eda-and-data-pre-processing

April 1, 2024

Predictive Analysis on Credit Card Defaults Based on Demographic Factors and Payment Behaviour

CIND 820

Project by: Md Fahim Ferdous ID: 501232653

1.0 Data Pre-processing and balancing

1.1 Import the dataset into colab

[ ]:
```
!pip3 install -U ucimlrepo
```

```
Collecting ucimlrepo
  Downloading ucimlrepo-0.0.6-py3-none-any.whl (8.0 kB)
Installing collected packages: ucimlrepo
Successfully installed ucimlrepo-0.0.6
```

[ ]:
```
from ucimlrepo import fetch_ucirepo

# fetch dataset
default_of_credit_card_clients = fetch_ucirepo(id=350)

# data (as pandas dataframes)
X = default_of_credit_card_clients.data.features
y = default_of_credit_card_clients.data.targets

# metadata
print(default_of_credit_card_clients.metadata)

# variable information
print(default_of_credit_card_clients.variables)
```


```
{'uci_id': 350, 'name': 'Default of Credit Card Clients', 'repository_url':
'https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients',
'data_url': 'https://archive.ics.uci.edu/static/public/350/data.csv',
'abstract': "This research aimed at the case of customers' default payments in
Taiwan and compares the predictive accuracy of probability of default among six
data mining methods.", 'area': 'Business', 'tasks': ['Classification'],
'characteristics': ['Multivariate'], 'num_instances': 30000, 'num_features': 23,
'feature_types': ['Integer', 'Real'], 'demographics': ['Sex', 'Education Level',
'Marital Status', 'Age'], 'target_col': ['Y'], 'index_col': ['ID'],
```

'has_missing_values': 'no', 'missing_values_symbol': None,
'year_of_dataset_creation': 2009, 'last_updated': 'Fri Mar 29 2024',
'dataset_doi': '10.24432/C55S3H', 'creators': ['I-Cheng Yeh'], 'intro_paper':
{'title': 'The comparisons of data mining techniques for the predictive accuracy
of probability of default of credit card clients', 'authors': 'I. Yeh, Che-hui
Lien', 'published_in': 'Expert systems with applications', 'year': 2009, 'url':
'https://www.semanticscholar.org/paper/1cacac4f0ea9fdff3cd88c151c94115a9fddcf33'
, 'doi': '10.1016/j.eswa.2007.12.020'}, 'additional_info': {'summary': "This
research aimed at the case of customers' default payments in Taiwan and compares
the predictive accuracy of probability of default among six data mining methods.
From the perspective of risk management, the result of predictive accuracy of
the estimated probability of default will be more valuable than the binary
result of classification - credible or not credible clients. Because the real
probability of default is unknown, this study presented the novel Sorting
Smoothing Method to estimate the real probability of default. With the real
probability of default as the response variable (Y), and the predictive
probability of default as the independent variable (X), the simple linear
regression result (Y = A + BX) shows that the forecasting model produced by
artificial neural network has the highest coefficient of determination; its
regression intercept (A) is close to zero, and regression coefficient (B) to
one. Therefore, among the six data mining techniques, artificial neural network
is the only one that can accurately estimate the real probability of default.",
'purpose': None, 'funded_by': None, 'instances_represent': None,
'recommended_data_splits': None, 'sensitive_data': None,
'preprocessing_description': None, 'variable_info': 'This research employed a
binary variable, default payment (Yes = 1, No = 0), as the response variable.
This study reviewed the literature and used the following 23 variables as
explanatory variables:\r\nX1: Amount of the given credit (NT dollar): it
includes both the individual consumer credit and his/her family (supplementary)
credit.\r\nX2: Gender (1 = male; 2 = female).\r\nX3: Education (1 = graduate
school; 2 = university; 3 = high school; 4 = others).\r\nX4: Marital status (1 =
married; 2 = single; 3 = others).\r\nX5: Age (year).\r\nX6 - X11: History of
past payment. We tracked the past monthly payment records (from April to
September, 2005) as follows: X6 = the repayment status in September, 2005; X7 =
the repayment status in August, 2005; . . .;X11 = the repayment status in April,
2005. The measurement scale for the repayment status is: -1 = pay duly; 1 =
payment delay for one month; 2 = payment delay for two months; . . .; 8 =
payment delay for eight months; 9 = payment delay for nine months and
above.\r\nX12-X17: Amount of bill statement (NT dollar). X12 = amount of bill
statement in September, 2005; X13 = amount of bill statement in August, 2005; .
. .; X17 = amount of bill statement in April, 2005. \r\nX18-X23: Amount of
previous payment (NT dollar). X18 = amount paid in September, 2005; X19 = amount
paid in August, 2005; . . .;X23 = amount paid in April, 2005.\r\n', 'citation':
None}}

```
   name     role     type      demographic                 description units  \
0    ID       ID  Integer             None                        None  None
1    X1  Feature  Integer             None                   LIMIT_BAL  None
2    X2  Feature  Integer              Sex                         SEX  None
```

```
3    X3  Feature  Integer  Education Level                        EDUCATION  None
4    X4  Feature  Integer   Marital Status                         MARRIAGE  None
5    X5  Feature  Integer              Age                              AGE  None
6    X6  Feature  Integer             None                            PAY_0  None
7    X7  Feature  Integer             None                            PAY_2  None
8    X8  Feature  Integer             None                            PAY_3  None
9    X9  Feature  Integer             None                            PAY_4  None
10  X10  Feature  Integer             None                            PAY_5  None
11  X11  Feature  Integer             None                            PAY_6  None
12  X12  Feature  Integer             None                        BILL_AMT1  None
13  X13  Feature  Integer             None                        BILL_AMT2  None
14  X14  Feature  Integer             None                        BILL_AMT3  None
15  X15  Feature  Integer             None                        BILL_AMT4  None
16  X16  Feature  Integer             None                        BILL_AMT5  None
17  X17  Feature  Integer             None                        BILL_AMT6  None
18  X18  Feature  Integer             None                         PAY_AMT1  None
19  X19  Feature  Integer             None                         PAY_AMT2  None
20  X20  Feature  Integer             None                         PAY_AMT3  None
21  X21  Feature  Integer             None                         PAY_AMT4  None
22  X22  Feature  Integer             None                         PAY_AMT5  None
23  X23  Feature  Integer             None                         PAY_AMT6  None
24    Y   Target   Binary             None  default payment next month  None

   missing_values
0              no
1              no
2              no
3              no
4              no
5              no
6              no
7              no
8              no
9              no
10             no
11             no
12             no
13             no
14             no
15             no
16             no
17             no
18             no
19             no
20             no
21             no
22             no
23             no
```

```
24               no
```

1.2 Data observation and anomaly finding

1.2.1 Checking Anomaly and missing data

[ ]:
```python
import pandas as pd

# URL of the dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/00350/
 ↪default%20of%20credit%20card%20clients.xls"

# Read the Excel file from the URL into a DataFrame
df = pd.read_excel(url, header=1)
#To verify if there is missing or anomalous data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ID                          30000 non-null  int64
 1   LIMIT_BAL                   30000 non-null  int64
 2   SEX                         30000 non-null  int64
 3   EDUCATION                   30000 non-null  int64
 4   MARRIAGE                    30000 non-null  int64
 5   AGE                         30000 non-null  int64
 6   PAY_0                       30000 non-null  int64
 7   PAY_2                       30000 non-null  int64
 8   PAY_3                       30000 non-null  int64
 9   PAY_4                       30000 non-null  int64
 10  PAY_5                       30000 non-null  int64
 11  PAY_6                       30000 non-null  int64
 12  BILL_AMT1                   30000 non-null  int64
 13  BILL_AMT2                   30000 non-null  int64
 14  BILL_AMT3                   30000 non-null  int64
 15  BILL_AMT4                   30000 non-null  int64
 16  BILL_AMT5                   30000 non-null  int64
 17  BILL_AMT6                   30000 non-null  int64
 18  PAY_AMT1                    30000 non-null  int64
 19  PAY_AMT2                    30000 non-null  int64
 20  PAY_AMT3                    30000 non-null  int64
 21  PAY_AMT4                    30000 non-null  int64
 22  PAY_AMT5                    30000 non-null  int64
 23  PAY_AMT6                    30000 non-null  int64
 24  default payment next month  30000 non-null  int64
dtypes: int64(25)
memory usage: 5.7 MB
```

1.2.2 Checking data by category

[ ]:
```
print(df['SEX'].value_counts()[[1,2]])
print(df['MARRIAGE'].value_counts())
print(df['EDUCATION'].value_counts())
pay_counts = df[['PAY_0','PAY_2','PAY_3','PAY_4','PAY_5', 'PAY_6']].
 ↪apply(lambda x: x.value_counts())

# Print the result
print(pay_counts)
```

```
1    11888
2    18112
Name: SEX, dtype: int64
2    15964
1    13659
3      323
0       54
Name: MARRIAGE, dtype: int64
2    14030
1    10585
3     4917
5      280
4      123
6       51
0       14
Name: EDUCATION, dtype: int64
    PAY_0  PAY_2  PAY_3  PAY_4    PAY_5    PAY_6
-2   2759   3782   4085   4348   4546.0   4895.0
-1   5686   6050   5938   5687   5539.0   5740.0
 0  14737  15730  15764  16455  16947.0  16286.0
 1   3688     28      4      2      NaN      NaN
 2   2667   3927   3819   3159   2626.0   2766.0
 3    322    326    240    180    178.0    184.0
 4     76     99     76     69     84.0     49.0
 5     26     25     21     35     17.0     13.0
 6     11     12     23      5      4.0     19.0
 7      9     20     27     58     58.0     46.0
 8     19      1      3      2      1.0      2.0
```

[ ]:
```
df['MARRIAGE'].unique()
```

[ ]:
```
array([1, 2, 3, 0])
```

[ ]:
```
df['EDUCATION'].unique()
```

[ ]:
```
array([2, 1, 3, 5, 4, 6, 0])
```

[ ]:
```
df['SEX'].unique()
```

[ ]:
```
array([2, 1])
```

[ ]:
```
df.describe().T
```

[ ]:
```
                              count           mean            std       min  \
ID                          30000.0   15000.500000    8660.398374       1.0
LIMIT_BAL                   30000.0  167484.322667  129747.661567   10000.0
SEX                         30000.0       1.603733       0.489129       1.0
EDUCATION                   30000.0       1.853133       0.790349       0.0
MARRIAGE                    30000.0       1.551867       0.521970       0.0
AGE                         30000.0      35.485500       9.217904      21.0
PAY_0                       30000.0      -0.016700       1.123802      -2.0
PAY_2                       30000.0      -0.133767       1.197186      -2.0
PAY_3                       30000.0      -0.166200       1.196868      -2.0
PAY_4                       30000.0      -0.220667       1.169139      -2.0
PAY_5                       30000.0      -0.266200       1.133187      -2.0
PAY_6                       30000.0      -0.291100       1.149988      -2.0
BILL_AMT1                   30000.0   51223.330900   73635.860576 -165580.0
BILL_AMT2                   30000.0   49179.075167   71173.768783  -69777.0
BILL_AMT3                   30000.0   47013.154800   69349.387427 -157264.0
BILL_AMT4                   30000.0   43262.948967   64332.856134 -170000.0
BILL_AMT5                   30000.0   40311.400967   60797.155770  -81334.0
BILL_AMT6                   30000.0   38871.760400   59554.107537 -339603.0
PAY_AMT1                    30000.0    5663.580500   16563.280354       0.0
PAY_AMT2                    30000.0    5921.163500   23040.870402       0.0
PAY_AMT3                    30000.0    5225.681500   17606.961470       0.0
PAY_AMT4                    30000.0    4826.076867   15666.159744       0.0
PAY_AMT5                    30000.0    4799.387633   15278.305679       0.0
PAY_AMT6                    30000.0    5215.502567   17777.465775       0.0
default payment next month  30000.0       0.221200       0.415062       0.0

                                 25%       50%        75%        max
ID                           7500.75   15000.5   22500.25    30000.0
LIMIT_BAL                   50000.00  140000.0  240000.00  1000000.0
SEX                             1.00       2.0       2.00        2.0
EDUCATION                       1.00       2.0       2.00        6.0
MARRIAGE                        1.00       2.0       2.00        3.0
AGE                            28.00      34.0      41.00       79.0
PAY_0                          -1.00       0.0       0.00        8.0
PAY_2                          -1.00       0.0       0.00        8.0
PAY_3                          -1.00       0.0       0.00        8.0
PAY_4                          -1.00       0.0       0.00        8.0
PAY_5                          -1.00       0.0       0.00        8.0
PAY_6                          -1.00       0.0       0.00        8.0
BILL_AMT1                    3558.75   22381.5   67091.00   964511.0
```

```
BILL_AMT2                   2984.75   21200.0   64006.25   983931.0
BILL_AMT3                   2666.25   20088.5   60164.75  1664089.0
BILL_AMT4                   2326.75   19052.0   54506.00   891586.0
BILL_AMT5                   1763.00   18104.5   50190.50   927171.0
BILL_AMT6                   1256.00   17071.0   49198.25   961664.0
PAY_AMT1                    1000.00    2100.0    5006.00   873552.0
PAY_AMT2                     833.00    2009.0    5000.00  1684259.0
PAY_AMT3                     390.00    1800.0    4505.00   896040.0
PAY_AMT4                     296.00    1500.0    4013.25   621000.0
PAY_AMT5                     252.50    1500.0    4031.50   426529.0
PAY_AMT6                     117.75    1500.0    4000.00   528666.0
default payment next month     0.00       0.0       0.00        1.0
```

1.3 Data Cleaning

[ ]:
```
#remaning the data label for uniformity
df.rename(columns={'PAY_0':'PAY_1'}, inplace=True)
df.head()
```

[ ]:
```
   ID  LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_1  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

   …  BILL_AMT4  BILL_AMT5  BILL_AMT6  PAY_AMT1  PAY_AMT2  PAY_AMT3  \
0  …          0          0          0         0       689         0
1  …       3272       3455       3261         0      1000      1000
2  …      14331      14948      15549      1518      1500      1000
3  …      28314      28959      29547      2000      2019      1200
4  …      20940      19146      19131      2000     36681     10000

   PAY_AMT4  PAY_AMT5  PAY_AMT6  default payment next month
0         0         0         0                           1
1      1000         0      2000                           1
2      1000      1000      5000                           0
3      1100      1069      1000                           0
4      9000       689       679                           0

[5 rows x 25 columns]
```

[ ]:
```
#unlabeled data for 'MARRIAGE' and 'EDUCATION', 0 counted as missing
df1=df.loc[(df['EDUCATION']!=0)& (df['MARRIAGE']!=0)]
df1
```

[ ]:

```
          ID  LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_1  PAY_2  PAY_3  \
0          1      20000    2          2         1   24      2      2     -1
1          2     120000    2          2         2   26     -1      2      0
2          3      90000    2          2         2   34      0      0      0
3          4      50000    2          2         1   37      0      0      0
4          5      50000    1          2         1   57     -1      0     -1
…        …          …  …       …         …  …      …     …    …
29995  29996     220000    1          3         1   39      0      0      0
29996  29997     150000    1          3         2   43     -1     -1     -1
29997  29998      30000    1          2         2   37      4      3      2
29998  29999      80000    1          3         1   41      1     -1      0
29999  30000      50000    1          2         1   46      0      0      0

       PAY_4  …  BILL_AMT4  BILL_AMT5  BILL_AMT6  PAY_AMT1  PAY_AMT2  \
0         -1  …          0          0          0         0       689
1          0  …       3272       3455       3261         0      1000
2          0  …      14331      14948      15549      1518      1500
3          0  …      28314      28959      29547      2000      2019
4          0  …      20940      19146      19131      2000     36681
…        …  …        …        …        …        …       …
29995      0  …      88004      31237      15980      8500     20000
29996     -1  …       8979       5190          0      1837      3526
29997     -1  …      20878      20582      19357         0         0
29998      0  …      52774      11855      48944     85900      3409
29999      0  …      36535      32428      15313      2078      1800

       PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6  default payment next month
0             0         0         0         0                           1
1          1000      1000         0      2000                           1
2          1000      1000      1000      5000                           0
3          1200      1100      1069      1000                           0
4         10000      9000       689       679                           0
…           …       …       …       …                         …
29995      5003      3047      5000      1000                           0
29996      8998       129         0         0                           0
29997     22000      4200      2000      3100                           1
29998      1178      1926     52964      1804                           1
29999      1430      1000      1000      1000                           1

[29932 rows x 25 columns]
```

[ ]:

```
#for 'EDUCATION' CONSIDERING 5 AND 6 UNDER CATEGORY 4
df1['EDUCATION'].replace({5:4,6:4},inplace=True)
df1['EDUCATION'].value_counts()
```

```
<ipython-input-11-64b42e0edc70>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['EDUCATION'].replace({5:4,6:4},inplace=True)
```

[ ]:
```
2    14024
1    10581
3     4873
4      454
Name: EDUCATION, dtype: int64
```

[ ]:
```python
#for PAY_1 to PAY_6, -1 means pay duely, so -1,0 and -2 has been adjusted to 0,␣
↪indicating paid duely
df1['PAY_1'].replace({-2:0,-1:0,0:0},inplace=True)
df1['PAY_2'].replace({-2:0,-1:0,0:0},inplace=True)
df1['PAY_3'].replace({-2:0,-1:0,0:0},inplace=True)
df1['PAY_4'].replace({-2:0,-1:0,0:0},inplace=True)
df1['PAY_5'].replace({-2:0,-1:0,0:0},inplace=True)
df1['PAY_6'].replace({-2:0,-1:0,0:0},inplace=True)
df1.PAY_1.value_counts()
df2=df1
df2
```

```
<ipython-input-12-6c576ac002ed>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['PAY_1'].replace({-2:0,-1:0,0:0},inplace=True)
<ipython-input-12-6c576ac002ed>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['PAY_2'].replace({-2:0,-1:0,0:0},inplace=True)
<ipython-input-12-6c576ac002ed>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['PAY_3'].replace({-2:0,-1:0,0:0},inplace=True)
<ipython-input-12-6c576ac002ed>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['PAY_4'].replace({-2:0,-1:0,0:0},inplace=True)
<ipython-input-12-6c576ac002ed>:6: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['PAY_5'].replace({-2:0,-1:0,0:0},inplace=True)
<ipython-input-12-6c576ac002ed>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['PAY_6'].replace({-2:0,-1:0,0:0},inplace=True)
```

[ ]:

```
          ID  LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_1  PAY_2  PAY_3  \
0          1      20000    2          2         1   24      2      2      0
1          2     120000    2          2         2   26      0      2      0
2          3      90000    2          2         2   34      0      0      0
3          4      50000    2          2         1   37      0      0      0
4          5      50000    1          2         1   57      0      0      0
…        …          … …        …         … …     …     …    …
29995  29996     220000    1          3         1   39      0      0      0
29996  29997     150000    1          3         2   43      0      0      0
29997  29998      30000    1          2         2   37      4      3      2
29998  29999      80000    1          3         1   41      1      0      0
29999  30000      50000    1          2         1   46      0      0      0

       PAY_4  …  BILL_AMT4  BILL_AMT5  BILL_AMT6  PAY_AMT1  PAY_AMT2  \
0          0  …          0          0          0         0       689
1          0  …       3272       3455       3261         0      1000
2          0  …      14331      14948      15549      1518      1500
3          0  …      28314      28959      29547      2000      2019
4          0  …      20940      19146      19131      2000     36681
…        …  …        …          …          …         …         …
29995      0  …      88004      31237      15980      8500     20000
29996      0  …       8979       5190          0      1837      3526
29997      0  …      20878      20582      19357         0         0
29998      0  …      52774      11855      48944     85900      3409
29999      0  …      36535      32428      15313      2078      1800

       PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6  default payment next month
0             0         0         0         0                           1
1          1000      1000         0      2000                           1
2          1000      1000      1000      5000                           0
3          1200      1100      1069      1000                           0
4         10000      9000       689       679                           0
…          …       …        …       …                           …
29995      5003      3047      5000      1000                           0
29996      8998       129         0         0                           0
```

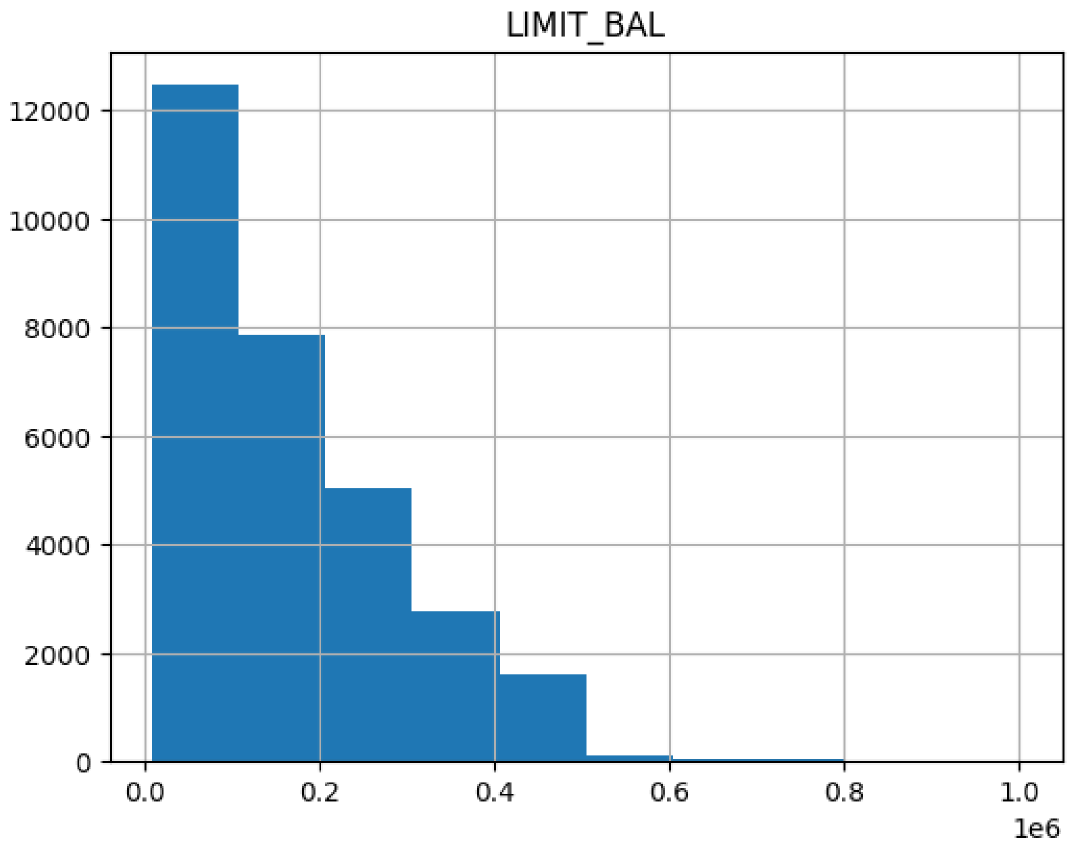
```
29997      22000       4200       2000       3100                             1
29998       1178       1926      52964       1804                             1
29999       1430       1000       1000       1000                             1

[29932 rows x 25 columns]
```

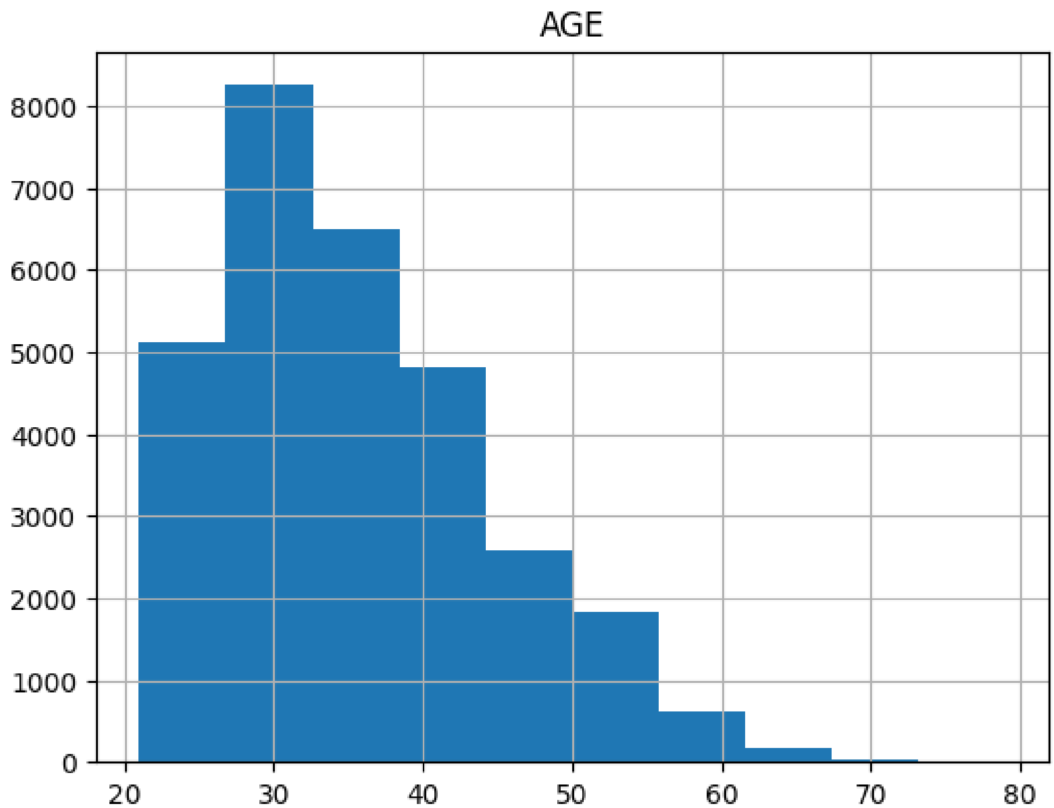
1.4 Exploratory Data Analysis

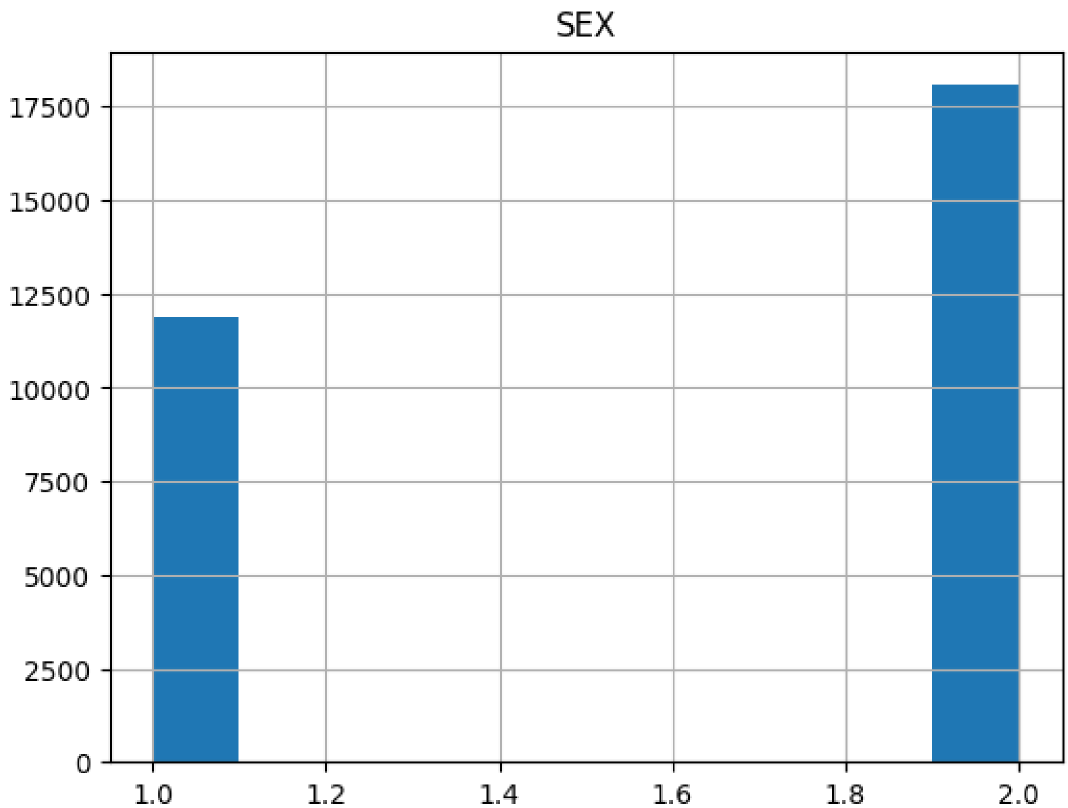
```
[ ]: #LIMIT_BAL BAR charts
print(df2[['LIMIT_BAL']].hist(bins=10, alpha=1))

#AGE BAR charts
print(df2[['AGE']].hist(bins=10, alpha=1))
#SEX BAR charts
print(df2[['SEX']].hist(bins=10, alpha=1))
#EDUCATION BAR charts
print(df2[['EDUCATION']].hist(bins=10, alpha=1))
#MARRIAGE BAR charts
print(df2[['MARRIAGE']].hist(bins=10, alpha=1))



#PAY_1 to PAY_6 BAR charts
print(df2[['PAY_1','PAY_2']].hist(bins=10, alpha=1))
print(df2[['PAY_3','PAY_4']].hist(bins=10, alpha=1))
print(df2[['PAY_5','PAY_5']].hist(bins=10, alpha=1))

#BILL_AMT1 to BILL_AMT6 BAR charts
print(df2[['BILL_AMT1','BILL_AMT2']].hist(bins=10, alpha=1))
print(df2[['BILL_AMT3','BILL_AMT4']].hist(bins=10, alpha=1))
print(df2[['BILL_AMT5','BILL_AMT6']].hist(bins=10, alpha=1))
#PAY_AMT1 to PAY_AMT6 BAR charts
print(df2[['PAY_AMT1','PAY_AMT2']].hist(bins=10, alpha=1))
print(df2[['PAY_AMT3','PAY_AMT4']].hist(bins=10, alpha=1))
print(df2[['PAY_AMT5','PAY_AMT6']].hist(bins=10, alpha=1))
```

