Machine Learning Engineer Nanodegree

Capstone Project
Gabriel dos Reis Estivalet
October 19th, 2018

I. Definition

Project Overview

This project utilizes Home Credit Group's data made available on Kaggle with the purpose to make predictions about the customers' abilities to repay a loan. With a better assessment of the financial capacity of a given client and whether he or she will potentially incur in risk of default, the company is able to make more accurate decisions regarding the approval of a credit request. Therefore, the final objective of this project is to predict whether a customer will default or not.

With that in mind, the data consisted of several files with data regarding credit card balances, other financial obligations from the credit bureau, cash and POS (point-of-sale) transactions, previous loan applications, installment history and other customer information, such as demographic and some behavioral data. As a financial institution who operates in the personal loan industry, it is paramount for Home Credit to be able to find the right requirements and analytical criteria to identify whose customers provide an increased risk of default without incurring in the risk of denying loans to customers that could normally repay all their credits. In other words, if the company is too lenient it will likely grant credit to risky customers. On the other hand, if it is too strict, it will incur in opportunity costs from denying loans to potentially good customers.

For that matter, it is critical for Home Credit to find an efficient way to identify these two groups of individuals. And to assess how this task could be approached in a real world scenario, this project aims to create a supervised machine learning technique, trained on different representations of a historical dataset of customer information in order to make predictions for new-unseen data. For simplification purposes, only the main customer table will be used[1].

Problem Statement

As aforementioned, the main objective is to identify the customers that are in risk of default so that the business can make informed decisions when analyzing a credit application.
In order to solve the problem, a tree-based classifier model will be trained using different representations of the data.

[1] **Data available at** https://www.kaggle.com/c/home-credit-default-risk/data. **Table used:** "application_train.csv"

In that sense, techniques such as PCA (Principal Component Analysis)[2] and automatic feature selection from scikit-learn will be used to transform the original dataset so that the same classifier can be applied to each new dataset so that the results can be compared and an assessment of which representation will provide the algorithm with the most informative features can be made.

Finally, to analyze which transformations (if any) provide better results, a random forest classifier[3] will be trained several times using the ROC curve as a way to evaluate performance. The model that most precisely identifies which customers have a risk of default will be deemed the higher performer.

Metrics

The main metric that will be used to assess the models' performances will be the ROC (Receiving Operating Characteristic) Curve[4]. More precisely, the area under the ROC curve (AUC) is the concise metric that will be calculated, whilst the ROC curve is a plot of the TPR (True Positive Rate, the rate of actual positive cases that are correctly classified by the model as being positive) by the FPR (False Positive Rate, the rate of negative cases that are wrongly classified as positive) at all possible thresholds.

Once the ROC Curve is calculated and plotted, the area under it (named AUC score) can be used as a representation of how well the model can distinguish both risky from non-risky clients. This metric is the most relevant to evaluate the model's performance, as it assesses how many risky customers the model can correctly identify and how many it wrongly classifies as risky. These metrics can clearly inform business decisions in this scenario, as the main goal is to identify non-risky customers so that opportunity costs of not approving them a loan can be reduced and the company can make, therefore, more accurate decisions without incurring in the risk of serving customers with a high chance of defaulting in their obligations.

In addition to the AUC, precision, recall , f1-score and the confusion matrix will also serve as basis for evaluating the classifier's performance[5]. Precision measures the rate of truly positive cases in face of all positive predictions, while recall evaluates the ratio of correct positive predictions in comparison with all truly positive cases. The f1-score is a weighted average of both the previous metrics. It is also called a harmonic mean of precision and recall. Finally, the confusion matrix displays both true and predicted classes for all the cases in analysis, which can further illustrate the power of the classifier. This last one is also crucial for evaluating how well the model performs, as it presents actual quantities of predicted versus real classes.

[2] Müller & Guido, Introduction to Machine Learning with Python - O'Reilly, USA. 2017. Pg. 140.

[3] Müller & Guido, Introduction to Machine Learning with Python - O'Reilly, USA. 2017. Pg. 83.

[4] Müller & Guido, Introduction to Machine Learning with Python - O'Reilly, USA. 2017. Pg. 292.

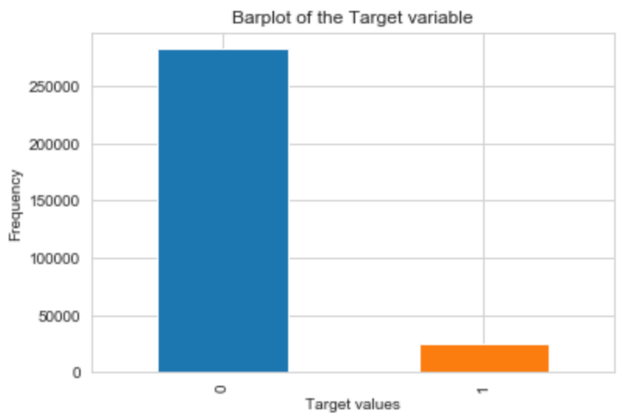
[5] Müller & Guido, Introduction to Machine Learning with Python - O'Reilly, USA. 2017. Pg. 279.

II. Analysis

1. Data Exploration

1.a) Overview

The data was made available through Home Credit Group's Kaggle challenge[6] that has ended on August, 2018. As mentioned previously, for simplicity only the "**application_train.csv**" file will be used in the current project.



The dataset contains 122 columns and 307,511 rows with a mix of continuous and categorical features. The target is also included and identifies the individuals that have defaulted with a value of "1" in contrast with the ones that have paid back their loans, which are marked as "0". However, it is heavily imbalanced, whereas merely 8% of the cases are positive (defaulted).

