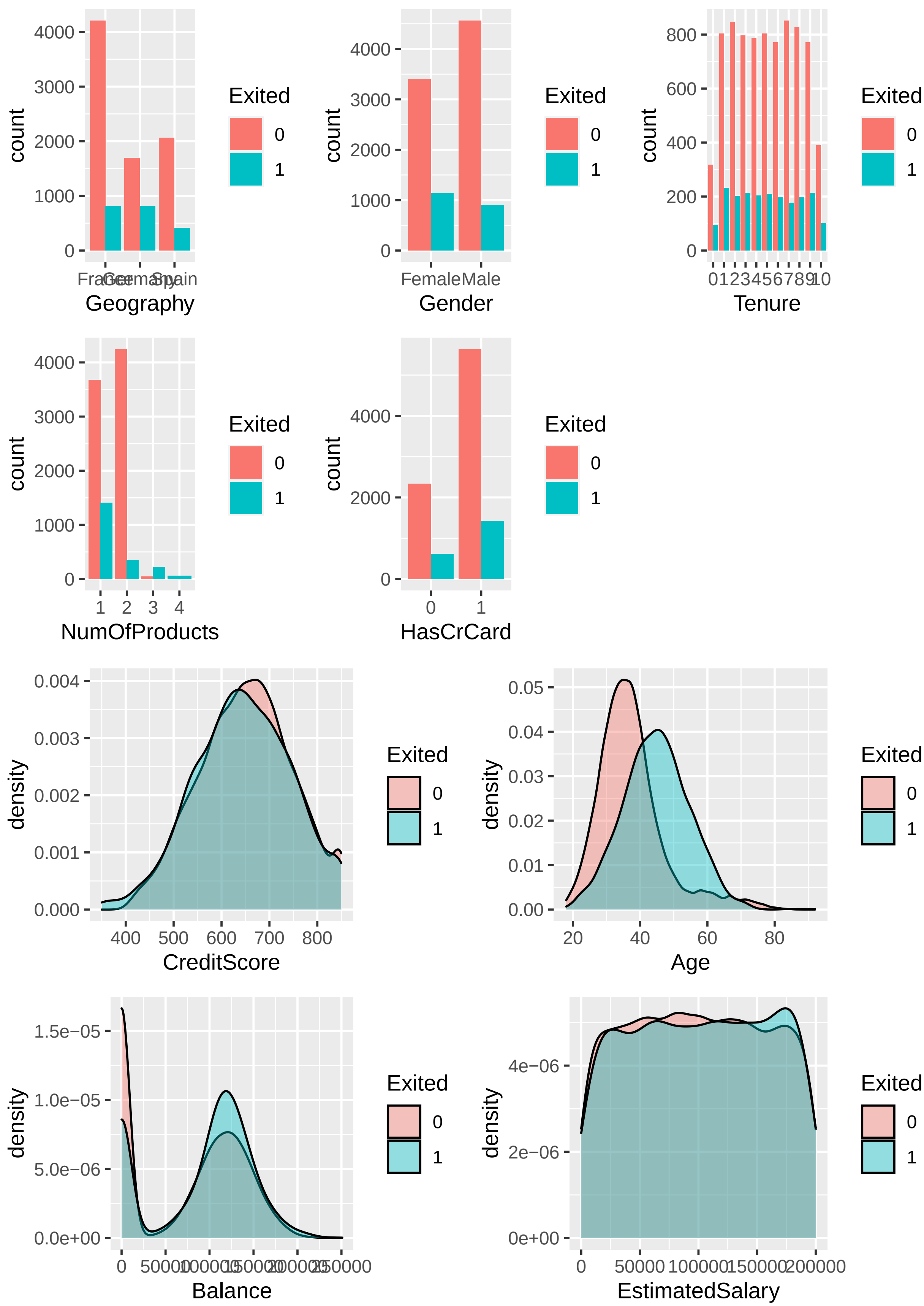
```r
cols <- c("Geography","Tenure","NumOfProducts","Gender","HasCrCard",
          "IsActiveMember","Exited")
df[cols] <- lapply(df[cols], as.factor)
```

The dataset does not contain any missings.

```r
sum(is.na(df))
```

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

2. Data visualization



Based on the plot it can be presumed that: * there are slightly less churns in Spain than in Germany and France * the churn rate is lower among men than women * the churn rate is higher among older clients

3. Data analysis

Before the very analysis starts, it is necessary to split the dataset into train, validation and test set. One splits the dataset in order to be able to compare the models of one type to one another (validation set) and the best models from each type to one another (test set). In this analysis the dataset is split into three subsets in the following proportion: 70%, 15%, 15%.

```
set.seed(1)
idx <- sample(c(1:3), size = nrow(df),
              replace = TRUE, prob = c(.7, .15, .15))
train <- (1:nrow(df))[idx == 1]
valid <- (1:nrow(df))[idx == 2]
test <- (1:nrow(df))[idx == 3]
```

In this analysis the dataset will be analized using four tree methods: plain decicion tree, random forest, bagging and gradient boosting. Between 2 and 4 models of each type will be constructed and their prediction will be compared on the validation set within the type. The best model of each type will be chosen and their prediction compared on the test set.

