Home Credit Default Risk

GDPR Members: Basso Gianluca, Vangjelofski Dejvid, Pasti Riccardo
Keggle selected: Home Credit Default Risk
Link to Keggle challenge: https://www.kaggle.com/c/home-credit-default-risk

STEP 1 - PROBLEM TO BE SOLVED & DATA SOURCES
Many people struggle to get loans due to insufficient or non-existent credit histories. And, unfortunately, this population is often taken advantage of by untrustworthy lenders. Home credit strives to broaden financial inclusion and in order to make sure this underserved population has a positive loan experience, Home Credit makes use of a variety of alternative data--including telco and transactional information--to predict their clients' repayment abilities.
There are 7 different sources of data:

- **application_train/application_test:** main training and testing data with information about each loan application at Home Credit.
- **bureau:** data concerning the client's previous credits from other financial institutions.
- **bureau_balance**: monthly data about the previous credits in a bureau.
- **previous_application**: previous applications for loans at Home Credit of clients.
- **POS_CASH_BALANCE**: monthly data about the previous point of sale or cash loans clients have had with Home Credit.
- **credit_card_balance**: monthly data about previous credit cards clients have had with Home Credit.
- **installments_payment**: the track of previous loans at Home Credit.

STEP 2 - DATA EXPLORATION
We started implementing the head of our dataset inside the files:
- 'application_train'. This .csv, provided by the Kaggle, contains the training data of our project.
- 'application_test'. It contains the test data for our challenge.
The aim of the next step was to understand how the dataset was organized, to get deep in the data in order to learn: what's actually inside, what types of data are contained, what ranges of values they have and other main features.

Successively, we set a practical overview of the first rows of the dataset with "app_train.head()" & "app_test.head()".
Then we examined the number of loans repaid as **"TARGET" = 0** and not able to repair as **"TARGET" = 1** through a histogram of the results.
We observed that there were far more loans repaid on time than those not repaid **(282686 repaid against 24825 not repaid)**, meaning a "Class Imbalance Problem" for our model.
Met this obstacle, we created a function to take a look at missing values in each column and then explore its statistical values.

The data frame contains 122 columns and 67 have missing values, with some that reach up to **69.9% of missing values with respect to total values!**
Subsequently, we found the number of each type columns (float 65, int 41, object 16), and we selected any categorical variable with 2 unique categories (dtype== object) to be processed by the label encoding. In parallel, any categorical variable with more than 2 unique categories to use one-hot encoding.

For label encoding we used **Scikit-Learn 'LabelEncoder'** and for **one-hot-encoding** the pandas` **'get_dummies(df)** function. After aligning both the training and test data, we saw that the same features required for machine learning were provided.
Then we have subset the anomalous clients to verify if they tend to have higher or lower rates of default than the rest of the clients, and it turned out that the anomalies had a lower rate of default. According to that, we filled in the anomalous values with (np.nan) for both training and test data and then created a column indicating whether or not the values were anomalous.

At this point we wanted to understand the relevance of the features using the **Pearson correlation coefficient** (.corr), we know that this is not the best method to represent 'relevance' of a feature but it gave us an idea how to proceed, so we found **the most positive 15 correlations** and the **most negative 15 correlations** and we sorted them:

Most Positive Correlations		Most Negative Correlation	
OCCUPATION_TYPE_Laborers	0.043019	EXT_SOURCE_3	-0.178919
FLAG_DOCUMENT_3	0.044346	EXT_SOURCE_2	-0.160472
REG_CITY_NOT_LIVE_CITY	0.044395	EXT_SOURCE_1	-0.155317
FLAG_EMP_PHONE	0.045982	NAME_EDUCATION_TYPE_Higher education	-0.056593
NAME_EDUCATION_TYPE_Secondary / secondary special	0.049824	CODE_GENDER_F	-0.054704
REG_CITY_NOT_WORK_CITY	0.050994	NAME_INCOME_TYPE_Pensioner	-0.046209
DAYS_ID_PUBLISH	0.051457	DAYS_EMPLOYED_ANOM	-0.045987
CODE_GENDER_M	0.054713	ORGANIZATION_TYPE_XNA	-0.045987
DAYS_LAST_PHONE_CHANGE	0.055218	FLOORSMAX_AVG	-0.044003
NAME_INCOME_TYPE_Working	0.057481	FLOORSMAX_MEDI	-0.043768
REGION_RATING_CLIENT	0.058899	FLOORSMAX_MODE	-0.043226
REGION_RATING_CLIENT_W_CITY	0.060893	EMERGENCYSTATE_MODE_No	-0.042201
DAYS_EMPLOYED	0.074958	HOUSETYPE_MODE_block of flats	-0.040594
DAYS_BIRTH	0.078239	AMT_GOODS_PRICE	-0.039645
TARGET	1.000000	REGION_POPULATION_RELATIVE	-0.037227

Finally, we set some plots to conceptualize our intent.
The first histogram to see the distribution of the ages in years and the second one, displays more clearly, the distribution of ages by target values.
But we want to be more sure of the result to use as a basis, so we put **"Age information"** into a separate data frame and binned the age data, then we calculated the averages grouped by the bin and finally we set a bar plot of the age bins and the average of the target.

As a result, we saw that the age group that has failed more to repay is [20-25] this is information that could be directly used by the bank because as we saw younger clients are less likely to repay the loan.

Subsequently we decided to extract the variables with the most (positive and negative) correlations with the target, ‘EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3', 'DAYS_BIRTH'. We set a heatmap with all of them and even a plot that shows the distribution of each EXT_SOURCE by target values, as a result, EXT_SOURCE_3 manifested the greatest in the gap between the values of the target. This means that this feature has some relationship with the probability of an applicant to repay a loan.

STEP 3 - FEATURE ENGINEERING

We started this step making a new dataframe for polynomial features, first with the biggest correlations (‘EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3', 'DAYS_BIRTH') and second with the fewest correlations ('REGION_POPULATION_RELATIVE', 'AMT_GOODS_PRICE', 'HOUSETYPE_MODE_block of flats', 'OCCUPATION_TYPE_Laborers', 'REG_CITY_NOT_LIVE_CITY') because we wanted to putting together them to see how and if their correlations score will changed.
So for both, we use Imputer to handled the missing values, we created the polynomial object with the degree of 3 then we trained and transformed the polynomial features. At this point we created for both groups of features a data frame, we added in the target and we finally found the correlations with the target:

Features with biggest correlations

EXT_SOURCE_2 EXT_SOURCE_3	-0.193939
EXT_SOURCE_1 EXT_SOURCE_2 EXT_SOURCE_3	-0.189605
EXT_SOURCE_2 EXT_SOURCE_3 DAYS_BIRTH	-0.181283
EXT_SOURCE_2^2 EXT_SOURCE_3	-0.176428
EXT_SOURCE_2 EXT_SOURCE_3^2	-0.172282
EXT_SOURCE_1 EXT_SOURCE_2	-0.166625
EXT_SOURCE_1 EXT_SOURCE_3	-0.164065
EXT_SOURCE_2	-0.160295
EXT_SOURCE_2 DAYS_BIRTH	-0.156873
EXT_SOURCE_1 EXT_SOURCE_2^2	-0.156867
Name: TARGET, dtype: float64	
DAYS_BIRTH	-0.078239
DAYS_BIRTH^2	-0.076672
DAYS_BIRTH^3	-0.074273
TARGET	1.000000
1	NaN
Name: TARGET, dtype: float64	

