```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, chi2, f_classif

df=pd.read_csv('UCI_Credit_Card.csv')
df.head()
```

{"type":"dataframe","variable_name":"df"}

```
df.shape
```

```
(30000, 25)
```

```
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_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  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  default.payment.next.month  30000 non-null  int64
dtypes: float64(13), int64(12)
memory usage: 5.7 MB
```

```
df.isnull().sum()
```

```
ID                            0
LIMIT_BAL                     0
SEX                           0
EDUCATION                     0
MARRIAGE                      0
AGE                           0
PAY_0                         0
PAY_2                         0
PAY_3                         0
PAY_4                         0
PAY_5                         0
PAY_6                         0
BILL_AMT1                     0
BILL_AMT2                     0
BILL_AMT3                     0
BILL_AMT4                     0
BILL_AMT5                     0
BILL_AMT6                     0
PAY_AMT1                      0
PAY_AMT2                      0
PAY_AMT3                      0
PAY_AMT4                      0
PAY_AMT5                      0
PAY_AMT6                      0
default.payment.next.month    0
dtype: int64
```

```
 dfcopy=df.copy()
 print(dfcopy.shape)
```

```
(30000, 25)
```

```
dfcopy.describe().T
```

```
{"summary":"{\n  \"name\": \"dfcopy\",\n  \"rows\": 25,\n  \"fields\":
[\n    {\n      \"column\": \"count\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.0,\n        \"min\":
30000.0,\n        \"max\": 30000.0,\n        \"num_unique_values\":
