TUNISIAN REPUBLIC
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
PRIVATE COLLEGE OF ENGINEERING AND TECHNOLOGY

MACHINE LEARNING PROJECT REPORT

Topic :Default of credit card clients

DIRECTED BY :

Aymen Ouhiba
Ahmed Gontara
Eya ghribi
Narjess Ben Slimene
Nader Abassi
Chaima Askri
Zayneb Chniba

Academic year: 2021-2022

TABLE OF CONTENTS

Contents

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Introduction

1.1 Project Definition

A Taiwan-based credit card issuer wants to better predict the likelihood of default for its customers, as well as identify the key drivers that determine this likelihood. This would inform the issuer's decisions on who to give a credit card to and what credit limit to provide.
It would also help the issuer have a better understanding of their current and potential customers, which would inform their future strategy, including their planning of offering targeted credit products to their customers.

Figure 1.1: credit card

1.2 Problem

Credit card is a flexible tool by which you can use bank's money for a short period of time. If you accept a credit card, you agree to pay your bills by the due date listed on your credit card statement. Otherwise, the credit card will be defaulted. When a customer is not able to pay back the loan by the due date and the bank is totally certain that they are not able to collect the payment, it will usually try to sell the loan. After that, if the bank recognizes that they are not able to sell it, they will write it off.
This is called a charge-off. This results in significant financial losses to the bank on top of the damaged credit rating of the customer and thus it is an important problem to be tackled.

1.3 Goal

In this project, we will build a machine learning model which will predict individuals who will default their credit card payment.

CHAPTER 2

EXPLORATORY DATA ANALYSIS

Introduction

In any data science project, data analysis and mining is a critical stage that aims to extract knowledge or insight from enormous amounts of data. It gives us an idea of the additional work involved in quantifying and extracting useful information from our data.

2.1 Data Set Informations

The credit card issuer has gathered information on 30000 customers. The dataset contains information on 24 variables, including demographic factors, credit data, history of payment, and bill statements of credit card customers from April 2005 to September 2005, as well as information on the outcome: did the customer default or not?

- **ID**:unique identification number assigned to each customer.
- **LIMIT_BAL**:amount of given credit access line.
- **SEX**:gender (1 = male; 2 = female).
- **EDUCATION**:highest degree obtained (1 = graduate school; 2 = university; 3 = high school; 4,5,6 = others)
- **MARRIAGE**: marital status (1 = married; 2 = single; 3 = others).
- **AGE**: age in year.
- **PAY0**: monthly payment record in September .
- **PAY2**: monthly payment record in August.
- **PAY3**: monthly payment record in July.
- **PAY4**: monthly payment record in June.
- **PAY5**: monthly payment record in May.
- **PAY6** : monthly payment record in April.
- **BILL_AMT1**: total amount owed in September.
- **BILL_AMT2**: total amount owed in August.

- **BILL_AMT3** : total amount owed in July.
- **BILL_AMT4**: total amount owed in June.
- **BILL_AMT5**: total amount owed in May.
- **BILL_AMT6**: total amount owed in April.
- **PAY_AMT1**: amount of previous payment in September.
- **PAY_AMT2**: amount of previous payment in August.
- **PAY_AMT3**: amount of previous payment in July.
- **PAY_AMT4**: amount of previous payment in June.
- **PAY_AMT5**: amount of previous payment in May.
- **PAY_AMT6**: amount of previous payment in April.

2.2 Exploratory Analysis

First of all, we renamed the first column from "0" to "ID", then we changed the column's names by shorter ones to facilitate handling "dafult_payment_next_month" by "def_pay",also we can see that the repayment status is indicated in columns PAY_0, PAY_2 ... with no PAY_1 column, so we rename PAY_0 to PAY_1 for ease of understanding. And finally we eliminated the first line of our dataset.

```
#we will rename the columns by the first row
df = df.rename(columns=df.iloc[0])
# we will modify the column names by a shorter name to facilitate handling
df = df.rename(columns={'Unnamed: 0': 'ID','default payment next month':'def_pay','PAY_0': 'PAY_1'})
# we will eliminate the first line of our dataset
df = df[1:]
```

```
df
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2
1	1	20000	2	2	1	24	2	2	-1	-1	...	0	0	0	0	689
2	2	120000	2	2	2	26	-1	2	0	0	...	3272	3455	3261	0	1000
3	3	90000	2	2	2	34	0	0	0	0	...	14331	14948	15549	1518	1500
4	4	50000	2	2	1	37	0	0	0	0	...	28314	28959	29547	2000	2019
5	5	50000	1	2	1	57	-1	0	-1	0	...	20940	19146	19131	2000	36681
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
29996	29996	220000	1	3	1	39	0	0	0	0	...	88004	31237	15980	8500	20000
29997	29997	150000	1	3	2	43	-1	-1	-1	-1	...	8979	5190	0	1837	3526
29998	29998	30000	1	2	2	37	4	3	2	-1	...	20878	20582	19357	0	0
29999	29999	80000	1	3	1	41	1	-1	0	0	...	52774	11855	48944	85900	3409
30000	30000	50000	1	2	1	46	0	0	0	0	...	36535	32428	15313	2078	1800

30000 rows × 25 columns

Figure 2.1: Dataset

2.2.1 Variable target

Our variable is default payment next month: whether a customer is defaulted on next months payment or not (1 = defaulter; 0 = non-defaulter).

2.2.2 Lines and columns

The sample size of this data is 30,000, of which 6,636 are in the positive category (default) and 23,364 in the negative category (no default). The sample has a total of 25 variables.

