LAB

Launch a Jupyter Notebook

Download the cc_headers.xls file from here:
https://github.com/fenago/python4DS/blob/main/cc_headers.xls

Please read what each column (variable / feature) means by looking at the data dictionary here:

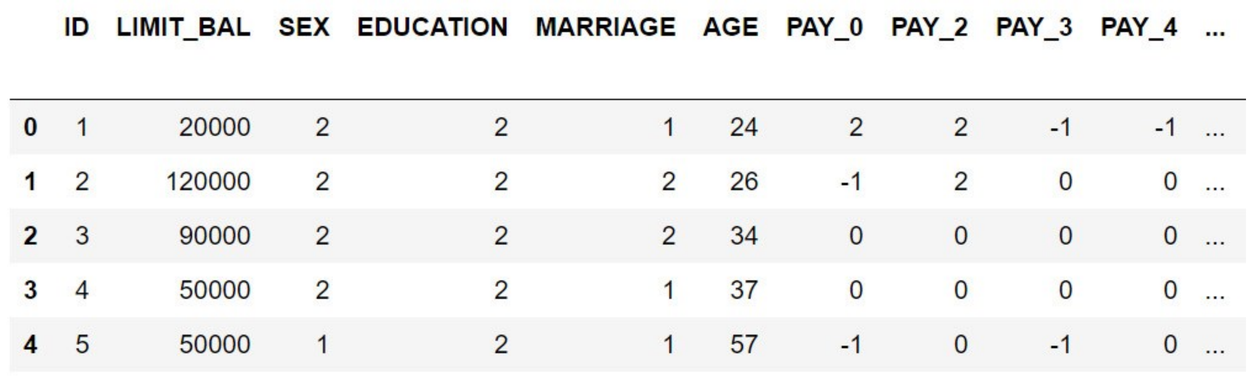
https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset

1. Import the Libraries:

```
# Import basic libraries

import numpy as np

import pandas as pd

# import visualization libraries

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline
```

2. Read in the dataset into your work environment:

```
df = pd.read_excel('default_credit_clients.xls')

df.head(5)
```



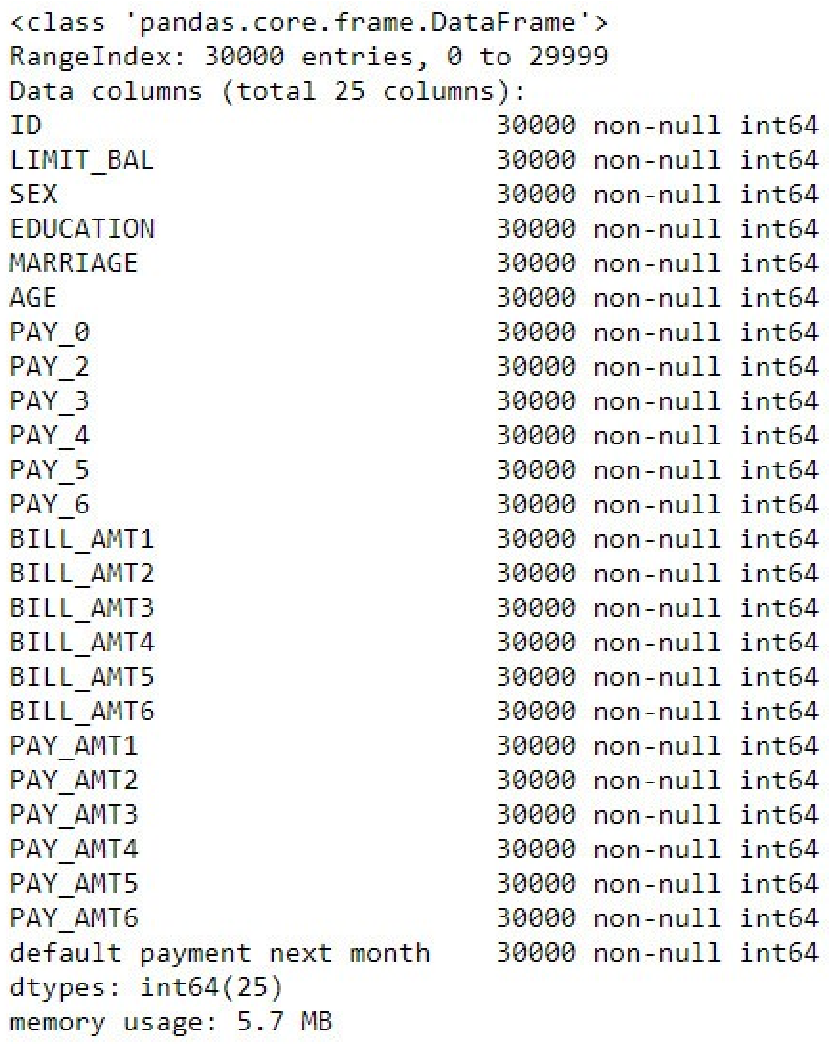
	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...
0	1	20000	2	2	1	24	2	2	-1	-1	...
1	2	120000	2	2	2	26	-1	2	0	0	...
2	3	90000	2	2	2	34	0	0	0	0	...
3	4	50000	2	2	1	37	0	0	0	0	...
4	5	50000	1	2	1	57	-1	0	-1	0	...

