Matias Morales Armijo (z5216410) Pablo Toro Pacheco (z5222810) Sebastian Castillo Castro (z5171921)

COMP9417 – Machine Learning Project – 2020 T2

Home Credit Default Risk

Predicting how capable each applicant is of repaying a loan

Introduction

The number of unbanked populations who do not have access to a loan from established financial institutions is considerable, not being able for those people to obtain credits to overcome financial issues, or to invest in a business. Home Credit is a company that aims to bring unbanked people close to the option of obtaining a loan, based mainly on the analysis of the behaviour of payments of basic services of the clients, including the payment of Telco services and transactional information. For this reason, two years ago, Home Credit created a Machine Learning competition called "Home Credit Default Risk" on the Kaggle platform, on which this project is based. The objective here is to predict probability of client´s repayments, in other words, how likely is that a client will pay a loan on time.

In this project, rather than focusing on obtaining maximum performance with a single model using all the datasets available to the competition (seven datasets), it aims to generate, compare and discuss results of a set of machine learning models that will be trained on the combination of the two main datasets called Applications and Bureau. This project corresponds to a classification problem, therefore, the models considered were Logistic Regression, Decision Tree, Random Forest, Ada Boosting and Stacking.

Firstly, the implementation stage and partial results of each algorithm will be described, to later comment on the results consolidation, comparison, and discussion of these. This report will be finalized with the conclusions and possible future work.

Implementation

The work done in this project can be divided mainly in three sections:

1. Dataset preparation: Involves the process of:
    - Performing data cleansing and Exploratory analysis over dataset Applications and Bureau
    - Grouping the data of Bureau dataset where aggregated columns for each feature are created and merging the Applications dataset with the resulting Bureau dataset to produce the base dataset used for all the models.

- Generation of baseline mode (Linear Regression) using default parameters
- Implementation of approaches to improve baseline model:
    - Part 1. Combinations of Featuring engineering (feature creation) and PCA to train Logistic Regression models
    - Part2. hyper-parameters optimization in ensemble methods

A general overview of the implementation is shown in the following illustration:

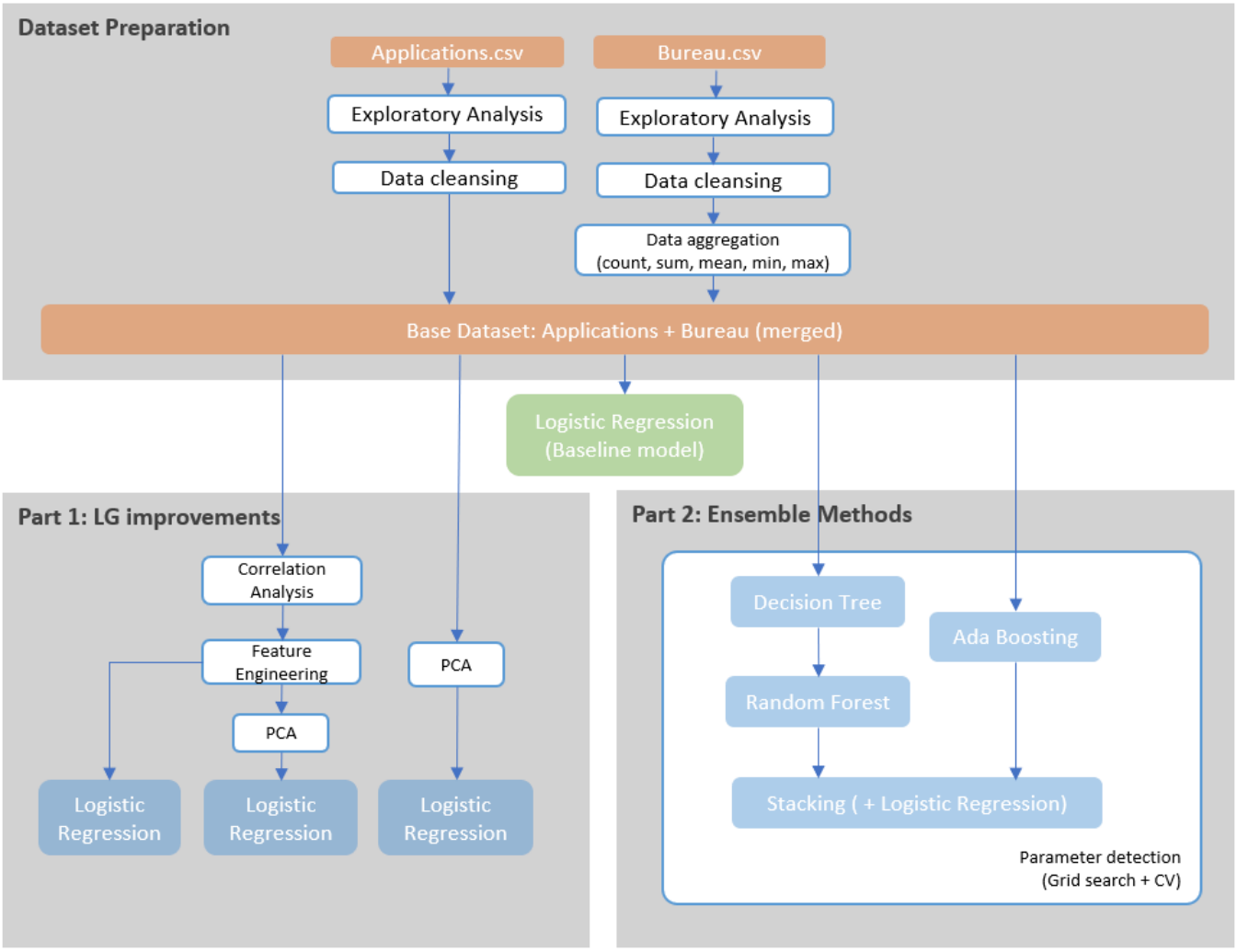


Fig. Solution Diagram

Dataset Preparation: Exploratory Analysis and Data Cleansing

Two of seven dataset available for this competition were used. The main dataset is called “Application.csv”, where each row represents one loan and target is a binary number with 0 if client will repay loan on time and 1 if client will have difficulty for repaying the loan. The second dataset is called “bureau.csv” which contains information about previous loans that client has from other institutions. Both files are related with the key “SK_ID_CURR” which is an ID of loan. About the number of registers, “application.csv” has 307,511 rows and 122 columns and “bureau.csv” it has 1,716,428 rows and 17 columns. To make the merge it was necessary create new columns for bureau dataset to have an analysis by observation. Statistical columns such as count, mean, sum, min and max were created by grouping the data on SK_ID_CURR column. Finally, the data was consolidated in 307,511 observations and 301 variables.

It was observed unbalanced data which has 282,686 observations with target 0 and 24,825 observations with target 1.

There are many techniques to analyse the missing values. One visual technique is display missing data with a heatmap. This visualization can show that there are many missing values in the data (figure 1, appendix). Analysing the percentage of missing values by column was possible to identify at least 35 variables have more than 50% missing

values and there are three top of variables which have 74% of missing values. This table shows the most critical variables:

Variable	% Missing Values
bureau_AMT_ANNUITY_mean	74%
bureau_AMT_ANNUITY_max	74%
bureau_AMT_ANNUITY_min	74%
COMMONAREA_MODE	70%
COMMONAREA_MEDI	70%
COMMONAREA_AVG	70%

After analysed the missing values by columns, it was necessary to also check the missing values per observation. A missing data histogram was able to expose the quantity of rows with missing values, where there are less than 5,000 observations with 0 missing values while other observations has missing values where more than 25,000 observations has 47 missing values (figure 2, appendix). Thus, to solve this problem, there are different alternatives like drop observations, drop features or make imputations in the missing values. To manage the missing values the decision was make imputations of the median values per variable instead of dropping the variable or observations.

The correlations were analysed because it is necessary identify which variables have high correlation with the target variable. The idea behind this is avoiding the high correlation between input variables because they show similar characteristics and depending of this some variables can be eliminated.

The procedure was calculate the correlation of each variable with the target where the top 3 columns with most positive correlations were bureau_DAYS_CREDIT_mean, DAYS_BIRTH and bureau_DAYS_CREDIT_min while the top 3 variables most negative correlations were EXT_SOURCE_3, EXT_SOURCE_2 and EXT_SOURCE_1. Checking the KDE density of this variables with target the graphs suggested that those variables could be important to get good separability (figure 5, appendix).

Top 3 Positive Correlations with target		Top 3 Negative Correlations with target	
bureau_DAYS_CREDIT_mean	0.089729	EXT_SOURCE_3	-0.178919
DAYS_BIRTH	0.078239	EXT_SOURCE_2	-0.160472
bureau_DAYS_CREDIT_min	0.075248	EXT_SOURCE_1	-0.155317

Other interesting analysis is calculating the correlation of each variable with every other variable. Thus, it is possible identify if there are highly collinear variables that could be removed from the data. To do this a threshold of 0.9 was defined taking any variables that have a greater than 0.9 correlation with other variables. The results detected 57 variables which at least have one variable highly correlated. For example, the variable bureau_DAYS_CREDIT_ENDDATE_count is highly correlated with 8 variables represented in a heatmap (figure 3, appendix).

An outlier is an observation numerically different from the rest of the data. Making revision of boxplot of variables is possible identified many outliers. One of these cases was DAYS_EMPLOYED where it had outliers with a

number in days higher than 350,000 days (figure 4, appendix). Then, a methodology to eliminate appropriate outliers was implemented.

The methodology used to detect outliers was IQR Score [4]. This rule says that any point which not in range of (Q1 – 1.5xIQR) and (Q3 + 1.5xIQR) is an outlier and can be removed. To create a margin of security a factor of 3 was implemented to determine the extreme outliers. However, instead of removing outliers, they were replaced by extreme whiskers (low whiskers: Q1 – 3xIQR, upper whiskers: Q1 + 3xIQR). This idea was getting after reading about Winsorizing [2]which replace the whiskers depending on the percentile. Thus, the outlier technique considered a mix between IQR-score and Winsorizing. The decision behind this is because the elimination data unknown it is not convenient when there is no knowledge of the business.