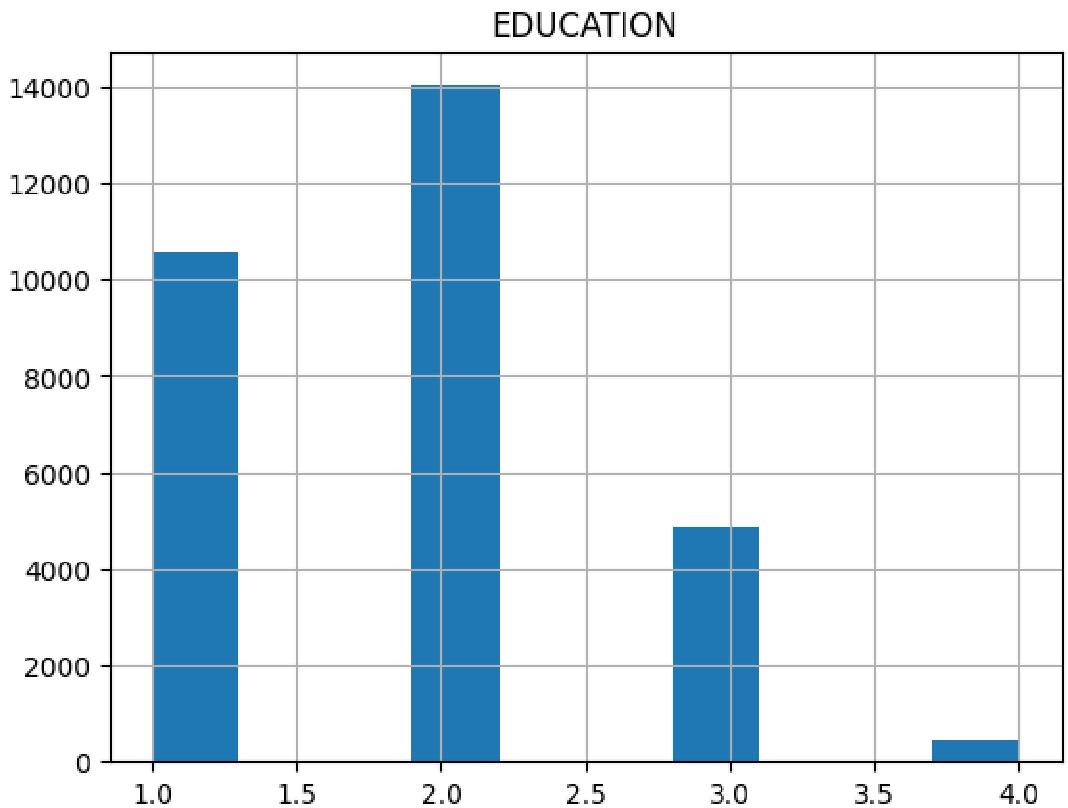
```
[[<Axes: title={'center': 'LIMIT_BAL'}>]]
[[<Axes: title={'center': 'AGE'}>]]
[[<Axes: title={'center': 'SEX'}>]]
[[<Axes: title={'center': 'EDUCATION'}>]]
[[<Axes: title={'center': 'MARRIAGE'}>]]
[[<Axes: title={'center': 'PAY_1'}> <Axes: title={'center': 'PAY_2'}>]]
[[<Axes: title={'center': 'PAY_3'}> <Axes: title={'center': 'PAY_4'}>]]
[[<Axes: title={'center': 'PAY_5'}> <Axes: title={'center': 'PAY_5'}>]]
[[<Axes: title={'center': 'BILL_AMT1'}>
  <Axes: title={'center': 'BILL_AMT2'}>]]
[[<Axes: title={'center': 'BILL_AMT3'}>
  <Axes: title={'center': 'BILL_AMT4'}>]]
[[<Axes: title={'center': 'BILL_AMT5'}>
```

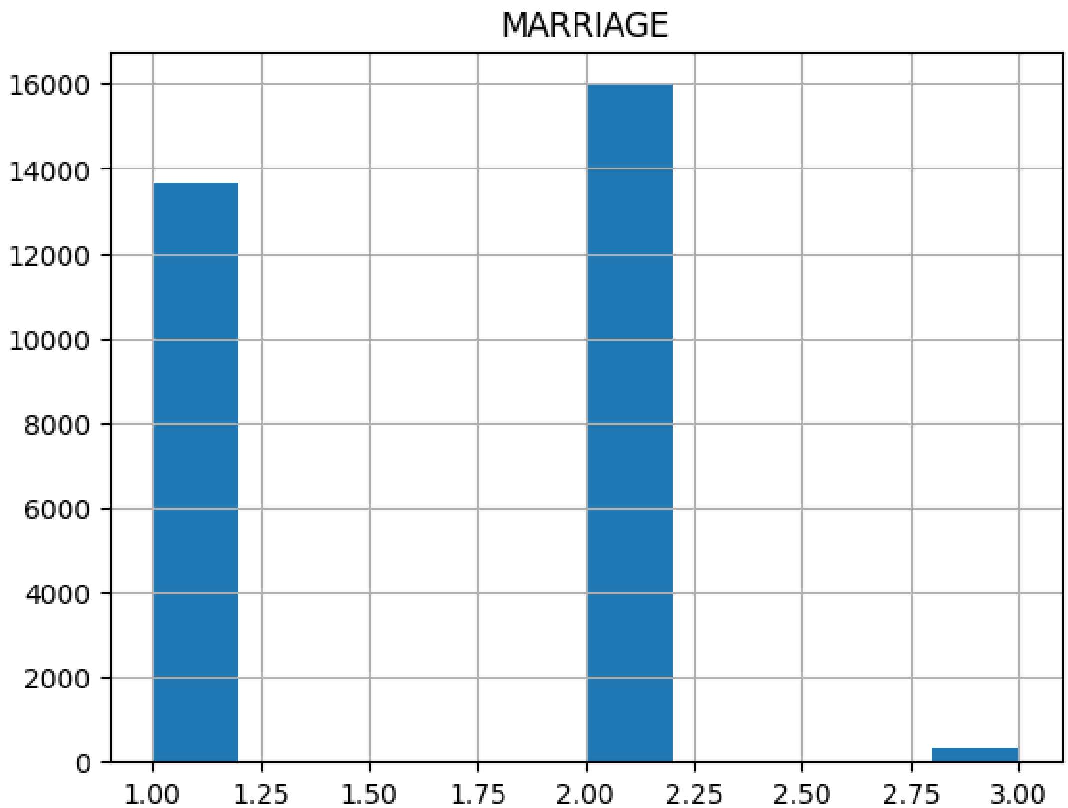
```
  <Axes: title={'center': 'BILL_AMT6'}>]]
[[<Axes: title={'center': 'PAY_AMT1'}>
  <Axes: title={'center': 'PAY_AMT2'}>]]
[[<Axes: title={'center': 'PAY_AMT3'}>
  <Axes: title={'center': 'PAY_AMT4'}>]]
[[<Axes: title={'center': 'PAY_AMT5'}>
  <Axes: title={'center': 'PAY_AMT6'}>]]
```

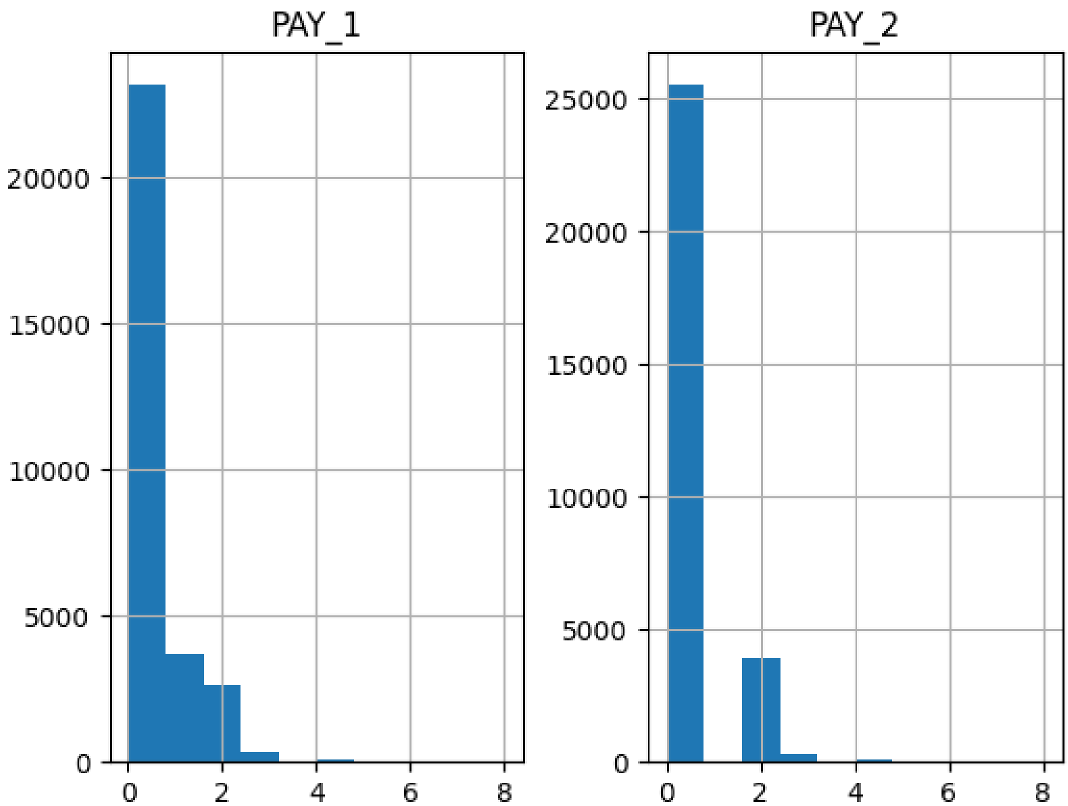


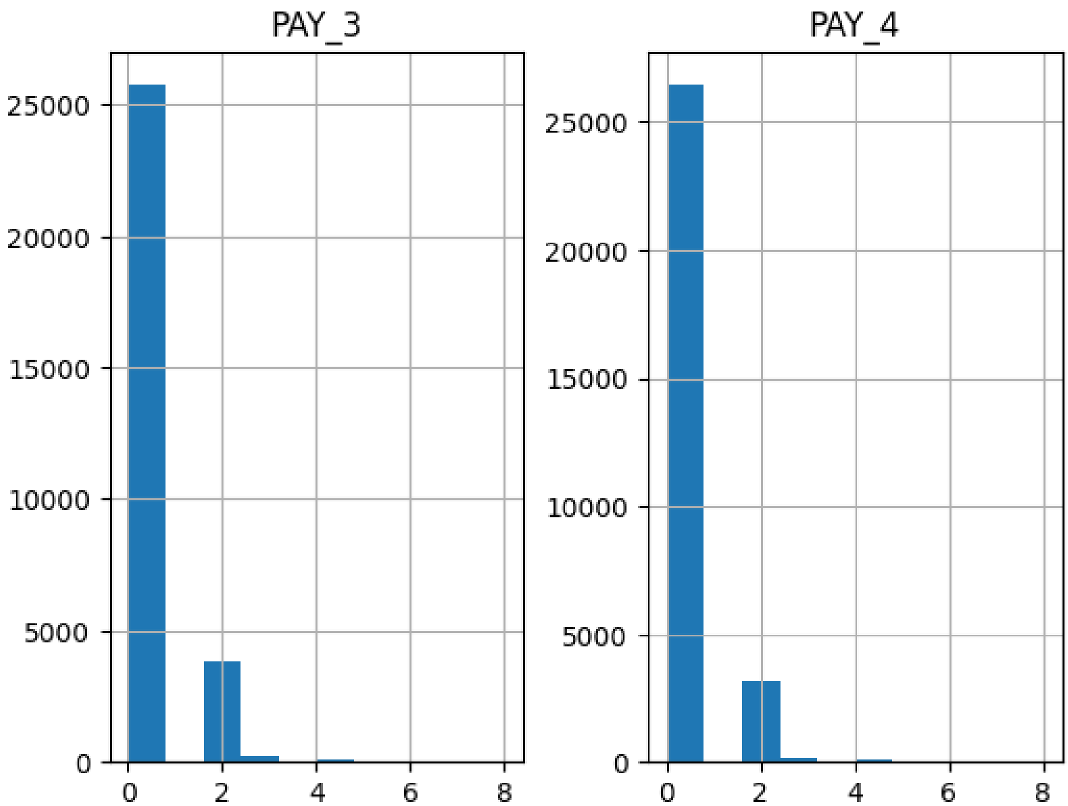

AGE
8000
7000
6000
5000
4000
3000
2000
1000
0
20
30
40
50
60
70
80

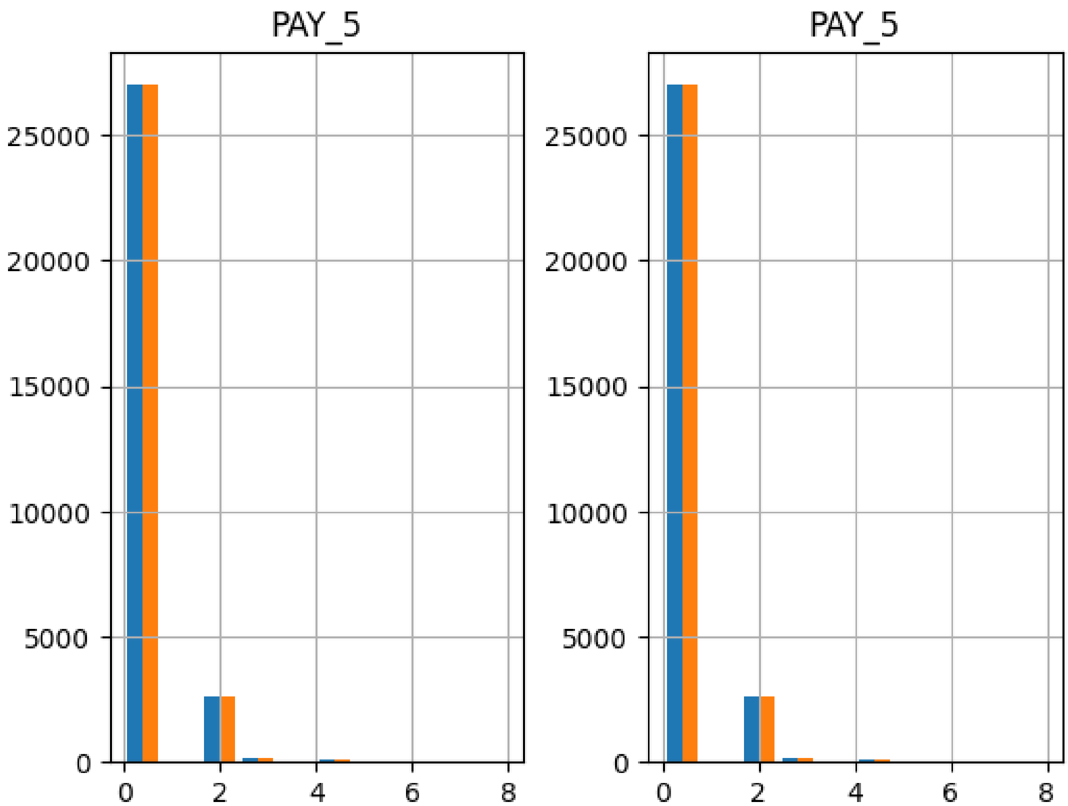

SEX
17500
15000
12500
10000
7500
5000
2500
0
1.0
1.2
1.4
1.6
1.8
2.0

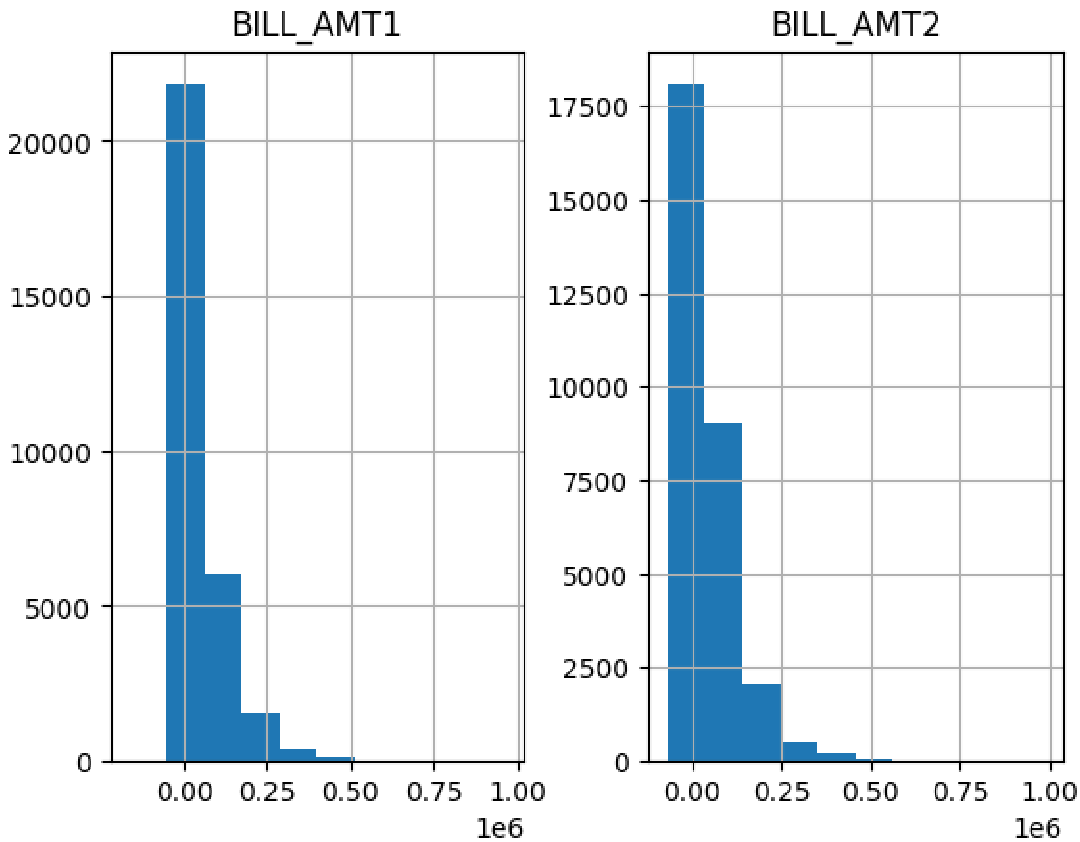

EDUCATION
14000
12000
10000
8000
6000
4000
2000
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0

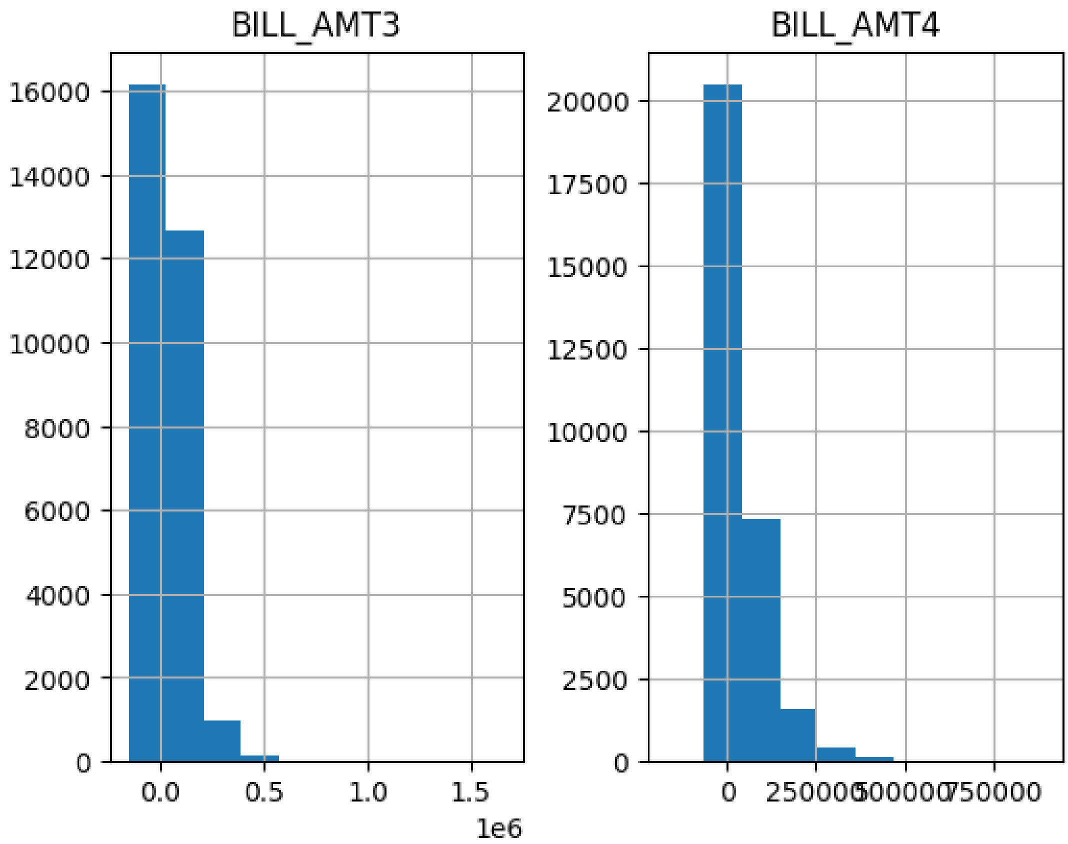

MARRIAGE
16000
14000
12000
10000
8000
6000
4000
2000
0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00