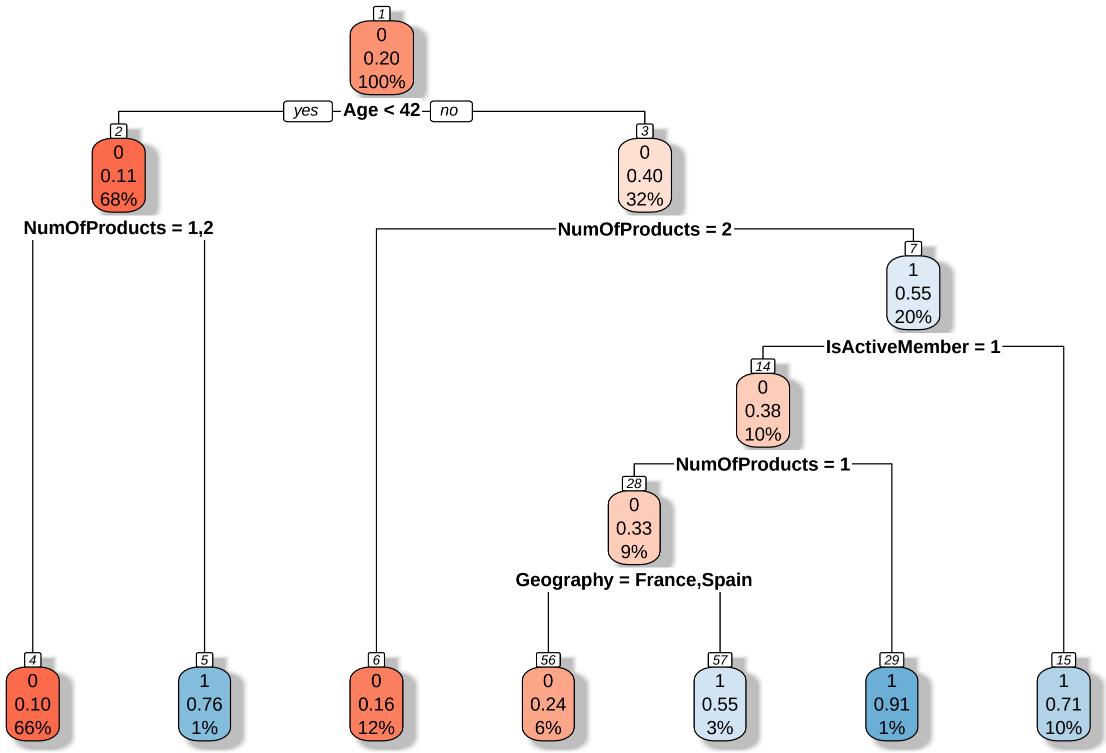
Due to the fact, that the aim of the company is to keep clients, who are going to leave it, the analysis focuses on the clients who have value 1 on the column *Exited*. Therefore it is more important to classify correctly all clients who are probably going to leave than to classify correctly those who are not or both groups. As a result, the comparison criterion of prediction will the sensitivity, computed as the ratio of true positive and sum of true positive and false negative. The higher the sensitivity, the better the prediction of the model is.

3.1. Decision trees

3.1.1. Tree 1

First, a simple decision tree is constructed using **rpart()** function from the rpart library

```
tree.model.1 <- rpart(Exited~.,data = df,subset = train,
                      method = "class",xval = 10)
rpart.plot(tree.model.1, box.palette="RdBu", shadow.col="gray", nn=TRUE)
```




```r
s.tree.model.1 <- summary(tree.model.1)
```

```
## Call:
## rpart(formula = Exited ~ ., data = df, subset = train, method = "class",
##     xval = 10)
##   n= 6964
##
##           CP nsplit rel error    xerror       xstd
## 1 0.06976744      0 1.0000000 1.0000000 0.02368810
## 2 0.03594080      3 0.7906977 0.7956307 0.02167483
## 3 0.03382664      4 0.7547569 0.7681466 0.02136822
## 4 0.01268499      5 0.7209302 0.7230444 0.02084404
## 5 0.01000000      6 0.7082452 0.7251586 0.02086922
##
## Variable importance
##   NumOfProducts             Age  IsActiveMember         Balance       Geography
##              40              38              11               5               4
## EstimatedSalary          Tenure       HasCrCard
##               1               1               1
##
## Node number 1: 6964 observations,    complexity param=0.06976744
##   predicted class=0  expected loss=0.2037622  P(node) =1
##     class counts:  5545  1419
##    probabilities: 0.796 0.204
##   left son=2 (4714 obs) right son=3 (2250 obs)
```

```
##   Primary splits:
##       Age            < 41.5     to the left,  improve=252.53710, (0 missing)
##       NumOfProducts  splits as  LLRR,         improve=212.33320, (0 missing)
##       Geography      splits as  LRL,          improve= 67.96232, (0 missing)
##       IsActiveMember splits as  RL,           improve= 56.21127, (0 missing)
##       Balance        < 87523.26 to the left,  improve= 34.31679, (0 missing)
##   Surrogate splits:
##       NumOfProducts splits as  LLRR,         agree=0.682, adj=0.016, (0 split)
##       CreditScore   < 390.5    to the right, agree=0.677, adj=0.002, (0 split)
##
## Node number 2: 4714 observations,    complexity param=0.0359408
##   predicted class=0  expected loss=0.110734  P(node) =0.6769098
##     class counts:  4192   522
##    probabilities: 0.889 0.111
##   left son=4 (4617 obs) right son=5 (97 obs)
##   Primary splits:
##       NumOfProducts splits as  LLRR,         improve=84.242240, (0 missing)
##       Geography     splits as  LRL,          improve=15.064160, (0 missing)
##       Age           < 34.5     to the left,  improve=13.566350, (0 missing)
##       Balance       < 97638.86 to the left,  improve= 9.492652, (0 missing)
##       CreditScore   < 407.5    to the right, improve= 9.361528, (0 missing)
##
## Node number 3: 2250 observations,    complexity param=0.06976744
##   predicted class=0  expected loss=0.3986667  P(node) =0.3230902
##     class counts:  1353   897
##    probabilities: 0.601 0.399
##   left son=6 (865 obs) right son=7 (1385 obs)
##   Primary splits:
##       NumOfProducts  splits as  RLRR,         improve=157.61750, (0 missing)
##       IsActiveMember splits as  RL,           improve=104.81670, (0 missing)
##       Geography      splits as  LRL,          improve= 50.02524, (0 missing)
##       Age            < 65.5     to the right, improve= 28.96179, (0 missing)
##       Balance        < 87573.12 to the left,  improve= 27.46949, (0 missing)
##   Surrogate splits:
##       Balance         < 6229.595 to the left,  agree=0.700, adj=0.218, (0 split)
##       Age             < 69.5     to the right, agree=0.624, adj=0.021, (0 split)
##       EstimatedSalary < 199442.8 to the right, agree=0.618, adj=0.006, (0 split)
##
## Node number 4: 4617 observations
##   predicted class=0  expected loss=0.09703271  P(node) =0.662981
##     class counts:  4169   448
##    probabilities: 0.903 0.097
##
## Node number 5: 97 observations
##   predicted class=1  expected loss=0.2371134  P(node) =0.01392878
##     class counts:    23    74
##    probabilities: 0.237 0.763
##
## Node number 6: 865 observations
##   predicted class=0  expected loss=0.1618497  P(node) =0.1242102
##     class counts:   725   140
##    probabilities: 0.838 0.162
##
## Node number 7: 1385 observations,    complexity param=0.06976744
```