Once exploratory analysis and data cleansing was applied, One-hot-encoding technique was applied for categorical columns with minimum 3 classes in their values. For cases where the values of a column had 2 classes, Label-Encoding method was applied. The new resulting dataset was used as base dataset to train the baseline model based on Logistic Regression and as an input for the rest of models.

Evaluation of model performance: The prediction score ROC AUC

In this project, ROC_AUC was the chosen prediction score to compare performance between models. ROC_AUC score corresponds to the area under the ROC curve (receiver operating characteristic). Considering that false positive rate (FPR) is the proportion of negatives which were classified as positive and true positive rate (TPR) is the proportion of positives which were classified as positives, ROC corresponds to the curve of FPR vs TPR. Then ROC_AUC is the area under ROC curve, thus if it is considered 1 as credit default and 0 as good payment, when ROC_AUC is used the model is not going to sacrifice precision on classify credit default in order to get a good recall. Considering that the data is imbalanced, the majority class is 0, so it would be important having a good precision on class 1. This could be the reason behind in this Kaggle competition problem this evaluation is used.

Baseline model (Logistic Regression)

A Logistic Regression model with all the parameters with default values was used as a baseline model. This was trained over the base dataset with 300 features, reaching the following ROC_AUC over the test and training datasets:

Logistic Regression – Baseline model	
ROC_AUC testing	0.6399972708038846
ROC_AUC training	0.6419389283537762

It can be observed that this algorithm practically does not present overfitting during training process.

These results will be taken as a base reference or starting point to compare them with the results that will be obtained through the application on logistic regression of feature engineering techniques, dimensionality reduction (PCA) and the use of other machine learning models, which will be described in the following sections.

Part 1: Logistic Regression model + Feature Engineering + PCA

1. Featuring engineering techniques

The objective of this section is to describe the steps taken to create new features from the base dataset, in order to generate new data that could be correlated with the Target and thus improve the prediction (ROC_AUC). Considering that the team does not have domain knowledge about credit risk, to search for new candidate features, the following steps were performed:

- Creation of statistical features based on features present in Bureau dataset. This process was performed in data preparation step. The data here was aggregated by column SK_ID_CURR to merge the results with the corresponding row in Application dataset. The new statistical features were created by applying aggregation functions such as COUNT, SUM, MEAN, MIN, MAX. All this process can be checked in the function `create_statistical_columns()`
- Calculation of the Pearson correlation coefficient of each feature with respect to the Target (details can be found in section "Exploratory Analysis and Data Cleansing").
- Selection of the 6 variables most correlated with the Target which correspond to `'bureau_DAYS_CREDIT_min', 'DAYS_BIRTH', 'bureau_DAYS_CREDIT_mean', 'EXT_SOURCE_3', 'EXT_SOURCE_2', 'EXT_SOURCE_1'`
- Creation of new polynomial features using the `PolynomialFeatures` library from `sklearn.preprocessing` module based on columns selected in previous step. The polynomial degree used was 3 (4 increased too much).

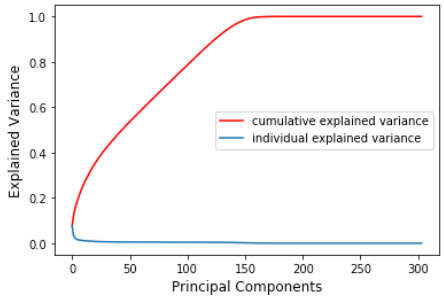
The new dataset contains 370 features (70 more thatn the base dataset). After training a logistic regression model over the dataset with the new features, the following results were obtained.

Logistic Regression – Feature Engineering	
ROC_AUC testing	0.7445525194632479
ROC_AUC training	0.7462086450710582

It represents a considerable improvement with respect to the baseline score.

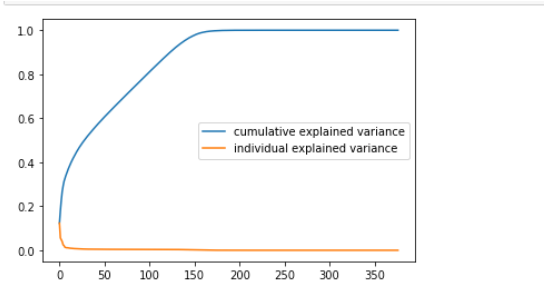
2. Logistic Regression + PCA

PCA is used to reduce dimensionality of data [4]. The data is grouped in principal components and after this iterate the components through which are enough to explain all the variance of the data. The data must be standardized to reduce the dimensionality. Then each component will be responsible of certain percentage of the variance. The figure of PCA represent the data merged, where, by simple inspection the variance can be explained since the 150 components. However, the chosen value was 170 components due to by using lower values the AUC_ROC results were slightly lower in logistic regression. That means using values less than 170 an important number of variables can be being eliminated. The following graph shows how the cumulative explained variance changes with respect to the number of components. It can be observed that about 170 components, the curve starts to stabilize.