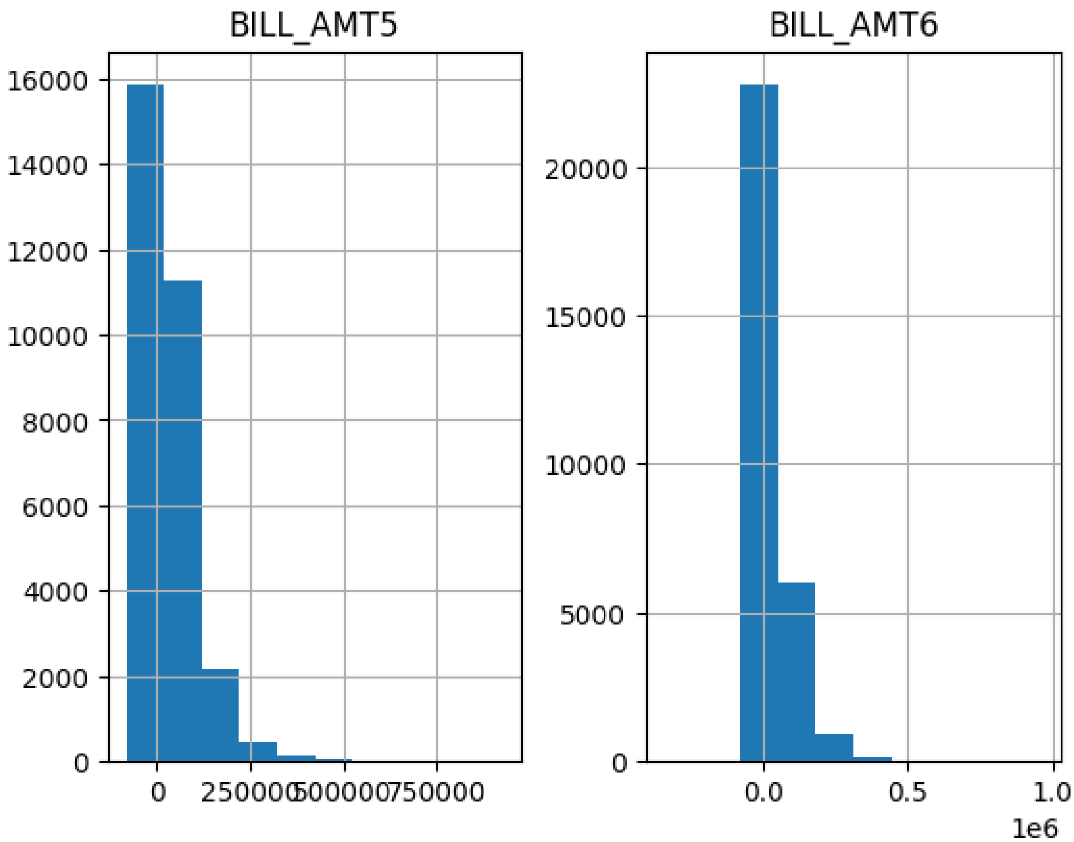

PAY_1
PAY_2
0
5000
10000
15000
20000
25000
0
2
4
6
8

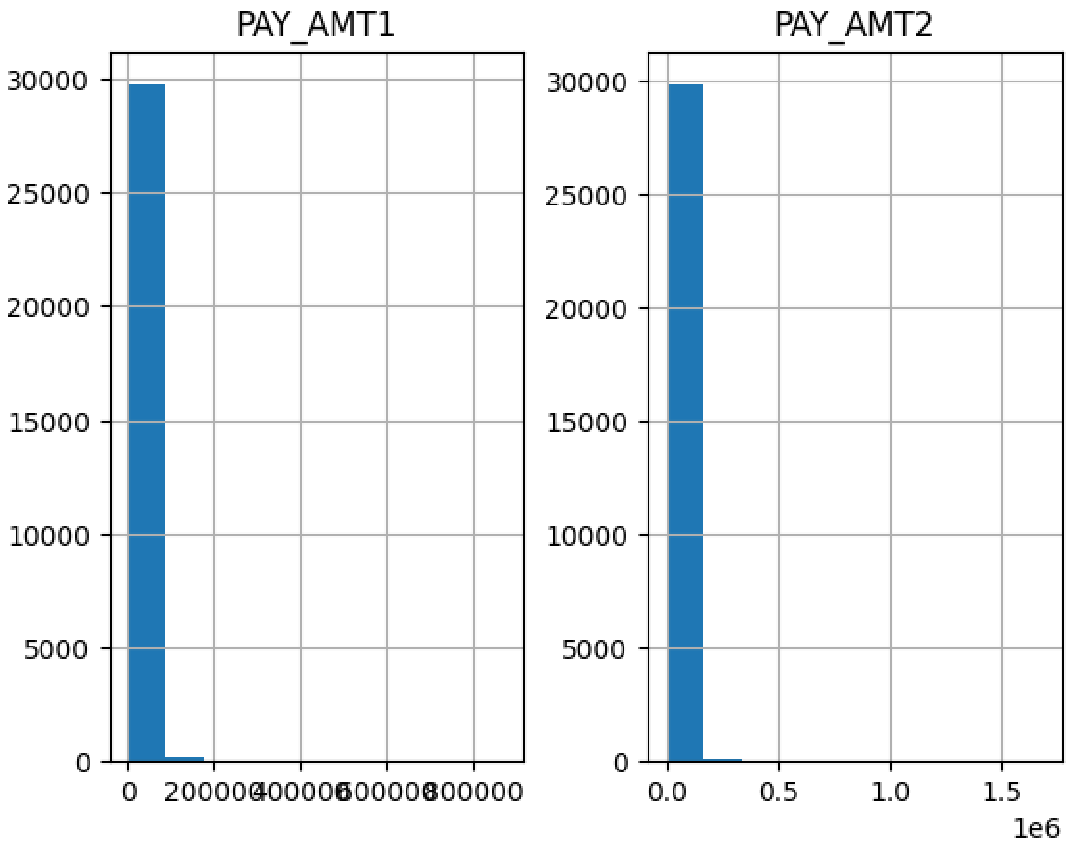

PAY_3
PAY_4
25000
20000
15000
10000
5000
0
0
2
4
6
8

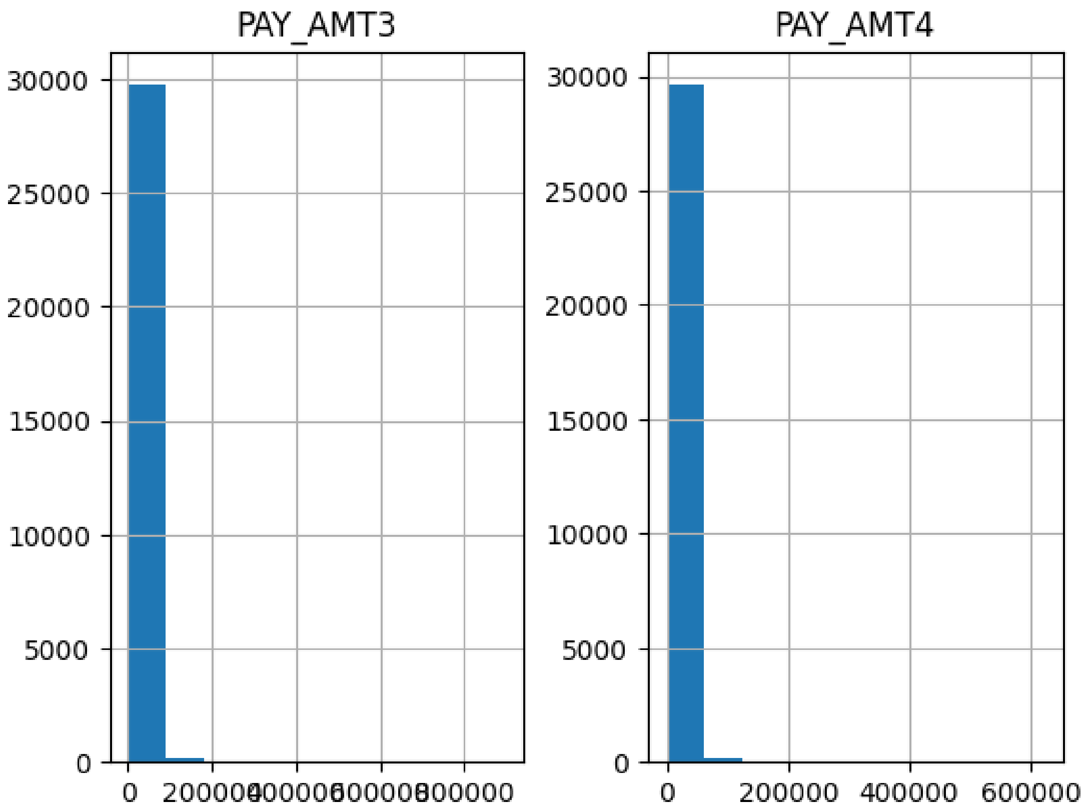

PAY_5
25000
20000
15000
10000
5000
0
0
2
4
6
8
PAY_5
25000
20000
15000
10000
5000
0
0
2
4
6
8

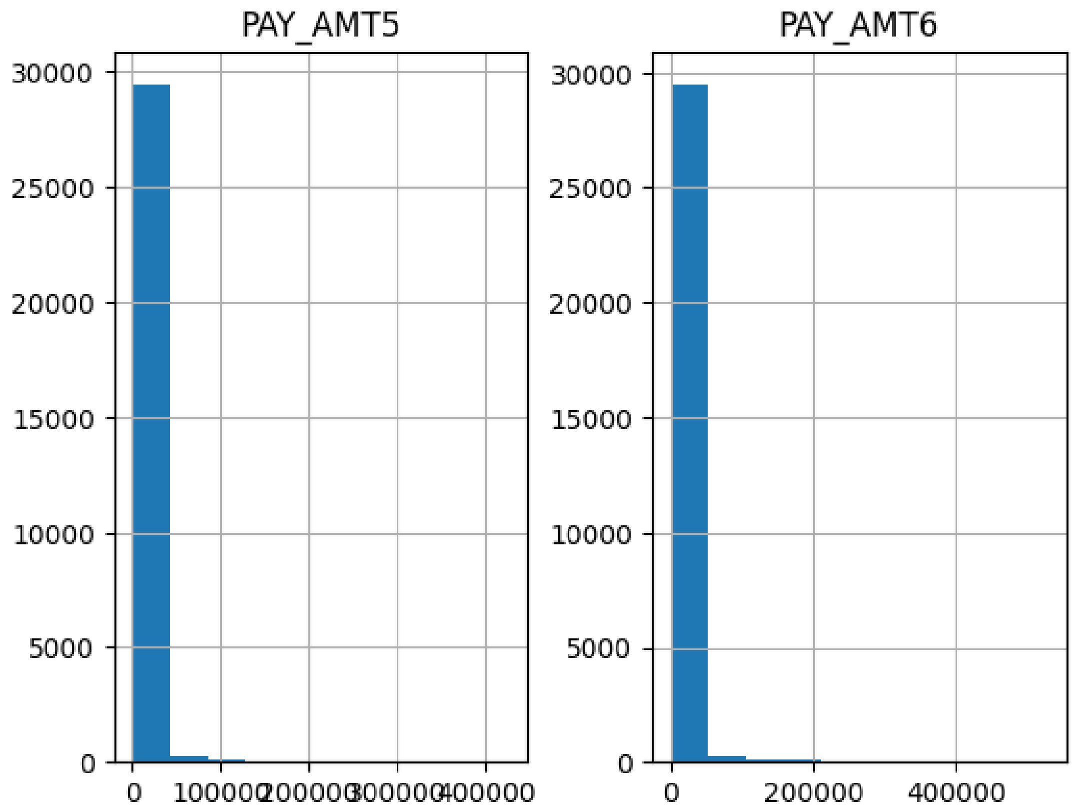

BILL_AMT1
20000
15000
10000
5000
0
0.00
0.25
0.50
0.75
1.00
1e6
BILL_AMT2
17500
15000
12500
10000
7500
5000
2500
0
0.00
0.25
0.50
0.75
1.00
1e6


BILL_AMT3
16000
14000
12000
10000
8000
6000
4000
2000
0
0.0
0.5
1.0
1.5
1e6
BILL_AMT4
20000
17500
15000
12500
10000
7500
5000
2500
0
0
250000
500000
750000


BILL_AMT5
16000
14000
12000
10000
8000
6000
4000
2000
0
0
250000
500000
750000
BILL_AMT6
20000
15000
10000
5000
0
0.0
0.5
1.0
1e6

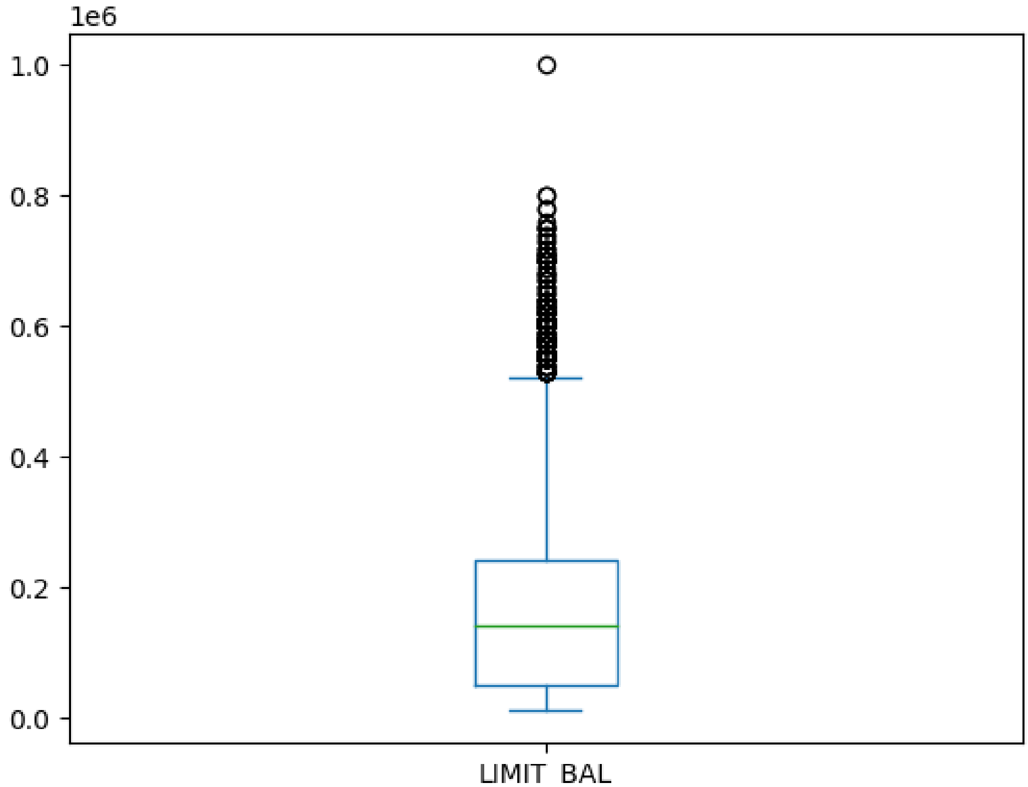

PAY_AMT1
30000
25000
20000
15000
10000
5000
0
0
200000
400000
600000
800000
PAY_AMT2
30000
25000
20000
15000
10000
5000
0
0.0
0.5
1.0
1.5
1e6


PAY_AMT3
30000
25000
20000
15000
10000
5000
0
0
200000
400000
600000
800000
PAY_AMT4
30000
25000
20000
15000
10000
5000
0
0
200000
400000
600000



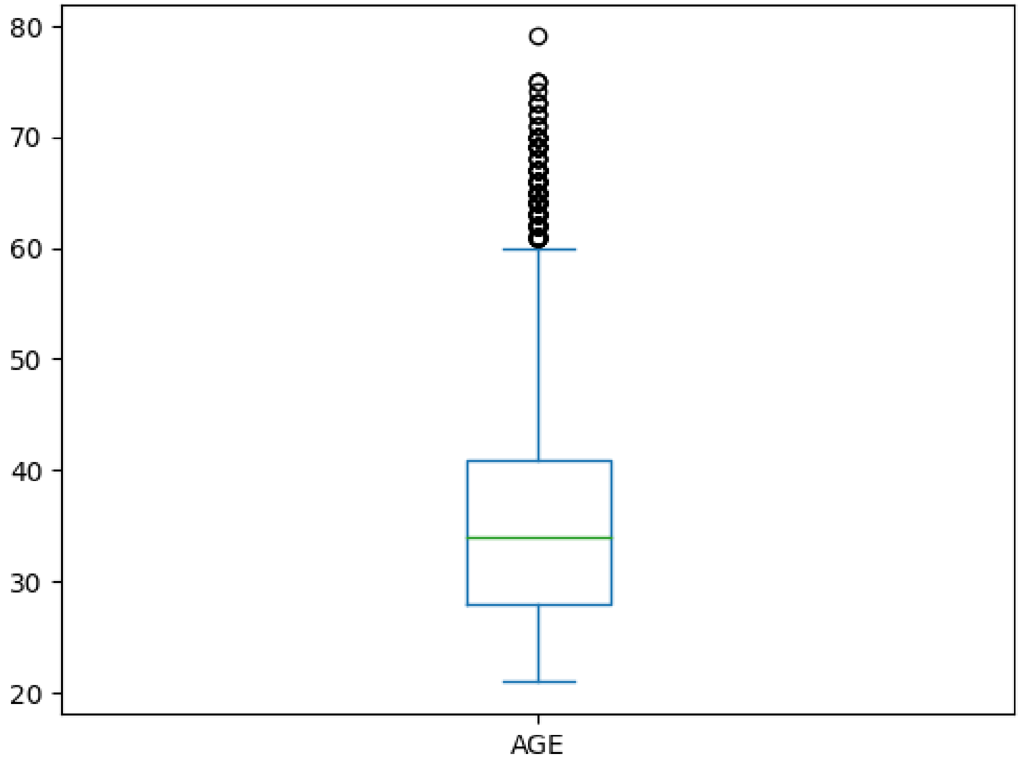
[ ]:
```
print(df2['LIMIT_BAL'].plot.box())
```

```
Axes(0.125,0.11;0.775x0.77)
```



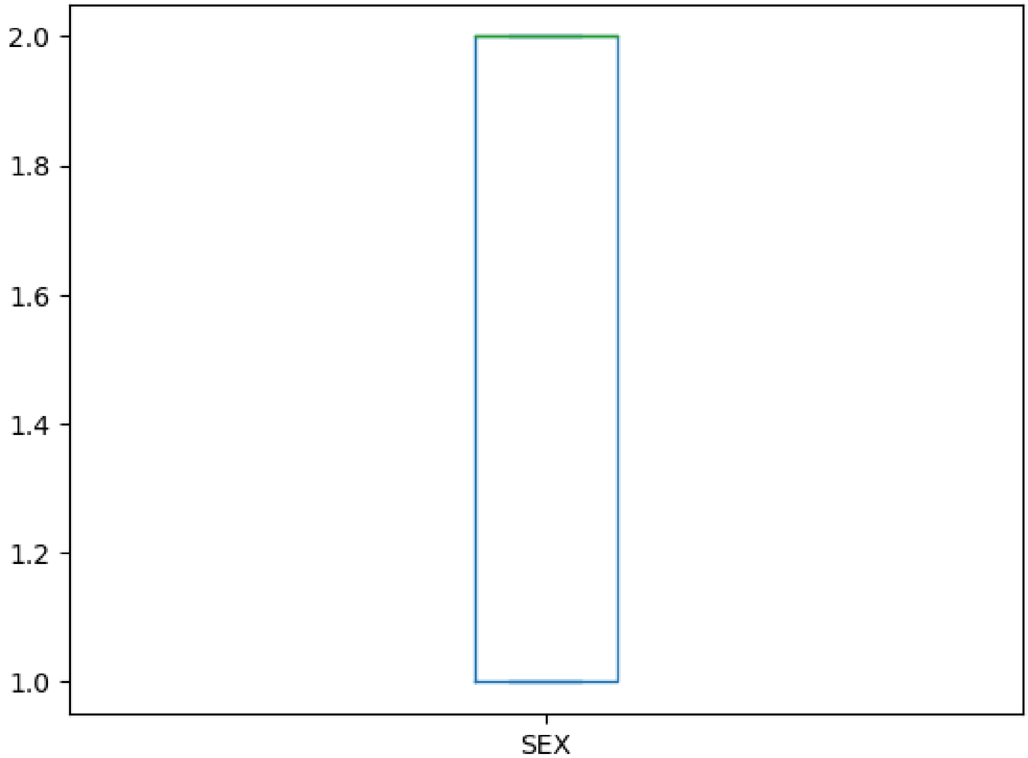
[ ]:
```
print(df2['AGE'].plot.box())
```

```
Axes(0.125,0.11;0.775x0.77)
```



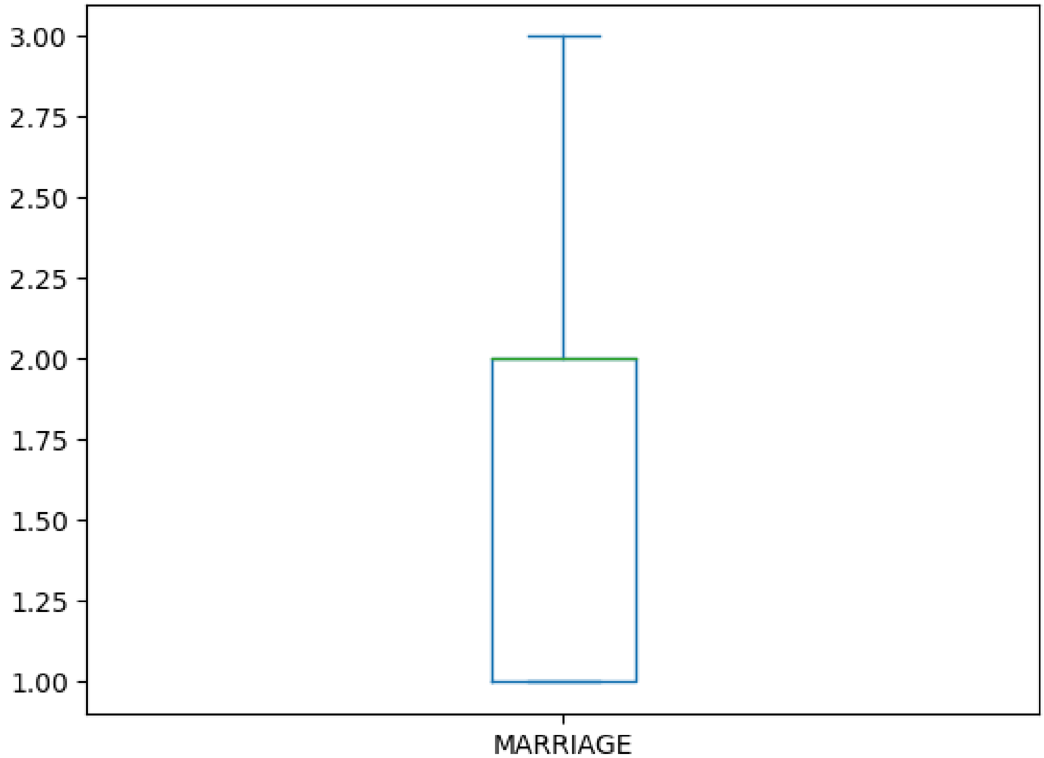

```
[ ]: print(df2['SEX'].plot.box())
```

Axes(0.125,0.11;0.775x0.77)



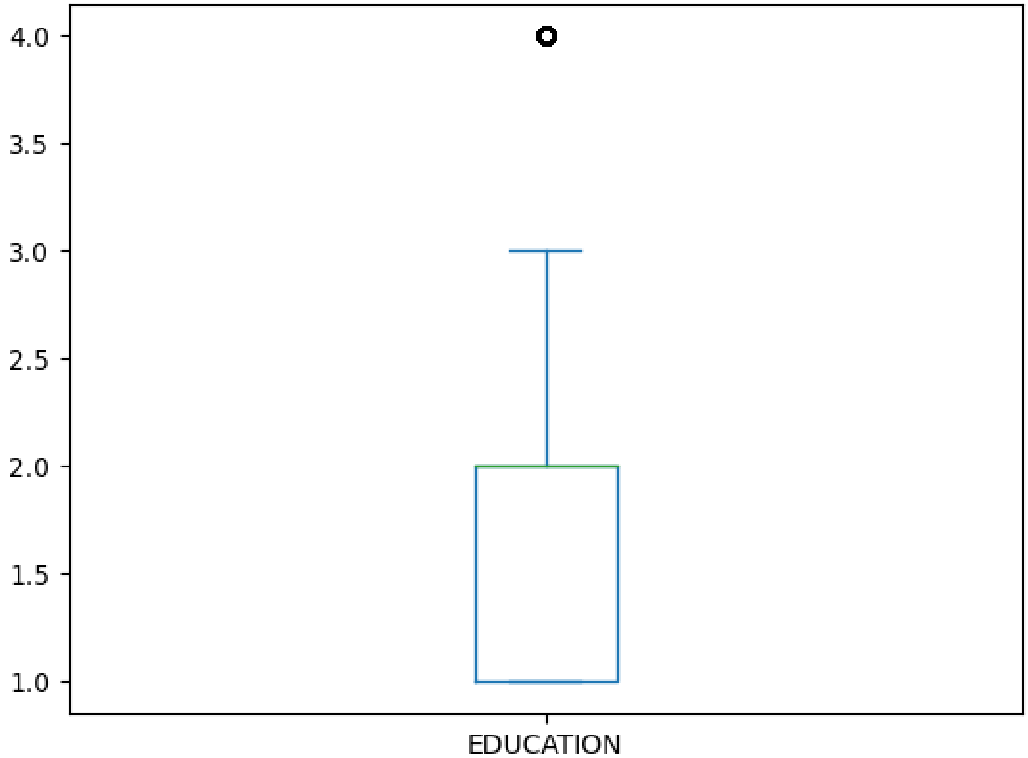
[ ]:
```
print(df2['MARRIAGE'].plot.box())
```

```
Axes(0.125,0.11;0.775x0.77)
```




```
[ ]: print(df2['EDUCATION'].plot.box())
```

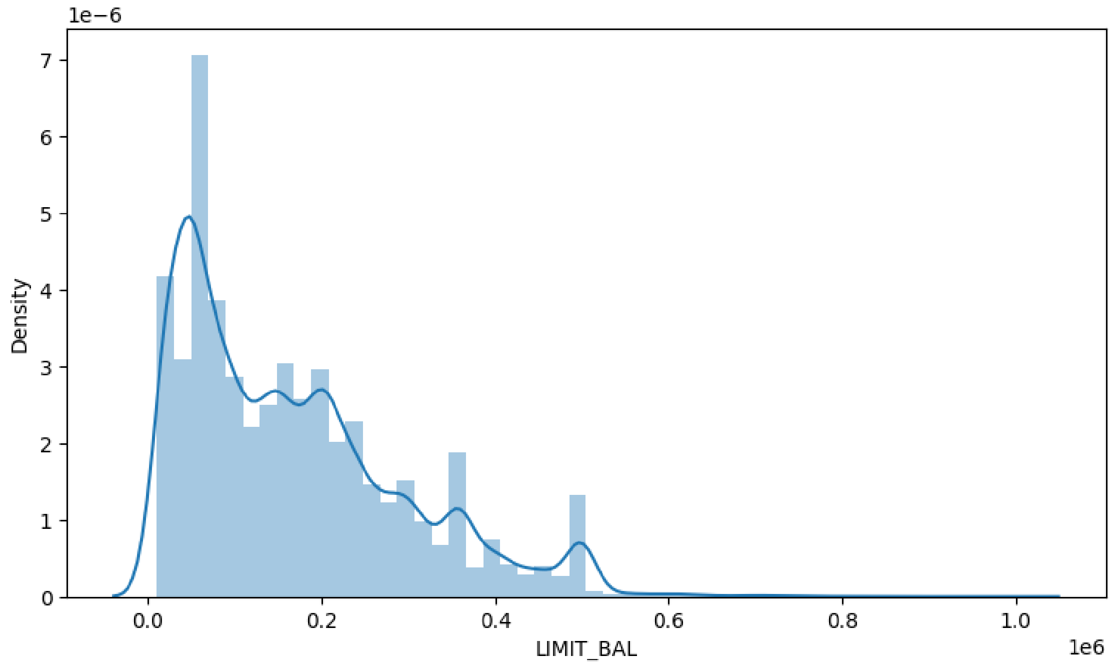
```
Axes(0.125,0.11;0.775x0.77)
```



[ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.subplots(figsize=(20,5))
plt.subplot(121)
sns.distplot(df2.LIMIT_BAL)


plt.show()
```

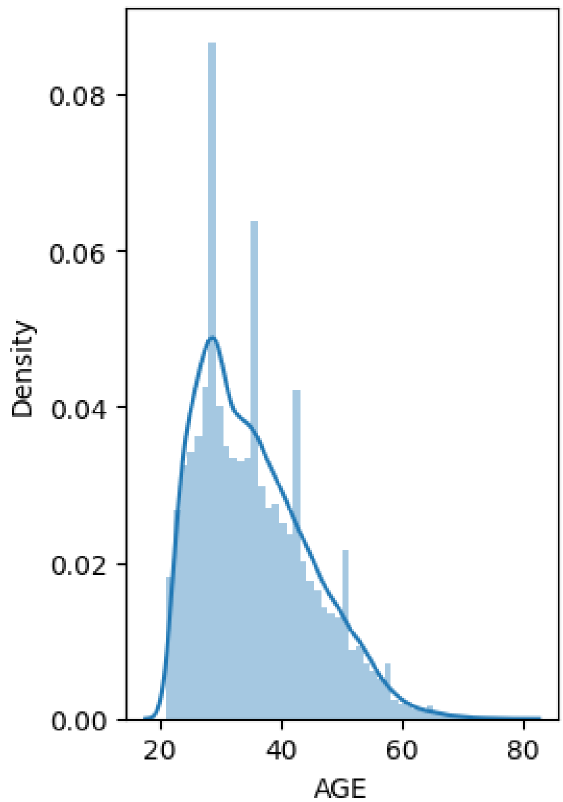
```
<ipython-input-19-3eb1f80b6859>:4: MatplotlibDeprecationWarning: Auto-removal of
overlapping axes is deprecated since 3.6 and will be removed two minor releases
later; explicitly call ax.remove() as needed.
  plt.subplot(121)
<ipython-input-19-3eb1f80b6859>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
sns.distplot(df2.LIMIT_BAL)
```



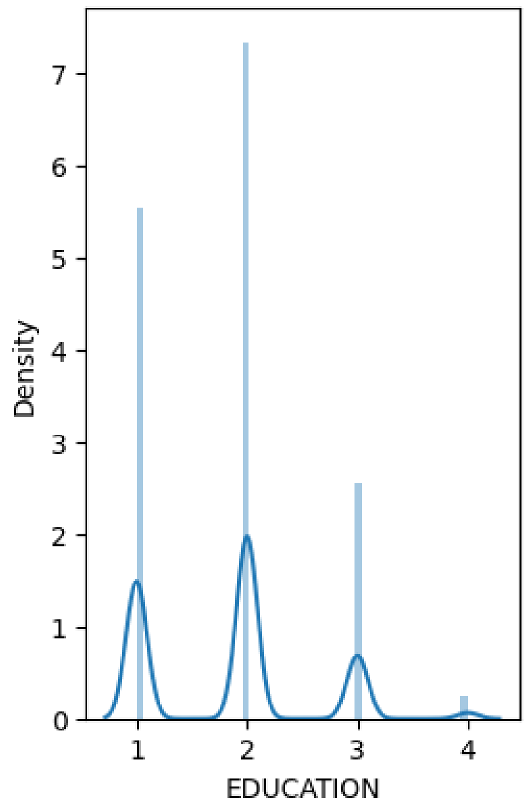
[ ]:
```
plt.subplot(122)
sns.distplot(df2.AGE)
plt.show()
```

```
<ipython-input-20-1200cca5c3a4>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df2.AGE)
```



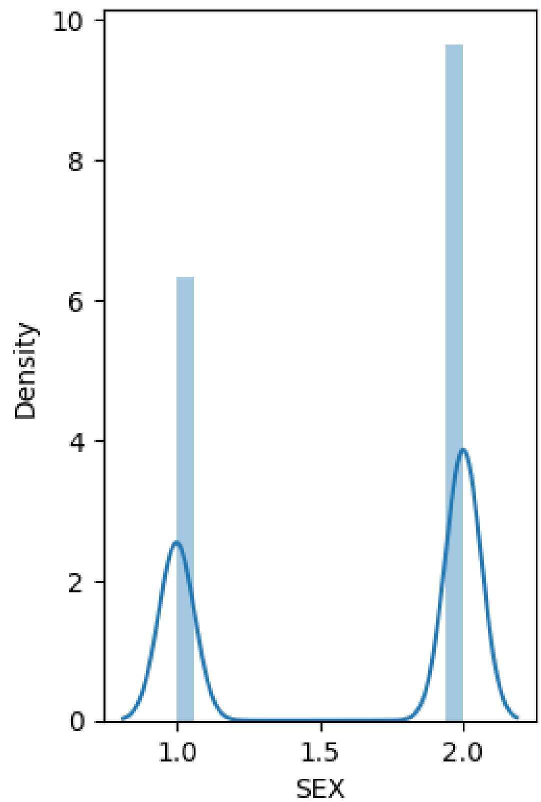
[ ]:
```
plt.subplot(122)
sns.distplot(df2.EDUCATION)
plt.show()
```