```
##   predicted class=1  expected loss=0.4534296  P(node) =0.19888
##     class counts:   628   757
##    probabilities: 0.453 0.547
##   left son=14 (692 obs) right son=15 (693 obs)
##   Primary splits:
##       IsActiveMember splits as  RL,           improve=78.02834, (0 missing)
##       NumOfProducts  splits as  L-RR,         improve=48.57020, (0 missing)
##       Geography      splits as  LRL,          improve=40.21642, (0 missing)
##       Age            < 65.5     to the right, improve=22.22556, (0 missing)
##       Gender         splits as  RL,           improve=14.77189, (0 missing)
##   Surrogate splits:
##       Age             < 54.5     to the right, agree=0.576, adj=0.152, (0 split)
##       Tenure          splits as  LRLRRLRLRRL,  agree=0.534, adj=0.066, (0 split)
##       EstimatedSalary < 107652.9 to the left,  agree=0.532, adj=0.064, (0 split)
##       Geography       splits as  RRL,          agree=0.529, adj=0.058, (0 split)
##       HasCrCard       splits as  LR,           agree=0.529, adj=0.056, (0 split)
##
## Node number 14: 692 observations,    complexity param=0.03382664
##   predicted class=0  expected loss=0.3786127  P(node) =0.09936818
##     class counts:   430   262
##    probabilities: 0.621 0.379
##   left son=28 (634 obs) right son=29 (58 obs)
##   Primary splits:
##       NumOfProducts splits as  L-RR,         improve=36.263960, (0 missing)
##       Geography     splits as  LRL,          improve=24.618560, (0 missing)
##       Age           < 65.5     to the right, improve=11.500840, (0 missing)
##       Balance       < 184583.9 to the left,  improve= 6.250211, (0 missing)
##       Gender        splits as  RL,           improve= 6.024906, (0 missing)
##
## Node number 15: 693 observations
##   predicted class=1  expected loss=0.2857143  P(node) =0.09951177
##     class counts:   198   495
##    probabilities: 0.286 0.714
##
## Node number 28: 634 observations,    complexity param=0.01268499
##   predicted class=0  expected loss=0.329653  P(node) =0.09103963
##     class counts:   425   209
##    probabilities: 0.670 0.330
##   left son=56 (452 obs) right son=57 (182 obs)
##   Primary splits:
##       Geography splits as  LRL,          improve=24.665950, (0 missing)
##       Age       < 65.5     to the right, improve= 8.878712, (0 missing)
##       Balance   < 184583.9 to the left,  improve= 6.361347, (0 missing)
##       Gender    splits as  RL,           improve= 3.288373, (0 missing)
##       Tenure    splits as  RRLRRRLLLLL,  improve= 2.759463, (0 missing)
##   Surrogate splits:
##       Age             < 79.5     to the left,  agree=0.716, adj=0.011, (0 split)
##       EstimatedSalary < 660.835  to the right, agree=0.716, adj=0.011, (0 split)
##
## Node number 29: 58 observations
##   predicted class=1  expected loss=0.0862069  P(node) =0.008328547
##     class counts:     5    53
##    probabilities: 0.086 0.914
##
```

```
## Node number 56: 452 observations
##   predicted class=0  expected loss=0.2411504  P(node) =0.06490523
##     class counts:   343   109
##    probabilities: 0.759 0.241
##
## Node number 57: 182 observations
##   predicted class=1  expected loss=0.4505495  P(node) =0.02613441
##     class counts:    82   100
##    probabilities: 0.451 0.549
```

```
tree.pred.1 = predict(tree.model.1, newdata = df[valid,], type = "class")
cm.tree.model.1 <- confusionMatrix(data = tree.pred.1, df[valid,]$Exited,
                                   positive = "1")
cm.tree.model.1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1174  161
##          1   78  159
##
##                Accuracy : 0.848
##                  95% CI : (0.8292, 0.8654)
##     No Information Rate : 0.7964
##     P-Value [Acc > NIR] : 9.174e-08
##
##                   Kappa : 0.481
##
##  Mcnemar's Test P-Value : 1.132e-07
##
##             Sensitivity : 0.4969
##             Specificity : 0.9377
##          Pos Pred Value : 0.6709
##          Neg Pred Value : 0.8794
##              Prevalence : 0.2036
##          Detection Rate : 0.1011
##    Detection Prevalence : 0.1508
##       Balanced Accuracy : 0.7173
##
##        'Positive' Class : 1
##
```