For simplification purposes, only a few characteristics are going to be discussed further in the report. For a full understanding of the dataset, all of the descriptive statistics and other special information are included in the "data_description.csv" file.

1.b) Catgorical Features

The categorical features present demographic characteristics of the applicants, such as family status, housing type, occupation type, has a vehicle, etc., as well as information regarding the nature of the loan, such as contract type (cash loan, revolving loan, etc.), weekday of application start (Monday, Tuesday, etc.) and documents provided (which are indicated by binary flags). There are many characteristics covered in the 45 categorical features, and they are concerned with the applicant, his/hers assets and contract details.

Regarding the distributions of these features, it is not possible to spot a feature that clearly separated the classes, as in general the proportions between positive and negative cases is maintained. This can be seen when looking at the count plots from the categorical features in the EDA_Preprocessing notebook.

[6]Data available at: https://www.kaggle.com/c/home-credit-default-risk/data

1.c) Continuous Features

Regarding the 84 continuous features, the information presented also range from demographics to behavioral and contract details. For example, some features include the number of days since the customer has changed phones, the total area of his/her house (if any), days since employment start, amount of the credit and annuity, etc. It is also important to note that these features have been already normalized with what seems to be an equivalent technique as implemented by the MinMaxScaler from scikit-learn[7], where the minimum value of the feature is subtracted from each value and the result is then divided by the range (maximun value minus the minimum one). The indications that this could have been the technique used is that many continuous features have values between 0 and 1, which is exactly the output one would expect when applying MinMaxScaler. In addition, Many features here are highly skewed to the right, as most values are closer to 0. On the other hand, some have bimodal distributions and are not skewed.

In contrast to the categorical features, when analyzing the discriminatory power of the continuous ones, we can attest that there are features that will very likely rank high on the model's feature importance. For example, features such as days_birth, and ext_source_1 (_2 and _3 also) stand out with regards with their ability of distinguishing the classes. These features encode the information regarding the number of days since birth at the time of the application, and normalized scores from external data sources (1, 2 and 3) respectively. These last 3 features do not present a more precise description of the information they possess.

1.d) Missing Values

When looking for missing values, it is possible to see that 70 columns have at least 1 missing entry. While there are columns that have almost 70% of the data missing, others have less than 1%. Since the model that will be applied is a random forest classifier, columns with missing values will have to either be imputed or dropped.

1.e) ECDF (Empirical Cumulative Distribution Functions)

Regarding the ECDFs, most features present a wide range of value occurrences in the first 30-40% of the cases. In such circumstances are valuable features such as ext_source_1. Some features, though, are more proportionally distributed. For example, basement_area_mode, which has the normalized information about the building in which the client lives. Finally, there are features that present the majority of occurrences concentrated in values close to zero. This is the case for ext_source_3, which has over 80% of all values close to zero. The other 15% are distributed in the range from 1 to 4.

1.e) Correlations (pearson's)

When one looks at the correlations between the features, it's possible to conclude that most features are not correlated. Some, however, present correlation degrees of above 80%, which is the case for features such as number of children and family size, as well as total living area

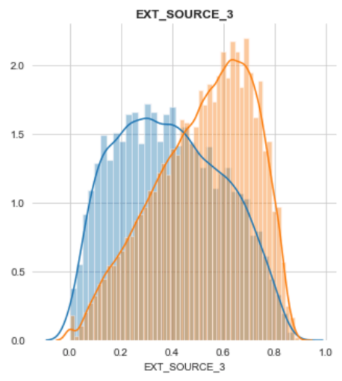
[7] Müller & Guido, Introduction to Machine Learning with Python - O'Reilly, USA. 2017. Pg. 134.

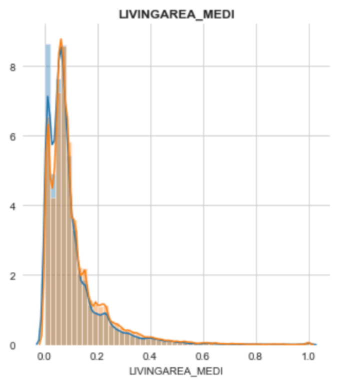
and areas of the house. These correlations are very logical, because some features already include the information presented in the others. For instance, the more children a person has, the bigger his/her family will be.

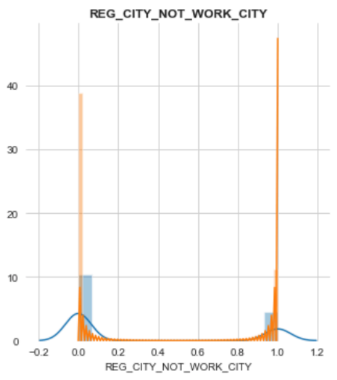
More importantly, perhaps, it the correlation of each feature with the target variable. In this scenario, the correlation coefficients varies from roughly 8% to -18%. The variables in these extremes are days_birth and ext_source_3, which were already identified in the feature distributions above. Once again, it is getting clearer that these features encode higher discriminatory power than the rest of the dataset and will very likely rank high on the classifier's feature importance.

