# bo2va4dui

August 13, 2023

## 1 Credit Card Default Clients Prediction

[80]:

```
#Import the required Librarys
import pandas as pd
import numpy as np
from numpy import random
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,␣
  ↪classification_report,confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
import warnings
warnings.filterwarnings("ignore")

pd.set_option("display.max_columns",200)
df = pd.read_csv("UCI_Credit_Card.csv",delimiter=",")
col1,col2 = df.shape
print("In this dataset {} rows and {} columns".format(df.shape[0],df.shape[1]))
```

```
In this dataset 30000 rows and 25 columns
```

[81]:

```
print(df.shape)
#take some random samples on the data
np.random.seed(1)
df.sample(n=10)
```

```
(30000, 25)
```

[81]:

```
          ID  LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_0  PAY_2  PAY_3  \
10747  10748   310000.0    1          3         1   32      0      0      0
12573  12574    10000.0    2          3         1   49     -1     -1     -2
```

```
29676  29677    50000.0    1          2         1   28     -1     -1     -1
8856    8857    80000.0    2          3         1   52      2      2      3
21098  21099   270000.0    1          1         2   34      1      2      0
17458  17459   140000.0    2          3         1   30      0      0      0
1476    1477   200000.0    1          2         2   26     -1     -1      0
5120    5121   150000.0    1          2         2   37     -1      0      0
18338  18339    20000.0    2          2         2   22      0      0      0
28279  28280   230000.0    2          2         2   36     -2     -1     -1

       PAY_4  PAY_5  PAY_6  BILL_AMT1  BILL_AMT2  BILL_AMT3  BILL_AMT4  \
10747      0      0      0   172772.0   152397.0   110375.0    84373.0
12573     -1      2      2       32.0     -358.0     -748.0     1690.0
29676      0     -1     -1      430.0        0.0    46257.0    45975.0
8856       3      3      2    36649.0    39448.0    40101.0    40748.0
21098      0      2      0    20979.0    17228.0    20924.0    22448.0
17458      0      2      0    93157.0    96304.0    98007.0    82227.0
1476       0      0      0     1747.0    11817.0    14225.0    16017.0
5120       0      0      0    69012.0    63265.0    64131.0    64942.0
18338      0      0      0    16990.0    17960.0    18923.0    19706.0
28279     -1     -1     -1      858.0      885.0      669.0      656.0

       BILL_AMT5  BILL_AMT6  PAY_AMT1  PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5  \
10747    57779.0    14163.0    8295.0    6000.0    4000.0    3000.0    1000.0
12573     1138.0      930.0       0.0       0.0    2828.0       0.0     182.0
29676     1300.0    43987.0       0.0   46257.0    2200.0    1300.0   43987.0
8856     39816.0    40607.0    3700.0    1600.0    1600.0       0.0    1600.0
21098    15490.0    17343.0       0.0    4000.0    2000.0       0.0    2000.0
17458    65000.0    60848.0    4700.0    4505.0   12906.0       0.0    2210.0
1476     12613.0     6600.0   12957.0    3884.0    5010.0     700.0     360.0
5120     61803.0    58987.0    2500.0    2500.0    3780.0    2200.0    2000.0
18338    19818.0    20006.0    3000.0    3001.0    3000.0    2000.0    1002.0
28279      827.0     2360.0     885.0     669.0     656.0     827.0    2376.0

       PAY_AMT6  default.payment.next.month
10747    2000.0                           0
12573       0.0                           1
29676    1386.0                           0
8856     1600.0                           1
21098    2000.0                           0
17458    2300.0                           0
1476     1713.0                           0
5120     2000.0                           0
18338     783.0                           0
28279     943.0                           0
```

[82]:
```
df.isnull().sum()
##missing values for surity
```

[82]:

```
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
```

[83]:

```
df.describe().T
```

[83]:

```
                              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
```

[84]:

```
'''change the column names
     in pay_0 and default.payment.next.month
     '''
df.rename(columns={"default.payment.next.month":"def_pay"},
          inplace=True)
df.rename(columns={"PAY_0":"PAY_1"},
          inplace=True)
#checking the datatypes of each feature
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  float64
 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_1      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  float64
 13  BILL_AMT2  30000 non-null  float64
 14  BILL_AMT3  30000 non-null  float64
 15  BILL_AMT4  30000 non-null  float64
 16  BILL_AMT5  30000 non-null  float64
 17  BILL_AMT6  30000 non-null  float64
 18  PAY_AMT1   30000 non-null  float64
 19  PAY_AMT2   30000 non-null  float64
 20  PAY_AMT3   30000 non-null  float64
 21  PAY_AMT4   30000 non-null  float64
 22  PAY_AMT5   30000 non-null  float64
 23  PAY_AMT6   30000 non-null  float64
 24  def_pay    30000 non-null  int64
dtypes: float64(13), int64(12)
memory usage: 5.7 MB
```

[85]:

```python
#Feature Transformation
bins = [20,30,40,50,60,70,80]
labels=["21-30","31-40","41-50",
        "51-60","61-70","71-80"]
df["AGE_BIN"] = pd.cut(x=df.AGE,bins=bins,
                       labels=labels,right=True)
df.head()
```

