BT2103 Group Project

Group Members:

Tan Ze En A0233293H
Glen Peh A0234472H
Matthew Robinson A0235240U

Brief Introduction of dataset and data modeling problem

The dataset contains information about customers in Taiwan from April 2005 to September 2005 and their default status for the next month. Hence, our group is interested to find out how well we are able to accurately predict the customer's default status based on the attributes of the customers using a support vector machine model and a logistic regression model. To find out attributes that we feel would be relevant in predicting the default status of the model, we will conduct an exploratory data analysis first.

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	default payment next month
0	1	20000	2	2	1	24	2	2	-1	-1	...	0	0	0	0	689	0	0	0	0	1
1	2	120000	2	2	2	26	-1	2	0	0	...	3272	3455	3261	0	1000	1000	1000	0	2000	1
2	3	90000	2	2	2	34	0	0	0	0	...	14331	14948	15549	1518	1500	1000	1000	1000	5000	0
3	4	50000	2	2	1	37	0	0	0	0	...	28314	28959	29547	2000	2019	1200	1100	1069	1000	0
4	5	50000	1	2	1	57	-1	0	-1	0	...	20940	19146	19131	2000	36681	10000	9000	689	679	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
29995	29996	220000	1	3	1	39	0	0	0	0	...	88004	31237	15980	8500	20000	5003	3047	5000	1000	0
29996	29997	150000	1	3	2	43	-1	-1	-1	-1	...	8979	5190	0	1837	3526	8998	129	0	0	0
29997	29998	30000	1	2	2	37	4	3	2	-1	...	20878	20582	19357	0	0	22000	4200	2000	3100	1
29998	29999	80000	1	3	1	41	1	-1	0	0	...	52774	11855	48944	85900	3409	1178	1926	52964	1804	1
29999	30000	50000	1	2	1	46	0	0	0	0	...	36535	32428	15313	2078	1800	1430	1000	1000	1000	1

Exploratory Data Analysis

First, we check for any missing values.

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

After checking for null values in our dataset, we found that there are no missing values within our data set. When further checking the sum for null values, the output is zero.

Next, we check to see how many individuals will default next month.

Default	6636
No default	23364

In the entire dataset, 22.1% of the customers default while 77.8% of the customers did not default. The customers who default is 6636/23364.

<u>Feature Columns in the dataset</u>

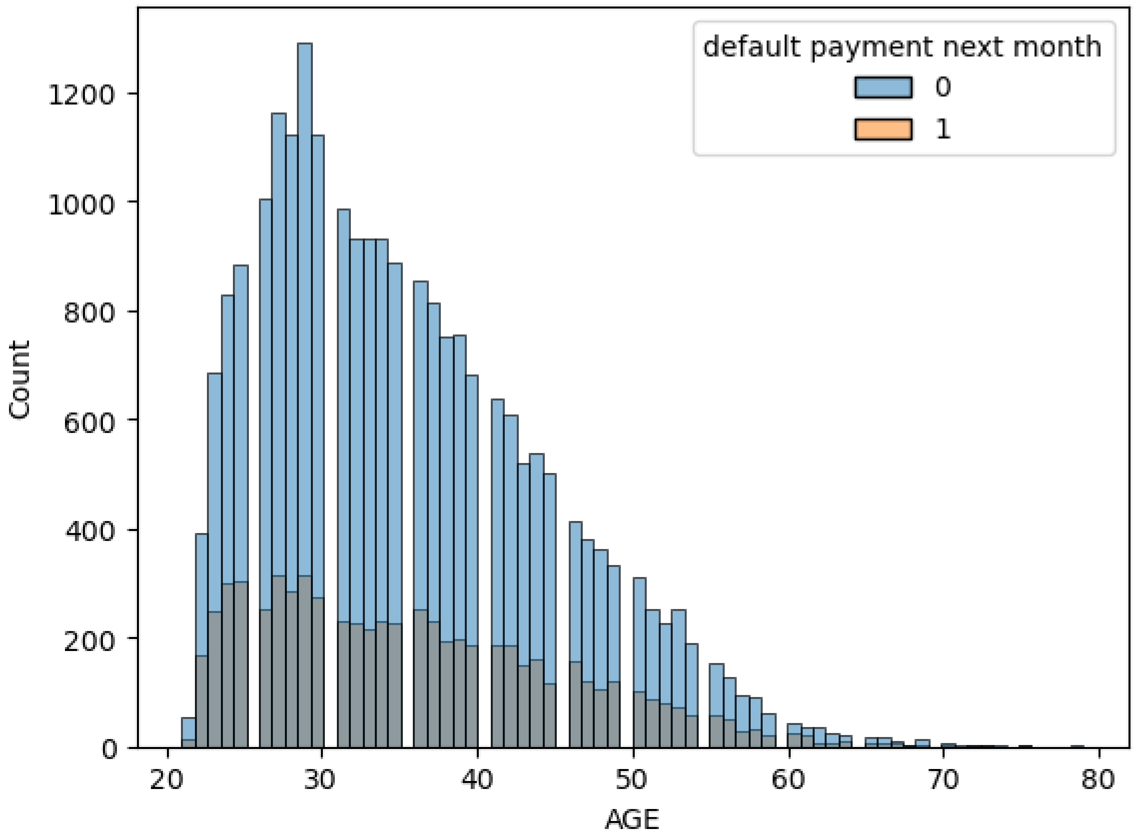
There are 24 feature columns in the dataset.