1,\n        \"samples\": [\n          30000.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"mean\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 36132.076175894224,\n
\"min\": -0.2911,\n        \"max\": 167484.32266666667,\n
\"num_unique_values\": 25,\n        \"samples\": [\n          -0.1662\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"std\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 34731.21810434085,\n        \"min\": 0.41506180569093337,\n
\"max\": 129747.66156720239,\n        \"num_unique_values\": 25,\n
\"samples\": [\n          1.1968675684465735\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"min\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 84175.21384208061,\n
\"min\": -339603.0,\n        \"max\": 10000.0,\n
\"num_unique_values\": 11,\n        \"samples\": [\n          -
165580.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"25%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 9941.9817541889,\n        \"min\": -1.0,\n        \"max\":
50000.0,\n        \"num_unique_values\": 18,\n        \"samples\": [\n
7500.75\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"50%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 28112.9577127701,\n         \"min\": 0.0,\n        \"max\":
140000.0,\n        \"num_unique_values\": 15,\n        \"samples\": [\
n          18104.5\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"75%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 50933.70590982703,\n        \"min\": 0.0,\n        \"max\":
240000.0,\n        \"num_unique_values\": 17,\n        \"samples\": [\
n          22500.25\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"max\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 551301.61266929,\n        \"min\": 1.0,\n        \"max\":
1684259.0,\n        \"num_unique_values\": 20,\n        \"samples\":
[\n          30000.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

```python
#Column Renaming
dfcopy.rename(columns={'default.payment.next.month':'def_pay'},
inplace=True)
dfcopy.rename(columns={'PAY_0':'PAY_1'}, inplace=True)

dfcopy.info()
```

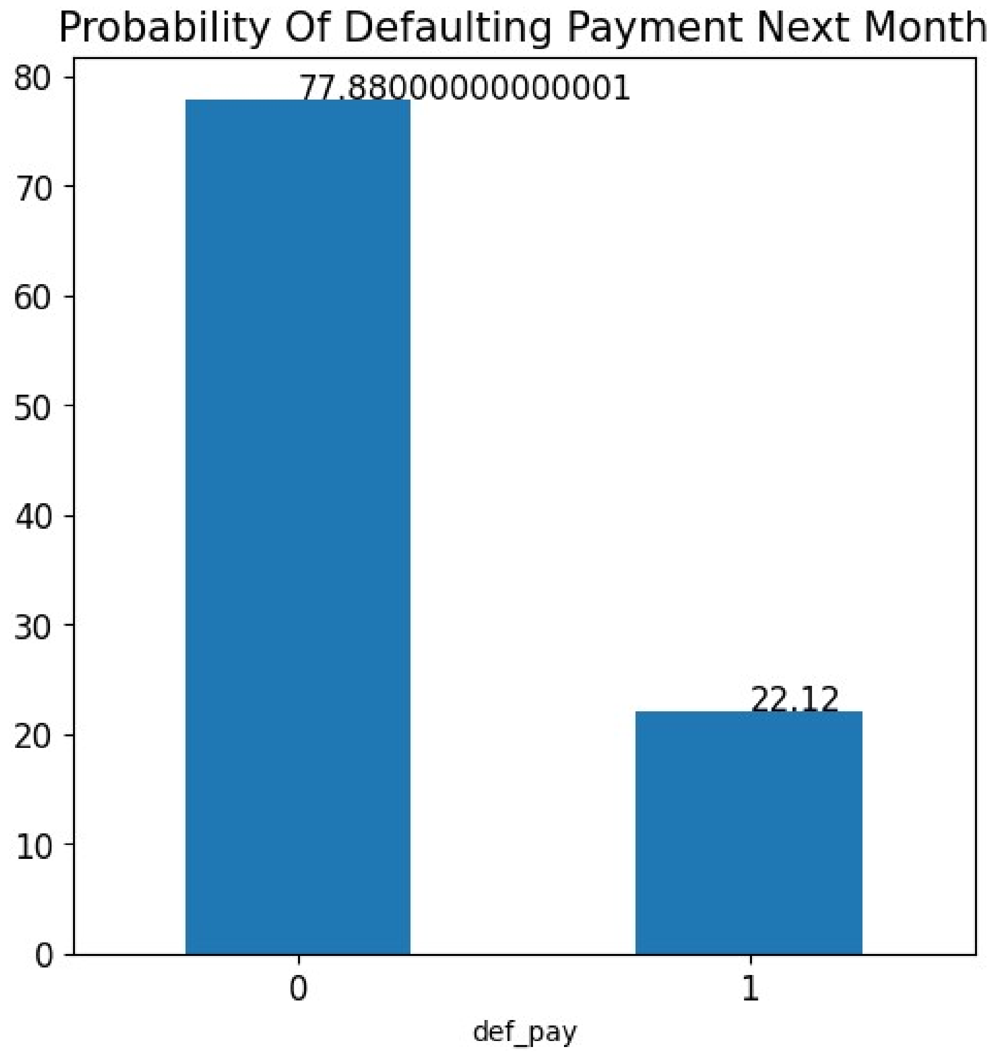
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   ID         30000 non-null  int64
 1   LIMIT_BAL  30000 non-null  float64
 2   SEX        30000 non-null  int64
 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
```

```
dfcopy.isnull().sum()
```

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

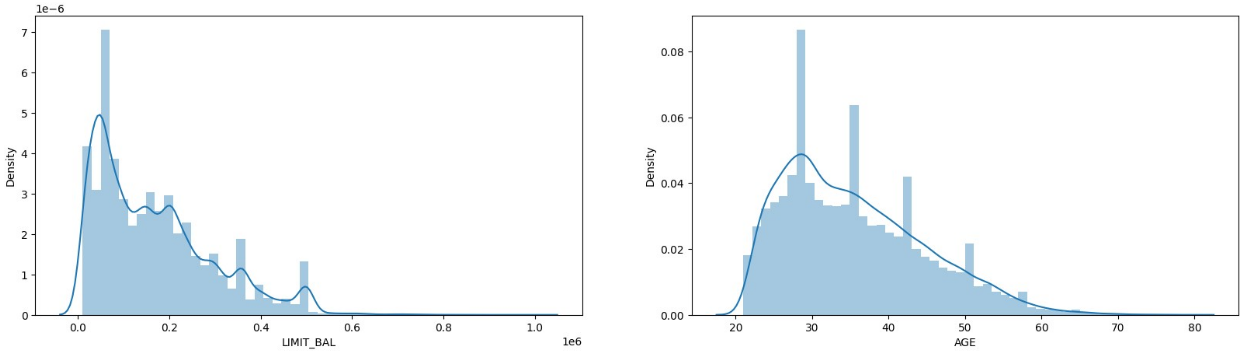
```
def_pay      0
dtype: int64
```

```
def_cnt = (dfcopy.def_pay.value_counts(normalize=True)*100)
def_cnt.plot.bar(figsize=(6,6))
plt.xticks(fontsize=12, rotation=0)
plt.yticks(fontsize=12)
plt.title("Probability Of Defaulting Payment Next Month", fontsize=15)
for x,y in zip([0,1],def_cnt):
    plt.text(x,y,y,fontsize=12)
plt.show()
```

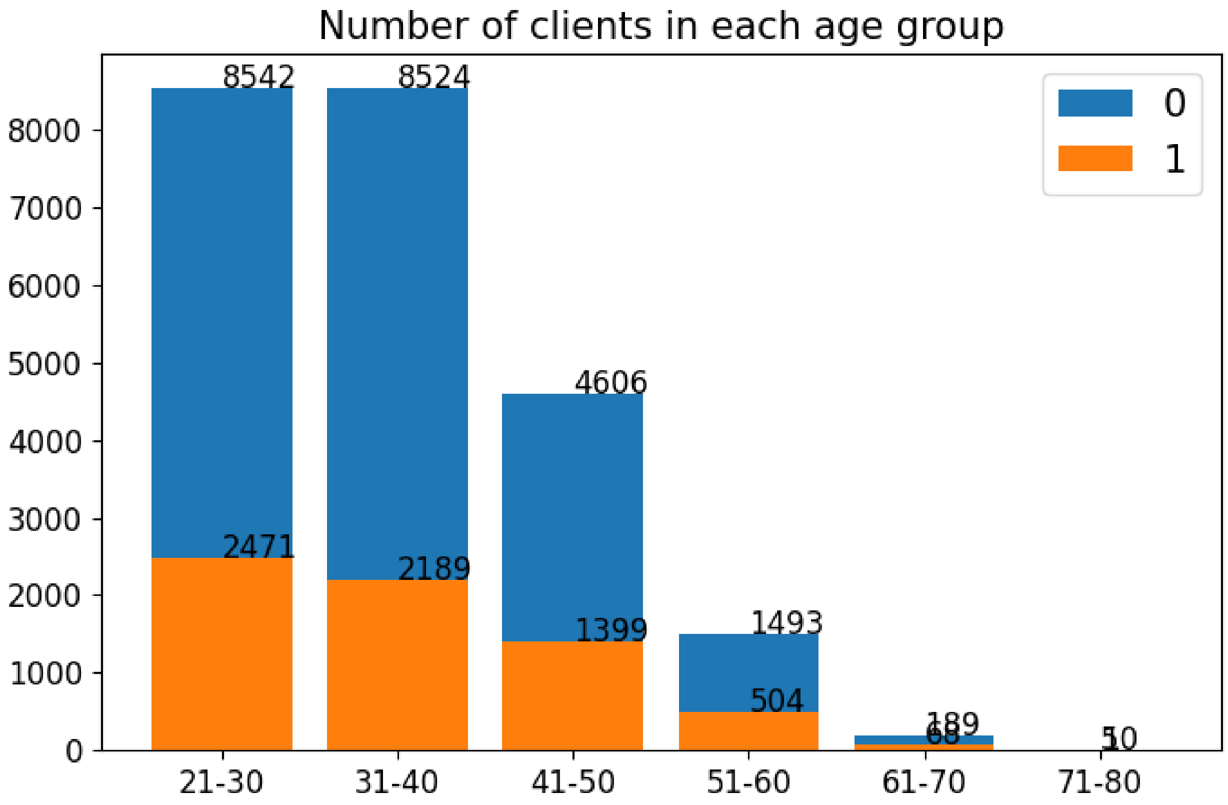


```
plt.subplots(figsize=(20,5))
plt.subplot(121)
sns.distplot(dfcopy.LIMIT_BAL)

plt.subplot(122)
```

```
sns.distplot(dfcopy.AGE)

plt.show()

<ipython-input-14-2d4331f022e0>:2: 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-14-2d4331f022e0>: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(dfcopy.LIMIT_BAL)
<ipython-input-14-2d4331f022e0>: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(dfcopy.AGE)