It is interesting to detect that doing the PCA graph incorporating the feature engineering it is possible to get similar behavior. That means, using principal component with 170 components could be appropriate also. However testing with values higher than 170 was possible to observe an improvement in logistic

regression model until a moment where it was not possible to see any changes and this means that the new variables are not necessary to explain the variance.



The following results were produced after training Logistic Regression over the base dataset and applying PCA

Logistic Regression – PCA	
ROC_AUC testing	0.7420849567553675
ROC_AUC training	0.7433752357005943

It also represents a considerable improvement with respect to the baseline score.

3. Logistic Regression + Feature Engineering + PCA

PCA method was applied over the dataset after performing Feature Engineering (330 feature), reducing them to 212 components, optimal number found by analysing corresponding cumulative explained variance graph. The following results were produced:

Logistic Regression – Feature Engineering + PCA	
ROC_AUC testing	0.7474006715873159
ROC_AUC training	0.747648342996449

This was the highest AUC_ROC score achieved for Logistic Regression models.

Part 2: Hyper-parameter optimization in ensemble models

In this section it is going to be applied different ensembles methods[1], particularly Random Forest due to its simplicity to train and tune (Hastie et al, 2017) and AdaBoost which is another popular method (Geron, 2019). It is going to be shown how hyperparameters optimization can improve performance.

Random forests method is a modification of bagging, where the base estimators are trees and bootstrapping is apply on the features (the idea behind is taking the average of all the base estimators which implies variance reduction). According to Hastie et al[2] Trees are ideal candidates for bagging, since they can capture complex interactions and could have low bias. Therefore, it was trained a Decision Tree in order to construct a Random Forest based on it. As a result, the variance should be decreased and there should be an improvement in the performance.

AdaBoost is a boosting method based on the idea of combine many "weak" classifier in order to produce a powerful one (Hastie et al, 2017)[3]. Boosting method generally decrease bias. AdaBoost constructs in the final classifier in a sequential way: after every weak classification, the next one is focus on the previous mistakes.

[1] Scikit-learn methods were used to make these models: RandomForestClassifier, AdaBoostClassifier and StackingClassifier.
[2] The Elements of Statistical Learning, p.587
[3] P.337

Stacking is another ensemble method, the main idea behind is trained a model to ensemble the predictions of different base estimators instead of use a function (Geron, 2019). So, it is used some base classifiers and then a meta classifier does the predictions based on the prediction of the base classifiers.

As a first step, a Random Forest was run using its default parameters (it was only adjusted 'class_weight' given the imbalanced data) which gave a much better ROC_AUC than baseline (0.71086 vs 0.63999). The problem was overfitting, the performance on the training data was ROC_AUC=1. Therefore, it was implemented GridSearch with cross-validation in order to find better hyperparameters and avoiding overfitting. Common choice for the number of folds (K) in cross-validation is 5 or 10 (Hastie et al, 2017), it was chosen 5 for time running reasons. The refit parameter of GridSearchCV was set on True, so the best estimator found is trained on the whole training data. Finally, the scoring parameter was fixed in 'ROC_AUC' which is how the model is going to be evaluated at the end.

In order to construct a better Random Forest, it was trained a Decision Tree before, using GridSearchCV to obtain hyperparameters. Then, a Random Forest was made based on the Decision Tree and using GridSearchCV as well. All this process produced a better Random Forest model (ROC_AUC= 0.72709 and a negligible overfitting). In appendix it can be seen details.

In addition, it was trained an AdaBoost classifier. The default setting of AdaBoost[4] yield a good ROC_AUC=0.74516 and has a negligible overfitting. Therefore, using GridSearchCV for ranges in 'n_estimators'[5] and 'learning_rate', they were set on 150 and 1.0 respectively, which yielded a better AdaBoost (ROC_AUC=0.75101) and a negligible overfitting[6].

Then, it was constructed a Stacking model based on the found Random Forest and AdaBoost and using Logistic Regression[7] as meta classifier[8]. As a result, the model yielded a ROC_AUC=0.7284 and a negligible overfitting. The result was not better than its best base classifier (AdaBoost) due to the performance of the other base classifier (RF) which is less.

Finally, regarding AdaBoost model gave the best performance, it was done a new GridSearchCV focus on a higher number of estimators[9]. The new AdaBoost produced a ROC_AUC=0.75360 and has a negligible overfitting.

Results and Discussion

In the next figure, it can be seen a performance comparison between every model.

[4] SAMME is the algorithm version of Scikit-learn for AdaBoost. SAMME.R is set by default which performs better (Geron,2019) and it can be use when the base_estimator has 'predict_proba' which is the case.

[5] Regarding that the default AdaBoost has a negligible overfitting, it could be test increasing the number of estimators in order to fit better the training data.