5 rows × 25 columns

3. Check the metadata of your data:

# Getting Meta Data Information about the dataset

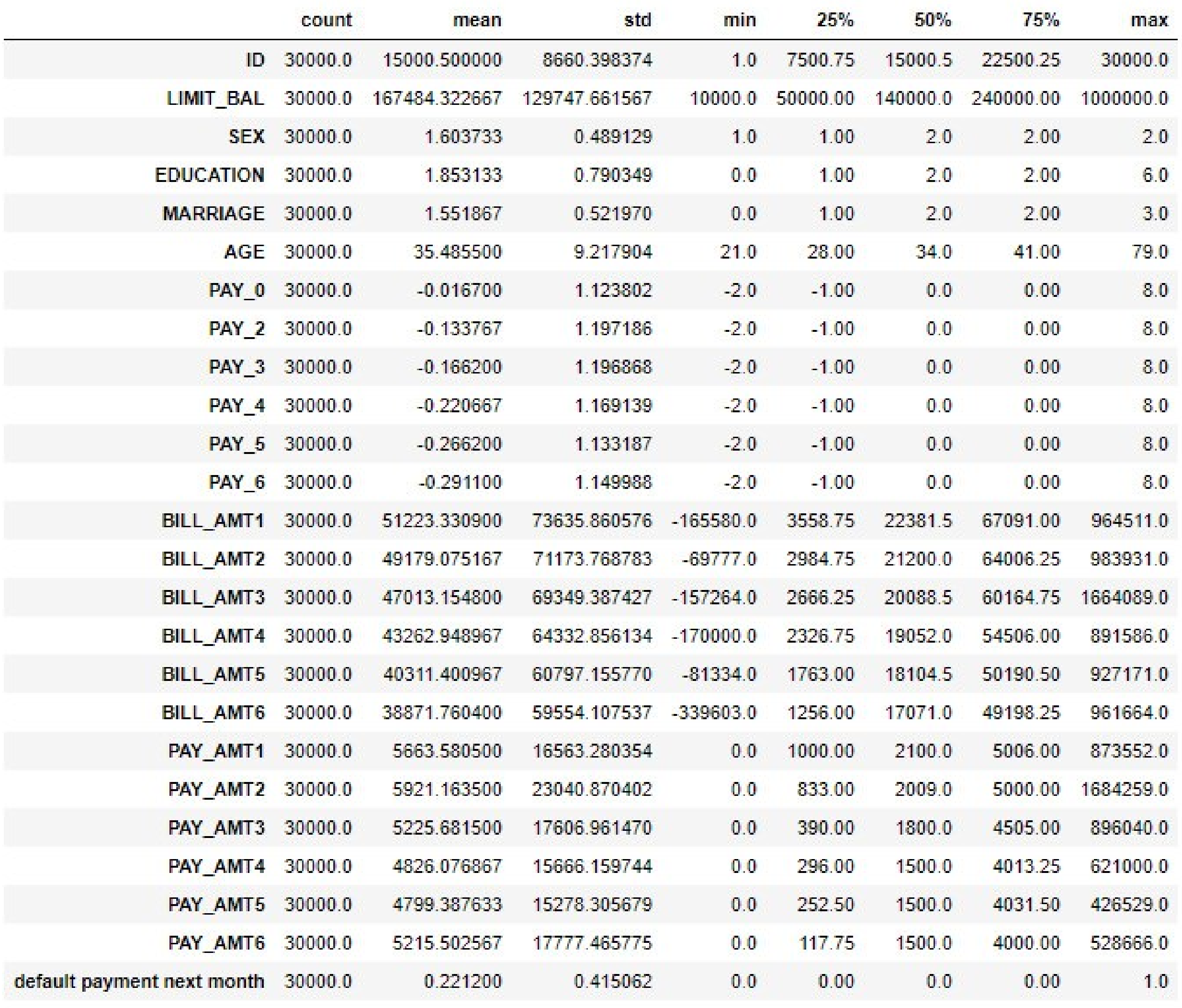
df.info()