<u>Age</u>

The summary statistics for Age is as shown below.

count	30 000
mean	35.485
std	9.21
min	21
25%	28
50%	34
75%	41
max	79

In order to view this in a distribution, a histogram can be plotted.



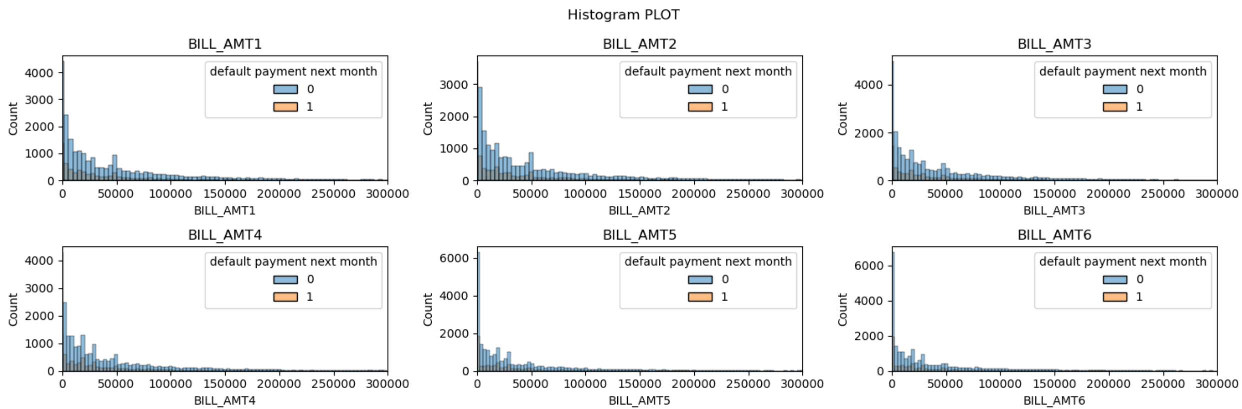
From the histogram, it seems that there is a greater percentage of individuals belonging to the age group of 20-40 to default. We will consider performing data pre processing on this category for the data to be easier to understand.