Exploratory Visualization

To illustrate some of the variable's descriptions above, examples of features distributions are:

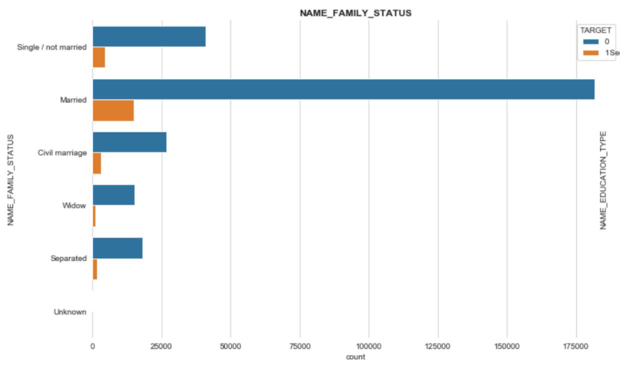


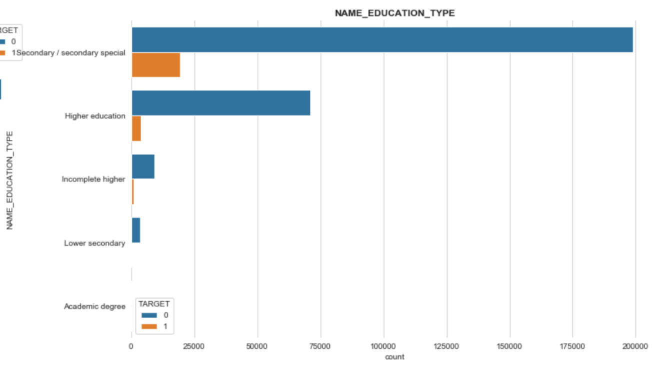




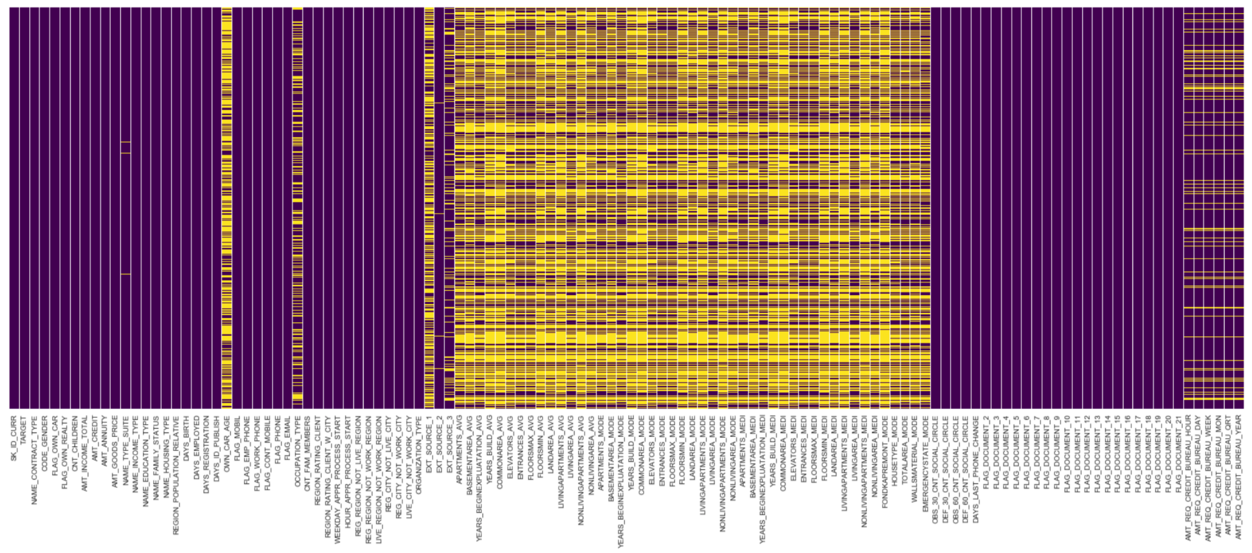
Above we can see the positive cases identified in orange and the negative ones in blue. The plot on the left shows the variable ext_source_3, which has a great discriminatory power as it can better separate positive and negative cases. The one in the middle shows the median living area and it is skewed to the right. Finally, the plot on the right is of a categorical feature that encodes the information regarding whether a client's permanent address does not match his/hers work address. In this case, 1 does not match and 0 matches.

When looking at count plots of the initially identified categorical variables, in general they do not portrait any clearly distinguishable patter as some of the continuous features. Below are the name_family_status and name_education_type:



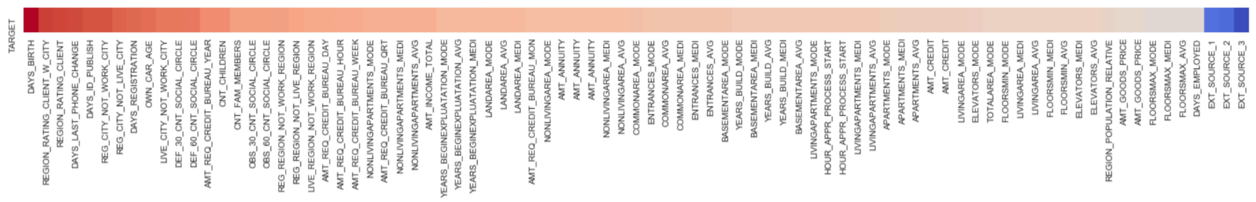


Another interesting plot is a heat map of all the missing values in the data. Despite the fact that it does not tell you precisely how many missing values you have in every column, it can paint a very complete picture of the overall, say, "completeness" of the dataset. See below:



In the picture above, the purple values represent filled fields, whereas the yellow ones represent missing values. This way we can observe that the majority of features are either very incomplete or fully complete. Some have a smaller percentage of missing values.