```
print("Les dimensions de la bade de données sont : " )
print(" * Nonmbre de lignes est :", df.shape[0] )
print(" * Nonmbre de colonnes est :", df.shape[1] )
```

```
Les dimensions de la bade de données sont :
 * Nonmbre de lignes est : 30001
 * Nonmbre de colonnes est : 25
```

Figure 2.2: Lines and columns

2.2.3 Type of features

Secondly, we checked the type of each feature and decided to modify them.

The column “ID” consists of randomly generated ID values and it does not affect the customers ability to pay back the bill. That’s why, we decided to set it as the index.

```
print(df.dtypes)
df.dtypes.value_counts()
```

```
Unnamed: 0    object
X1            object
X2            object
X3            object
X4            object
X5            object
X6            object
X7            object
X8            object
X9            object
X10           object
X11           object
X12           object
X13           object
X14           object
X15           object
X16           object
X17           object
X18           object
X19           object
X20           object
X21           object
X22           object
X23           object
Y             object
dtype: object

object    25
dtype: int64
```

(a) Type of features before modification

```
ID             int64
LIMIT_BAL    float64
SEX            int64
EDUCATION      int64
MARRIAGE       int64
AGE            int64
PAY_1          int64
PAY_2          int64
PAY_3          int64
PAY_4          int64
PAY_5          int64
PAY_6          int64
BILL_AMT1    float64
BILL_AMT2    float64
BILL_AMT3    float64
BILL_AMT4    float64
BILL_AMT5    float64
BILL_AMT6    float64
PAY_AMT1     float64
PAY_AMT2     float64
PAY_AMT3     float64
PAY_AMT4     float64
PAY_AMT5     float64
PAY_AMT6     float64
def_pay        int64
dtype: object
```

(b) Type of features after modification

Figure 2.3: Type of features

No other column contains redundant information, poorly formatted or has no impact on customers ability to pay for the balance.

Therefore, we have 23 attributes left to build the model and the last column in the data set will be our target column.

We looked at the first 5 rows of the data to check if data has been read correctly after changing it.

```
#set the ID as the index of our dataset
df.index = df['ID']
df.drop('ID',axis=1,inplace=True)
```

```
df.head(5)
```

PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	def_pay
2	2	-1	-1	-2	...	0.0	0.0	0.0	0.0	689.0	0.0	0.0	0.0	0.0	1
-1	2	0	0	0	...	3272.0	3455.0	3261.0	0.0	1000.0	1000.0	1000.0	0.0	2000.0	1
0	0	0	0	0	...	14331.0	14948.0	15549.0	1518.0	1500.0	1000.0	1000.0	1000.0	5000.0	0
0	0	0	0	0	...	28314.0	28959.0	29547.0	2000.0	2019.0	1200.0	1100.0	1069.0	1000.0	0
-1	0	-1	0	0	...	20940.0	19146.0	19131.0	2000.0	36681.0	10000.0	9000.0	689.0	679.0	0

Figure 2.4: The first 5 rows

2.2.4 Null values

We checked the missing values by counting the number of Null values in each feature. We can see from the figure 2.5 that we don't have any missing values to handle.

```
#Count the number of NULL values in each column
df.isna().sum()
```

```
LIMIT_BAL    0
SEX          0
EDUCATION    0
MARRIAGE     0
AGE          0
PAY_1        0
PAY_2        0
PAY_3        0
PAY_4        0
PAY_5        0
PAY_6        0
BILL_AMT1    0
BILL_AMT2    0
BILL_AMT3    0
BILL_AMT4    0
BILL_AMT5    0
BILL_AMT6    0
PAY_AMT1     0
PAY_AMT2     0
PAY_AMT3     0
PAY_AMT4     0
PAY_AMT5     0
PAY_AMT6     0
def_pay      0
dtype: int64
```

Figure 2.5: Sum of null values

2.2.5 Unicity of features

We verified in this section the unicity of each feature.

```
# Check the uniqueness of the contents of the variables (the unique value of each column)
for col in df.columns:
    print(col)
    print(df[col].unique())
```

Figure 2.6: unicity of features

After that process, we can look into more details about the data by doing the description of each feature.

```
#a detailed statistical description on the columns
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
LIMIT_BAL	30000.0	167484.322667	129747.661567	10000.0	50000.00	140000.0	240000.00	1000000.0
SEX	30000.0	1.603733	0.489129	1.0	1.00	2.0	2.00	2.0
EDUCATION	30000.0	1.853133	0.790349	0.0	1.00	2.0	2.00	6.0
MARRIAGE	30000.0	1.551867	0.521970	0.0	1.00	2.0	2.00	3.0
AGE	30000.0	35.485500	9.217904	21.0	28.00	34.0	41.00	79.0
PAY_1	30000.0	-0.016700	1.123802	-2.0	-1.00	0.0	0.00	8.0
PAY_2	30000.0	-0.133767	1.197186	-2.0	-1.00	0.0	0.00	8.0
PAY_3	30000.0	-0.166200	1.196868	-2.0	-1.00	0.0	0.00	8.0
PAY_4	30000.0	-0.220667	1.169139	-2.0	-1.00	0.0	0.00	8.0
PAY_5	30000.0	-0.266200	1.133187	-2.0	-1.00	0.0	0.00	8.0
PAY_6	30000.0	-0.291100	1.149988	-2.0	-1.00	0.0	0.00	8.0
BILL_AMT1	30000.0	51223.330900	73635.860576	-165580.0	3558.75	22381.5	67091.00	964511.0
BILL_AMT2	30000.0	49179.075167	71173.768783	-69777.0	2984.75	21200.0	64006.25	983931.0
BILL_AMT3	30000.0	47013.154800	69349.387427	-157264.0	2666.25	20088.5	60164.75	1664089.0
BILL_AMT4	30000.0	43262.948967	64332.856134	-170000.0	2326.75	19052.0	54506.00	891586.0
BILL_AMT5	30000.0	40311.400967	60797.155770	-81334.0	1763.00	18104.5	50190.50	927171.0
BILL_AMT6	30000.0	38871.760400	59554.107537	-339603.0	1256.00	17071.0	49198.25	961664.0
PAY_AMT1	30000.0	5663.580500	16563.280354	0.0	1000.00	2100.0	5006.00	873552.0
PAY_AMT2	30000.0	5921.163500	23040.870402	0.0	833.00	2009.0	5000.00	1684259.0
PAY_AMT3	30000.0	5225.681500	17606.961470	0.0	390.00	1800.0	4505.00	896040.0
PAY_AMT4	30000.0	4826.076867	15666.159744	0.0	296.00	1500.0	4013.25	621000.0
PAY_AMT5	30000.0	4799.387633	15278.305679	0.0	252.50	1500.0	4031.50	426529.0
PAY_AMT6	30000.0	5215.502567	17777.465775	0.0	117.75	1500.0	4000.00	528666.0
def_pay	30000.0	0.221200	0.415062	0.0	0.00	0.0	0.00	1.0

