Summer Project Internship

May - July, 2022

Customer Churn Analysis on Banking Dataset

Submitted by
Sampriti Dutta, *M.Sc.(Statistics), IIT Kanpur*
Sankhadeep Mitra, *M.Sc.(Statistics), IIT Kanpur*

under the supervision of
Dr. Biswabrata Pradhan
SQC and OR Unit
ISI, Kolkata

Acknowledgement

We would like to express a deep sense of thanks and gratitude to Dr. Biswabrata Pradhan for providing us the opportunity to prepare this project and constantly motivating us with constructive advices. It has been a great learning experience building practical insights of the theoretical knowledge gathered during the project.

Last, but not the least, our parents provided us with continuous encouragement and extensive support throughout the session. So, with due regards we express our gratitude to them for completion of the project within the stipulated time-period.

ABSTRACT

Banking is one of those traditional industries that has gone through a steady transformation over the decades, yet many banks today with a sizeable customer base hoping to gain a competitive edge have not tapped into the vast amounts of data they have, especially in solving one of the most acknowledged problems – customer churn. Churn is expressed as a degree of customer inactivity or disengagement, observed over a given time. This manifests within the data in various forms such as the recency of account actions or change in the account balance. While retaining existing customers and thereby increasing their lifetime value is something everyone acknowledges as being important, there is little the banks can do about customer churn when they don't see it coming in the first place. This is where predicting churn at the right time becomes important, especially when clear customer feedback is absent.

This project targets to retain as many customers as possible in the bank and to know what leads a client towards the decision to leave the bank on the basis of the predictors like credit-score, age, gender, balance, estimated salary etc.

At first we have a look of the shape and description of the data and address the Data Imbalance Issue and apply Synthetic Minority Oversampling Technique (SMOTE) to oversample the minority class labels and then we perform Exploratory Data Analysis. After that we build Classification Tree and Regression Tree by estimating the survival probability through Cox Proportional Hazards Model and finally analyze and evaluate the performance of the models using Brier Score .

Contents

1 Introduction

Customer churn (also known as customer attrition) refers to when a customer (player, subscriber, user, etc.) ceases his or her relationship with a company. The full cost of churn includes both lost revenue and the marketing costs involved with replacing those customers with new ones. Reducing churn is a key goal of every business.

The ability to predict that a particular customer is at a high risk of churning, while there is still time to do something about it, represents a huge additional potential revenue source for every business. Besides the direct loss of revenue that results from a customer abandoning the business, the costs of initially acquiring that customer may not have already been covered by the customer's spending to date. (In other words, acquiring that customer may have actually been a losing investment.) Furthermore, it is always more difficult and expensive to acquire a new customer than it is to retain a current paying customer.

1.1 Objectives of the Study

In order to succeed at retaining customers who would otherwise abandon the business, marketers and retention experts must be able to predict in advance which customers are going to churn through churn analysis and know which actions will have the greatest retention impact on each particular customer. Armed with this knowledge, a large proportion of customer churn can be eliminated.

To meet this purpose we have fitted the Classification Tree and Regression Tree on the censored customer churn dataset of a bank .

- After dealing with Data Imbalance using SMOTE, dummy variables are being introduced for the categorical variables and then the fitted classification tree, bagging and random forest model have given an insight about the churn of the customers.
- The Cox model is then fitted to deal with the censored observations in the way to predict the survival probability of the customers. After that the regression tree and random forest regression is fitted on the training datasets in order to predict the churn probability of the retained customers so that proper actions can be taken to curb the attrition of the customers who have high chance.
- All the model's accuracy have been checked and compared to get an idea about how well the prediction works on any dataset by which the model is not trained.

2 Methodology

In the previous section we have briefly explained about the problem statement of customer churn in a bank. In this section, we will provide step by step procedure. At first we introduce the dataset, collected from the kaggle website, which has 13 different columns among which all are not significant with respect to the prediction problem and the last one **Exited** is the response variable. The processed data has been used for model fitting. Here is a quick view of the columns presented in the data.

2.1 Dataset Description

Variable name	Description
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 the bank.
Surname	The surname of a customer has no impact on their decision to leave the bank.
CreditScore	Can have an effect on customer churn, since a customer with a higher credit score is less likely to leave the bank.
Geography	A customer's location can affect their decision to leave the bank.
Gender	It's interesting to explore whether gender plays a role in a customer leaving the bank.
Age	This is certainly relevant, since older customers are less likely to leave their bank than younger ones.
Tenure	Refers to the number of years that the customer has been a client of the bank. Normally, older clients are more loyal and less likely to leave a bank.
Balance	A very good indicator of customer churn, as people with a higher balance in their accounts are less likely to leave the bank compared to those with lower balances.
NumOfProducts	Refers to the number of products that a customer has purchased through the bank.
HasCrCard	Denotes whether or not a customer has a credit card. This column is also relevant, since people with a credit card are less likely to leave the bank.
IsActiveMember	Active customers are less likely to leave the bank.
EstimatedSalary	As with balance, people with lower salaries are more likely to leave the bank compared to those with higher salaries.
Exited	Whether or not the customer left the bank.

Table 1 : Description of Explanatory and Response Variables

As shown in Table 1, there are total 13 features. 3 of them named **RowNumber, CustomerId, Surname** are insignificant and are dropped for further work, 4 variables **Geography, Gender, HasCrCard, IsActiveMember** are categorical variable and rest are continuous variable. The dataset consists of 10,000 observations of different individuals. The **Tenure** column is the lifetime or survival time of the customers.

2.2 Dataset Quality Assessment

Before moving on to assessing the quality of the dataset, we first introduce the dummy variables for the categorical variables otherwise this may lead to a lot of trouble in building the model .

2.2.1 Data Imbalance

Imbalanced data refers to those types of datasets where the target class has an uneven distribution of observations, i.e., one class label has a very high number of observations and the other has a very low number of observations. Here one of the shortcomings of our dataset is that it is highly imbalanced (80% of 0's(Not Exited) and 20% of 1's(Exited)). So, we need to deal with the problem of data imbalance.

Data imbalance can be treated using different techniques such as Random undersampling, Random oversampling, cluster based over sampling, Synthetic Minority Oversampling Technique (SMOTE), Modified synthetic minority oversampling technique (MSMOTE) etc. for imbalanced data. Oversampling and Undersampling are opposite and roughly equivalent techniques of dealing with Data Imbalance, where they adjust the class distribution of a dataset. The process of Oversampling increases the class distribution of the minority class label whereas Undersampling decreases the class distribution of the majority class label. In our project, we have used Synthetic Minority Oversampling Technique or SMOTE to deal with the data imbalance problem.

Synthetic Minority Oversampling Technique for Imbalanced Data (SMOTE)

One of the widely used oversampling technique is 'SMOTE'. To illustrate how this technique works consider some training data which has s samples and f features in the feature space of the data. For simplicity, assume the features are continuous. As an example, let us consider a dataset of birds for clarity. The feature space for the minority class for which we want to oversample could be beak length, wingspan and weight. To oversample, take a sample from the dataset, and consider its k nearest neighbors in the feature space. To create a synthetic data point, take the vector between one of those k neighbors, and the current data point. Multiply this vector by a random number x which lies between 0 and 1. Adding this to the current data point will create the new synthetic data point. SMOTE was implemented from the *smotefamily* library.
The dataset consists of 10,000 observations. After applying SMOTE, a total of 14074 observations are obtained, among which there are 7963 0's and 6111 1's.

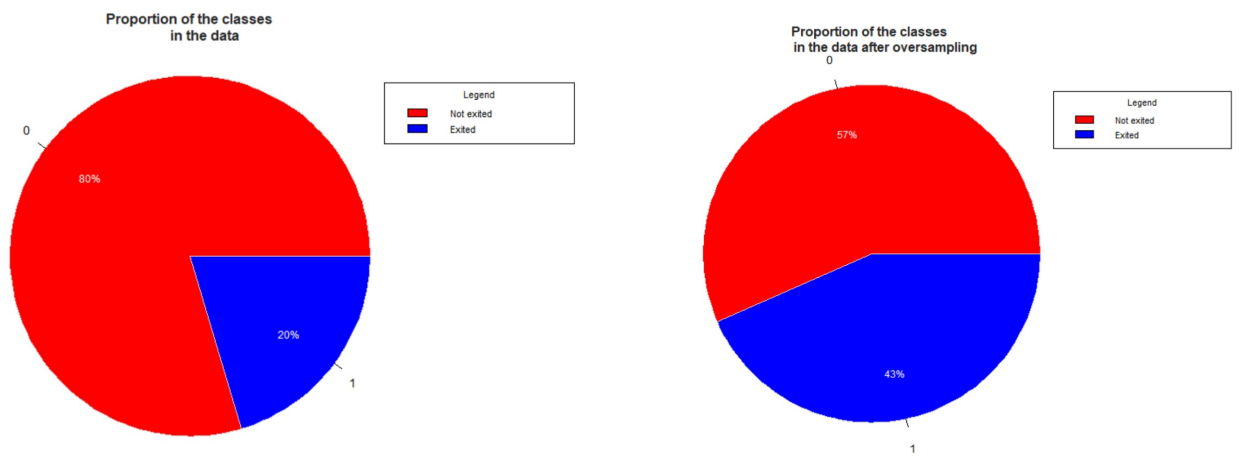


Fig. 1 : The proportion of response classes. Left: Before oversampling. Right: After oversampling.

3 Train and Test Data

To fit a model on a data the procedure involves taking that dataset and dividing it into two subsets.
1.Training set: a subset of the data which is used to train or teach our method to estimate the model.
2.Testing Set: a subset of the data which is not seen by the model and used to test the trained model. Performance of the model is measured using the test data.

Here, out of the 14,074 observations we have sampled 80% (11254) observation for the training set and remaining 20% (2820) observation for the test set.

We fitted the models on the training dataset and obtained the summary of the respective models. After that we used that fitted model to predict the response variable for the testing dataset and then measured how well the fitted model could predict the responses compared to the observed value of the response variable.

4 Model Building

4.1 Classification Trees

A classification tree is used to predict a qualitative response rather than a quantitative one. For a classification tree, we predict that each observation belong to the most commonly occurring class of training observations in the region to which it belongs. In interpreting the results of a classification tree, we are often interested not only in the class prediction corresponding to a particular terminal node region, but also in the class proportions among the training obsevations that fall in that region.

We use recursive binary splitting to grow a classification tree. In the classification tree, classification error rate can be used as a criterion for making the binary splits. Since we plan to assign an observation in a given region to the most commonly occurring class of training observations in that region, the classification error rate is simply the fraction of the training observations in that region that do not belong to the most common class:

$$E = 1 - \max_k(\hat{p}_{mk})$$

Here $\hat{p}_{mk}$ represents the proprtion of training observations in the m^{th} region that are from the k^{th} class. However it turns out that classification error rate is not sufficiently sensitive for tree growing, and in practice two other measures are preferable.