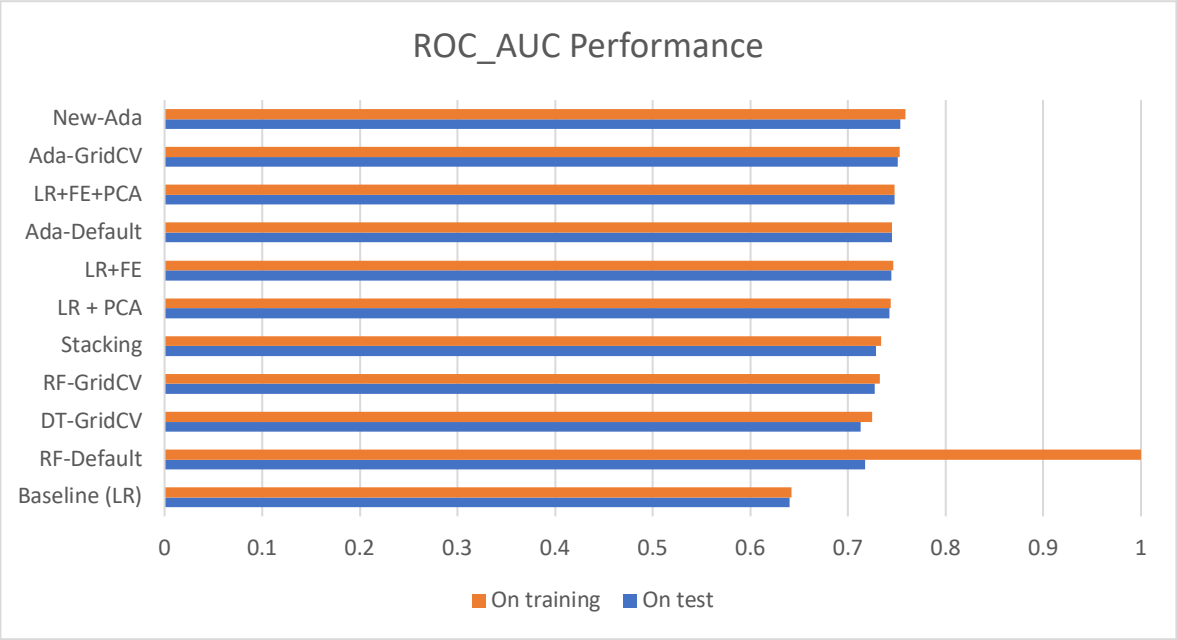
[6] Details in appendix.

[7] It was used the default setting except for 'class_weight' which was set on 'balanced'.

[8] StackingClassifier was set on 5 folds to do the CV and the stack method was 'predict_proba'.

[9] The last AdaBoost was set (by SearchGridCV) in the largest possible number of estimators (considering the given range).

Fig.: Performance by models



Firstly, it can be observed that with algorithms from both approaches (Part 1 with logistic regression models, and part 2 with ensembles methods) was possible to achieve considerable improvement in ROC_AUC score in comparison with the baseline. Secondly, in each algorithm overfitting was controlled very well by setting the right parameters values to the models.

Regarding to the first approach (featuring engineering and PCA), it can be seen that applying featuring engineering and then using a logistic regression classifier was more effective than the results obtained by Random Forest. In addition, PCA technique was almost as effective as the applied featuring engineering. Moreover, the combination of featuring engineering and PCA gave best results than each of them by itself.

About the second approach (optimization of hyper-parameters), it can be noticed that hyper-parameter optimization was effective to resolve the cross-validation problem of the Random Forest with the default parameters (RF-Default). In addition, using GridSearchCV produced better model in terms of performance, actually the best-performance model is an AdaBoost whose hyper-parameters were found by GridSearchCV.

Conclusions

This project helped us in an important way to understand the process of applying Machine Learning techniques on a problem such as Home Credit Risk. Particularly, it was possible to understand in depth techniques such as Feature Engineering, PCA, and how to optimize/choose the hyper-parameters of the models.

With respect to the results, this can be changed depending on the conditions of data and the decisions taking in data cleansing. The problem with this data it had a considerable number of missing values, then whatever decision about how to manage these values can affect positively or negatively the scores gotten.

It was possible an improvement of ROC_AUC in logistic using the techniques of PCA and feature engineering together. The feauture engineering implemented with logistic shows good results because even the values were similar using PCA. That means, selecting appropriate variables according to business and iterating with different polynomial grade it is likely to get better scores.

Adaboost shows the best performance, it was better that all techniques that was implemented. Even using PCA and feauture engineering were not possible get better results than adaboost. That means that this algorithm is more efficient, and can be get the best relations between variables and reduce the dimensionality to get better classification.

Future Work

1. This work considered 2 of 7 databases. It is likely that the results will be different using all the databases either increasing the number of value nulls, variables less correlated with the target which can be do more difficult classify to whatever model. Then, it should be interesting know in which percentage would affect the results doing the same research in the future.
2. Implement other techniques to manage null values instead of the median. One interesting way is grouping common variables and depending of this groups implement for the null values the median, the mean, max, etc depending on each group.
3. About Bureau, to merge with the principal table we convert the values in statistical values. This values can be a problem sometimes because there is an increment of dimensionality of data. Also this values can affect in the correlations between variables making difficult the classification.

References

Hastie, T et al. 2009 'The Elements of Statistical Learning'. Second Edition. Springer. USA.

