6dxawsirt

January 18, 2023

[1]:
```
import numpy as np
import pandas as pd
import random
import matplotlib.pyplot as plt
import seaborn as sns
```

ID: ID of each client LIMIT_BAL: Amount of given credit in NT dollars (includes individual and family/supplementary credit SEX: Gender (1=male, 2=female) EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown) MARRIAGE: Marital status (1=married, 2=single, 3=others) AGE: Age in years PAY_0: Repayment status in September, 2005 (-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) PAY_2: Repayment status in August, 2005 (scale same as above) PAY_3: Repayment status in July, 2005 (scale same as above) PAY_4: Repayment status in June, 2005 (scale same as above) PAY_5: Repayment status in May, 2005 (scale same as above) PAY_6: Repayment status in April, 2005 (scale same as above) BILL_AMT1: Amount of bill statement in September, 2005 (NT dollar) BILL_AMT2: Amount of bill statement in August, 2005 (NT dollar) BILL_AMT3: Amount of bill statement in July, 2005 (NT dollar) BILL_AMT4: Amount of bill statement in June, 2005 (NT dollar) BILL_AMT5: Amount of bill statement in May, 2005 (NT dollar) BILL_AMT6: Amount of bill statement in April, 2005 (NT dollar) PAY_AMT1: Amount of previous payment in September, 2005 (NT dollar) PAY_AMT2: Amount of previous payment in August, 2005 (NT dollar) PAY_AMT3: Amount of previous payment in July, 2005 (NT dollar) PAY_AMT4: Amount of previous payment in June, 2005 (NT dollar) PAY_AMT5: Amount of previous payment in May, 2005 (NT dollar) PAY_AMT6: Amount of previous payment in April, 2005 (NT dollar) default.payment.next.month: Default payment (1=yes, 0=no)

[2]:
```
df=pd.read_csv("UCI_Credit_Card.csv")
df
```

[2]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	\
0	1	20000.0	2	2	1	24	2	2	-1	
1	2	120000.0	2	2	2	26	-1	2	0	
2	3	90000.0	2	2	2	34	0	0	0	
3	4	50000.0	2	2	1	37	0	0	0	
4	5	50000.0	1	2	1	57	-1	0	-1	
…	…	…	…	…	…	…	…	…	…	
29995	29996	220000.0	1	3	1	39	0	0	0	
29996	29997	150000.0	1	3	2	43	-1	-1	-1	

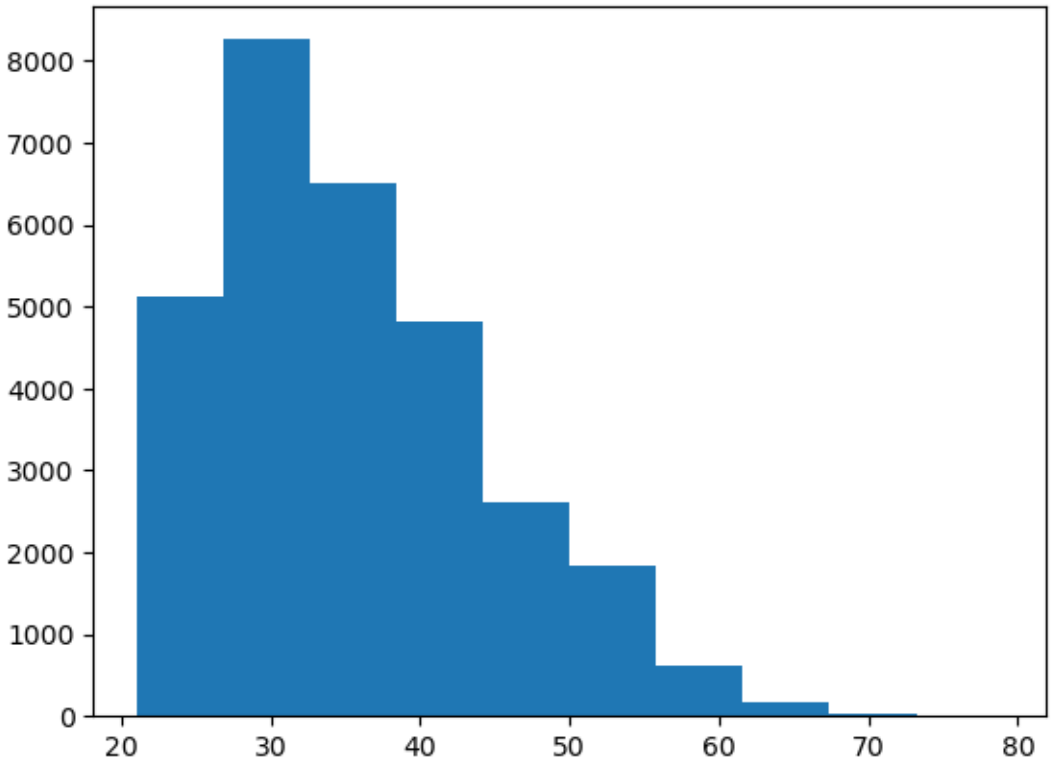
```
29997  29998    30000.0    1          2         2   37      4      3      2
29998  29999    80000.0    1          3         1   41      1     -1      0
29999  30000    50000.0    1          2         1   46      0      0      0

       PAY_4  …  BILL_AMT4  BILL_AMT5  BILL_AMT6  PAY_AMT1  PAY_AMT2  \
0         -1  …        0.0        0.0        0.0       0.0     689.0
1          0  …     3272.0     3455.0     3261.0       0.0    1000.0
2          0  …    14331.0    14948.0    15549.0    1518.0    1500.0
3          0  …    28314.0    28959.0    29547.0    2000.0    2019.0
4          0  …    20940.0    19146.0    19131.0    2000.0   36681.0
…        …  …        …          …          …         …
29995      0  …    88004.0    31237.0    15980.0    8500.0   20000.0
29996     -1  …     8979.0     5190.0        0.0    1837.0    3526.0
29997     -1  …    20878.0    20582.0    19357.0       0.0       0.0
29998      0  …    52774.0    11855.0    48944.0   85900.0    3409.0
29999      0  …    36535.0    32428.0    15313.0    2078.0    1800.0

       PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6  default.payment.next.month
0           0.0       0.0       0.0       0.0                          1
1        1000.0    1000.0       0.0    2000.0                          1
2        1000.0    1000.0    1000.0    5000.0                          0
3        1200.0    1100.0    1069.0    1000.0                          0
4       10000.0    9000.0     689.0     679.0                          0
…           …         …         …         …                          …
29995    5003.0    3047.0    5000.0    1000.0                          0
29996    8998.0     129.0       0.0       0.0                          0
29997   22000.0    4200.0    2000.0    3100.0                          1
29998    1178.0    1926.0   52964.0    1804.0                          1
29999    1430.0    1000.0    1000.0    1000.0                          1

[30000 rows x 25 columns]
```

