2020

APPLICATIONS OF BIG DATA

PROJECT REPORT

Caroline Azeufack

Sommaire

Goal of the project

The goal is of the project I to focus on integration of coding best practice, MLFlow and XAI library. The project is organized into 3 parts.

About the dataset

Link: https://www.kaggle.com/c/home-credit-default-risk/data

The dataset used to this project is: **Home Credit Risk.** There are organized in many datasets, but we will use application_train.csv and application_test.csv. Each row represents loan application and is identified by the feature SK_ID_CURR.

In the application_train has one more column than application_test (TARGET) composed by two values [0 => loan was repaid, 1 =>loan was not repaid]

Tools

In this project, I used common tools:

PART I: Building Classical ML project

To start, I import some libraries like: numpy, pandas, matplotlib... and the two datasets in the notebook.

○ Manage missing values

`missing_values_table(df)` I will use this function to know the percentage of each column and sort it by descending on application_train and application_test.

- **Application train**:

I compute the function and the result shows that the dataset has 67 columns missing values. After dropping the first columns with the percent > 50, the dataset has now 81 instead of 122 columns in the beginning.

The CODE_GENDER column has 3 values: (F, M, XNA). XNA value is a null value, so I will replace it by F because it is occurring several times.

- **Application test**:

I compute the function and the result shows that the dataset has 64 columns missing values. After dropping the first columns with the percent > 50, the dataset has now 92 instead of 121 columns in the beginning.

```
Shape before align the data

-> Train Shape : (307511, 80)
-> Test Shape : (48744, 92)

_____________________________________

Shape after align the data

-> Train Shape : (307511, 81)
-> Test Shape : (48744, 80)
```

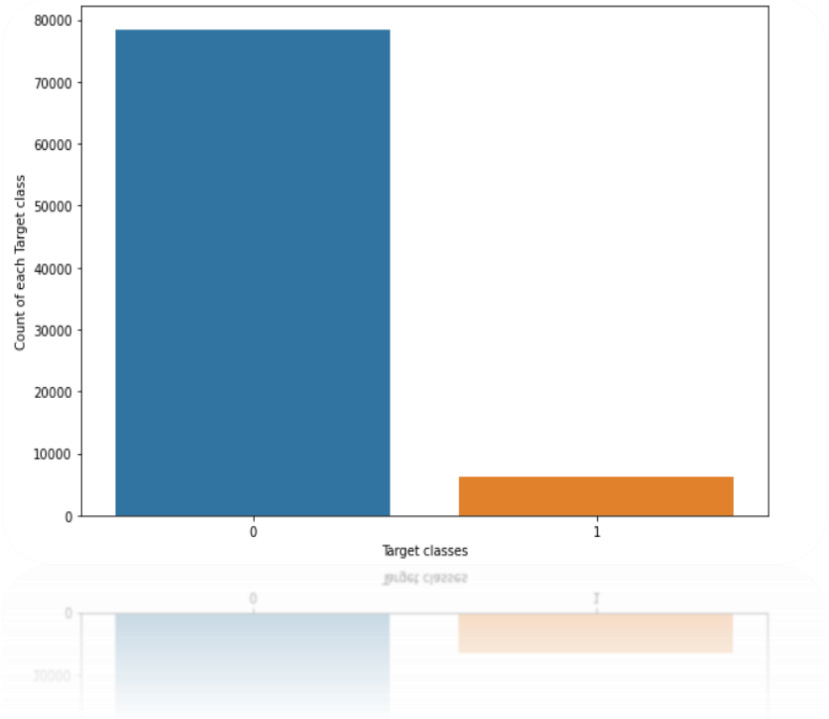
- **Align the datasets**: After the previous operation, there are not the same columns, so I decided to use the pandas align to synchronize both.
  We can see just beside the difference.

- **Row drop missing values**: There are still missing values in the dataset, so I decided to drop all rows where missing values appears.
  We can see just beside the

