Data Scientist Application Exercise

Author:
José Carlos Dávila Almazán

Evaluator:
Konfio

September 31st, 2021

Contents

Introduction

The present document resumes my interpretation on the exercise given. The document is written on LaTeX, and the language employed is Python using Jupyter Notebooks on VS Code, also,the full code and resources can be fount on GitHub as a repository. [1] The project is developed under the Cross Industry Standard Process for Data Mining (CRISP-DM) shown on Figure 1.

Figure 1: CRISP-DM

1 Business Understanding

Konfio is a FinTech that aims to provide financial solutions for small and medium-sized enterprises (SMEs). Taking this into consideration, the profitability of the company is founded on the reliability of this enterprises and a good estimation of the amount to lend, as well as the interest and the period to cover the loan.

On first instance we take into account the "users.csv" file, from which we can have a first approach about "good" and "bad" clients from the criteria of the dataset provider. With this information we obtain that out of the 1,000 clients, only the 53.5% (535) has a good credit record. Also, the relationship between the income and outcome.

Further can be determined by taking into consideration the "credit_reports.csv", in which we find data such as the opening date of the account which can indicate as long with the records under that user whether that particular client would be a good candidate for a loan. The previous records as long as the delinquency can be an indicator to limit the amount to be lend based on previous experiences. This record allows to place trust in better candidates, which are the ones which have been able to increment the previous loans, and maintaining the payments on time.

2 Data Understanding

Going through the data we can find that there is categorical and no categorical values. Having a total of 17 variables for the credit reports data set. In this case we have 7 columns which are categorical, leaving 10 with non-categorical values 1. This data set has a total of 16,309 registers of the 1,000 clients.

Table 1: Credit reports data

Variable	Type of variable
user_id	int64
institution	object
account_type	object
credit_type	object
total_credit_payments	float64
payment_frequency	object
amount_to_pay_next_payment	float64
account_opening_date	object
account_closing_date	object
maximum_credit_amount	float64
current_balance	float64
credit_limit	float64
past_due_balance	float64
number_of_payments_due	float64
worst_delinquency	float64
worst_delinquency_date	object
worst_delinquency_past_due_balance	float64

On the case of the users data set, we only have non-categorical values, which can easily be used to work around.

Table 2: Users data

Variable	Type of variable
id	int64
monthly_income	int64
monthly_outcome	int64
class	int64

Once we know the relevant data, we can already portray some insights about the data set. From the "users.csv" we can start comparing the percentage of people with good records vs the ones with bad ones, as shown on figure 2.

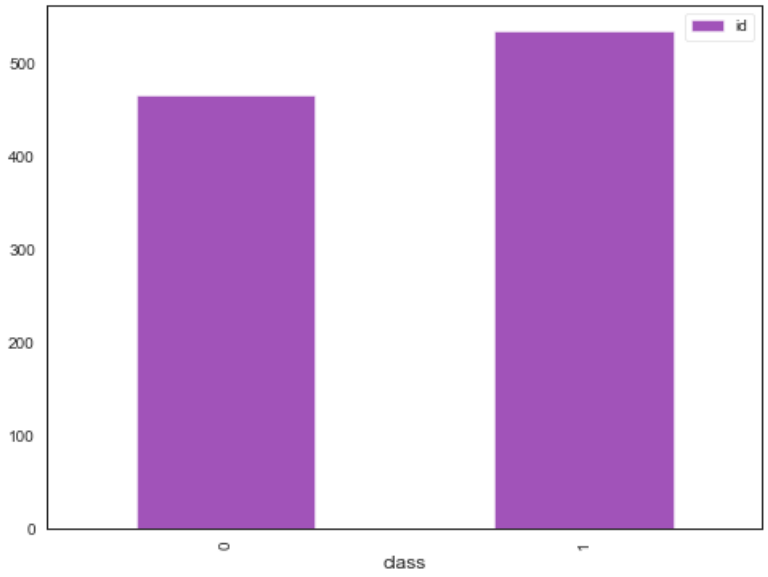

Figure 2: Users classification

After this we now compare the users income after being classified on figure 3. It can be noticed that there is one subject that might be of interest, given the income and the good record.

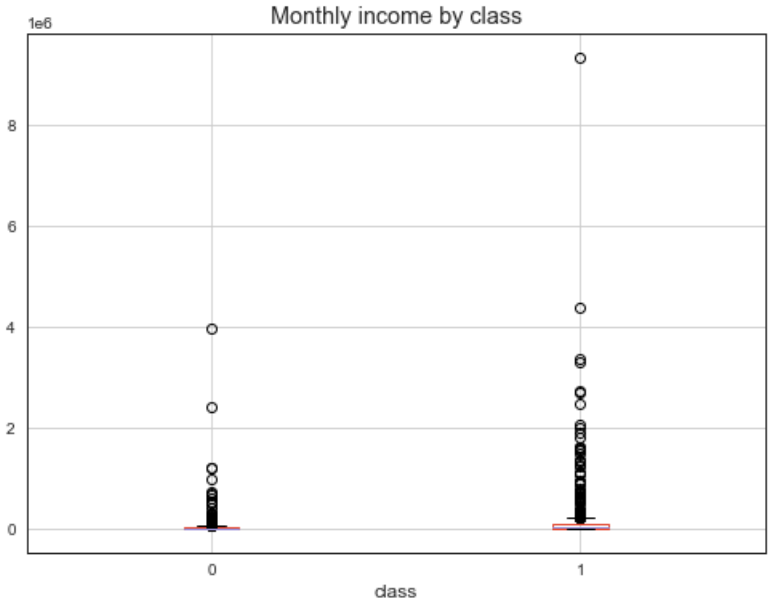

Figure 3: Users income

Another simple, yet important consideration is comparing the income and outcome of the users, which separates potential recipients of a loan determined on their financial capability to meet a payment. This is presented on figure 4.