Sensitivity of the simple tree model is equal to 0.496875.

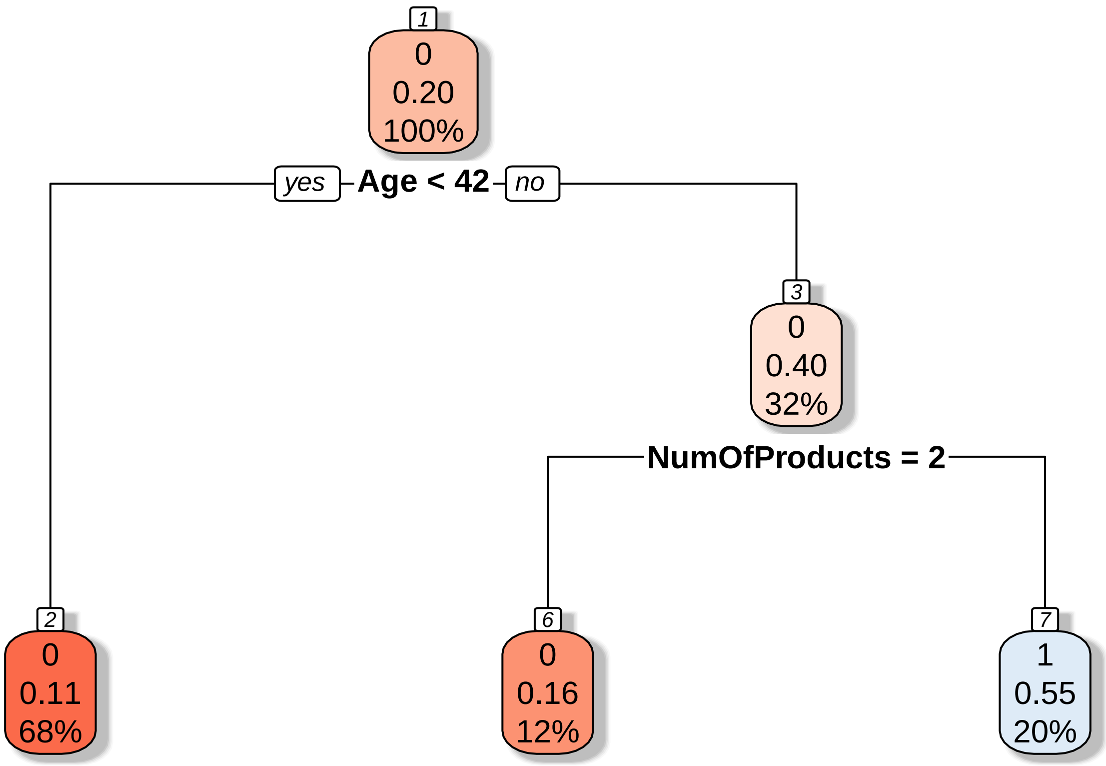
3.1.1. Tree 2

In order to gain better trade-off betweeen stability of the tree and higher purity of terminal nodes, the number of observations in terminal nodes will be set to 500.

```
tree.model.2 <- rpart(Exited~.,data = df,subset = train,
                      method = "class",xval = 10,
```

```r
                    minbucket = 500)

rpart.plot(tree.model.2, box.palette="RdBu", shadow.col="gray", nn=TRUE)
```




```r
s.tree.model.2 <- summary(tree.model.2)
```

```
## Call:
## rpart(formula = Exited ~ ., data = df, subset = train, method = "class",
##     xval = 10, minbucket = 500)
##   n= 6964
##
##           CP nsplit rel error    xerror       xstd
## 1 0.04545455      0 1.0000000 1.0000000 0.02368810
## 2 0.01000000      2 0.9090909 0.9069767 0.02282637
##
## Variable importance
##           Age NumOfProducts       Balance
##            56            36             8
##
## Node number 1: 6964 observations,    complexity param=0.04545455
##   predicted class=0  expected loss=0.2037622  P(node) =1
##     class counts:  5545  1419
##    probabilities: 0.796 0.204
##   left son=2 (4714 obs) right son=3 (2250 obs)
##   Primary splits:
```

```
##       Age            < 41.5     to the left,  improve=252.53710, (0 missing)
##       NumOfProducts  splits as  RLRR,         improve=205.64920, (0 missing)
##       Geography      splits as  LRL,          improve= 67.96232, (0 missing)
##       IsActiveMember splits as  RL,           improve= 56.21127, (0 missing)
##       Balance        < 87523.26 to the left,  improve= 34.31679, (0 missing)
##   Surrogate splits:
##       NumOfProducts splits as  LLRR,         agree=0.682, adj=0.016, (0 split)
##       CreditScore   < 390.5    to the right, agree=0.677, adj=0.002, (0 split)
##
## Node number 2: 4714 observations
##   predicted class=0  expected loss=0.110734  P(node) =0.6769098
##     class counts:  4192   522
##    probabilities: 0.889 0.111
##
## Node number 3: 2250 observations,    complexity param=0.04545455
##   predicted class=0  expected loss=0.3986667  P(node) =0.3230902
##     class counts:  1353   897
##    probabilities: 0.601 0.399
##   left son=6 (865 obs) right son=7 (1385 obs)
##   Primary splits:
##       NumOfProducts  splits as  RLRR,         improve=157.61750, (0 missing)
##       IsActiveMember splits as  RL,           improve=104.81670, (0 missing)
##       Geography      splits as  LRL,          improve= 50.02524, (0 missing)
##       Balance        < 87573.12 to the left,  improve= 27.46949, (0 missing)
##       Age            < 44.5     to the left,  improve= 23.84179, (0 missing)
##   Surrogate splits:
##       Balance         < 6229.595 to the left,  agree=0.700, adj=0.218, (0 split)
##       Age             < 69.5     to the right, agree=0.624, adj=0.021, (0 split)
##       EstimatedSalary < 199442.8 to the right, agree=0.618, adj=0.006, (0 split)
##
## Node number 6: 865 observations
##   predicted class=0  expected loss=0.1618497  P(node) =0.1242102
##     class counts:   725   140
##    probabilities: 0.838 0.162
##
## Node number 7: 1385 observations
##   predicted class=1  expected loss=0.4534296  P(node) =0.19888
##     class counts:   628   757
##    probabilities: 0.453 0.547
```

```
tree.pred.2 = predict(tree.model.2, newdata = df[valid,], type = "class")
cm.tree.model.2 <- confusionMatrix(data = tree.pred.2, df[valid,]$Exited,
                                   positive = "1")
cm.tree.model.2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1119  151
##          1  133  169
##
##                Accuracy : 0.8193
##                  95% CI : (0.7994, 0.8381)
```

```
##     No Information Rate : 0.7964
##     P-Value [Acc > NIR] : 0.01224
##
##                   Kappa : 0.4309
##
##  Mcnemar's Test P-Value : 0.31309
##
##             Sensitivity : 0.5281
##             Specificity : 0.8938
##          Pos Pred Value : 0.5596
##          Neg Pred Value : 0.8811
##              Prevalence : 0.2036
##          Detection Rate : 0.1075
##    Detection Prevalence : 0.1921
##       Balanced Accuracy : 0.7109
##
##        'Positive' Class : 1
##
```

Sensitivity of the second tree model is equal to 0.496875. In comparison, the second model is better than the first one.

```
which.max(c(cm.tree.model.1$byClass["Sensitivity"],
cm.tree.model.2$byClass["Sensitivity"]))
```

```
## Sensitivity
##           2
```

3.2. Random forests

3.2.1. Random forest 1

A random forest of 500 trees will be grown. **mtry** parameter will be set to 3, because there are 10 independent variables and sqrt(10) is around 3.

```
forest.model.1 <- randomForest(Exited~.,data = df,subset = train,
                                mtry = 3, ntree = 500, importance = TRUE)
forest.pred.1 = predict(forest.model.1, newdata = df[valid,], type = "class")
cm.forest.model.1 <- confusionMatrix(data = forest.pred.1,                                       df[valid,]$
```