[3]:
```
#checking the null value
df.isnull().sum()
```

[3]:
```
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
```

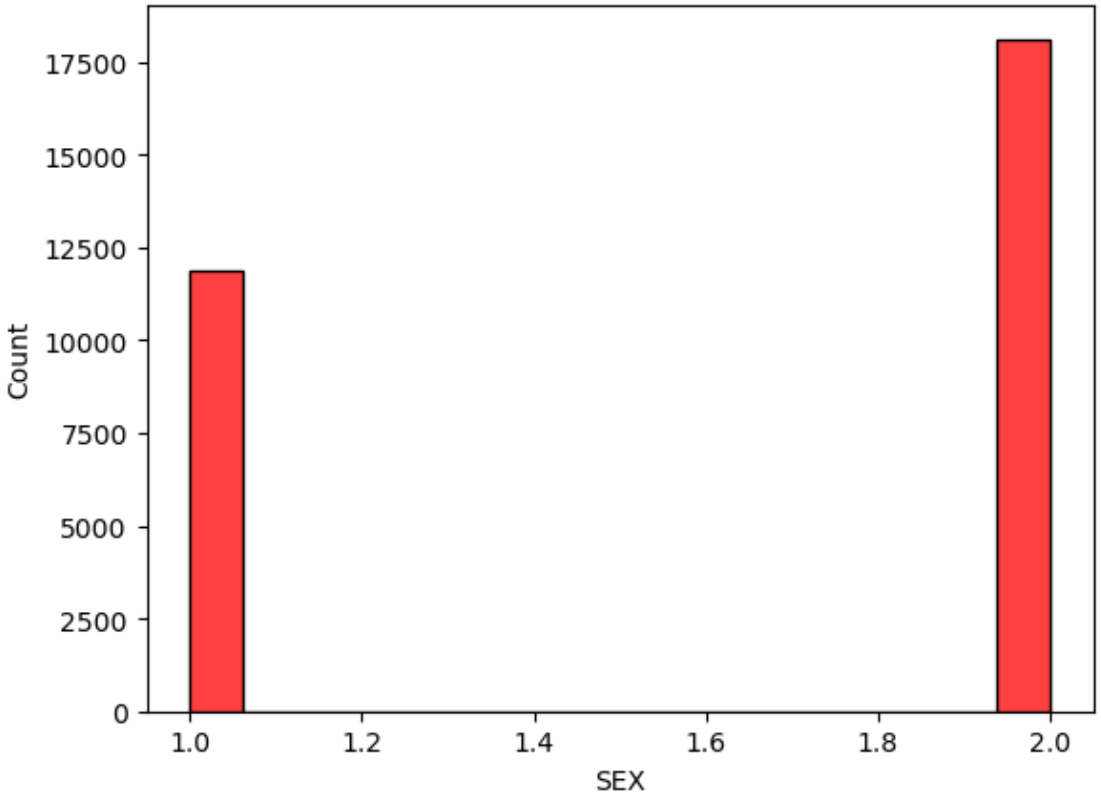
```
BILL_AMT1                    0
BILL_AMT2                    0
BILL_AMT3                    0
BILL_AMT4                    0
BILL_AMT5                    0
BILL_AMT6                    0
PAY_AMT1                     0
PAY_AMT2                     0
PAY_AMT3                     0
PAY_AMT4                     0
PAY_AMT5                     0
PAY_AMT6                     0
default.payment.next.month   0
dtype: int64
```

[4]:
```
#age between 28 to 35 are more
plt.hist(df['AGE'])
plt.show()
```



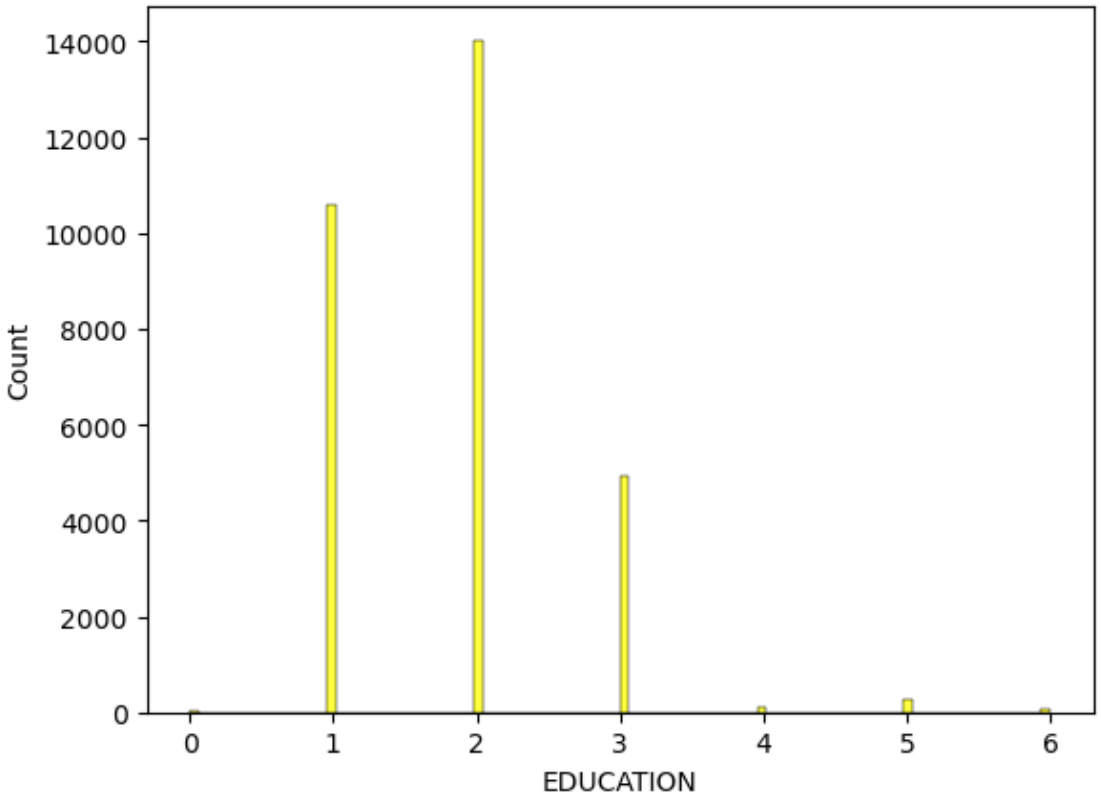
[5]:
```
#number of females are more than number of males(1=male,2=female)
sns.histplot(df['SEX'],color='r')
```