```



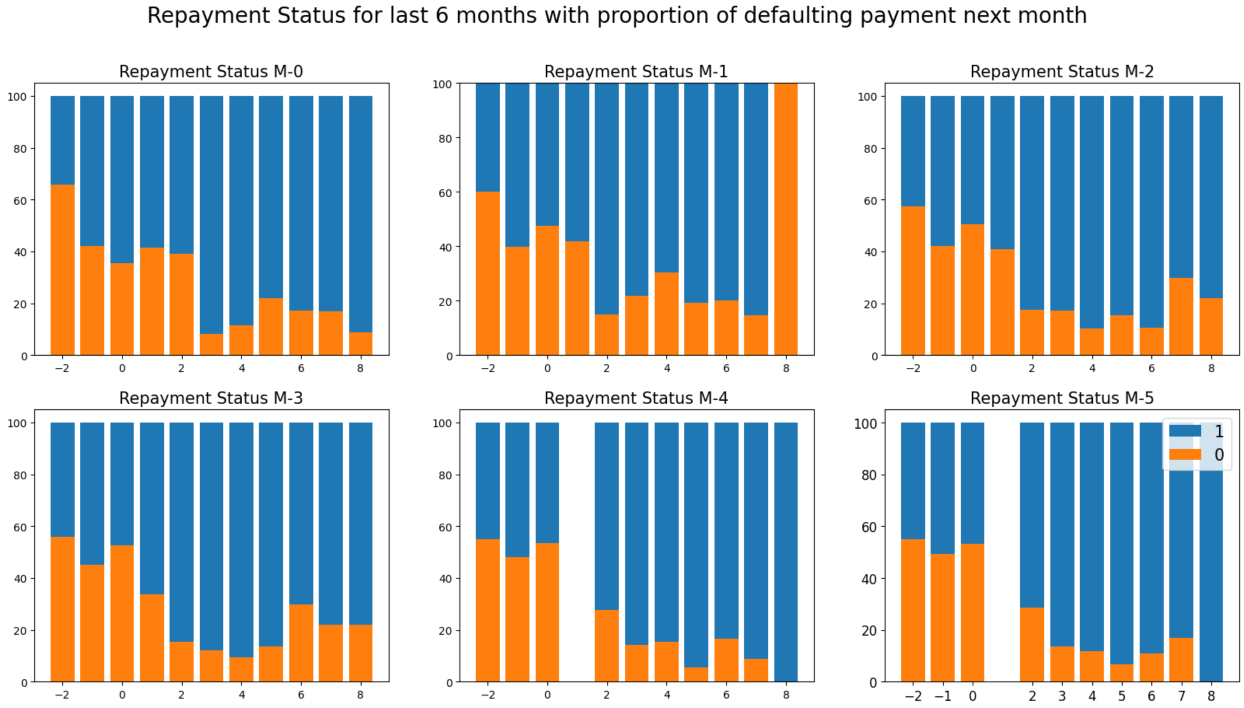
```
bins = [20,30,40,50,60,70,80]
names = ['21-30','31-40','41-50','51-60','61-70','71-80']
dfcopy['AGE_BIN'] = pd.cut(x=dfcopy.AGE, bins=bins, labels=names,
right=True)

age_cnt = dfcopy.AGE_BIN.value_counts()
age_0 = (dfcopy.AGE_BIN[dfcopy['def_pay'] == 0].value_counts())
age_1 = (dfcopy.AGE_BIN[dfcopy['def_pay'] == 1].value_counts())

plt.subplots(figsize=(8,5))
# sns.barplot(data=defaulters, x='AGE_BIN', y='LIMIT_BAL',
hue='def_pay', ci=0)
plt.bar(age_0.index, age_0.values, label='0')
plt.bar(age_1.index, age_1.values, label='1')
for x,y in zip(names,age_0):
    plt.text(x,y,y,fontsize=12)
for x,y in zip(names,age_1):
    plt.text(x,y,y,fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.title("Number of clients in each age group", fontsize=15)
plt.legend(loc='upper right', fontsize=15)
plt.show()
```



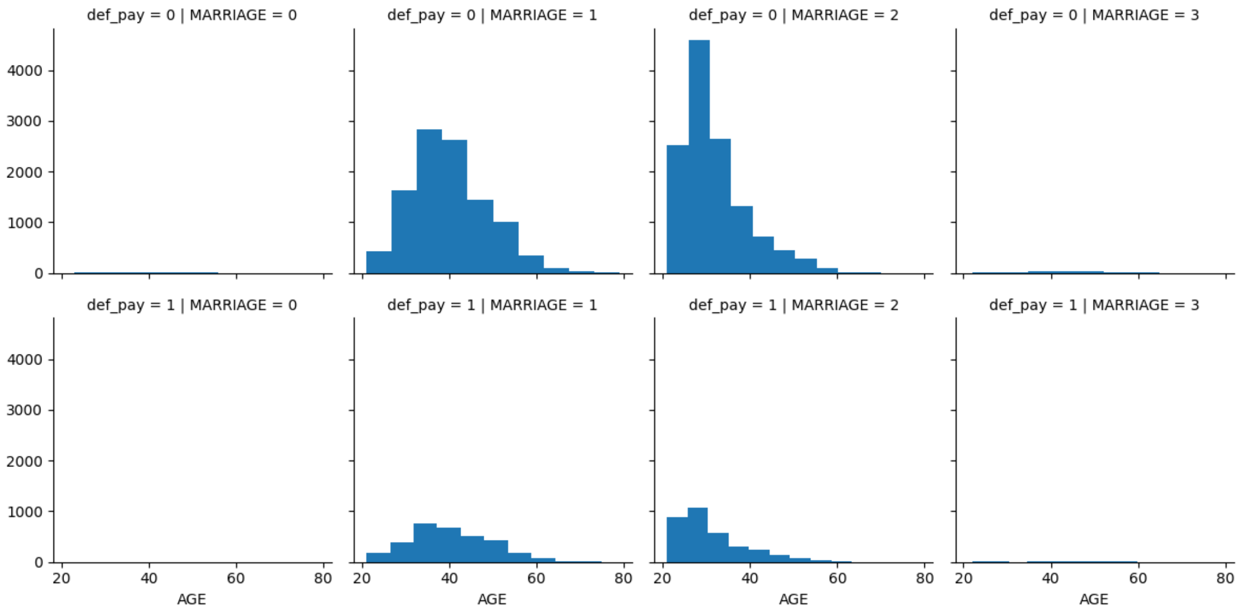
```
plt.subplots(figsize=(20,10))

ind = sorted(dfcopy.PAY_1.unique())
pay_0 = (dfcopy.PAY_1[dfcopy['def_pay'] ==
0].value_counts(normalize=True))
pay_1 = (dfcopy.PAY_1[dfcopy['def_pay'] ==
1].value_counts(normalize=True))
total = pay_0.values+pay_1.values
pay_0_prop = np.true_divide(pay_0, total)*100
pay_1_prop = np.true_divide(pay_1, total)*100
plt.subplot(231)
plt.bar(ind, pay_1_prop, bottom=pay_0_prop, label='1')
plt.bar(ind, pay_0_prop, label='0')
plt.title("Repayment Status M-0", fontsize=15)

ind = sorted(dfcopy.PAY_2.unique())
pay_0 = (dfcopy.PAY_2[dfcopy['def_pay'] ==
0].value_counts(normalize=True))
pay_1 = (dfcopy.PAY_2[dfcopy['def_pay'] ==
1].value_counts(normalize=True))
for i in pay_0.index:
  if i not in pay_1.index:
        pay_1[i]=0
total = pay_0.values+pay_1.values
pay_0_prop = np.true_divide(pay_0, total)*100
pay_1_prop = np.true_divide(pay_1, total)*100
plt.subplot(232)
plt.bar(ind, pay_1_prop, bottom=pay_0_prop, label='1')
plt.bar(ind, pay_0_prop, label='0')
plt.title("Repayment Status M-1", fontsize=15)

ind = sorted(dfcopy.PAY_3.unique())
pay_0 = (dfcopy.PAY_3[dfcopy['def_pay'] ==
0].value_counts(normalize=True))
pay_1 = (dfcopy.PAY_3[dfcopy['def_pay'] ==
1].value_counts(normalize=True))
for i in pay_0.index:
    if i not in pay_1.index:
        pay_1[i]=0
total = pay_0.values+pay_1.values
pay_0_prop = np.true_divide(pay_0, total)*100
pay_1_prop = np.true_divide(pay_1, total)*100
plt.subplot(233)
plt.bar(ind, pay_1_prop, bottom=pay_0_prop, label='1')
plt.bar(ind, pay_0_prop, label='0')
plt.title("Repayment Status M-2", fontsize=15)

ind = sorted(dfcopy.PAY_4.unique())
pay_0 = (dfcopy.PAY_4[dfcopy['def_pay'] ==
0].value_counts(normalize=True))
```

```
pay_1 = (dfcopy.PAY_4[dfcopy['def_pay'] ==
1].value_counts(normalize=True))
for i in pay_0.index:
    if i not in pay_1.index:
        pay_1[i]=0
total = pay_0.values+pay_1.values
pay_0_prop = np.true_divide(pay_0, total)*100
pay_1_prop = np.true_divide(pay_1, total)*100
plt.subplot(234)
plt.bar(ind, pay_1_prop, bottom=pay_0_prop, label='1')
plt.bar(ind, pay_0_prop, label='0')
plt.title("Repayment Status M-3", fontsize=15)


ind = sorted(dfcopy.PAY_5.unique())
pay_0 = (dfcopy.PAY_5[dfcopy['def_pay'] ==
0].value_counts(normalize=True))
pay_1 = (dfcopy.PAY_5[dfcopy['def_pay'] ==
1].value_counts(normalize=True))
for i in pay_0.index:
    if i not in pay_1.index:
        pay_1[i]=0