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
ID                            30000 non-null int64
LIMIT_BAL                     30000 non-null int64
SEX                           30000 non-null int64
EDUCATION                     30000 non-null int64
MARRIAGE                      30000 non-null int64
AGE                           30000 non-null int64
PAY_0                         30000 non-null int64
PAY_2                         30000 non-null int64
PAY_3                         30000 non-null int64
PAY_4                         30000 non-null int64
PAY_5                         30000 non-null int64
PAY_6                         30000 non-null int64
BILL_AMT1                     30000 non-null int64
BILL_AMT2                     30000 non-null int64
BILL_AMT3                     30000 non-null int64
BILL_AMT4                     30000 non-null int64
BILL_AMT5                     30000 non-null int64
BILL_AMT6                     30000 non-null int64
PAY_AMT1                      30000 non-null int64
PAY_AMT2                      30000 non-null int64
PAY_AMT3                      30000 non-null int64
PAY_AMT4                      30000 non-null int64
PAY_AMT5                      30000 non-null int64
PAY_AMT6                      30000 non-null int64
default payment next month    30000 non-null int64
dtypes: int64(25)
memory usage: 5.7 MB
```

4. Check the descriptive statistics for the numerical columns in your dataset:

df.describe().T

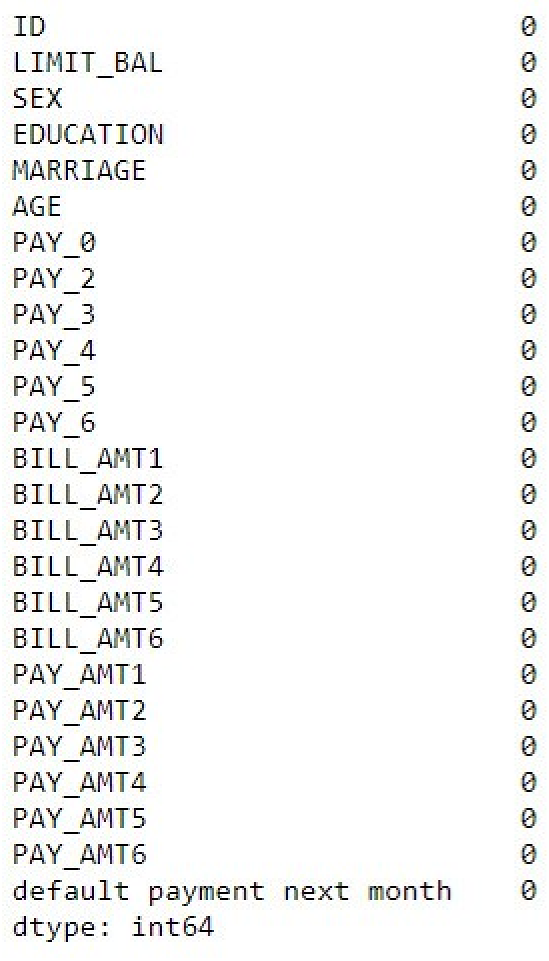


	count	mean	std	min	25%	50%	75%	max
ID	30000.0	15000.500000	8660.398374	1.0	7500.75	15000.5	22500.25	30000.0
LIMIT_BAL	30000.0	167484.322667	129747.661567	10000.0	50000.00	140000.0	240000.00	1000000.0
SEX	30000.0	1.603733	0.489129	1.0	1.00	2.0	2.00	2.0
EDUCATION	30000.0	1.853133	0.790349	0.0	1.00	2.0	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_0	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
default payment next month	30000.0	0.221200	0.415062	0.0	0.00	0.0	0.00	1.0

5. Check for empty values in your data:

# Checking for Null Values

df.isnull().sum()