[5]: <AxesSubplot:xlabel='SEX', ylabel='Count'>



[6]:
```
#more number of people are from university
#university > graduate school > high school > others(1=graduate school,␣
 ↪2=university, 3=high school, 4=others)
sns.histplot(df['EDUCATION'],color='yellow')
```

[6]: <AxesSubplot:xlabel='EDUCATION', ylabel='Count'>



[7]:
```
#maximum age of a person=79
#75% people are from 41 age group
#50% people are from 34 age group
#25% people are from 28 age group
#minimum age of a person=21
df.describe()
```

[7]:

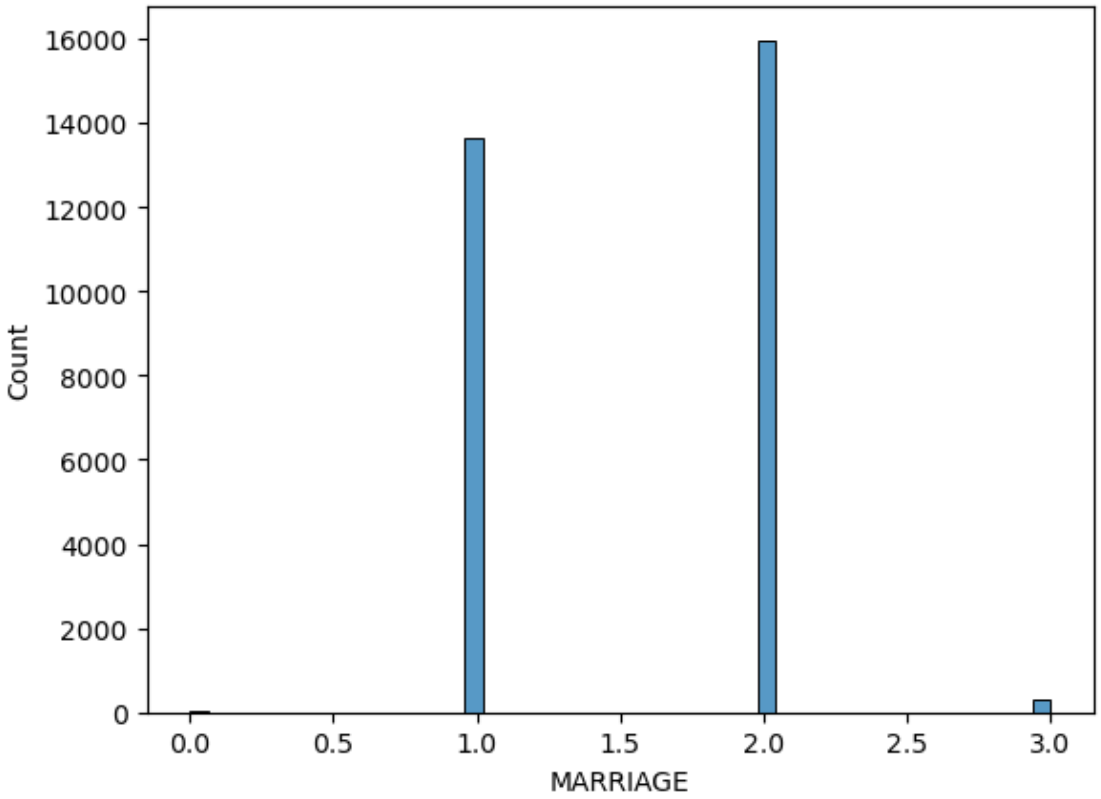
	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	\
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	
mean	15000.500000	167484.322667	1.603733	1.853133	1.551867	
std	8660.398374	129747.661567	0.489129	0.790349	0.521970	
min	1.000000	10000.000000	1.000000	0.000000	0.000000	
25%	7500.750000	50000.000000	1.000000	1.000000	1.000000	
50%	15000.500000	140000.000000	2.000000	2.000000	2.000000	
75%	22500.250000	240000.000000	2.000000	2.000000	2.000000	
max	30000.000000	1000000.000000	2.000000	6.000000	3.000000	

	AGE	PAY_0	PAY_2	PAY_3	PAY_4	\
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	
mean	35.485500	-0.016700	-0.133767	-0.166200	-0.220667	
std	9.217904	1.123802	1.197186	1.196868	1.169139	

```
min       21.000000     -2.000000     -2.000000     -2.000000     -2.000000
25%       28.000000     -1.000000     -1.000000     -1.000000     -1.000000
50%       34.000000      0.000000      0.000000      0.000000      0.000000
75%       41.000000      0.000000      0.000000      0.000000      0.000000
max       79.000000      8.000000      8.000000      8.000000      8.000000

       …      BILL_AMT4      BILL_AMT5      BILL_AMT6       PAY_AMT1  \
count  …   30000.000000   30000.000000   30000.000000   30000.000000
mean   …   43262.948967   40311.400967   38871.760400    5663.580500
std    …   64332.856134   60797.155770   59554.107537   16563.280354
min    … -170000.000000  -81334.000000 -339603.000000       0.000000
25%    …    2326.750000    1763.000000    1256.000000    1000.000000
50%    …   19052.000000   18104.500000   17071.000000    2100.000000
75%    …   54506.000000   50190.500000   49198.250000    5006.000000
max    …  891586.000000  927171.000000  961664.000000  873552.000000

           PAY_AMT2      PAY_AMT3       PAY_AMT4       PAY_AMT5  \
count  3.000000e+04   30000.00000   30000.000000   30000.000000
mean   5.921163e+03    5225.68150    4826.076867    4799.387633
std    2.304087e+04   17606.96147   15666.159744   15278.305679
min    0.000000e+00       0.00000       0.000000       0.000000
25%    8.330000e+02     390.00000     296.000000     252.500000
50%    2.009000e+03    1800.00000    1500.000000    1500.000000
75%    5.000000e+03    4505.00000    4013.250000    4031.500000
max    1.684259e+06  896040.00000  621000.000000  426529.000000

            PAY_AMT6  default.payment.next.month
count   30000.000000                30000.000000
mean     5215.502567                    0.221200
std     17777.465775                    0.415062
min         0.000000                    0.000000
25%       117.750000                    0.000000
50%      1500.000000                    0.000000
75%      4000.000000                    0.000000
max    528666.000000                    1.000000

[8 rows x 25 columns]
```