Another important plot is the correlation one. For simplification, only the correlation of each continuous feature with the target will be displayed in the image below:



On the left, with the darkest shad of red, there is the highest positively correlated feature, which is days_birth. This variable have a roughly 8% positive correlation with the target variable. On the other hand, the ext_source_3 at the right most corner of the plot presents a negative correlation of around -18% with the target. These features are likely to be considered important once the algorithm has a change to analyze them.

More details and other plots of the exploratory data analysis are included in the jupyter notebooks.

Algorithms and Techniques

The algorithm that is going to be used is a random forest classifier from the scikit-learn python module. This algorithm will be fitted to the data in combination with the RandomizedSearchCV, which will fit the model several times with different parameter settings. These parameters will be picked at random from a given range of values, in which the best performing setting according to the ROC-AUC metric will be chosen.

As this algorithm is based on an ensemble of trees, the parameter setting experimented will be:

- max_depth: [3, 5, 9, 13],
- max_features: range(5,21,3),
- min_samples_split: range(4,20,4),
- min_samples_leaf: range(4,20,4),
- bootstrap: [True, False],
- criterion: ["gini", "entropy"]}

Every parameter has a say on how the model will better fit the data, but given the nature of the algorithm, some carry a heavier weight. Max_depth and max_features are two examples. The first one will limit how many splits every tree in the ensemble can have. The values range from 3 to 13 as described above. Max_features represents how many features are randomly selected at each split from which the algorithm can pick the "best one", meaning the one that better separates the data between classes. In this sense, the model can only pick a feature from these randomly selected features at each split.

Furthermore, the "best feature" at each split can be quantified through two criteria: gini or entripy. These two methods can be suitable to select which variable better separates classes as stated above.

The model will always be applied in the same format, with the same parameters and with RandomizedSearchCV approach. The difference will be the representation of the dataset at each training. In this sense, four models will be fitted in the data with or without transformations. The four scenarios follow below:

- original features with no transformations
- original continuous features and automated feature selection for the categorical ones (SelectFromModel)
- Robust Scaler with PCA for continuous features and automated feature selection for the categorical ones (SelectFromModel)
- Quantile Transform with PCA for continuous features and automated feature selection for the categorical ones (SelectFromModel)

The PCA transformation is an unsupervised transformation that aims to find principal components that capture most of the data's variation. In general, they are used either for visualization purposes or to be used as a dimensionality reduction method that can be applied during a preprocessing step of a supervised model approach. The number of components can vary from 1 to the number of features present in the data and each component can quantify how much of the variability it captures. This is important as each component is a hybrid feature that will replace the original ones.

As PCA utilizes a distance metric to calculate the new components, it is not really suitable for categorical features. For that matter, the automated feature selection will be applied to categorical features. This method uses the feature importance of the model that is fitted to the data and presents a level of how important each variable is to distinguish the classes of the target. This way, a simpler version of the random forest classifier will be run, as this model provides a feature importance capability.

The four models will be trained on different representations of the data and their results will be compared to see which performs better according to the ROC-AUC metric.

Benchmark

For comparison purposes, the first model that will be trained will serve as a benchmark to evaluate if any of the transformations applied to the data will result in a better final performance. This way, this model - Model 1, will use no further transformations after the initial ones, which comprehend balancing the data, imputing missing values and creating dummy variables. These transformations will be applied to all models, but some will have further modifications to reduce the data's dimensionality. All of these steps will be further explained in the following sections.

III. Methodology

Data Preprocessing

a) Balancing the data

Initially, as the data is highly imbalanced between positive and negative samples, meaning the the positive (default cases) are very rare compared to the negative ones, an under-sampling method will be adopted for the negative targets.

The original data has 307,511 rows, from which only 24,825 are positive. With a 20% random under-sampling of the negative cases, the data presents a new distribution of 30% positive and 70% negative cases. This new proportion is going to give the algorithm a better chance of correctly identifying both classes.

b) Imputation

As many algorithms in the scikit-learn python module, random forests can not handle missing values. For that matter, instead of dropping the columns with missing values, these values were imputed with the mode, or the most common value in each column. Since the algorithm is a tree-based model, the mode will not potentially create a new value, in contrast with mean and mode), but despite the fact of being a very simple strategy, it won't influence the model. In the worst case, the trees may consider a feature that had many missing values as unimportant as it potentially won't be able to distinguish the classes.

c) Dummy Variables

Finally, the initial preprocessing was concluded with the transformation of the categorical variables for dummy ones. This step takes a categorical feature with, say, 3 distinct values and replaces it for 3-1 (three minus one) binary columns. This is important for a classifier to be able to handle this data properly.

Implementation

Once tha data was balanced and the missing values were filled, the data was further treated depending on the strategy that was going to be applied. As mentioned above, the strategies range from different standardization techniques, the use of PCA and automated feature selection to not performing any of these transformations at all.