Figure 2.7: Description

This figure 2.8 give us an idea about the average amount of a given credit access line which is 167484 with an extremely high standard deviation close to 130000, that can be explained by the large number of maximum credit card limits of 1 million.

The average education level is 1.853, which indicates that most customers obtained bachelor or master degrees.

More customers are either married or single, compared to the number of unknown.
The median age is 35.5 with 9.2 standard deviation and the youngest customer is 21 years old, while the oldest is 79 years old.

2.3 Data visualisation

We are starting by modifying the values "0", "5" and "6" of the education column and regrouping them in one value "4". ("4":means others)

```
df.loc[(df.EDUCATION == 0) | (df.EDUCATION == 5) | (df.EDUCATION == 6) , 'EDUCATION'] = 4
df['EDUCATION'].value_counts()
```

```
2    14030
1    10585
3     4917
4      468
Name: EDUCATION, dtype: int64
```

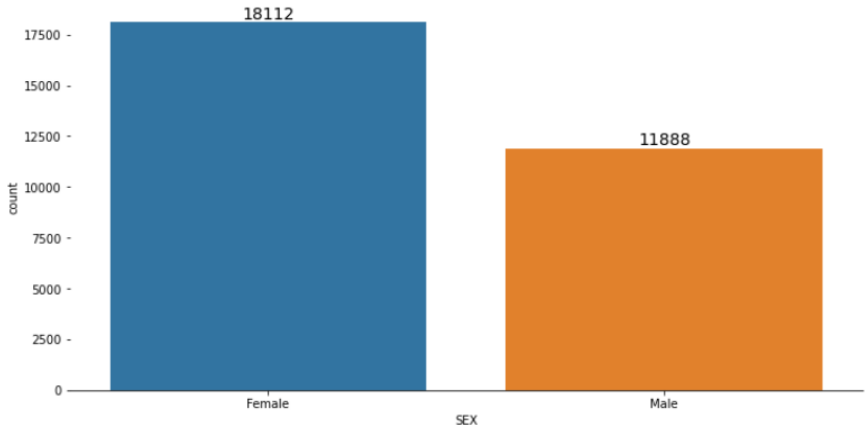
Figure 2.8: The value counts of education feature

2.3.1 Quantitative variables

To begin our visualization, we created a copy of our dataset named preprocessed_df, then started the creation of our charts.

- **SEX:**

Starting with the first plot which indicate the number of male and female in our dataset.




```
Percentage share of each category:
Female    60.373333
Male      39.626667
Name: SEX, dtype: float64
```

Figure 2.9: SEX-count

We can see from figure 2.9 that the number of female are more important with 60,37% comparing it with the number of male with 39,62% which indicate the imbalance between the number of male and female in our dataset.

- **Def_pay:**

Secondly, we are going to compare the number of "YESdefaulters" and "NONdefaulters".

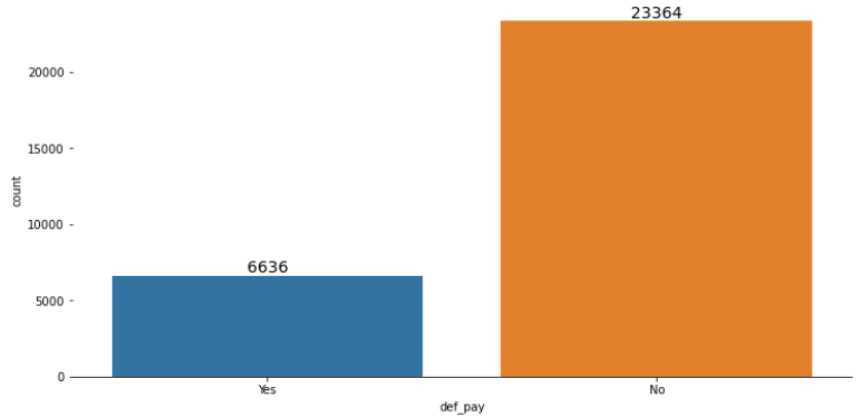


Figure 2.10: Def_pay - count

We can notice from the graph in the figure 2.10 that 6636 customers (approximately 22.12% of the entire data set) will default in the next month's payment, while 23364 will not default. This result is quite reasonable because the difference is insignificant so that this data set should not have much biased information and can be used in our further analysis.