```
<ipython-input-21-01bc445327b9>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df2.EDUCATION)
```



[ ]:
```
plt.subplot(122)
sns.distplot(df2.SEX)
plt.show()
```

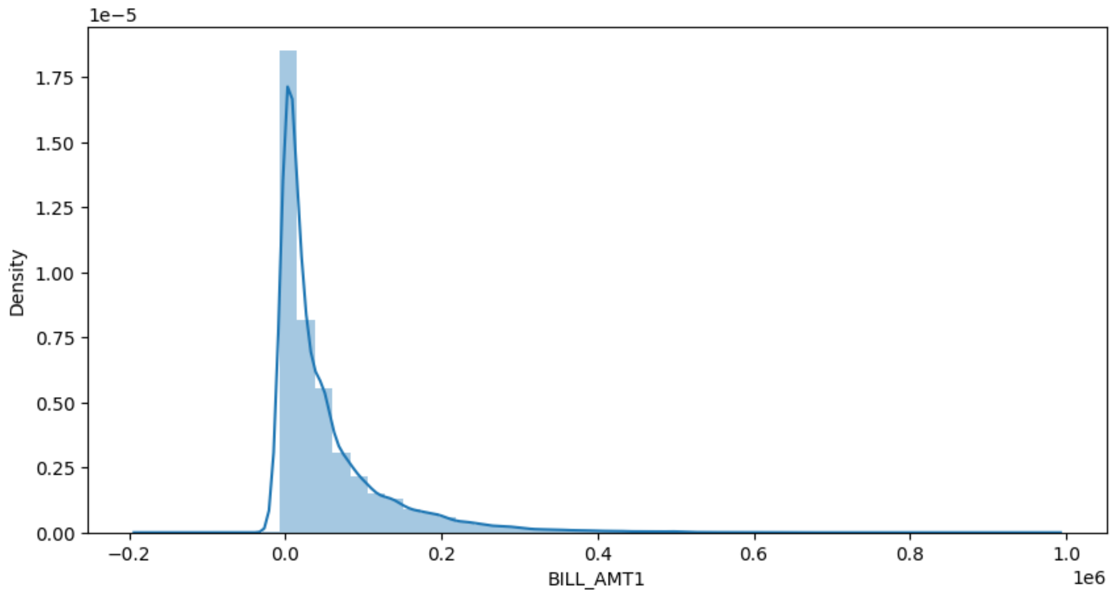
```
<ipython-input-22-717f58625892>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df2.SEX)
```



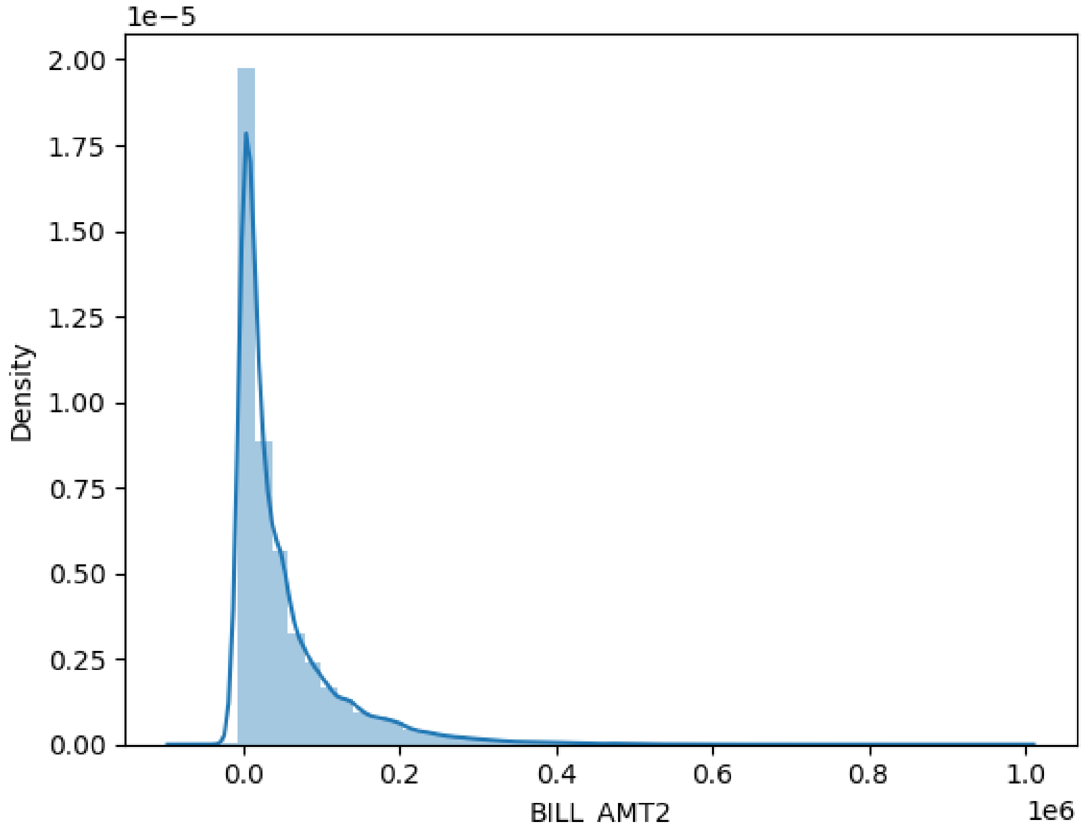
[ ]:
```
plt.subplots(figsize=(10,5))
plt.subplot(111)
sns.distplot(df2.BILL_AMT1)
plt.show()

sns.distplot(df2.BILL_AMT2)
plt.show()
sns.distplot(df2.BILL_AMT3)
plt.show()
sns.distplot(df2.BILL_AMT4)
plt.show()
sns.distplot(df2.BILL_AMT5)
plt.show()
sns.distplot(df2.BILL_AMT6)
plt.show()
```

```
<ipython-input-23-45f8b3e66573>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
```

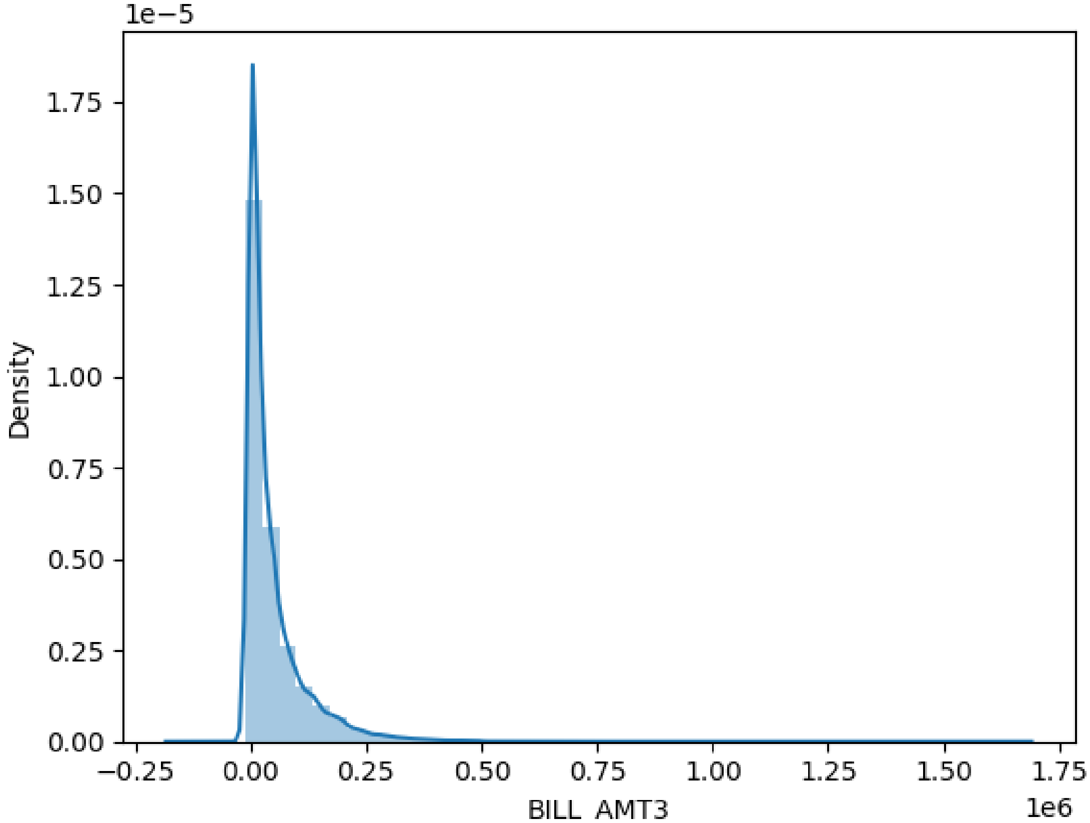
```
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df2.BILL_AMT1)
```



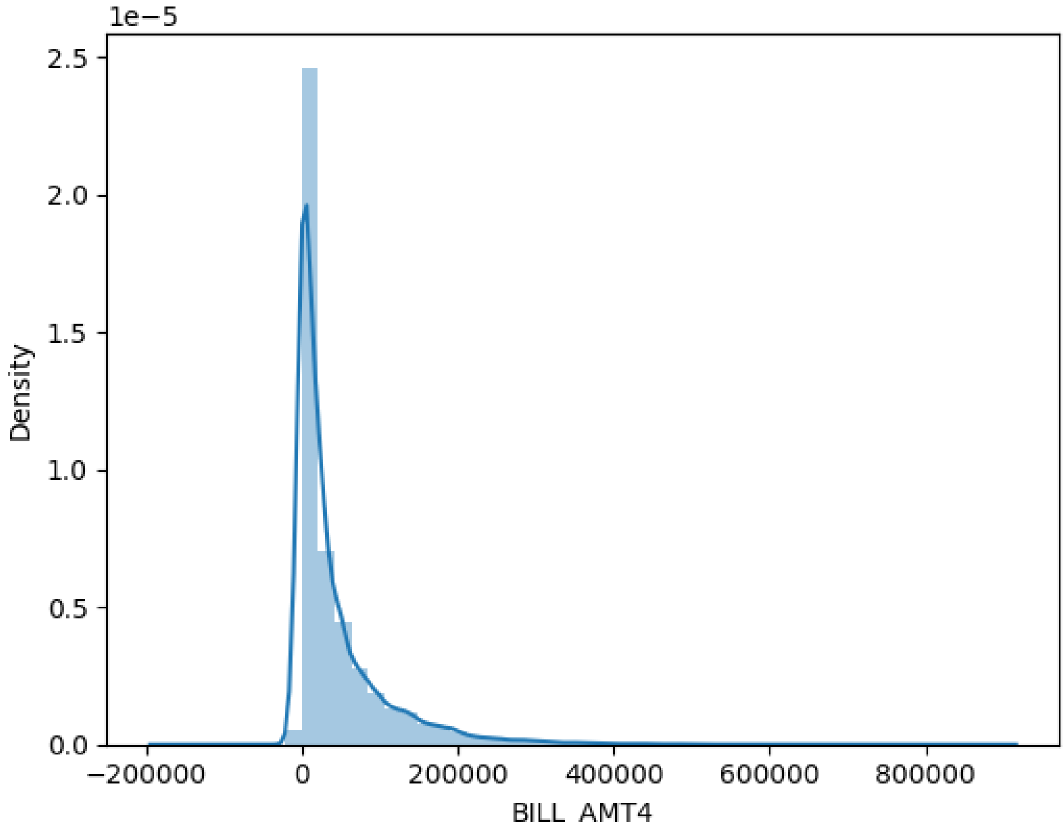

```
<ipython-input-23-45f8b3e66573>:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df2.BILL_AMT2)
```



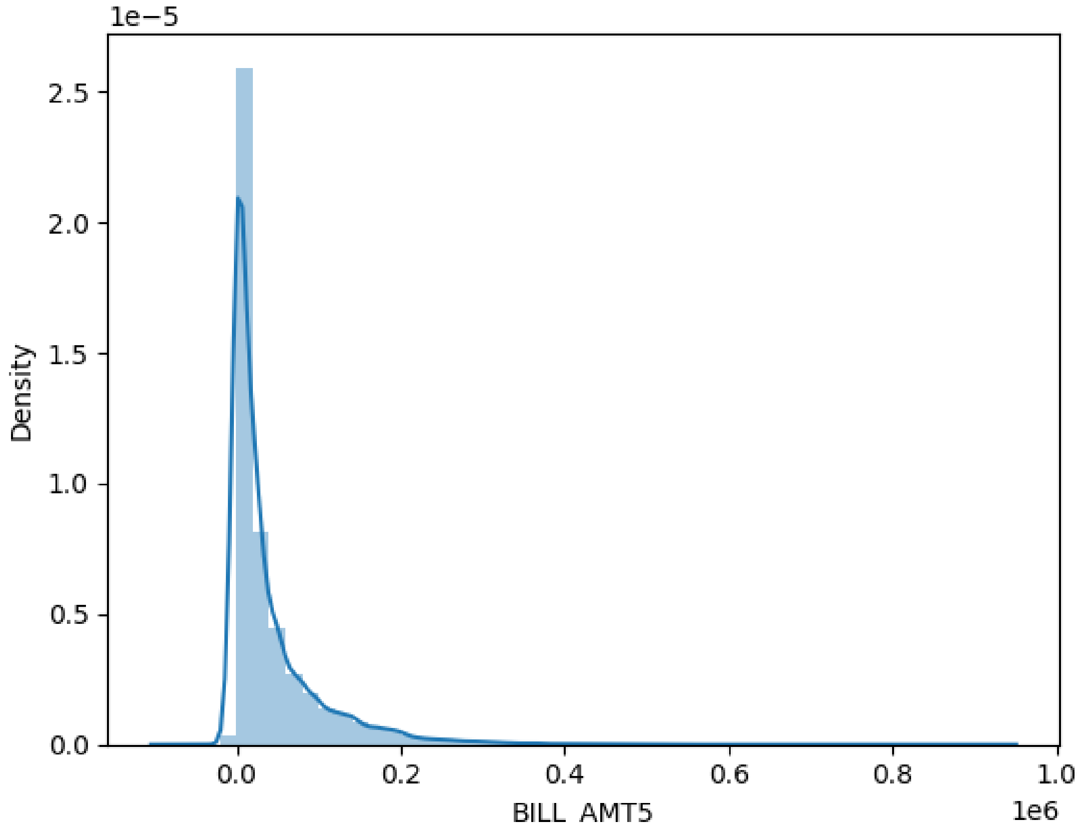

```
<ipython-input-23-45f8b3e66573>:8: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df2.BILL_AMT3)
```



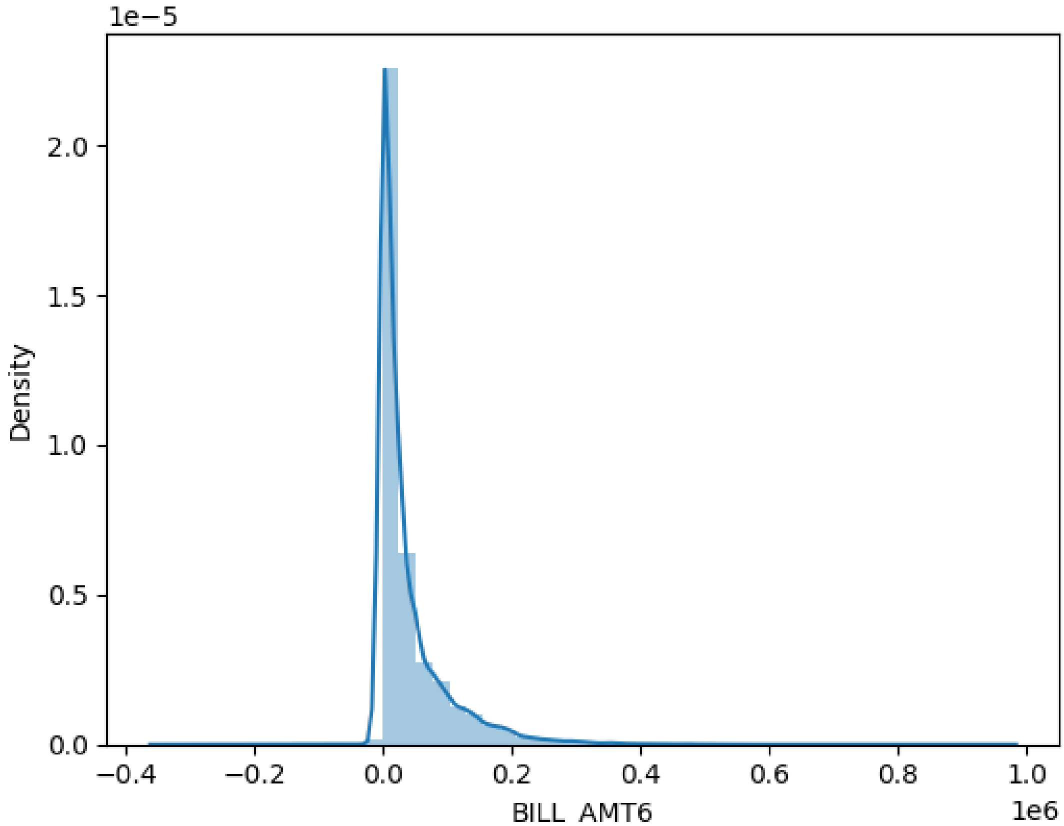

```
<ipython-input-23-45f8b3e66573>:10: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df2.BILL_AMT4)
```




```
<ipython-input-23-45f8b3e66573>:12: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df2.BILL_AMT5)
```

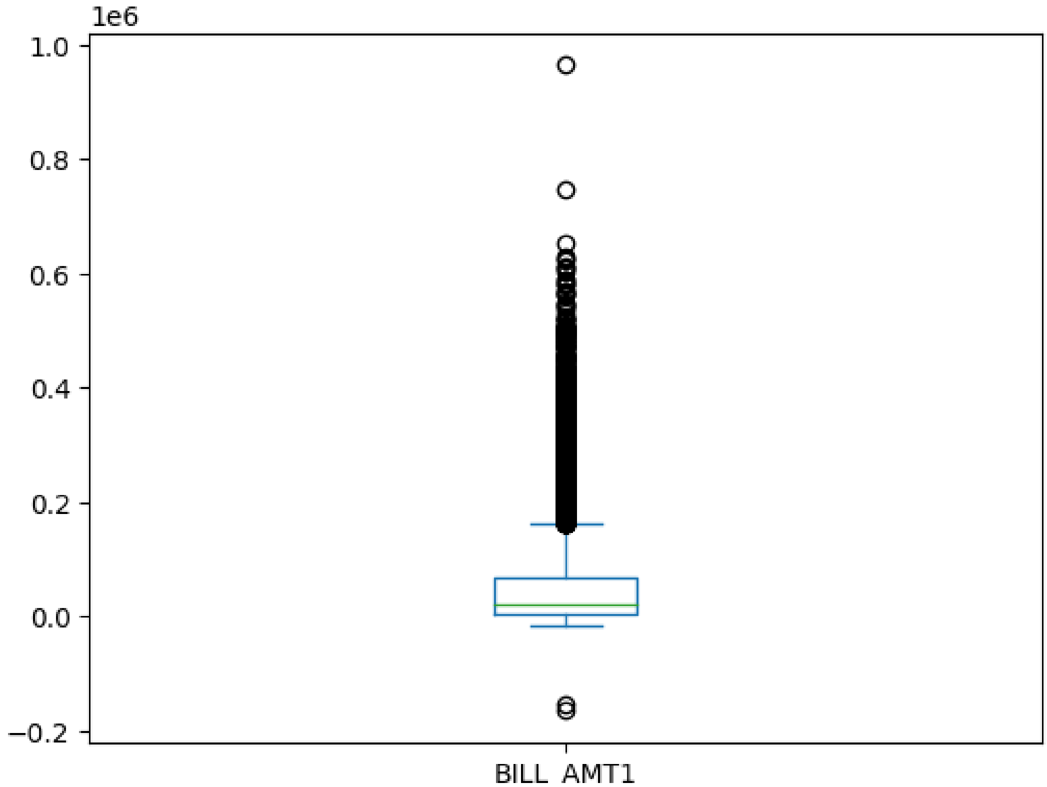