After any transformation, the model described in the Algorithm section of this report was used with no alterations from model to model. This approach was utilized so comparable results could be produced. Therefore, four models were created in the following manner:

a) Model 1 - Benchmark: original dataset

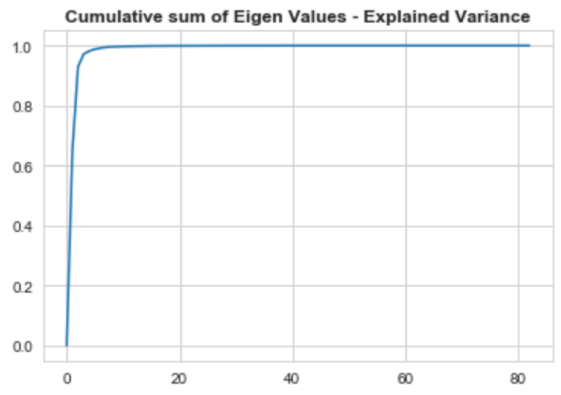
As it is stated in the title, after balancing, imputing the data and creating dummy variables for the categorical features, no further transformations were applied and the model was trained. This model is going to serve as the benchmark.

b) Model 2: RobustScaler + PCA + SelectFromModel

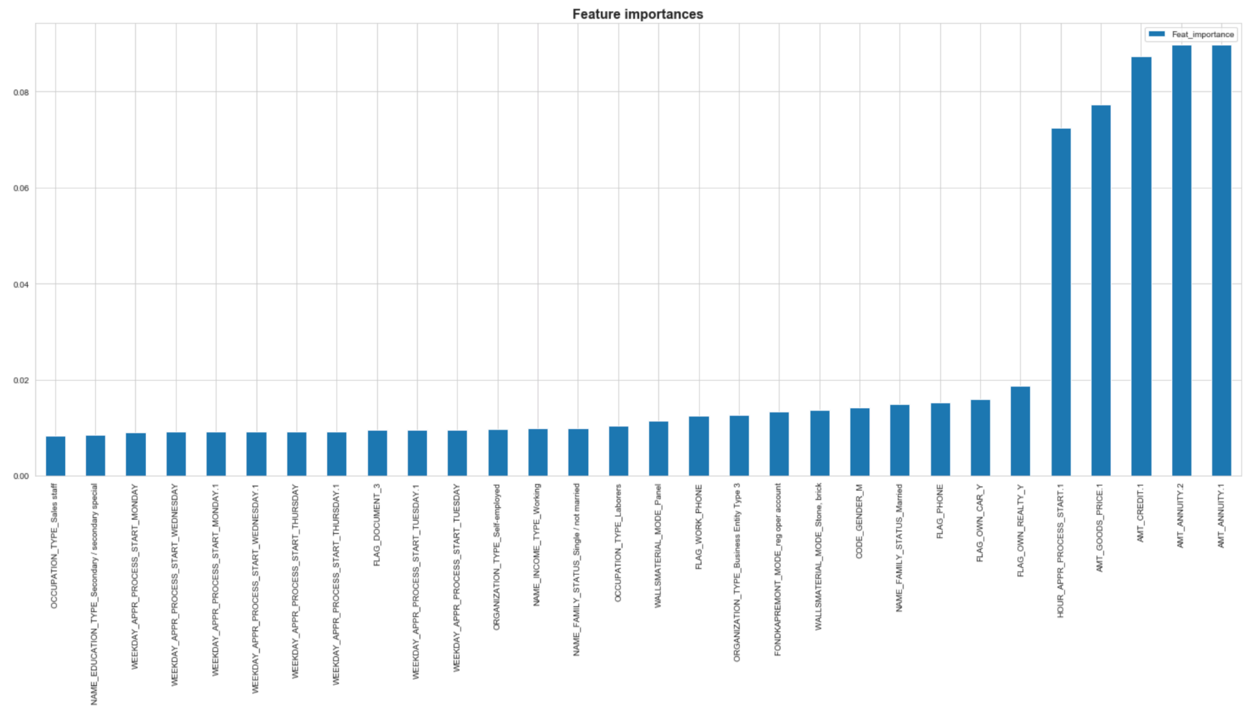
Following balancing and imputing the data, the RobustScaler was applied to the continuous features so all are in a similar range and PCA could be applied to the data. Needless to say, these 2 methods were only used in the continuous features.

RobustScaler used the median and quartiles of the columns to standardize them, while maintaining the statistical their properties. This way, this method tends to ignore data points that are very different from the rest, which can be measurement errors, outliers or simply extreme values.

Once all the features were in a similar scale, PCA was initially trained to produce as many components as there were continuous features. With the new representation, a mere 3 components were able to collect about 97% of the variability of the continuous features. This can be seen in the cumulative explained variance plot above. The y axis show the percentage of variability and the x one displays the number of components required to capture the corresponding percentage of variability.



As for the categorical variables, a random forest was trained so that its feature importances could be examined. It turns out that this technique corroborates the exploratory data analysis, when it showed that there weren't many variables that could clearly distinguish the classes. The top-30 features with regards to their respective importances are displayed below:



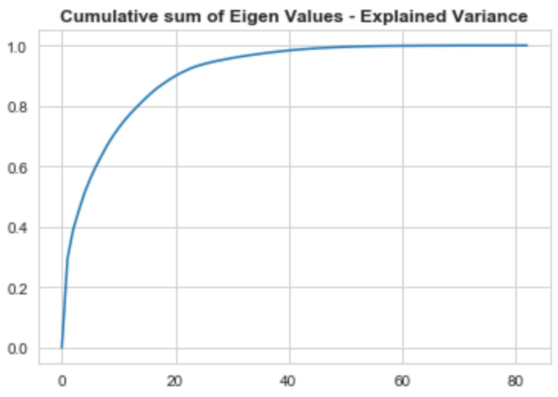
Once this was confirmed, the SelectFromModel automatic feature selection was used in combination, again, with a random forest classifier to select the most informative ones. For that purpose, a threshold of 1.5 times the median of all the importances was used, reducing the dataset from the original 168 categorical columns (dummies from the original 45 categorical variables) to 68. Therefore, the 100 least important variables according to the model used were dropped, greatly reducing the dimensions of the dataset.