The Gini index defined by

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

is a measure of total variance across the K classes. It is not hard to see that Gini index takes a small value if all of the $\hat{p}_{mk}$'s are close to zero or one. For this reason it is referred to as a measure of node purity - a small value indicates that a node contains predominantly observations from a single class.

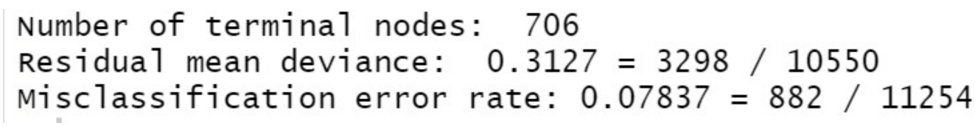
An alternative to the Gini index is cross-entropy, given by

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} log \hat{p}_{mk}$$

.

Cross-entropy also takes a value near zero if all of the $\hat{p}_{mk}$'s are close to zero or one. In fact Gini index and cross entropy are quite similar numerically.

When building a classification tree, either the Gini index or cross entropy are typically used to evaluate the quality of a particular split, since these two approches are more sensitive to node purity than is the classification error rate. In our problem we have taken the Gini index as the splitting criterion of the tree. The tree predicts for the customers in the testing dataset, if he/she exited from the bank. We have modelled the classification tree with the train data using *tree* package from R. Here is the summary and confusion of the fitted tree.



```
Number of terminal nodes:  706
Residual mean deviance:  0.3127 = 3298 / 10550
Misclassification error rate: 0.07837 = 882 / 11254
```

The error rate of the model, fitted on the training data is **7.837%** and for testing data is **13.546%**. So, the model provides a good result for the test data too. Though the misclassification error rate and residual mean deviance of the tree is small for the training data, the number of terminal nodes in the tree is 706, which is quite large and is complex to interpret. Sometimes this kind of large tree causes overfitting to the training data, which is not required. Therefore the tree is needed to be pruned in further steps and to give a better accuracy in terms of fitting, for the test data.

4.2 Tree Pruning

Sometimes the resulting tree may produce good predictions on the training set, but is likely to overfit the data, leading to poor test performance. This is because the resulting tree might be too complex. A smaller tree with fewer splits might lead to lower variance and better interpretation at the cost of a little bias. One possible alternative to the process described above is to build the tree only so long as the decrease in the residual sum of squares (RSS) due to each split exceed some threshold. This strategy will result in smaller trees, but is too short-sighted since a seemingly worthless split early on in the tree might be followed by a very good split, that is, a split that leads to a large reduction in RSS later on.

A better strategy is to grow a very large tree T_0, and then prune it back in order to obtain a subtree. Our goal is to select a subtree that leads to the lowest test error rate. Given a subtree, we can estimate its test error using cross validation or the validation set approach. However cross validation error is sometimes too cumbersome .

Cost complexity pruning, also known as weakest link pruning gives us a way to select a small set of subtrees for consideration, indexed by a non-negative tuning parameter α. For each value of α there corresponds a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

is as small as possible. Here $|T|$ indicates the number of terminal nodes of the tree T, R_m is the rectangle corresponding to the m^{th} terminal node, and $\hat{y}_{R_m}$ is the predicted response associated with R_m, that is, the mean of the training observations in R_m. The tuning parameter α controls a trade-off between the subtree's complexity and its fit to the training data.

As discussed above, we have checked for the best pruned subtree with respect to the cross validation error rate and plot them using *tree* from R packages. The summary of the study is

```
> cv$size
 [1] 706 329 323 318 302 295 271 266 238 233 224 212 206 183 175 169 164 150
[19] 135 123 114 110 105  98  93  87  81  73  71  66  63  53  49  45  42  39
[37]  36  35  33  32  31  28  27  26  25  24  22  21  20  18  16  15  11  10
[55]   9   8   6   4   1
> cv$dev
 [1] 1796 1796 1796 1796 1796 1796 1796 1796 1796 1796 1796 1796 1796 1796
[15] 1796 1796 1796 1796 1796 1796 1796 1796 1796 1796 1796 1796 1796 1796
[29] 1796 1796 1796 1796 1796 1796 1796 1796 1796 1796 1796 1796 1796 1796
[43] 1796 1796 1796 1796 1796 1967 1983 1983 2025 2126 2126 2171 2285 2619
[57] 2771 2771 2771
```

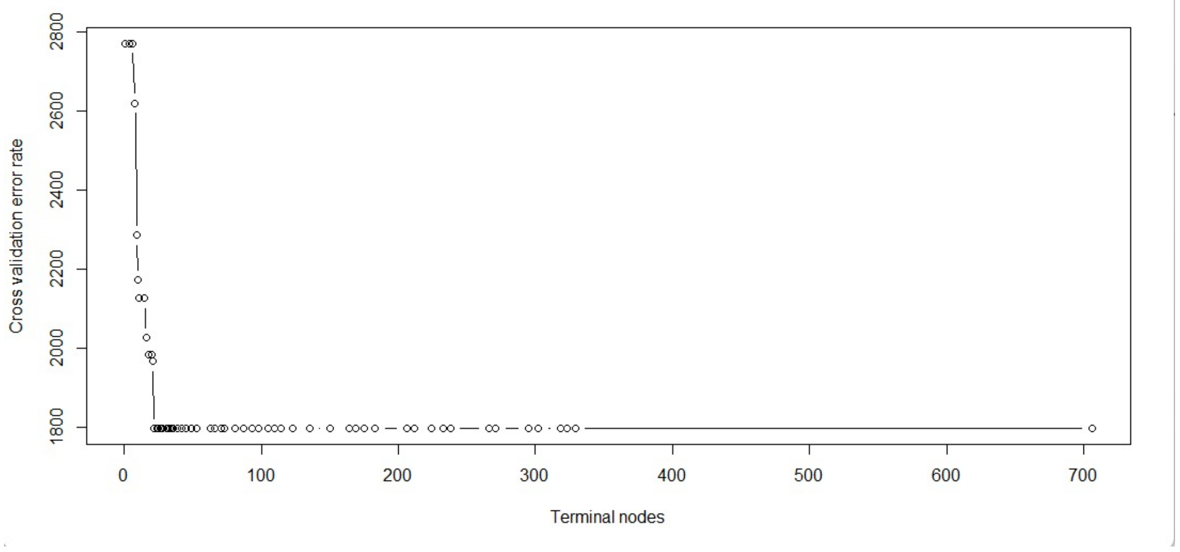
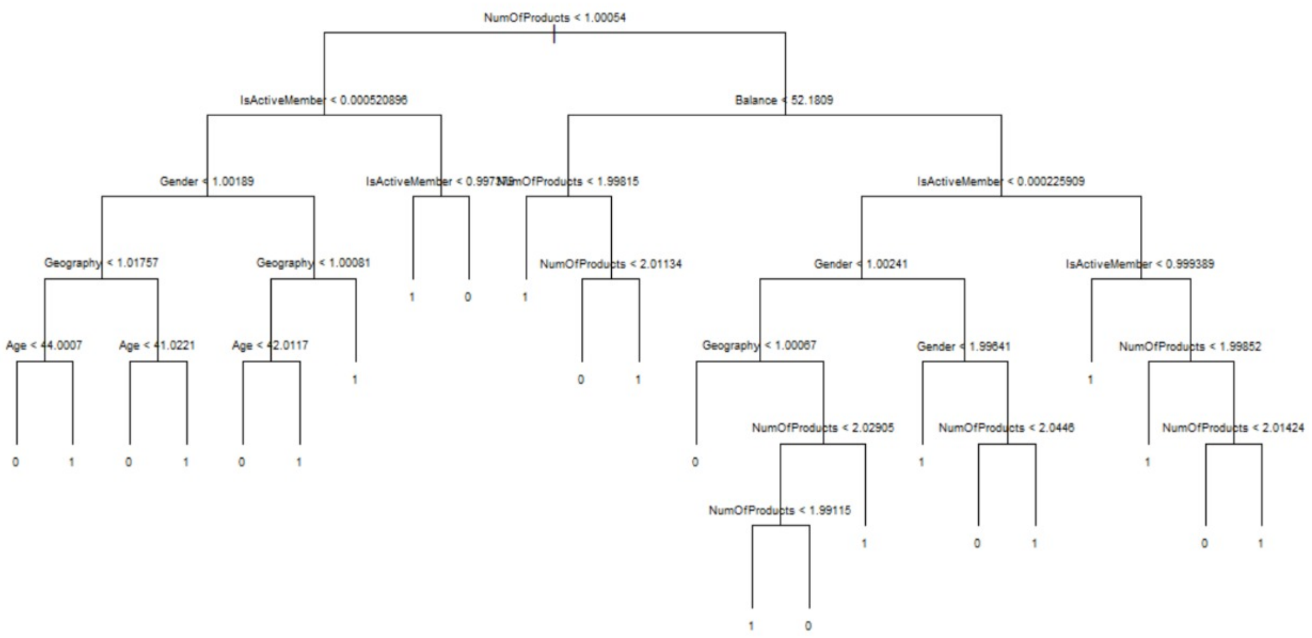


Fig. 2 : The cross validation error rate shown for different sizes of the pruned tree

The best pruned tree according to the cross validation score is achieved where the number of terminal node is 23. This tree with 23 terminal nodes gives an accuracy of **83.89%** for train data and **83.62%** for the test data.



Variable	IsActiveMember	Gender	Geography
Index	1 = Yes, 0 = No	1 = Male, 0 = Female	1 = France, 2 = Germany, 3 = Spain

Fig. 3 : The Pruned tree with 23 terminal nodes.

4.3 Regression Trees

A regression tree is built through a process known as binary recursive partitioning, which is an iterative process that splits the data into partitions or branches, and then continues splitting each partition into smaller groups as the method moves up each branch.

Initially, all records in the Training Set (pre-classified records that are used to determine the structure of the tree) are grouped into the same partition. The algorithm then begins allocating the data into the first two partitions or branches, using every possible binary split on every field. The algorithm selects the split that minimizes the sum of the squared deviations from the mean in the two separate partitions. This splitting rule is then applied to each of the new branches. We repeat this process, looking for the best predictor and best cutpoint in order to split the data further so as to minimize the RSS within each of the resulting region.