Features with fewest correlations

AMT_GOODS_PRICE HOUSETYPE_MODE_block of flats^2	-0.045723
AMT_GOODS_PRICE HOUSETYPE_MODE_block of flats	-0.045723
REGION_POPULATION_RELATIVE AMT_GOODS_PRICE	-0.045124
REGION_POPULATION_RELATIVE AMT_GOODS_PRICE HOUSETYPE_MODE_block of flats	-0.044130
REGION_POPULATION_RELATIVE HOUSETYPE_MODE_block of flats^2	-0.043955
REGION_POPULATION_RELATIVE HOUSETYPE_MODE_block of flats	-0.043955
AMT_GOODS_PRICE^2	-0.042902
REGION_POPULATION_RELATIVE AMT_GOODS_PRICE^2	-0.041442
AMT_GOODS_PRICE^2 HOUSETYPE_MODE_block of flats	-0.040789
HOUSETYPE_MODE_block of flats^2	-0.040594
Name: TARGET, dtype: float64	
REG_CITY_NOT_LIVE_CITY^3	0.044395
REG_CITY_NOT_LIVE_CITY^2	0.044395
REG_CITY_NOT_LIVE_CITY	0.044395
TARGET	1.000000
1	NaN
Name: TARGET, dtype: float64	

As we can see severally of the new variables have increased their correlations score, but the variations are not so big so we can’t already determine if they actually help the model learn.

STEP 4 MODELLING

The next step, using also other resources, we found out some financial features that may have a high influence on the repayment of a loan. These new features are:

- CREDIT_INCOME_PERCENT: it is the percentage of the credit amount with respect to the client's income.
- ANNUITY_INCOME_PERCENT: the percentage of the loan annuity relative to a client's income.
- CREDIT_TERM: the length of the payment in months.
- DAYS_EMPLOYED_PERCENT: the percentage of the days employed relative to the client's age.

After creating them, we explored these new features with a density plot, looking at their distribution with respect to the target. The graphs though have not been useful to explore the data because of the similarity between the distributions.

Then, we implemented the Logistic Regression as a baseline. The main change we made was to set a low regularization parameter (C = 0.0001) in order to avoid overfitting. We trained the model on the training data using ".fit" and then tried to predict the probability that a loan will default (thus we selected the second column where "TARGET"=1) with the method "predict_proba". Then we saved our prediction in a ".csv" file made of 2 columns ("SK_IS_CURR" & "TARGET").

The logistic regression baseline scored 0.671, meaning that its predictions are not accurate enough.

Next, we gave a try at the Random Forest to see for if we could get better performance and more accuracy. With 100 trees we got a score of 0.678 which was neither satisfying.

The second-last step was to test Polynomial Features and the ones we created by training and testing the data with and without them and then comparing them. Not much surprisingly we found out that even with the new features the scores of the models were of 0.678 for the one with Polynomial Features and 0.679 for the one with the Features we made. Sadly, these scores meant that the features had no effect on model accuracy.

Finally, we checked the Feature Importance of the random forest. As in the correlations we saw in the EDA step, e expected that the most important ones had to be "EXT_SOURCE" and the age of the client ("DAYS_BIRTH"). We created a function that could sort the features according to their importance, normalize them and create a horizontal bar chart displaying the Feature with the highest importance at the top. Observing the graphs, our expectations were satisfied, but we also found that the new features we implemented were in the highest positions of this plot.

STEP 5 - CONCLUSIONS

The aim of this notebook was to create a model that could predict weather a client requesting a loan could repay it or not. To do so, we followed the general rules of machine learning projects.

First: we tried to understand the problem and scraped the data to have an idea of what we had for our model.

Second: we cleaned and formatted the data, even though the dataset was well organised and did not require lot of efforts.

Third: we executed an exploratory data analysis to get more insights and knowledge about the data, their distributions, correlations and more.

Fourth: we tried to engineer new variables and built a baseline model with the Logistical Regression and then tried to make a better one with the Random Forest model.

In conclusion we got predictive models with an accuracy of around 0.67 which, maybe, could be improved for instance by implementing XGBoost, but due to time limitations we were not able to do it.