Important to note that this exact same selection of the categorical features was also applied to model's 3 and 4.

c) Model 3: quantile_transform + PCA + SelectFromModel

In comparison with model 2's transformations, model 3 differs with regards to the standardization applied to the continuous features. Here the quantile_transform was used to make the variables resemble a normal distribution, with mean 0 and standard deviation 1.

Once PCA was applied, it was necessary to select the first 36 components in order to have the same 97% of captured variability as the RobustScaler used in model 2 provided. The cumulative sum of the explained variances obtained in this model can be seen in the plot on the right.



The categorical features were used exactly as it was done in model 2: only 68 of the original 168 were selected.

d) Model 4: original continuous features + SelectFromModel

Finally, the last model used the original representation of the continuous variables in combination with only the 68 most important categorical ones according to the feature importance and SelectFromModel techniques explained above.

Refinement

For selecting the best parameters, all models were fitted to the data using the RandomizedSearchCV, which searches for random combinations of the parameter setting in order to find the ones that will yield the best performance for the model. This technique is highly utilized for supervised algorithms and it presents results much faster than the somewhat similar GridSearchCV, which searches all possible combinations of the parameter settings. Naturally, this last one can take considerably more time to run and, in general, presents slightly higher performances.

IV. Results

Model Evaluation and Validation

The final results produced by all the four models trained were somewhat similar. Evidentially, some performed better due to the representation fo the features used as inputs. It is important to highlight, though, that even though Randomized Search CV was used so that each model could separately search for the best parameter settings, all four models return the exact same parameters. This shows that there is clearly an optimal hyper parameter setting and this will eliminate other explanations as why each model's performance could vary, as the only difference between them is the representation of the data that was used. See the optimal parameters below:

Algorithm	bootstrap	class_weight	criterion	max_depth	max_features
Random Forest Classifier	TRUE	None	entropy	13	14

max_leaf_nodes	min_impurity_decrease	min_impurity_split	min_samples_leaf	min_samples_split	min_weight_fraction_leaf
None	0	None	16	4	0

n_estimators	n_jobs	oob_score	random_state	verbose	warm_start
1000	1	FALSE	None	0	FALSE

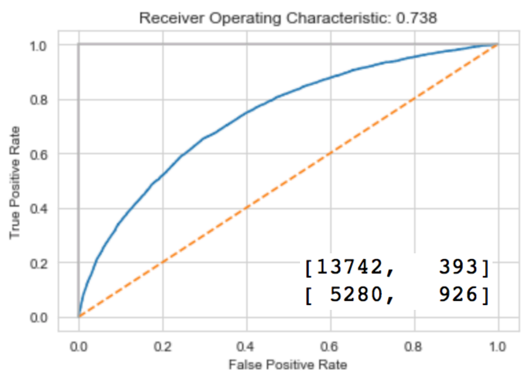
The max_depth parameter, which regulates how deep each tree can be presents a value of 13, which is somewhat high. This means that each tree had to ask 13 questions (the maximum allowed in the parameter search range) in order to do a reasonable job in separating the classes. This could indicate, perhaps, that if a higher value was allowed for this setting, the trees could have grown even deeper.

Another important parameter is max_features, which represents how many randomly picked features each tree could use to split the data in each of its nodes. In general, it is extremely difficult to predict before hand how these parameters will be, but the results often tell a lot about how complex is the problem.

As it will be shown in the next sections, the models generalize well to new-unseen data. Some, naturally, perform better due to the representation of the features. However, in general they all do a relatively good job in separating the classes. This is important, specially due to the minimalistic approach in terms of feature engineering applied. Moreover, around 81 thousand samples were used between training and testing sets, which presents the models with enough samples to have a good understanding of the dataset.

Below we can visualize the ROC Curves and AUC scores for all the four models:

a) Model 1 results: original dataset



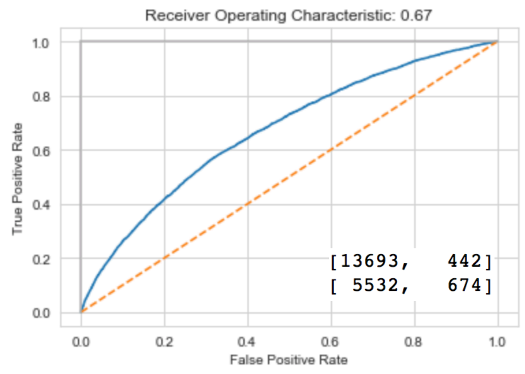
```
[13742,    393]
[ 5280,    926]
```

```
              precision    recall  f1-score   support

          0       0.72      0.97      0.83     14135
          1       0.70      0.15      0.25      6206

avg / total       0.72      0.72      0.65     20341
```

This model presented a performance of 73.8% according to the AUC score. Inspecting the classification report on the right, it is also possible to see that was able to match both precision and recall’s performances at an even 72%, meaning that it correctly identified 72% of the positive cases at the same time that from the total predicted positive cases, 72% of them were in fact positive. The actual classification results can be seen in the confusion matrix that is included in the ROC plot.

According to the scikit-learn documentation, the value on coordinate 0x0 represents the true negative cases; 0x1 the false positives; 1x0 the false negatives; and 1x1 the true positives[8].

b) Model 2: RobustScaler + PCA + SelectFromModel



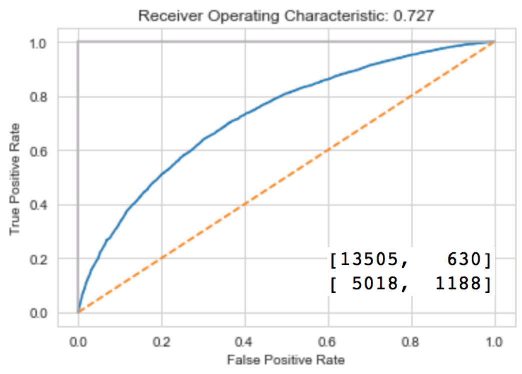
```
[13693,   442]
[ 5532,   674]
```

```
             precision    recall  f1-score   support

          0       0.71      0.97      0.82     14135
          1       0.60      0.11      0.18      6206

avg / total       0.68      0.71      0.63     20341
```

Model 2 presented a lower performance, with a 67% AUC score. This model only used 3 of the principal components and, thus, it is not a surprise that some aspects of the data were not captured in detail enough so that the classifier could use them.

Inspecting the classification report on the right, it is also possible to see that it boosts a recall performance that is somewhat superior to both precision and the f1-score, meaning that it correctly identified 71% of the positive cases.

c) Model 3: quantile_transform + PCA + SelectFromModel

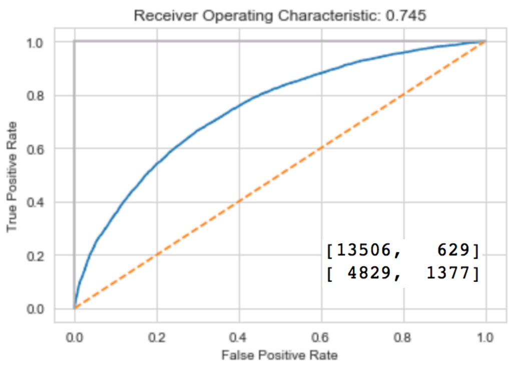


```
[13505,   630]
[ 5018,  1188]
```

```
             precision    recall  f1-score   support

          0       0.73      0.96      0.83     14135
          1       0.65      0.19      0.30      6206

avg / total       0.71      0.72      0.67     20341
```

Model 3's performance was almost as high as the model 1's according to the AUC score. However, the total corrected predictions surpass model 1's by about 25 samples, which could be represented in the precision score of the classification report. Overall, though, the performances of both these models are very similar.

d) Model 4: original continuous features + SelectFromModel



```
[13506,   629]
[ 4829,  1377]
```

```
             precision    recall  f1-score   support

          0       0.74      0.96      0.83     14135
          1       0.69      0.22      0.34      6206

avg / total       0.72      0.73      0.68     20341
```

Model 4 was the winner, with a 74.5% performance according to the AUC score. Moreover, recall and the f1-score are both slightly higher when compared to the other models. Finally, this increase in performance can be seen in the confusion matrix, where it correctly classified 14.883 samples, outperforming model 3 by 190 correct predictions.

Justification

As described in the previous section, model 4's performance is slightly better than the benchmark's. This can be quite striking, given the fact that the best model uses 100 features less than the benchmark, which was trained in the original dataset with the dummies for the categorical features. It would be logical to think that with less features, the model would have access to less information, but if certain features do not have discriminating power, they can decrease the effectiveness of a tree-based model such as the one used[9].

One of the reasons why this can be the case is because of the max_feature parameter of the classifier, which randomly select x many features at each split for the algorithm to select the one and the threshold that better separates the data. For the benchmark model, 251 features were used in total and, since there were many more and perhaps uninformative features for the task at hand in comparison with model 4's 151 features, many of the splits in the trees could

[9] Müller & Guido, Introduction to Machine Learning with Python - O'Reilly, USA. 2017. Pg. 84.

have potentially randomly picked several of these to look for a split. This would result in the model to try to find other explanations to separate the classes, apart from using the most obvious ones, which could have likely produced very impure nodes.

On the other hand, model 4 was trained on a dataset that had 100 of the least informative categorical features removed. Therefore, it is very likely that the randomly selected features in each split had more discriminating power. For that matter, this could very well justify the difference in performance between the models.

Overall, the final solution achieved with model 4 is robust enough to enhance business decisions. However, there are many False Negatives, which represents all the risky cases where people would likely default. In a scenario where the objective is to exploit opportunity costs without risking the business, false negatives can have a devastating effect. Therefore, more work should be done in order to enhance the model's performance. However, considering that this was a rather simplified approach to use only the main table provided by the company, the results can't be ignored and they surely display the astounding capacity of machine learning to solve complex tasks such as detecting risky customers.

V. Conclusion

Regarding the objective to correctly identify the risky customers, the ones that had a high risk of defaulting, the model seem to perform reasonably well given the simplicity of the techniques and algorithms used. The transformations applied to the data are used across many industries in real-life machine learning tasks, but there is in general a higher degree of complexity specially concerning feature engineering. Nowadays, practical machine learning is mostly about it rather than tuning parameters and even the type of algorithm that will be used.

