Machine Learning Engineer Nanodegree

Capstone Project

Abdulellah Alnumay
August 11th, 2018

I. Definition

Project Overview

From frontline support teams to C-suites, customer satisfaction is a key measure of success. Unhappy customers don't stick around. What’s more, unhappy customers rarely voice their dissatisfaction before leaving. In this problem I'll try to identify dissatisfied customers early in their relationship. Doing so would allow the bank to take proactive steps to improve a customer's happiness before it's too late. I'll work with hundreds of anonymized features to predict if a customer is satisfied or dissatisfied with their banking experience.
This is Santander Customer Satisfaction competition on Kaggle. The Data for this project was provided by Santander Bank on Kaggle as a part of the competition.The dataset was a large data set with a huge number of anonymized attributes.
The project is to try to build a model that can predict if a customer is satisfied with the banking experience that he/she is having or not, in order to prevent losing customer due to being unhappy with services provided or bad experience.

Problem Statement

Correctly predicting if the customer is satisfied with the services provided or not if important for a business owner to make decisions that concern their customer and affects them.
I'll try to build a classifier that uses previous data of customer to determine whether if they are satisfied or not with their banking experience. It's useful to know if a customer is not satisfied while they are still customers to correct whatever is making them unhappy with their experience.
I will build a few models and try to determine which one will perform better than the other and focus on that model to improve it's performance.
My goal is to end up with a model that scores better than the benchmark model.

Metrics

For this problem, and due to the nature of the dataset and how imbalanced it was, I was faced with a few option to choose between for handling the class imbalance of the dataset. One of the options was using area under ROC as a metric to measure the performance of the models, which happens to be the scoring that's being used in the Kaggle competition.
The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) [1]. The area under the ROC curve, then, is defined as follow:

$$A = \int_{\infty}^{-\infty} \mathrm{TPR}(T)\mathrm{FPR}'(T)\,dT = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(T' > T) f_1(T') f_0(T)\,dT'\,dT = P(X_1 > X_0)$$

And since this performance method would handle an unbalanced set, I've selected this metric as my metric for the project.

II. Analysis

Data Exploration

The data was provided by Santander Bank as part of a Kaggle competition, and is available to anyone who's registered in the competition.
The dataset had 70,000 instances, with 369 attributes, which are anonymized due to the nature of the data, and to ensure the customer's privacy.
First thing I did was get a description of the data:

	ID	var3
count	76020.000000	76020.000000
mean	75964.050723	-1523.199277
std	43781.947379	39033.462364
min	1.000000	-999999.000000
25%	38104.750000	2.000000
50%	76043.000000	2.000000
75%	113748.750000	2.000000
max	151838.000000	238.000000

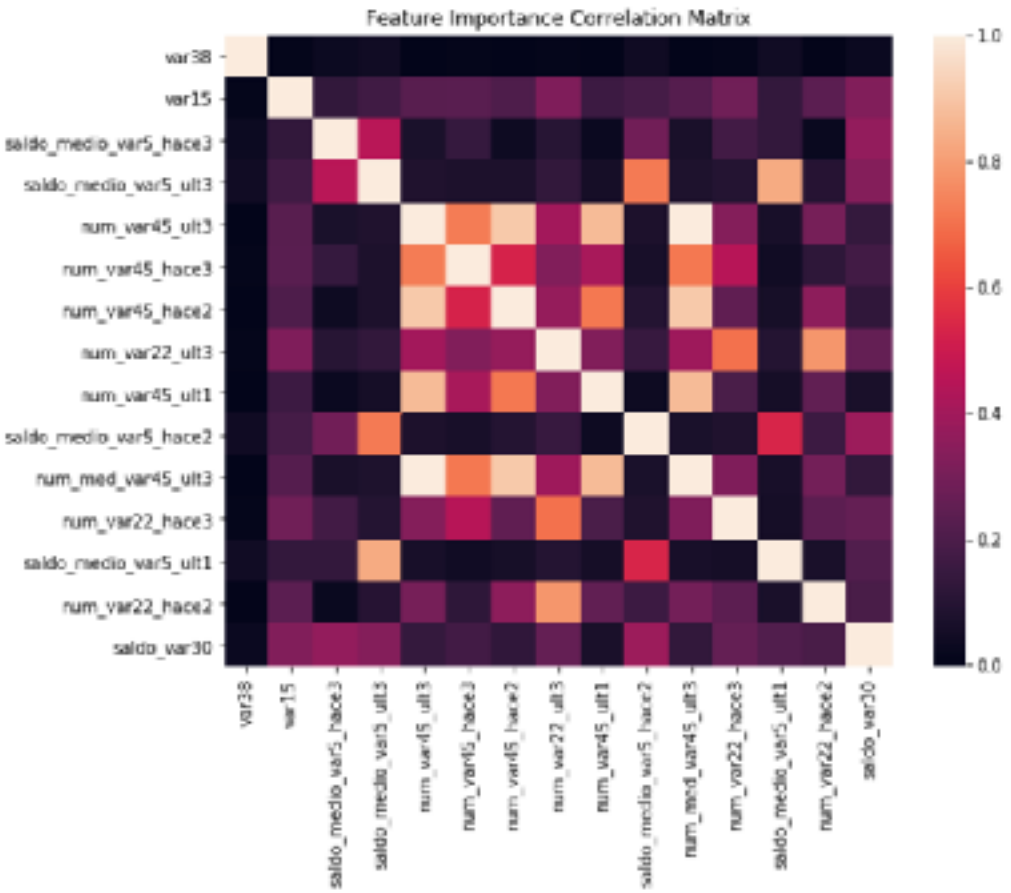
Which immediately reveled something interesting, the extreme values in one of the attributes.
After investigating that column, I've found that this value is present in about 100 instance of the data. Also, exploring extreme values in the dataset, I've found that there are 126 columns with extreme values (grater that 10000, or less that -10000), which led me to believe that the values in attribute 'var3' do not have a negative effect on the training process.

After that, there was that matter of useless attributes. I checked the dataset and found that there is 34 columns that have a standard deviation of zero, which meant that they were constant, and wouldn't have an effect on the mode. Then I looked for duplicate columns, and found 29 duplicate columns.