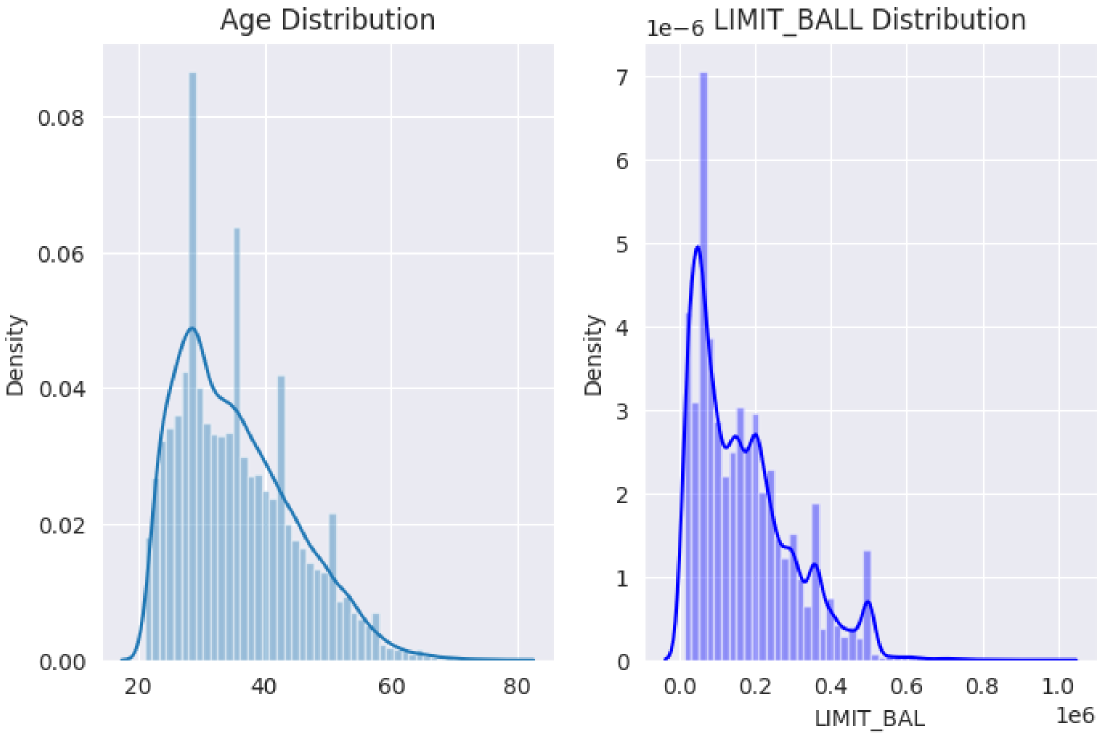
[85]:

```
   ID  LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_1  PAY_2  PAY_3  PAY_4  \
0   1    20000.0    2          2         1   24      2      2     -1     -1
1   2   120000.0    2          2         2   26     -1      2      0      0
2   3    90000.0    2          2         2   34      0      0      0      0
3   4    50000.0    2          2         1   37      0      0      0      0
4   5    50000.0    1          2         1   57     -1      0     -1      0
```

```
   PAY_5  PAY_6  BILL_AMT1  BILL_AMT2  BILL_AMT3  BILL_AMT4  BILL_AMT5  \
0     -2     -2     3913.0     3102.0      689.0        0.0        0.0
1      0      2     2682.0     1725.0     2682.0     3272.0     3455.0
2      0      0    29239.0    14027.0    13559.0    14331.0    14948.0
3      0      0    46990.0    48233.0    49291.0    28314.0    28959.0
4      0      0     8617.0     5670.0    35835.0    20940.0    19146.0

   BILL_AMT6  PAY_AMT1  PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6  \
0        0.0       0.0     689.0       0.0       0.0       0.0       0.0
1     3261.0       0.0    1000.0    1000.0    1000.0       0.0    2000.0
2    15549.0    1518.0    1500.0    1000.0    1000.0    1000.0    5000.0
3    29547.0    2000.0    2019.0    1200.0    1100.0    1069.0    1000.0
4    19131.0    2000.0   36681.0   10000.0    9000.0     689.0     679.0

   def_pay AGE_BIN
0        1   21-30
1        1   21-30
2        0   31-40
3        0   31-40
4        0   51-60
```

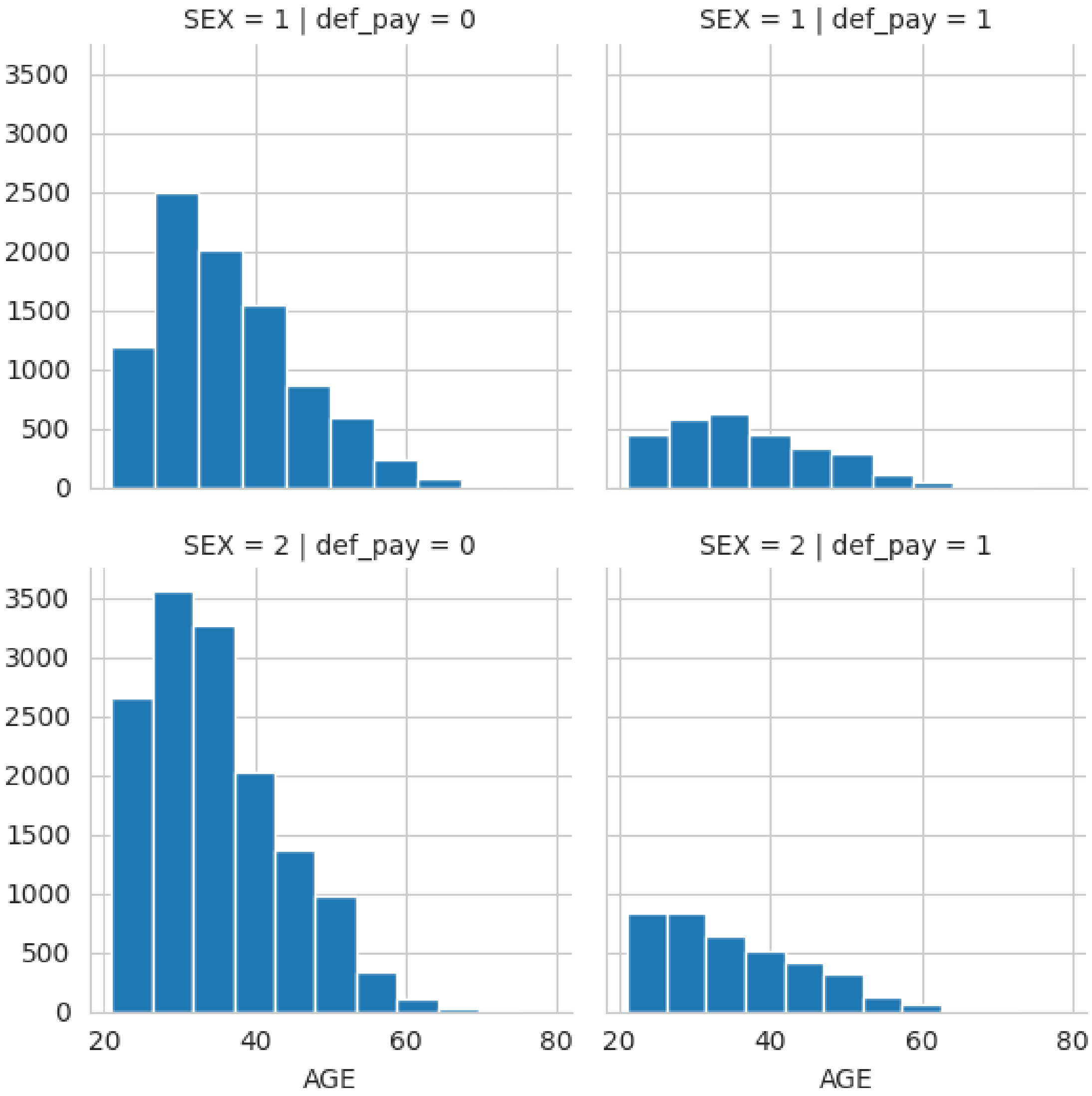
[86]:

```
'''Distribution of LIMIT_BAL
      and AGE fratures
      '''
plt.figure(figsize=(8,5))
plt.subplot(1,2,1)
plt.title("Age Distribution")
sns.distplot(x=df["AGE"])
plt.subplot(1,2,2)
sns.set_style("whitegrid")
sns.distplot(df["LIMIT_BAL"],
             kde=True,
            color="blue")
plt.title("LIMIT_BALL Distribution")
plt.show()
```