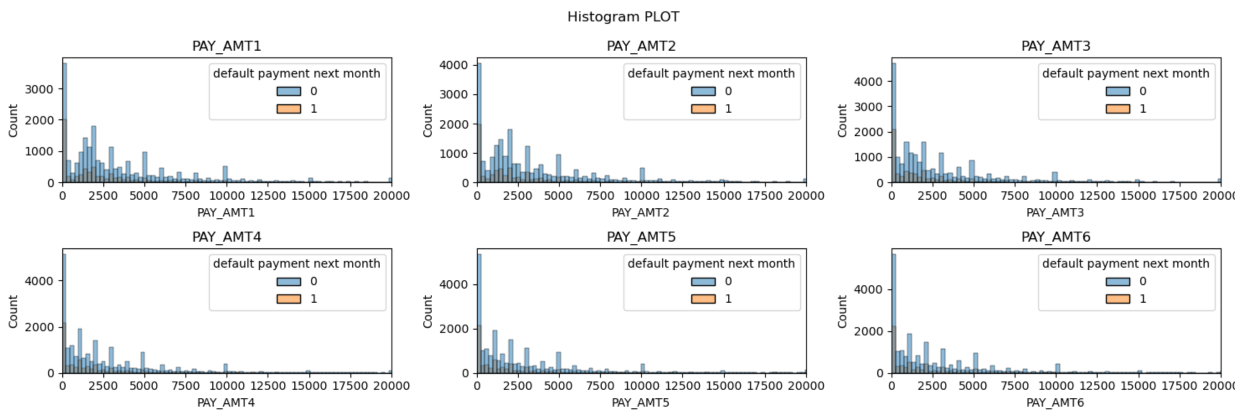
When evaluating the Kurtosis, it is greater than 3. The distribution of Age is a leptokurtic Skew and is greater than 0, so the distribution of Age is positively skewed. The skew function is said to output 0.732.

Kurtosis	3.044
Skew	0.732

Bill_AMT and Pay_AMT

We view all the histograms of all Bill_AMT and Pay_AMT





We can see that the bill amount are mostly in the range of [0, 50, 000] while the pay amount are usually in the range of [0, 2500]

Limit_Bal

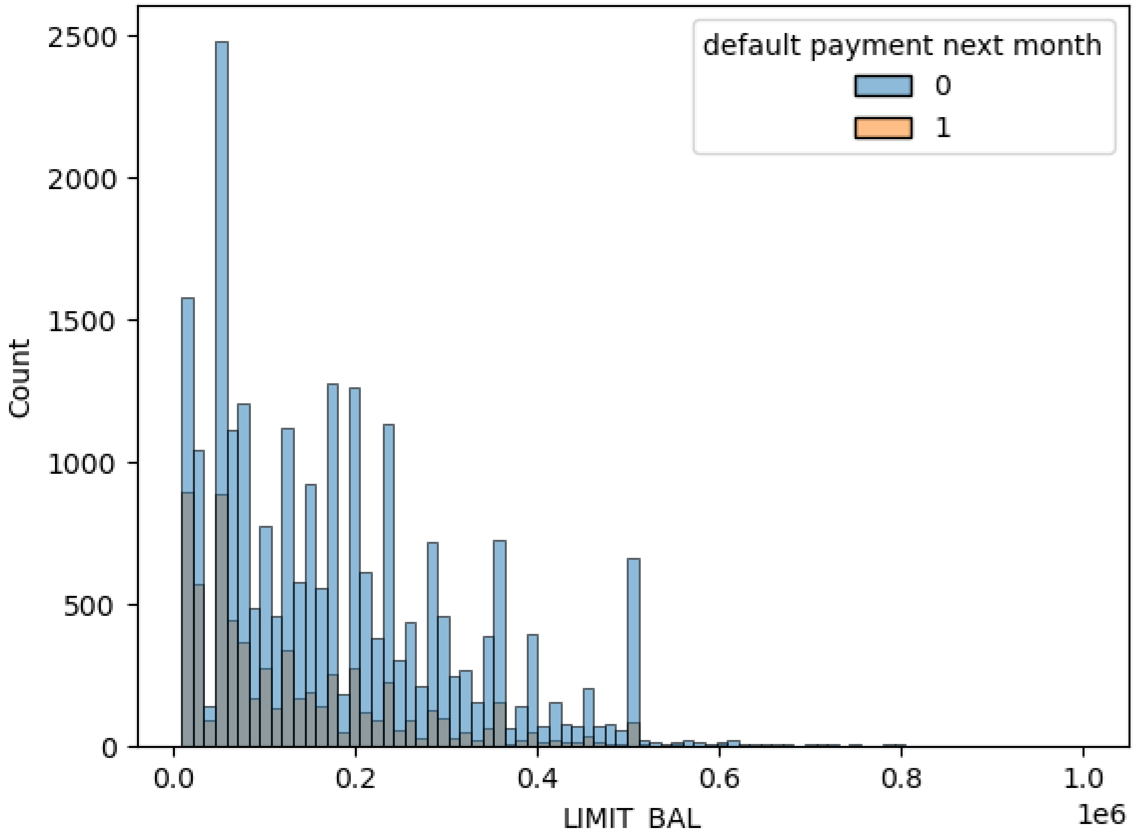
The summary statistics for Limit_Bal is as shown below.

