Machine Learning Engineer Nanodegree

Capstone Project

- Stiven López Giraldo
- May 12, 2018

I.Definition

Project Overwiew

Churn analysis is one of the most important problems for different companies, from sectors such as telecommunications to banking. A proper analysis together with a well-defined strategy allows to increase the user retention rate, and not only that, it can also help to focus on improving the user experience of those who are more likely to migrate to another campaign. Since users are a valuable asset for different companies and the constant creation of financial services from different companies in the sector, the need for customer retention increases.

The bank's objective is to find out which users are most likely to leave the company. Certain user characteristics are calculated in order to generate that probability, and depending on each user, a business action is taken to retain this user.

Problem Statement

The solution to the problem of interest is to build a classification model that generates the probability that a user will leave the company, with this probability the users in the area in charge will create ranges and define actions for the different users and encourage them to don't leave the company. The model will be an Inference Pipeline trained in Amazon SageMaker, this will be made up of a first container that does all the preprocessing of the data, and the second a container that houses an XGBoost for training.

The first container will perform transformations such as: select columns, convert categorical variables to dummy variables, among others. Scikit-learn transformers and custom pandas will be built.

The second container will receive the ready data from the first container, and will train the XGBoost to later make inferences.

The goal of developing such a model is to avoid data leakage problems, and to have the same transformations in training data as in new data.

Metrics

Our problem presents a class imbalance, although it is not as serious as certain problems that are usually found. In our case we have around 80% who are still clients and the remaining 20% who closed the account.

To evaluate performance, the F1 Score is used, which is the weighted average of precision and recall. Therefore, this score takes into account both false positives and false negatives and is useful for problems of unequal class distribution.

II. Analysis

Data Exploration

The data set provided by Kaggle contains details of the users of the bank, in addition to the binary variable that reflects whether a user closed the account or is still our user, which is the phenomenon that we are interested in predicting and understanding to help make better informed decisions. .

The data set is loaded in S3, the data is downloaded and split in training, validation, testing using a stratified sampling on the dependent variable, this with the aim of leaving the same proportions of the users who closed the account and the who are still customers. The following files remain:

- train.csv: training data with its respective target variable.
- validation.csv: validation data with its respective target variable.
- test.csv: test data without response (observed value)
- test_label.csv: observed value of the test data set.

The dimensions of each data set are as follows:

- train.csv has 13 variables (characteristics) and 7000 observations.
- validation.csv has 13 variables (characteristics) and 1800 observations.
- test.csv has 12 variables (characteristics) and 1200 observations.
- test_label.csv has 1 variable and 1200 observations.

The variables of the data sets are:

- **RowNumber**: Row Numbers from 1 to 10000.
- **CustomerId**: Unique Ids for bank customer identification.
- **Surname**: Customer's last name.
- **CreditScore**: Credit score of the customer.
- **Geography**: The country from which the customer belongs.
- **Gender**: Male or Female.
- **Age**: Age of the customer.
- **Tenure**: Number of years for which the customer has been with the bank.
- **Balance**: Bank balance of the customer.
- **NumOfProducts**: Number of bank products the customer is utilizing.
- **HasCrCardBinary**: Flag for whether the customer holds a credit card with the bank or not.
- **IsActiveMember**: Binary Flag for whether the customer is an active member with the bank or not.
- **EstimatedSalary**: Estimated salary of the customer in Dollars.
- **Exited**: Binary flag 1 if the customer closed account with bank and 0 if the customer is retained.

Exploratory Visualization

The idea is to do an exploratory analysis on the data set, in order to understand the data more, but additionally to understand the distributions of the data and choose an appropriate algorithm, based on what the data shows us.

First, a univariate analysis is done together with descriptive statistics, then a multivariate analysis that gives us indications that variables may be the best predictors and have more discriminative power.

Univariate analysis

A univariate analysis will be made for the different variables, however, only the most interesting findings are shown so as not to generate as much noise in the document.

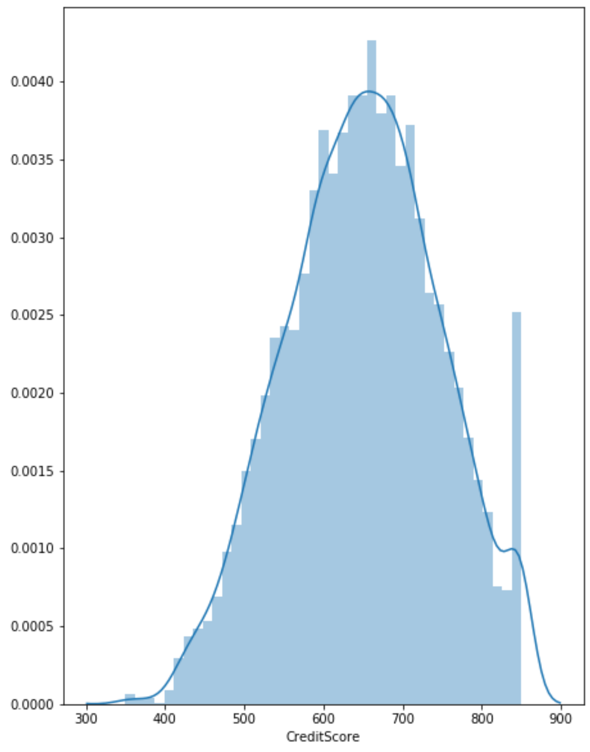
Some descriptive statistics of the numerical variables can be seen below:

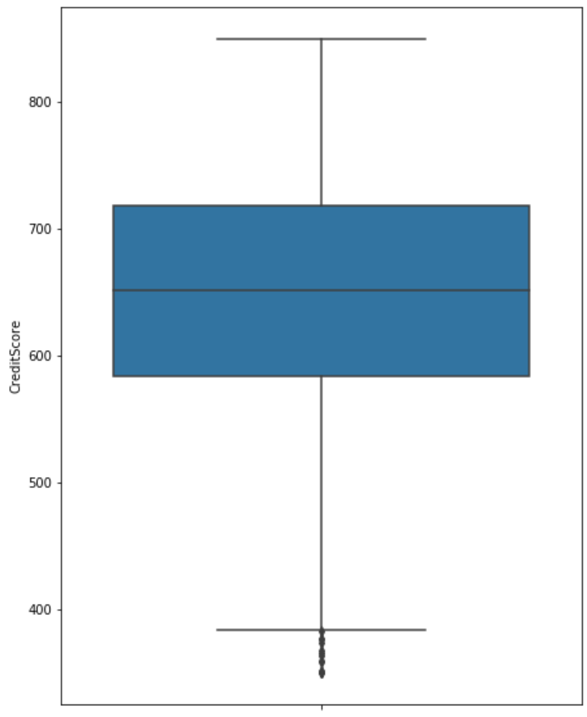
	count	mean	std	min
RowNumber	10000.0	5.000500e+03	2886.895680	1.00
CustomerId	10000.0	1.569094e+07	71936.186123	15565701.00
CreditScore	10000.0	6.505288e+02	96.653299	350.00
Age	10000.0	3.892180e+01	10.487806	18.00
Tenure	10000.0	5.012800e+00	2.892174	0.00
Balance	10000.0	7.648589e+04	62397.405202	0.00
NumOfProducts	10000.0	1.530200e+00	0.581654	1.00
EstimatedSalary	10000.0	1.000902e+05	57510.492818	11.58
Exited	10000.0	2.037000e-01	0.402769	0.00

We also have some statistics for the categorical variables:

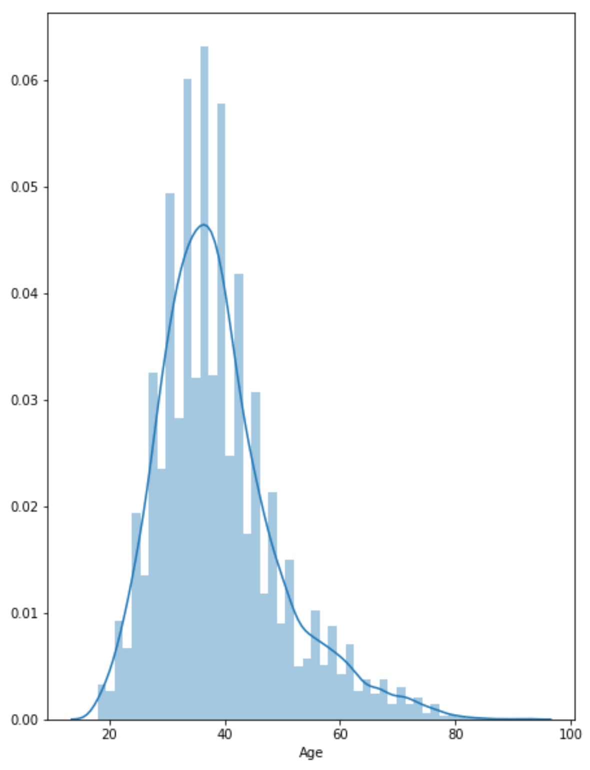
	Surname	Geography	Gender	HasCrCard	IsActiveMember
count	10000	10000	10000	10000	10000
unique	2932	3	2	2	2
top	Smith	France	Male	Yes	Yes
freq	32	5014	5457	7055	5151

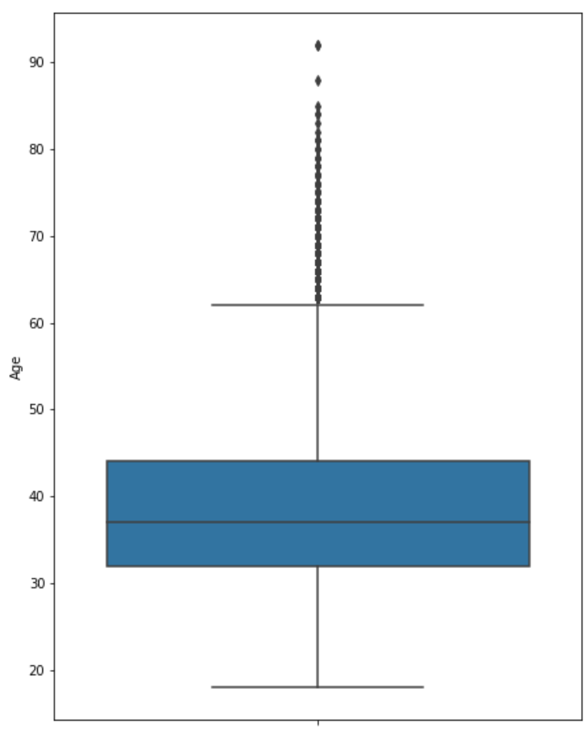
We see that the credit score variable presents a distribution very similar to the normal one, so it would not be necessary to apply any monotonic transformation, and without taking into account that the algorithm to be used does not need a specific distribution in the independent variables. In addition, unusual values (outliers) are presented below approximately 400.



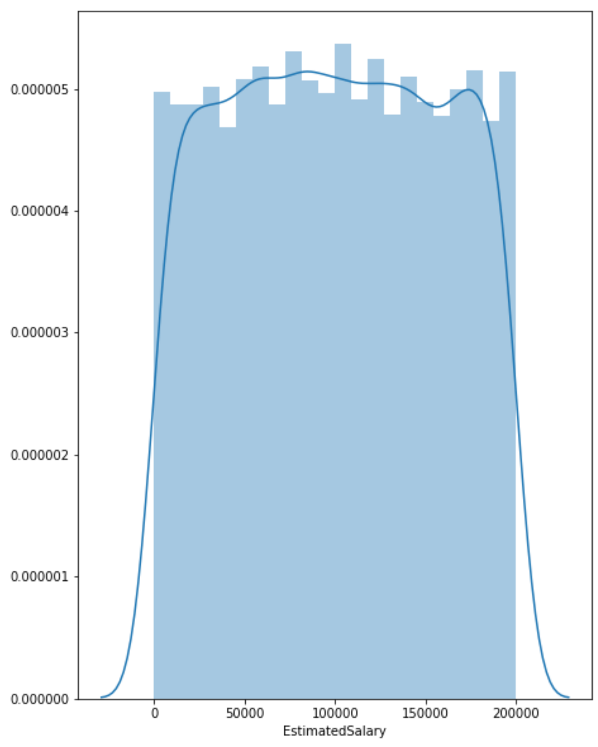


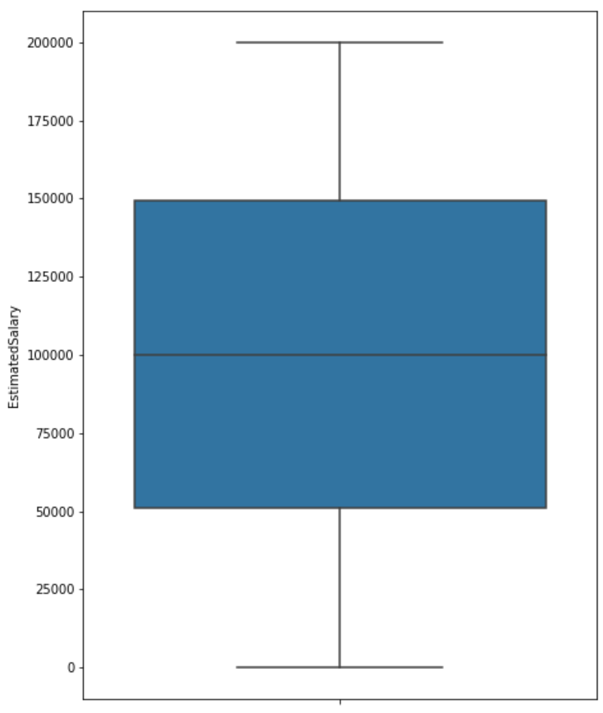
In the case of age, there is a positive bias in this variable, and we can see that most of the bank's users are around 40 years old, and due to the bias, it seems that it is not so usual for the bank to have very old users, even in the boxplot we can see that values above approximately 63 years are usually unusual ages with respect to a large part of the bank's users.



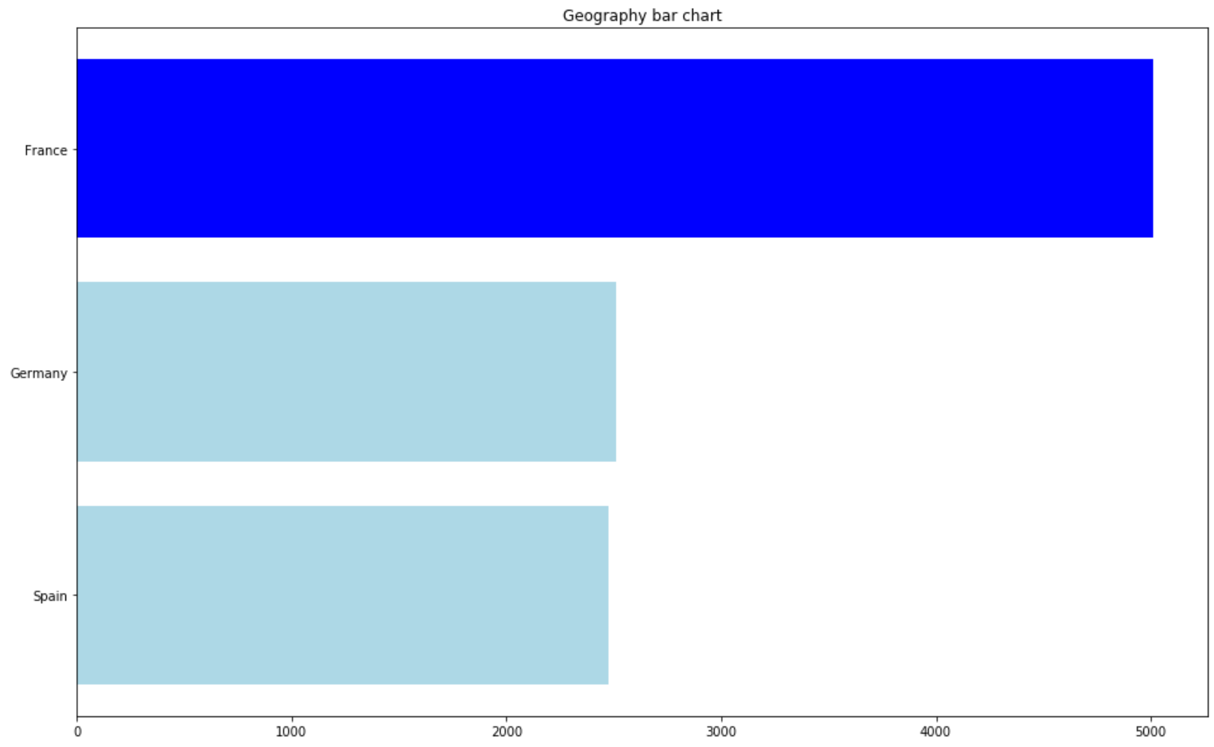


The estimated salary is a very interesting variable in real life, however, in this data set it is not necessarily so and this is because the salary is distributed evenly, so it is very likely that the estimated salary will fall between 0 and 200000.

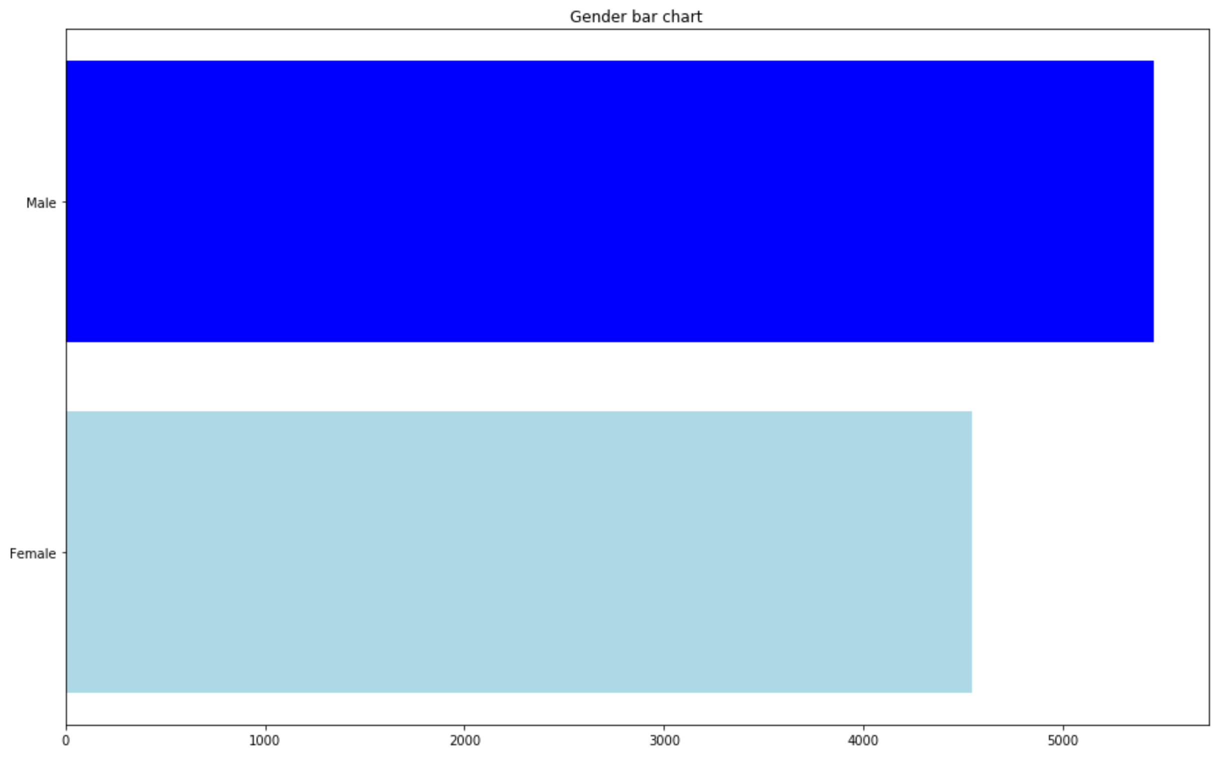




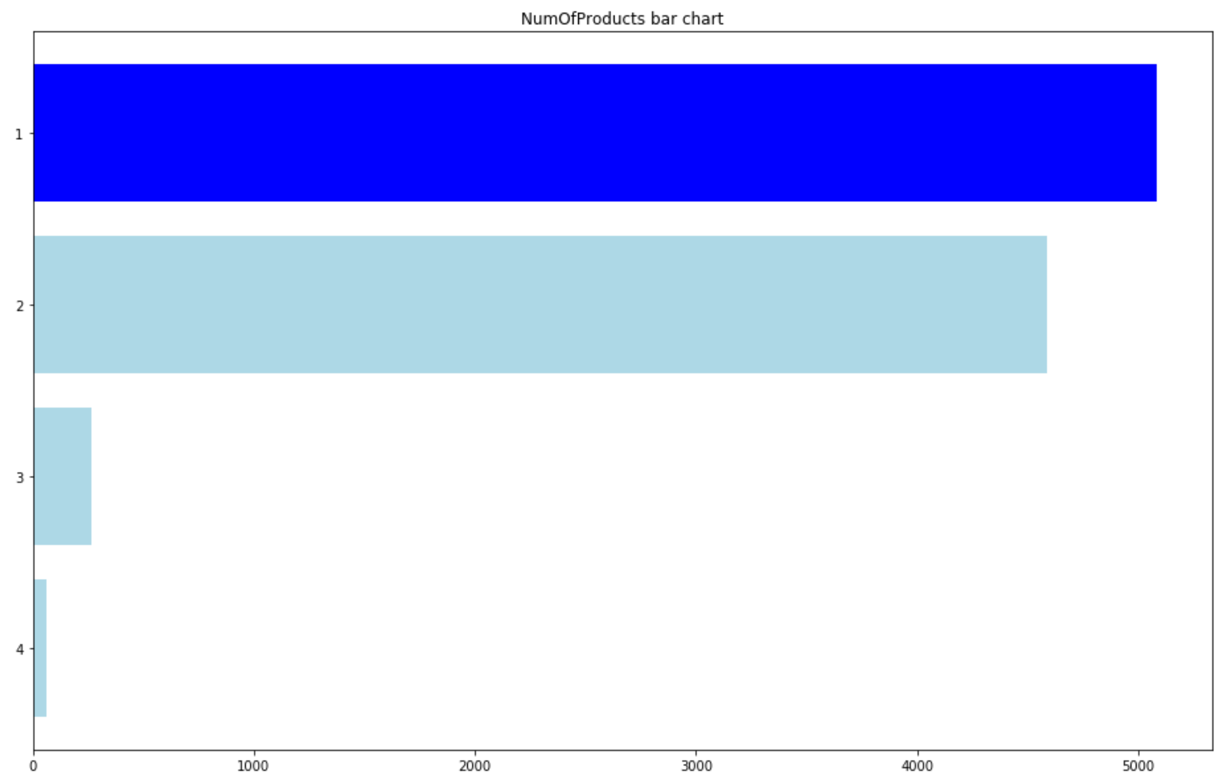
In the bar graph we see that the bank only has clients from three countries, which are: France, Germany, and Spain; but it has much more of France. So a suggestion for the bank would be to start encouraging more German and Spanish users to use the bank's financial products and even diversify users from other nationalities.



In general there is no significant difference in the gender of the users, something that seems good to me and it is because if the proportion of one with respect to the other would be high, the model would possibly be falling into a gender bias simply because it has more observations of the one or the other.



We see that the majority of users have only one product, however, the difference between users who have one product and two is not much, while there are very few users who have three or four products. So the bank should encourage them to consume more financial products to improve profitability, but also by improving the experience and meeting the needs of different users. It would be necessary to review in real life the quantity of products offered by the bank, because if it has a small offer this could explain why certain users close the account and that they would not find everything they need in the bank.

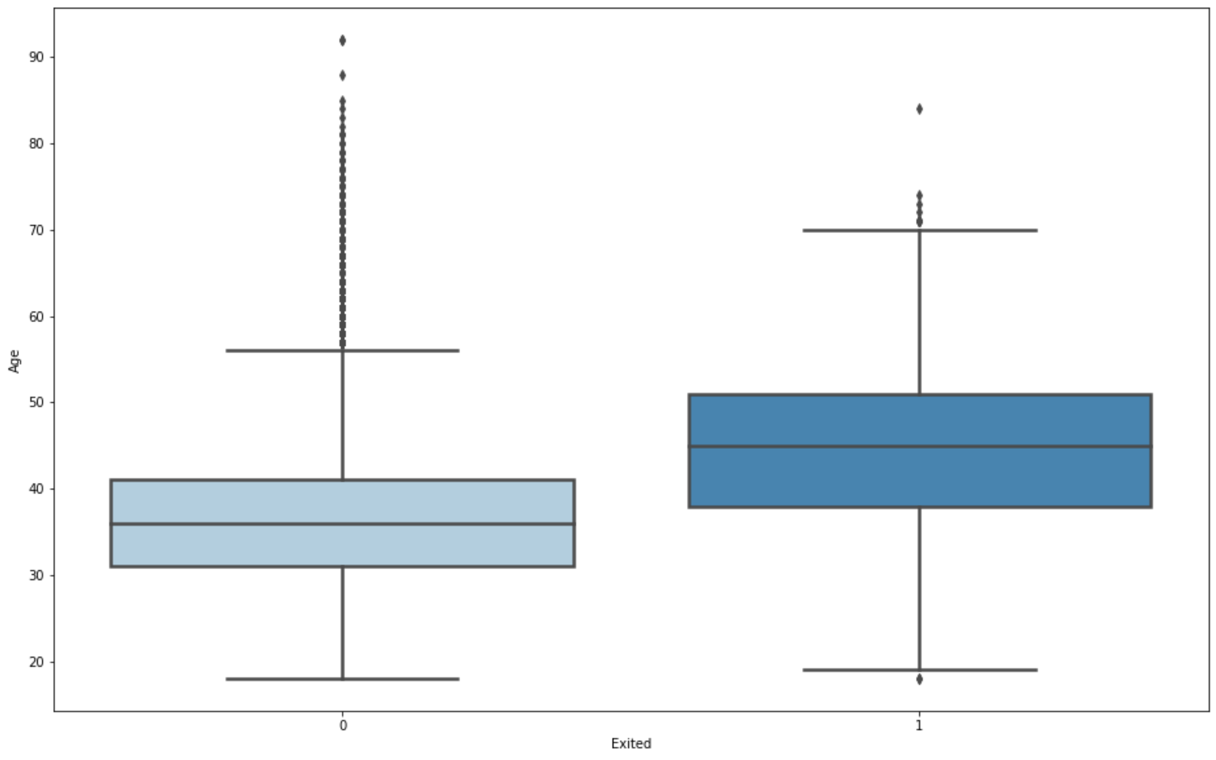


Multivariate Analysis

In the multivariate analysis, the numerical variables that help to discriminate between users who close and those who do not have an account at the bank will be shown.

The variable such as age shows us something interesting, and that is that older people are more likely to close the account, while those below 40 usually leave it open and it is very unusual for users with high ages and with the account open, so this generates a hypothesis and it is something that was seen previously in the histogram and that is that users are usually younger, so it is possible that the bank's financial products are very focused on this younger segment.

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb149398c18>
```



In addition to the above, the other variables presented a very similar distribution, both when the account was blocked or not.

Algorithms and Techniques

Extreme Gradient Boosting (XGBoost) is an implementation of the gradient boosting machines that is highly flexible and versatile while being scalable and fast. XGBoost works with most regression, classification.

In short, XGBoost is a variation of boosting - an ensemble method algorithm that tries to fit the data by using a number of "weak" models, typically decision trees. The idea is that a "weak" classifier which only performs slightly better than random guessing can be improved ("boosted") into a "stronger" one that is arbitrarily more accurate (source: Y. Freund, R. E. Schapire) (source: R. E. Schapire). Building on the weak learners sequentially, at every round each learner aims to reduce the bias of the whole ensemble of learners, thus the weaker learners eventually combined into a powerful model. This idea gave birth to various boosting algorithms such as AdaBoost, Gradient Tree Boosting, etc.

There is a number of advantages in using XGBoost over other classification methods:

- **Work with large data**: XGBoost packs many advantageous features to facilitate working with data of enormous size that typically can't fit into the system's memory such as distributed or cloud computing. It is also implemented with automatic handling of missing data (sparse) and allows continuation of training, or batch training which was a tremendous help for me in this project.
- **Built-in regularization**: XGBoost supports several options when it comes to controlling regularization and keeping the model from overfitting, including gamma (minimum loss reduction to split a node further), L1 and L2 regularizations, maximum tree depth, minimum sum of weights of all observations required in a child, etc.
- **Optimization for both speed and performance**: XGBoost provides options to reduce computation time while keeping model accuracy using parallelization with multi-core

CPU, cache optimization, and GPU mode that makes use of the graphics unit for tree training.

The XGBoost is a very powerful algorithm, additionally in the data we find some variables with outliers, so it is important to use an algorithm that is robust to these as this, and additionally the algorithm as others belonging to the family of non-assembly methods. they need variables to have a certain specific distribution, so it makes for a very good model to use.

Benchmark

The benchmark chosen is to infer for each new user the most frequent value of the users who are in our sample, that is, for each new user it is predicted that he will remain in the bank and that he does not have a high probability of leaving.

If we do this on the test data, we get the following metrics:

- AUC : 0.5
- Accuracy: 0.8
- Precision: 0
- Recall: 0
- F1 Score: 0

III. Methodology

Data Preprocessing

In our case there are no missing values, nor duplicates. However, to the `HasCrCard` and `IsActiveMember` variables that come as one or zero for when it is positive or negative respectively, a transformation is applied so that they remain as Booleans (True or False) that are later transformed into dummies variables along with the categorical variables: Geography, and Gender. In the case of numeric variables, no monotonic transformation is done for them, since the algorithm is able to work perfectly with data with different numerical scales, so it is not necessary.

Here is a step-by-step discussion of preprocessing which is the first container to train in Amazon SageMaker:

I. The transformation is done in boolean values for `HasCrCard`, and `IsActiveMember`.

II. The data is converted to numeric or categorical type as the case may be to improve performance, and in case any variable does not load as it should be.

III. Only the variables that serve to train the model are selected, variables such as id's, and names are not selected.

IV. Of the selected variables, the numerical and categorical ones are selected separately to apply the appropriate transformations to each type, in this case the categorical variables become dummies (one-hot-encoding) variables and the numerical ones do not undergo any transformation.

V. The pipeline is adjusted, with which the same transformations will be applied to the new data, so things are guaranteed that there is no data leak.