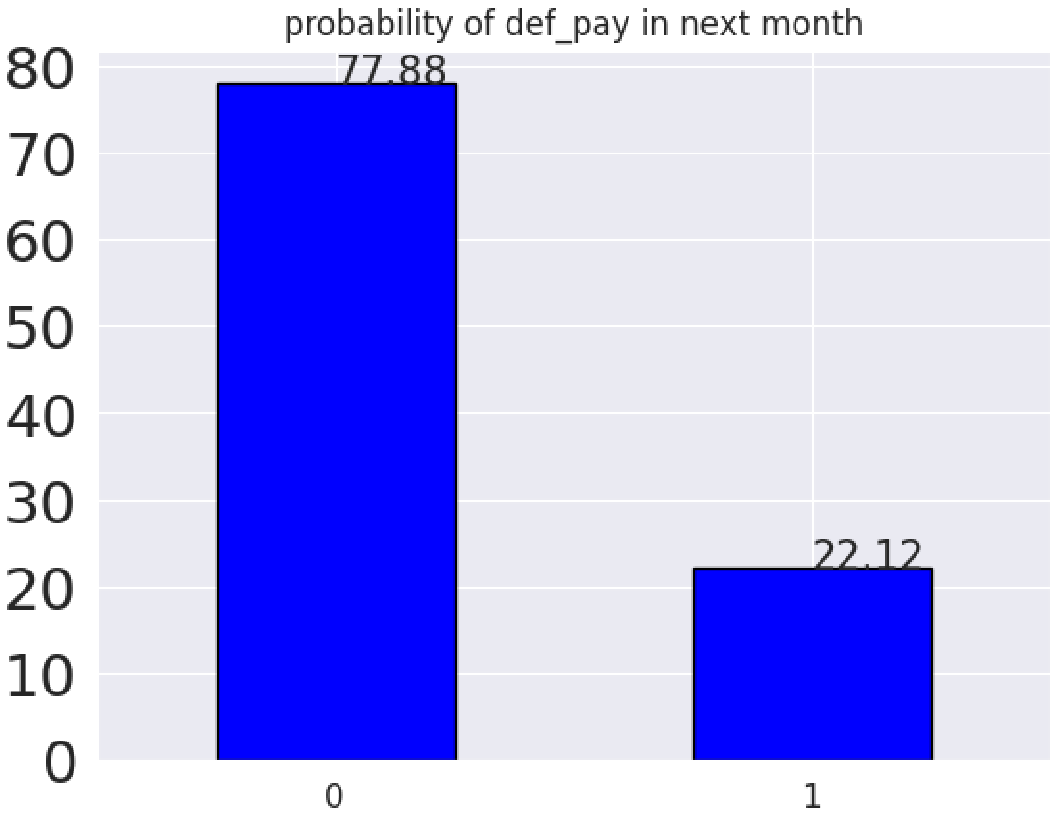
[87]:

```
'''Default paymenats
     on Status on sex column'''
g = sns.FacetGrid(df,col="def_pay",
                  row="SEX")
g.map(plt.hist,"AGE")
plt.show()
```



[88]:
```
'''
 probability of next month def_pay
 '''
sns.set_style("darkgrid")
def_cnt = (df.def_pay.value_counts(normalize=True)*100).round(2)
def_cnt.plot.bar(color="blue",
                edgecolor="black")
plt.xticks(fontsize=12,
           rotation=0)
plt.yticks(fontsize=22)
plt.title("probability of def_pay in next month")
for i,j in list(zip([0,1],
                    def_cnt)):
```