```
<ipython-input-23-45f8b3e66573>:14: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df2.BILL_AMT6)
```

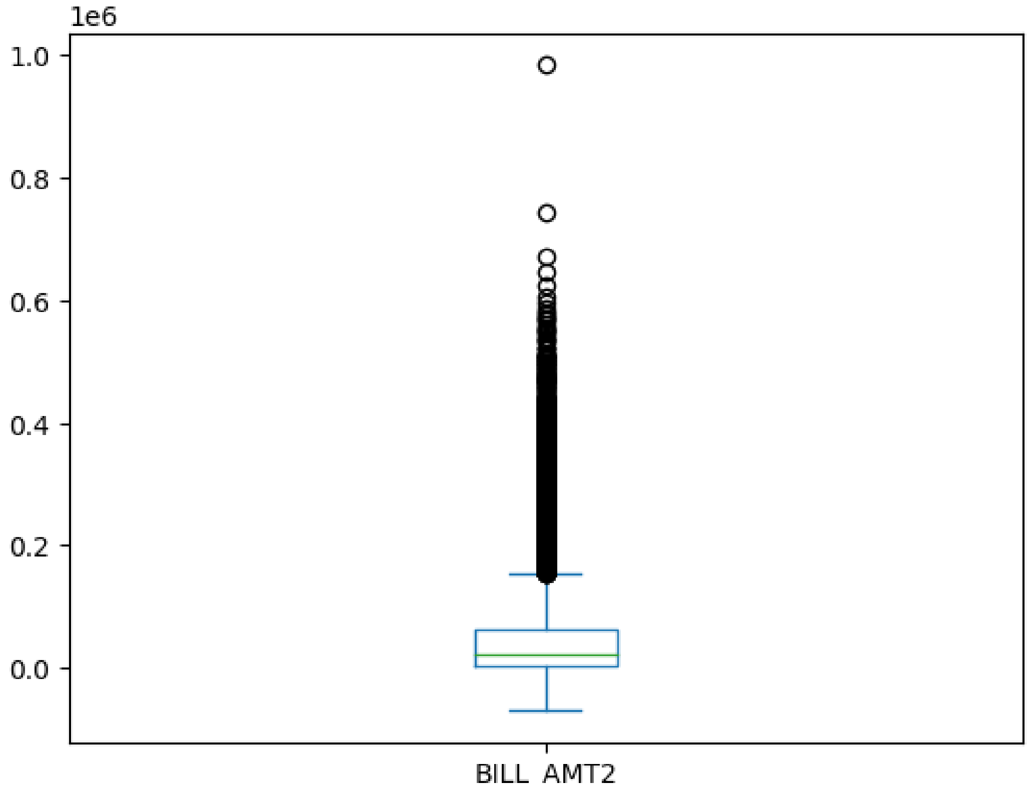


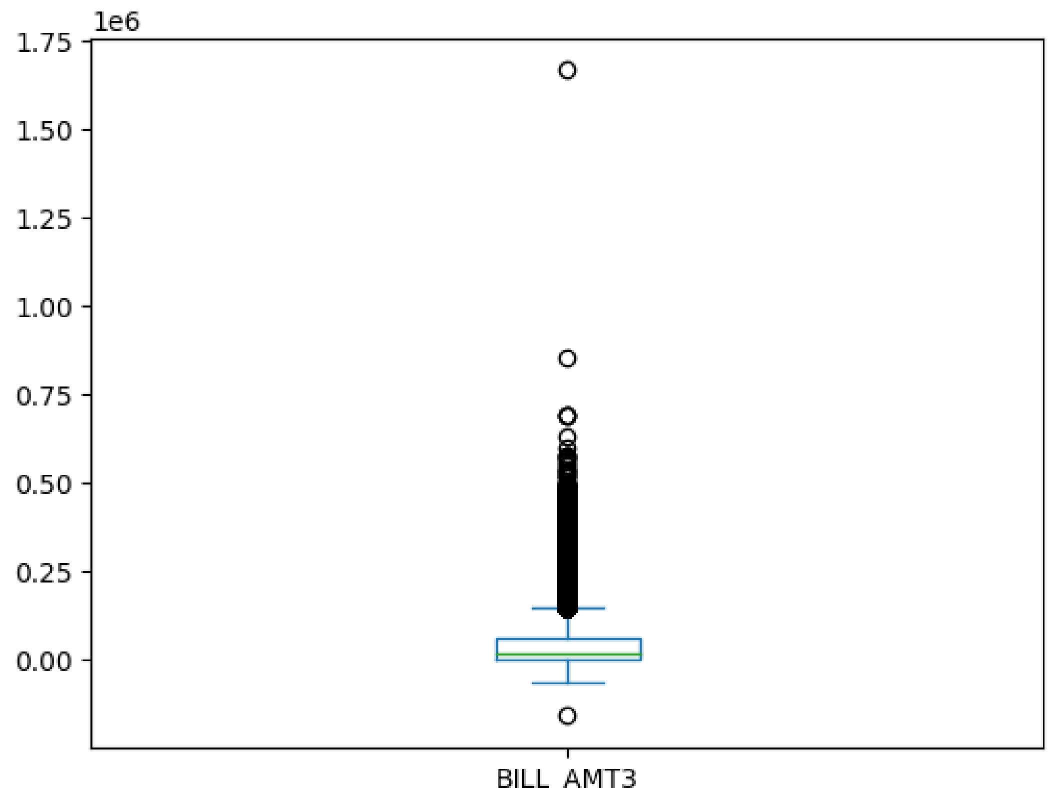
[ ]:
```
df2[['BILL_AMT1']].plot.box()
df2[['BILL_AMT2']].plot.box()
df2[['BILL_AMT3']].plot.box()
df2[['BILL_AMT4']].plot.box()
df2[['BILL_AMT5']].plot.box()
df2[['BILL_AMT6']].plot.box()
```

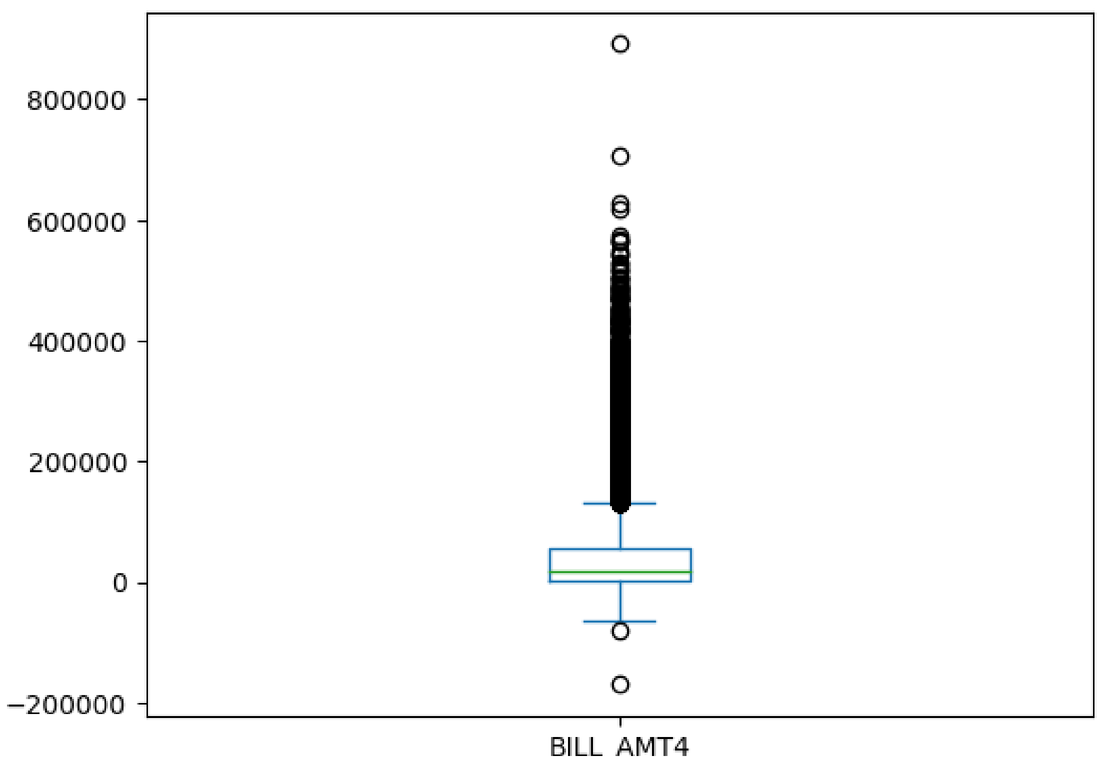
[ ]: <Axes: >

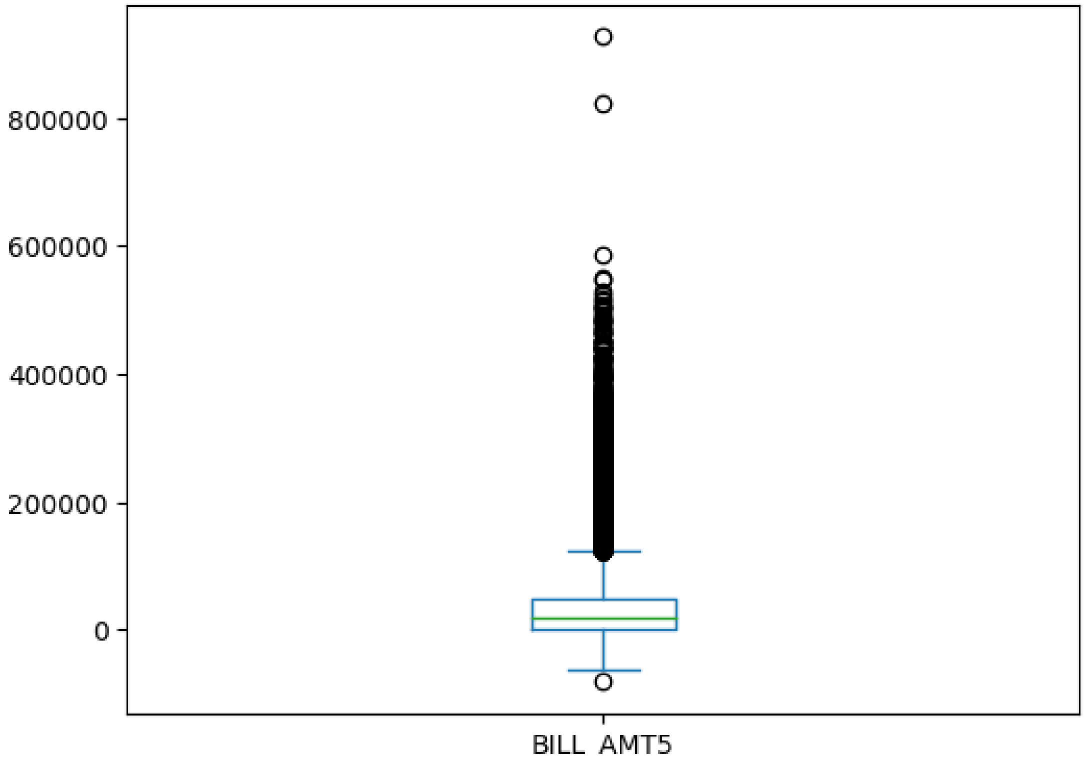


1e6
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
BILL_AMT1

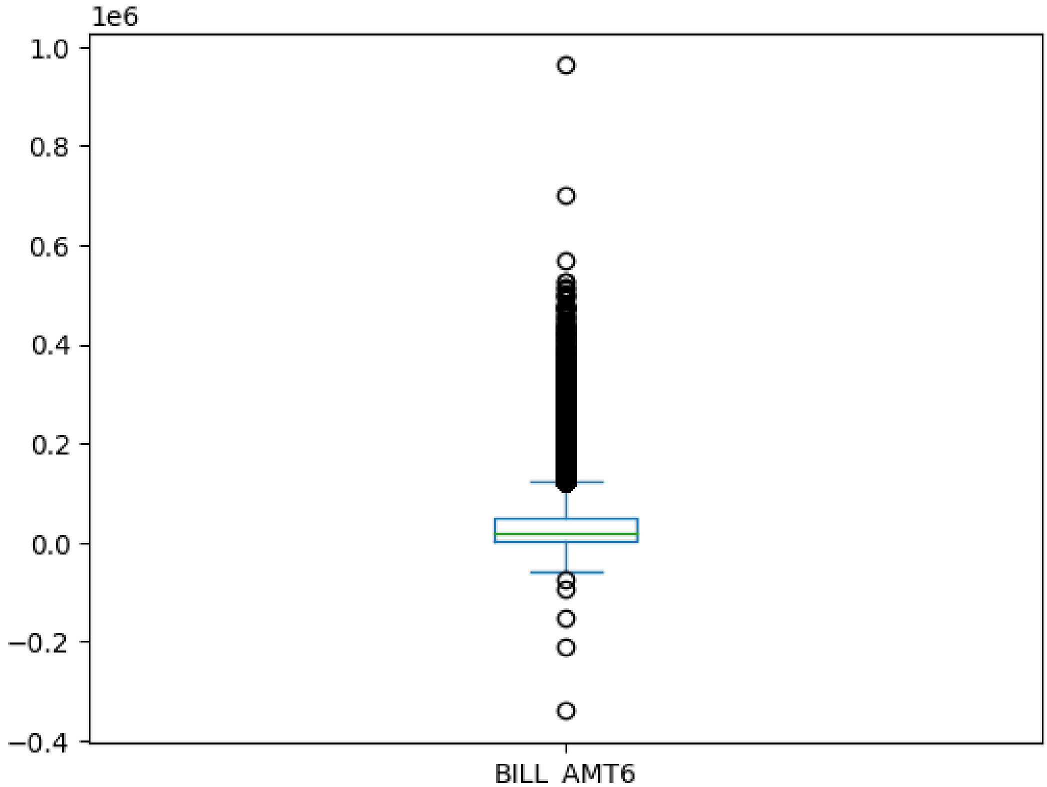

1e6
1.0
0.8
0.6
0.4
0.2
0.0
BILL_AMT2


1e6
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
BILL_AMT3


800000
600000
400000
200000
0
−200000
BILL_AMT4

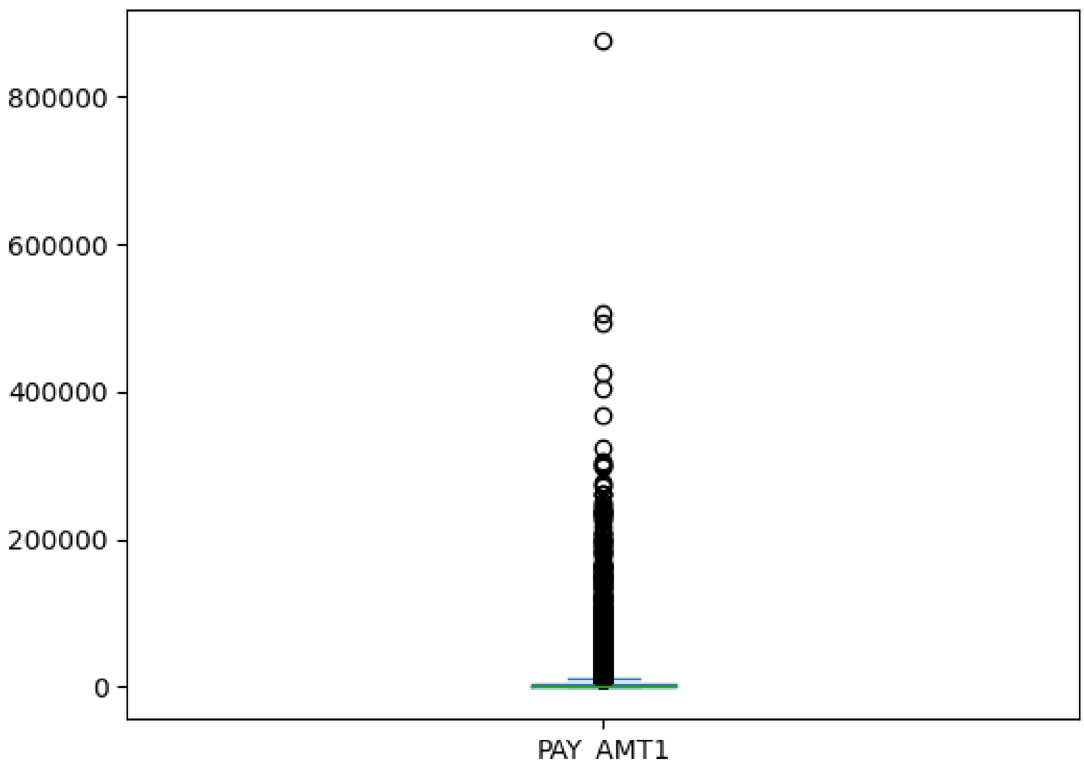


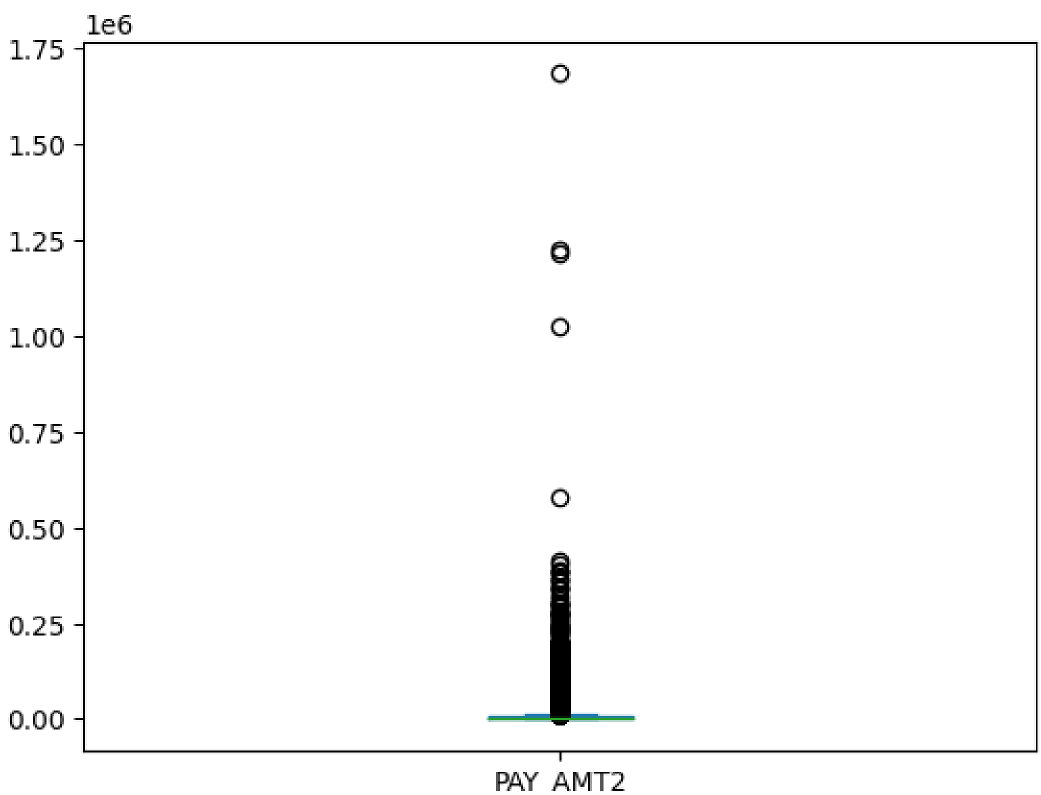
800000
600000
400000
200000
0
BILL_AMT5

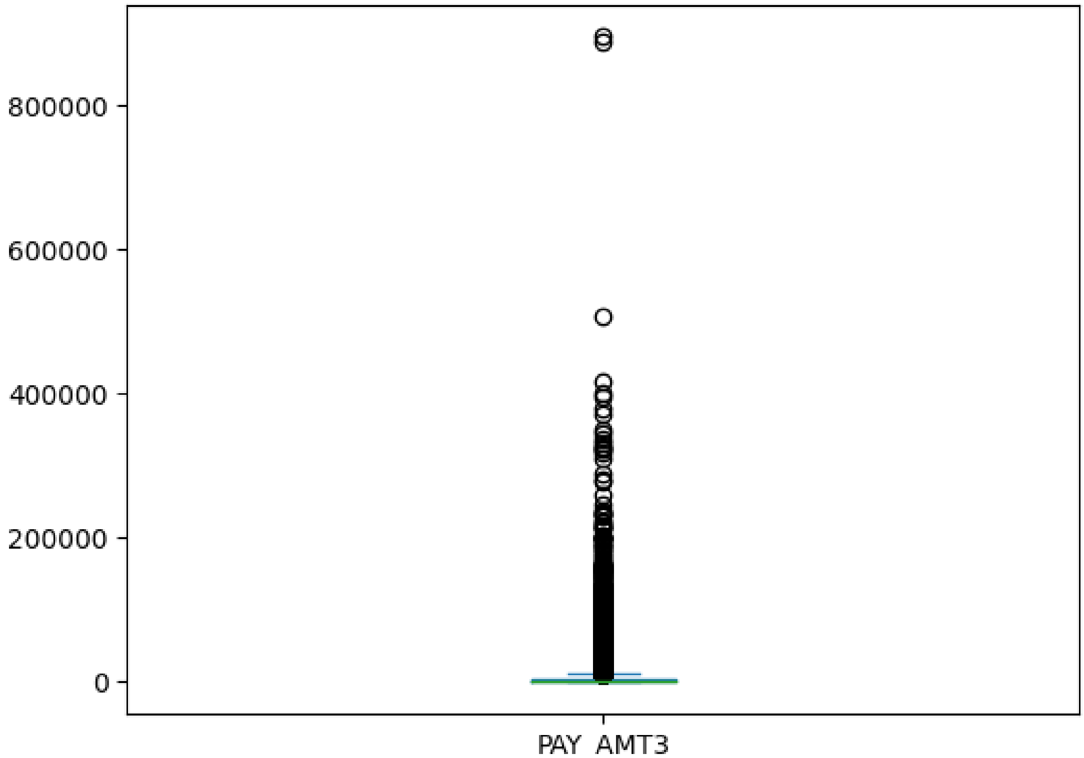


[ ]:
```
df2[['PAY_AMT1']].plot.box()
df2[['PAY_AMT2']].plot.box()
df2[['PAY_AMT3']].plot.box()
df2[['PAY_AMT4']].plot.box()
df2[['PAY_AMT5']].plot.box()
df2[['PAY_AMT6']].plot.box()
```

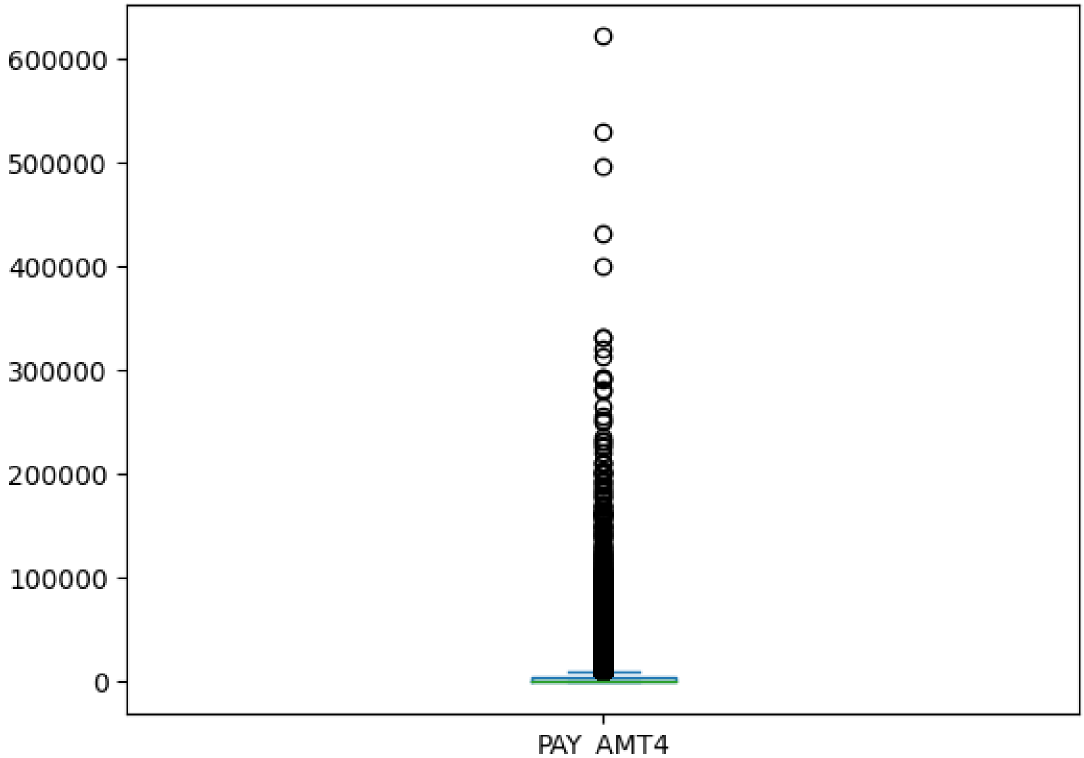
[ ]: <Axes: >

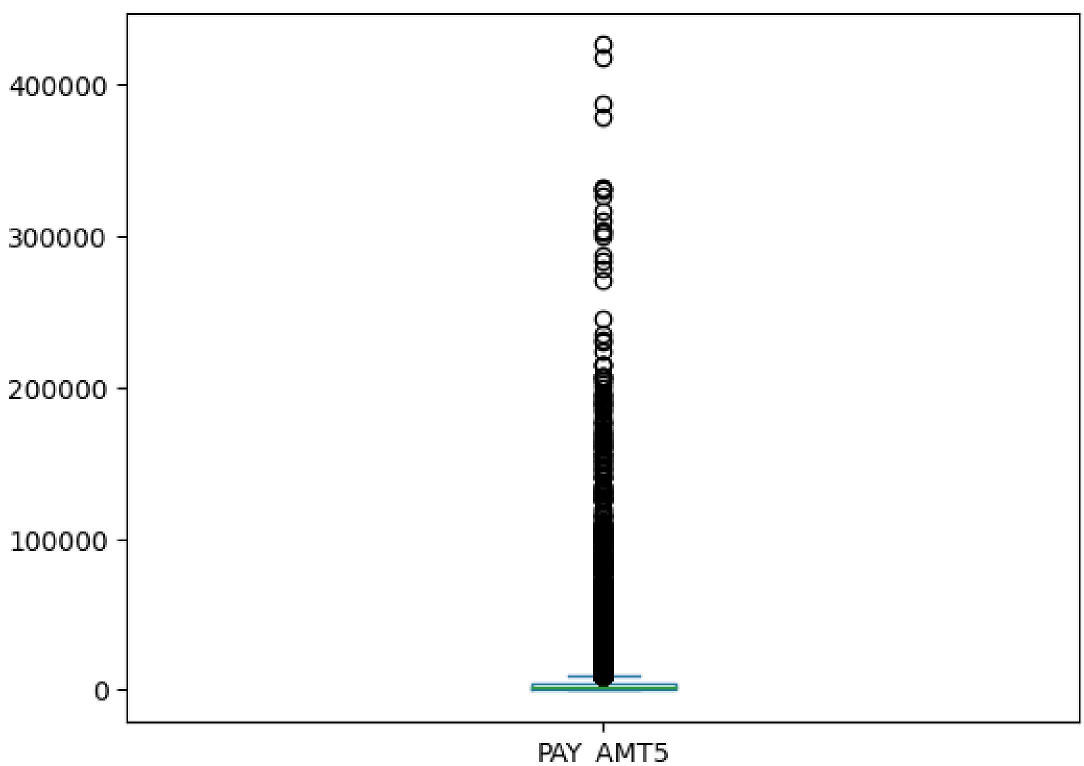

800000
600000
400000
200000
0
PAY_AMT1

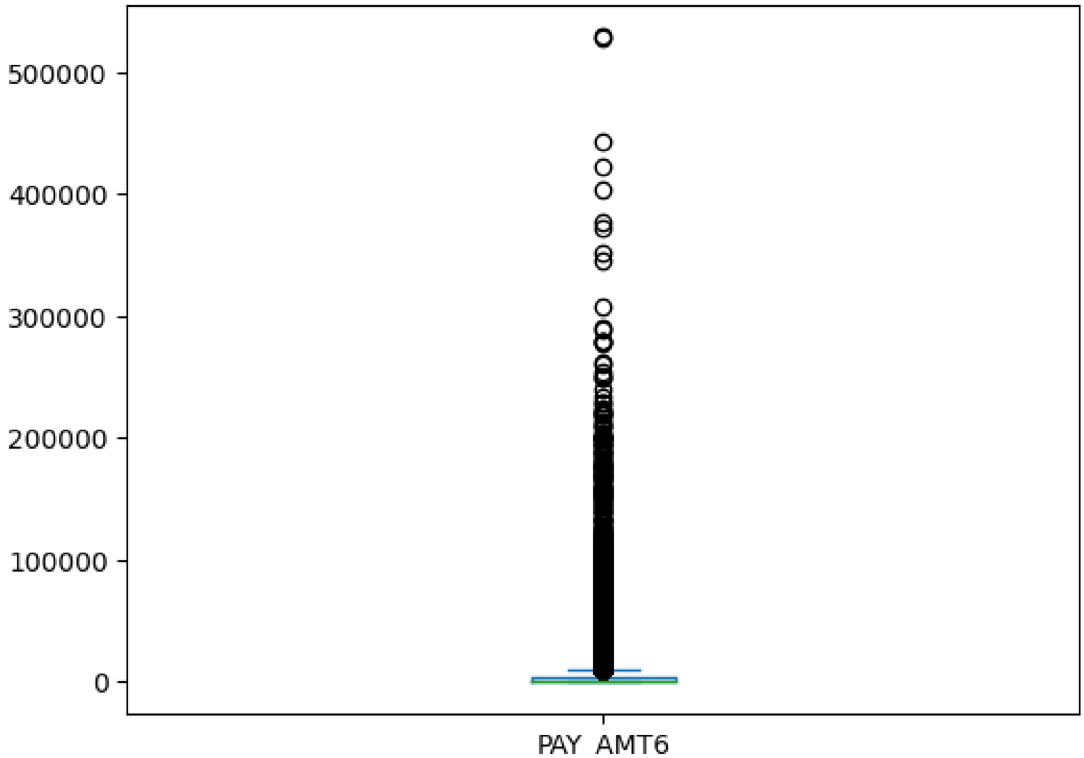

1e6
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
PAY_AMT2



800000
600000
400000
200000
0
PAY_AMT3


600000
500000
400000
300000
200000
100000
0
PAY_AMT4


400000
300000
200000
100000
0
PAY_AMT5



[ ]:
```
pd.crosstab(df2['BILL_AMT1'], df2['PAY_AMT1'])
```

[ ]:
```
PAY_AMT1   0       1       2       3       4       5       6       7       \
BILL_AMT1
-165580         0       0       0       0       0       0       0       0
-154973         0       0       0       0       0       0       0       0
-15308          0       0       0       0       0       0       0       0
-14386          0       0       0       0       0       0       0       0
-11545          0       0       0       0       0       0       0       0
…            …     …     …     …     …     …     …     …
 626648         0       0       0       0       0       0       0       0
 630458         0       0       0       0       0       0       0       0
 653062         0       0       0       0       0       0       0       0
 746814         0       0       0       0       0       0       0       0
 964511         0       0       0       0       0       0       0       0

PAY_AMT1   8       9       …  300039  302000  304815  323014  368199  \
BILL_AMT1                  …
-165580         0       0  …       0       0       0       0       0
-154973         0       0  …       0       0       0       0       0
```

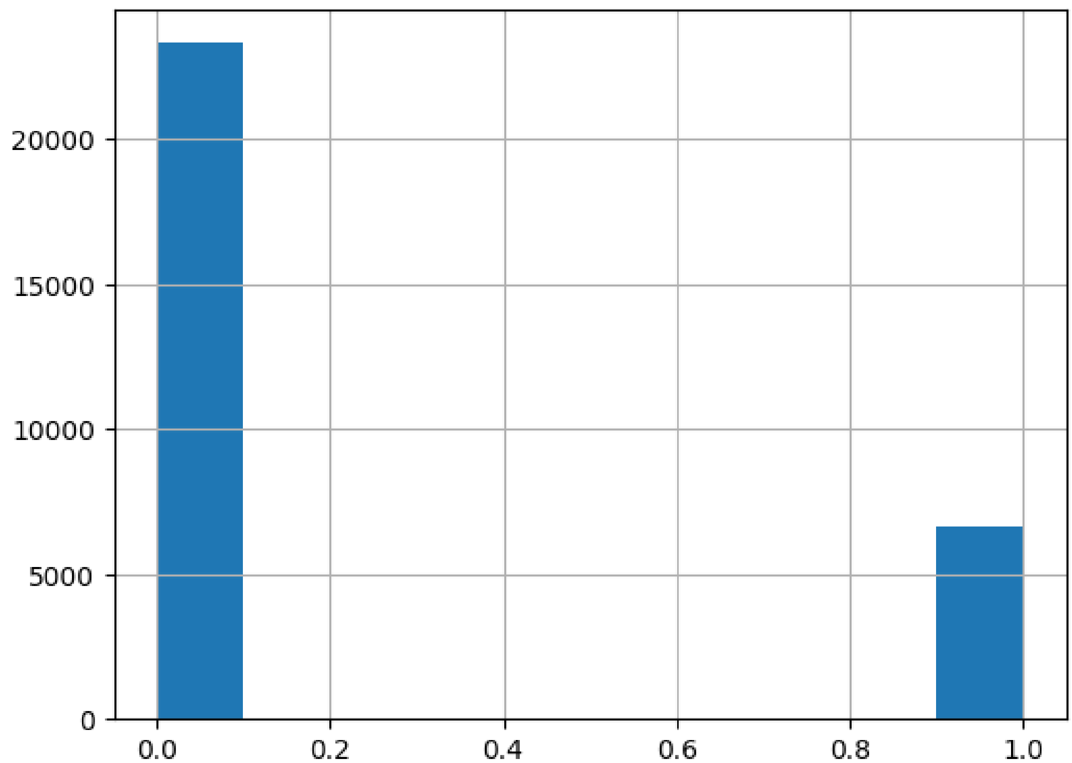
```
-15308          1       0  …      0       0       0       0       0
-14386          0       0  …      0       0       0       0       0
-11545          0       0  …      0       0       0       0       0
…            …     …  …     …     …     …     …     …
 626648         0       0  …      0       0       0       0       0
 630458         0       0  …      0       0       0       0       0
 653062         0       0  …      0       0       0       0       0
 746814         0       0  …      0       0       0       0       0
 964511         0       0  …      0       0       0       0       0

PAY_AMT1   405016  423903  493358  505000  873552
BILL_AMT1
-165580         0       0       0       1       0
-154973         0       0       0       0       0
-15308          0       0       0       0       0
-14386          0       0       0       0       0
-11545          0       0       0       0       0
…            …     …     …     …     …
 626648         0       0       0       0       0
 630458         0       0       0       0       0
 653062         0       0       0       0       0
 746814         0       0       0       0       0
 964511         0       0       0       0       0

[22685 rows x 7928 columns]
```

1.5 Data imbalance

[ ]:
```
print(df2['default payment next month'].hist())
Default_ratio=df2['default payment next month'].value_counts()[1]/df2['default␣
 ↪payment next month'].count()
Default_ratio
```

```
Axes(0.125,0.11;0.775x0.77)
```

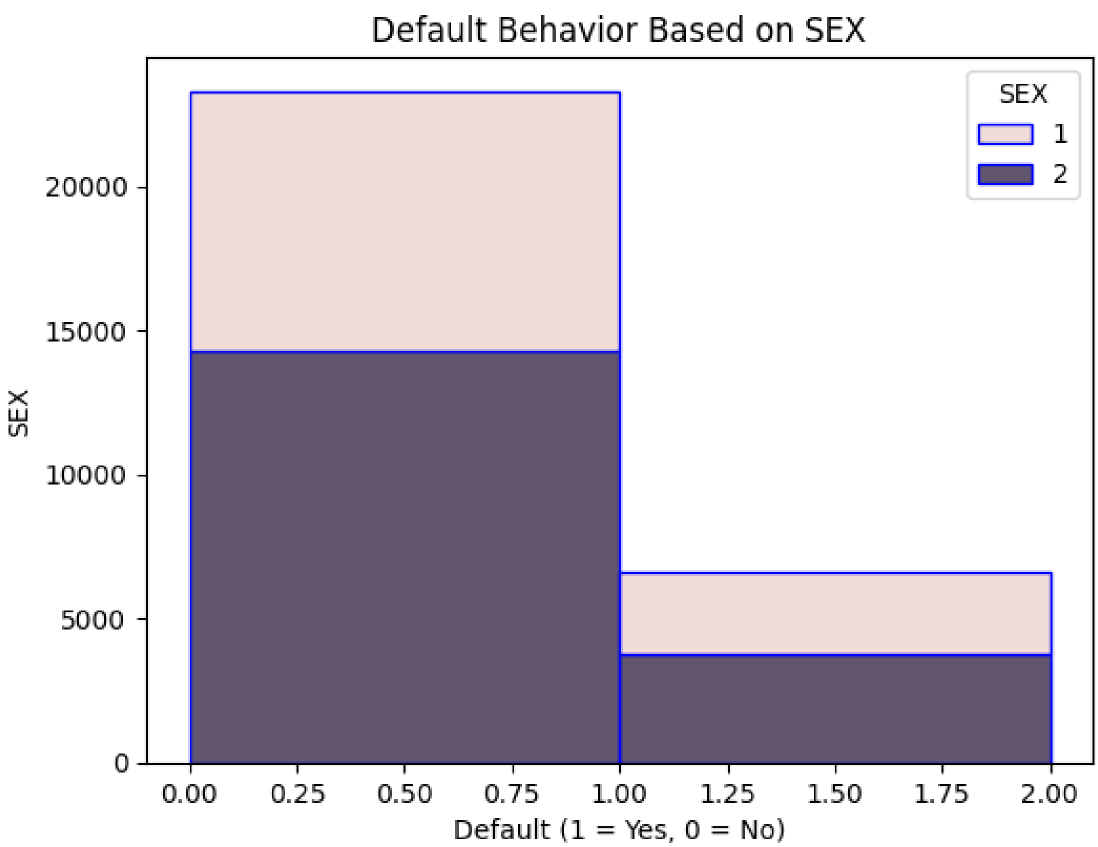
[ ]:
```
0.22153548042229051
```

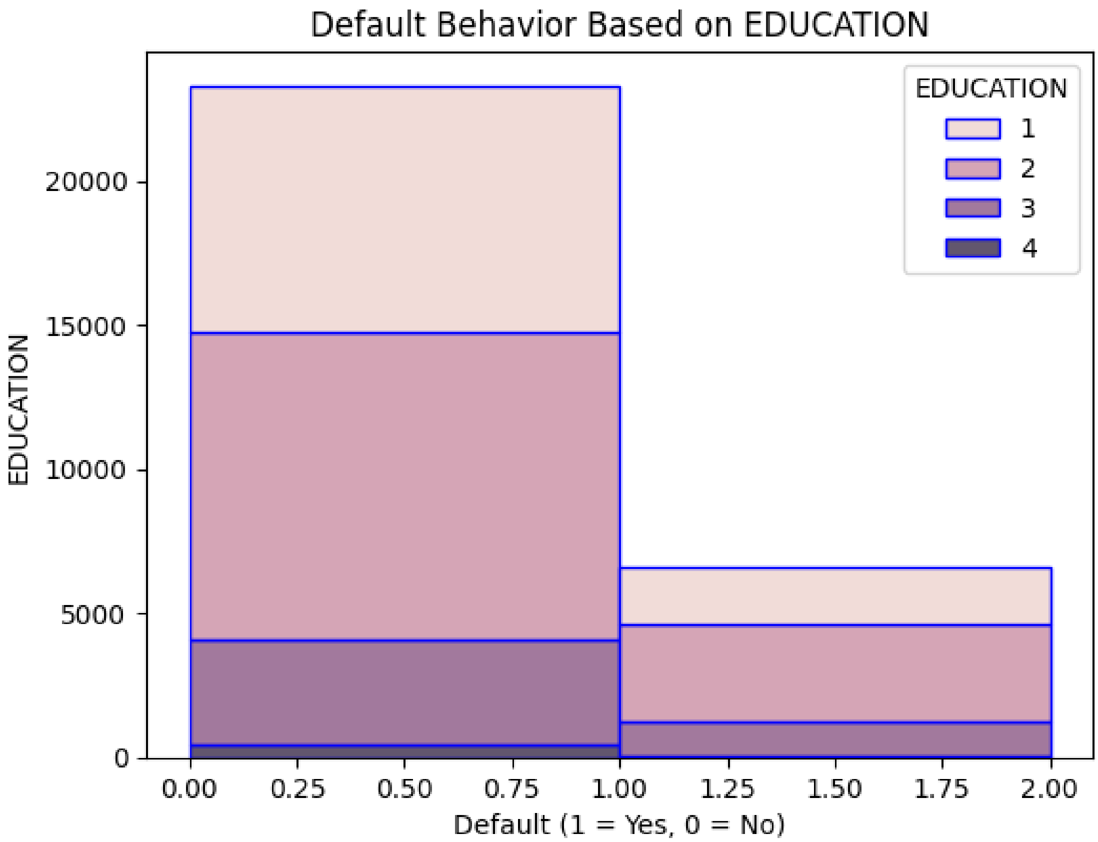


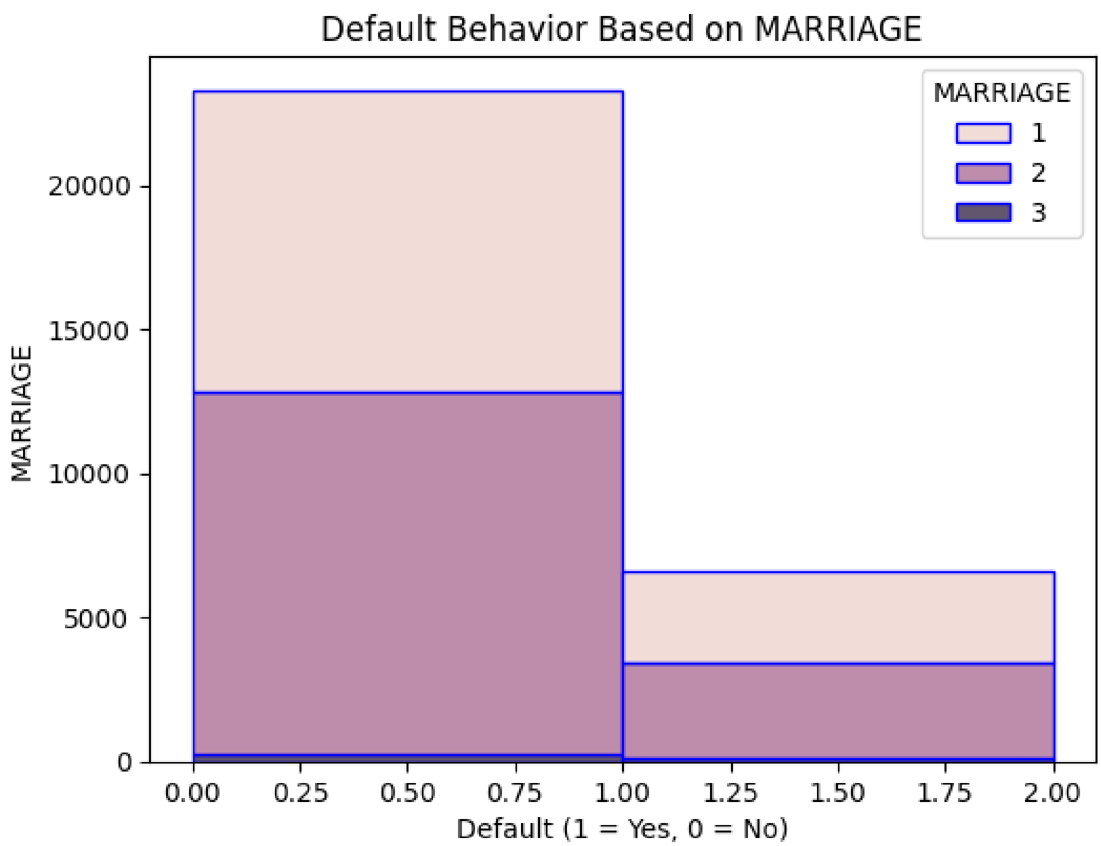
1.6 Default of payment based on demographic variable