count	30 000
mean	167 484.32
std	129 747.66
min	10 000
25%	50 000
50%	140 000
75%	240 000
max	1 000 000

Next, when finding out the degree of kurtosis, we see that it has a value of **3.54**, hence the distribution of LIMIT_BAL is leptokurtic. Skew is greater than 0, at **0.993**, so it is positively skewed.

Kurtosis	3.54
Skew	0.993

This can be seen in the Histogram shown below.



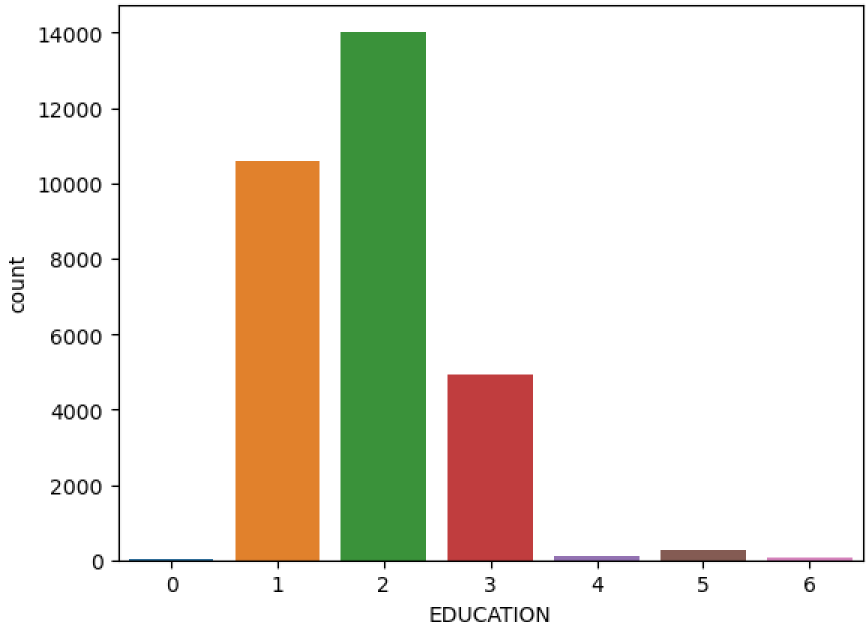
From the histogram, we see that more people default when their limit_bal is between 0 and 0.2 * 10^6. We will conduct data discretization for this variable.