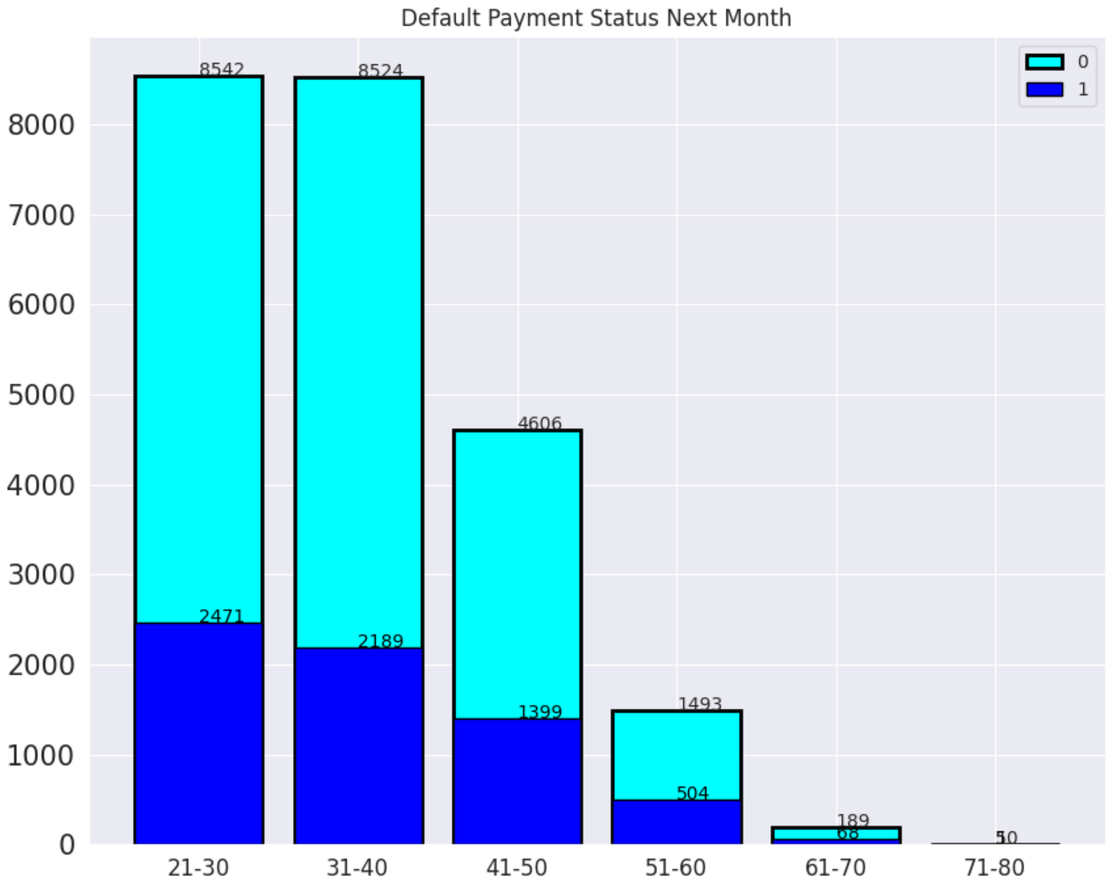
```
    plt.text(i,j,j,
             fontsize=15)
plt.show()
```



[89]:

```
'''defaut payment status on AGE_BINS
'''
plt.figure(figsize=(10,8))
age_0 = (df.AGE_BIN[df.def_pay==0].value_counts())
age_1 = (df.AGE_BIN[df.def_pay==1].value_counts())
plt.bar(age_0.index,
        age_0.values,
        label="0",
       edgecolor="black",
       lw=2,
       color="cyan")
plt.bar(age_1.index,
        age_1.values,
        label="1",
       edgecolor="black",
       color="blue")
```

```
plt.xticks(fontsize=12)
plt.yticks(fontsize=15)
for x,y in list(zip(age_0.index,
                    age_0.values)):
    plt.text(x,y,y)
for i,j in list(zip(age_1.index,
                    age_1.values)):
    plt.text(i,j,j,
            color="black",
            fontweight=1)
plt.title("Default Payment Status Next Month")
plt.legend()
plt.show()
```