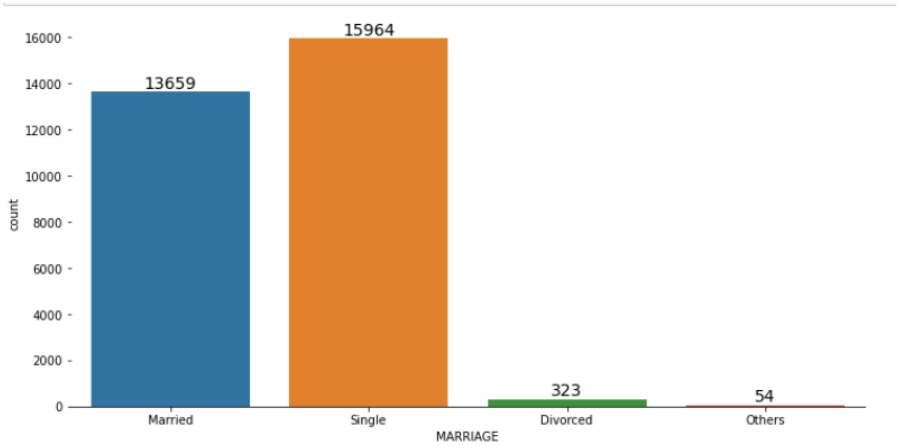
The amount of the credit line may be a good indicator of defaulting behavior. Intuitively, people who have good credit scores are likely given high credit lines compared to those who have very low credit scores.

On the other hand, people who have had default payment in the past are usually hard to raise their credit lines. Therefore, the number of defaulters with credit lines in the top 25% should be much smaller than the number of defaulters having low credit lines.

Finally this figure indicates that we have an imbalance between the two classes(YESdefaulters and Non Defaulters).

- **MARRIAGE:**

Then, we did a chart that devise the number of customers according to their marital status.



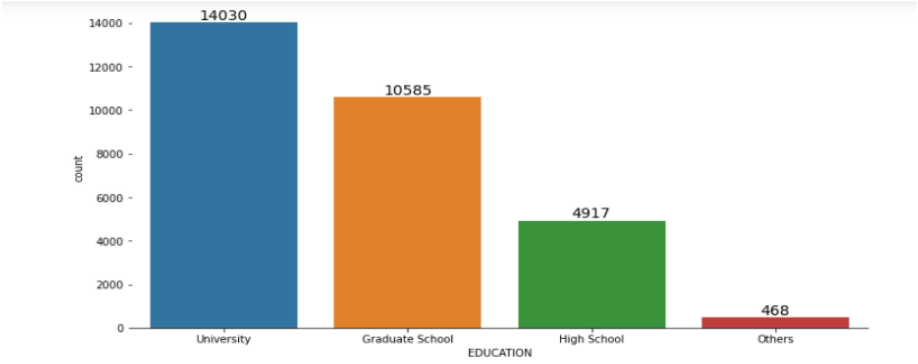

```
Percentage share of each category:
Single      53.213333
Married     45.530000
Divorced     1.076667
Others       0.180000
Name: MARRIAGE, dtype: float64
```

Figure 2.11: MARRIAGE-count

From the figure 2.11, we can see that the number of "Single" clients is the most important with 53.21% (15964 clients), then we have the "Married" ones with 45.53%. At the bottom we have the "Divorced" and others with 1.18% (377 clients).

- **EDUCATION:**

For the education feature, we also did a chart to devise each academic status by its number of clients.




```
Percentage share of each category:
University         46.766667
Graduate School    35.283333
High School        16.390000
Others              1.560000
Name: EDUCATION, dtype: float64
```

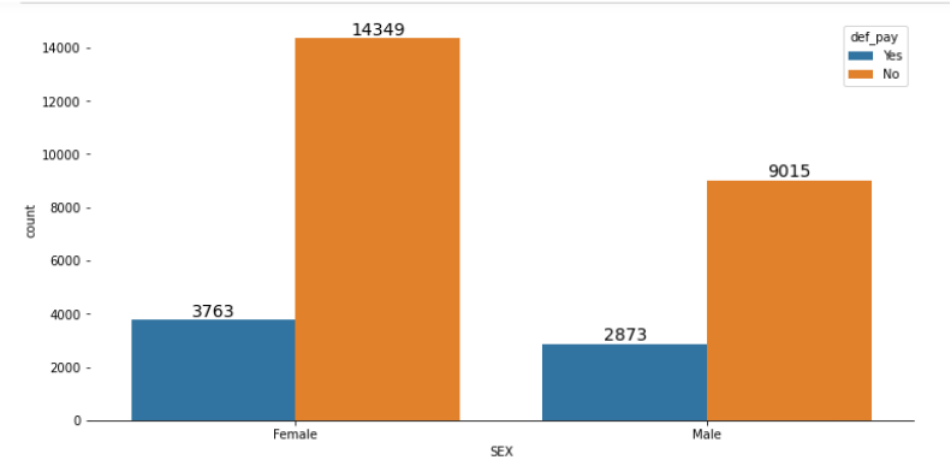
Figure 2.12: EDUCATION-count

From the visualisation in figure 2.12, more credit holders are "University" students with 46.79% (14030 clients) followed by "Graduated" ones with 35.28% (10585 clients) and then

"High school" students 16.39% (4917 clients).

- **Sex vs default payment:**

Next, we checked the impact of sex on the number of default payments by classifying clients depending to their sex to "YESdefaulters" and "NONdefaulters".