Anomalies and Inconsistencies in the dataset

Anomalies

<u>EDUCATION</u>

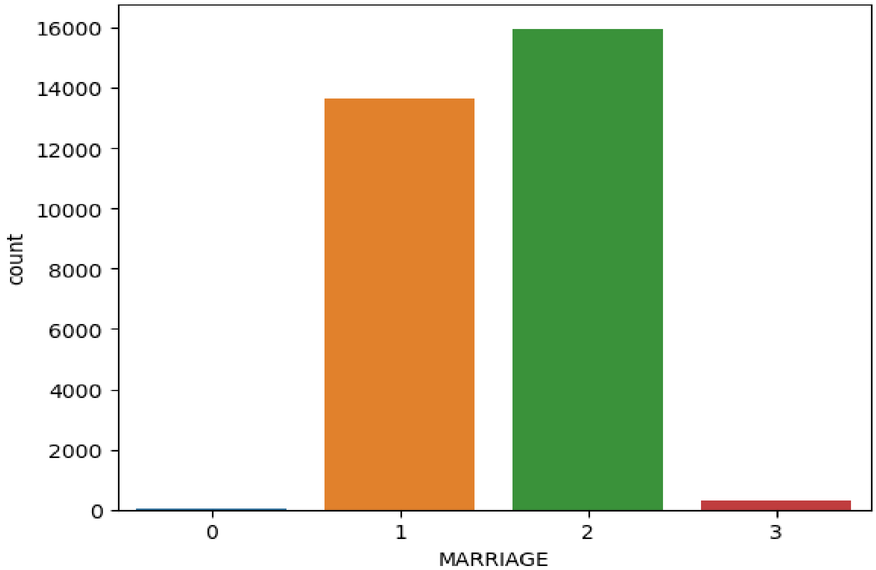
We use a barplot to examine EDUCATION.



There are 14 rows of data that have EDUCATION coded as 0 which was not explained by the author. Hence, we do not know whether the extra value was a typographical error or it means another form of education level that the author did not explain.

We similarly plot a barplot for MARRIAGE.

<u>MARRIAGE</u>



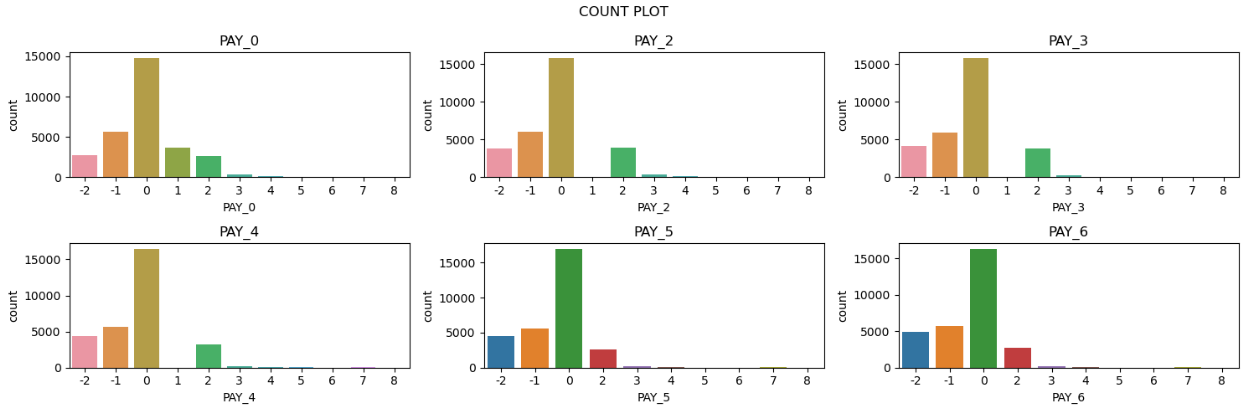
The numerical count of each value of Marriage is as shown above.

2	15964
1	13659
3	323
0	54

Similarly, the MARRIAGE variable also has an extra possible value that is 0 which was also not explained by the author. There are 54 rows of data that have MARRIAGE encoded as 0.

PAY

We plot all the COUNT PLOTs of all attributes of PAY below.



There are 14737 counts of PAY_0 encoded as 0. When putting it as a fraction of the total count, the output is 0.49123.

We can observe that there are extra values that the repayment status in our dataset can take on. Specifically, we see that the repayment status can be 0 or -2 which was not explained in the dataset description. Furthermore, there are many rows that have these extra values. For example in PAY_0 alone there are already 14737 rows of data that have PAY_0 being encoded as 0 which is already about 50% of the entire dataset. Therefore, we should not filter out all the data that have these extra values encoded since that would mean that the size of our dataset will reduce dramatically. Given that there are many rows of data that contain values that were not explained, we need to perform data cleaning and preprocessing later before starting to train our model.

SEX

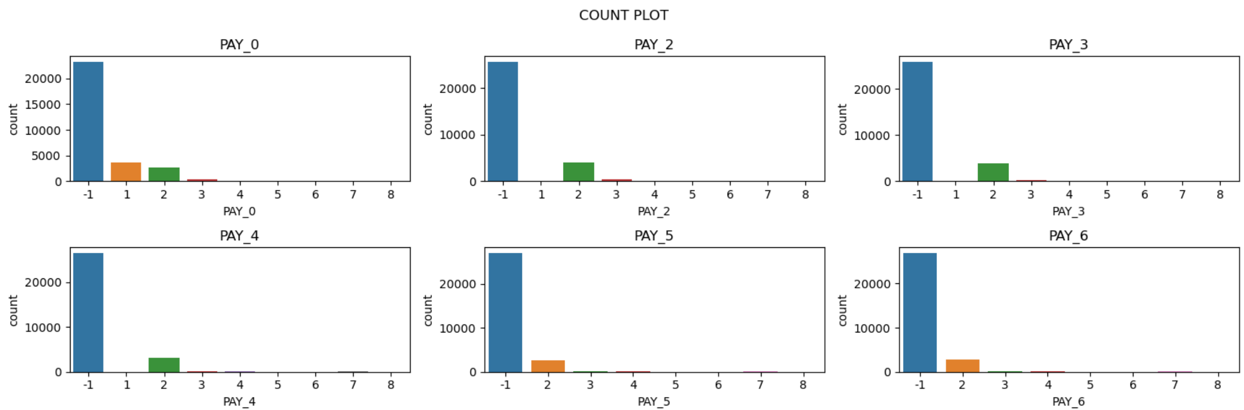
As for SEX, there are no unexplained encodings observed. Value counts are either 1 or 2.

Data Cleaning and preprocessing

We have seen from our exploratory data analysis that there are several values that were not explained and many rows in the dataset contain such values. Hence, we decide to see if we are able to encode these values to another value that makes sense for our dataset.

Repayment Status

Since the value in repayment status represents the number of months for payment delay, our group feels that 0 and -2 is indicative that there are 0 months in payment delay. Hence, we conclude to encode 0 and -2 to -1 which represents duly payments. When plotting the counts for all PAY in a barplot, the following is shown.



We create a feature labeled "LATE" to separate customers into two groups for simpler understanding. The two main groups are split according to individuals who have been late in their payment before, and those who have never been late before. There is a greater tendency for individuals who have been late in their payments to default as they may be experiencing some financial difficulties in their lives.

LATE	1
NOT LATE	0

Education and Marriage

Since the value 5 and 6 for Education is unknown and the value of 0 for Education is not explained, we group all of them under the value 4, which is ‘others’. For the value 0 under Marriage, we decide to group them under value 3 which is ‘others’.

Age

We perform data discretization and have set the age into three distinct classes.

20 < Age <= 40	1
40 < Age <= 60	2
Age > 60	3

The new feature column is named “NEW_AGE”

Limit Balance

We perform data discretization and have set the age into three distinct classes.

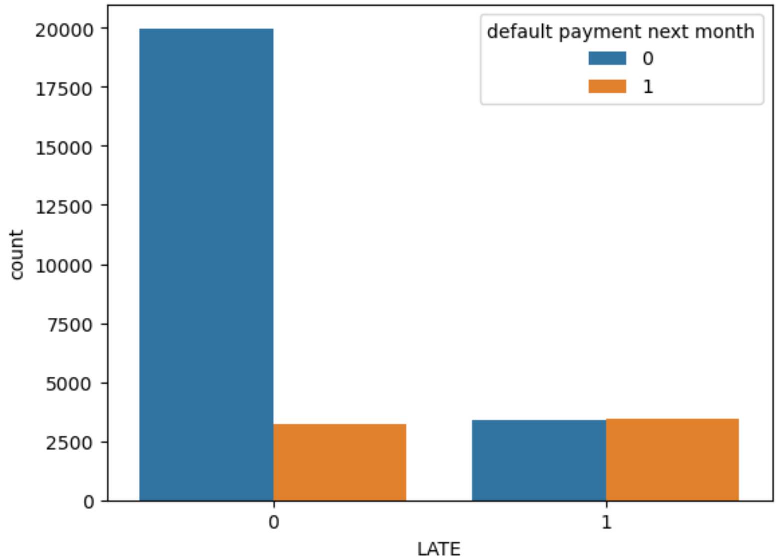
0 < LIMIT_BAL <= 200 000	1
200 000 < LIMIT_BAL <= 400 000	2
LIMIT_BAL > 400 000	3

The new feature column is named “NEW_LIMIT_BAL”

Three new feature columns are created which are “NEW_LIMIT_BAL”, “NEW_AGE” and “LATE”.