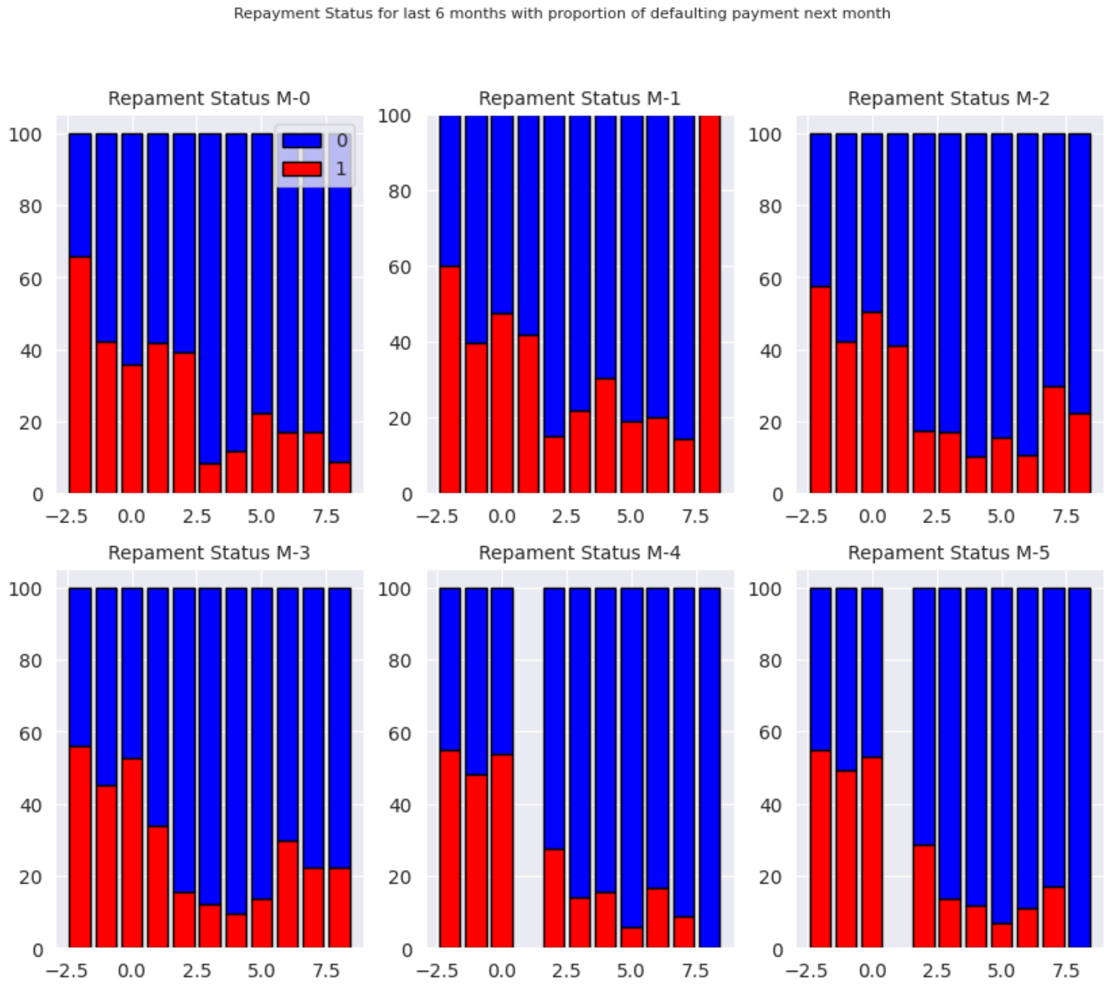


[90]:

```
'''repayment status for last six months
   praportion of defaut payment on
   next month
'''
plt.subplots(figsize=(10,8))
```

```
#month0
ind = sorted(df.PAY_1.unique())
pay_0 = df.PAY_1[df.def_pay==0].value_counts(normalize=True)
pay_1 = df.PAY_1[df.def_pay==1].value_counts(normalize=True)


total = pay_0.values+pay_1.values

pay_prop_0 = np.true_divide(pay_0,total)*100
pay_prop_1 = np.true_divide(pay_1,total)*100

plt.subplot(2,3,1)

plt.bar(ind,pay_prop_1,
        label="0",
        bottom=pay_prop_0,
       color="blue",
       edgecolor="black")
plt.bar(ind,pay_prop_0,
        label="1",
       color="red",
       edgecolor="black")
plt.title("Repament Status M-0",fontsize=10)
plt.legend()

#month1
ind2 = sorted(df.PAY_2.unique())
pay_2_0 = df.PAY_2[df.def_pay==0].value_counts(normalize=True)
pay_2_1 = df.PAY_2[df.def_pay==1].value_counts(normalize=True)

for i in pay_2_0.index:
    if i not in pay_2_1.index:
        pay_2_1[i]=0

total_2 = pay_2_0.values+pay_2_1.values

pay_2_prop_0 = np.true_divide(pay_2_0,total_2)*100
pay_2_prop_1 = np.true_divide(pay_2_1,total_2)*100

plt.subplot(2,3,2)

plt.bar(ind2,pay_2_prop_1,
        label="0",
        bottom=pay_2_prop_0,
       color="blue",
       edgecolor="black")
plt.bar(ind2,pay_2_prop_0,
        label="1",
       color="red",
       edgecolor="black")
plt.title("Repament Status M-1",fontsize=10)

#month2
ind3 = sorted(df.PAY_3.unique())
pay_3_0 = df.PAY_3[df.def_pay==0].value_counts(normalize=True)
pay_3_1 = df.PAY_3[df.def_pay==1].value_counts(normalize=True)

for i in pay_3_0.index:
    if i not in pay_3_1.index:
        pay_3_1[i]=0

total_3 = pay_3_0.values+pay_3_1.values

pay_3_prop_0 = np.true_divide(pay_3_0,total_3)*100
pay_3_prop_1 = np.true_divide(pay_3_1,total_3)*100

plt.subplot(2,3,3)

plt.bar(ind3,pay_3_prop_1,
        label="0",
        bottom=pay_3_prop_0,
       color="blue",
       edgecolor="black")
plt.bar(ind3,pay_3_prop_0,label="1",
       color="red",
       edgecolor="black")
plt.title("Repament Status M-2",fontsize=10)

#month3
ind4 = sorted(df.PAY_4.unique())
pay_4_0 = df.PAY_4[df.def_pay==0].value_counts(normalize=True)
pay_4_1 = df.PAY_4[df.def_pay==1].value_counts(normalize=True)
for i in pay_4_0.index:
    if i not in pay_4_1.index:
        pay_4_1[i]=0

total_4 = pay_4_0.values+pay_4_1.values
pay_4_prop_0 = np.true_divide(pay_4_0,total_4)*100
pay_4_prop_1 = np.true_divide(pay_4_1,total_4)*100
plt.subplot(2,3,4)
plt.bar(ind4,pay_4_prop_1,
        label="0",
        bottom=pay_4_prop_0,
       color="blue",
       edgecolor="black")
plt.bar(ind4,pay_4_prop_0,
        label="1",
       color="red",
       edgecolor="black")
plt.title("Repament Status M-3",fontsize=10)

#month4
ind5 = sorted(df.PAY_5.unique())
pay_5_0 = df.PAY_5[df.def_pay==0].value_counts(normalize=True)
pay_5_1 = df.PAY_5[df.def_pay==1].value_counts(normalize=True)

for i in pay_5_0.index:
    if i not in pay_5_1.index:
        pay_5_1[i]=0

for i in pay_5_1.index:
    if i not in pay_5_0.index:
        pay_5_0[i]=0


total_5 = pay_5_0.values+pay_5_1.values

pay_5_prop_0 = np.true_divide(pay_5_0,total_5)*100
pay_5_prop_1 = np.true_divide(pay_5_1,total_5)*100

plt.subplot(2,3,5)

plt.bar(ind5,pay_5_prop_1,
        label="0",
        bottom=pay_5_prop_0,
       color = "blue",
       edgecolor="black")
plt.bar(ind5,pay_5_prop_0,
        label="1",
       color="red",
       edgecolor="black")
plt.title("Repament Status M-4",fontsize=10)

#month5
ind6 = sorted(df.PAY_6.unique())
pay_6_0 = df.PAY_6[df.def_pay==0].value_counts(normalize=True)
pay_6_1 = df.PAY_6[df.def_pay==1].value_counts(normalize=True)

for i in pay_6_0.index:
    if i not in pay_6_1.index:
        pay_6_1[i]=0
```

```
for i in pay_6_1.index:
    if i not in pay_6_0:
        pay_6_0[i] = 0

total_6 = pay_6_0.values+pay_6_1.values

pay_6_prop_0 = np.true_divide(pay_6_0,total_6)*100
pay_6_prop_1 = np.true_divide(pay_6_1,total_6)*100

plt.subplot(2,3,6)

plt.bar(ind6,pay_6_prop_1,label="0",
        bottom=pay_6_prop_0,
       color="blue",
       edgecolor="black")
plt.bar(ind6,pay_6_prop_0,
        label="1",
       color="red",
       edgecolor="black")
plt.title("Repament Status M-5",fontsize=10)
plt.suptitle("Repayment Status for last 6 months with proportion of defaulting␣
 ↪payment next month",
             fontsize=8)

plt.show()
```