```
[ ]: #Data balance identification based on dependent variable default payment

import seaborn as sns
import matplotlib.pyplot as plt

sns.histplot(data=df2, x='default payment next month', hue='SEX',␣
 ↪multiple='stack', bins=[0, 1, 2], edgecolor='b')
plt.xlabel('Default (1 = Yes, 0 = No)')
plt.ylabel('SEX')
plt.title('Default Behavior Based on SEX')

plt.show()

pd.crosstab(df2['EDUCATION'], df2['default payment next month'])

sns.histplot(data=df2, x='default payment next month', hue='EDUCATION',␣
 ↪multiple='stack', bins=[0, 1, 2], edgecolor='b')
plt.xlabel('Default (1 = Yes, 0 = No)')
plt.ylabel('EDUCATION')
plt.title('Default Behavior Based on EDUCATION')
```

```
plt.show()

pd.crosstab(df2['MARRIAGE'], df2['default payment next month'])

sns.histplot(data=df2, x='default payment next month', hue='MARRIAGE',␣
 ↪multiple='stack', bins=[0, 1, 2], edgecolor='b')
plt.xlabel('Default (1 = Yes, 0 = No)')
plt.ylabel('MARRIAGE')
plt.title('Default Behavior Based on MARRIAGE')

plt.show()
```




Default Behavior Based on EDUCATION
EDUCATION
1
2
3
4
20000
15000
10000
5000
0
EDUCATION
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Default (1 = Yes, 0 = No)



1.7 Outlier Consideration

```
[ ]: #outlier consideration

     df2[df2.PAY_AMT1 > 300000][['LIMIT_BAL', 'PAY_1', 'PAY_2',␣
      ↪'BILL_AMT2','PAY_AMT1', 'BILL_AMT1']]
```

[ ]:

	LIMIT_BAL	PAY_1	PAY_2	BILL_AMT2	PAY_AMT1	BILL_AMT1
2687	500000	0	0	367979	368199	71921
5687	480000	0	0	400000	302000	106660
8500	400000	0	0	405016	405016	6500
12330	300000	1	0	324392	505000	-165580
25431	170000	0	0	167941	304815	30860
28003	510000	0	0	481382	493358	71121
28716	340000	0	0	176743	873552	139808
29820	400000	1	0	394858	423903	396343
29867	340000	0	0	331641	300039	44855
29963	610000	0	0	322228	323014	348392

1.8 Cleaned Data exported to CSV file

[ ]:
```
file_path = '/content/card_default_cleaned.csv'

# Export the DataFrame to CSV
df2.to_csv(file_path, index=False)

print(f"DataFrame 'df2' has been successfully exported to: {file_path}")
```

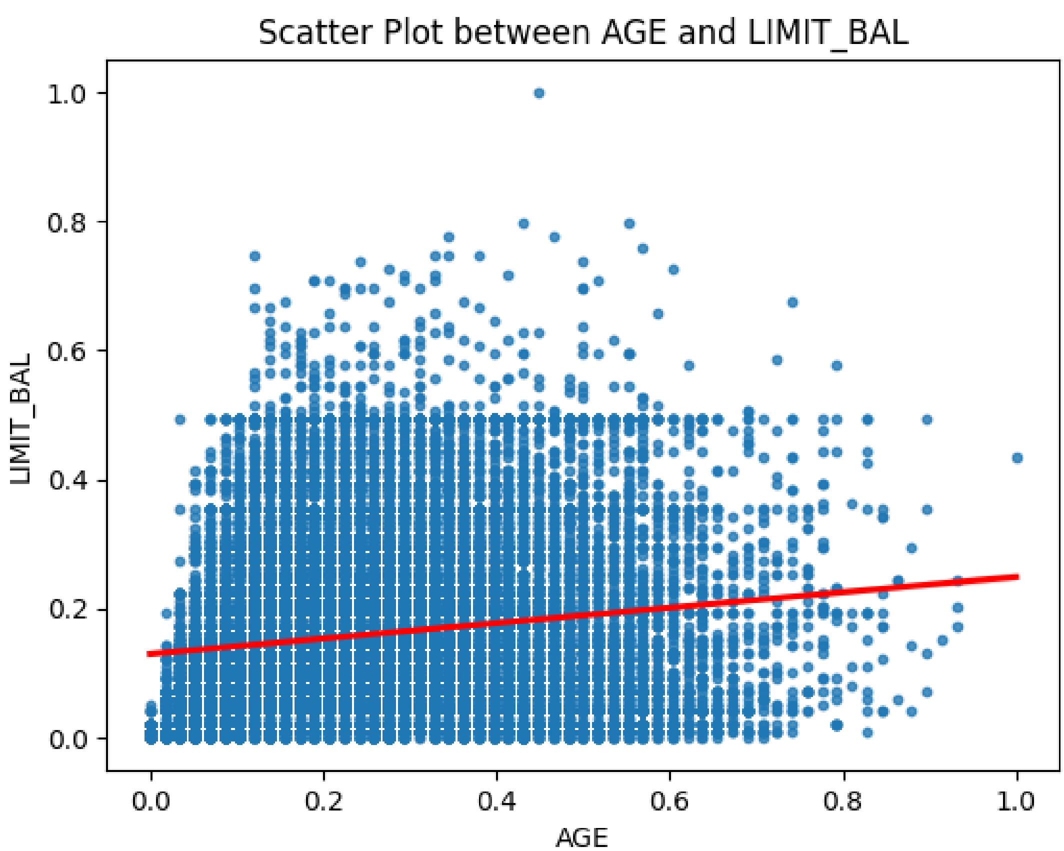
```
DataFrame 'df2' has been successfully exported to:
/content/card_default_cleaned.csv
```

1.9 Correlation Analysis with LIMIT_BAL

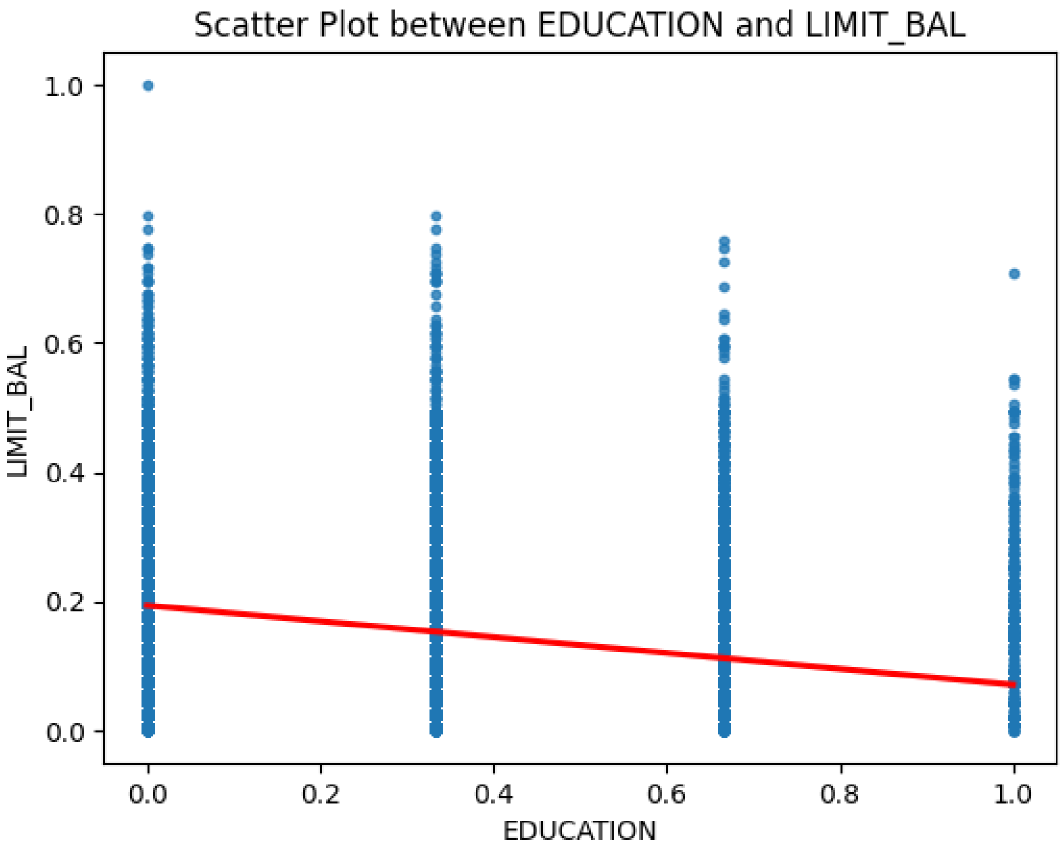
[ ]:
```
import numpy as np
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df2_scaled = scaler.fit_transform(df2)
df2_normalized = pd.DataFrame(df2_scaled, columns=df2.columns)

pearson_corr = df2_normalized['AGE'].corr(df2_normalized['LIMIT_BAL'])

print(pearson_corr)

sns.regplot(x='AGE', y='LIMIT_BAL', data=df2_normalized, scatter_kws={'s': 10},␣
↪line_kws={'color': 'red'})

# Adding labels and title
plt.xlabel('AGE')
plt.ylabel('LIMIT_BAL')
plt.title('Scatter Plot between AGE and LIMIT_BAL')
plt.show()


pearson_corr = df2_normalized['EDUCATION'].corr(df2_normalized['LIMIT_BAL'])

print(pearson_corr)
sns.regplot(x='EDUCATION', y='LIMIT_BAL', data=df2_normalized, scatter_kws={'s':
↪ 10}, line_kws={'color': 'red'})

# Adding labels and title
plt.xlabel('EDUCATION')
plt.ylabel('LIMIT_BAL')
plt.title('Scatter Plot between EDUCATION and LIMIT_BAL')
plt.show()

pearson_corr = df2_normalized['MARRIAGE'].corr(df2_normalized['LIMIT_BAL'])

print(pearson_corr)
```

```
sns.regplot(x='MARRIAGE', y='LIMIT_BAL', data=df2_normalized, scatter_kws={'s':
 ↪10}, line_kws={'color': 'red'})

# Adding labels and title
plt.xlabel('MARRIAGE')
plt.ylabel('LIMIT_BAL')
plt.title('Scatter Plot between MARRIAGE and LIMIT_BAL')
plt.show()

pearson_corr = df2_normalized['SEX'].corr(df2_normalized['LIMIT_BAL'])

print(pearson_corr)
sns.regplot(x='SEX', y='LIMIT_BAL', data=df2_normalized, scatter_kws={'s': 10},
 ↪line_kws={'color': 'red'})

# Adding labels and title
plt.xlabel('SEX')
plt.ylabel('LIMIT_BAL')
plt.title('Scatter Plot between SEX and LIMIT_BAL')
plt.show()
```

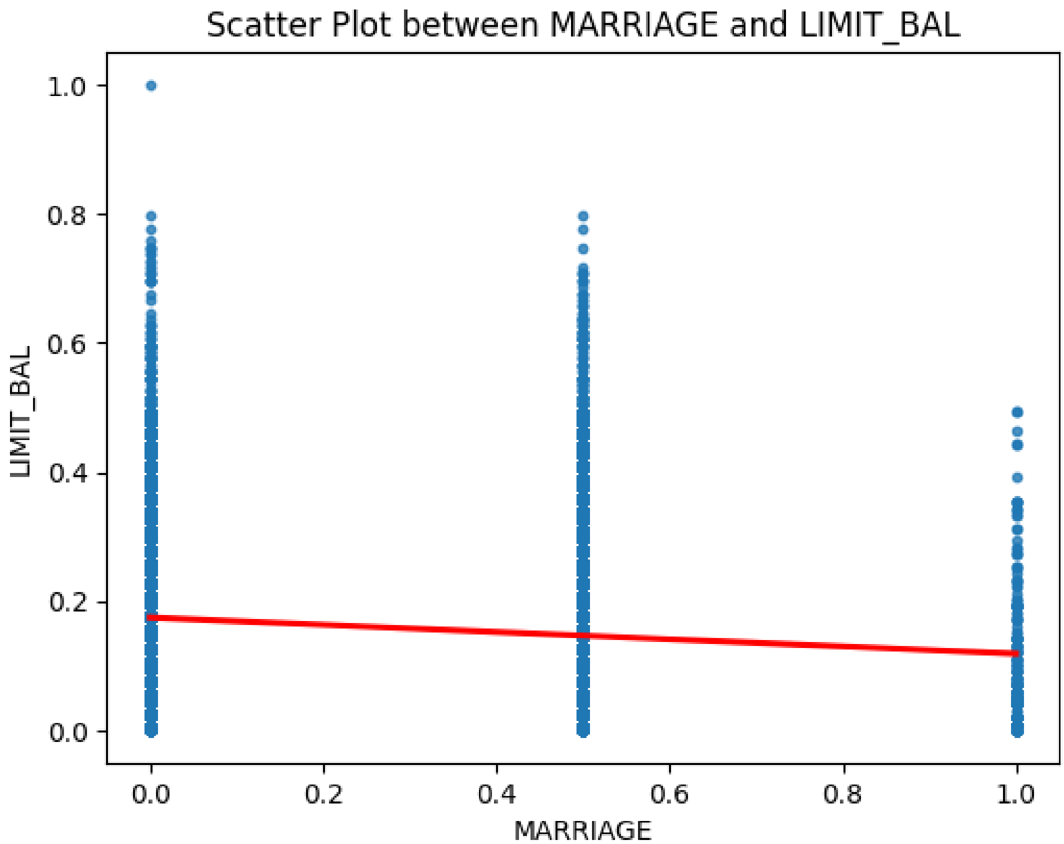
```
0.14480248557927614
```



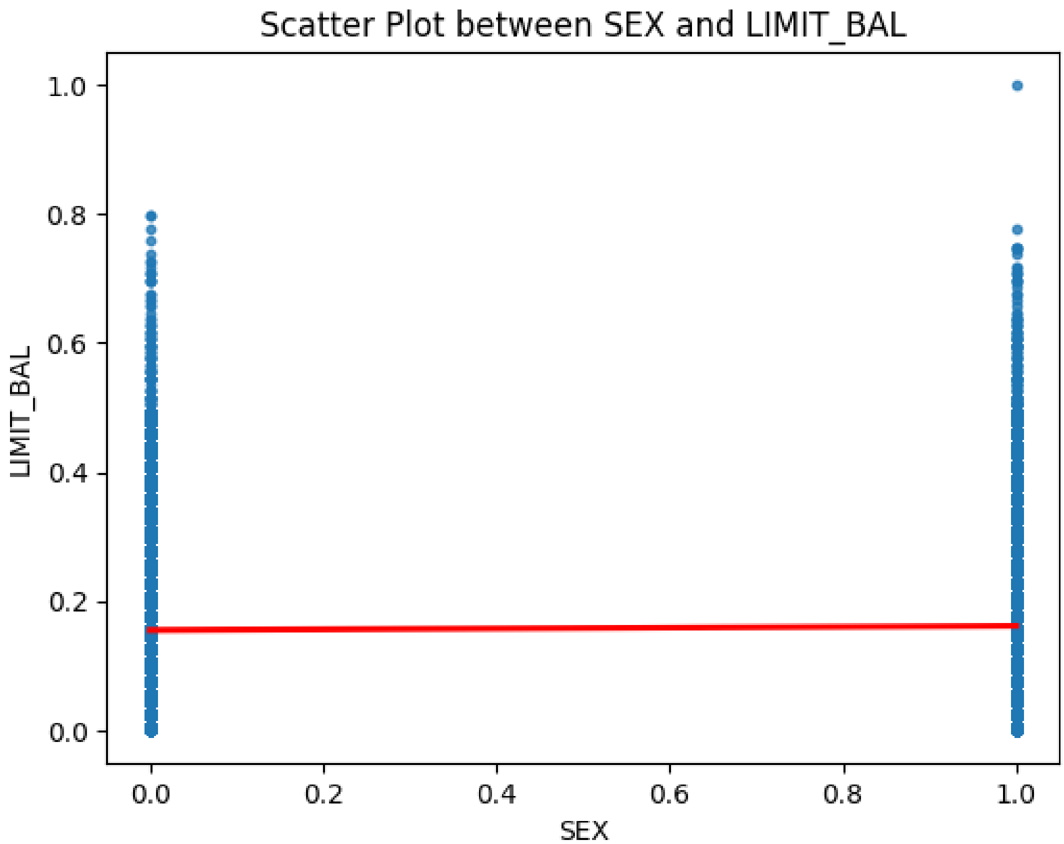
-0.2317397640347224



-0.1106832473738504




```
0.024952818456164105
```

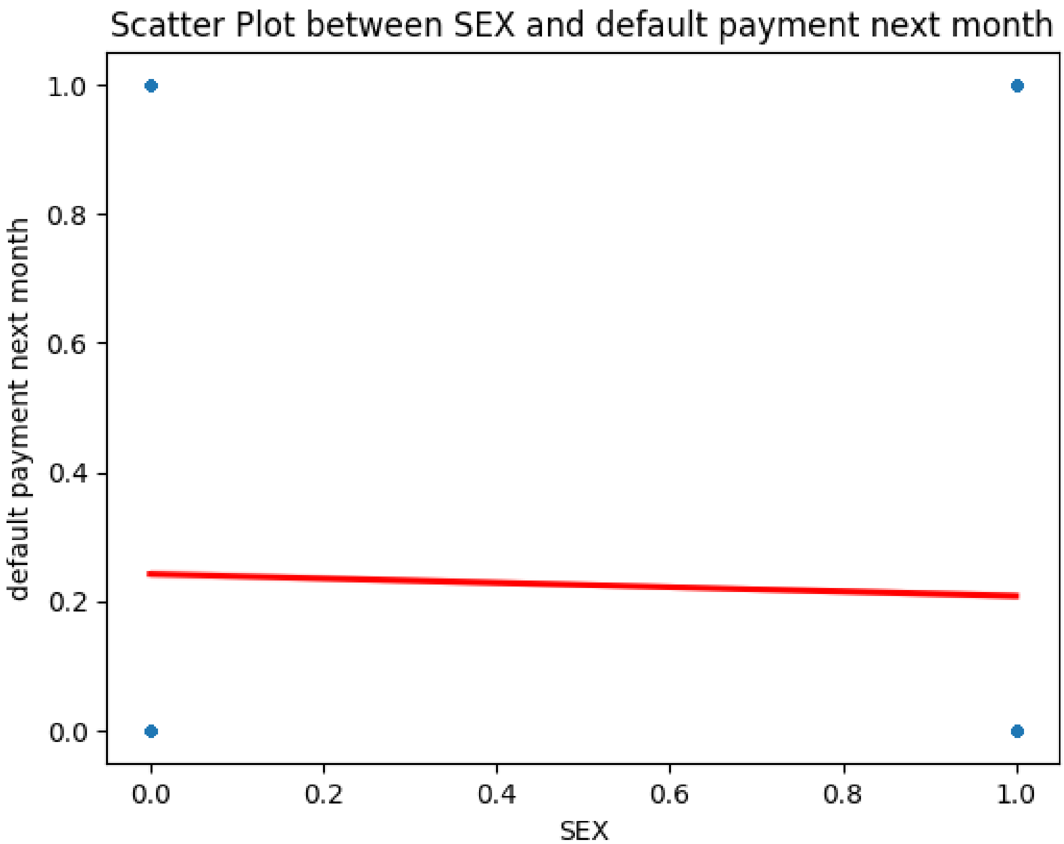


1.10 Correlation Analysis with default payment next month

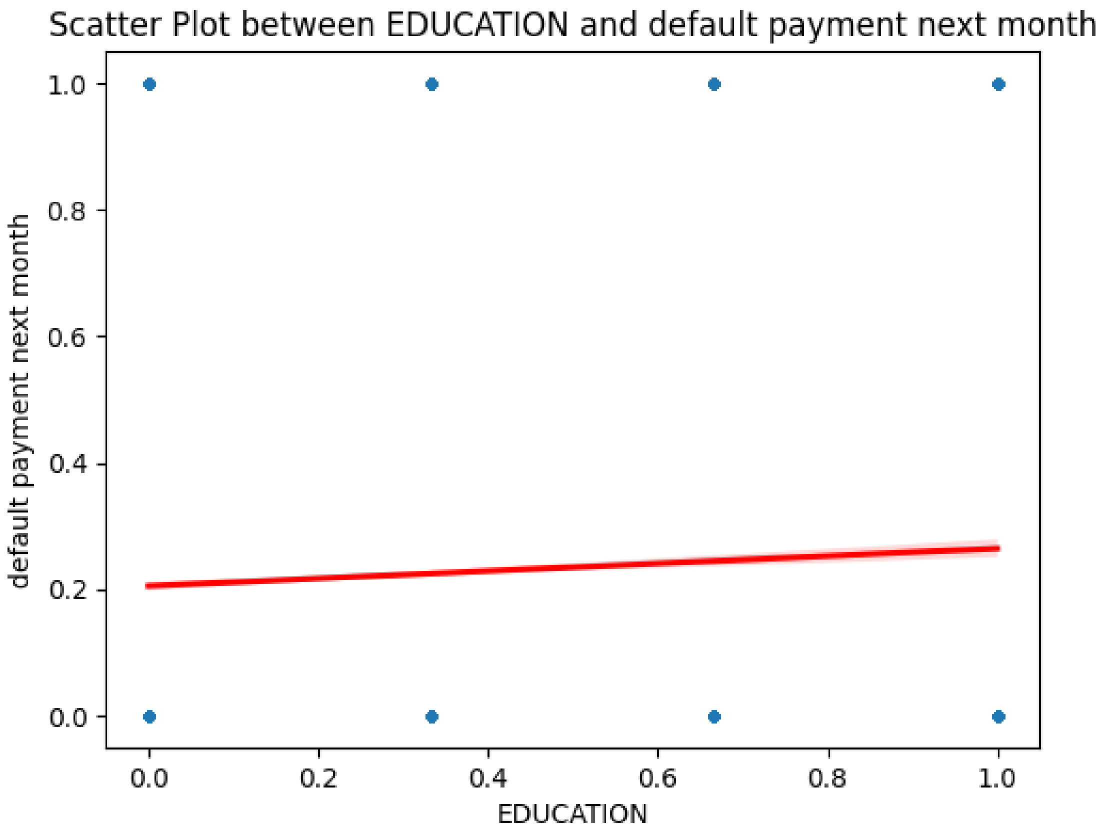
[ ]:
```
pearson_corr = df2_normalized['SEX'].corr(df2_normalized['default payment next␣
↪month'])

print(pearson_corr)
sns.regplot(x='SEX', y='default payment next month', data=df2_normalized,␣
↪scatter_kws={'s': 10}, line_kws={'color': 'red'})

# Adding labels and title
plt.xlabel('SEX')
plt.ylabel('default payment next month')
plt.title('Scatter Plot between SEX and default payment next month')
plt.show()


pearson_corr = df2_normalized['EDUCATION'].corr(df2_normalized['default payment␣
↪next month'])

print(pearson_corr)
```

```
sns.regplot(x='EDUCATION', y='default payment next month', data=df2_normalized,␣
↪scatter_kws={'s': 10}, line_kws={'color': 'red'})

# Adding labels and title
plt.xlabel('EDUCATION')
plt.ylabel('default payment next month')
plt.title('Scatter Plot between EDUCATION and default payment next month')
plt.show()


pearson_corr = df2_normalized['MARRIAGE'].corr(df2_normalized['default payment␣
↪next month'])

print(pearson_corr)
sns.regplot(x='MARRIAGE', y='default payment next month', data=df2_normalized,␣
↪scatter_kws={'s': 10}, line_kws={'color': 'red'})

# Adding labels and title
plt.xlabel('MARRIAGE')
plt.ylabel('default payment next month')
plt.title('Scatter Plot between MARRIAGE and default payment next month')
plt.show()


pearson_corr = df2_normalized['AGE'].corr(df2_normalized['default payment next␣
↪month'])

print(pearson_corr)
sns.regplot(x='AGE', y='default payment next month', data=df2_normalized,␣
↪scatter_kws={'s': 10}, line_kws={'color': 'red'})