for i in pay_1.index:
    if i not in pay_0.index:
        pay_0[i]=0
total = pay_0.values+pay_1.values
pay_0_prop = np.true_divide(pay_0, total)*100
pay_1_prop = np.true_divide(pay_1, total)*100
plt.subplot(235)
plt.bar(ind, pay_1_prop, bottom=pay_0_prop, label='1')
plt.bar(ind, pay_0_prop, label='0')
plt.title("Repayment Status M-4", fontsize=15)

ind = sorted(dfcopy.PAY_6.unique())
pay_0 = (dfcopy.PAY_6[dfcopy['def_pay'] ==
0].value_counts(normalize=True))
pay_1 = (dfcopy.PAY_6[dfcopy['def_pay'] ==
1].value_counts(normalize=True))
for i in pay_0.index:
    if i not in pay_1.index:
        pay_1[i]=0
for i in pay_1.index:
    if i not in pay_0.index:
        pay_0[i]=0
total = pay_0.values+pay_1.values
pay_0_prop = np.true_divide(pay_0, total)*100
pay_1_prop = np.true_divide(pay_1, total)*100
plt.subplot(236)
plt.bar(ind, pay_1_prop, bottom=pay_0_prop, label='1')
plt.bar(ind, pay_0_prop, label='0')
plt.title("Repayment Status M-5", fontsize=15)

plt.xticks(ind, fontsize=12)
plt.yticks(fontsize=12)
plt.legend(loc="upper right", fontsize=15)
plt.suptitle("Repayment Status for last 6 months with proportion of
defaulting payment next month", fontsize=20)
plt.show()
```

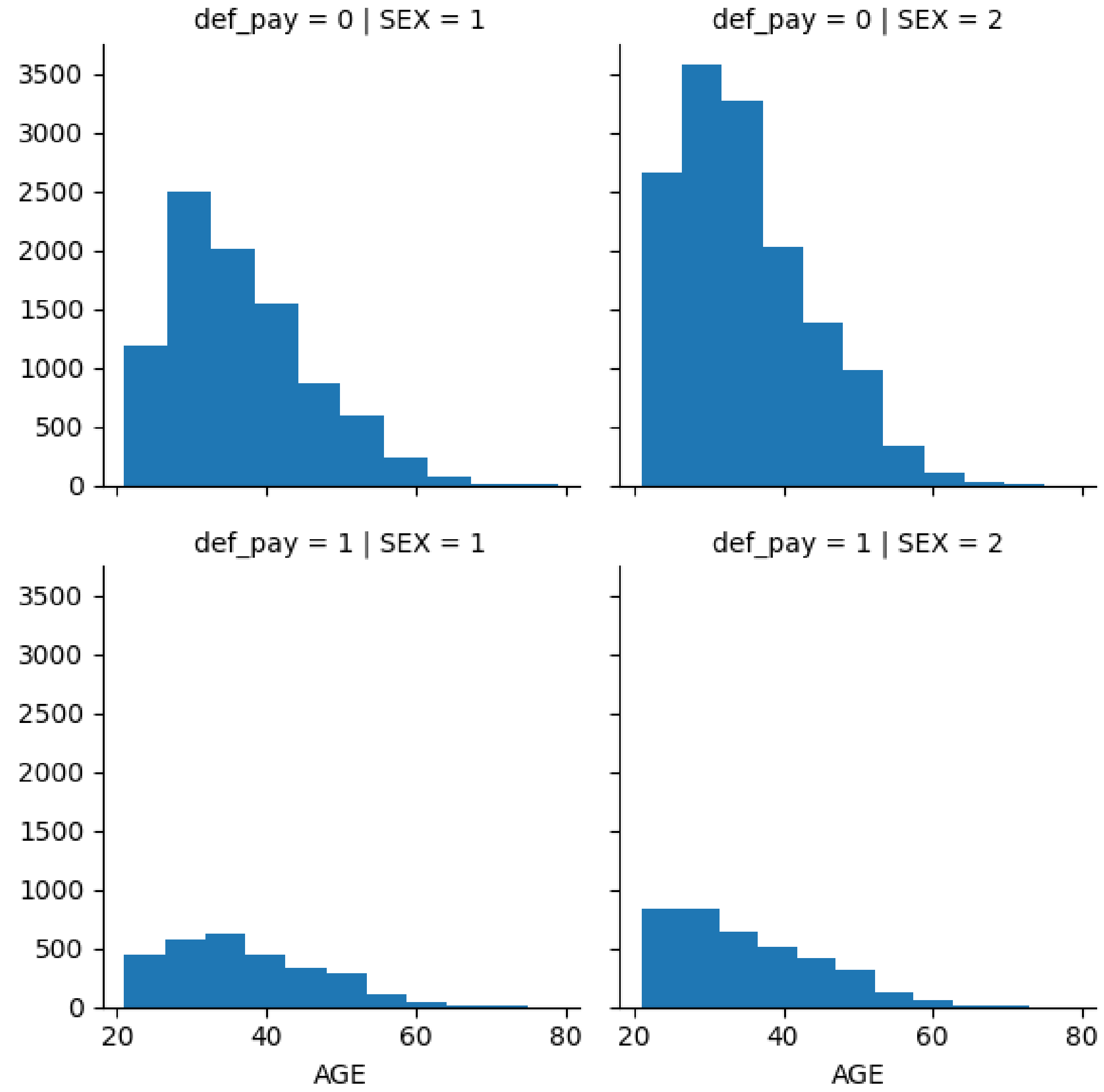
```
<ipython-input-16-17d502144844>:9: MatplotlibDeprecationWarning: Auto-
removal of overlapping axes is deprecated since 3.6 and will be
removed two minor releases later; explicitly call ax.remove() as
needed.
  plt.subplot(231)
```



```
g = sns.FacetGrid(dfcopy, row='def_pay', col='MARRIAGE')
g = g.map(plt.hist, 'AGE')
plt.show()
```

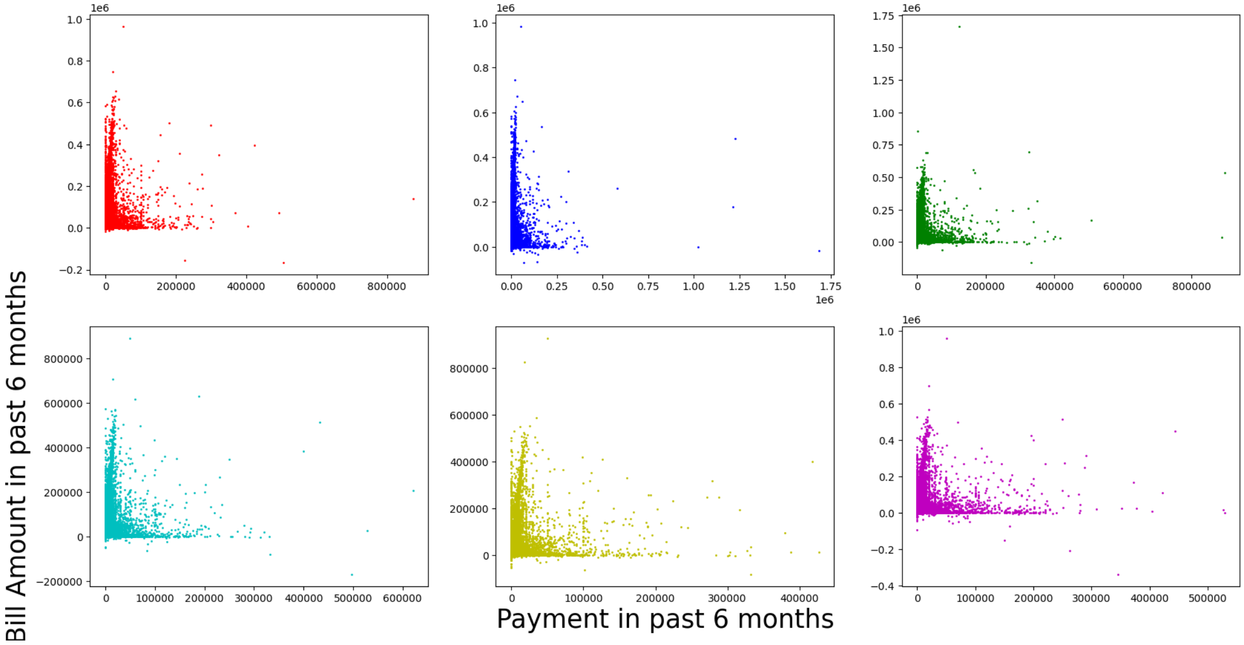


```
g = sns.FacetGrid(dfcopy, row='def_pay', col='SEX')