Geron, A. 2019 'Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. Second Edition .O'Reilly Media, Inc. Canada.

Appendix

Figure 1: Heatmap of Missing Values

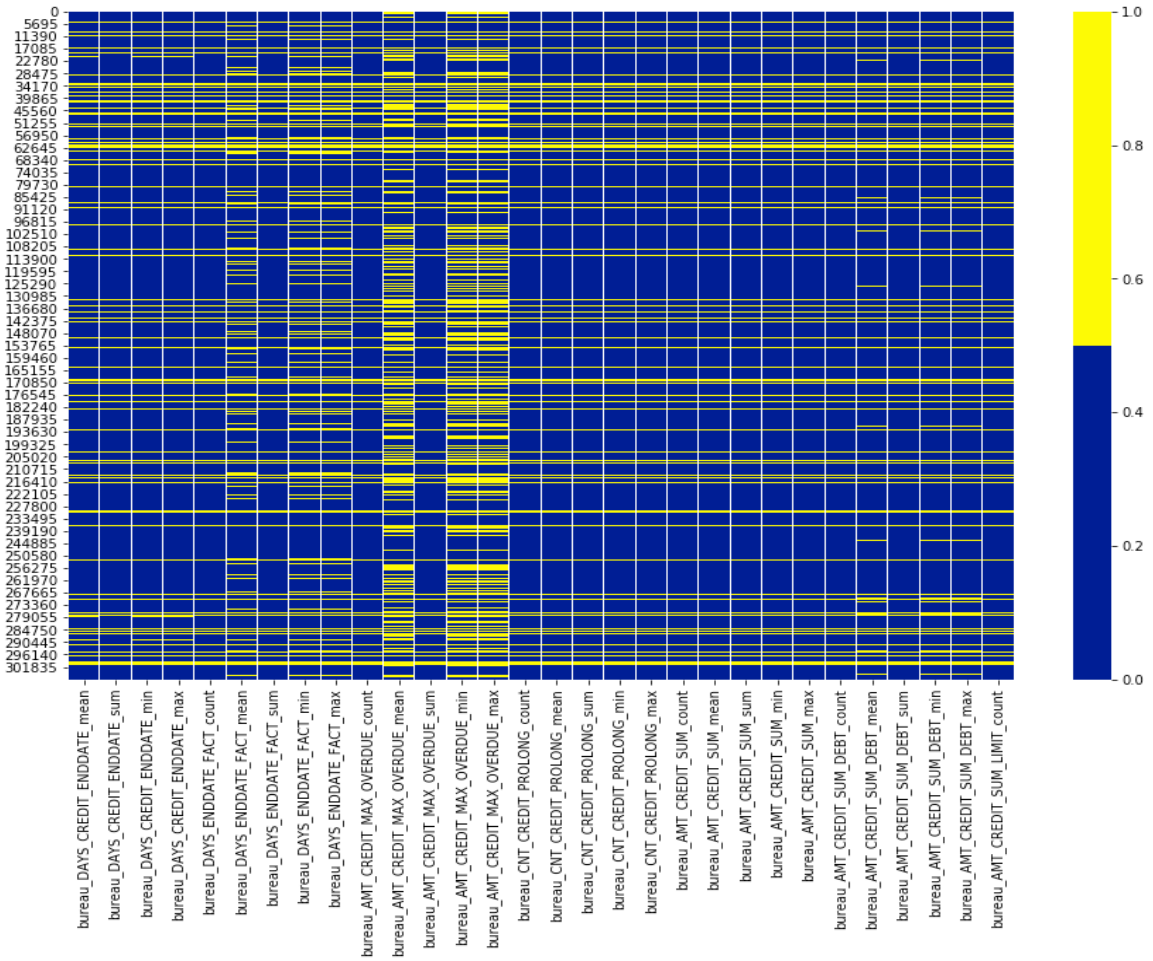


Figure 2: Histogram missing values per obsevation

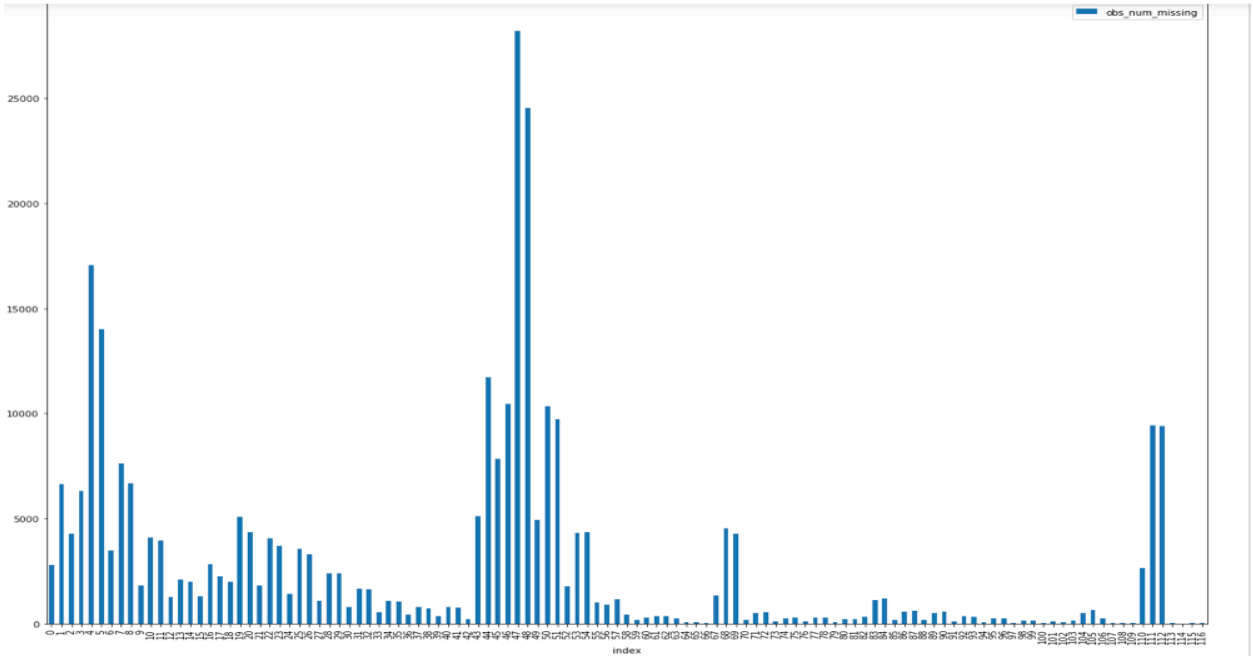


Figure 3: HeatMap of correlations one variable with other 7 variables

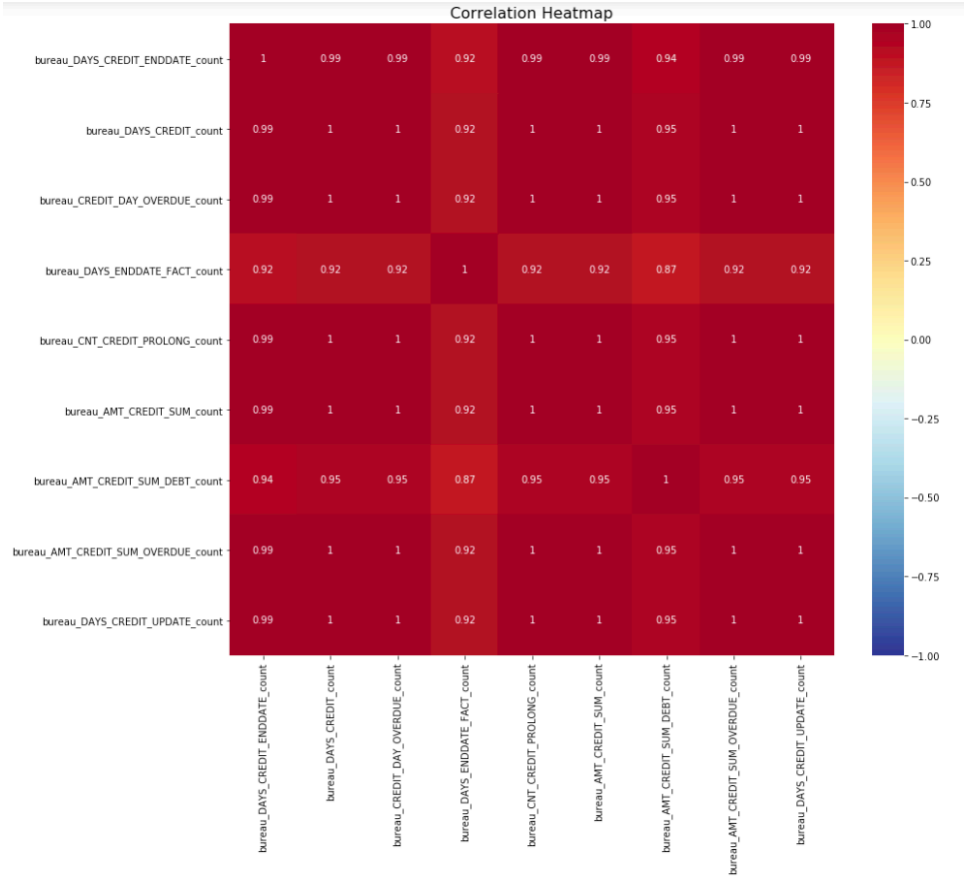


Figure 4: Boxplot to demonstrate the presence of outliers

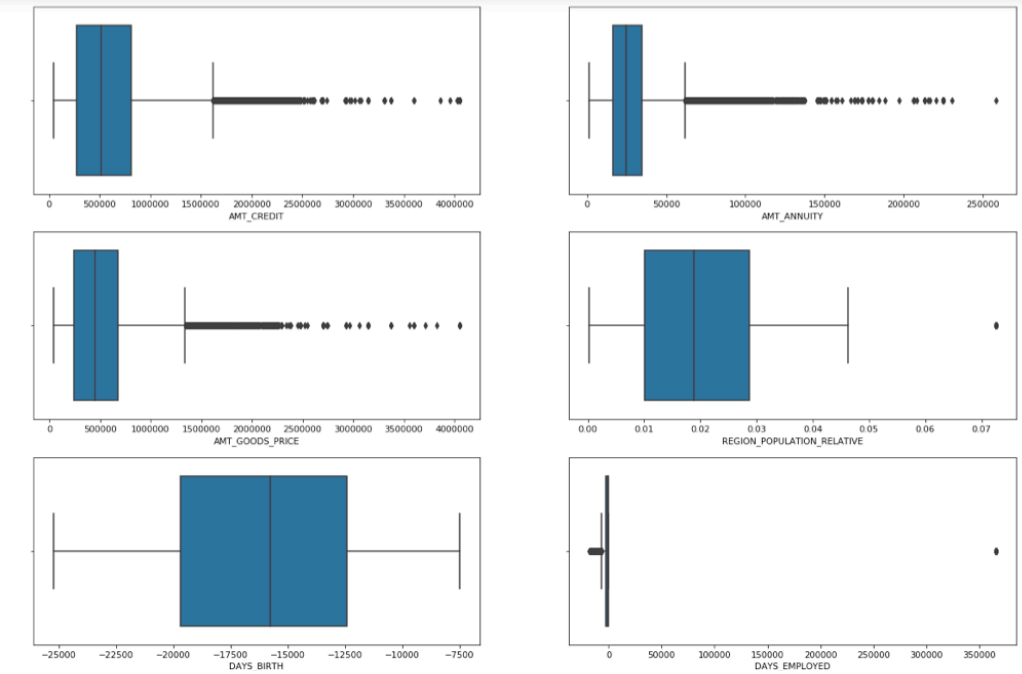
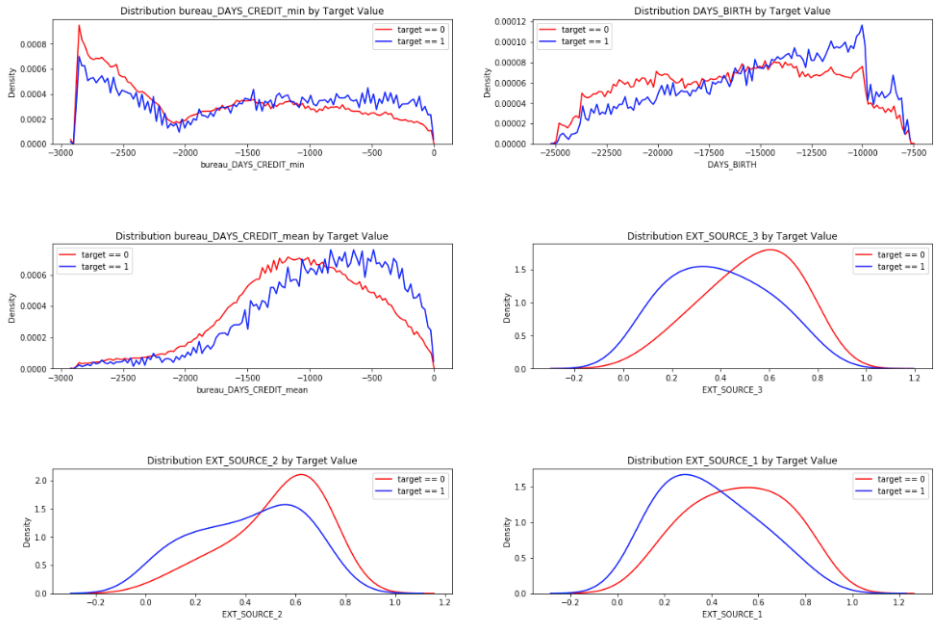