```
ID                            0
LIMIT_BAL                     0
SEX                           0
EDUCATION                     0
MARRIAGE                      0
AGE                           0
PAY_0                         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
default payment next month    0
dtype: int64
```

ANALYZE WHAT YOU CAN INFER FROM THE DATA BASED ON WHAT YOU HAVE SEEN SO FAR. DOCUMENT YOUR ANALYSIS.

PRE-PROCESS DATA

This section deals with data preprocessing before heading into exploratory data analysis. The purpose behind doing preprocessing is that the data has to be cleaned of any errors such as spellings, along with identifying the unique values in a column and making the data more meaningful by clubbing the data to form groups. In data preprocessing, we also look at data consistency, for example, a categorical column such as SEX (0: Female, 1: Male) is displayed as an integer and so on. Without data preprocessing, visualizing the data or building a machine learning model becomes very difficult.

Before proceeding onto univariate analysis, let's look at the unique values in the columns. The motive behind looking at the unique values in a column is to identify the subcategory in each column. By knowing the subcategory in each column, we would be in a position to understand which subcategory has a higher count or vice versa. For example, let's take the EDUCATION column. We are interested in finding what the different subcategories in the EDUCATION column are and which subcategory has the higher count; that is, do our customers have their highest education as College or University?

6. Print the unique values in the SEX column:

```
print('SEX ' + str(sorted(df['SEX'].unique())))
```

7. Print the unique values in the EDUCATION column:

```
print('EDUCATION ' + str(sorted(df['EDUCATION'].unique())))
```

8. Print the unique values in the MARRIAGE column:

```
print('MARRIAGE ' + str(sorted(df['MARRIAGE'].unique())))
```

9. Please do the same for the PAY_0 and default.payment.next.month columns.
10. The EDUCATION column has 7 unique values, but as per our data description, we have only 4 unique values, so we are going to combine categories 0, 5, and 6 with category 4:

```
fill = (df.EDUCATION == 0) | (df.EDUCATION == 5) \
    | (df.EDUCATION == 6)
df.loc[fill, 'EDUCATION'] = 4
print('EDUCATION ' + str(sorted(df['EDUCATION'].unique())))
```

11. Similarly, in the MARRIAGE column, according to the data description, we should have 3 unique values. But here, we have 4 values in our data. As per our data description, the MARRIAGE column should have three subcategories. So, we combine category 0 with category 2 (Single):

```
fill = (df.MARRIAGE == 0)

df.loc[fill, 'MARRIAGE'] = 2

print('MARRIAGE ' + str(sorted(df['MARRIAGE'].unique())))
```

12. Rename the PAY_0 column to PAY_1 and the default payment next month column to DEFAULT to maintain consistency with the naming of other columns:

```
df = df.rename(columns={'default payment next month': 'DEFAULT', \

                 'PAY_0': 'PAY_1'})

df.head()
```

Data Science / Analytic Steps

The majority of time in a data science project is spent on Exploratory Data Analysis (EDA). In EDA, we investigate data to find hidden patterns and outliers with the help of visualization. By performing EDA, we can uncover the underlying structure of data and test our hypotheses with the help of summary statistics. We can split EDA into three parts:

Univariate analysis

Bivariate analysis

Correlation

Univariate Analysis

Univariate analysis is the simplest form of analysis where we analyze each feature (that is, each column of a DataFrame) and try to uncover the pattern or distribution of the data.

In univariate analysis, we will be analyzing the categorical columns (DEFAULT, SEX, EDUCATION, and MARRIAGE) to mine useful information about the data:

13. Let's look at the count of the DEFAULT column by drawing the count plot from the seaborn library:

```
sns.countplot(x="DEFAULT", data=df)
```

14. Let’s look at the numbers:

```
df['DEFAULT'].value_counts()
```

WHAT CAN YOU INFER? How many customers defaulted? What is the percentage of customers who defaulted?

15. Do the same analysis with SEX, EDUCATION, and MARRIAGE

Bivariate Analysis

Bivariate analysis is performed between two variables to look at their relationship.

In this section, you will consider the relationship between the DEFAULT column and other columns in the DataFrame with the help of the crosstab function and visualization techniques.

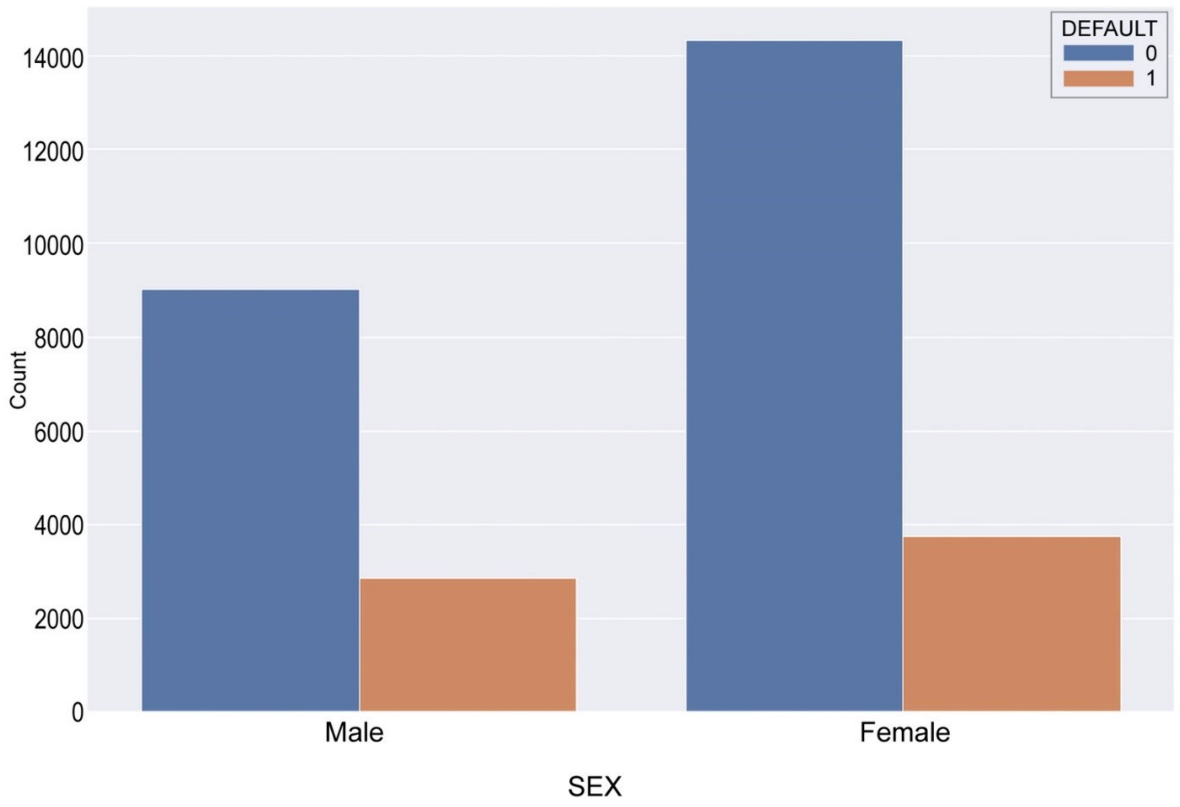
16. In this section, you will look at the relationship between the SEX and DEFAULT columns by plotting a count plot with the hue as DEFAULT to compare the number of male customers who have defaulted with the number of female customers who have defaulted:

sns.set(rc={'figure.figsize':(15,10)})

edu = sns.countplot(x='SEX', hue='DEFAULT', data=df)

edu.set_xticklabels(['Male','Female'])

plt.show()



From the preceding graph, you can see that females have defaulted more than males. But this graph doesn't show us the complete picture. To determine what percentage of each sex has defaulted, we will perform cross-tabulation.

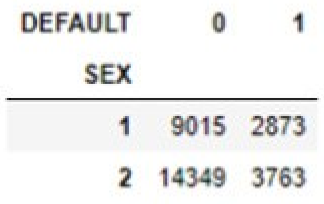
Cross-tabulation is a technique used to show the relationship between two or more categorical values. For example, in this scenario, we would like to find the relationship between DEFAULT and SEX. A crosstab table will show you the count of customers for each of the following combinations:

1	Male and defaulted
2	Male and hasn’t defaulted
3	Female and defaulted
4	Female and hasn't defaulted

17. Do a cross tabulation:

pd.crosstab(df.SEX,df.DEFAULT, margins=True)

The output of a cross-tabulation will look similar to this:

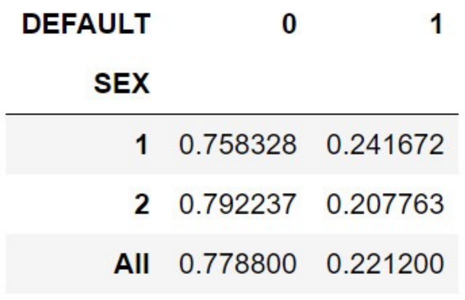