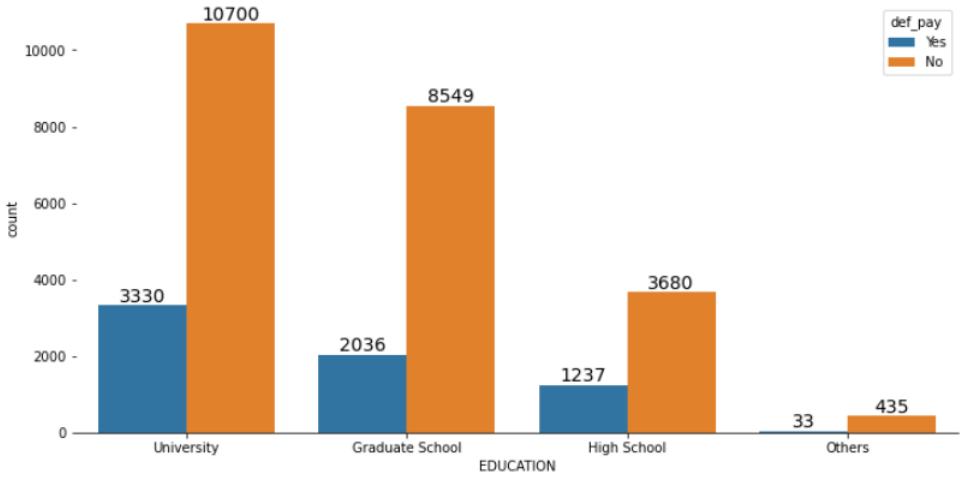
def_pay	total_no_default	total_yes_default	total_default	percentage_no_default	percentage_yes_default
SEX					
Female	14349	3763	18112	0.792237	0.207763
Male	9015	2873	11888	0.758328	0.241672
All	23364	6636	30000	0.778800	0.221200

Figure 2.13: SEX vs def_pay

It is clear from the above result, from the figure 2.13, that women generally have less percentage of default payments with 3763 female clients(20.77%) than men with 2873 (24.16%).
Non-Defaults have a higher proportion of Females with 14394 female client (79.22%).

- **Type of education vs default payment:**

In this section, we checked the impact of the type of education on the number of default payments by classifying clientsdepending to their sex to "YESdefaulters" and "NONdefaulters".



Out[28]:

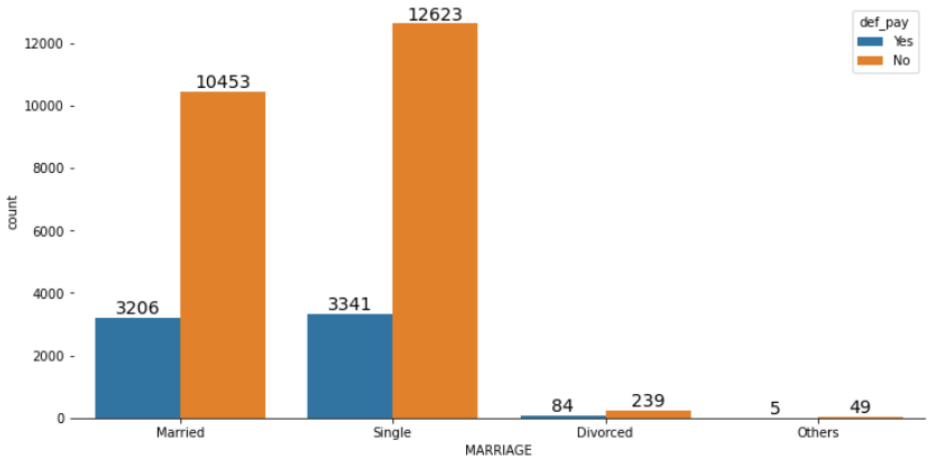
def_pay	total_no_default	total_yes_default	total_default	percentage_no_default	percentage_yes_default
EDUCATION					
Graduate School	8549	2036	10585	0.807652	0.192348
High School	3680	1237	4917	0.748424	0.251576
Others	435	33	468	0.929487	0.070513
University	10700	3330	14030	0.762651	0.237349
All	23364	6636	30000	0.778800	0.221200

Figure 2.14: Type of education vs def_pay

According to the above plot (figure2.15), graduated people have a lower percentage of default payment than university students and high school students.

- **Marital status vs default payment:**

We checked also the impact of the martial status ofclients on the number of default payments by classifying clientsdepending to their sex to "YESdefaulters" and "NONdefaulters".



Out[29]:

MARRIAGE	def_pay	total_no_default	total_yes_default	total_default	percentage_no_default	percentage_yes_default
Divorced		239	84	323	0.739938	0.260062
Married		10453	3206	13659	0.765283	0.234717
Others		49	5	54	0.907407	0.092593
Single		12623	3341	15964	0.790717	0.209283
All		23364	6636	30000	0.778800	0.221200

Figure 2.15: Marital status vs def_pay

From the above plot it is clear that those people who have marital status single have less default payment than married status people and divorced people.

- **Age vs default payment:**

Next, We checked the impact of age of clients on the number of default payments.

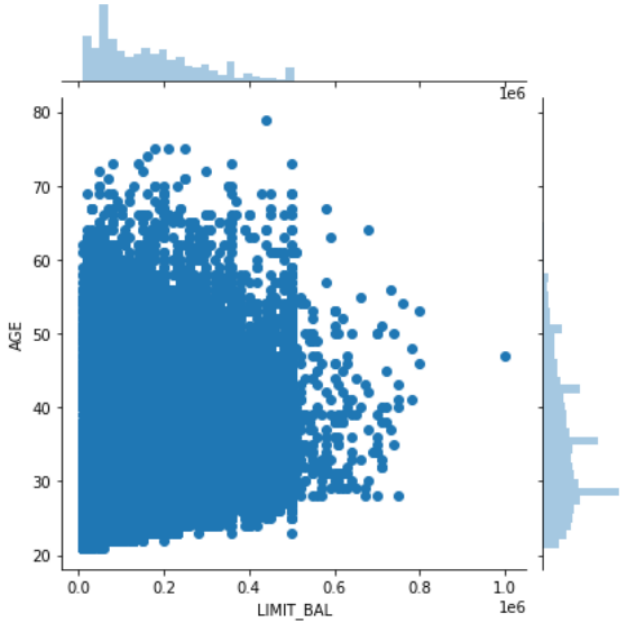


Figure 2.16: Age vs def_pay

We can conclude from the figure 2.16 that when the age increases the limit balance (LIMIT_BAL) increases as well.