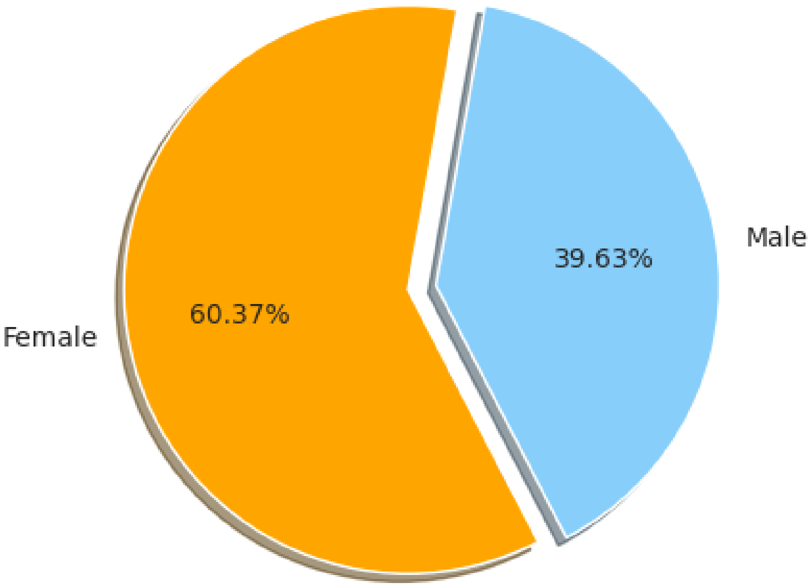
[91]:
```
df.head()
```

[91]:
```
   ID  LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_1  PAY_2  PAY_3  PAY_4  \
0   1    20000.0    2          2         1   24      2      2     -1     -1
1   2   120000.0    2          2         2   26     -1      2      0      0
2   3    90000.0    2          2         2   34      0      0      0      0
3   4    50000.0    2          2         1   37      0      0      0      0
4   5    50000.0    1          2         1   57     -1      0     -1      0

   PAY_5  PAY_6  BILL_AMT1  BILL_AMT2  BILL_AMT3  BILL_AMT4  BILL_AMT5  \
0     -2     -2     3913.0     3102.0      689.0        0.0        0.0
1      0      2     2682.0     1725.0     2682.0     3272.0     3455.0
2      0      0    29239.0    14027.0    13559.0    14331.0    14948.0
3      0      0    46990.0    48233.0    49291.0    28314.0    28959.0
4      0      0     8617.0     5670.0    35835.0    20940.0    19146.0
```

```
   BILL_AMT6  PAY_AMT1  PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6  \
0        0.0       0.0     689.0       0.0       0.0       0.0       0.0
1     3261.0       0.0    1000.0    1000.0    1000.0       0.0    2000.0
2    15549.0    1518.0    1500.0    1000.0    1000.0    1000.0    5000.0
3    29547.0    2000.0    2019.0    1200.0    1100.0    1069.0    1000.0
4    19131.0    2000.0   36681.0   10000.0    9000.0     689.0     679.0

   def_pay AGE_BIN
0        1   21-30
1        1   21-30
2        0   31-40
3        0   31-40
4        0   51-60
```

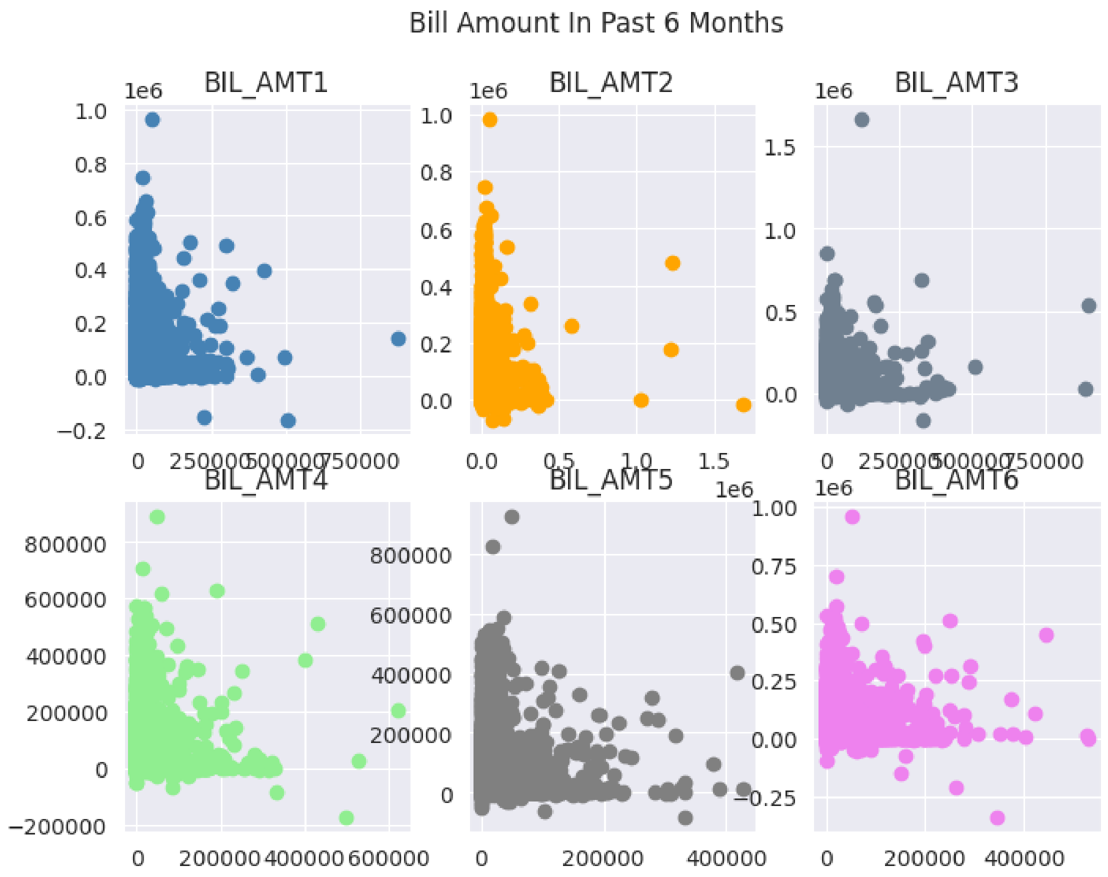
[92]:

```
'''check the Gender
count on the given data
'''
values = df.SEX.value_counts(normalize=False)
values = list(values)
lab = ["Female","Male"]
col = ["orange","lightskyblue"]
ex = [0.01,0.09]
plt.pie(values,labels=lab,
       colors=col,
       autopct="%.2f%%",
       explode=ex,
       startangle=80,
       shadow=True)
plt.show()
```