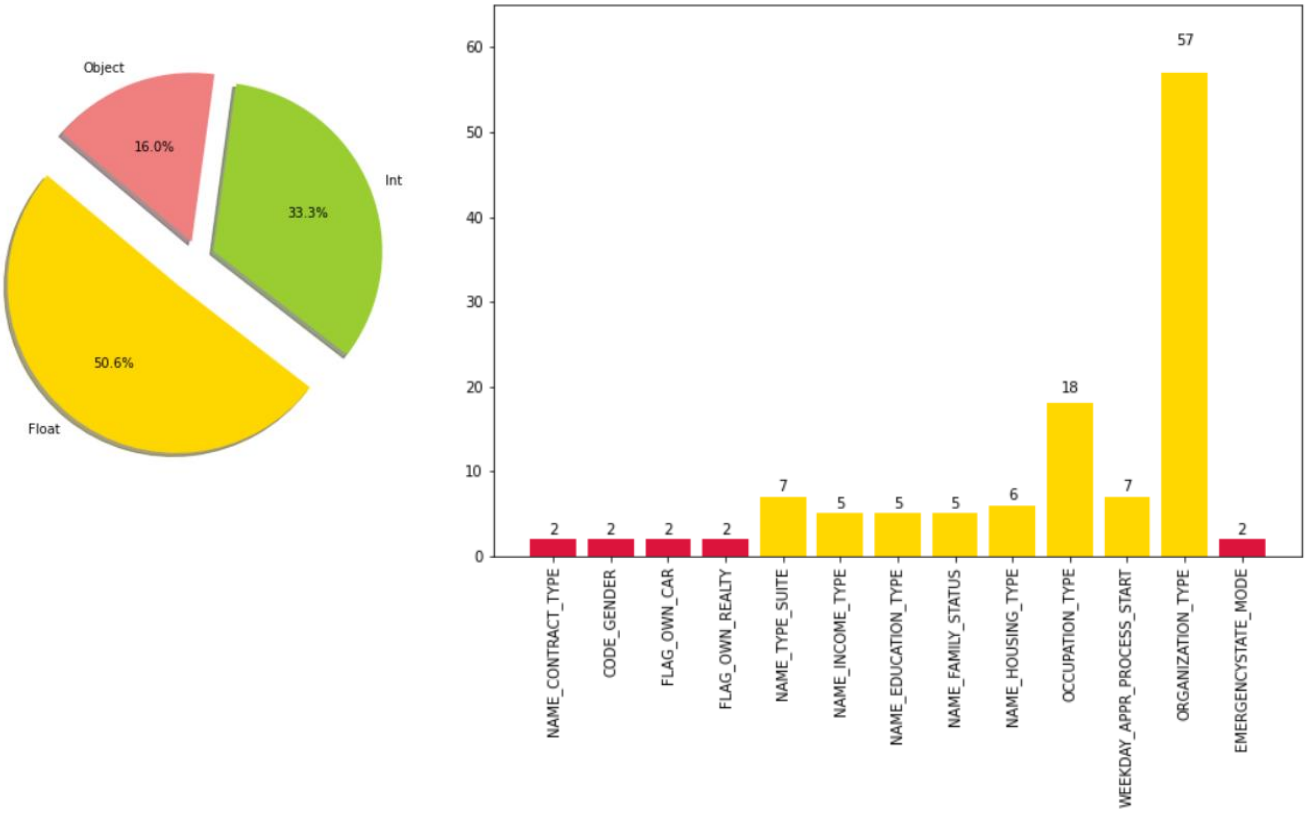
```
Shape before removing missing values on row

-> Train Shape : (307511, 81)
-> Test Shape : (48744, 80)

_____________________________________

Shape after removing missing values

-> Train Shape : (84575, 81)
-> Test Shape : (13858, 80)
```

Data preparation & exploration

First, I will look at the TARGET column on application_train set. We see that there are more 0 values than 1, so that more loan was repaid.



Here, I will look at the column's types of the dataset, because ML models does not accept categorical variables. As we can see the left plot, there are 16% of objects columns.



Most Machine Learning cannot deal with object type, so I will transform it in numeric variables. I defined two functions: LE (data), OHE (data).

LE(data) to transform categorical variables with 2 unique values(represent by red color in right graph above) and OHE(data) to transform categorical variables with more than 2 unique values(represent by yellow color in right graph above).

```
Shape before the labelization
-> Train shape : (84575, 81)
-> Test shape : (13858, 80)

_____________________________________________________________________

Shape after the labelization
-> Shape : (84575, 183)
-> Shape : (13858, 179)

_____________________________________________________________________

Shape after align the datasets to have the same columns
-> Train shape : (84575, 180)
-> Test shape : (13858, 179)
```

- Models

- XGBoost

Model XGboost Report

	precision	recall	f1-score	support
0	0.93	1.00	0.96	15660
1	0.62	0.00	0.01	1255
accuracy			0.93	16915
macro avg	0.78	0.50	0.48	16915
weighted avg	0.90	0.93	0.89	16915

Accuracy for model : 92.59 %

- Random Forest

Model Random Forest Report

	precision	recall	f1-score	support
0	0.93	1.00	0.96	15660
1	0.00	0.00	0.00	1255
accuracy			0.93	16915
macro avg	0.46	0.50	0.48	16915
weighted avg	0.86	0.93	0.89	16915

Accuracy for Random Forest Model: 92.58 %

- GradientBoosting

Model Gradient Boosting Report

	precision	recall	f1-score	support
0	0.93	1.00	0.96	15660
1	0.54	0.01	0.02	1255
accuracy			0.93	16915
macro avg	0.73	0.51	0.49	16915
weighted avg	0.90	0.93	0.89	16915

Accuracy for Random Forest Model: 92.59 %

PART II: Integrate MLFlow to your project

- XGBoost

Metrics for XGBoost:

RMSE	27.2%
MAE	7.4%
R2	-7.8%
ACC	92.6%

Model saved in run cfb3cbd82c264b069dd43c02c5ab07aa

- RandomForest

```
Metrics for Random Forest:
_____________
| RMSE | 27.2% |
_____________
| MAE  | 7.4%  |
_____________
| R2   | -8.0% |
_____________
| ACC  | 92.6% |
_____________

Model saved in run 189fa4243eb947f7a7da2ae550e771cf
```

- GradientBoosting

```
Metrics for Gradient Boosting:
_____________
| RMSE | 27.2% |
_____________
| MAE  | 7.4%  |
_____________
| R2   | -8.0% |
_____________
| ACC  | 92.6% |
_____________

Model saved in run d52c814744f64363b676dc93667d5677
```

PART III: Integrate ML interpretability to your project