In this step, we did two plots to compare the evaluation of each BILL_AMT2-6 according to the BILL_AMT the month before.

•Bill amount vs Bill amount of other months (high BILL_AMT):

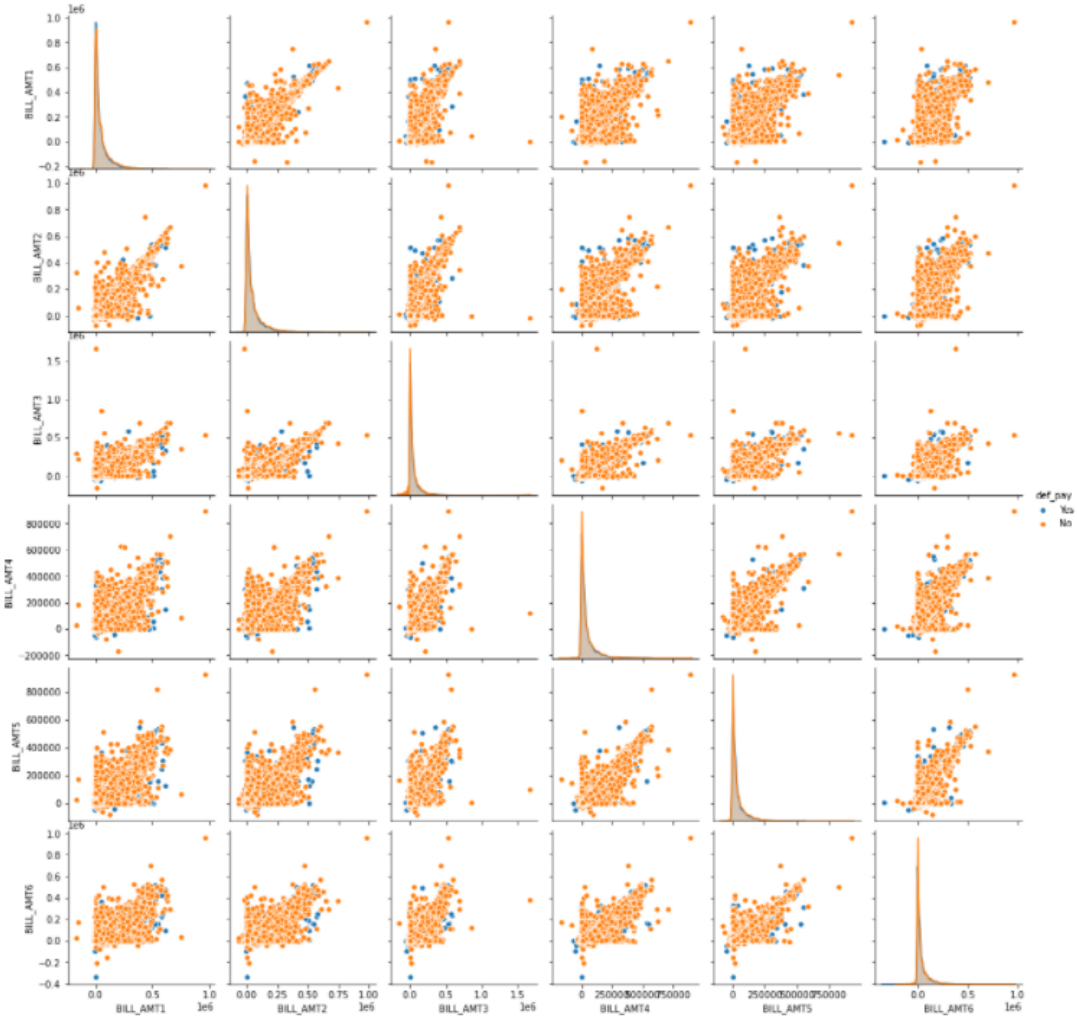


Figure 2.17: BILL_AMT vs BILL_AMT OF OTHER MONTHS (high BILL_AMT)

The slope is increasing so we can conclude from this graph that if the amount of the BILL_AMT is high, the next month's bill will be more expensive than the month before.