```
DEFAULT      0     1
    SEX
      1   9015  2873
      2  14349  3763
```

The preceding table can be used as a sample. In this table, we can see that SEX subcategory 1 has 9015 people who have not defaulted (DEFAULT :0) and 2873 people who have defaulted, while subcategory 2 in SEX has 14349 people who have not defaulted and 3763 people who have defaulted.

18. Normalize the results to get the most accurate insights from your data:

pd.crosstab(df.SEX,df.DEFAULT,normalize='index',margins=True)

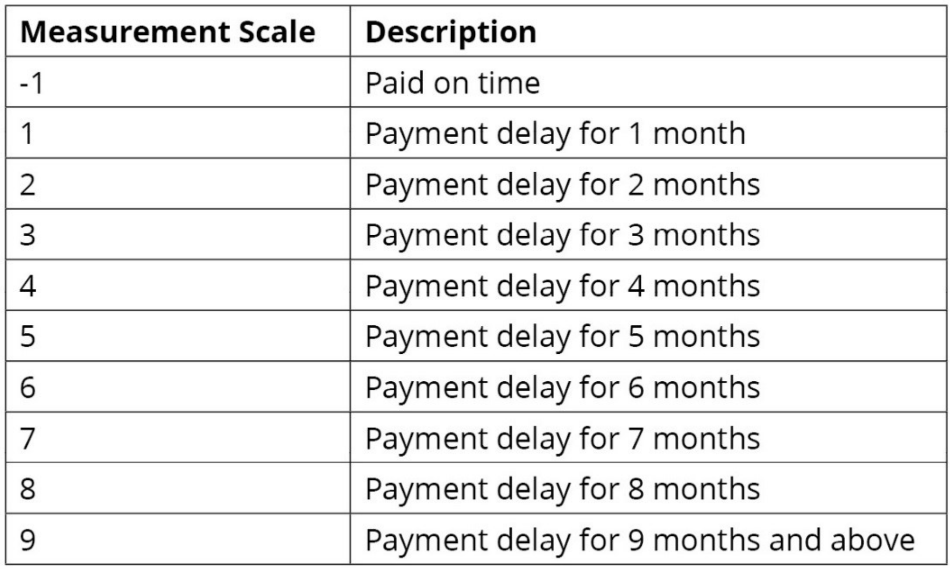


```
DEFAULT         0         1
    SEX
      1  0.758328  0.241672
      2  0.792237  0.207763
    All  0.778800  0.221200
```

As you can see, around 24% of male customers have defaulted and around 20% of female customers have defaulted.

19. On your own, analyze the relationship between the DEFAULT column and EDUCATION and the DEFAULT column and MARRIAGE and the DEFAULT column and AGE. Write down your insights.
20. Analyze PAY_1 versus Default

Here is the scale:



Measurement Scale	Description
-1	Paid on time
1	Payment delay for 1 month
2	Payment delay for 2 months
3	Payment delay for 3 months
4	Payment delay for 4 months
5	Payment delay for 5 months
6	Payment delay for 6 months
7	Payment delay for 7 months
8	Payment delay for 8 months
9	Payment delay for 9 months and above

21. Create a correlation of the features:

```
sns.set(rc={'figure.figsize':(30,10)})

sns.set_context("talk", font_scale=0.7)

sns.heatmap(df.iloc[:,1:].corr(method='spearman'), \

        cmap='rainbow_r', annot=True)
```

22. Get the exact values:

```
df.drop("DEFAULT", axis=1)\

.apply(lambda x: x.corr(df.DEFAULT,method='spearman'))
```

Jot down the results of your analysis and provide a profile for a high-risk customer.