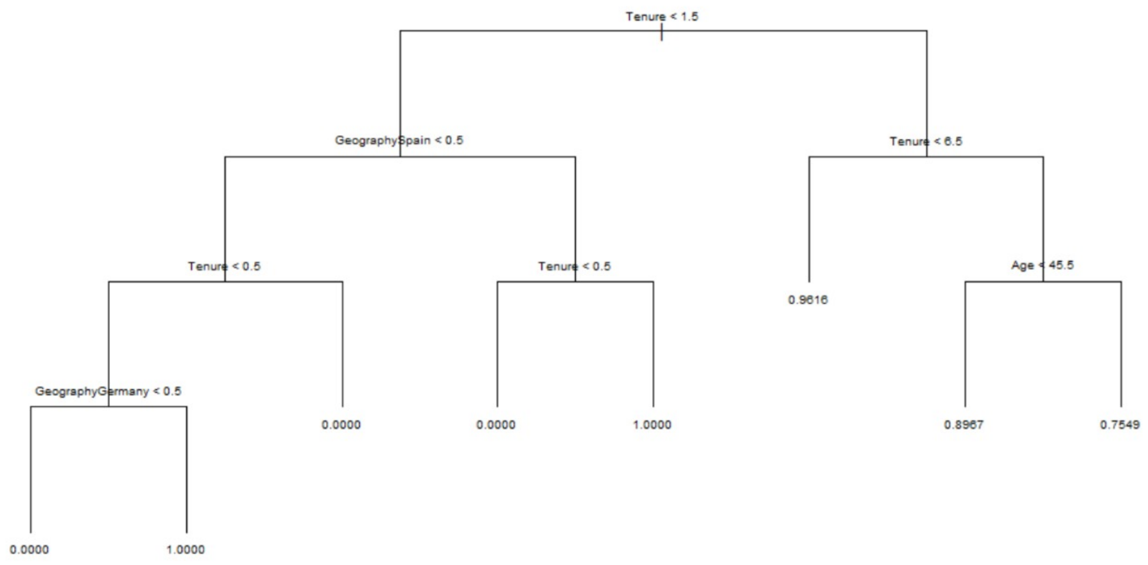
This splitting aim to divide the predictor space into say, J distinct and non-overlapping regions $R_1, R_2, ..., R_J$. The RSS is given by,

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

,where $\hat{y}_{R_j}$ is the mean response for the training observations within the j^{th} box. In this approach instead of splitting of entire predict space, we split one of the two previously identified regions. We now have three regions. Again we look to split one of the three regions further, so as to minimize the RSS. The process continues until a stopping criterion is reached.

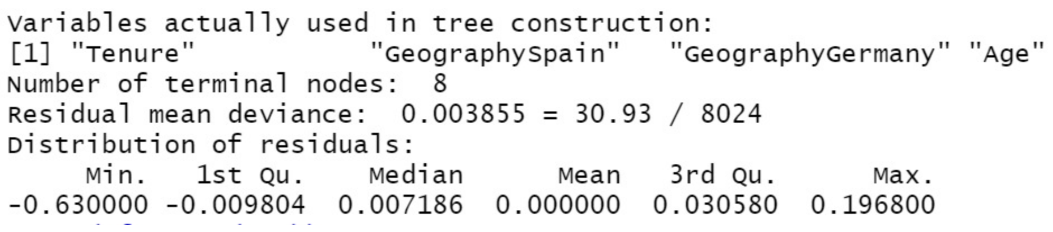
Once the regions $R_1, R_2, ..., R_J$ have been created, we predict the response for a given test observation using the mean of the training observations in the region to which that test observation belongs.

For the discussed problem we are not only aware of getting information about the customer leaves or not, we are also interested to get an idea of the probability of a customer's retention so that required measures can be taken. In that context, Classification tree cannot meet the purpose and we fit Regression tree to predict the survival probability of a customer from the train data. As the data is censored, we have adopted the method of Cox proportional hazards to model the survival probability of the observed individuals. Summary of the fitted regression tree is as follows



Variable	GeographyGermany	GeographySpain
Index	1 = Yes, 0 = No	1 = Yes, 0 = No

Fig. 4 : The Regression Tree fitted on the train data, using the survival probability of the customers.



```
Variables actually used in tree construction:
[1] "Tenure"           "GeographySpain"   "GeographyGermany" "Age"
Number of terminal nodes:  8
Residual mean deviance:  0.003855 = 30.93 / 8024
Distribution of residuals:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.630000 -0.009804  0.007186  0.000000  0.030580  0.196800
```

Here, the regression tree has only 8 terminal nodes and further pruning does not improve the fitting ability of the tree. The RMSE for the fitted tree is 0.0261.

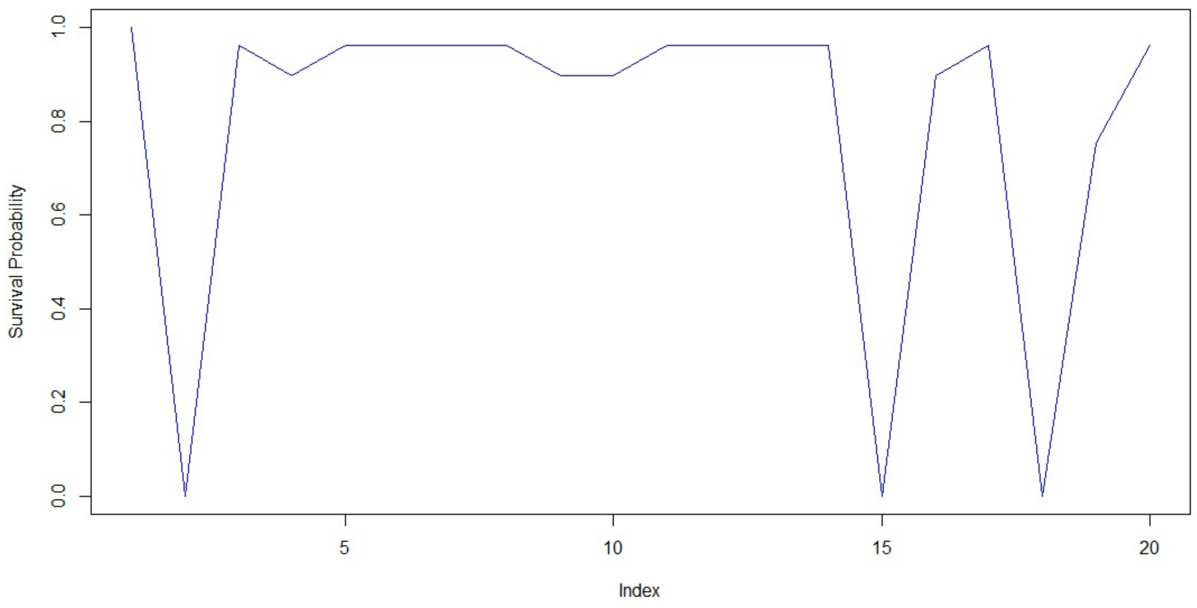


Fig. 5 : Survival probability of 20 individuals from the testing data, as predicted by the Regression Tree

4.4 Bagging

The decision trees suffer from high variance. This means that if we split the training data into two parts at random, and fit a decision tree to both halves, the results that we get could be quite different. In contrast, a procedure with low variance will yield similar results if applied repeatedly to distinct data sets; linear regression tends to have low variance, if the ratio of n to p is moderately large. Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method; we introduce it here because it is particularly useful and frequently used in the context of decision trees.

Recall that given a set of n independent observations $Z_1, Z_2, \cdots, Z_n$ each with variance σ^2, the variance of the mean $\bar{Z}$ of the observations is given by σ^2/n. In other words, averaging a set of observations reduces variance. Hence a natural way to reduce the variance and hence increase the prediction accuracy of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions. In other words, we could calculate $\hat{f}^1(x), \hat{f}^2(x), \cdots, \hat{f}^B(x)$ using B separate training sets, and average them in order to obtain a single low-variance statistical learning model, given by

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{1}^{B} \hat{f}^b(x)$$

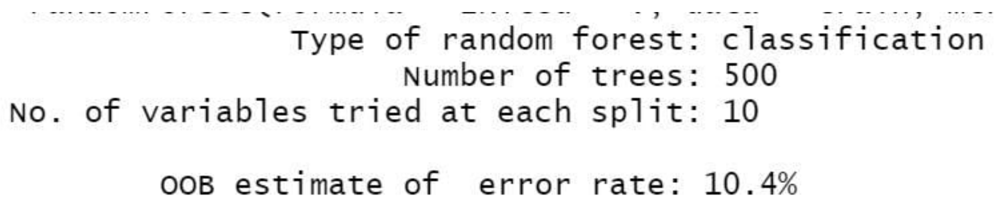
.

Of course, this is not practical because we generally do not have access to multiple training sets. Instead, we can bootstrap, by taking repeated samples from the (single) training data set. In this approach we generate B different bootstrapped training data sets. We then train our method on the b^{th} bootstrapped training set in order to get $\hat{f}^{*b}(x)$, and finally average all the predictions, to obtain

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{1}^{B} \hat{f}^{*b}(x)$$

.

For classification problem, there are a few possible approaches, but the simplest is as follows. For a given test observation, we can record the class predicted by each of the B trees, and take a majority vote: the overall prediction is the most commonly occurring class among the B predictions.



```
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 10

        OOB estimate of  error rate: 10.4%
```

In Bagging model for classification (using *randomForest* package) of the customers, the model has used 500 trees and all the 10 predictors at each split. The OOB error obtained here is 10.4%.

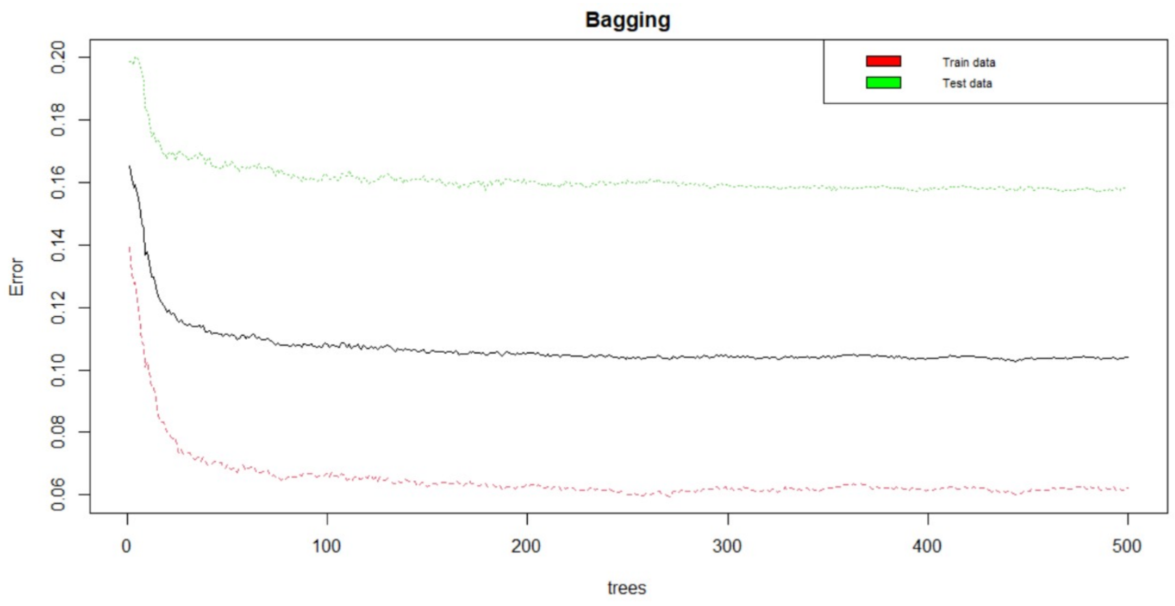


Fig. 6 : The train and test error as a function of the number of bootstrapped training sets used in Bagging.