# Adding labels and title
plt.xlabel('AGE')
plt.ylabel('default payment next month')
plt.title('Scatter Plot between AGE and default payment next month')
plt.show()
```

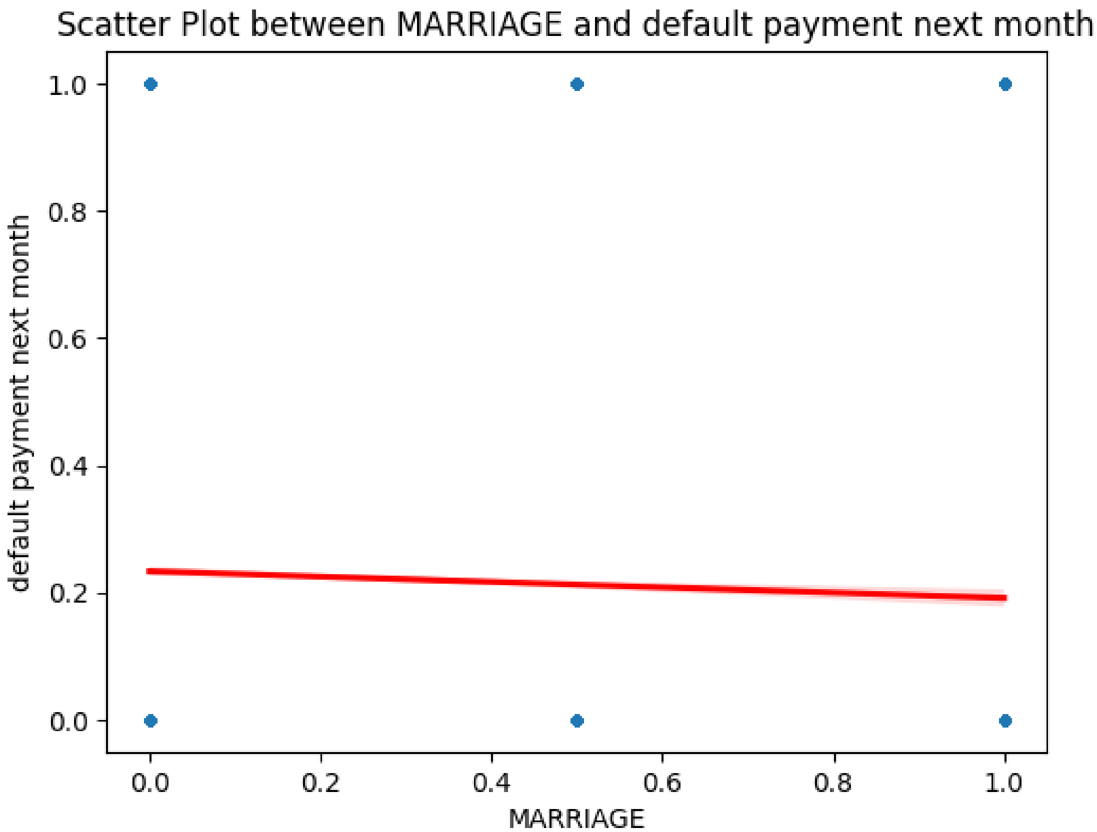
```
-0.03984349816263203
```



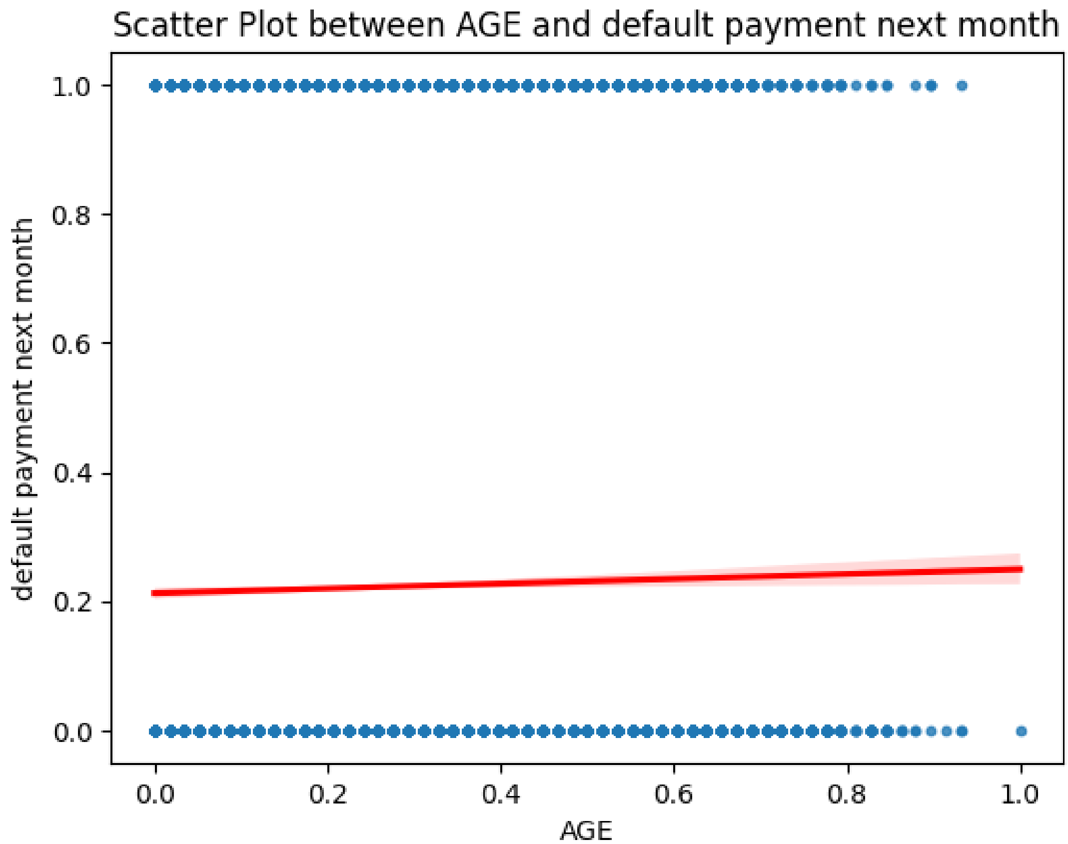

```
0.03536359285631867
```



-0.026154110816613868



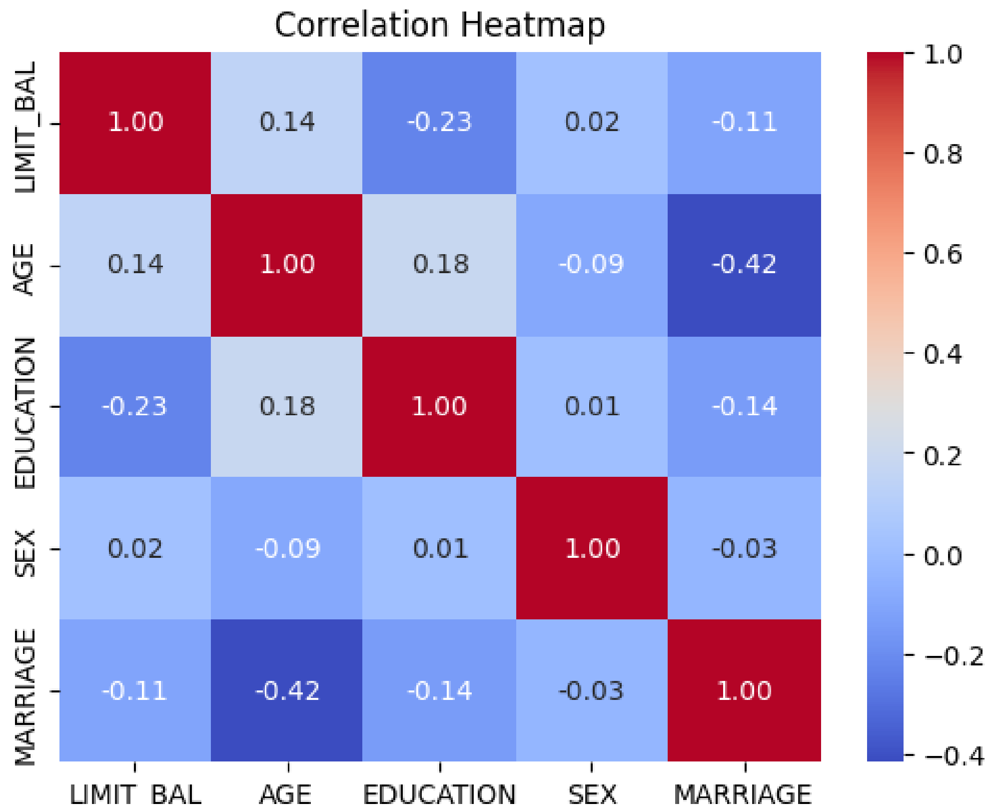

```
0.014224098031244962
```

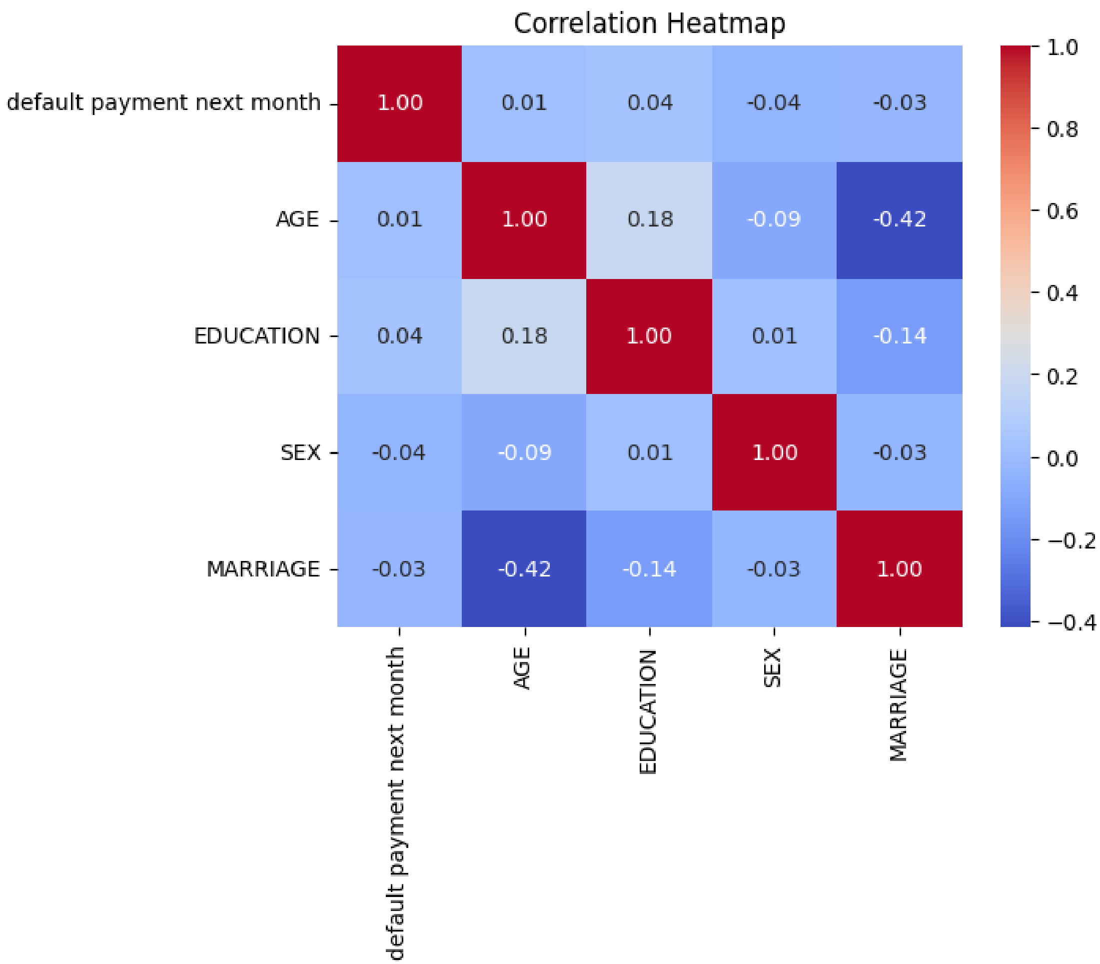


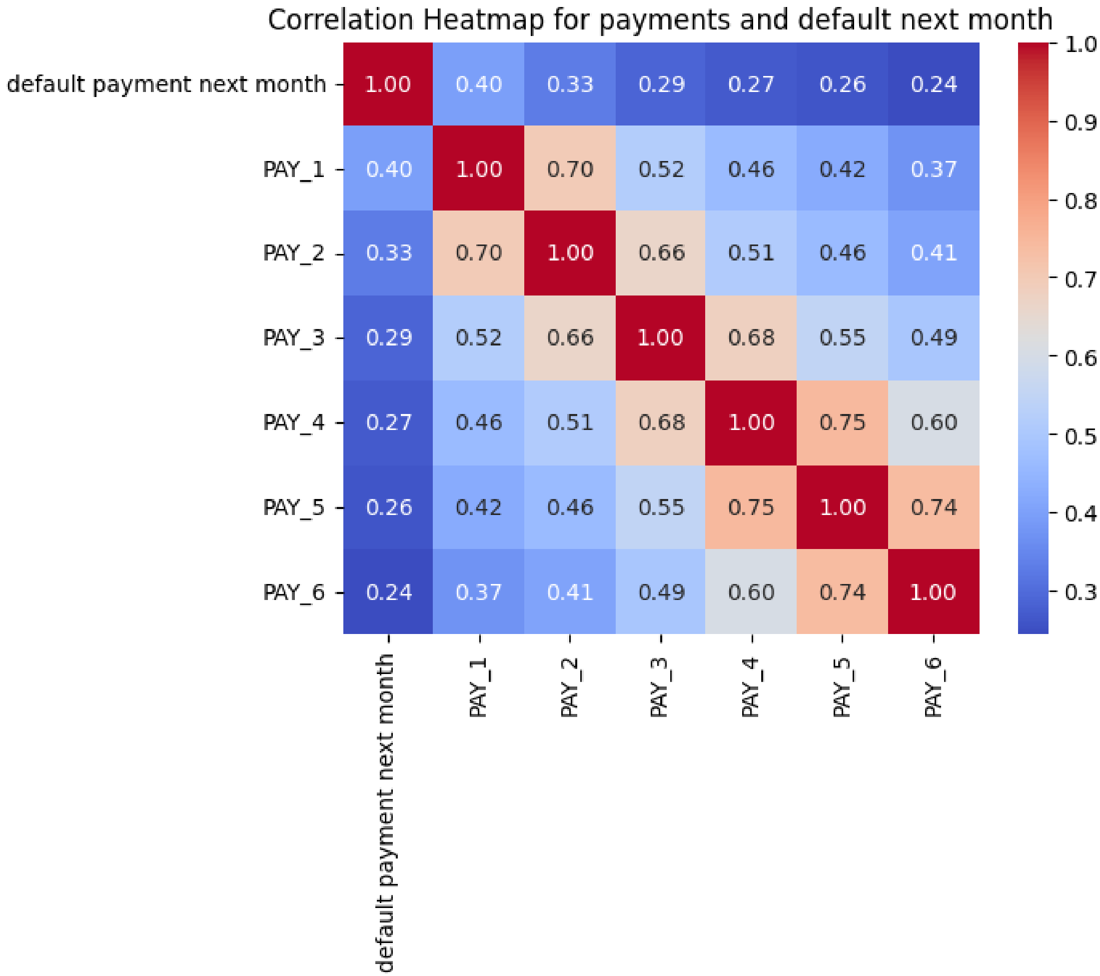
1.11 Correlation Heat map

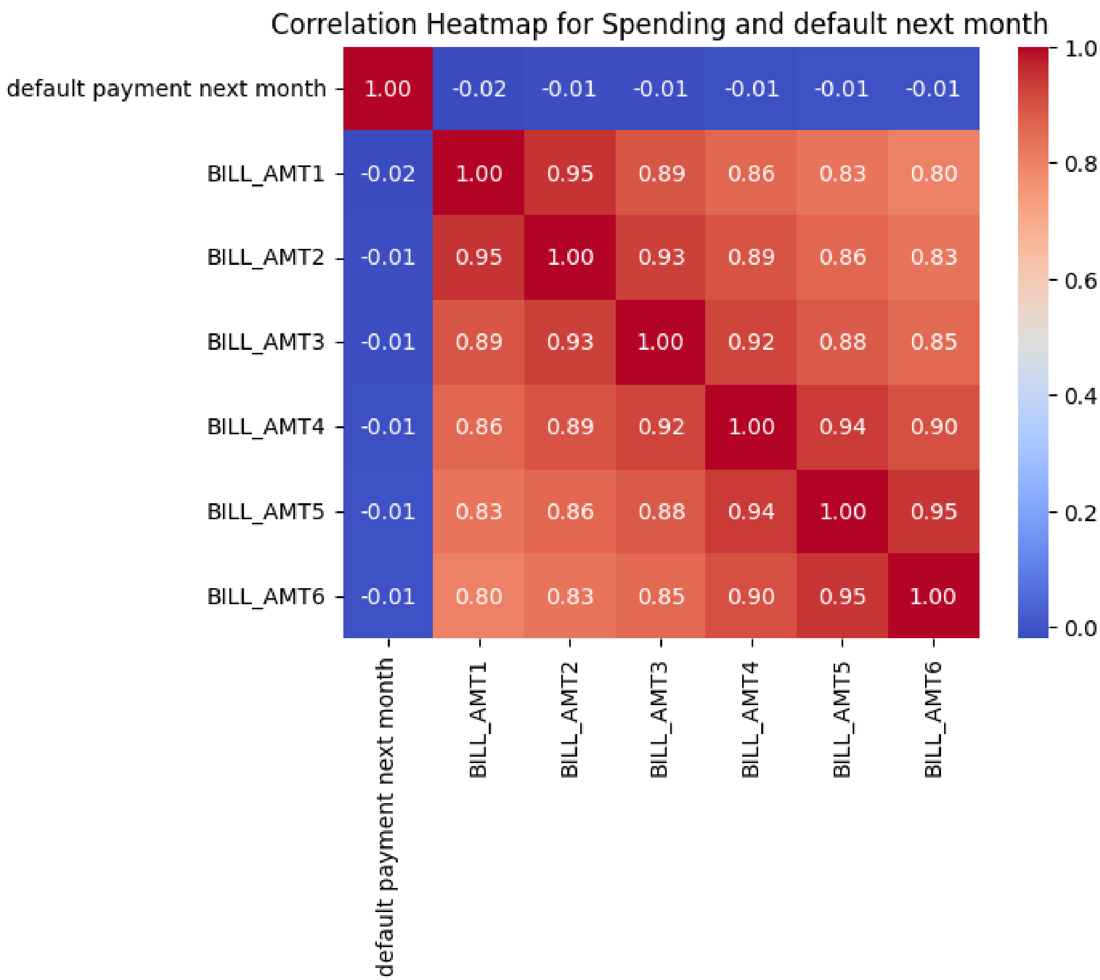
[ ]:
```
from sklearn.preprocessing import StandardScaler
heatmap_data1 = df2_normalized[['LIMIT_BAL',␣
↪'AGE','EDUCATION','SEX','MARRIAGE']]

# Create a heatmap for the correlation
correlation_matrix = heatmap_data1.corr()
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")

# Show the plot
plt.title("Correlation Heatmap")
plt.show()

heatmap_data2 = df2_normalized[['default payment next month',␣
↪'AGE','EDUCATION','SEX','MARRIAGE']]

correlation_matrix = heatmap_data2.corr()
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
```

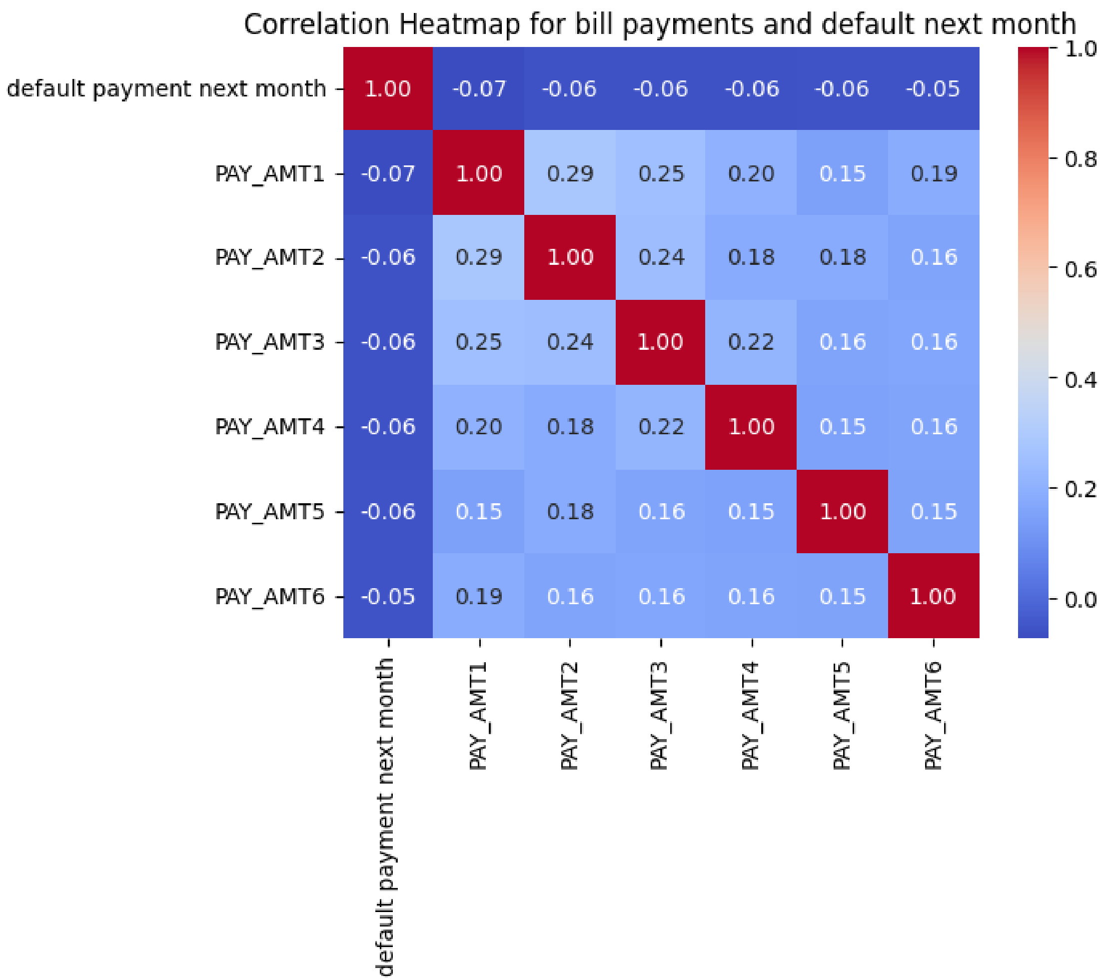
```python
# Show the plot
plt.title("Correlation Heatmap")
plt.show()

#Payment pattern and default next month correlation for PCA

heatmap_data2 = df2_normalized[['default payment next month', 'PAY_1',␣
↪'PAY_2','PAY_3','PAY_4','PAY_5','PAY_6']]

correlation_matrix = heatmap_data2.corr()
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")

# Show the plot
plt.title("Correlation Heatmap for payments and default next month")
plt.show()


heatmap_data2 = df2_normalized[['default payment next month','BILL_AMT1',␣
↪'BILL_AMT2','BILL_AMT3','BILL_AMT4','BILL_AMT5','BILL_AMT6']]

correlation_matrix = heatmap_data2.corr()
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")

# Show the plot
plt.title("Correlation Heatmap for Spending and default next month")
plt.show()


heatmap_data2 = df2_normalized[['default payment next month','PAY_AMT1',␣
↪'PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6']]
correlation_matrix = heatmap_data2.corr()
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")

# Show the plot
plt.title("Correlation Heatmap for bill payments and default next month")
plt.show()
```


Correlation Heatmap
LIMIT_BAL
AGE
EDUCATION
SEX
MARRIAGE
1.00
0.14
-0.23
0.02
-0.11
0.14
1.00
0.18
-0.09
-0.42
-0.23
0.18
1.00
0.01
-0.14
0.02
-0.09
0.01
1.00
-0.03
-0.11
-0.42
-0.14
-0.03
1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


Correlation Heatmap
default payment next month
AGE
EDUCATION
SEX
MARRIAGE
1.00 0.01 0.04 -0.04 -0.03
0.01 1.00 0.18 -0.09 -0.42
0.04 0.18 1.00 0.01 -0.14
-0.04 -0.09 0.01 1.00 -0.03
-0.03 -0.42 -0.14 -0.03 1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


Correlation Heatmap for payments and default next month
default payment next month
PAY_1
PAY_2
PAY_3
PAY_4
PAY_5
PAY_6
1.00 0.40 0.33 0.29 0.27 0.26 0.24
0.40 1.00 0.70 0.52 0.46 0.42 0.37
0.33 0.70 1.00 0.66 0.51 0.46 0.41
0.29 0.52 0.66 1.00 0.68 0.55 0.49
0.27 0.46 0.51 0.68 1.00 0.75 0.60
0.26 0.42 0.46 0.55 0.75 1.00 0.74
0.24 0.37 0.41 0.49 0.60 0.74 1.00
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3

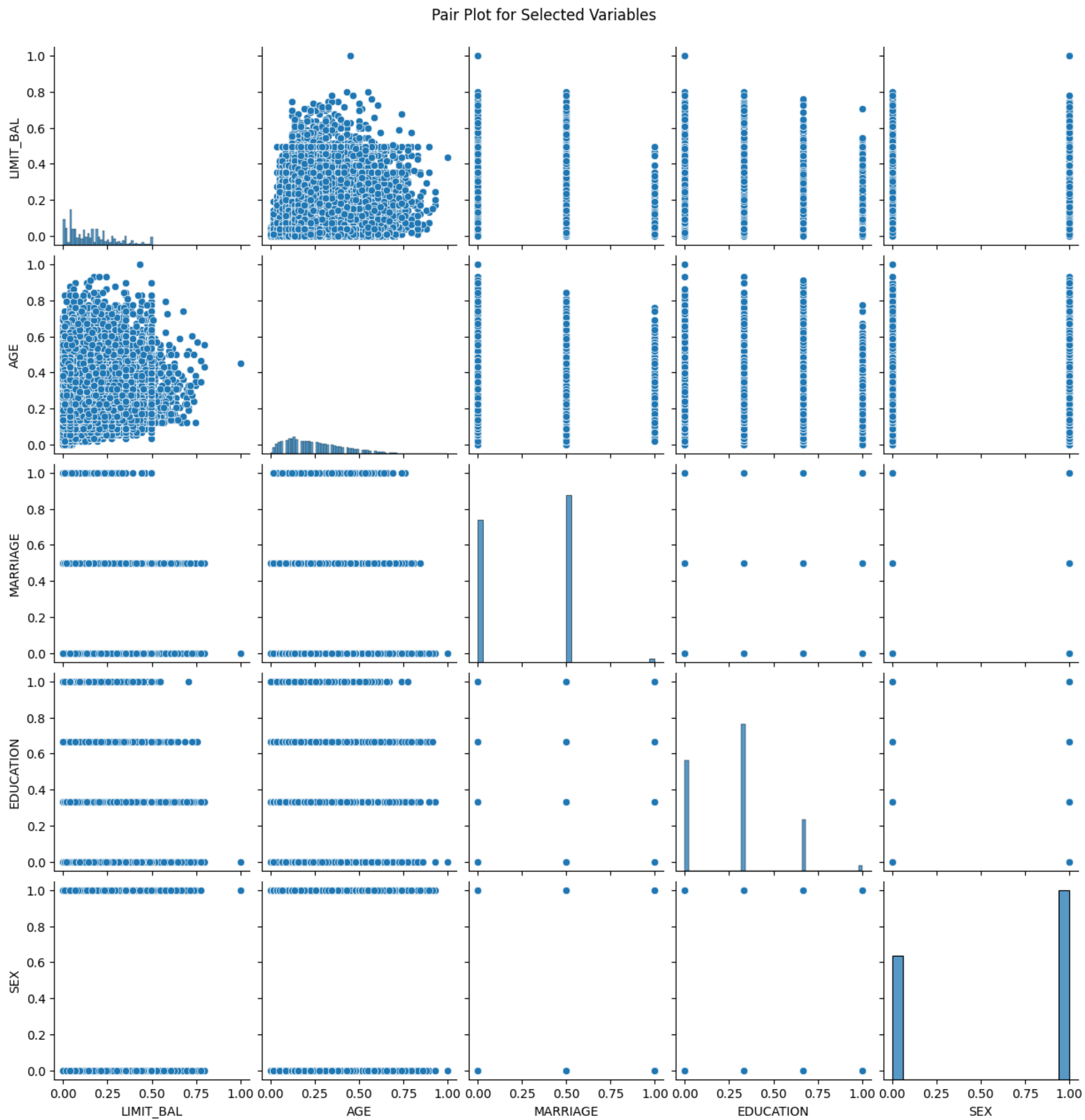

Correlation Heatmap for Spending and default next month
default payment next month
BILL_AMT1
BILL_AMT2
BILL_AMT3
BILL_AMT4
BILL_AMT5
BILL_AMT6
1.00 -0.02 -0.01 -0.01 -0.01 -0.01 -0.01
-0.02 1.00 0.95 0.89 0.86 0.83 0.80
-0.01 0.95 1.00 0.93 0.89 0.86 0.83
-0.01 0.89 0.93 1.00 0.92 0.88 0.85
-0.01 0.86 0.89 0.92 1.00 0.94 0.90
-0.01 0.83 0.86 0.88 0.94 1.00 0.95
-0.01 0.80 0.83 0.85 0.90 0.95 1.00
1.0
0.8
0.6
0.4
0.2
0.0

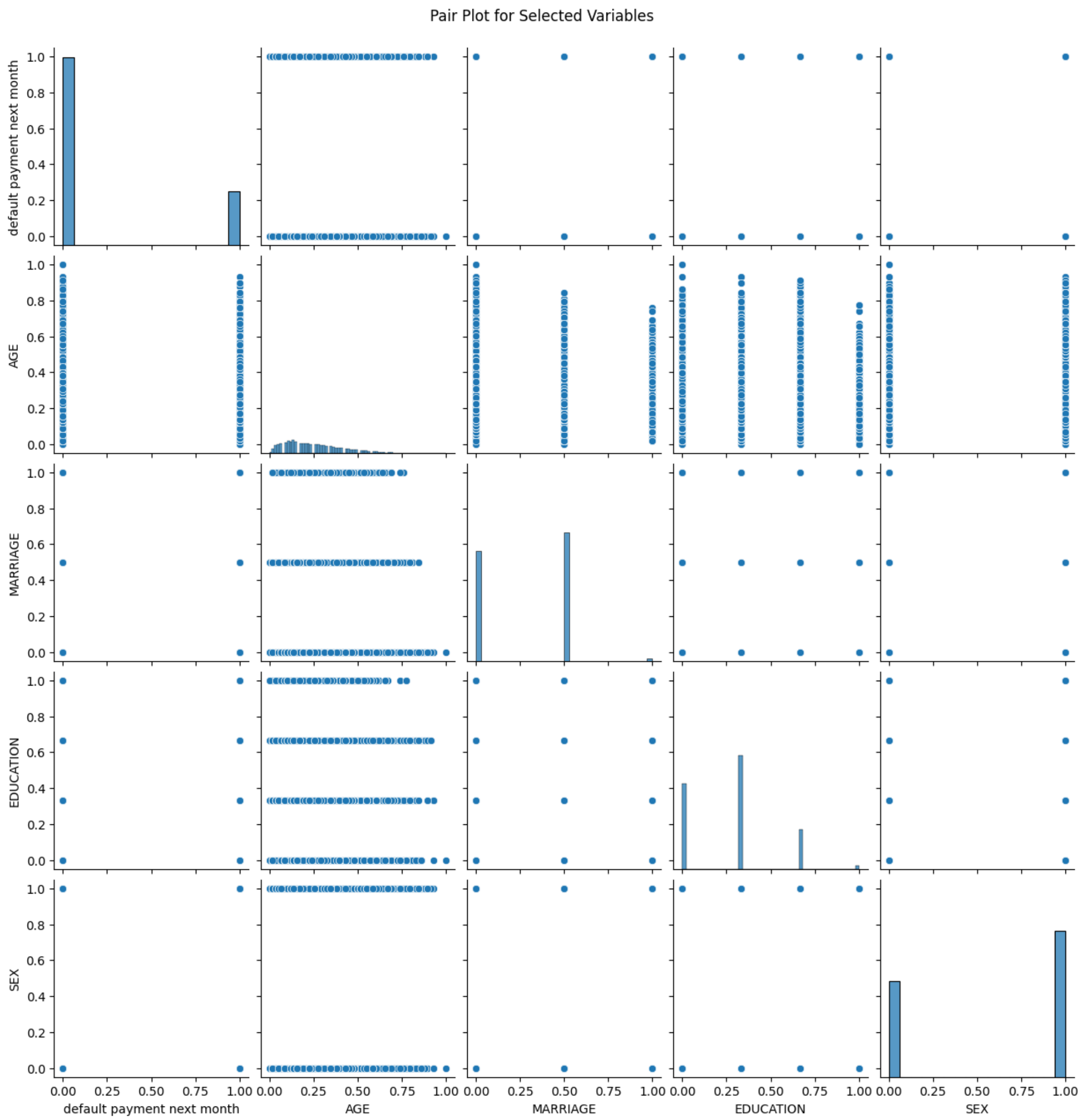


1.12 Pair Plot for demographic variables

[ ]:
```
#creating pair plot of Demographic variables, LIMIT_BAL and default payment␣
↪next month
selected_columns = ['LIMIT_BAL','AGE', 'MARRIAGE','EDUCATION','SEX']
pair_plot_data = df2_normalized[selected_columns]