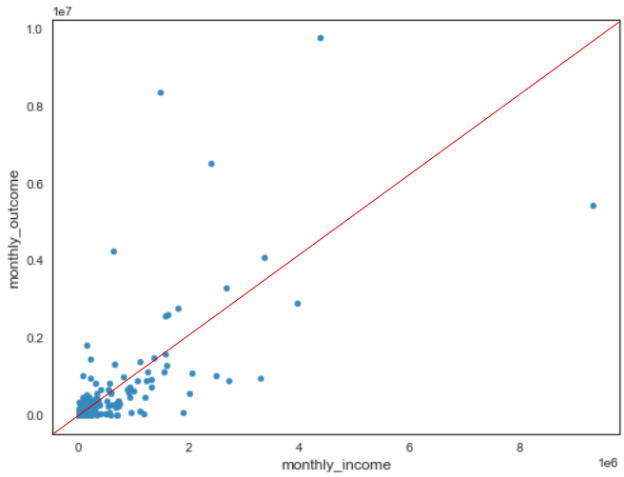

Figure 4: Users income

Finally, a correlation between the data in the table is generated to determine relations between columns. The result of that comparison is shown on figure 5

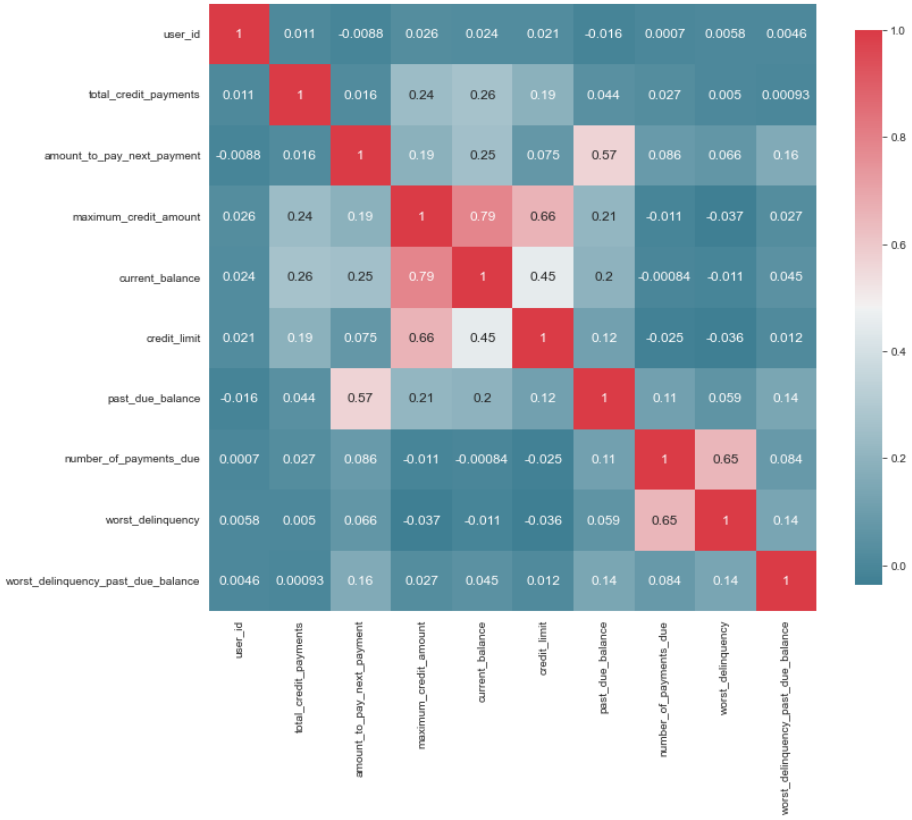

Figure 5: Fields correlation

3 Data Preparation

Once we have a general idea of the data contained within the data set, it is necessary to clean the information and transform it in case it is needed. The categorical values that result of importance can be assigned a "dummy" value in order to be identified and taken into consideration.

The original "credit_reports.csv" presented codification errors while trying to read through accents on the account_type and credit_type columns. On the account_opening-_date, account_closing_date and worst_delinquency_date columns the dates where transformed to datetype, including the one with the "0000-00-00" format which was not recognized directly. Finally four columns where added: institution_cat, account_type_cat, credit_type_cat and payment_frequency_cat; which correspond to institution, account_type, credit_type and payment_frequency respectively. This fields are the non-categorical representation of those categorical values. The data types of this dataset are represented on the table 3

Table 3: Clean and reduced credit registers dataset

Variable	Type of variable
user_id	int64
amount_to_pay_next_payment	float64
account_opening_date	datetime64[ns]
account_closing_date	datetime64[ns]
maximum_credit_amount	float64
current_balance	float64
credit_limit	float64
past_due_balance	float64
number_of_payments_due	float64
worst_delinquency	float64
worst_delinquency_date	datetime64[ns]
worst_delinquency_past_due_balance	float64
institution_cat	int8
account_type_cat	int8
credit_type_cat	int8
payment_frequency_cat	int8

4 Modeling

5 Evaluation

6 Deployment

Listing 1: Ejemplo

```
\begin{minted}{python}
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
\end{minted}
```

7 Conclusiones

References

[1] C. Almazán, "Data science application," 2021. [Online]. Available: https://github.com/CarlosAlmazan/Konfio_DS_A