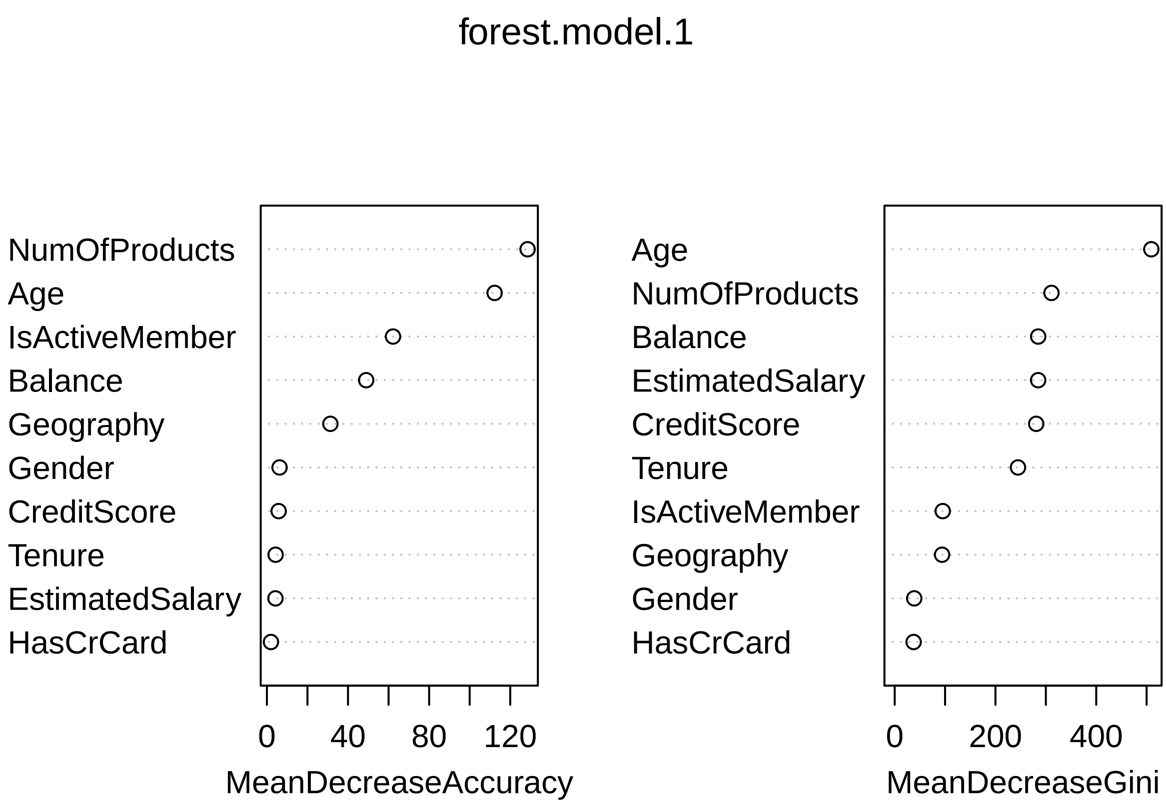
Sensitivity of a simple random forest model is equal to 0.509375. In order to see which variables are the most important a table and a plot of variable importance will be produced:

```
importance(forest.model.1)
```

```
##                        0          1 MeanDecreaseAccuracy MeanDecreaseGini
## CreditScore    4.6088230   3.719148             5.806380        280.89006
## Geography      3.2209972  46.416941            31.311542         94.00467
## Gender         4.6714479   4.267440             6.260397         38.78122
## Age           75.1263075 107.379227           112.295117        509.38282
## Tenure         2.4246986   4.480965             4.304318        244.67360
```

```
## Balance           31.2982980  34.244287          48.957393        284.78065
## NumOfProducts    103.6391989 101.468648         128.557852        311.17046
## HasCrCard          0.6212303   2.978714           2.012905         37.54321
## IsActiveMember    51.8525973  39.180613          62.197837         95.31917
## EstimatedSalary    3.2729810   2.940051           4.225342        284.72173
```

```
varImpPlot(forest.model.1)
```



As expected before, *Age* and *NumOfProducts* have the biggest influence on the dependent variable. Therefore a stratification on the variable *NumOfProducts* will be applied.

3.2.2. Random forest 2

```
forest.model.2 <- randomForest(Exited~.,
                               data = df,subset = train,
                               mtry = 3, ntree = 500, importance = TRUE, strata = NumOfProducts)

forest.pred.2 = predict(forest.model.2, newdata = df[valid,], type = "class")
cm.forest.model.2 <- confusionMatrix(data = forest.pred.2, df[valid,]$Exited,positive = "1")
```

Sensitivity of the second random forest model with stratification on variable *NumOfProducts* is equal to 0.48125, so there is an improvement. A model with stratification on variables *NumOfProducts* and *IsActiveMember* will be applied.

3.2.3. Random forest 3

```
forest.model.3 <- randomForest(Exited~.,data = df,subset = train, mtry = 3, ntree = 500, importance = T

forest.pred.3 = predict(forest.model.3, newdata = df[valid,], type = "class")
cm.forest.model.3 <- confusionMatrix(data = forest.pred.3, df[valid,]$Exited,
                                     positive = "1")
```

Sensitivity of the third random forest model with stratification on variables *NumOfProducts* and *IsActiveMember* is equal to 0.5, so the improvement is no more so crucial. Among all random forest models, the last model is the best.

```
which.max(c(cm.forest.model.1$byClass["Sensitivity"],
cm.forest.model.2$byClass["Sensitivity"],
cm.forest.model.3$byClass["Sensitivity"]))
```

```
## Sensitivity
##           1
```

3.3. Bagging

3.3.1. Bagging 1

```
bagging.model.1 <- bagging(Exited~.,data = df,subset = train, nbagg = 25, method = "double")
bagging.pred.1 = predict(bagging.model.1, newdata = df[valid,], type = "class")
cm.bagging.model.1 <- confusionMatrix(data = bagging.pred.1, df[valid,]$Exited, positive = "1")
```

Sensitivity of a simple bagging model is equal to 0.49375.

3.3.2. Bagging 2

```
bagging.model.2 <- bagging(Exited~.,data = df,subset = train,
                           nbagg = 25, method = "standard",
                           coob = TRUE)

bagging.pred.2 = predict(bagging.model.2, newdata = df[valid,], type = "class")
cm.bagging.model.2 <- confusionMatrix(data = bagging.pred.2, df[valid,]$Exited, positive = "1")
```

Sensitivity of the second bagging model is equal to 0.490625.

```
which.max(c(cm.bagging.model.1$byClass["Sensitivity"],
cm.bagging.model.2$byClass["Sensitivity"]))
```

```
## Sensitivity
##           1
```