g = g.map(plt.hist, 'AGE')
```

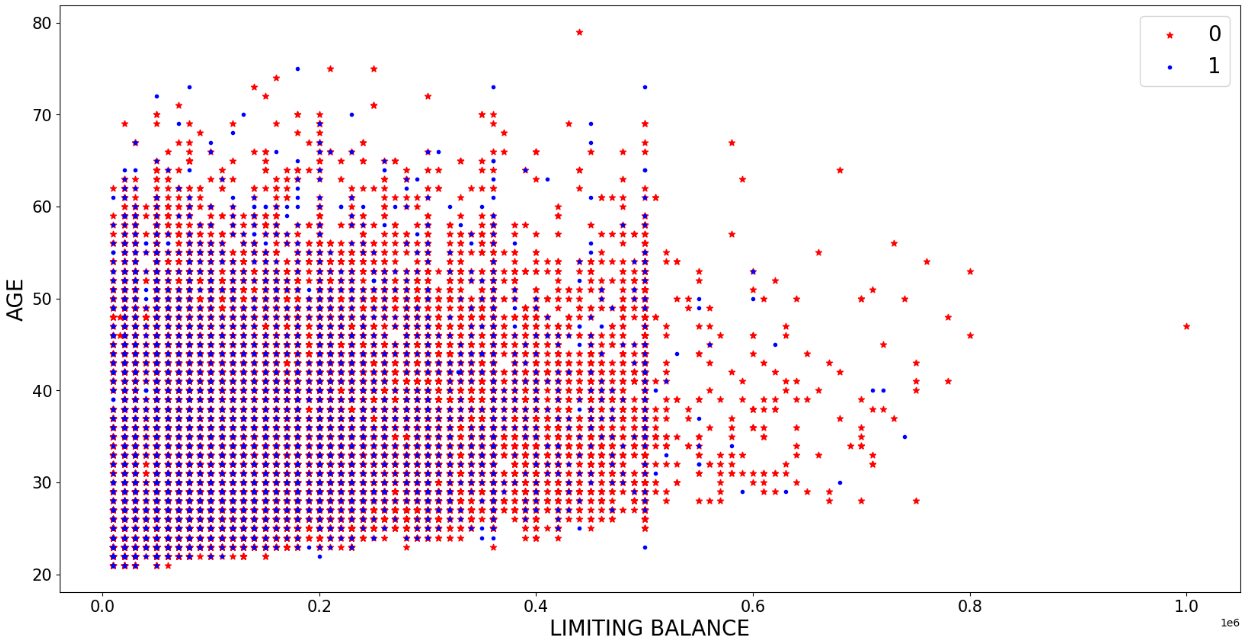


```
plt.subplots(figsize=(20,10))

plt.subplot(231)
plt.scatter(x=dfcopy.PAY_AMT1, y=dfcopy.BILL_AMT1, c='r', s=1)

plt.subplot(232)
plt.scatter(x=dfcopy.PAY_AMT2, y=dfcopy.BILL_AMT2, c='b', s=1)

plt.subplot(233)
plt.scatter(x=dfcopy.PAY_AMT3, y=dfcopy.BILL_AMT3, c='g', s=1)

plt.subplot(234)
plt.scatter(x=dfcopy.PAY_AMT4, y=dfcopy.BILL_AMT4, c='c', s=1)
plt.ylabel("Bill Amount in past 6 months", fontsize=25)

plt.subplot(235)
plt.scatter(x=dfcopy.PAY_AMT5, y=dfcopy.BILL_AMT5, c='y', s=1)
plt.xlabel("Payment in past 6 months", fontsize=25)

plt.subplot(236)
plt.scatter(x=dfcopy.PAY_AMT6, y=dfcopy.BILL_AMT6, c='m', s=1)

plt.show()
```

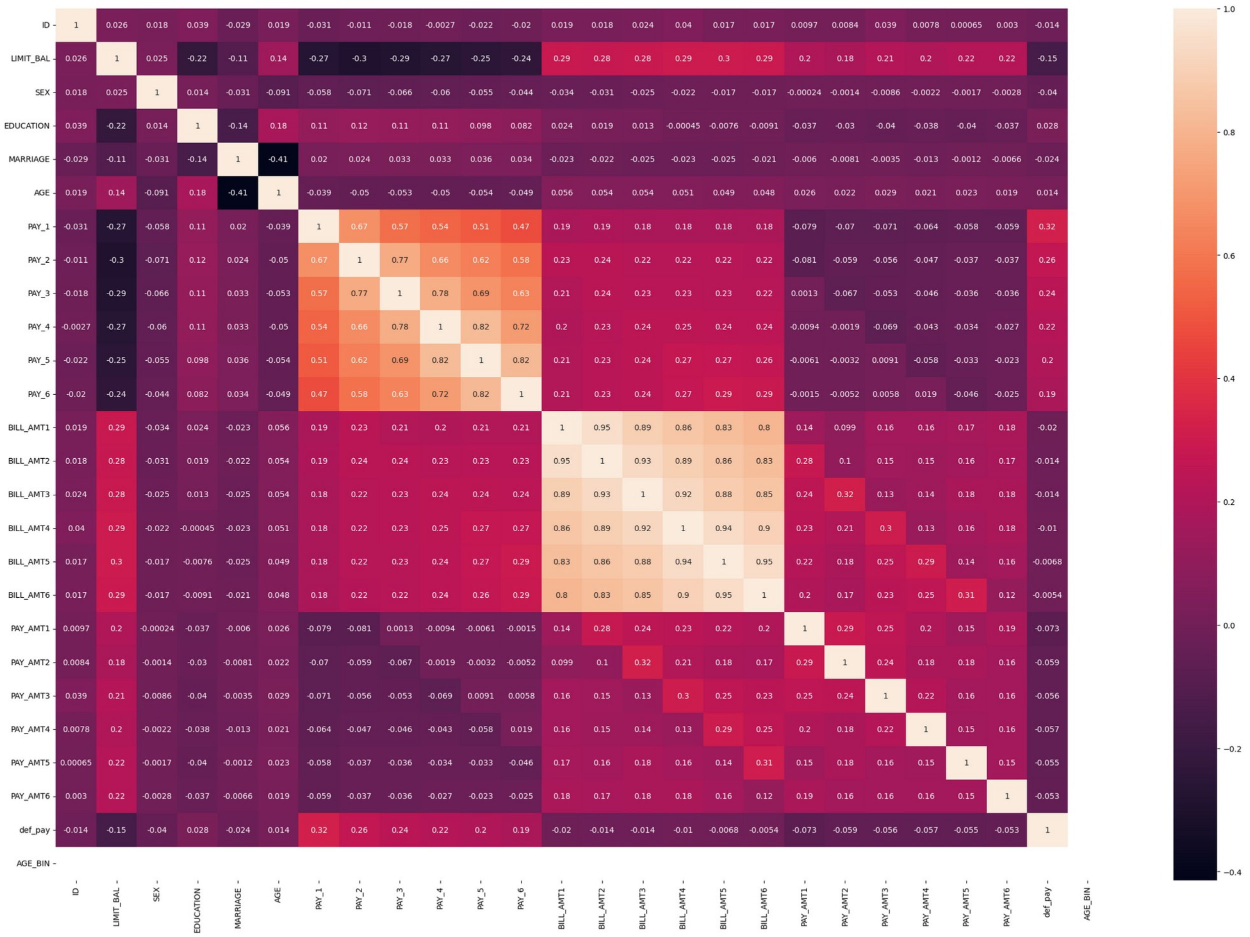
```
<ipython-input-19-166d85847eaf>:3: MatplotlibDeprecationWarning: Auto-
removal of overlapping axes is deprecated since 3.6 and will be
removed two minor releases later; explicitly call ax.remove() as
needed.
  plt.subplot(231)
```



```
y1 = dfcopy.AGE[dfcopy["def_pay"] == 0]
y2 = dfcopy.AGE[dfcopy["def_pay"] == 1]
x1 = dfcopy.LIMIT_BAL[dfcopy["def_pay"] == 0]
x2 = dfcopy.LIMIT_BAL[dfcopy["def_pay"] == 1]

fig,ax = plt.subplots(figsize=(20,10))
plt.scatter(x1,y1, color="r", marker="*", label='0')
plt.scatter(x2,y2, color="b", marker=".", label='1')
plt.xlabel("LIMITING BALANCE", fontsize=20)
plt.ylabel("AGE", fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.legend(loc='upper right', fontsize=20)
plt.show()
```



```
defaulters_numeric = dfcopy.apply(pd.to_numeric, errors='coerce')

# Compute correlation matrix
corr_matrix = defaulters_numeric.corr()