- **Bill amount vs Bill amount of other months (less BILL_AMT):**

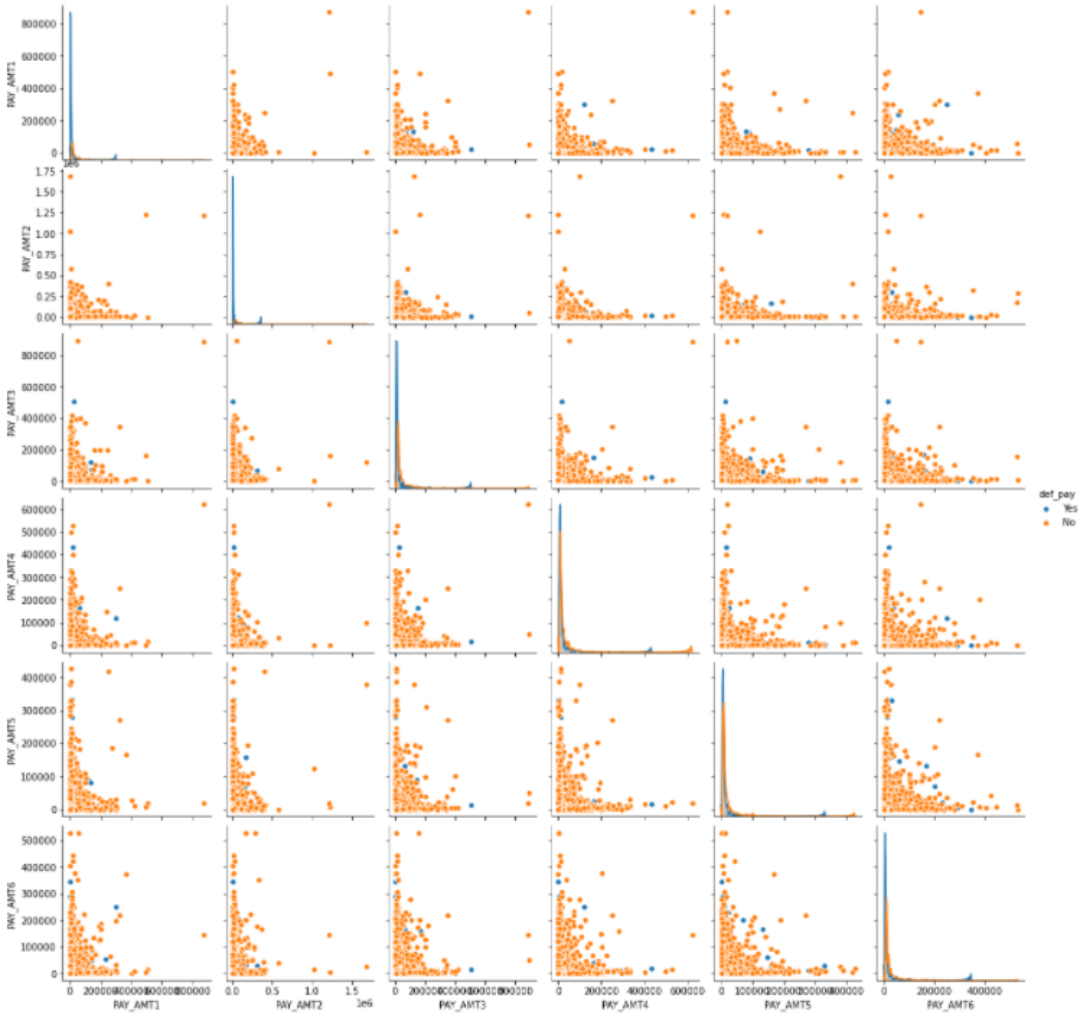


Figure 2.18: BILL_AMT vs BILL_AMT OF OTHER MONTHS (less BILL_AMT)

However, The above graph helps us to visualize that if someone pay a little bit more this month the next month it will pay more less.

- **Age count - Sex:**

To finish our data visualisation , we did a plot to classify the age of each client according to their sex and his class (YESdefaulter or NONdefaulter).

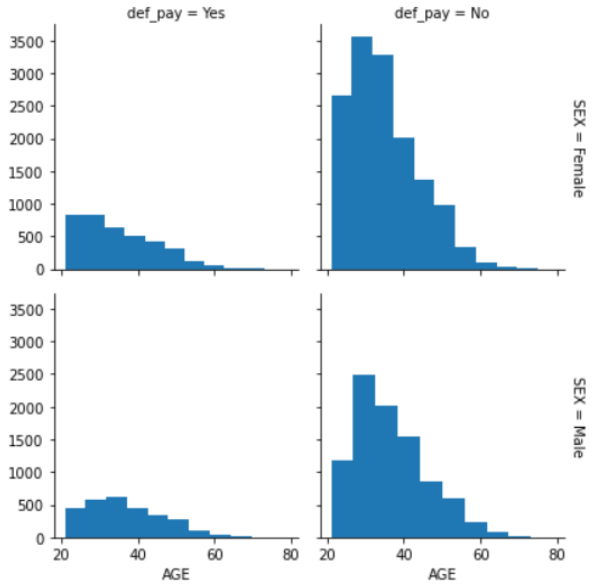


Figure 2.19: AGE-count (SEX)

From the above FaceGrid Plot we can see that "NonDefaults" have a higher proportion of people 30-40years and specially female clients with a number between 3000 and 3500, followed by male clients between 30-40 years with approximately 2500 customers.
Then we have male and female between 40-60 years old.

Finally, we can see that the "YesDefaulters" are higher with female clients at age of 20-30 and male clients 30-40 years old of numbers between 500-700.

2.3.2 Correlation analysis

To better understand our data, it is essential to study the relationships between the different variables. To do this, we checked the correlation of the independent variables with our target (dependent) variable.

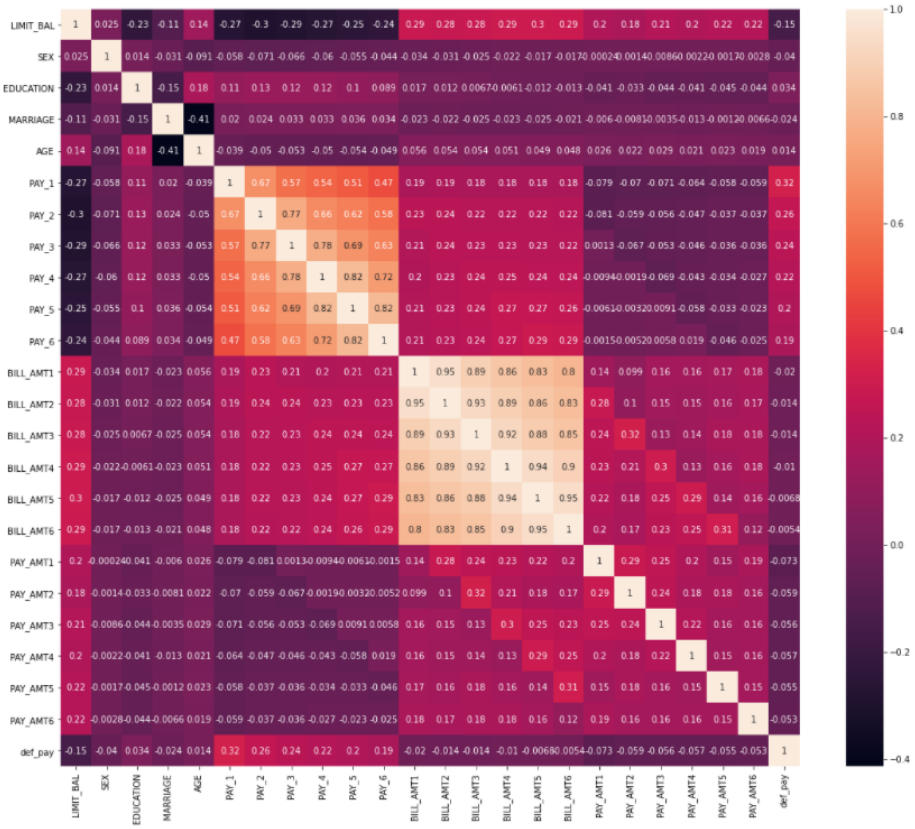


Figure 2.20: AGE-count (SEX)

From figure 2.20 we can see that the variables PAY_0 -> PAY_X (around the degradation 0.2) are the strongest predictors of default pay, followed by the LIMIT_BAL and PAY_AMT_X variables.

And we can say that PAY_0 -> PAY_X are strongly correlated. Same for BILL_AMT.

Looking at the heatmap, we figured out that the target variable def_pay "default.payment.next.month" depends on pay variables more.

We can't drop the other features because it will be the loss of information.

2.4 Data cleaning

Now move towards data cleaning .First ,we checked the false data month by month (if there is a contradiction between the values of features pay = -2 and the values of two features bill_amt and pay_amt. we will erase the erroneous row)

```
indexNames = df[ ((df['BILL_AMT1']==(df['BILL_AMT2']))&(df['PAY_AMT1']==0) &(df['BILL_AMT1']<=0)  &(df['PAY_1'] != -2)) ].index
df.drop(indexNames , inplace=True)
indexNames = df[ ((df['BILL_AMT2']==(df['BILL_AMT3']))&(df['PAY_AMT2']==0) &(df['BILL_AMT2']<=0)  &(df['PAY_2'] != -2)) ].index
df.drop(indexNames , inplace=True)
indexNames = df[ ((df['BILL_AMT3']==(df['BILL_AMT4']))&(df['PAY_AMT3']==0) &(df['BILL_AMT3']<=0)  &(df['PAY_3'] != -2)) ].index
df.drop(indexNames , inplace=True)
indexNames = df[ ((df['BILL_AMT4']==(df['BILL_AMT5']))&(df['PAY_AMT4']==0)  &(df['BILL_AMT4']<=0) &(df['PAY_4'] != -2)) ].index
df.drop(indexNames , inplace=True)
indexNames = df[ ((df['BILL_AMT5']==(df['BILL_AMT6']))&(df['PAY_AMT5']==0)  &(df['BILL_AMT5']<=0) &(df['PAY_5'] != -2)) ].index
df.drop(indexNames , inplace=True)
```

Then,we did a check up of the false data month by month (if there is a contradiction between the values of features pay = -1 and the values of two features bill_amt and pay_amt. we will erase the erroneous row)

```
indexNames = df[ ((df['BILL_AMT1']-df['PAY_AMT1']<0)&(df['BILL_AMT1']<=0)&(df['PAY_1'] != -1)&(df['PAY_1']!=-2) )].index
df.drop(indexNames , inplace=True)
indexNames = df[ ((df['BILL_AMT2']-df['PAY_AMT2']<0)&(df['BILL_AMT2']<=0)&(df['PAY_2'] != -1)&(df['PAY_2']!=-2)) ].index
df.drop(indexNames , inplace=True)
indexNames = df[ ((df['BILL_AMT3']-df['PAY_AMT3']<0)&(df['BILL_AMT3']<=0)&(df['PAY_3'] != -1)&(df['PAY_3']!=-2)) ].index
df.drop(indexNames , inplace=True)
indexNames = df[ ((df['BILL_AMT4']-df['PAY_AMT4']<0)&(df['BILL_AMT4']<=0)&(df['PAY_4'] != -1)&(df['PAY_4']!=-2)) ].index
df.drop(indexNames , inplace=True)
indexNames = df[ ((df['BILL_AMT5']-df['PAY_AMT5']<0)&(df['BILL_AMT5']<=0)&(df['PAY_5'] != -1)&(df['PAY_5']!=-2)) ].index
df.drop(indexNames , inplace=True)
indexNames = df[ ((df['BILL_AMT6']-df['PAY_AMT6']<0)&(df['BILL_AMT6']<=0)&(df['PAY_6'] != -1)&(df['PAY_6']!=-2)) ].index
df.drop(indexNames , inplace=True)
```

After that,we examined to see if there are any negative values in PAY_AMT that have to be dropped

```
indexNames = df[ (df['PAY_AMT1']<0)   ].index
df.drop(indexNames , inplace=True)
indexNames = df[ (df['PAY_AMT2']<0)   ].index
df.drop(indexNames , inplace=True)
indexNames = df[ (df['PAY_AMT3']<0)   ].index
df.drop(indexNames , inplace=True)
indexNames = df[ (df['PAY_AMT4']<0)   ].index
df.drop(indexNames , inplace=True)
indexNames = df[ (df['PAY_AMT5']<0)   ].index
df.drop(indexNames , inplace=True)
indexNames = df[ (df['PAY_AMT6']<0)   ].index
df.drop(indexNames , inplace=True)
```

and we looked over the number of rows we are having after the cleaning

```
df.shape[0]
```

```
25652
```

Finally, we can have a try of training the model with the most dependent features and evaluate the model also.

2.5 Conclusion

During this chapter, we gave an overview about our dataSet information and the process of Data cleaning. The next chapter will be dedicated to the process of modeling.