3.4. Gradient boosting

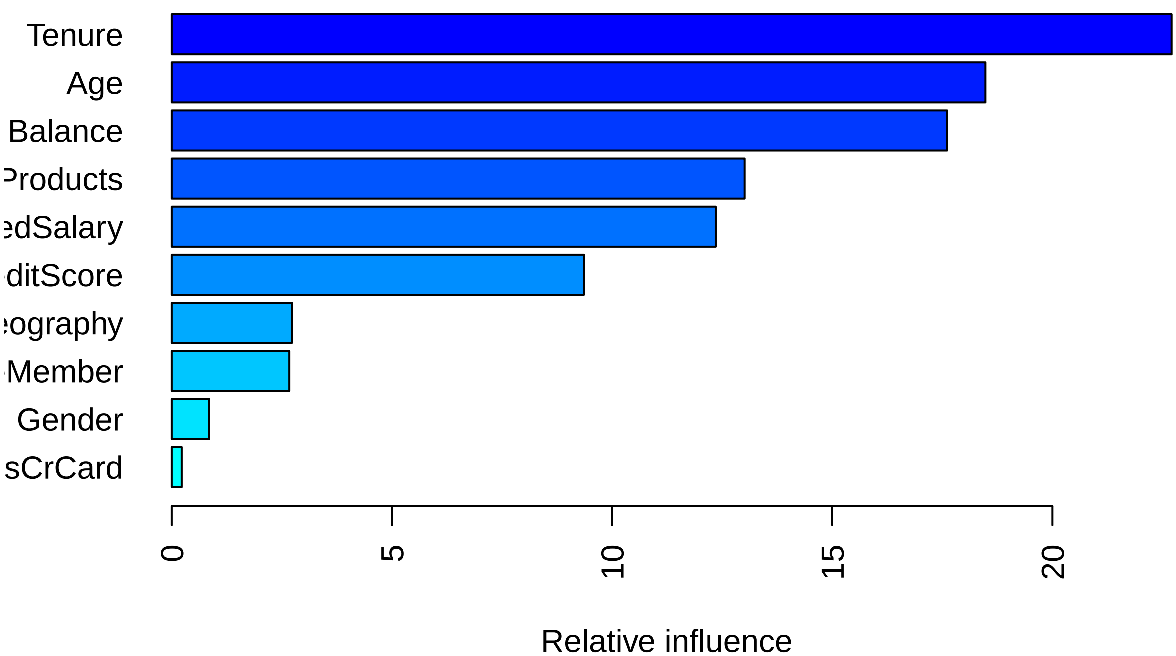
3.4.1. Boosting 1

A simple model of gradient boosting with 5000 trees will be constructed.

```
boosting.model.1 <- gbm(as.character(Exited)~.,data = df[train,], distribution = "bernoulli", n.trees =
boosting.pred.1 <- predict(boosting.model.1, newdata = df[valid,], n.trees = 5000, type = "response")
boosting.pred.1 <- ifelse(boosting.pred.1 >= .5, 1, 0)
cm.boosting.model.1 <- confusionMatrix(data = factor(boosting.pred.1), factor(df[valid,]$Exited), posit
```

Sensitivity of a simple gradient boosting model is equal to 0.471875. In order to see which variables are the most important a table and a plot of variable importance will be produced:

```
summary(boosting.model.1, las = 2)
```

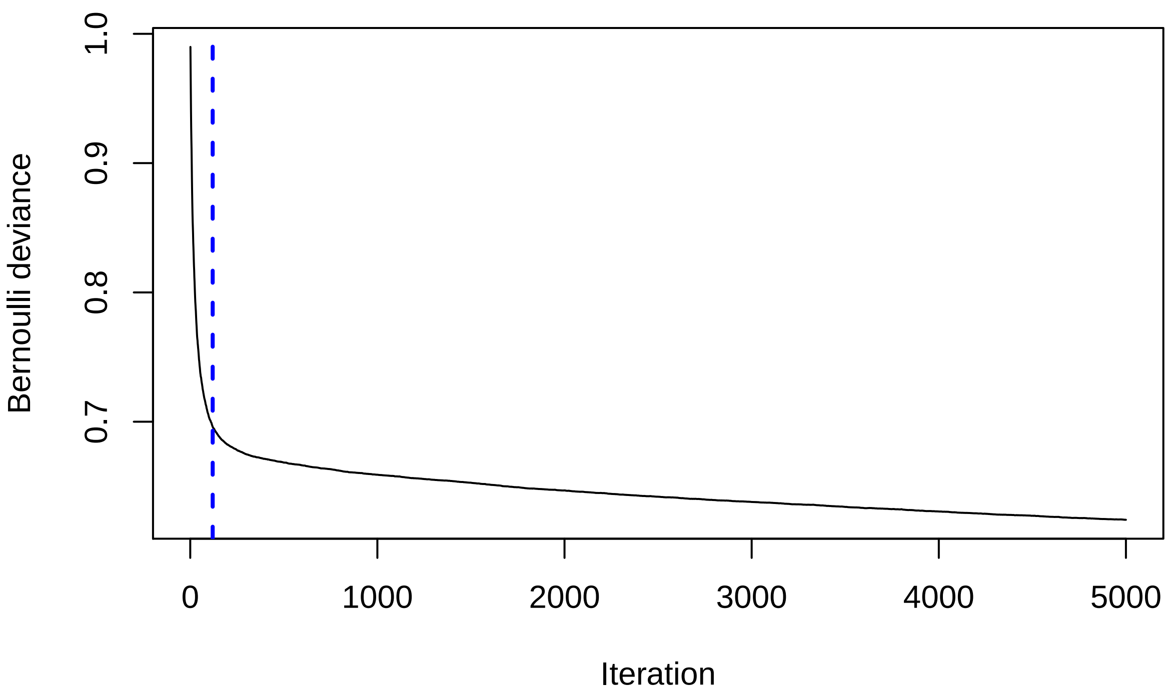



```
##                               var    rel.inf
## Tenure                     Tenure 22.7071681
## Age                           Age 18.4793146
## Balance                   Balance 17.6109166
## NumOfProducts       NumOfProducts 13.0100261
## EstimatedSalary   EstimatedSalary 12.3548277
## CreditScore           CreditScore  9.3603559
## Geography               Geography  2.7307243
## IsActiveMember     IsActiveMember  2.6713076
```

```
## Gender                        Gender  0.8482047
## HasCrCard                  HasCrCard  0.2271545
```

The variables with the highest relative influence are: Tenure, Age, Balance, NumOfProducts. In order to avoid overfitting, a reduction of number of ensemble tree will be apllied based on the out-of-bag estimate:

```
ntree.opt.oob.1 <- gbm.perf(boosting.model.1, method = "OOB", plot.it = T)
```