# Plot correlation heatmap
plt.subplots(figsize=(30, 20))
sns.heatmap(corr_matrix, annot=True)
plt.show()
```



```
dfcopy.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 26 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
 25  AGE_BIN    30000 non-null  category
dtypes: category(1), float64(13), int64(12)
memory usage: 5.8 MB
```

```python
X = dfcopy.drop(columns=['ID', 'def_pay'])  # Features
y = dfcopy['def_pay']  # Target variable

from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
X_encoded = X.apply(label_encoder.fit_transform)
selector_chi2 = SelectKBest(score_func=chi2, k=7)
X_selected_chi2 = selector_chi2.fit_transform(X_encoded, y)

selector_anova = SelectKBest(score_func=f_classif, k=7)
X_selected_anova = selector_anova.fit_transform(X_encoded, y)

X_selected = pd.concat([pd.DataFrame(X_selected_chi2),
pd.DataFrame(X_selected_anova)], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X_selected, y,
test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

model = LogisticRegression()
model.fit(X_train_scaled, y_train)

y_pred = model.predict(X_test_scaled)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

print("Classification Report:")
print(classification_report(y_test, y_pred))

conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Accuracy: 0.8048333333333333
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.96      0.89      4687
           1       0.65      0.23      0.34      1313

    accuracy                           0.80      6000
   macro avg       0.73      0.60      0.61      6000
weighted avg       0.78      0.80      0.77      6000

Confusion Matrix:
[[4522  165]
 [1006  307]]
```