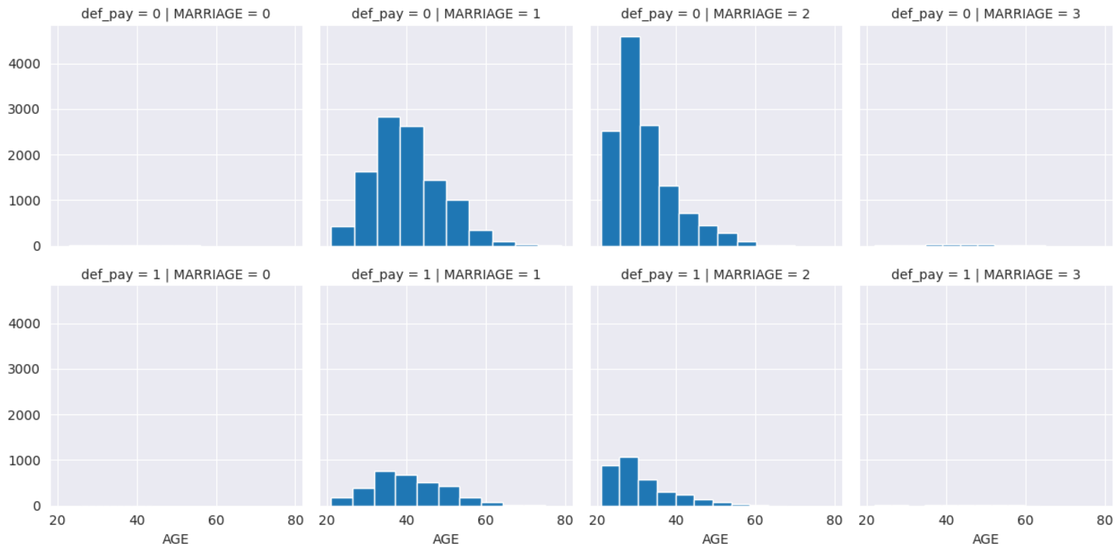
[93]:

```
#bill amount in past 6 months
plt.figure(figsize=(8,6))
plt.subplot(2,3,1)
plt.scatter(x=df["PAY_AMT1"],
            y=df["BILL_AMT1"],
            c="steelblue")
plt.title("BIL_AMT1")
plt.subplot(2,3,2)
plt.scatter(x=df["PAY_AMT2"],
            y=df["BILL_AMT2"],
            c="orange")
plt.title("BIL_AMT2")
plt.subplot(2,3,3)
plt.scatter(x=df["PAY_AMT3"],
            y=df["BILL_AMT3"],
            c="slategray")
plt.title("BIL_AMT3")
plt.subplot(2,3,4)
plt.scatter(x=df["PAY_AMT4"],
            y=df["BILL_AMT4"],
            c="lightgreen")
plt.title("BIL_AMT4")
plt.subplot(2,3,5)
plt.scatter(x=df["PAY_AMT5"],
            y=df["BILL_AMT5"],
            c="grey")
plt.title("BIL_AMT5")
plt.subplot(2,3,6)
plt.scatter(x=df["PAY_AMT6"],
            y=df["BILL_AMT6"],
            c="violet")
plt.title("BIL_AMT6")
plt.suptitle("Bill Amount In Past 6 Months")
plt.show()
```

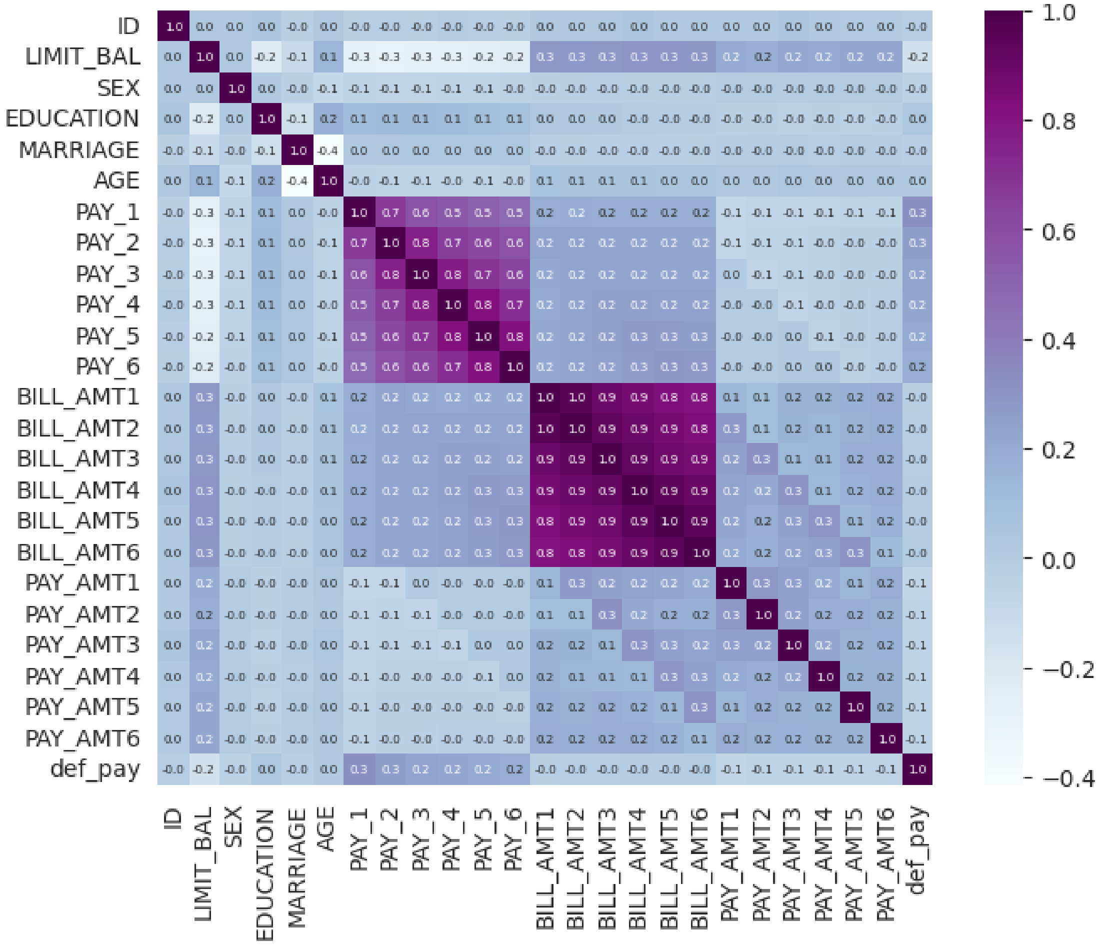


[94]:

```
#default payments on merriage
sns.set_style("darkgrid")
grid = sns.FacetGrid(df,
                     row="def_pay",
                     col="MARRIAGE")
grid.map(plt.hist,"AGE")
plt.show()
```