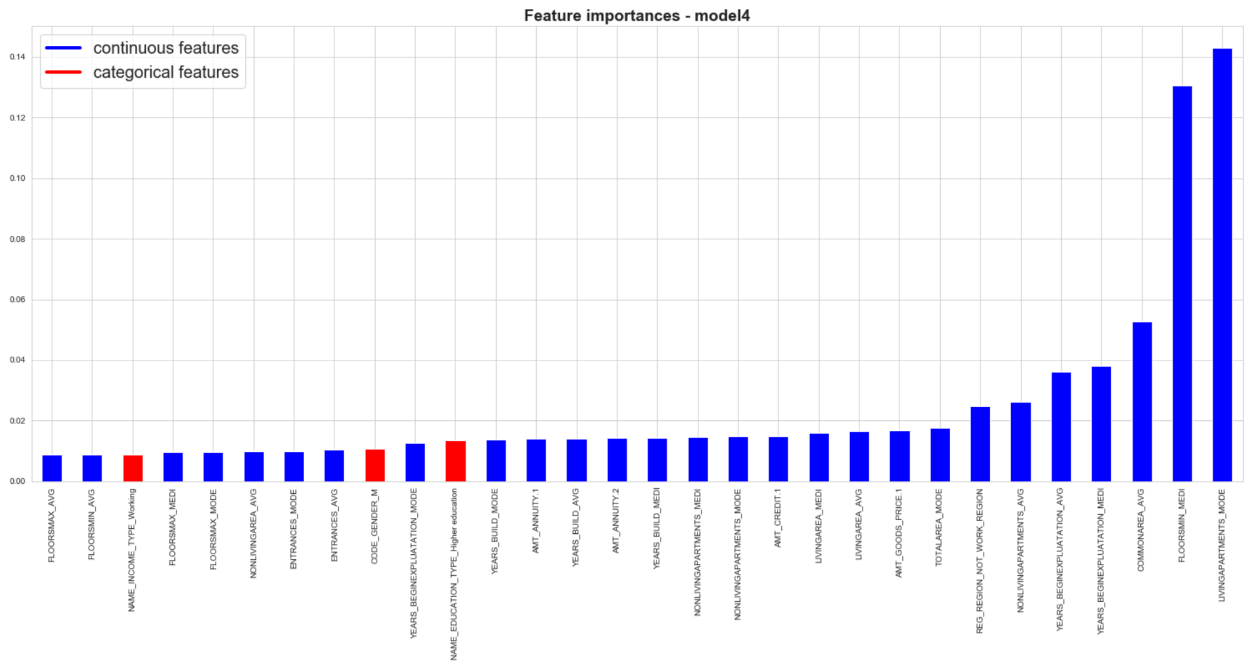
For the project, balancing the data to a 70%-30% split between negative and positive classes respectively allowed the algorithm to better identify the positive cases. This is true, as in a highly imbalanced dataset the model can sometimes focus too much on the negative cases and "miss" the important characteristics of the positive classes. Here, an under-sampling was performed to randomly select 20% of the negative samples from the original data.

Next, for the scikit-learn implementation of the Random Forest to work properly, no missing values can be fed into the classifier. For that matter, an impure was utilized with the simple strategy of filling the missing values per column with the mode (the most frequent value). This way it is not only possible to use the data for a supervised task, but also prevent information loss by dropping columns with missing values.

In addition to wrangling the data with the techniques described above, several plots were used to describe the dataset, such as distribution and countplots, ECDFs (Empirical Cumulative Distribution Function), correlation plots between the variables and between the variables and the target. All this insights were taken into account when deciding the best way to represent the information and apply a supervised model.

After the initial transformations, four (4) identical models were trained on different representations of the data. These representations were created by using a combination of PCA (Principal Component Analysis) and automatic feature selection (SelectFromModel with RandomForestClassifier). With the models fitted in different representations of the dataset, their respective performances was evaluated based on the results of the ROC Curve, confusion matrix and classification report. A scikit-learn implementation was used for all of these techniques.

Below are the feature importances according to the best performing model:



As it is possible to see that among the top-30 most important features according to model 4, only 3 are categorical. This corroborates the fact that most of the performance fluctuations among the models was evident when the continuous features were transformed using a scaler and PCA. It also indicates that most of the discriminating power of the features come from the continuous variables. Therefore, it is logical that model 4 performed better once the least informative categorical features were removed and all the continuous ones were used. This most certainly had an impact in every split of every tree due to the max_feature parameter.

Based on the results, we can see that feature engineering indeed is where most effort should be made, as different representations of the data can certainly impact the classifier's performance. The results are not surprising, as most data professionals in general spend most of their time organizing, cleaning and engineering features, rather than applying supervised techniques for example.

Reflections

One of the most remarkable aspects of results of the analysis is the power of techniques such as PCA. Even though the models that went through the process of dimensionality reduction did not perform as well as the others, the loss in the AUC score was of around 1,8% when observing model 2's results of 72,7% (model 4's performance was 74,5%). This performance was achieve using only three (3) principal components that were representing 84 continuous features. So, with a reduction of roughly 96,4% of the continuous features, the model only lost 1,8% of performance.

Improvement

As previously mentioned, the objective of this project was to compare how different representations of a dataset can influence the overall model's performance. For that matter, several relatively simple and common techniques were used, also with computing time in mind, as all the computations were ran in a hardware with limited capacity.

Nonetheless, the results were clear enough to demonstrate how feature engineering is nowadays one of the most important areas of practical applications of data science. It is true that the model should not be applied as it is in a real world scenario, since it has a high rate of false negatives, which would do the business perhaps more harm than good. Nonetheless, a performance of around 74% is quite astonishing for such a simple implementation of a supervised model.

In order to improve it, more of the data could be used, as more features would provide the classifier with mode information. In addition, these features would have to be more throughly analyzed and engineered in order to push their informative powers to the limit so that the model could make better predictions. To achieve this, different or the same set of techniques could be used, as they are already established industry standard for data science.