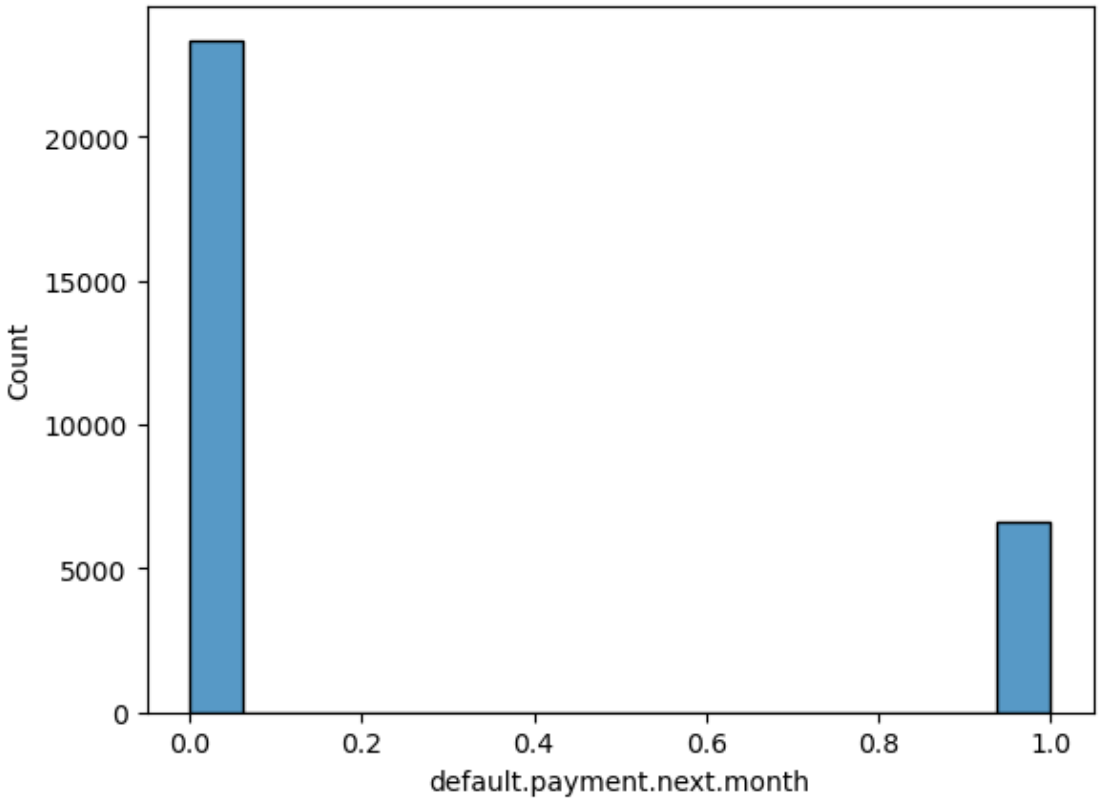
[8]:
```
#single > married > others (1=married,2=single,3=others)
sns.histplot(df['MARRIAGE'])
```

[8]:
```
<AxesSubplot:xlabel='MARRIAGE', ylabel='Count'>
```



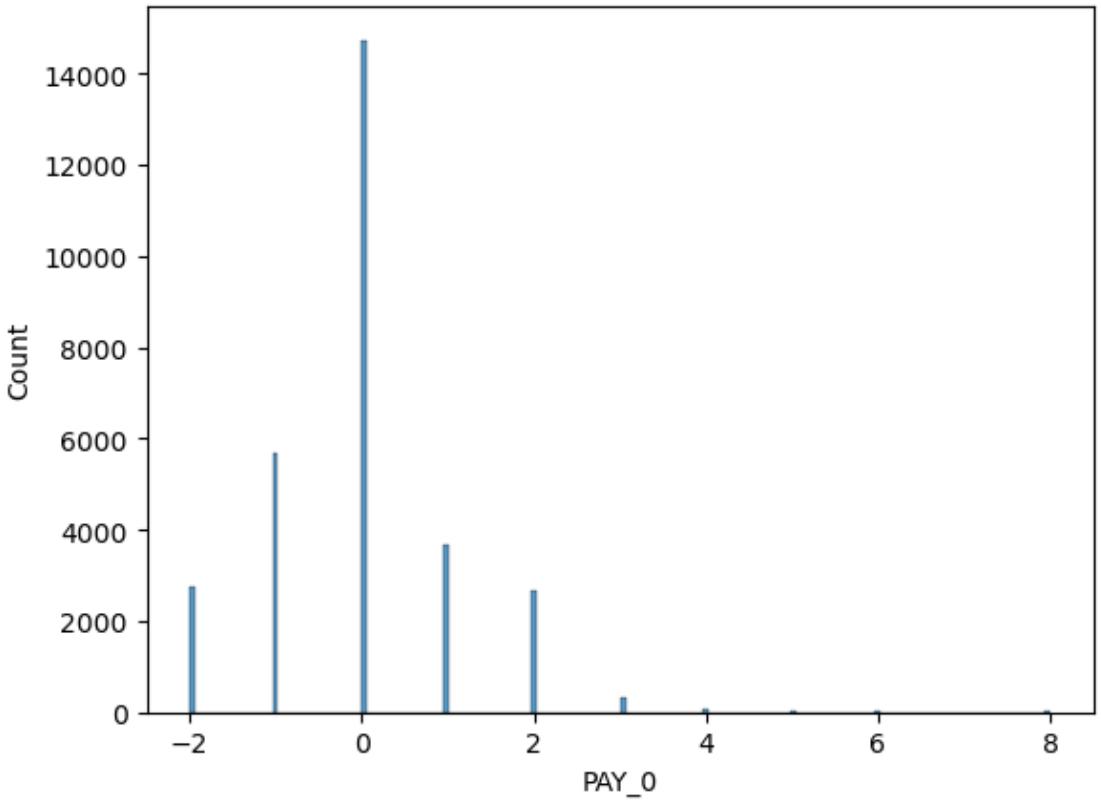
[9]:
```
#number of non defaulter is more than defaulter(1=defaulter,0=non defaulter)
sns.histplot(df['default.payment.next.month'])
```