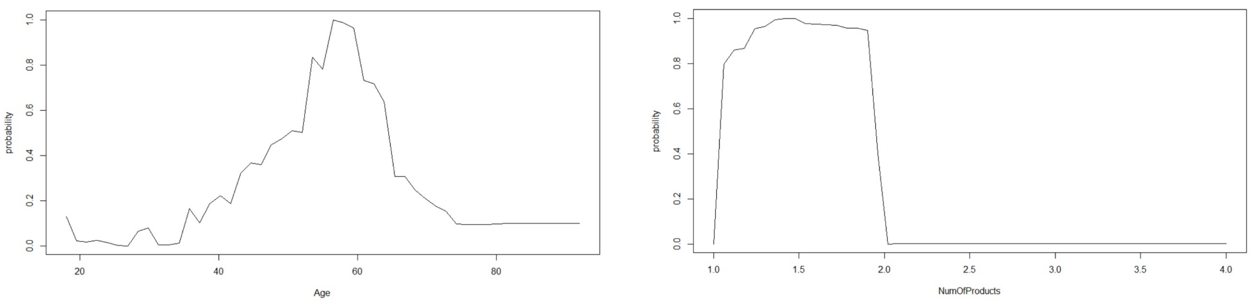


Fig. 7 : The partial dependence of the churn probability on the two most important variables, *Age* and *Number of Products*.

4.5 Random Forest

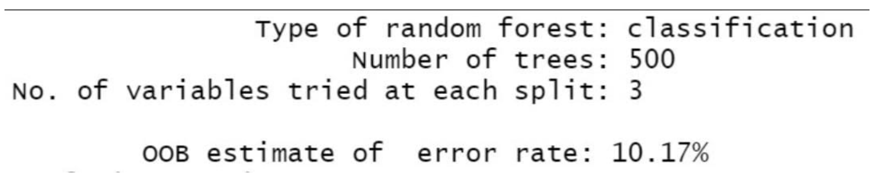
Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees. As in bagging, we build a number of decision trees on bootstrapped training samples. But when building these decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors. A fresh sample of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$, that is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors .

In other words, in building a random forest, at each split in the tree, the algorithm is not even allowed to consider a majority of the available predictors. Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors. Then in the collection of bagged trees, most or all of the trees will use this strong predictor in the top split. Consequently, all of the bagged trees will look quite similar to each other. Hence the predictions from the bagged trees will be highly correlated. Unfortunately, averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities. In particular, this means that bagging will not lead to a substantial reduction in variance over a single tree in this setting.

Random forests overcome this problem by forcing each split to consider only a subset of the predictors. Therefore, on average $(p-m)/p$ of the splits will not even consider the strong predictor, and so other predictors will have more of a chance. We can think of this process as decorrelating the trees, thereby making the average of the resulting trees less variable and hence more reliable.

The main difference between bagging and random forests is the choice of predictor subset size m. For instance, if a random forest is built using $m = p$, then this amounts simply to bagging.

For Random Forest Classification of the customers, the tree has used 500 trees and $\sqrt{10}$ i.e. 3 predictors at each split.



```
                   Type of random forest: classification
                         Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 10.17%
```

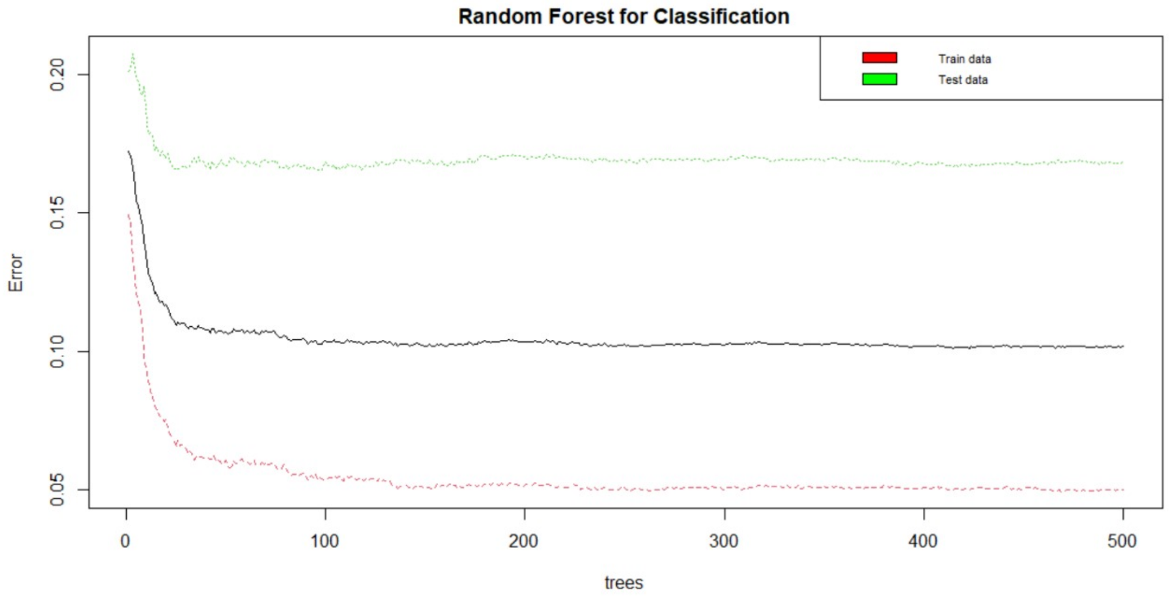


Fig. 8 : The train and test error as a function of the number of bootstrapped training sets used in Random Forest.

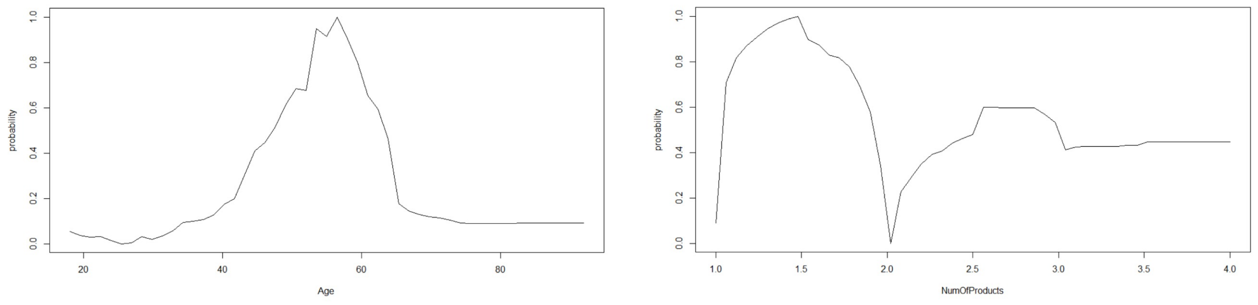


Fig. 9 : The partial dependence of the churn probability on the two most important variables, *Age* and *Number of Products*.

To predict the probability of the attrition of the customers from the bank, random forest regression have been fitted with 500 trees and used 3 variables at each split.The RSS is very low and the model can explain about 97.5% of the variability. The RMSE for the test data is 0.0209, which is better than the regression tree.

```
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 3

          Mean of squared residuals: 0.002268017
                    % Var explained: 97.5
```

4.6 Cox Proportional Hazards Regression

The Cox proportional hazard regression model and its extensions are very often used to study survival variables with censoring. These parametric (and semi–parametric) models are quite useful, for they allow simple interpretations of the covariate effects and can readily be used for inference (hypothesis testing and so on). However, such models force a specific link between the covariates and the response. Even though interactions between covariates can be incorporated, they must be specified by the analyst.In the Cox model, the proportional hazard assumption implies that the hazard function in the right node is proportional to the one in the left node. A Cox proportional hazard model is used to model the several covariates. In other words, it allows us to examine how specified factors influence the rate of a particular event happening at a particular point in time. This rate is commonly referred as the hazard rate.

The Cox Proportional Hazards Model is usually given in terms of the time t, covariate vector x, and coefficient vector β as

$$\lambda(t) = \lambda_0(t)e^{x^T\beta}$$

, where the λ_0 is an arbitrary function of time, the baseline hazard. The dot product of X and β is taken in the exponent just like in standard linear regression. Regardless of the values covariates, all subjects share the same baseline hazard . Thereafter, adjustments are made based on the covariates.

$x^T\beta = \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip}$ and $X_i = (X_{i1}, X_{i2}, \cdots, \mathrm{X}_{ip})$,be the realized values of the covariates for subject i.

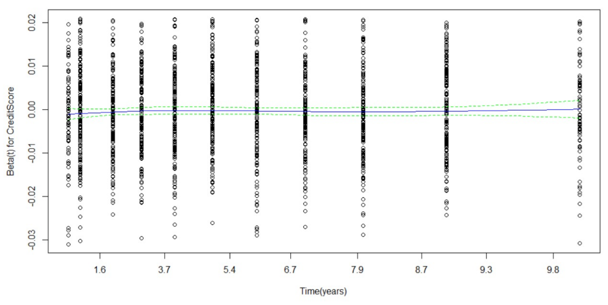
If the proportional hazards assumption holds, then logarithm of cumulative hazard becomes

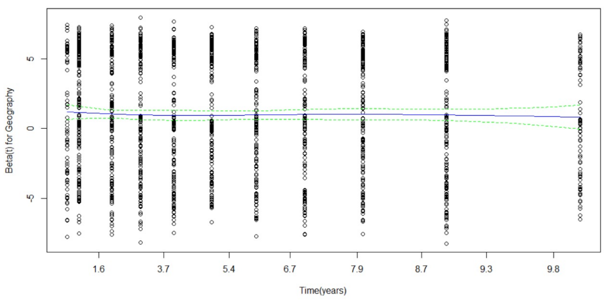
$$log(\Lambda(t|Z)) = log(\Lambda_0(t).exp\beta^T Z) = \beta^T Z + log(\Lambda_0(t)),$$

which means that the difference of log-cumulative hazard of two individuals is independent of time:

$$log(\Lambda(t|Z_1)) - log(\Lambda(t|Z_2)) = \beta^T(Z_1 - Z_2).$$

Therefore, plotting such difference can be a useful tool for revealing potential problems with satisfying the proportional hazards assumption:





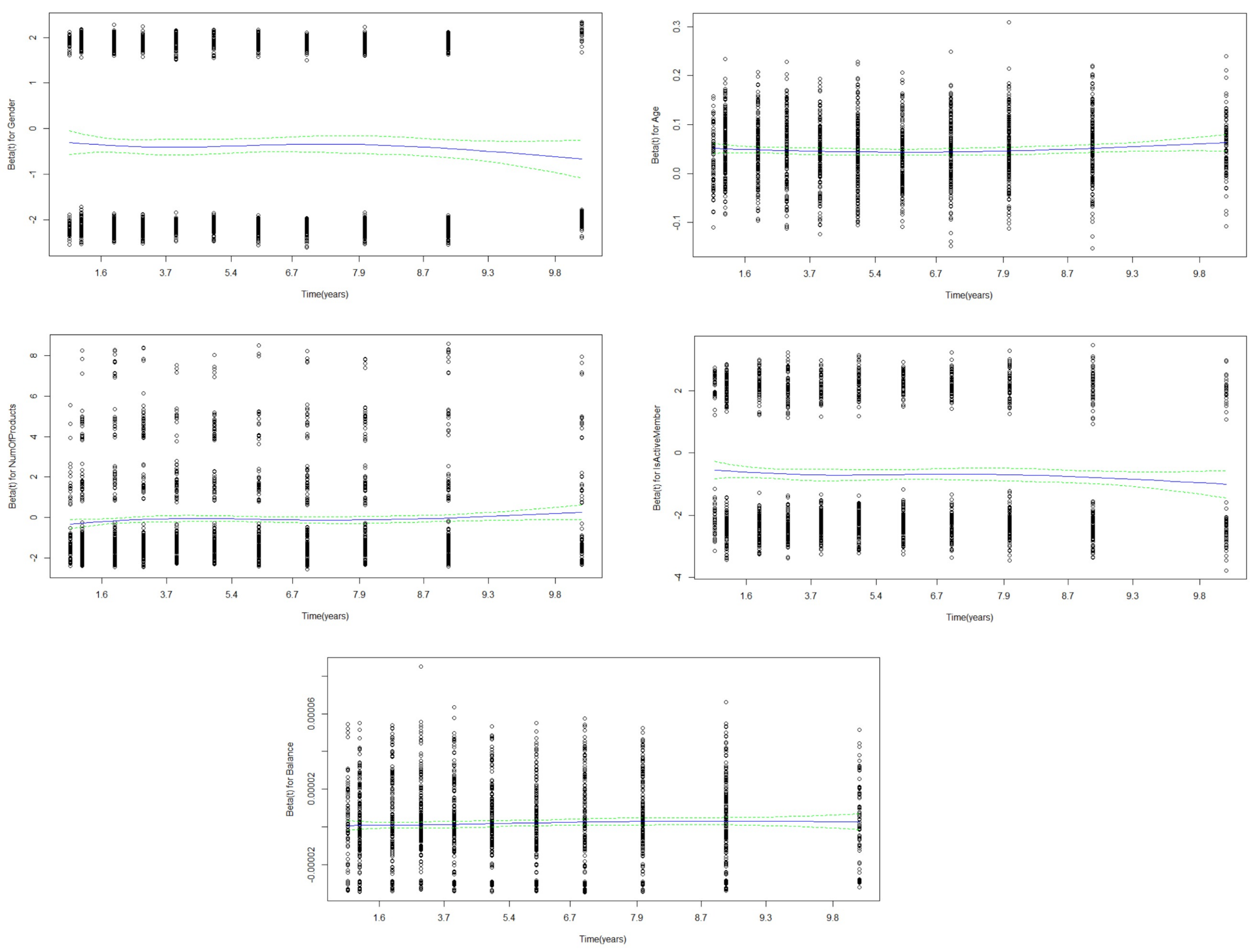
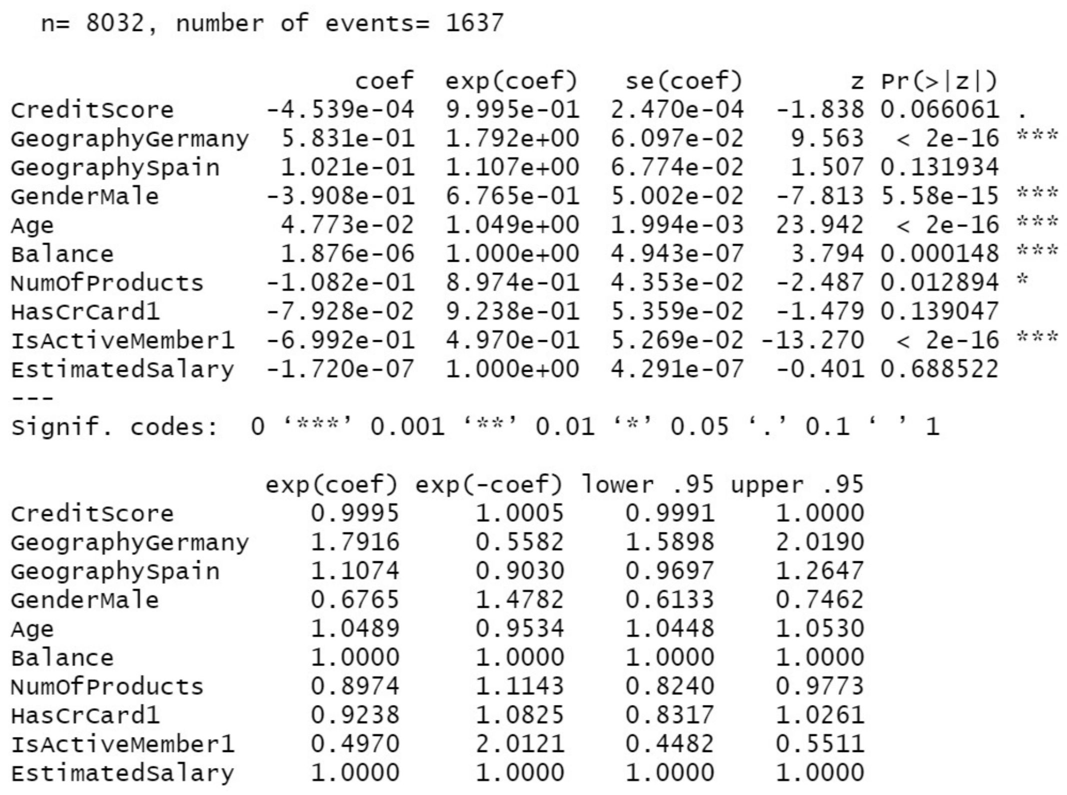


Fig. 10 : Difference in log cumulative hazards for each covariate used in the Cox model as a function of time.

We have performed Cox proportional hazards regression including the main effects of all covariates on the training dataset with the help of the funcion *coxph()* from *survival* package and estimated the coefficients of the covariates and the corresponding p-values, obtained through Cox Propotional Hazard Model:

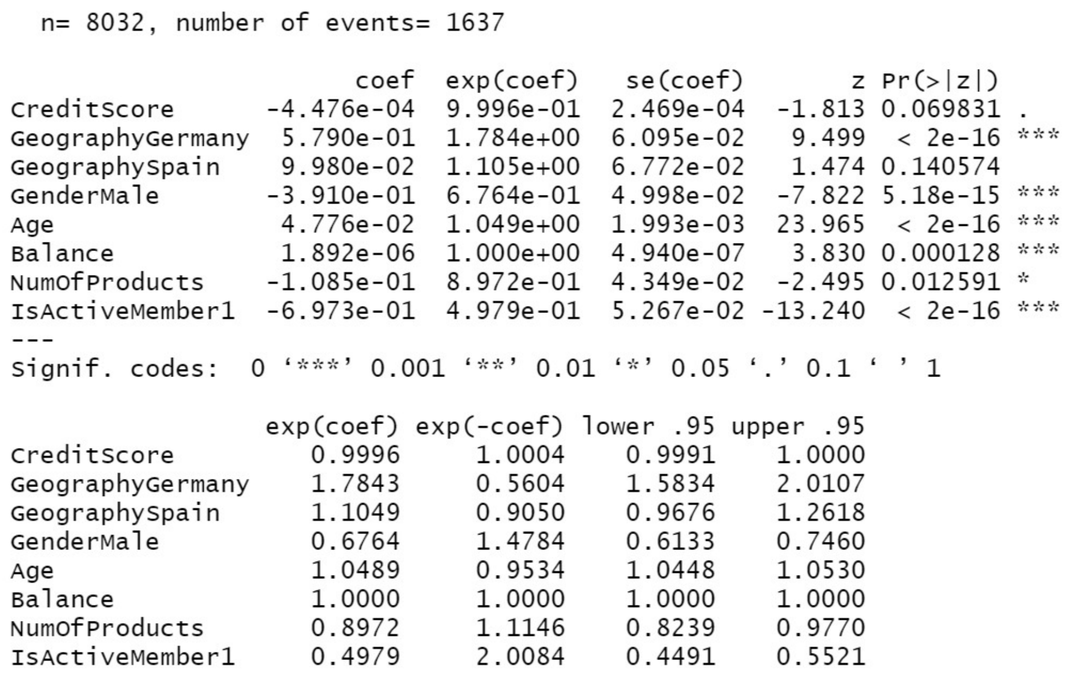


```
  n= 8032, number of events= 1637

                      coef  exp(coef)   se(coef)        z Pr(>|z|)    
CreditScore     -4.539e-04  9.995e-01  2.470e-04   -1.838 0.066061 .  
GeographyGermany  5.831e-01  1.792e+00  6.097e-02    9.563  < 2e-16 ***
GeographySpain   1.021e-01  1.107e+00  6.774e-02    1.507 0.131934    
GenderMale      -3.908e-01  6.765e-01  5.002e-02   -7.813 5.58e-15 ***
Age              4.773e-02  1.049e+00  1.994e-03   23.942  < 2e-16 ***
Balance          1.876e-06  1.000e+00  4.943e-07    3.794 0.000148 ***
NumOfProducts   -1.082e-01  8.974e-01  4.353e-02   -2.487 0.012894 *  
HasCrCard1      -7.928e-02  9.238e-01  5.359e-02   -1.479 0.139047    
IsActiveMember1 -6.992e-01  4.970e-01  5.269e-02  -13.270  < 2e-16 ***
EstimatedSalary -1.720e-07  1.000e+00  4.291e-07   -0.401 0.688522    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

                exp(coef) exp(-coef) lower .95 upper .95
CreditScore        0.9995     1.0005    0.9991    1.0000
GeographyGermany   1.7916     0.5582    1.5898    2.0190
GeographySpain     1.1074     0.9030    0.9697    1.2647
GenderMale         0.6765     1.4782    0.6133    0.7462
Age                1.0489     0.9534    1.0448    1.0530
Balance            1.0000     1.0000    1.0000    1.0000
NumOfProducts      0.8974     1.1143    0.8240    0.9773
HasCrCard1         0.9238     1.0825    0.8317    1.0261
IsActiveMember1    0.4970     2.0121    0.4482    0.5511
EstimatedSalary    1.0000     1.0000    1.0000    1.0000
```

Again removing the covariates **HasCrCard1, EstimatedSalary**, which are not significant at **5%** level. The Cox model is fitted again :



```
  n= 8032, number of events= 1637

                      coef  exp(coef)   se(coef)        z Pr(>|z|)    
CreditScore     -4.476e-04  9.996e-01  2.469e-04   -1.813 0.069831 .  
GeographyGermany  5.790e-01  1.784e+00  6.095e-02    9.499  < 2e-16 ***
GeographySpain   9.980e-02  1.105e+00  6.772e-02    1.474 0.140574    
GenderMale      -3.910e-01  6.764e-01  4.998e-02   -7.822 5.18e-15 ***
Age              4.776e-02  1.049e+00  1.993e-03   23.965  < 2e-16 ***
Balance          1.892e-06  1.000e+00  4.940e-07    3.830 0.000128 ***
NumOfProducts   -1.085e-01  8.972e-01  4.349e-02   -2.495 0.012591 *  
IsActiveMember1 -6.973e-01  4.979e-01  5.267e-02  -13.240  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

                exp(coef) exp(-coef) lower .95 upper .95
CreditScore        0.9996     1.0004    0.9991    1.0000
GeographyGermany   1.7843     0.5604    1.5834    2.0107
GeographySpain     1.1049     0.9050    0.9676    1.2618
GenderMale         0.6764     1.4784    0.6133    0.7460
Age                1.0489     0.9534    1.0448    1.0530
Balance            1.0000     1.0000    1.0000    1.0000
NumOfProducts      0.8972     1.1146    0.8239    0.9770
IsActiveMember1    0.4979     2.0084    0.4491    0.5521
```

For example, the predicted survival probabilities of the first individual from the test data are

Time(years)	Survival Probability
1	0.974
2	0.957
3	0.938
4	0.915
5	0.889
6	0.859
7	0.824
8	0.772
9	0.687
10	0.567

Table 2 : The survival probability of a particular individual.

This individual possesses specific covariate values as follows

```
CreditScore Geography Gender Age Tenure  Balance NumOfProducts HasCrCard
        608     Spain      0  41      1 83807.86             1         0
IsActiveMember EstimatedSalary Exited delta GeographyGermany GeographySpain
             1        112542.6      0 FALSE                0              1
```

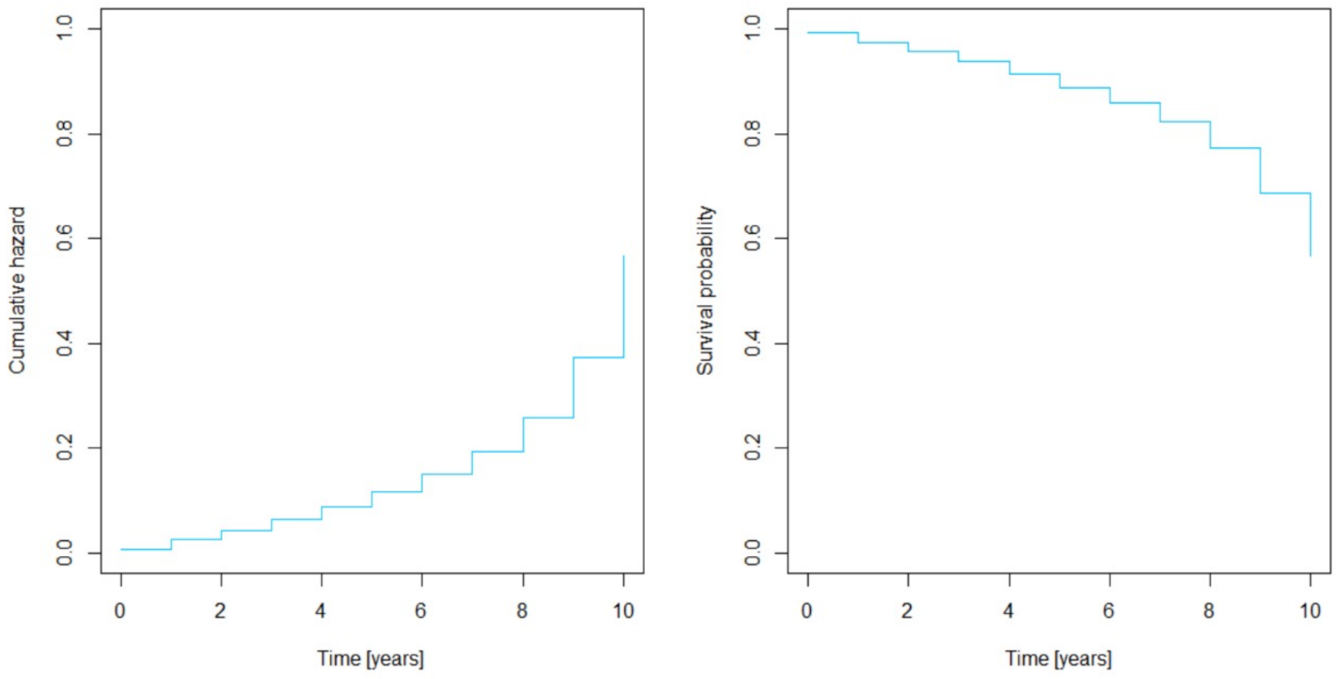


Fig. 11 : The cumulative hazard rate and survival probability of a particular individual from the test data, observed over different years

Considering these survival probabilities, the number of individuals exposed at the risk (here attrition) at different time points is are

Time(years)	**No. of Customers**
0	7698
1	6870
2	6024
3	5232
4	4437
5	3620
6	2838
7	2015
8	1179
9	374
10	0

Table 3 : The count of individuals who are at the risk of attrition for different years.

5 Variable Importance Measures

The collection of bagged trees is much more difficult to interpret than a single tree, one can obtain an overall summary of the importance of each predictor using the RSS (for bagging regression trees) or the Gini index (for bagging classification trees). In the case of bagging regression trees, we can record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees. A large value indicates an important predictor. Similarly, in the context of bagging classification trees, we can add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees.

We pursue the analysis by studying the importance of the covariates. Our approach is compute-intensive and has been used to measure the performance of different covariates. Each covariate is removed one at a time. We then estimate the errors (MSE, node purity for regression and accuracy, Gini index for classification) without the covariate. Thus, the whole process is repeated ten times (one time for each covariate). Fig. 7 represents the percent increase in the respective errors when a single covariate is removed compared to the model with all covariates.

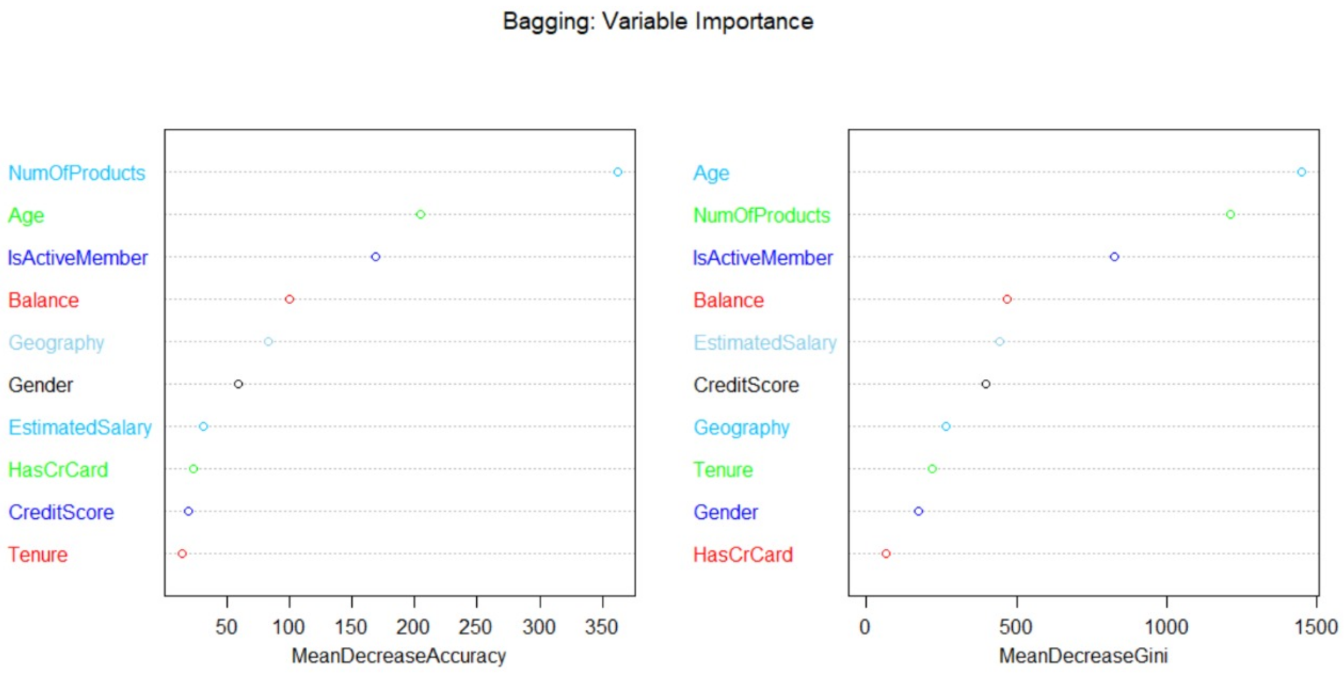


Fig. 12 : Mean decrease accuracy and mean decrease Gini of the different covariates for Bagging.

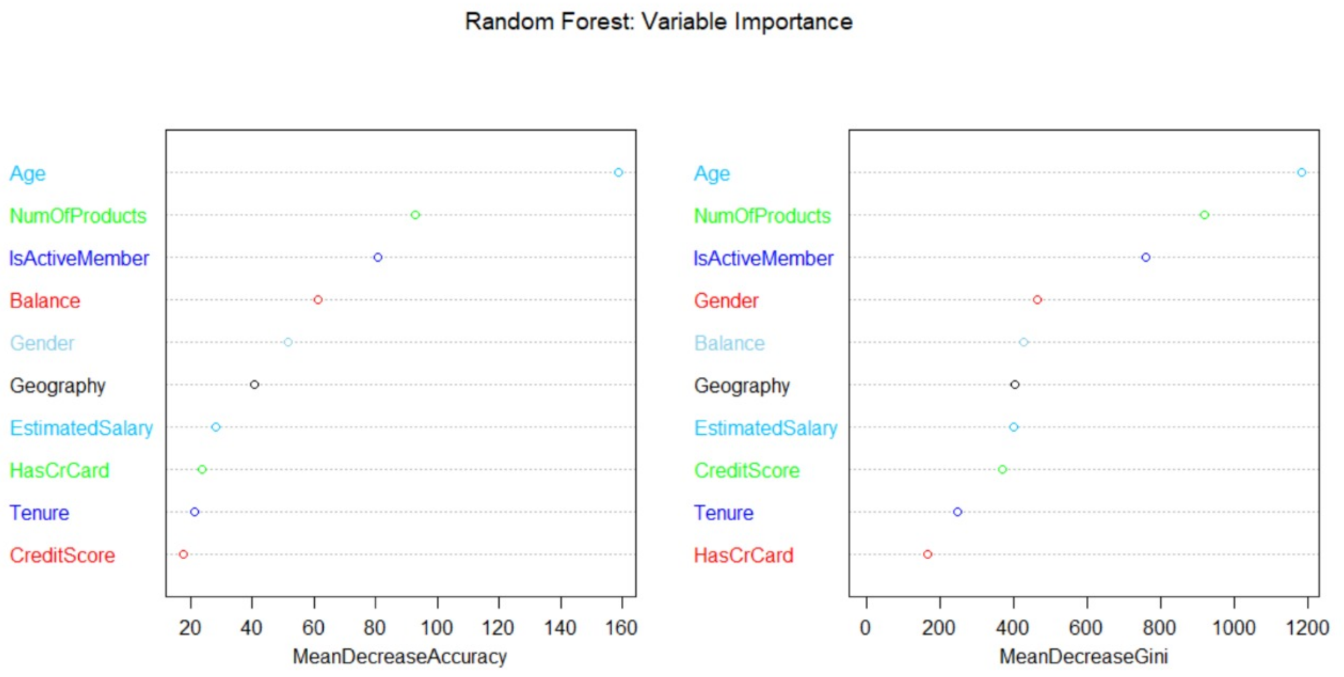


Fig. 13 : Mean decrease accuracy and mean decrease Gini of the different covariates for Random Forest.

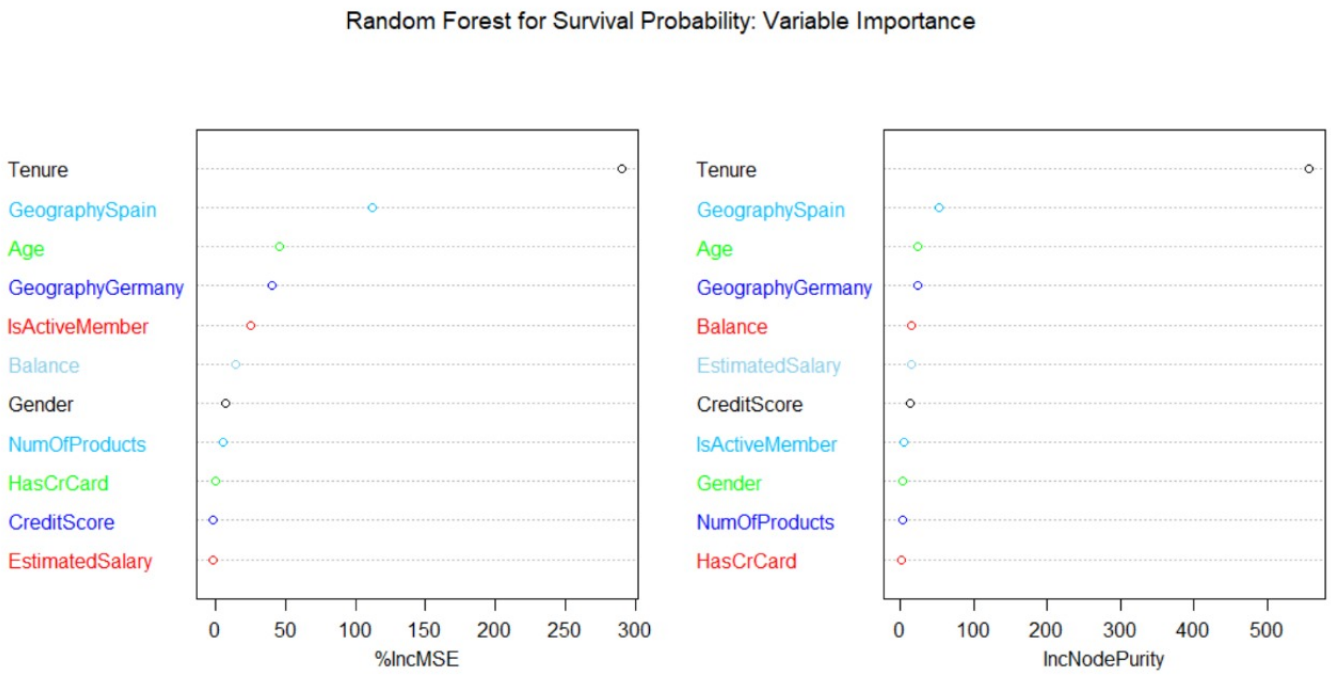


Fig. 14 : MSE increase and Node Purity increase of the different covariates for Random Forest Regression.

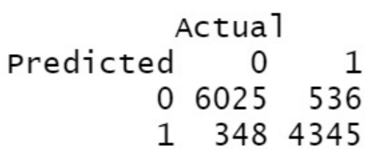
6 Model Diagnostics and Comparisons

6.1 Confusion Matrix

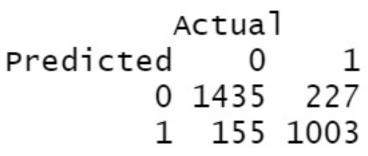
Confusion matrix is a technique for summarizing the performance of the model and as the name suggests it gives us a matrix as a output. It gives information about errors made by the classifier and the types of errors that are being made. It reflects how a classification model is disorganized and confused while making predictions.
Confusion matrix is a very popular measure used while solving classification problems. It visualizes and summarizes the performance of a classification algorithm. In general classification accuracy fails on classification problems with a skewed class distribution because of the intuitions developed by practitioners on data sets with an equal class distribution. The 4 important terms are mentioned below.

- **True Positives(TP)**: In this case the model correctly predicts the positive class.
- **True Negatives(TN)**: In this case the model correctly predicts the negative class.
- **False Positives(FP)**: In this case the model incorrectly predicts the positive class.
- **False Negatives(FN)**: In this case the model incorrectly predicts the negative class.

The Confusion matrix of the unpruned classification tree, for training data and test data respectively, looks like:

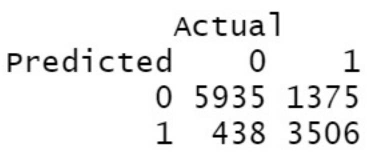


```
         Actual
Predicted    0    1
        0 6025  536
        1  348 4345
```

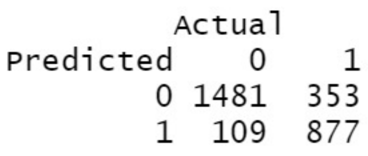


```
         Actual
Predicted    0    1
        0 1435  227
        1  155 1003
```

The same for the pruned classification tree is (left: train data, right: test data):

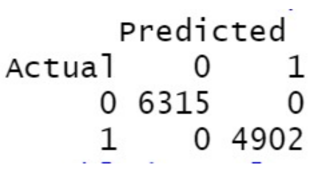


```
         Actual
Predicted    0    1
        0 5935 1375
        1  438 3506
```

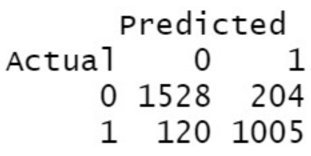


```
         Actual
Predicted    0    1
        0 1481  353
        1  109  877
```

For Bagging (left: train data, right: test data):

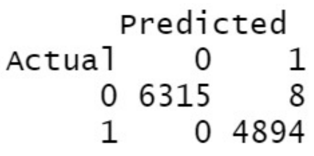


```
      Predicted
Actual    0    1
     0 6315    0
     1    0 4902
```

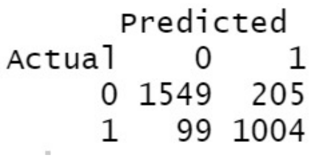


```
      Predicted
Actual    0    1
     0 1528  204
     1  120 1005
```

For Random Forest (left: train data, right: test data):



```
      Predicted
Actual    0    1
     0 6315    8
     1    0 4894
```



```
      Predicted
Actual    0    1
     0 1549  205
     1   99 1004
```

From the confusion matrices, we see that the classification tree performs better after pruning and the random forest model performs better than the bagged model. Of these three models, random forest performs the best in correctly predicting whether a customer is churned or not.

6.2 Metrics of Confusion Matrix

6.2.1 Accuracy

Accuracy is how close or far off a given set of measurements are to their true value.So, as a rule of thumb accuracy can tell us immediately whether a model is being trained correctly and how it may perform generally. Generally accuracy is most suited when we just care about single individuals instead of multiple classes, in case of multiclass problem it computes subset accuracy.

The main problem behind using accuracy as the main performance metric is that it does not perform well in presence of severe class imbalance.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

6.2.2 Precision

Precision indicates that how many of the actual positives our model is able to capture labelling it as positive. It is a fraction of true positive elements divided by total no. of positively predicted elements. Precision is a good measure to determine if the cost of False Positive id high.

$$\text{Precision}=\frac{TP}{TP+FP}$$

6.2.3 Recall

The Recall measures the model's predictive accuracy for the positive class. It is computed as a fraction of true positive classes divided by total no. of positively classified classes. This is also termed as sensitivity.

$$\text{Recall}=\frac{TP}{TP+FN}$$

6.2.4 True Negative Rate(Specificity)

It is just an opposite measure of recall score. It is a fraction of true negative elements divided by total no. of negatively predicted elements.

$$\text{True Negative Rate}=\frac{TN}{TN+FP}$$

6.2.5 F1 Score

F1 score is a measure of robustness and preciousness of the model. It is needed when we want to seek a balance between precision and recall. It is the harmonic mean of precision and recall. It is generally used if false negatives and false positives play a crucial role in the model. This score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall, thus F1 score is the appropriate measure in those situations. It takes maximum score of 1 and minimum score of 0.

$$\text{F1 Score} = \frac{2(Precision X Recall)}{Precision+Recall}$$

$$= \frac{2TruePositive}{2TruePositive+FalsePositive+FalseNegative}$$

6.3 ROC Curve

A ROC (Receiver Operating Characteristic) curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system at its different classification thresholds. This curve is produced by plotting sensitivity as the y coordinate versus false positive rate (FPR) as the x coordinate for a single classifier at different thresholds.

6.3.1 AUC-ROC Score

The Receiver Operator Characteristic (ROC) curve is an evaluation metric for binary classification problems. It is a probability curve that plots the TPR against FPR at various threshold values. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

Higher the AUC, better is the performance of the model at distinguishing between the positive and negative classes.

When AUC = 1, then the classifier is able to perfectly distinguish between all the Positive and the Negative class points correctly. If, however, the AUC had been 0, then the classifier would be predicting all Negatives as Positives, and all Positives as Negatives.

When $0.5 < \text{AUC} < 1$, there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values. This is so because the classifier is able to detect more numbers of True positives and True negatives than False negatives and False positives.

When AUC = 0.5, then the classifier is not able to distinguish between Positive and Negative class points, meaning either the classifier is predicting random class or constant class for all the data points.

So, higher the AUC value for a classifier, better is its ability to distinguish between positive and negative classes.

Model	**Accuracy**	**Sensitivity**	**Specificity**	**F1 Score**	**AUC**	**Gini Index**
Pruned Classification Tree	84.35%	0.831	0.866	0.871	0.848	0.696
Bagging	88.66%	0.882	0.893	0.904	0.888	0.776
Random Forest	89.36%	0.883	0.91	0.911	0.897	0.793

Table 4 : Accuracy, Sensitivity, Specificity, F1 Score, AUC, Gini Index for different models.

Analysing these measures, we get random forest as our final model for classifying the customers with respect to the *Exited* class or *Not Exited* class, as the accuracy, sensitivity, specificity, F1 score, AUC is the highest and Gini index is the lowest for that model.

6.4 Out-of-Bag Error

It turns out that there is a very straightforward way to estimate the test error of a bagged model, without the need to perform cross-validation or the validation set approach. Recall that the key to bagging is that trees are repeatedly fit to bootstrapped subsets of the observations. One can show that on average, each bagged tree makes use of around two-thirds of the observations. The remaining one-third of the observations not used to fit a given bagged tree are referred to as the out-of-bag (OOB) observations. We can predict the response for the i^{th} observation using each of the trees in which that observation was OOB. This will yield around $B/3$ predictions for the i^{th} observation, B being the number of bootstrapped training samples.

In order to obtain a single prediction for the i_{th} observation, we can average these predicted responses (if regression is the goal) or can take a majority vote (if classification is the goal). This leads to a single OOB prediction for the i^{th} observation. An OOB prediction can be obtained in this way for each of the n observations, from which the overall OOB MSE (for a regression problem) or classification error (for a classification problem) can be computed. The resulting OOB error is a valid estimate of the test error for the bagged model, since the response for each observation is predicted using only the trees that were not fit using that observation.

The OOB estimate of error rate for random forest classification and bagging classification are 10.17% and 10.4%, respectively. So in terms of having minimum OOB error, random forest outperforms bagging.

6.5 Brier Score

The Brier score is used for evaluating time-to-event predictions in the form of survival estimates. We will then introduce the topic of right-censoring in survival analysis, and show how the IPCW (Inverse Probability of Censoring Weighting) Brier score accounts for censored observations.

Let T_i denote the event time of individual i and let $f_i(t)$ denote the density function of this event time. The survival function of individual i is then given by

$$S_i(t) = P(T_i > t) = \int_t^{\infty} f_i(u)\,du$$

In time-to-event prediction, we want to estimate $S_i(t)$ for all individuals, and we will let $\pi_i(t)$ denote these estimates. To simplify the presentation, we will disregard the estimation uncertainty, and consider the $\pi_i(t)$'s as known (non-random) functions.

A reasonable metric for evaluating the predictive performance of the $\pi_i(t)$'s, is to calculate the mean squared error of the estimates to the true survival functions. For a data set with n individuals, this score is

$$MSE(t) = \frac{1}{n}\sum_{i=1}^{n}[S_i(t) - \pi_i(t)]^2.$$

However, $S_i(t)$ is not known outside of simulations, and we instead observe event times T_i drawn from the event time distribution. The Brier score for uncensored data approximates the true survival functions with step-functions with jumps at the event times, giving

$$BS(t) = \frac{1}{n}\sum_{i=1}^{n}[\mathbb{1}\{T_i > t\} - \pi_i(t)]^2$$

$$BS(t) = \frac{1}{n}\sum_{i=1}^{n}[\pi_i(t)^2\mathbb{1}\{T_i \le t\} + [1 - \pi_i(t)]^2\mathbb{1}\{T_i > t\}]$$

Now we consider the situation of possibly censored time to event data. For each patient we observe $T_i^* = min(C_i, T_i)$ and $\delta_i = \mathbb{I}(T_i \le C_i)$, where T_i represents the time to the event of interest and C_i the (hypothetical) time under observation ($i = 1, ..., n$). We assume that T and the covariate X are independent of C - this is a common assumption of random censorship - and that C is distributed according to $G(t) = P(C > T)$. For notational convenience we assume no ties between censored and uncensored observations.

For a fixed time point t, the contributions to the Brier score can be split up into three categories:
Category 1: $T_i^* \leq t$ and $\delta_i = 1$;
Category 2: $T_i^* > t$ and ($\delta_i = 1$ or $\delta_i = 0$);
Category 3: $T_i^* \leq t$ and $\delta_i = 0$;

For the uncensored observations of category 1 the event occurred before t, and the event status at t is equal to $\mathbb{I}(T_i^* > t) = 0$; thus the contribution to the Brier score is $(0 - \pi(t|X_i))^2$. In category 2 the observed event status at t is equal to 1 since all of these patients are known to be event-free at t; the resulting contribution to the Brier score is $(1 - \pi(t|X_i))^2$. For the censored observations of category 3 the censoring occurred before t so that the event status at t is unknown; thus their contribution to the Brier score cannot be calculated.

To compensate for the loss of information due to censoring, the individual contributions have to be reweighted: observations in category 1 get the weight $\frac{1}{\hat{G}(T_i^*)}$, those of category 2 get the weight $\frac{1}{\hat{G}(t)}$ and observations of category 3 get weight zero; here $\hat{G}(t)$ denotes the Kaplan- Meier estimate of the censoring distribution G, i.e. the Kaplan-Meier estimate based on $(T_i^*, 1 - \delta_i), i = 1, ..., n$. Divided by n, these weights sum up to 1, and the empirical Brier score under random censorship can be defined as

$$BS(t) = \frac{1}{n}\{(\pi(t|X_i))^2\mathbb{I}(T_i^* \leq t, \delta_i = 1)(1/\hat{G}(T_i^*)) + (1 - \pi(t|X_i))^2\mathbb{I}(T_i^* > t)(1/\hat{G}(t)).$$

The Brier Score calculated on the training dataset by the function *Brier()* from *measures* for different years for the cox model is as follows:

Time(years)	**Brier Score**
1	0.367
2	0.788
3	0.766
4	0.748
5	0.728
6	0.706
7	0.681
8	0.655
9	0.626
10	0.586

Table 5 : Brier Score for training dataset.

And for the test data as well

Time(years)	Brier Score
1	0.372
2	0.784
3	0.754
4	0.728
5	0.701
6	0.671
7	0.639
8	0.607
9	0.571
10	0.525

Table 6 : Brier Score for testing dataset.

The Integrated Brier Score is then given by

$$IBS = \frac{1}{max(T_i^*)} \int_0^{max(T_i^*)} BS(t),,$$

and lower values indicate better predictive performances. Using *pec* and *prodlim* from R packages, the Integrated Brier Score is calculated for the Cox model. The IBS is **0.108**, which is used here as the measure of accuracy of the cox model in predicting the survival probability. We also visualize the prediction error of the cox model at different time points:

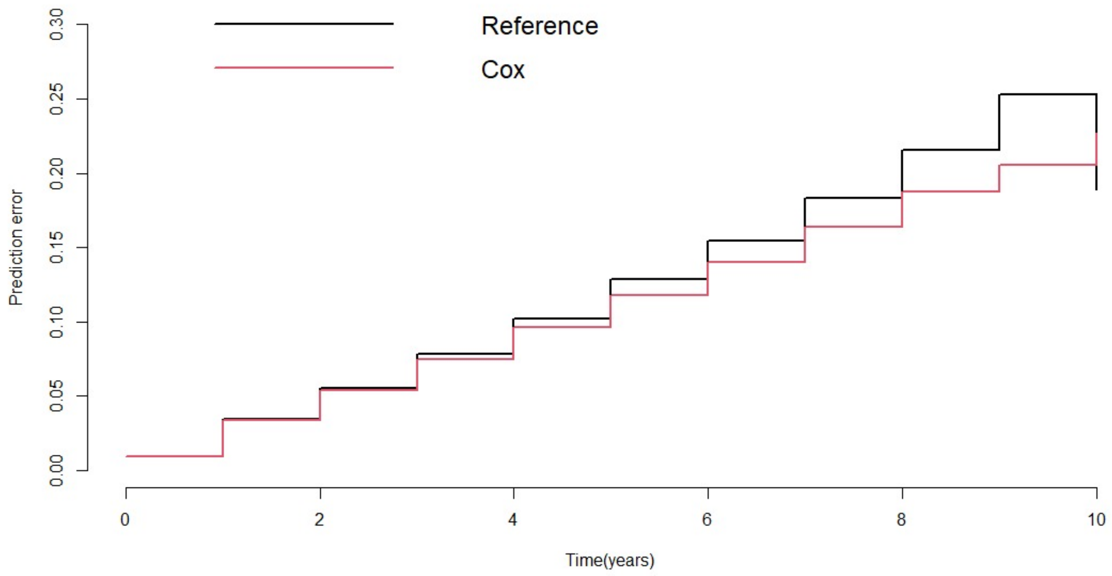


Fig. 15 : The prediction error rate of the Cox model (red line).

7 Results and Conclusion

Our study consists of 2 sections, classification and regression. We classify the bank customers into 2 classes – '1' as churned and '0' as retained, using 3 models – Classification tree, Bagging and Random Forest and estimate the survival probability of the bank customers using Cox model, CART and Random Forest, and compare the models in each of the sections.

For classification, we oversample the minority class (class '1') due to presence of data imbalance and then fit the models on the new data. From the measures of accuracy (Table 4), we see that all the 3 models have accuracy more than 84%, sensitivity more than 83%, specificity more than 86% and AUC more than 0.84 on the testing data, depicting that these are able to predict the positive (churned) and negative (not churned) classes correctly with a high accuracy. Of these 3 models, random forest outperforms classification tree and bagging, with 89.36% accuracy and AUC = 0.897.

For estimation of survival probability, we first fit Cox proportional hazards regression model on the censored data (original data) to obtain the survival probabilities of the customers at different times points (1, . . . , 10 years), i.e., we obtain 10 probabilities for each customer. The survival probability of a customer is his/her survival probability corresponding to the number of years he/she is with the bank. The Cox model yields an integrated brier score (a measure of mean squared error) of 0.108. Now, a regression tree and a random forest is fitted to predict the survival probability of the customers. The random forest, with RSS = 0.0022, achieves slightly better accuracy than the regression tree with RSS = 0.0039, both having nearly same RMSE (= 0.02) on the testing data. Moreover, the random forest is able to explain 97.5% of variability in the survival probability with respect to the factors like age, geography, number of products, tenure, gender, credit score, balance, estimated salary, has credit card or not, is an active member or not.

The variable importance figures depict that age and number of products of the bank, like credit and debit cards, loans etc., a customer is using, are 2 vital factors in predicting whether the individual will continue to be a customer. Hence, the bank should take care of these factors in order to retain its customers as far as possible.

8 References

1. https://www.kaggle.com/datasets/mathchi/churn-for-bank-customers
2. An Introduction to Statistical Learning, with Applications in R, by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani.
3. Imad Bou-Hamad, Denis Larocque and Hatem Ben-Ameur, A Review of Survival Tree
4. Graf, E., Schmoor, C., Sauerbrei, W. and Schumacher, M. (1999). Assessment and Comparisons of Prognostic Classification Schemes for Survival Data. Statistics in Medicine.
5. https://www2.karlin.mff.cuni.cz/ vavraj/cda/exercise07.html
6. https://towardsdatascience.com/customer-churn-analysis-4f77cc70b3bd