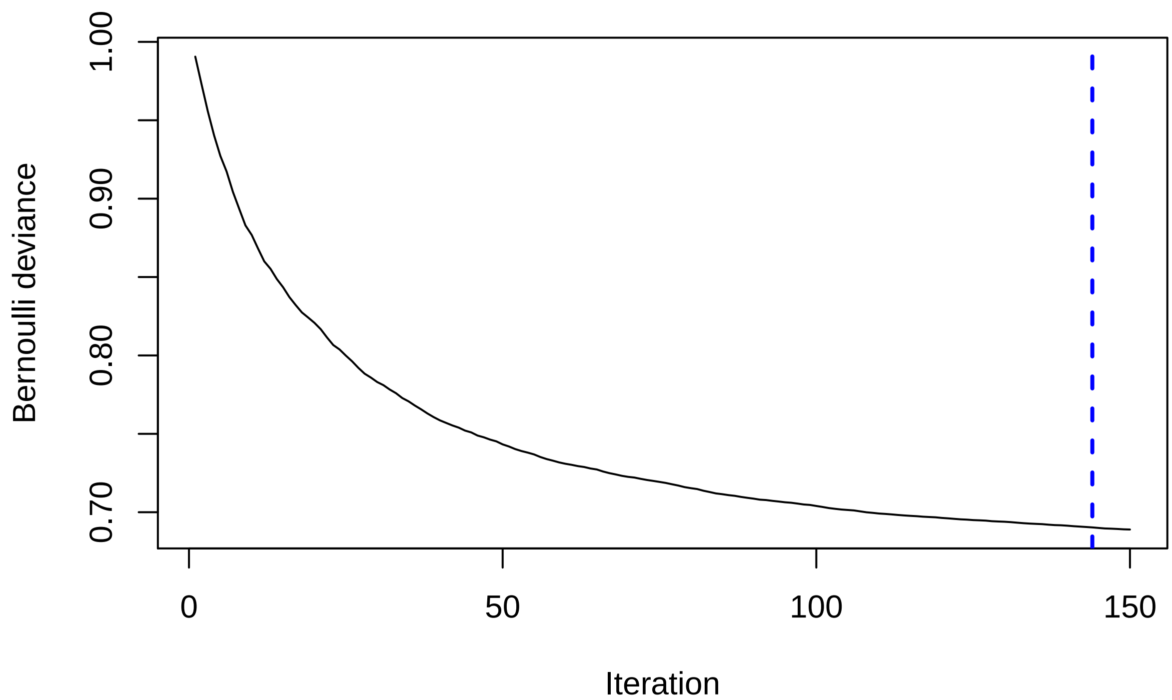
Estimated best number of trees is 120 and will be applied in the next model. The number of trees computed by the **gbm.perf()** function is always underestimated, so it is better to apply a greater number of trees in the model.

3.4.2. Boosting 2

```
boosting.model.2 <- gbm(as.character(Exited)~.,data = df[train,], distribution = "bernoulli", n.trees =
boosting.pred.2 <- predict(boosting.model.2, newdata = df[valid,], n.trees = 150, type = "response")
boosting.pred.2 <- ifelse(boosting.pred.2 >= .5, 1, 0)
cm.boosting.model.2 <- confusionMatrix(data = factor(boosting.pred.2),             factor(df[valid,]$Ex
```

Sensitivity of the second gradient boosting model is equal to 0.45, so here a decrease is observed. In order not to loose predictive power, a model with interactions and minimal number of observations in terminal nodes equal to 100 will be implemented. First the optimal number of trees will be double-checked.

```
ntree.opt.oob.2 <- gbm.perf(boosting.model.2, method = "OOB", plot.it = T)
```



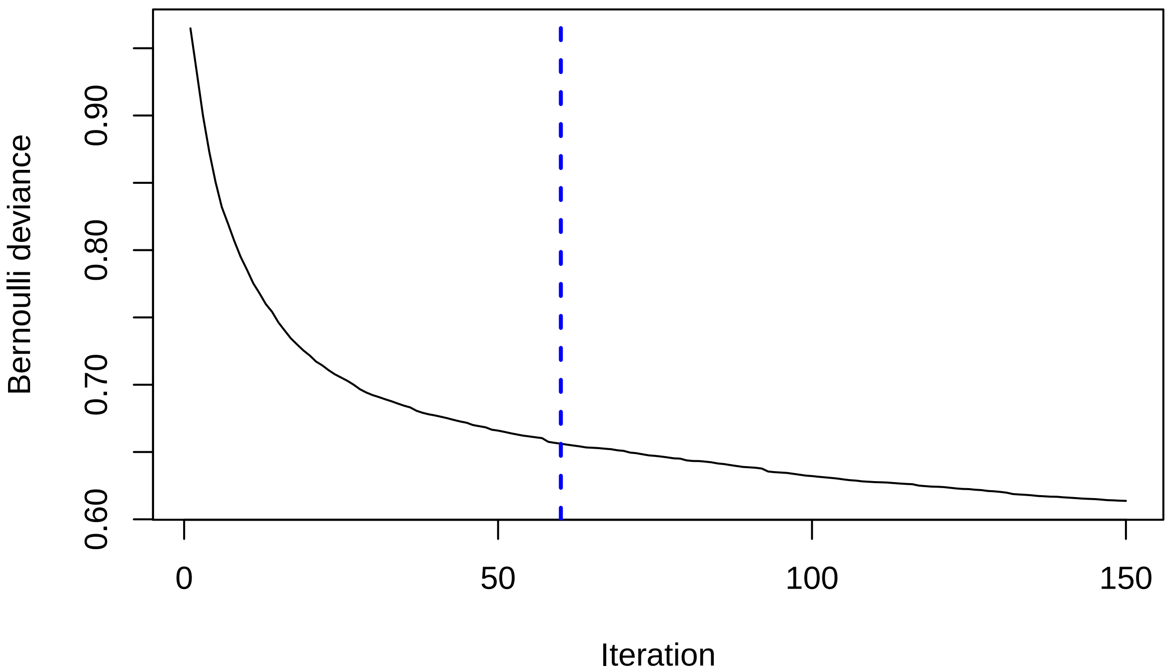
Estimated best number of trees is 144, so similar to the previous one. There is no need for change in the number of trees.