When the raw data is entered, they are passed through this pipeline, the aforementioned transformations are applied and the result is saved as an artifact in S3 (csv file with the transformations applied), later these data enter the second container where the XGBoost is housed to adjust the model and to be able to generate inferences later.

Implementation

In the previous point, the pre-processing of the data is mentioned, which is run in a container and the code is the combination of different custom pipelines inheriting some Scikit-Learn classes and the transformations are done with Pandas, later to that the data goes to the second container where the model simply fits. However, at the time of building the Inference Pipeline in Amazon SageMaker I had a complication, and that is that the content_type becomes application/json for when it is going to be invoked from the second container, and the solution in generating an environment variable of the form: {"SAGEMAKER_DEFAULT_INVOCATIONS_ACCEPT": "text / csv"} and thus no problems occur between containers, which should only be done in the case of the XGBoost because for another SageMaker algorithm such as Linear-Learner it is not necessary and does not occur this problem.

Refinement

In order to improve the result of the adjusted model, the best hyperparameters are searched using the search done by Amazon SageMaker, which is an informed search on hyperparameter distributions, which has good performance and is much faster than a grid search. .

The distribution spaces for the variables are:

- **max_depth**: It is the maximum depth of the tree, the bigger this hyperparameter is, the model will be more complex and there will be a greater risk of overfitting. A distribution space of (3, 12) was established
- **subsample**: Proportion of subsample of training instances, that is, if we choose 0.5, 50% of the data will be randomly sampled to build the tree, and thus there will be a lower risk of overfitting. A distribution space of (0.5, 0.9) was established
- **num_round**: The number of boosting rounds. A gap of (100, 300) was established

---

IV. Results

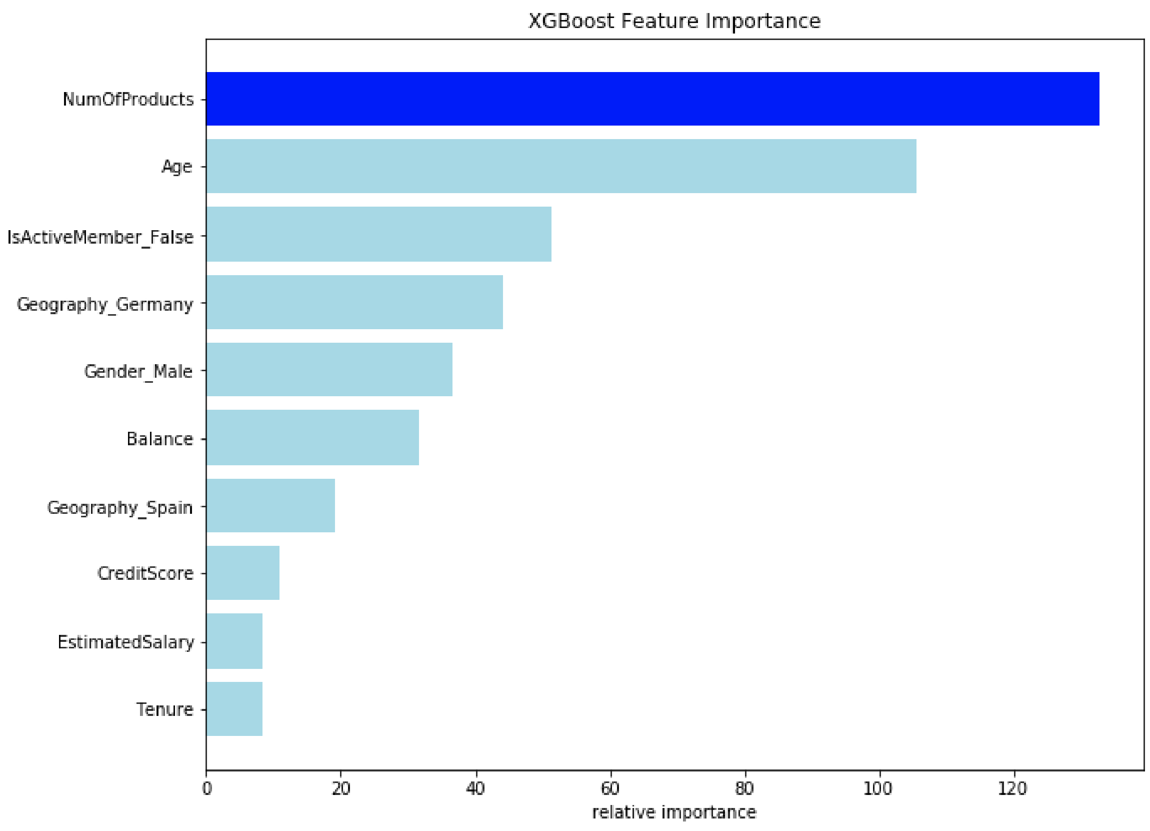
Model Evaluation and Validation

After evaluating the performance of the model with respect to the benchmark and the best model after executing hyperparameter optimization, it was decided to leave the initial model without hyperparameter optimization because it presents better results and generally performs better than our benchmark. The results are shown below:

	benchmark	xgboost	xgboost_cv
AUC	0.5	0.78	0.73
Accuracy	0.8	0.80	0.83
Precision	0.0	0.50	0.57
Recall	0.0	0.75	0.57
F1 score	0.0	0.60	0.57

Feature importance

Something very interesting about the XGBoost is that although it is a model that has a good predictive performance, it is logical that it sacrifices a little interpretability, however, with the importance of the variables it calculates, we have a measure to understand which are the variables that are most helping for the prediction that a user does not close the account or does so. In this case, the two most important variables were the number of products they have with the bank, clearly this variable would affect different users positively or negatively, however, one would expect that the more products the customer is more loyal to the brand. and do not close the account, while a customer with few products will not feel the same attachment to the brand; and on the other hand, age.



Justification

The model evaluated in the test data did indeed show better performance, and obtained interesting metrics so as not to consider feature engineering where more variables could be created, there would probably be a better one more interesting in the current model.

After the improvements, this model could be deployed through a REST API for business users to enter an internal website of the bank and enter the raw values and generate the inference, additionally with the first container it is guaranteed that it will not there will be problems in the model because the necessary transformations will be made in the custom pipeline, you only need to send the values separated by ";", for example, `'4635;15583353;Floyd;610;Spain;Female;45;3;0.0;1;1;0;38276.84'` and the inference will be made, additionally that the model has a very good latency because it takes approximately 0.25 seconds to generate the inference.