[95]:
```
#corr relation
corr = df.corr()
plt.figure(figsize=(8,6))
sns.heatmap(corr,annot=True,
             cbar=True,
            square=True,
             annot_kws={"size":5},
            fmt=".1f",
            cmap="BuPu")
plt.show()
```



[96]:
```
'''Model Performance on
   Logistic Regression'''

X = df.drop(["ID","AGE_BIN","AGE"],axis=1).values
y = df[["def_pay"]].values

#model initialisation
sc= StandardScaler()
'''split the data into training and testing
 20% of the data testing and remaining data training
 '''
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,
                                                 random_state=4)
X_train = sc.fit_transform(X_train)#feature scaling
X_test = sc.transform(X_test)

logrig = LogisticRegression()
```

```
logrig.fit(X_train,y_train)

pred = logrig.predict(X_test)#prediction
#chek the accuracy of the model
accuracy = accuracy_score(y_test,pred)*100
print('accuracy of logistic regression:',accuracy)
```

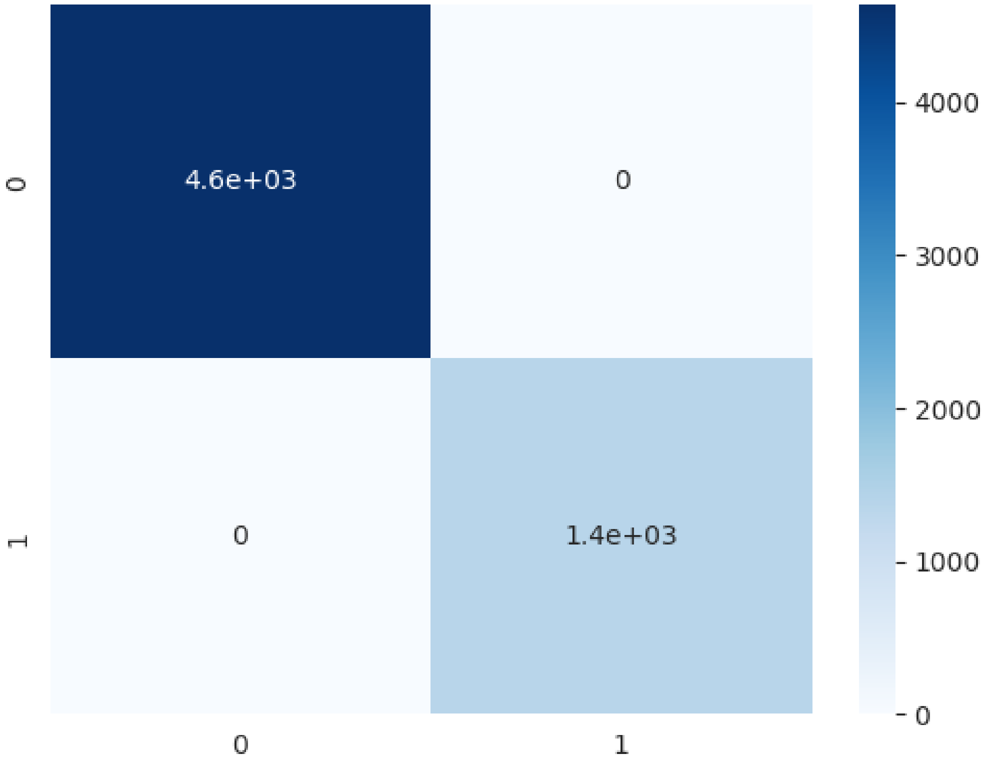
```
accuracy of logistic regression: 100.0
```

```
[97]: logrig.predict(X_test)
```

```
[97]: array([0, 0, 0, …, 0, 0, 1], dtype=int64)
```

```
[98]: '''Confusion Matrix'''
      cm = confusion_matrix(y_test,pred)
      print(cm)
      sns.heatmap(cm,annot=True,
                  cbar=True,cmap="Blues")
      plt.show()
```

```
[[4638    0]
 [   0 1362]]
```



[99]:
```
clr = classification_report(y_test,pred)
'''
  classification report Logistic Regression'''
print(clr)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4638
           1       1.00      1.00      1.00      1362

    accuracy                           1.00      6000
   macro avg       1.00      1.00      1.00      6000
weighted avg       1.00      1.00      1.00      6000
```

[100]:
```
'''Let's check each algorithm
    and
 see which one gives the best results
 '''
def process_post():
    global accuracy_
    accuracy_ = []
    list(accuracy_)
 #algorithms inirtailsation
    algo = {
        "Logistic Regression": LogisticRegression(),
        "SVM":SVC(kernel="linear"),
        "KNN":KNeighborsClassifier(algorithm="auto"),
        "Naive Bayes":GaussianNB(),
        "Random Forest Classifier":RandomForestClassifier(n_estimators=12)
        }
    algos = list(algo.values())
    for method in algos:
        method.fit(X_train,y_train)
        pred = method.predict(X_test)
        accuracy_.append(accuracy_score(y_test,pred)*100)
    return accuracy_
process_post()
#algorithm variables
algo_process = {"Algorithms":["Logistic Regression",
                     "KNeighbors Classiier",
                     "Support Vector Machine",
                     "KNearest Neighbors",
                     "Random Forest Classifier"
                    ],"Accuracy":accuracy_

    }
```

```python
#insert the values in the dataframe
df2 = pd.DataFrame(algo_process)
df2
```

[100]:

```
                Algorithms    Accuracy
0      Logistic Regression  100.000000
1      KNeighbors Classiier  100.000000
2    Support Vector Machine   99.433333
3        KNearest Neighbors  100.000000
4  Random Forest Classifier  100.000000
```

[ ]:

[ ]:

[ ]:

[ ]: