Credit card: churn prediction

Mattia Birti 897092, Paola Maria Cavana 859341, Davide Prati 845926, Yulia Tsymbal 894213

Abstract
One of the most common problems for a business manager of a consumer credit card bank is having to deal with the phenomenon of customer attrition, i.e. the tendency of clients to leave the bank. Managers want to analyze the data to find out the reasons behind this phenomenon and leverage the same to predict customers who are likely to drop off.
The chosen dataset was made available on the Kaggle platform and it contains the details of more than 10 000 customers of a consumer credit card bank. (1)

Keywords
Machine Learning - Classification - Credit card - Churners

Contents

Introduction

Which are the main reasons why a person decides to leave his bank?
Being able to answer this seemingly simple question could really change the destiny of a consumer credit card bank, as in the case of our dataset.
Of course we will never know exactly which are the specific reasons for each single client, because we should consider personal and emotional factors, but we can analyze objective data to try to understand which main characteristic of the client can suggest that he will decide to leave the bank.
The goal of our work is to make this analysis, and then, after having understood that and having obtained those characteristics, exploit our considerations to develop models capable of predicting which are the customers who are at greater risk of churning.
Then, making use of some theoretical results to compare those models, we will evaluate which of them is the best one to apply to our dataset.
That is fundamental, because finding the most accurate method (or, in other words, the one with the lowest margin of error) can be crucial because it can lead to make more precise decisions and to develop more effective strategies.

Description of the dataset

The chosen dataset was obtained from the Kaggle platform and it originally consists of 10 000 customers who mention their age, salary, marital status, credit card limit, credit card category, and more, for a total of 23 features, even if the last two will have no utility at all.
Here we report all the 23 attributes:

- CLIENTNUM (numeric – ratio):
  The client number. Unique identifier for the customer
- Attrition_Flag (categorical – binary):
  Customer activity (existing or attrited): it's equal to 1 if the account is closed and to 0 if it's open
- Customer_Age (numeric - ratio):
  Age of the customer in years
- Gender (categorical – binary):
  Gender of the customer: M stands for male, F for female
- Dependent_count (numeric - ratio):
  Number of dependents
- Education_Level (categorical – ordinal):
  Educational qualification of the account holder (example: high school, college graduate, etc.)

- Marital_Status (categorical – ordinal):
  Married, Single, Divorced, Unknown

- Income_Category (categorical – ordinal):
  Annual Income Category of the account holder ($<$ \$40K; \$40K - \$60K; \$60K - \$80K; \$80K - \$120K; $>$ \$120K)

- Card_Category (categorical – ordinal):
  Type of Card (Blue, Silver, Gold, Platinum)

- Months_on_book (numeric - ratio):
  Period of relationship with bank in months

- Total_Relationship_Count (numeric - ratio):
  Total number of products held by the customer

- Months_Inactive_12_mon (numeric - ratio):
  Number of months inactive in the last 12 months

- Contacts_Count_12_mon (numeric - ratio):
  Number of contacts in the last 12 months

- Credit_Limit (numeric - ratio):
  Credit limit on the Credit Card, in american dollars

- Total_Revolving_Bal (numeric - ratio):
  Total revolving balance on the Credit Card

- Avg_Open_To_Buy (numeric - ratio):
  Open to buy credit line (average of last 12 months)

- Total_Amt_Chng_Q4_Q1 (numeric - ratio):
  Change in transaction amount (Q4 over Q1)

- Total_Trans_Amt (numeric - ratio):
  Total transaction amount (last 12 months)

- Total_Trans_Ct (numeric - ratio):
  Total transaction count (last 12 months)

- Total_Ct_Chng_Q4_Q1 (numeric - ratio):
  Change in transaction count (Q4 over Q1)

- Avg_Utilization_Ratio (numeric - ratio):
  Average card utilization ratio

- Naive_Bayes_Classifier_1

- Naive_Bayes_Classifier_2

Structure of the report

This report is organized as follows:

1. **Data exploration**: we examine features from the dataset.

2. **Preprocessing**: we remove some columns from the original dataset, transform some features and handle missing value in order to make the dataset more suitable for analysis.

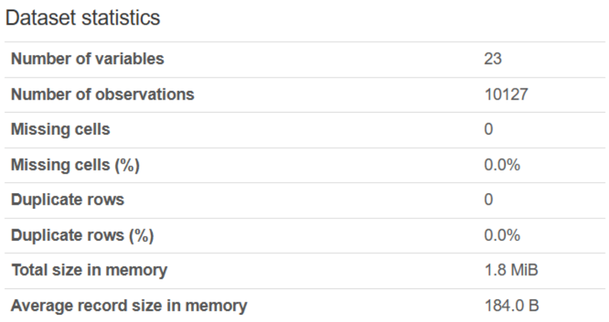
3. **Models**: we describe the different models used to predict the value of the attribute Attrition_Flag.

4. **Performance measures**: we introduce the criteria we will exploit to evaluate our models.

5. **Evaluation**: we evaluate and compare the models described two sections ago exploiting the measures presented in the previous section.

1. Data exploration

Before doing anything else, we start our work by having a first glance at the original dataset as provided by Kaggle, without any change.

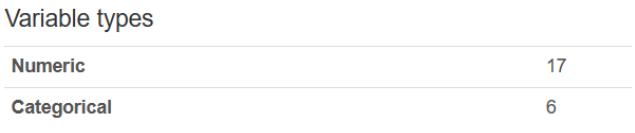


Dataset statistics

Number of variables	23
Number of observations	10127
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	1.8 MiB
Average record size in memory	184.0 B

Figure 1. Overview of the dataset

Using Pandas Profiling Report (2), we developed a brief overview of the dataset: as we can see in the figure above, to be precise our dataset is constituted by 10 127 observations and 23 variables.
An absolutely relevant consideration is that luckily we don't have any missing values or any duplicate rows, so we don't have the issue to deal with them during the phase of preprocessing.



Variable types

Numeric	17
Categorical	6

Figure 2. Numeric and categorical attributes

Regarding variable types, as we can see the dataset has 17 numerical ones and 6 categorical ones.

Then, changing our perspective and focusing ourselves just on the attribute that gives us information about churners, that is the one called Attrition_Flag, we noticed that it just reports two string values, which are 'existing customer' or 'attrited customer', when the customer is in fact existing or attrited, so in other words when the account is respectively open or closed.

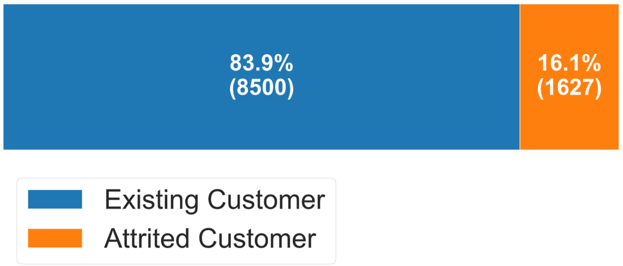


Figure 3. Analysis of the values assumed by Attrition_Flag

In particular, our exploration of the attribute Attrition_Flag shows that a solid 84% of the clients about which we have information is classified as existing, while just 16% of them is attrited. So we can conclude we are facing a situation with an imbalance regarding the class attribute, which we have to take in consideration throughout our analysis.

2. Preprocessing

After completing data exploration, first of all we noticed the last two columns, which were calculated by someone and which are not directly derived from the data source, and that also do not make any special sense for our task, so we decided to delete them, as suggested by the creator of the dataset himself.
Then we should have continued by checking each attribute for missing values. But as we already said in the previous section there are no missing values in our dataset, so we can proceed immediately to prepare and eventually modify individual attributes for further modeling.
Subsequently, we noticed that attributes such as Gender, Education Level, Marital Status, Income Category, Card Category are presented in the dataset as string values. This type of data is not suitable for being used in some of our selected models, so we decided to re-code each level of attribute values into numerical values. To do this, the inbuilt LabelEncoder module was used (3). Of course ordinal values as the ones of the Education_Level or Income_Category attributes were initially sorted and just after then replaced by the corresponding numerical value.
The target column was also transformed into ‘y’ and ‘n’ labels, according to whether the person is still a client of the bank or has refused their services, and accordingly to that we decided to change its name from Attrition_Flag to a clearer Current_Customer, a name that fits with the choice of values ’y’ and ’n’. Then we created also a different attribute, that we called Current_Customer_Num, that is exactly the same as Current_Customer, but with the numerical values 1 and 0 instead of the strings ’y’ and ’n’.

2.1 Feature selection

In order to understand which features have the greatest influence on our target, we used the approach of selecting these features by correlation (4, 5). Only features with the highest correlation with the target were included in further modeling. It was also checked whether there are any of the selected features that are dependent on each other. Having identified certain dependencies, we proceeded by excluding those that had a lower correlation with the target.

Here we report the correlation matrix of our attributes, that is a symmetric matrix in which we have all the correlation coefficients between our attributes, each of which belongs to the interval $[-1, 1]$. The closer this coefficient as its absolute value is to 1 more the correlation between the two corresponding features is marked.

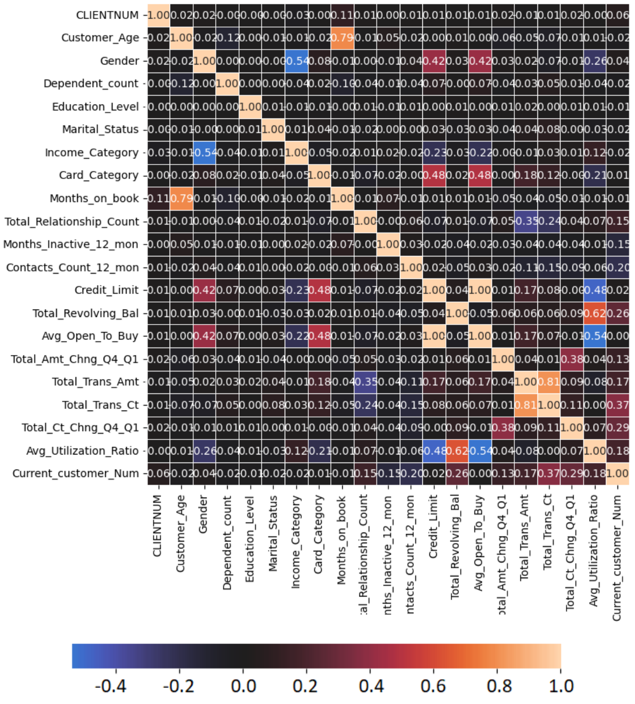


Figure 4. Correlation Matrix

Thus, looking at the correlation matrix and focusing ourselves just on the last row (or column), we selected the features that we would exploit through our analysis.
We chose the best 8 attributes based on the correlation coefficients, so we had, sorted by their coefficients in descending order:

- Total_Trans_Ct
- Total_Ct_Chng_Q4_Q1
- Total_Revolving_Bal
- Contacts_Count_12_mon
- Avg_Utilization_Ratio
- Total_Trans_Amt
- Total_Relationship_Count

- Months_Inactive_12_mon.

Then we noticed how Total_Trans_Ct and Total_Trans_Amt have a huge correlation coefficient between them (0,81) so, as we said before, we exclude the one with the lowest coefficient with the Current_Customer_Num attribute, that results to be Total_Trans_Amt (which coefficient is 0,17 against 0,37 of Total_Trans_Ct).
So we proceed our further modeling with these 7 features, obtained from the initial 23.

2.2 Dataset imbalance problem

Solving the problem of imbalance is critically important, because the model on the imbalanced dataset does not have a sufficient number of instances of the class that is less represented in the target and therefore cannot perform correct predictions. An undersampling approach was used to solve this problem (6). As a result, we got equally balanced training and test samples (2 600 and 654 records, respectively), where the exact half of both sets consists of existing customers, while the other one of attrited customers.
Anyway to make a comparison we will use also an approach which does not involve the undersampling, and so does not deal with the imbalance problem.

The code that was used for these transformations and in general for all the preprocessing section can be found in more detail by the link (7) on the references.

3. Models

We decided to use different Machine Learning models to predict the value of the attribute Attrition_Flag, which provides the information about the churning status of the corresponding customer.
Here we report the list of all the models we used:

- As heuristic models we chose decision trees and random forests. For both of them we considered both the standard version implemented by Knime and the version provided by the Weka software, which is respectively called J48 and simply RandomForest.

- Logistic Regression (LR) to represent the regression based methods. In this case we employed two instances of the binomial Logistic Regression classifier: the basic one and the simple logistic node, which uses the boosting approach. Both of them are implemented by Weka.