3.4.3. Boosting 3

```
boosting.model.3 <- gbm(as.character(Exited)~.,data = df[train,], distribution = "bernoulli", n.trees =
boosting.pred.3 <- predict(boosting.model.3, newdata = df[valid,], n.trees = 150, type = "response")
boosting.pred.3 <- ifelse(boosting.pred.3 >= .5, 1, 0)
cm.boosting.model.3 <- confusionMatrix(data = factor(boosting.pred.3), factor(df[valid,]$Exited), posit
```

Sensitivity of the third gradient boosting model is equal to 0.503125, so there is a significant increase. Now the optimal number of trees will be double-checked.

```
ntree.opt.oob.3 <- gbm.perf(boosting.model.3, method = "OOB", plot.it = T)
```



Estimated best number of trees is 60, so a decrease in the number of trees may improve the predictive power.

3.4.4. Boosting 4

```
boosting.model.4 <- gbm(as.character(Exited)~.,data = df[train,], distribution = "bernoulli", n.trees =
boosting.pred.4 <- predict(boosting.model.4, newdata = df[valid,], n.trees = 60, type = "response")
boosting.pred.4 <- ifelse(boosting.pred.4 >= .5, 1, 0)
cm.boosting.model.4 <- confusionMatrix(data = factor(boosting.pred.4), factor(df[valid,]$Exited), posit
```

Sensitivity of the fourth gradient boosting model is equal to 0.48125, so unfortunately the predictive power has not been improved. In comparison, the third model of all gradient boosting models is the best one.

```
which.max(c(cm.boosting.model.1$byClass["Sensitivity"],
cm.boosting.model.2$byClass["Sensitivity"],
cm.boosting.model.3$byClass["Sensitivity"],
cm.boosting.model.4$byClass["Sensitivity"]))
```

```
## Sensitivity
##           3
```

4. Final prediction

Now each best model of each type will do the prediction on a test set. Then their predictive power will be compared and the best model will be interpreted.

```r
tree.pred.final <- predict(tree.model.2, newdata = df[test,], type = "class")
forest.pred.final <- predict(forest.model.1, newdata = df[test,], type = "class")
bagging.pred.final <- predict(bagging.model.1, newdata = df[test,], type = "class")
boosting.pred.final <- predict(boosting.model.3, newdata = df[test,], n.trees = 60, type = "response")

cm.tree.model.final <- confusionMatrix(data = tree.pred.final, df[test,]$Exited, positive = "1")
cm.forest.model.final <- confusionMatrix(data = forest.pred.final, df[test,]$Exited, positive = "1")
cm.bagging.model.final <- confusionMatrix(data = bagging.pred.final, df[test,]$Exited, positive = "1")
boosting.pred.final <- ifelse(boosting.pred.final >= .5, 1, 0)
cm.boosting.model.final <- confusionMatrix(data = factor(boosting.pred.final), df[test,]$Exited, positi

which.max(c(cm.tree.model.final$byClass["Sensitivity"],
            cm.forest.model.final$byClass["Sensitivity"],
            cm.bagging.model.final$byClass["Sensitivity"],
            cm.boosting.model.final$byClass["Sensitivity"]))
```

```
## Sensitivity
##           1
```

The best model is the tree model No. 2 and this one will be interpreted.

5. Interpretation

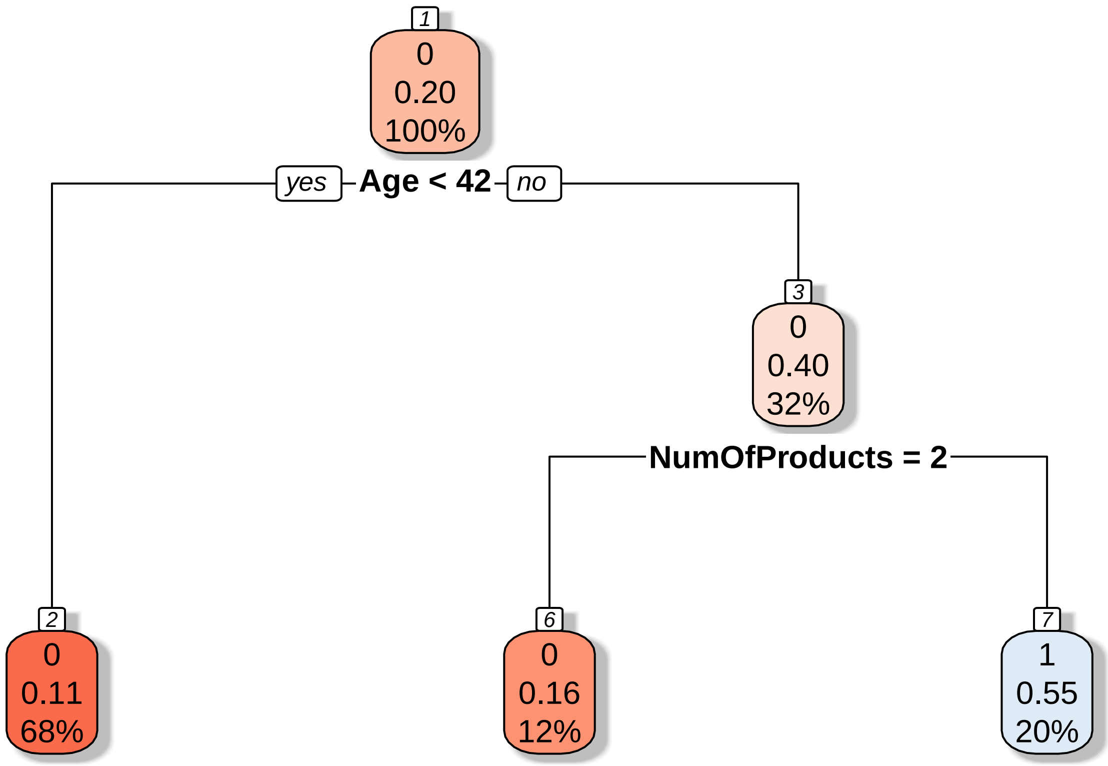
The confusion matrix of the chosen tree model looks as follows:

```r
cm.tree.model.final$table
```

```
##           Reference
## Prediction    0    1
##          0 1019  151
##          1  147  147
```

The number of true positive values is a little less than the number of false positive. It results in the sensitivity being equal to . On the other hand the accuracy is relatively high, being equal to 0.7964481. The tree lookes like this:

```r
rpart.plot(tree.model.2, box.palette="RdBu", shadow.col="gray", nn=TRUE)
```



The highest risk of churn is among clients who are 42 years old or older and have 0, 1 or more than 2 products of the company. Is it highly recommended to organize a marketing campaign aiming in these clients to retain them.