Figure 5: Kernel Density distribution of better variables



In the next table it can be seen the changed hyperparameters, the found values and the running time (best estimator and hyper-parameters searching):

Table: GridSearchCV in Decision Tree and Random Forest

	Hyper-parameters			**Running time best estimator / running time GridSearchCV**
Method	'min_samples_leaf'	'max_features'	'n_estimators'	
Decision Tree	0.01 (1% of the total samples) Range: [0.01,0.05,0.1]	0.8 (80% of total features) Range: ["sqrt",0.9,0.8,0.7]	-	0.16 min / 4.44 min
Random Forest	0.01	0.8	150 Range: [50,100,150]	13.33 min / 119.64 min

Table: GridSearchCV on AdaBoost

	Hyper-parameters		**Running time best estimator / running time GridSearchCV**
Method	'n_estimators'	'learning_rate'	
AdaBoost	100 Range: [5,10,25,50,100]	1.0 Range: [0.5,1.0,1.5]	3.38 min / 71.63 min

Table: Second GridSearchCV on AdaBoost

	Hyper-parameters		**Running time best estimator / running time GridSearchCV**
Method	'n_estimators'	'learning_rate'	
AdaBoost	200 Range: [70,150,200]	1.0 Range: [0.1,0.5,1.0,1.5]	6.78 min / 244.93 min