Exploratory Visualization

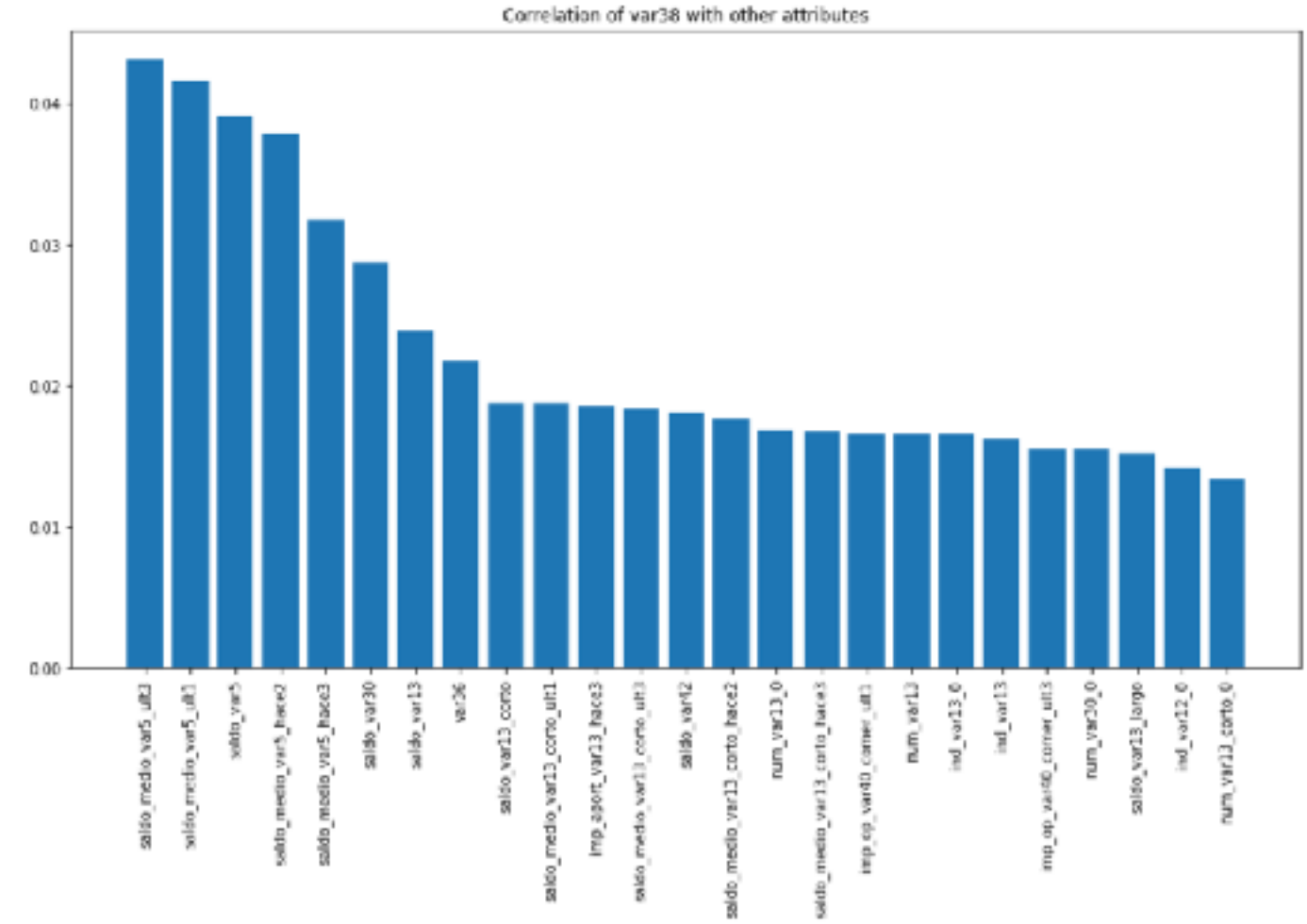
Since the number of attributes is very large to visually explore, I will be performing my visual exploration on a dataset reduced to selected features that I will explain in the preprocessing section.

Firstly, I check the correlation matrix of the dataset:

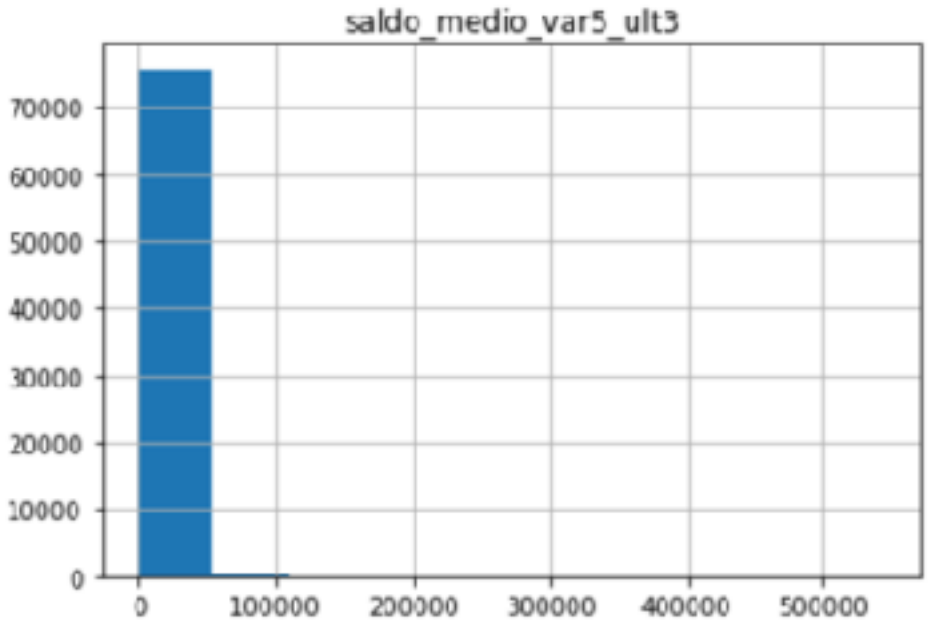


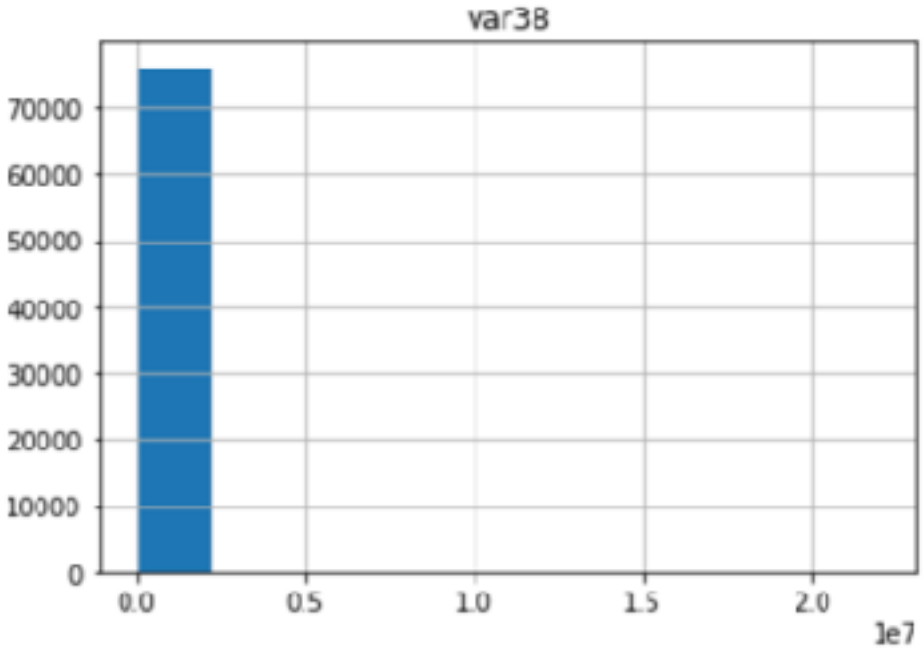
And this show some correlation between few attributes. There's a really good correlation between 'num_var45_utl3' and 'num_medio_var45_utl3', and by the name I think it's safe to guess that these attributes supposed to represent similar data.
What's interesting is that the attribute 'var38', which XGBoost considers the most important attribute, doesn't have any correlation with any of the selected features, which made me

wonder if there was any feature in the full dataset that had any correlation with ‘var38’:

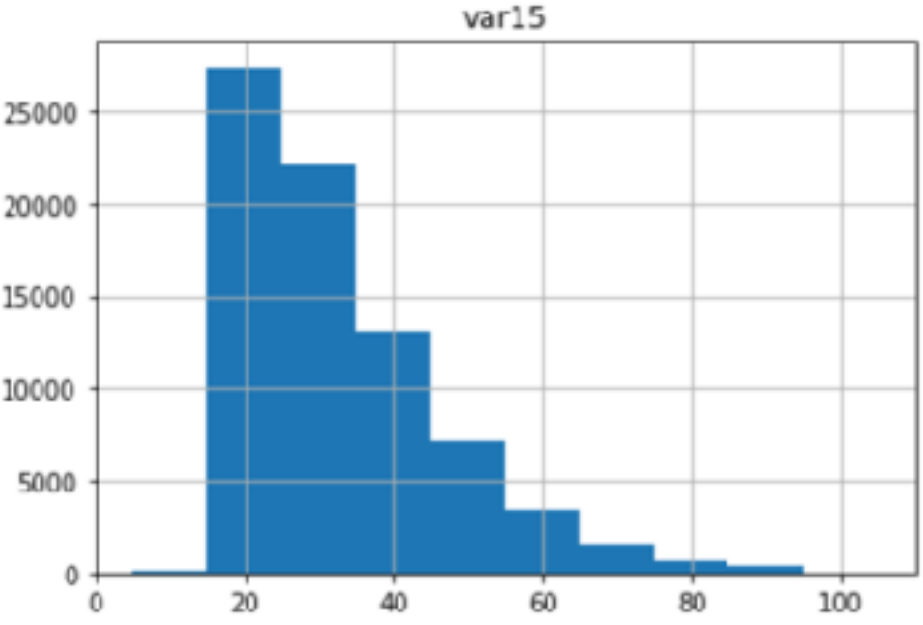


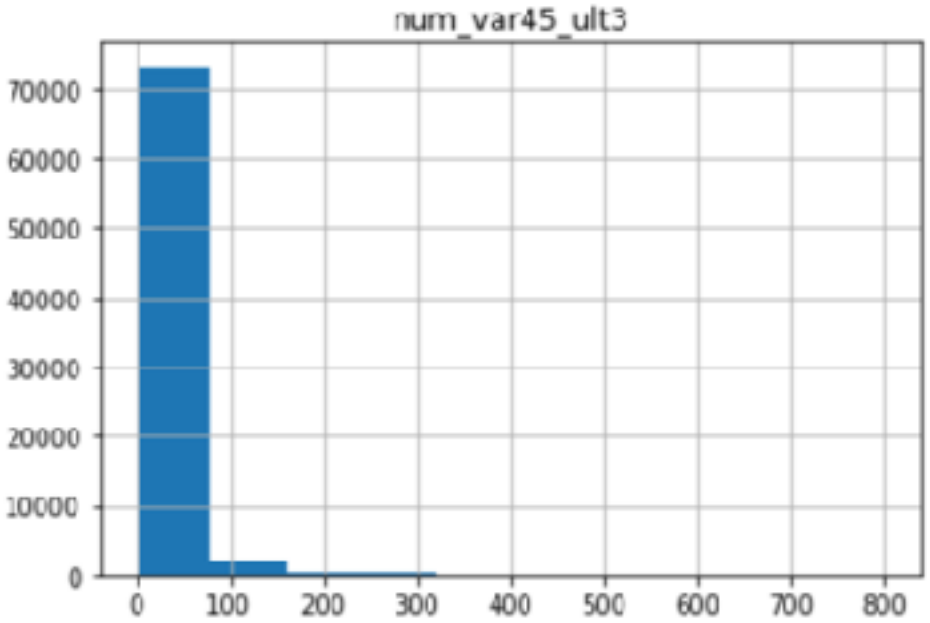
After that I looked at the histogram of the selected features:

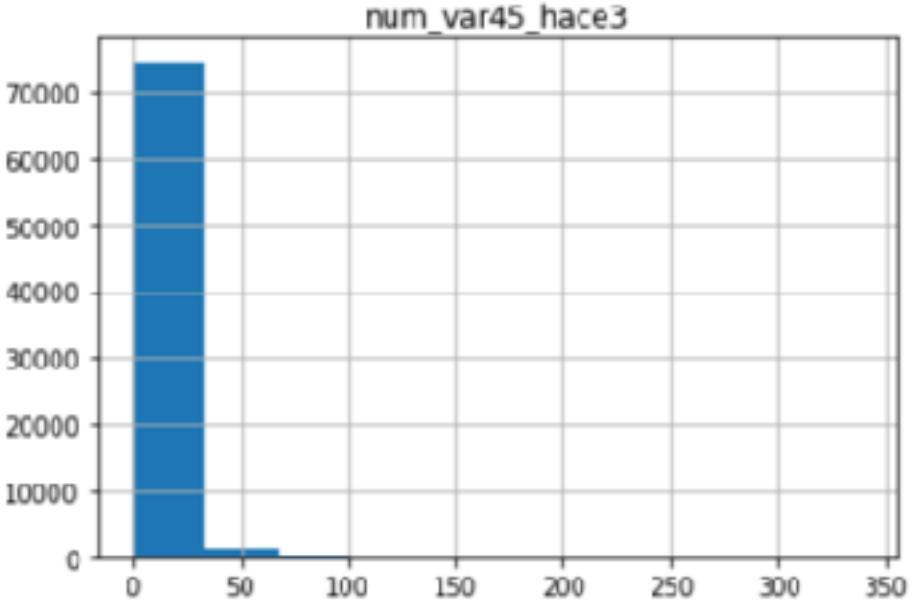


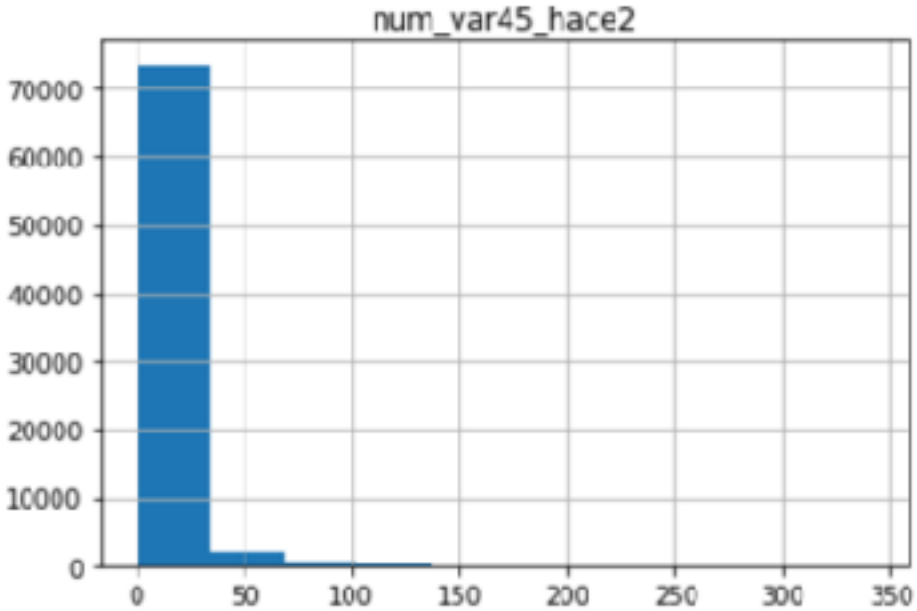

var38
70000
60000
50000
40000
30000
20000
10000
0
0.0
0.5
1.0
1.5
2.0
1e7

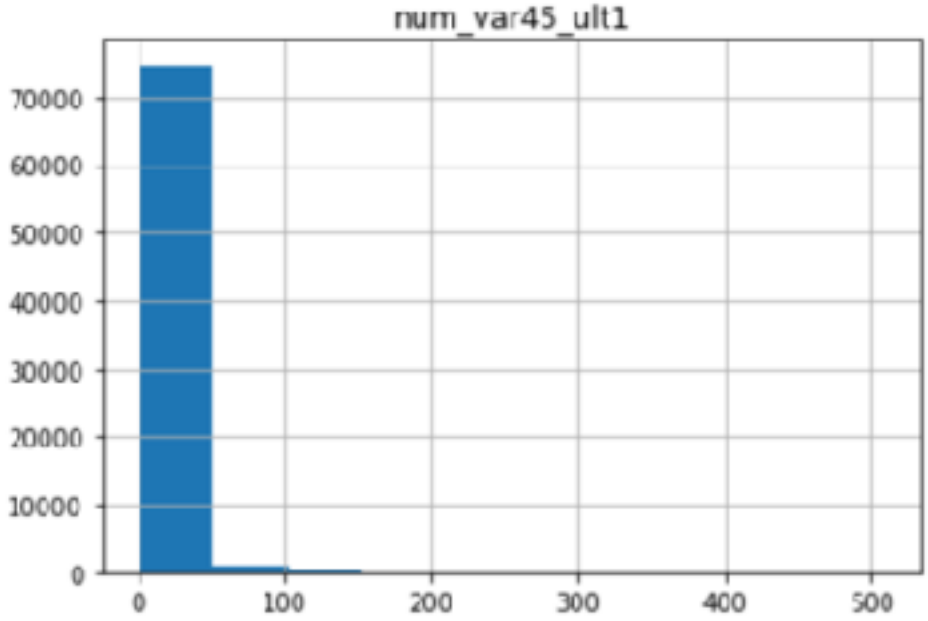

saldo_medio_var5_hace3
70000
60000
50000
40000
30000
20000
10000
0
0
200000
400000
600000
800000
1000000
1200000
1400000
1600000

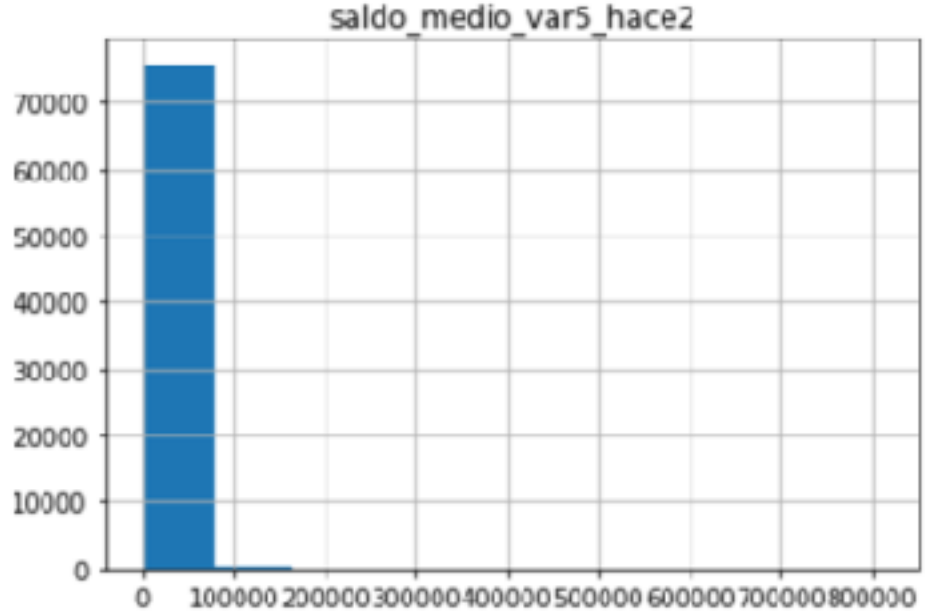

var15
25000
20000
15000
10000
5000
0
0
20
40
60
80
100

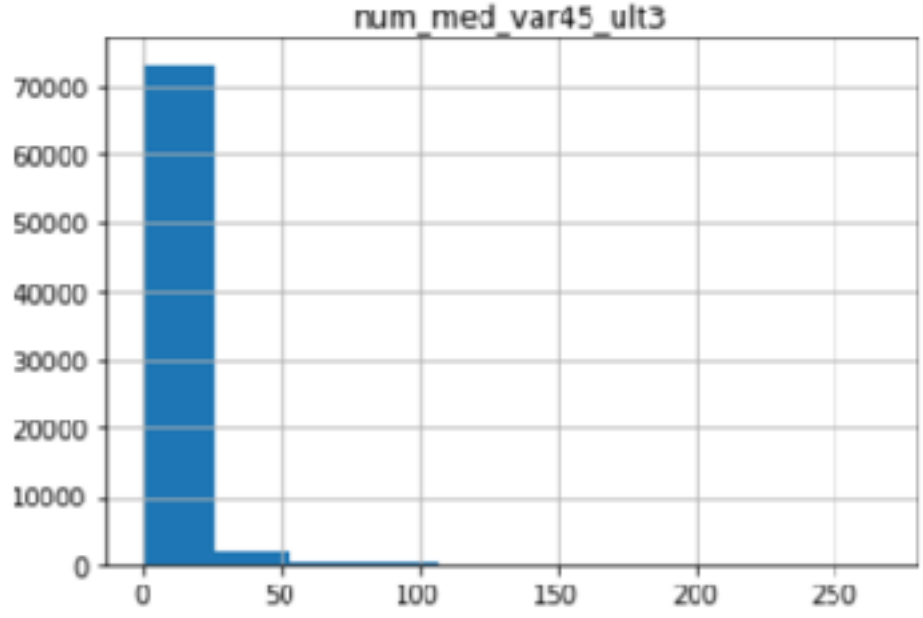

num_var45_ult3
70000
60000
50000
40000
30000
20000
10000
0
0
100
200
300
400
500
600
700
800

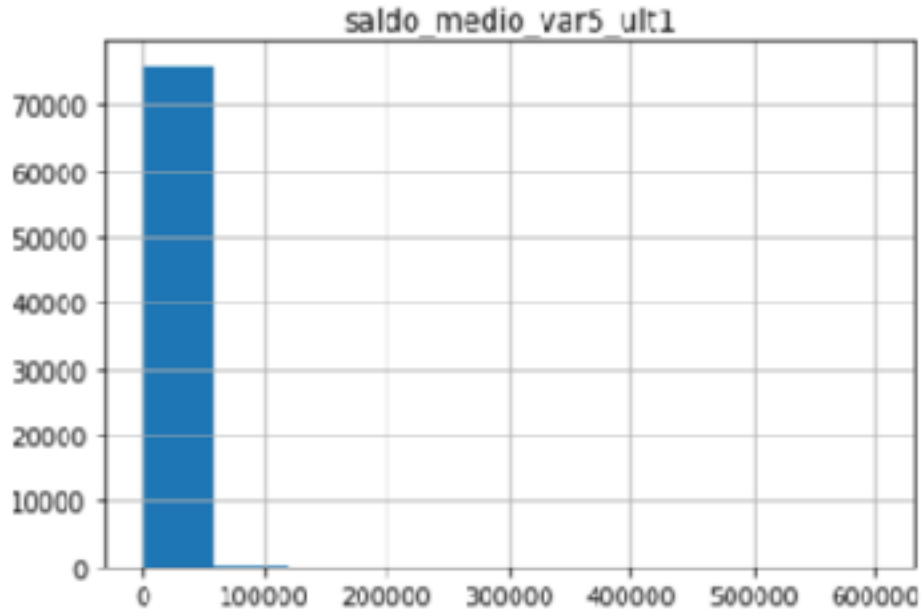

num_var45_hace3
70000
60000
50000
40000
30000
20000
10000
0
0
50
100
150
200
250
300
350

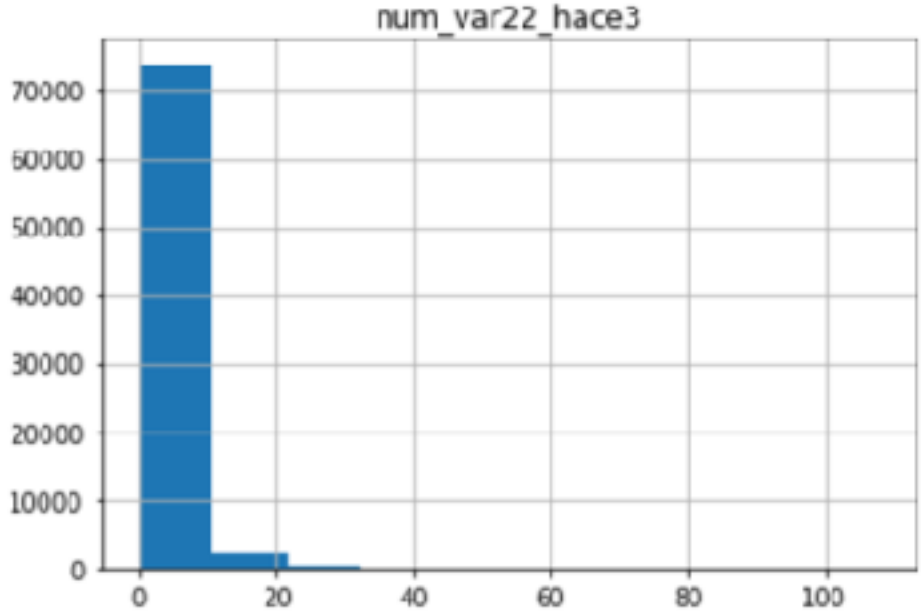

num_var45_hace2
70000
60000
50000
40000
30000
20000
10000
0
0
50
100
150
200
250
300
350

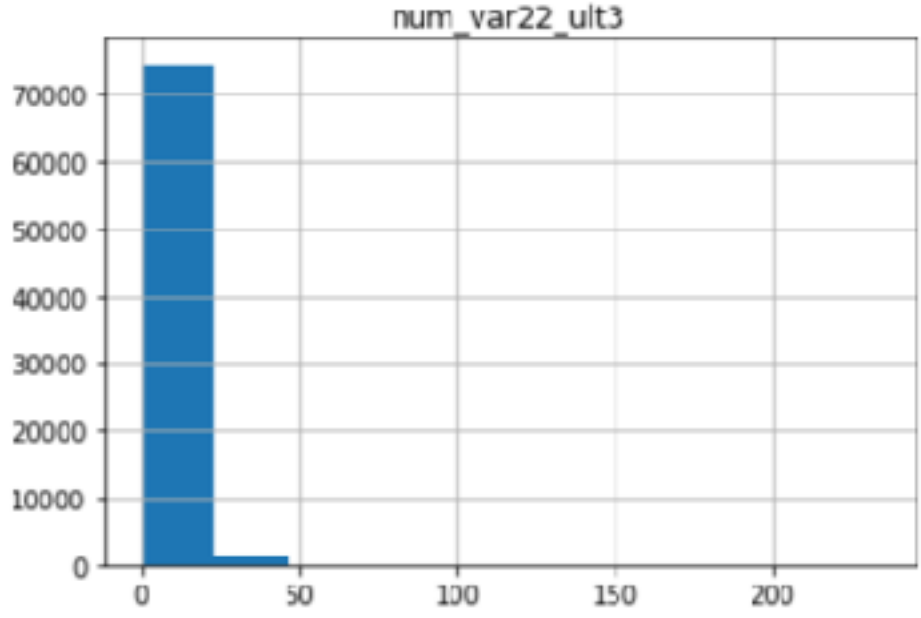

num_var45_ult1
70000
60000
50000
40000
30000
20000
10000
0
0
100
200
300
400
500


saldo_medio_var5_hace2
70000
60000
50000
40000
30000
20000
10000
0
0
100000
200000
300000
400000
500000
600000
700000
800000

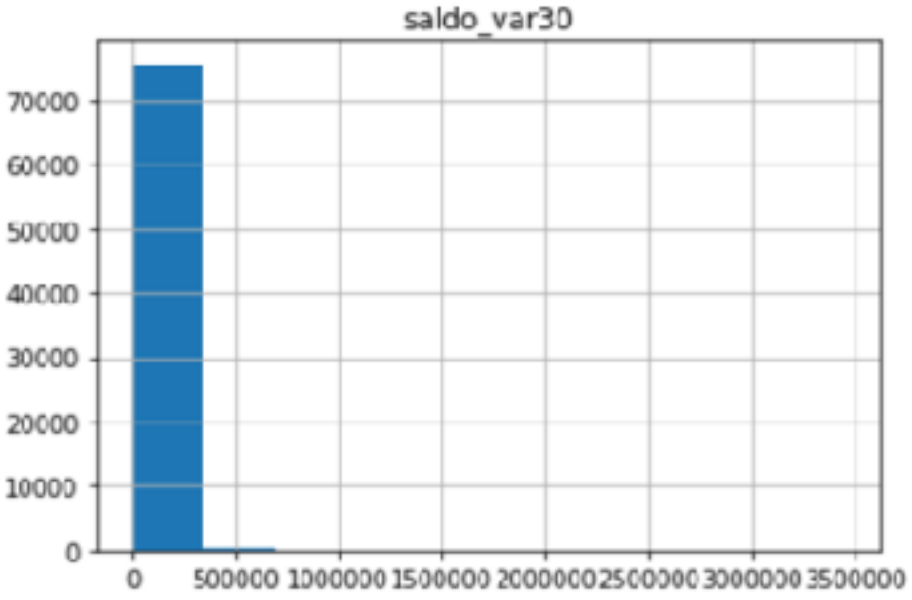

num_med_var45_ult3
70000
60000
50000
40000
30000
20000
10000
0
0
50
100
150
200
250


saldo_medio_var5_ult1
70000
60000
50000
40000
30000
20000
10000
0
0
100000
200000
300000
400000
500000
600000


num_var22_hace3
70000
60000
50000
40000
30000
20000
10000
0
0
20
40
60
80
100


num_var22_ult3
70000
60000
50000
40000
30000
20000
10000
0
0
50
100
150
200


num_var22_hace2
70000
60000
50000
40000
30000
20000
10000
0
0
20
40
60
80
100
120


saldo_var30
70000
60000
50000
40000
30000
20000
10000
0
0
500000
1000000
1500000
2000000
2500000
3000000
3500000

Algorithms and Techniques

The classification algorithms that were chosen in the proposal were SVM, Neural Network, and XGBoost. And the reasoning behind each one is:

- SVM: SVMs with rbf kernels can classify non-linear data very effectively, and they are one of the most powerful algorithms for classification and regression, so it would be unusual not to consider SMVs for a classification problem.

- Neural Network: Neural Networks are a very powerful algorithm if built correctly, as they are excellent in finding patterns in the input data. NNs, also, can handle large datasets better than most algorithms, which makes them a good candidate for this problem.

- XGBoost: One of the most successful algorithms in Kaggle, and the go-to in competitions. XGBoost is fast, highly accurate, handles missing values in data, and prevents over-fitting. What can one want more from an algorithm?

Once I looked at the data and saw 369 attributes, I immediately knew that I will need to perform some feature selection to reduce the number of attributes of the data. Therefor, I performed five feature selection method on the dataset:

- F-Class
- Mutual Information
- Recursive Feature Elimination
- Feature Importance
- Single Feature Performance

And the number of feature selection methods is explained by my desire to make sure that the methods chosen are the most effective. After applying the five methods on the dataset I resulted in five sets, and to determine what I should continue with, I train a Neural Network and an XGBoost model using each set to find the highest scoring set:

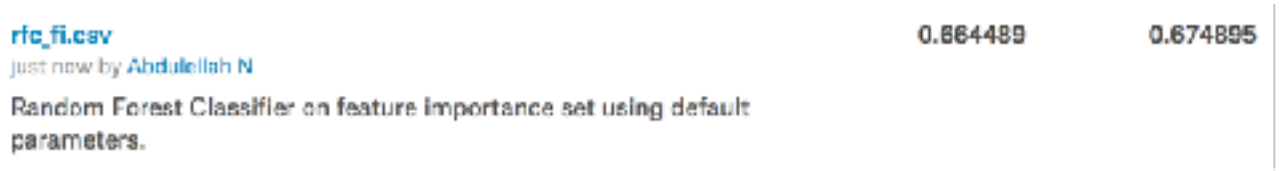
xgb_sf_basic.csv 11 hours ago by Abdulellah N XGB model on single feature performance set using default parameters	0.804491	0.821448
xgb_fi_basic.csv 11 hours ago by Abdulellah N XGB model on feature importance set using default parameters	0.817715	0.833499
xgb_rfe_basic.csv 11 hours ago by Abdulellah N XGB model on RFE set using default parameters	0.721641	0.734105
xgb_mi_basic.csv 11 hours ago by Abdulellah N XGB model on mutual information set using default parameters	0.804841	0.825178
xgb_fclass_basic.csv 11 hours ago by Abdulellah N XGB model on F-class set using default parameters	0.791519	0.809937
mlp_sf.csv 11 hours ago by Abdulellah N MLP model on single feature performance set without using scaler	0.697812	0.724938
mlp_fi.csv 11 hours ago by Abdulellah N MLP model on feature importance set without using scaler	0.507224	0.510443
mlp_rfe.csv 11 hours ago by Abdulellah N MLP model on RFE set without using scaler	0.527543	0.544621
mlp_mi.csv 11 hours ago by Abdulellah N MLP model on mutual information set without using scaler	0.731126	0.755266
mlp_fclass.csv 11 hours ago by Abdulellah N MLP model on F-class set without using scaler	0.760479	0.786103

And after seeing the results, I settled with the set the I acquired through the feature importance method.

Benchmark

The benchmark proposed in the proposal was a Random Forest Classifier. The reason why I chose this model as a benchmark is that this algorithm is really powerful and can give more than decent result without much tuning. Also, this choice was made in attempt to make the problem a bit more challenging.

The score that the Random Forest classifier achieved:



III. Methodology

Data Preprocessing

As mentioned in the Exploration section, one of the first things I did was investigate the abnormal values in attribute ‘var3’. One technique that I found online was to use the rest of the data to try and predict the abnormal values as if they were missing. But after looking at the other attributes, I’ve found that there are many columns with large values, so I assumed that the these values in ‘var3’ won’t affect that training.
Then I looked at the constant columns, and removed them from the dataset. And did the same for the duplicate columns.

Then proceeded to one of the most exhausting things in this project, feature selection. I performed five methods for feature selection to try and reach the most effective features.

Implementation

Most of the implementation was made easier thanks to Scikit-learn, Pandas, and Numpy libraries.
Firstly, I started with loading the data, and taking a peak to the first few instances, the use the describe function which helped me see the extreme values in attribute ‘var3’.
After that I started the feature selection process, where, in the filter methods, I used the SelectKBest module the Scikit-learn provides. I also used the RFE module that Scikit-learn also provides. Using the extra trees classifier that’s implemented in the ExtraTreesClassifier module in Scikit-learn I obtained the feature importance set of features.

Then I proceeded with training models to choose the best set of feature. I trained an SVM using the SVC class, then measured it’s performance using the evaluation metric, AUROC, in the module which is, also, implemented in Scikit-learn library. I also imported the XGBoost library and it’s Python API, along with the XGBoost wrapper provided form Scikit-learn to help dealing with XGBoost easier.

At the last step, I used the GridSearchCV class to grid search the final model’s parameters to find the best performing set of parameters.

Refinement

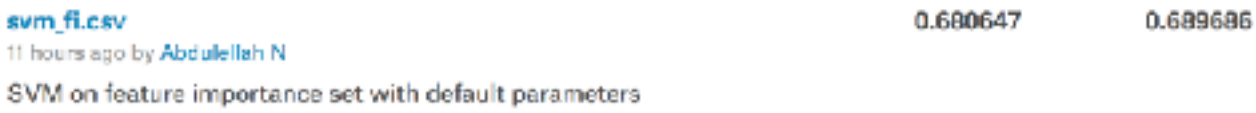
The most significant refinement happened to the final model (XGBoost) using the GridSearchCV class from Scikit-library. It’s an implementation of the grid search algorithm the also utilizes a cross validation to improve the accuracy of the choice of parameters. The cross validation was performed using StratifiedKFold object that’s from Scikit-learn library.

IV. Results

Model Evaluation and Validation

After finishing the feature selection process, and choosing one set to continue with I started running my selection of chosen models and comparing their performance.

First I ran an SVM:



Then I ran the Neural Network:

After that I ran the XGBoost model:
And it's clear, it blows the other two model out of the water. This

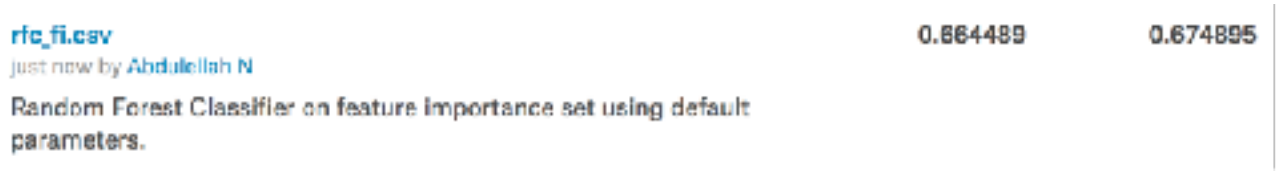
made me focus on this as model and fine tune it to reach my final model.

To tune the hyper parameters, I chose GridSearchCV class from Scikit-learn. After applying the best parameters that resulted from the exhaustive search the model improve in score:

And, as it can be seen, the model's score improved, but not a dramatic improvement. Yet, that much improvement at that scale is really good.
As the score is from predicting totally unseen data, I think it's safe to assume that this model's predictions are reliable.

Justification

The final model is significantly better performing than the benchmark model.
The benchmarks score:

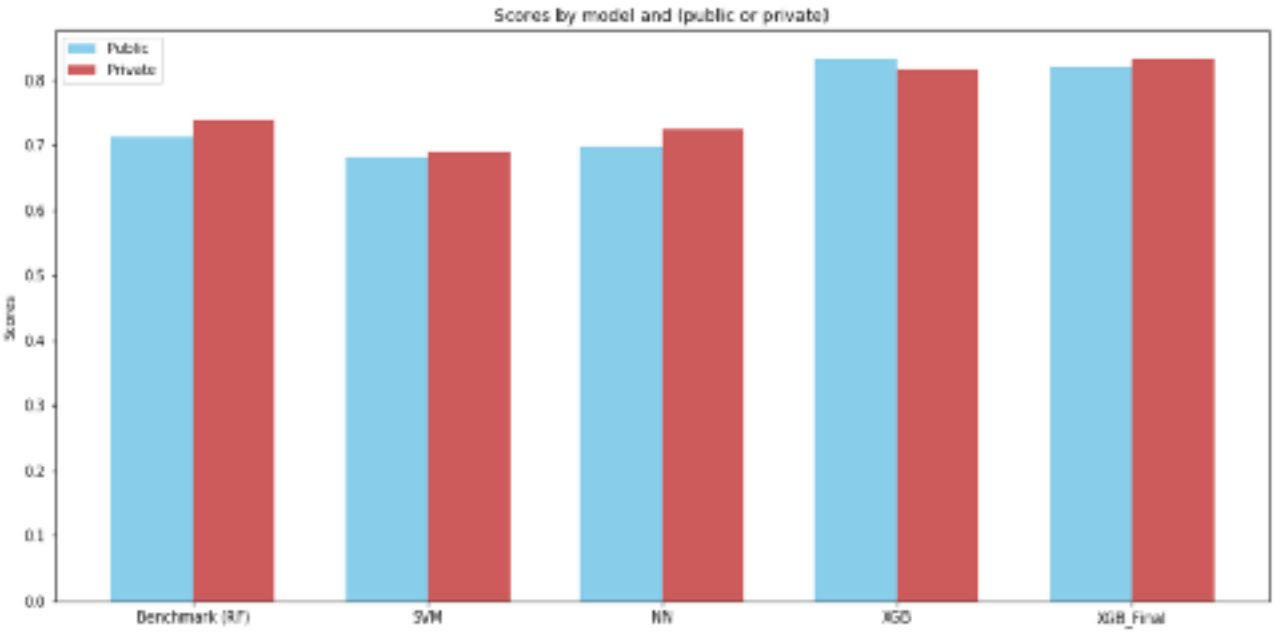


And the final model's score:

The final model clearly out performs the benchmark. The result are as what I envisioned at the beginning of the project; a model that performs better than the benchmark model.

V. Conclusion

Free-Form Visualization



This is a graph of the scores that the models achieved, and it shows how ‘mind-blowingly’ good XGBoost is. Even without any tuning it outperforms all the models in this project, and can achieve state-of-the-art results. No wonder it’s the go-to model on Kaggle.

Reflection

In this project, I developed a classifier that is able to predict whether a customer is satisfied with the banking experience with his bank or not. The huge number of attributes formed a challenge to deal with, especially with the attributes’ names being cryptic, and couldn’t derive much information by looking at the data. And performing visualization on the entire data just proved time consuming, that just skipping to feature selection methods seemed

like a better choice. That was difficult and exciting to deal with at the same time.
At the end, the model achieved a really good score (given that the winning scores were, public: 0.845325, private: 0.829072). I think that due to XGBoost's ability to generalize, this model would perform well in a general setting.

Improvement

One improvement that can be done to the model is performing a larger, and more thorough grid search on the parameters. Also, Bayesian Optimization can be used in the grid search step, and could improve the results even more.
The data can be explored better, and there can be attempts to decrypt the meaning of the names of the attributes to understand what they represent, and deal with the data better than being blind to what the attributes represent and how they relate to each other.
Applying these two improvements, I believe, would result in an even better and more accurate model than the model that I've reached.