[9]: <AxesSubplot:xlabel='default.payment.next.month', ylabel='Count'>



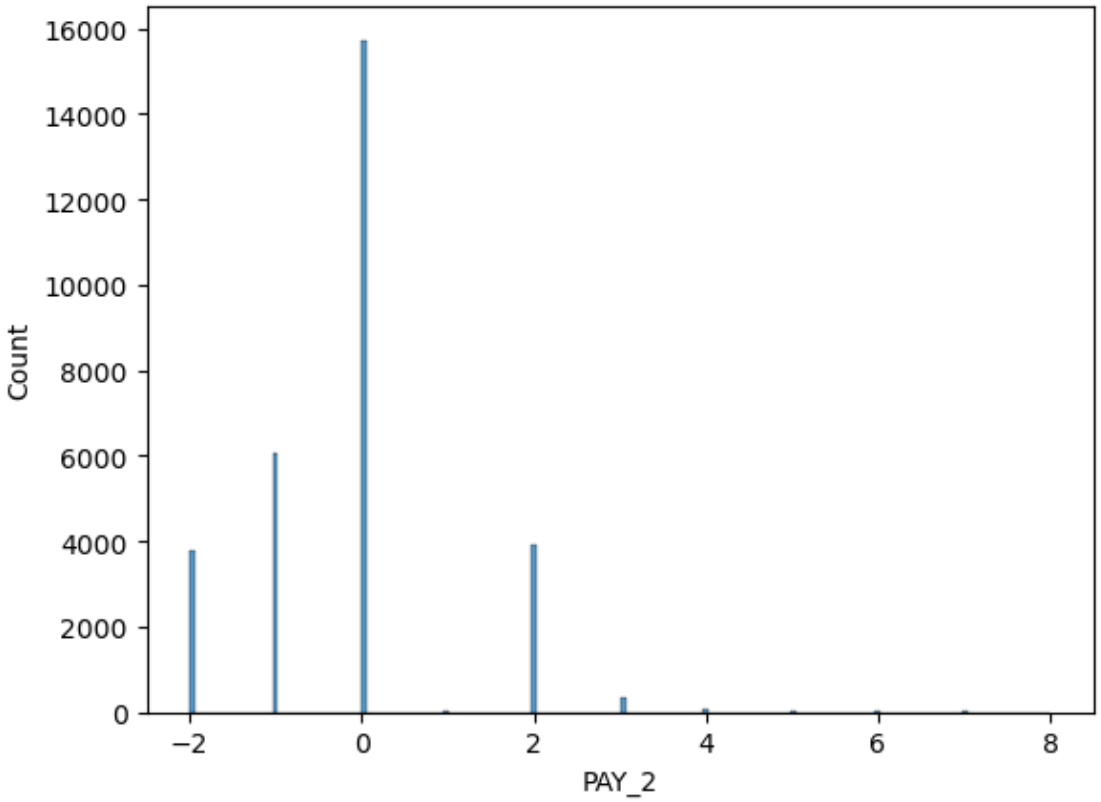
[10]:
```
#SEPTEMBER 2005
#number of people who pay credit on time are more
#on time > pay 1 monthe earlier > 1 month delayed > 2 month earlier > 2 month␣
 ↪delayed
sns.histplot(df['PAY_0'])
```

[10]: <AxesSubplot:xlabel='PAY_0', ylabel='Count'>



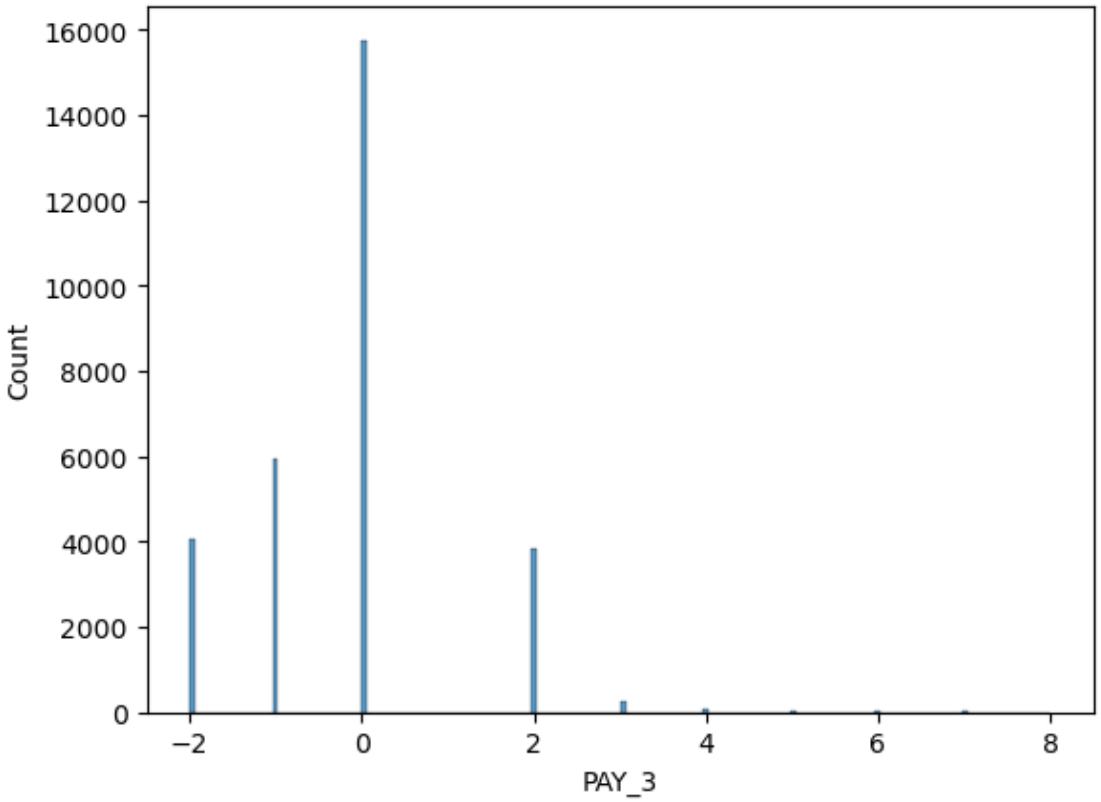
[11]:
```
#AUGUST 2005
#on time > pay 1 monthe earlier > 1 month delayed > 2 month delayed > 2 month␣
 ↪earlier
sns.histplot(df['PAY_2'])
```