# Create a pair plot
sns.pairplot(pair_plot_data)
plt.suptitle("Pair Plot for Selected Variables", y=1.02)
plt.show()

selected_columns2 = ['default payment next month','AGE',␣
↪'MARRIAGE','EDUCATION','SEX']
pair_plot_data = df2_normalized[selected_columns2]

# Create a pair plot
```

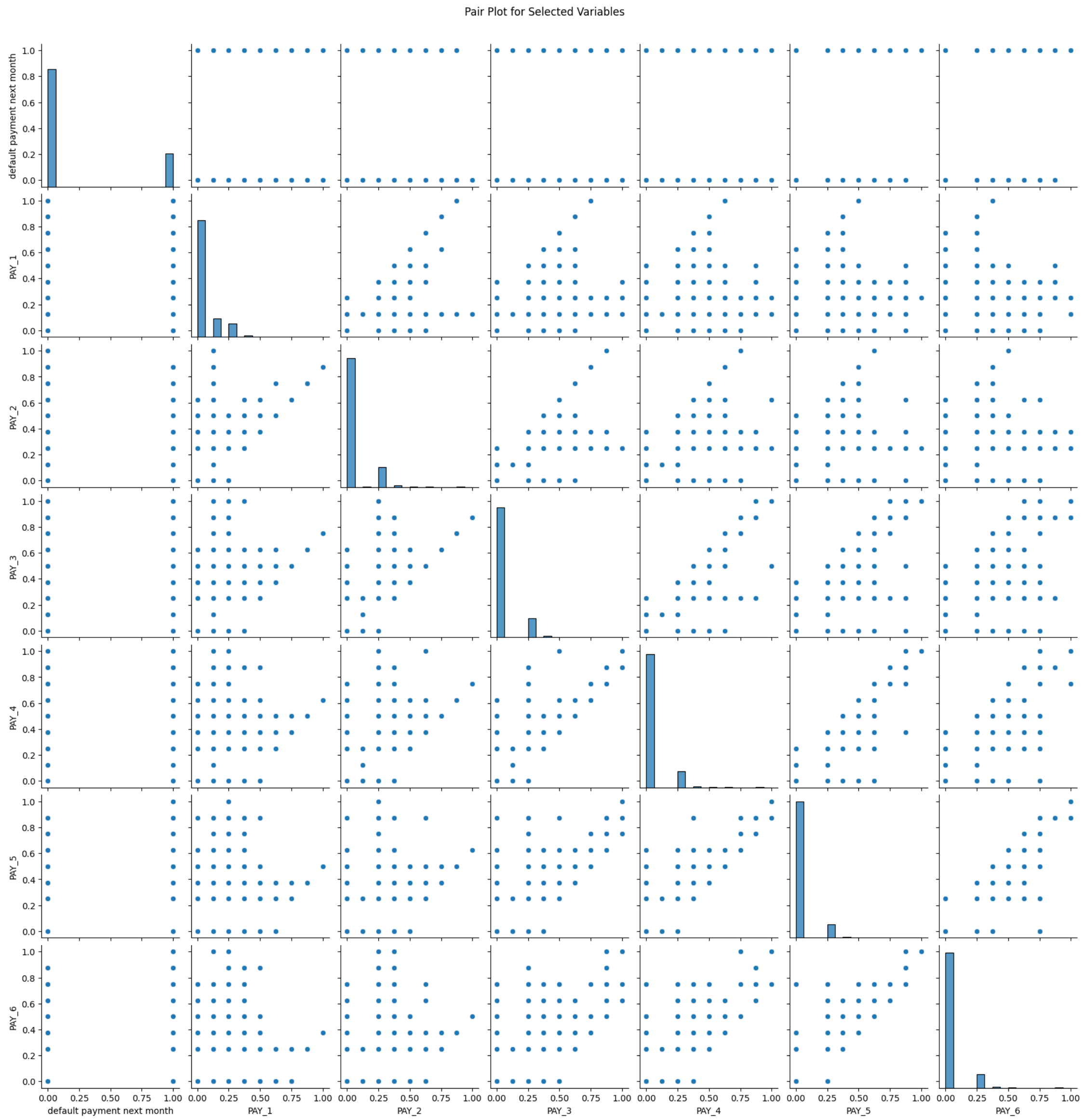
```
sns.pairplot(pair_plot_data)
plt.suptitle("Pair Plot for Selected Variables", y=1.02)
plt.show()
```



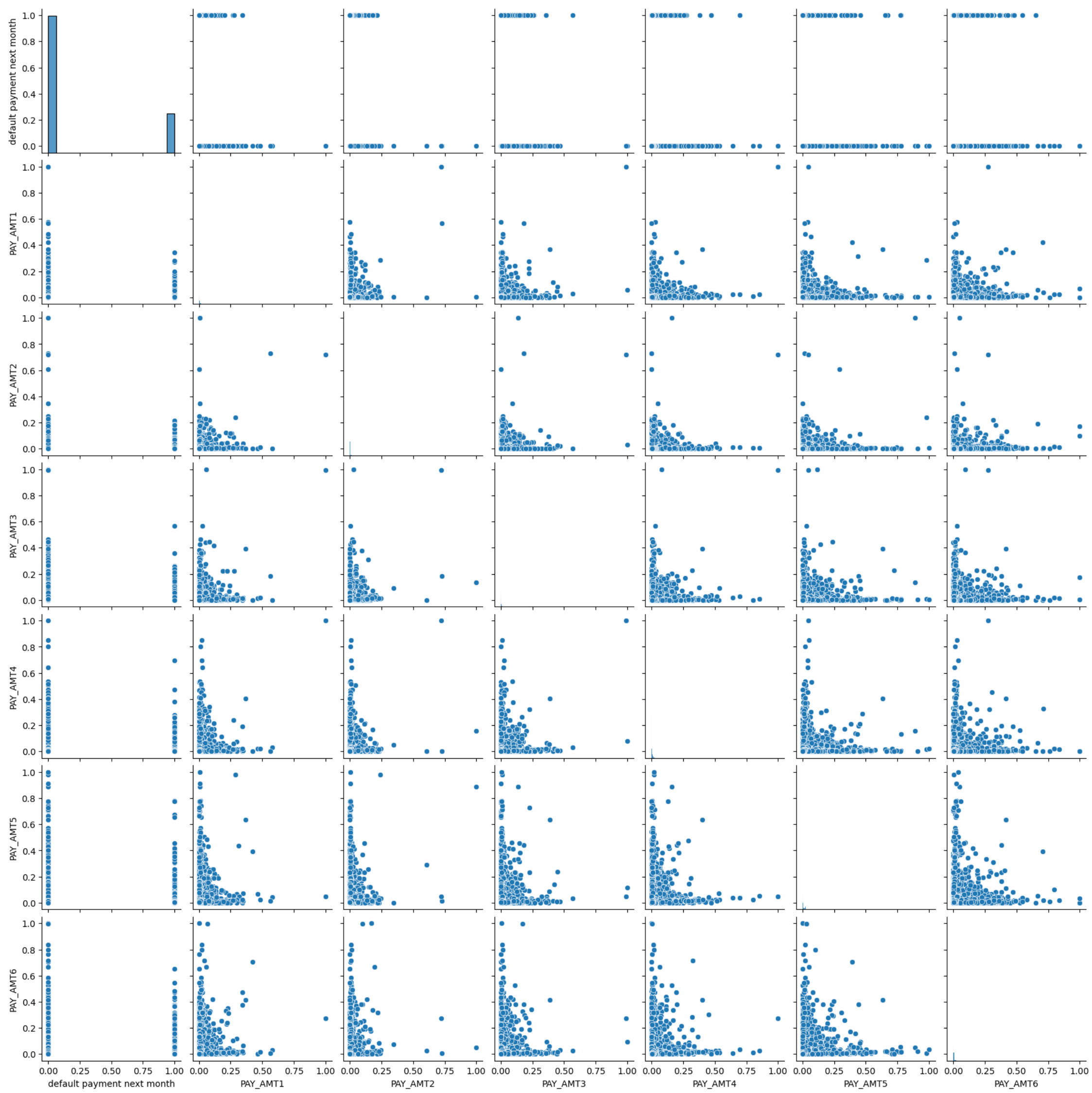


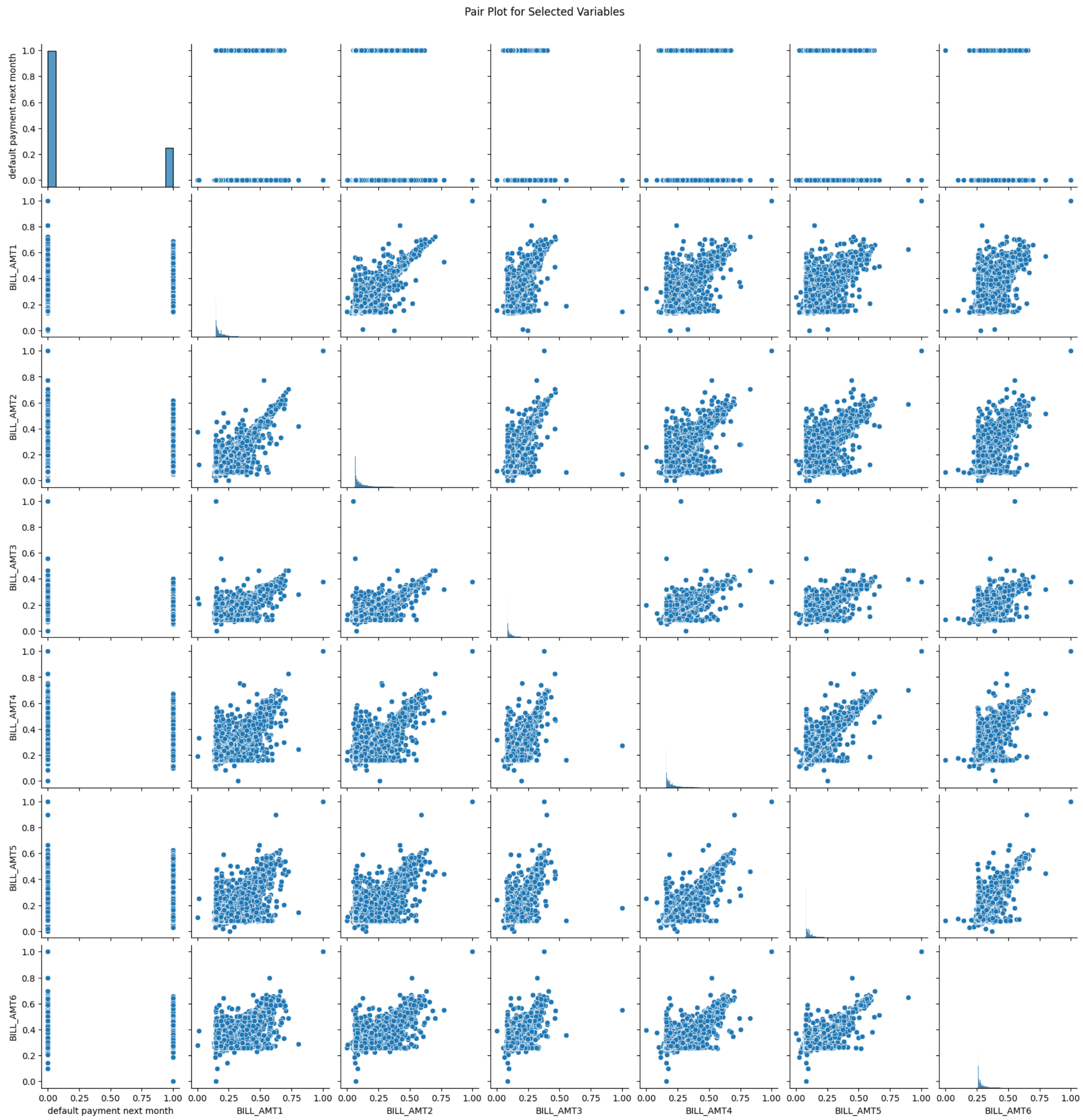

```
[ ]: #creating pair plot of payment behavior and default payment next month
     selected_columns3 = ['default payment next month','PAY_1',␣
      ↪'PAY_2','PAY_3','PAY_4','PAY_5','PAY_6']
     pair_plot_data = df2_normalized[selected_columns3]

     # Create a pair plot
     sns.pairplot(pair_plot_data)
     plt.suptitle("Pair Plot for Selected Variables", y=1.02)
     plt.show()

     selected_columns4 = ['default payment next month','PAY_AMT1',␣
      ↪'PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6']
```

```
pair_plot_data = df2_normalized[selected_columns4]

# Create a pair plot
sns.pairplot(pair_plot_data)
plt.suptitle("Pair Plot for Selected Variables", y=1.02)
plt.show()



selected_columns5 = ['default payment next month','BILL_AMT1',␣
 ↪'BILL_AMT2','BILL_AMT3','BILL_AMT4','BILL_AMT5','BILL_AMT6']
pair_plot_data = df2_normalized[selected_columns5]

# Create a pair plot
sns.pairplot(pair_plot_data)
plt.suptitle("Pair Plot for Selected Variables", y=1.02)
plt.show()
```

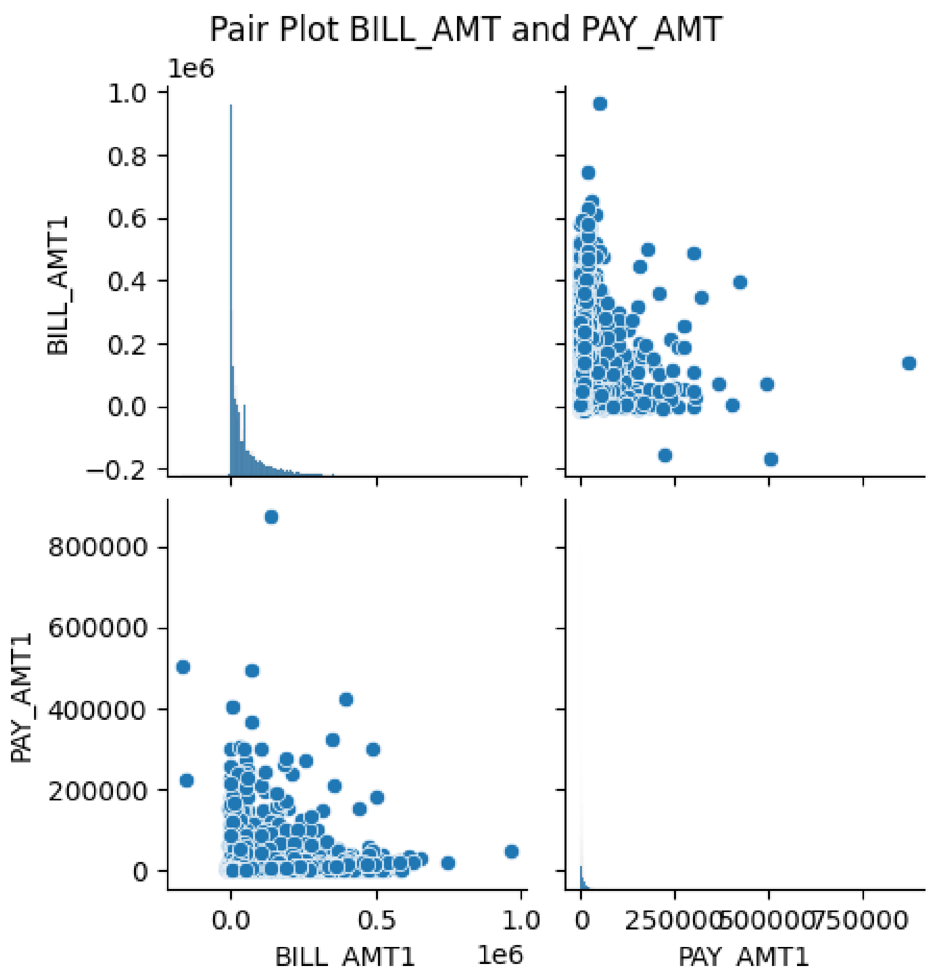

Pair Plot for Selected Variables
default payment next month
PAY_1
PAY_2
PAY_3
PAY_4
PAY_5
PAY_6
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.25
0.50
0.75
1.00

Pair Plot for Selected Variables





[ ]:
```
selected_columns = ['BILL_AMT1','PAY_AMT1']
pair_plot_data = df2[selected_columns]

# Create a pair plot
sns.pairplot(pair_plot_data)
plt.suptitle("Pair Plot BILL_AMT and PAY_AMT", y=1.02)
plt.show()
```



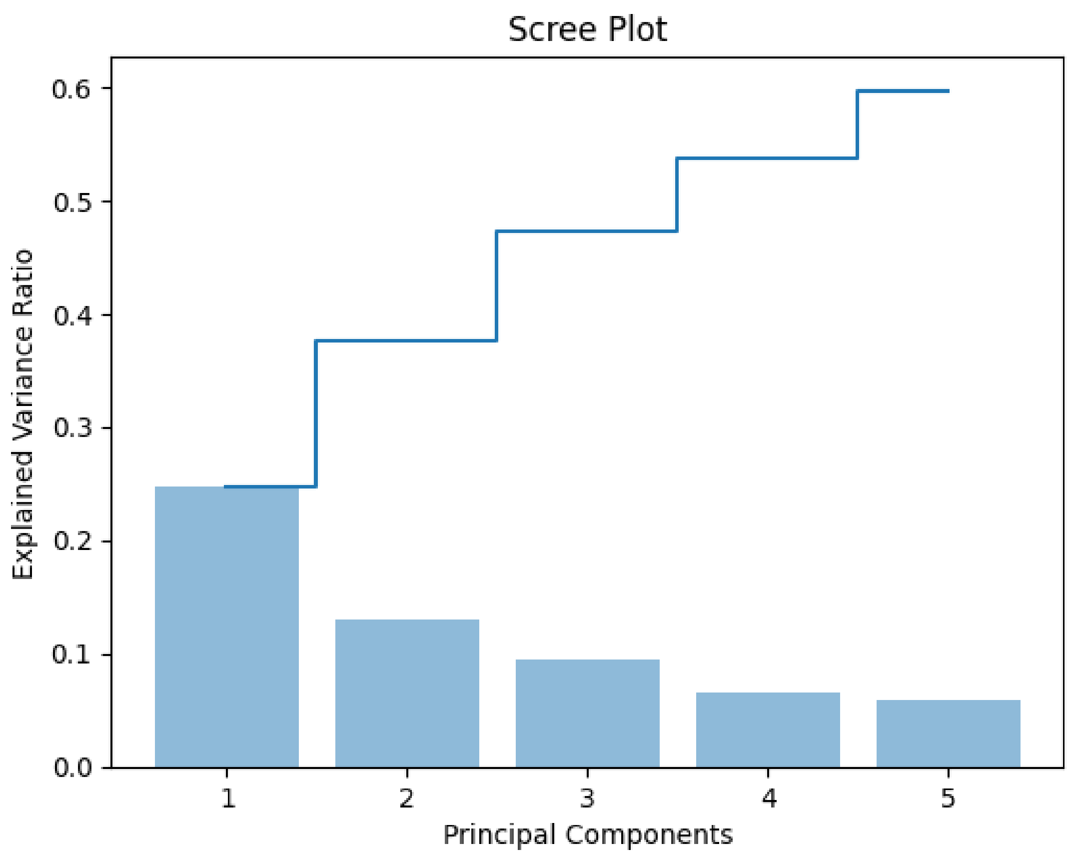
1.13 Principal Component Analysis

```
[ ]: #Principal component analysis is conducted for the dimentionality reduction
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

features1 = ['LIMIT_BAL', 'AGE','EDUCATION', 'MARRIAGE', 'PAY_1','PAY_2',␣
 ↪'PAY_3',
             'PAY_4', 'PAY_5', 'PAY_6','PAY_AMT1',
             'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6']
pca_df_columns=df2[features1]
scaler = StandardScaler()
features1_standardized = scaler.fit_transform(pca_df_columns)

# Apply PCA with five components
num_components = 5
```

```python
# Apply PCA with the specified number of components
pca = PCA(n_components=num_components)
principal_components = pca.fit_transform(features1_standardized)
# Explained variance ratio
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_explained_variance = explained_variance_ratio.cumsum()

# Create a DataFrame with the principal components
columns_pca = [f'PC{i}' for i in range(1, num_components + 1)]
df_pca = pd.DataFrame(data=principal_components, columns=columns_pca)

# Concatenate the original DataFrame with the PCA DataFrame
df_with_pca = pd.concat([pca_df_columns, df_pca], axis=1)

# Display the resulting DataFrame
print(df_with_pca.head())

# Plot the scree plot
plt.bar(range(1, len(pca.explained_variance_ratio_) + 1), pca.
 ↪explained_variance_ratio_, alpha=0.5, align='center')
plt.step(range(1, len(pca.explained_variance_ratio_) + 1), pca.
 ↪explained_variance_ratio_.cumsum(), where='mid')
plt.xlabel('Principal Components')
plt.ylabel('Explained Variance Ratio')
plt.title('Scree Plot')
plt.show()

# Print the explained variance ratio
print("Explained Variance Ratio:")
for i, ratio in enumerate(pca.explained_variance_ratio_):
    print(f"PC{i + 1}: {ratio:.4f}")
```

```
   LIMIT_BAL   AGE  EDUCATION  MARRIAGE  PAY_1  PAY_2  PAY_3  PAY_4  PAY_5  \
0    20000.0  24.0        2.0       1.0    2.0    2.0    0.0    0.0    0.0
1   120000.0  26.0        2.0       2.0    0.0    2.0    0.0    0.0    0.0
2    90000.0  34.0        2.0       2.0    0.0    0.0    0.0    0.0    0.0
3    50000.0  37.0        2.0       1.0    0.0    0.0    0.0    0.0    0.0
4    50000.0  57.0        2.0       1.0    0.0    0.0    0.0    0.0    0.0

   PAY_6  …  PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6       PC1  \
0    0.0  …     689.0       0.0       0.0       0.0       0.0 -1.426849
1    2.0  …    1000.0    1000.0    1000.0       0.0    2000.0 -1.376657
2    0.0  …    1500.0    1000.0    1000.0    1000.0    5000.0  0.633666
3    0.0  …    2019.0    1200.0    1100.0    1069.0    1000.0  0.581719
4    0.0  …   36681.0   10000.0    9000.0     689.0     679.0  0.826469

        PC2       PC3       PC4       PC5
```

```
0 -0.932142 -0.056344  0.538526 -2.200192
1 -0.560279  1.136244  0.350529  0.118354
2 -0.972100  0.545119  0.511617  0.094271
3 -0.944609 -0.881759  0.297464  0.264844
4  0.208086 -2.189210  0.603687  0.521773

[5 rows x 21 columns]
```




```
Explained Variance Ratio:
PC1: 0.2467
PC2: 0.1298
PC3: 0.0952
PC4: 0.0654
PC5: 0.0597
```

1.14 Data Balancing

[ ]:
```
#Data is imbalanced, data needs to be balanced to get an efficient model
#Create Train dataset and test dataset: considering the correlation the␣
↪demographic variables are: "AGE","EDUCATION","MARRIAGE"
```

```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from imblearn.over_sampling import SMOTE

features = ['LIMIT_BAL', 'AGE','EDUCATION', 'MARRIAGE', 'PAY_1','PAY_2',␣
↪'PAY_3',
            'PAY_4', 'PAY_5', 'PAY_6','BILL_AMT1', 'BILL_AMT2',
            'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1',
            'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6']

y = df2_normalized['default payment next month'].copy() #Target variable
X = df2_normalized[features].copy()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,␣
↪random_state=42)

# Apply oversample the minority class
smote = SMOTE(sampling_strategy='auto', random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)

df_train = X_train.join(y_train)
print(df_train['default payment next month'].value_counts())

df_majority = df_train[df_train['default payment next month'] == 0]

df_minority = df_train[df_train['default payment next month'] == 1]


from sklearn.utils import resample

df_minority_upsampled = resample(df_minority,replace=True,␣
↪n_samples=18641,random_state=587)
# Combine majority class with upsampled minority class
df_upsampled= pd.concat([df_majority, df_minority_upsampled])
# Display new class counts
print(df_upsampled['default payment next month'].value_counts())


#Apply downsample to minority class
df_majority_downsampled = resample(df_majority,replace=True,␣
↪n_samples=5304,random_state=587)
# Combine minority class with downsampled majority class
df_downsampled= pd.concat([df_minority, df_majority_downsampled])
# Display new class counts
print(df_downsampled['default payment next month'].value_counts())
```

```python
#So we have 2 dataset, Upsampled data creates synthetic data and downsampled␣
↪data creates bias.
```

```
0.0    18641
1.0     5304
Name: default payment next month, dtype: int64
0.0    18641
1.0    18641
Name: default payment next month, dtype: int64
1.0    5304
0.0    5304
Name: default payment next month, dtype: int64
```