- SVM (Support Vector Machines), which belongs to the class of separation models. In this case we used three classifiers which are all offered by the Weka environment: they are two SMOs, one with the puk and the other with the polynomial kernel, and a Spegasos, a different instance of SVM, which in our case exploits the logloss function.

- Finally, we considered Naive Bayes as representative of the last kind of methods, the probabilistic ones. Again we use both the standard node and the one provided by the Weka environment.

All the methods implemented by standard Knime software are predicted with their specific predictor node implemented again by Knime itself, while the Weka versions of the methods simply use the Weka Predictor node, that is provided by Weka too.

Each of these five algorithms was trained having in input the 7 explanatory attributes obtained by feature selection, and using three different initial approaches to partition the dataset into Training and Test Set: Holdout, undersampling (already seen in point 2.2) and undersampling with Cross Validation.

3.1 Holdout

The first approach we used is the one called Holdout, in which we have partitioned the preprocessed dataset without the undersampling in two different parts with a stratified sampling. The first one, called Training Set, consists of 67% (2/3) of the records, while the second one, the Test Set, consists just of 33% (1/3). The models are all trained exploiting just the first part as input data, to then be evaluated with the Test Set, which data are considered as unseen.

3.2 Undersampling

As already said in section 2.2, we chose to use this approach to work around the class imbalance problem. It significantly reduces our records (we will work with less than a third of the originals) but we will have datasets - Training and Test Sets - perfectly balanced on the class attribute. We will use Training and Test Set, obtained directly applying the undersampling technique (7), with the exact same roles of the previous case.

3.3 Undersampling with Cross Validation

We used a 10-fold Cross Validation technique to split the whole dataset obtained with the undersampling (the union of the Training and Test Sets which we already have) into 10 different datasets. Then we trained all the models ten times, each of them with 9 out of 10 datasets to have the Training Set role while the remaining one plays the role of the Test Set.

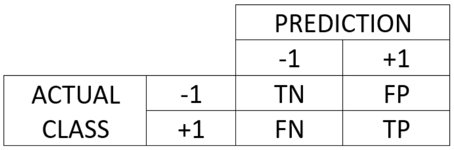
4. Performance measures

Throughout our analysis, we used a considerable number of criteria to evaluate the performances of our models. The first criterion we considered is the most standard one, i.e. the Accuracy, which points out the amount of correct predictions (both positive and negative) weighted by the total number of predictions. Formally

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

where TP and TN stand for the instances correctly classified, respectively as belonging to positive and negative class, while

FP and FN indicate the amount of instances wrongly classified as positive and negative, as shown in the figure below.



		PREDICTION	
		-1	+1
ACTUAL CLASS	-1	TN	FP
	+1	FN	TP

Figure 5. Confusion Matrix

Thus, the higher the accuracy, the more effective the method will be.
But in a case like ours, in which we have an imbalance between the frequencies of the two possible values of the class attribute - as we already said during data exploration, the most common value has a frequency of 84%, while the rarest just of 16% - accuracy is not the most consistent measure.
So, to properly address this problem we have to consider a more effective quantity, the F_1 measure.
Before we can present it, we have to introduce other two measures, called recall and precision:

$$\text{recall} = \mathrm{r} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}},$$
$$\text{precision} = \mathrm{p} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}.$$

Recall represents the amount of positive records which have been correctly classified, while precision represents the number of records that effectively belong to the positive class among all the records predicted as such. It is often possible to optimize one of them but not both. So we present the F_1 measure, a quantity that aims to merge both precision and recall computing their harmonic mean:

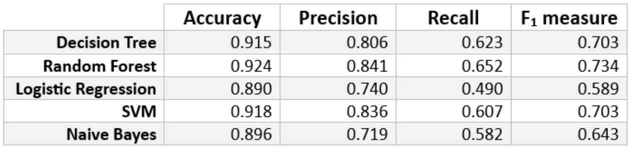
$$F_1 = \frac{2 \cdot \mathrm{r} \cdot \mathrm{p}}{\mathrm{r} + \mathrm{p}},$$

where *r* stands for recall and *p* for precision.

5. Evaluation

5.1 Holdout

Here we report the values of the performance measures obtained by implementing the first approach, the Holdout one:



	Accuracy	Precision	Recall	F_1 measure
Decision Tree	0.915	0.806	0.623	0.703
Random Forest	0.924	0.841	0.652	0.734
Logistic Regression	0.890	0.740	0.490	0.589
SVM	0.918	0.836	0.607	0.703
Naive Bayes	0.896	0.719	0.582	0.643

Figure 6. Performance measures with Holdout

Please notice that in this table we have reported just the best version of each model, with a choice based on its value of accuracy, and this selected version will also be the only one that we will take in consideration also when we will work using the other two approaches.
These surviving versions are the Knime one regarding Random Forest model, while for Random Tree and Naive Bayes we chose the Weka one, for SVM the Weka version with the puk kernel and for Logistic Regression the simple logistic version, again provided by Weka.

Looking at the table we can observe how for each of the selected models the Holdout strategy gives us back a relatively high accuracy, while the other measures, precision, recall and F_1, are not that good.
This result is consistent and predictable, taking into account the imbalanced nature of the dataset: accuracy, which makes no difference between positive and negative values, is favored by this situation, but the other classifiers are negatively conditioned.

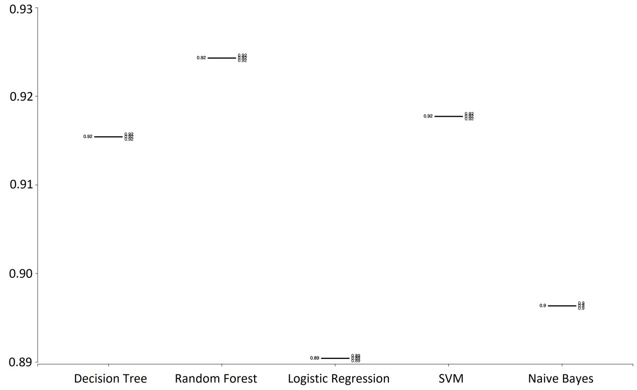


Figure 7. Accuracy for each model with Holdout

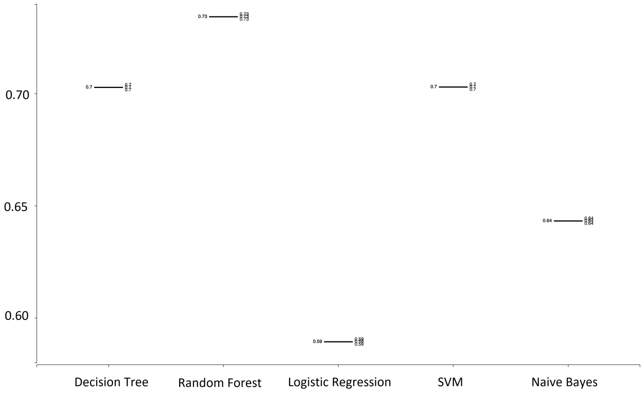


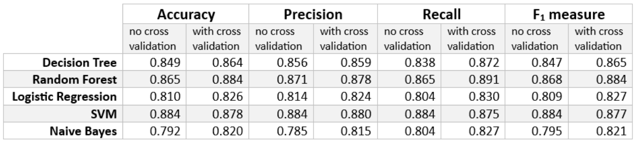
Figure 8. F_1 measure for each model with Holdout

So we can conclude that if we apply an Holdout strategy, and thus if we ignore the imbalance problem, these models result to be not able to understand the phenomenon, and therefore to give us back acceptable data.

Thus we proceed with the other two approaches already presented, which unlike Holdout take care of the class imbalance problem and therefore they should improve the results regarding the measures which are affected by that imbalance.

5.2 Undersampling with and without Cross Validation

In the table below we report the new values of the performance measures, that this time are computed on the preprocessed and undersampled dataset, both with no additional strategies and with a 10 fold Cross Validation.



	Accuracy		Precision		Recall		F_1 measure	
	no cross validation	with cross validation	no cross validation	with cross validation	no cross validation	with cross validation	no cross validation	with cross validation
Decision Tree	0.849	0.864	0.856	0.859	0.838	0.872	0.847	0.865
Random Forest	0.865	0.884	0.871	0.878	0.865	0.891	0.868	0.884
Logistic Regression	0.810	0.826	0.814	0.824	0.804	0.830	0.809	0.827
SVM	0.884	0.878	0.884	0.880	0.884	0.875	0.884	0.877
Naive Bayes	0.792	0.820	0.785	0.815	0.804	0.827	0.795	0.821

Figure 9. Performance measures on the undersampled dataset

Looking at the table above and developing an overview in which we compare these results also with the ones reported by the previous table, we can point out that models have a significant worsening regarding accuracy, but that is consistent with what we expected, because accuracy is not the measure that we thought to improve by exploiting the undersampling approach.

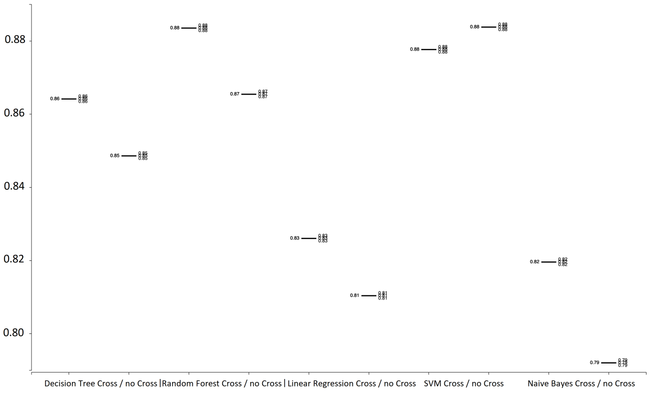


Figure 10. Accuracy for each model with undersampling and respectively with and without 10 fold Cross Validation

Our aim was instead about precision and recall, and consequentially of course on F_1 measure, that are the ones which signal the improvement of models' performances when you get a balance on the class attribute.

And indeed, focusing on those, we can easily notice their improvement compared to the case of the Holdout approach; and in particular we can also point out that all the F_1 measures turn to assume higher values when we choose to apply Cross Validation instead of not doing it, except regarding the case of SVM, that is also the only one for which the accuracy is not improved either.

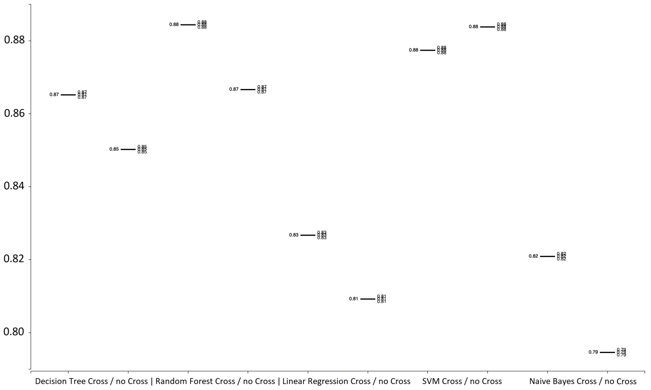


Figure 11. F_1 measure for each model with undersampling and respectively with and without 10 fold Cross Validation

Therefore, being the Cross Validation applied to the undersampled dataset the best initial approach, we decided to develop a further analysis computing the ROC Curve (8) for the models in that case, but with the exception of SVM, that, as we have just said, is the only method that gets worse when applying Cross Validation.

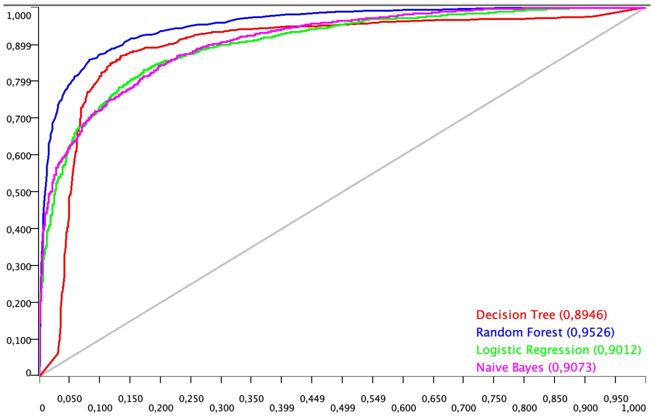


Figure 12. ROC Curve for the models with undersampled dataset and cross validation. On the right are reported the values of AUC (Area Under Curve).
On X axis we have False Positive Rate and on Y axis True Positive Rate.

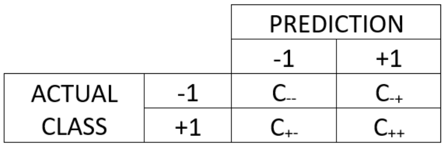
Looking at the figure we can observe that all the ROC curves of our methods are very higher than the straight line that represents the model "Zero Rule", i.e. the one which does no add any further information; and we can also notice how the values of AUC (a measure obtained from the calculation of the area under the curve) are close to 1, their best possible value. From here we therefore have a further confirmation of the goodness of our models and our approach.

Conclusion and future developments

One of the questions we aimed to answer at the beginning of our work concerned what were the main reasons that led the client to decide to quit the bank services. After feature selection we can say, looking at the surviving attributes, that these reasons are purely economic in nature, while demographic factors are not as relevant as one might think.

About the choice regarding the best model to use, after a comparison on our performance measures we can conclude that with an Holdout approach the most preferable method is Random Forest for all of the four parameters, as whether we are working with just undersampling or with undersampled Cross Validation the best model is always SVM, while Random Forest drops in second position. Talking about the worst ones, they are Naive Bayes and Logistic Regression.
Focusing instead at the ROC Curve, that we have implemented just in the third case (the one with undersampled 10 fold Cross Validation), we can say that taking into account the AUC the best model results to be Random Forest again, that is also the best for each value of the False Positive Rate, with the exception of some extreme and irrelevant cases at the ends of the $[0,1]$ range.
If instead we ask ourselves which is the most accurate approach to create Training and Test Set, it results to be the Cross Validation applied to the undersampled dataset, with the only exception of SVM, for which however the best one is still performed with undersampling. So, despite the imbalance is not that high (the ratio between common and rare class is just 5,25), we can conclude that it was worth it to consider the issue, although this has resulted in the request of a more complicated strategy.

Regarding future developments, we can surely take in consideration the idea of developing similar analysis to other dasatets, perhaps with a larger amount of data and a greater variety of attributes, even thought our dataset was certainly a good one, complete and without any missing data.
Moreover, in addition to the undersampling that has already been done, we could also develop its complementary strategy, called oversampling (6).
Furthermore, it would certainly be interesting to consider also different analysis, which may have the goal of calculating the cost of the classification we make, consistently with the schema below.



		PREDICTION	
		-1	+1
ACTUAL CLASS	-1	C_{--}	C_{-+}
	+1	C_{+-}	C_{++}

Figure 13. Cost Matrix

$$\text{Cost} = C_{--} \cdot \text{TN} + C_{-+} \cdot \text{FP} + C_{+-} \cdot \text{FN} + C_{++} \cdot \text{TP}.$$

This is an analysis that can surely have its relevance, because there can be a scenario in which there is no symmetry in the cost matrix, and that happens when in the real life problem we are referring to we have a situation in which predicting wrongly a positive data is better or worse than predicting wrongly the negative one, that is quite likely in our case.
Thus we can have a final result for which the model with the best performance measures isn't the same of the one with the lowest cost.
Unfortunately we couldn't develop that in our work because we are not aware of the actual cost coefficients for our credit card bank regarding the prediction of churners, so we have to trust our performance measures, which do not take into account these economic factors.

References

(1) The original dataset:
https://www.kaggle.com/datasets/anwarsan/credit-card-bank-churn

(2) Pandas profiling report:
https://github.com/JuliaTsymbal/MachineLearningProject/blob/main/ChurnProfiling.html

(3) Label Encoder:
https://scikitlearn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

(4) The importance of feature correlation:
https://towardsdatascience.com/why-feature-correlation-matters-a-lot-847e8ba439c4

(5) Feature selection based on correlation in python:
https://johfischer.com/2021/08/06/correlation-based-feature-selection-in-python-from-scratch/

(6) Undersampling to deal with imbalanced datasets:
https://towardsdatascience.com/having-an-imbalanced-dataset-here-is-how-you-can-solve-it-1640568947eb

(7) Our data preparation, preprocessing and undersampling:
https://github.com/JuliaTsymbal/MachineLearningProject

(8) ROC Curve:
https://www.displayr.com/what-is-a-roc-curve-how-to-interpret-it/