[11]: <AxesSubplot:xlabel='PAY_2', ylabel='Count'>



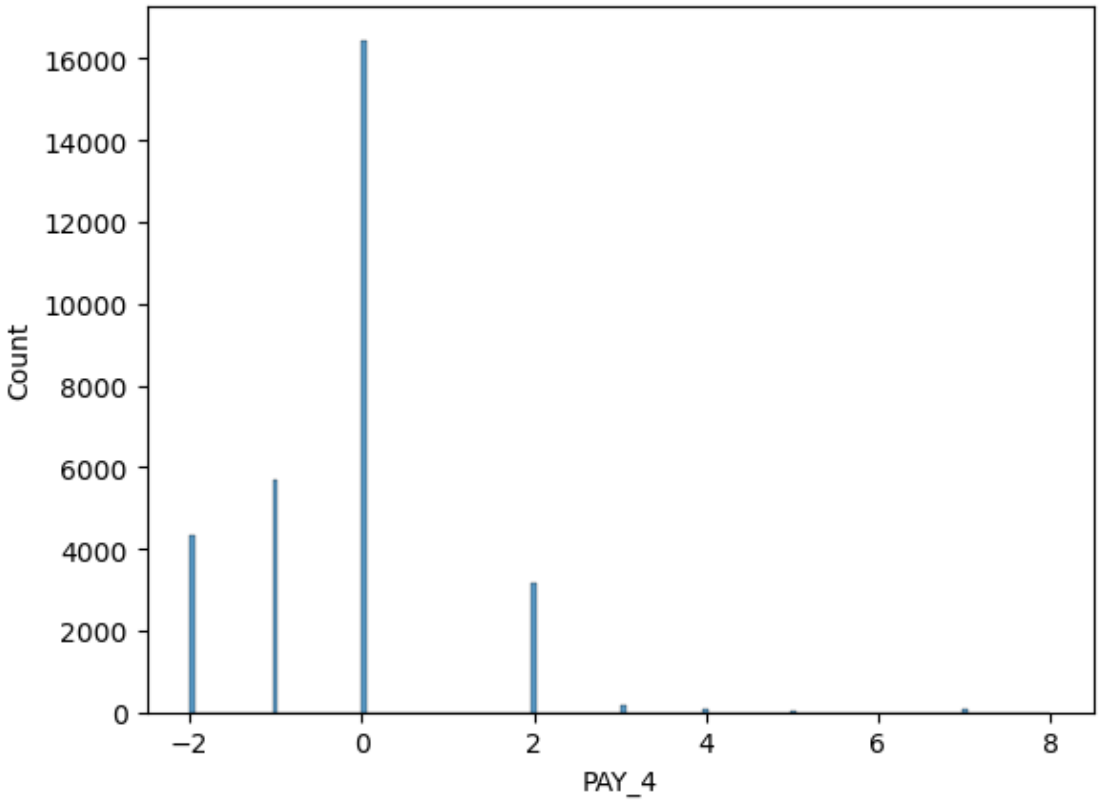
[12]:
```
sns.histplot(df['PAY_3'])
```

[12]: <AxesSubplot:xlabel='PAY_3', ylabel='Count'>



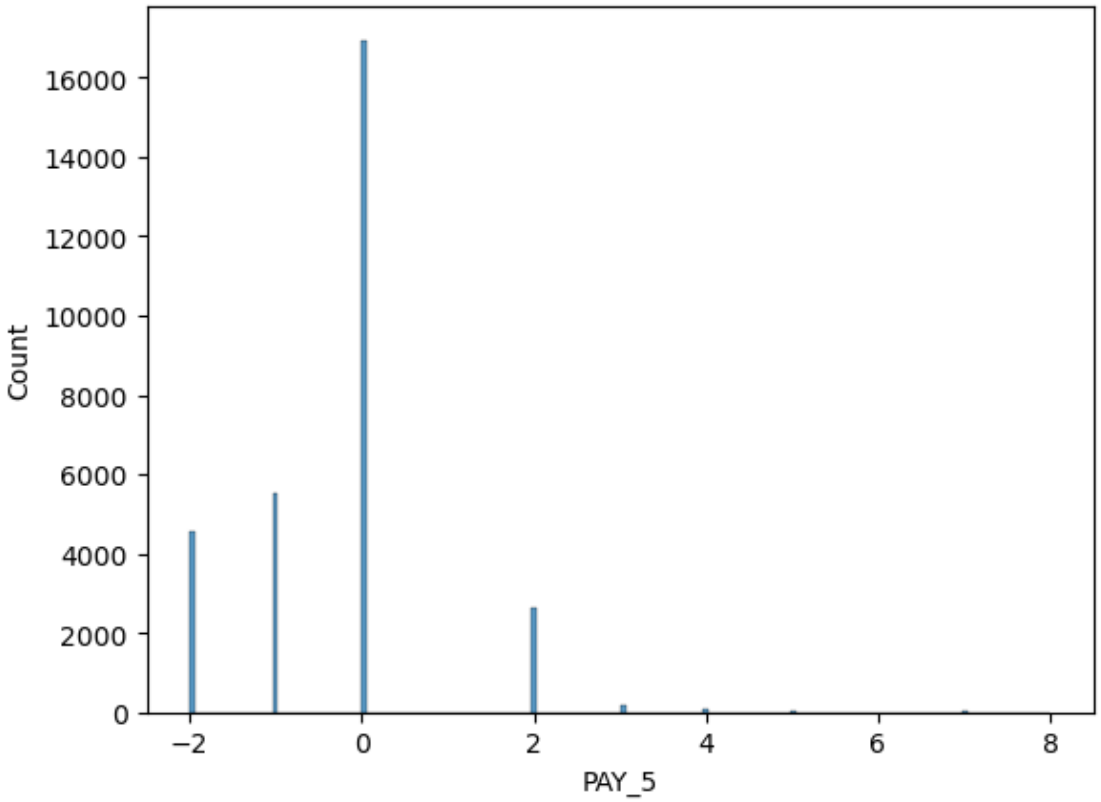
[13]:
```
sns.histplot(df['PAY_4'])
```

[13]:
```
<AxesSubplot:xlabel='PAY_4', ylabel='Count'>
```



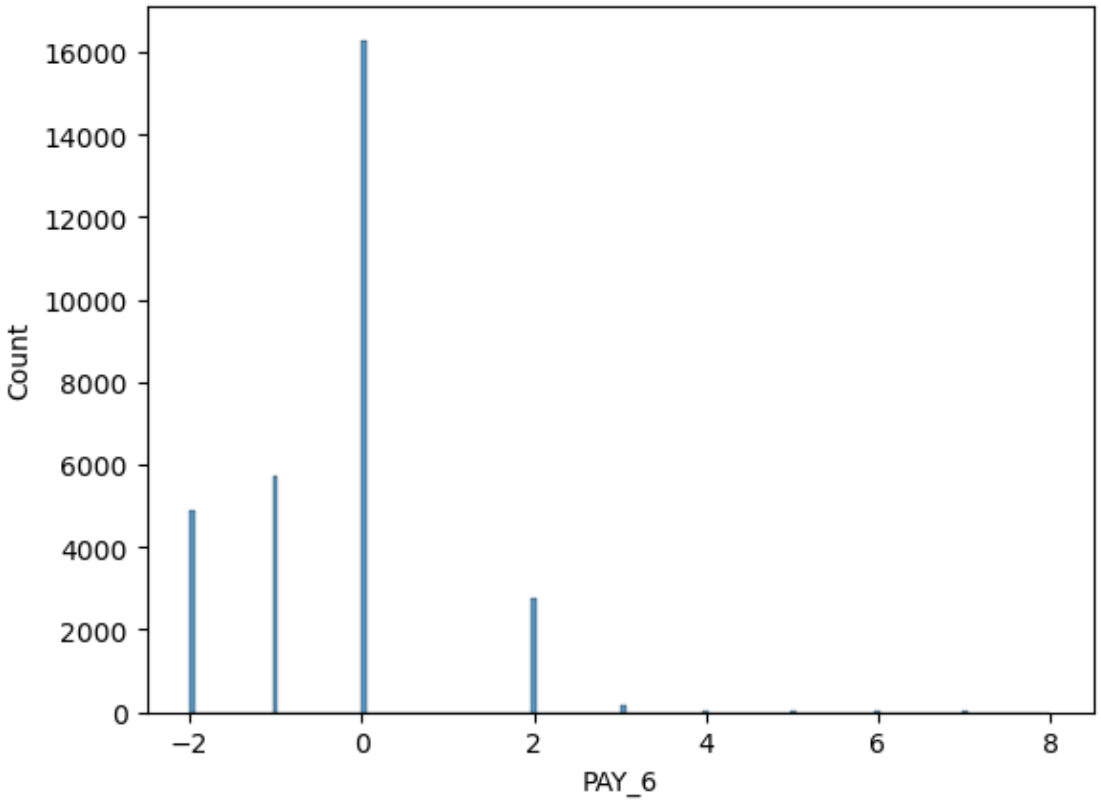
[14]:
```
sns.histplot(df['PAY_5'])
```

[14]: <AxesSubplot:xlabel='PAY_5', ylabel='Count'>



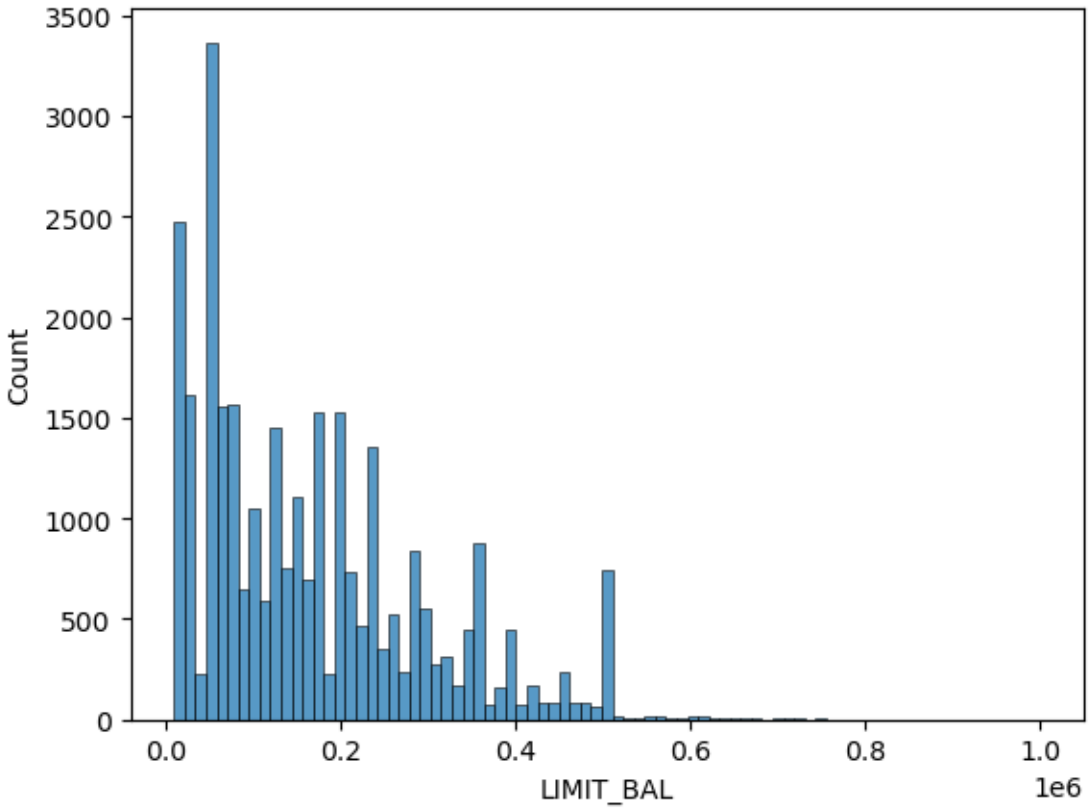
[15]:
```
sns.histplot(df['PAY_6'])
```

[15]: <AxesSubplot:xlabel='PAY_6', ylabel='Count'>



[16]:
```
sns.histplot(df['LIMIT_BAL'])
```

[16]: <AxesSubplot:xlabel='LIMIT_BAL', ylabel='Count'>



[17]:
```
df2=df.drop('ID',axis=1)
df2
```

[17]:

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	\
0	20000.0	2	2	1	24	2	2	-1	-1	
1	120000.0	2	2	2	26	-1	2	0	0	
2	90000.0	2	2	2	34	0	0	0	0	
3	50000.0	2	2	1	37	0	0	0	0	
4	50000.0	1	2	1	57	-1	0	-1	0	
…	…	…	…	…	…	…	…	…	…	
29995	220000.0	1	3	1	39	0	0	0	0	
29996	150000.0	1	3	2	43	-1	-1	-1	-1	
29997	30000.0	1	2	2	37	4	3	2	-1	
29998	80000.0	1	3	1	41	1	-1	0	0	
29999	50000.0	1	2	1	46	0	0	0	0	

	PAY_5	…	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	\
0	-2	…	0.0	0.0	0.0	0.0	689.0	
1	0	…	3272.0	3455.0	3261.0	0.0	1000.0	
2	0	…	14331.0	14948.0	15549.0	1518.0	1500.0	
3	0	…	28314.0	28959.0	29547.0	2000.0	2019.0	

```
4          0  …    20940.0    19146.0    19131.0    2000.0   36681.0
…      …   …        …          …          …          …
29995      0  …    88004.0    31237.0    15980.0    8500.0   20000.0
29996      0  …     8979.0     5190.0        0.0    1837.0    3526.0
29997      0  …    20878.0    20582.0    19357.0       0.0       0.0
29998      0  …    52774.0    11855.0    48944.0   85900.0    3409.0
29999      0  …    36535.0    32428.0    15313.0    2078.0    1800.0

       PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6  default.payment.next.month
0           0.0       0.0       0.0       0.0                           1
1        1000.0    1000.0       0.0    2000.0                           1
2        1000.0    1000.0    1000.0    5000.0                           0
3        1200.0    1100.0    1069.0    1000.0                           0
4       10000.0    9000.0     689.0     679.0                           0
…           …         …         …         …                           …
29995    5003.0    3047.0    5000.0    1000.0                           0
29996    8998.0     129.0       0.0       0.0                           0
29997   22000.0    4200.0    2000.0    3100.0                           1
29998    1178.0    1926.0   52964.0    1804.0                           1
29999    1430.0    1000.0    1000.0    1000.0                           1

[30000 rows x 24 columns]
```

[18]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 24 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   LIMIT_BAL                   30000 non-null  float64
 1   SEX                         30000 non-null  int64
 2   EDUCATION                   30000 non-null  int64
 3   MARRIAGE                    30000 non-null  int64
 4   AGE                         30000 non-null  int64
 5   PAY_0                       30000 non-null  int64
 6   PAY_2                       30000 non-null  int64
 7   PAY_3                       30000 non-null  int64
 8   PAY_4                       30000 non-null  int64
 9   PAY_5                       30000 non-null  int64
 10  PAY_6                       30000 non-null  int64
 11  BILL_AMT1                   30000 non-null  float64
 12  BILL_AMT2                   30000 non-null  float64
 13  BILL_AMT3                   30000 non-null  float64
 14  BILL_AMT4                   30000 non-null  float64
 15  BILL_AMT5                   30000 non-null  float64
 16  BILL_AMT6                   30000 non-null  float64
```

```
 17  PAY_AMT1                    30000 non-null  float64
 18  PAY_AMT2                    30000 non-null  float64
 19  PAY_AMT3                    30000 non-null  float64
 20  PAY_AMT4                    30000 non-null  float64
 21  PAY_AMT5                    30000 non-null  float64
 22  PAY_AMT6                    30000 non-null  float64
 23  default.payment.next.month  30000 non-null  int64
dtypes: float64(13), int64(11)
memory usage: 5.5 MB
```

[19]:
```
X=df2.drop('default.payment.next.month',axis=1)
y=df2['default.payment.next.month']
```

[20]:
```
X.shape
```

[20]: (30000, 23)

[21]:
```
y.shape
```

[21]: (30000,)

[22]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.
 ↪2,random_state=51)
```

[23]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(24000, 23)
(6000, 23)
(24000,)
(6000,)
```

[24]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
lr.score(X_test,y_test)
```

[24]: 0.13106083439927785

[25]:
```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
dt.fit(X_train,y_train)
dt.score(X_test,y_test)
```

[25]: 0.7283333333333334

[26]:
```
from sklearn.ensemble import RandomForestClassifier
rff=RandomForestClassifier()
rff.fit(X_train,y_train)
rff.score(X_test,y_test)
```

[26]: 0.8221666666666667

[27]:
```
rff1=RandomForestClassifier(n_estimators=200)
rff1.fit(X_train,y_train)
rff1.score(X_test,y_test)
```

[27]: 0.8226666666666667

[28]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(X_train,y_train)
knn.score(X_test,y_test)
```

```
E:\anaconda new\lib\site-packages\sklearn\neighbors\_classification.py:228:
FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the
default behavior of `mode` typically preserves the axis it acts along. In SciPy
1.11.0, this behavior will change: the default value of `keepdims` will become
False, the `axis` over which the statistic is taken will be eliminated, and the
value None will no longer be accepted. Set `keepdims` to True or False to avoid
this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

[28]: 0.7555

[29]:
```
from sklearn.svm import SVC
svc=SVC()
svc.fit(X_train,y_train)
svc.score(X_test,y_test)
```

[29]: 0.7815

[35]:
```
from sklearn.naive_bayes import GaussianNB
gauss=GaussianNB()
gauss.fit(X_train,y_train)
gauss.score(X_test,y_test)
```

[35]: 0.37083333333333335

[36]:
```
y_pred=rff1.score(X_test,y_test)
y_pred
```

[36]: 0.8226666666666667

[37]:
```
predict=rff1.predict(X_test)
predict
```

[37]:
```
array([0, 0, 0, …, 0, 0, 0], dtype=int64)
```

[39]:
```
new=pd.DataFrame(predict,y_test,columns=['predicted'])
new
```

[39]:

default.payment.next.month	predicted
1	0
0	0
0	0
0	0
0	0
…	…
1	1
0	0
0	0
0	0
0	0

[6000 rows x 1 columns]

[41]:
```
new.shape
```

[41]:
```
(6000, 1)
```

Random forest classifier gives 82 % accuracy

[ ]:
```

```