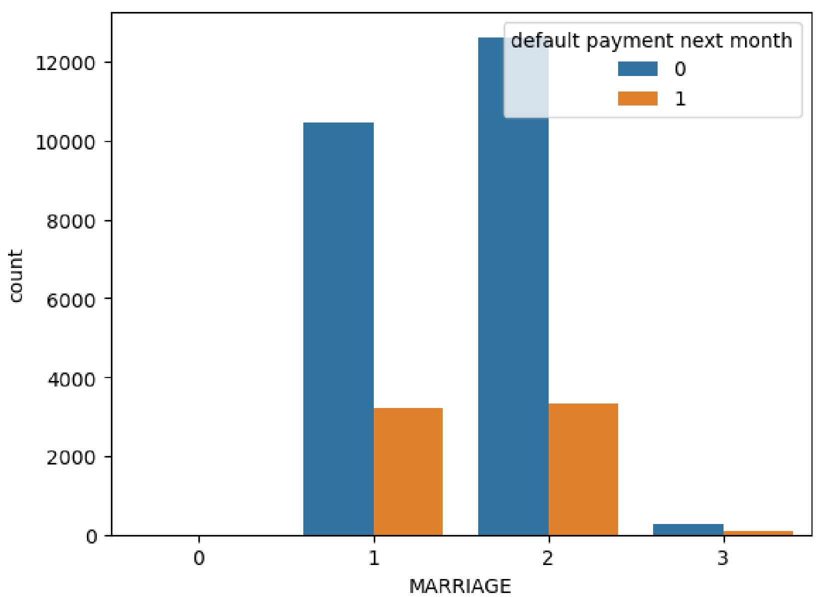
Checking for trends after data preprocessing

Countplot for individuals who have been late in payment before



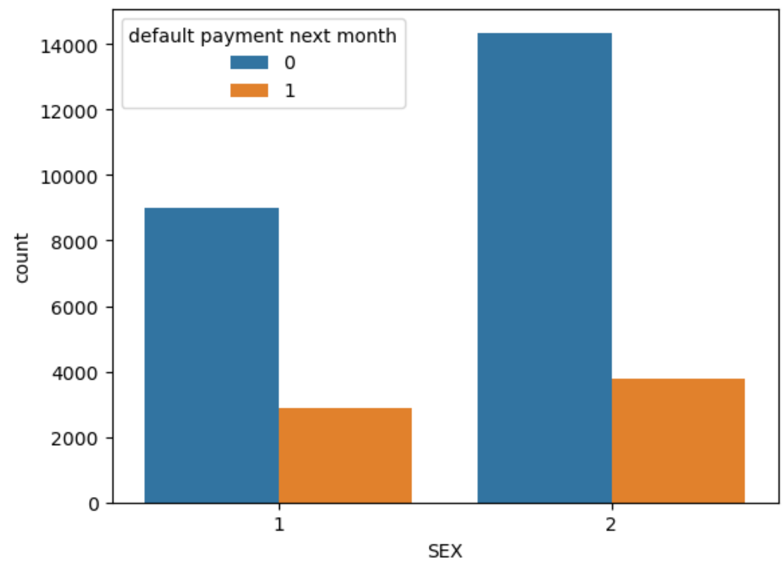
From the graph we can see that a greater percentage of those who have been late in their payments before to default in the next month. This can also be seen in our calculation where among customers who are late before, 50.29% will default the next month. It seems that this feature will be useful in predicting whether customers will default.

Countplot for individuals based on their marriage status



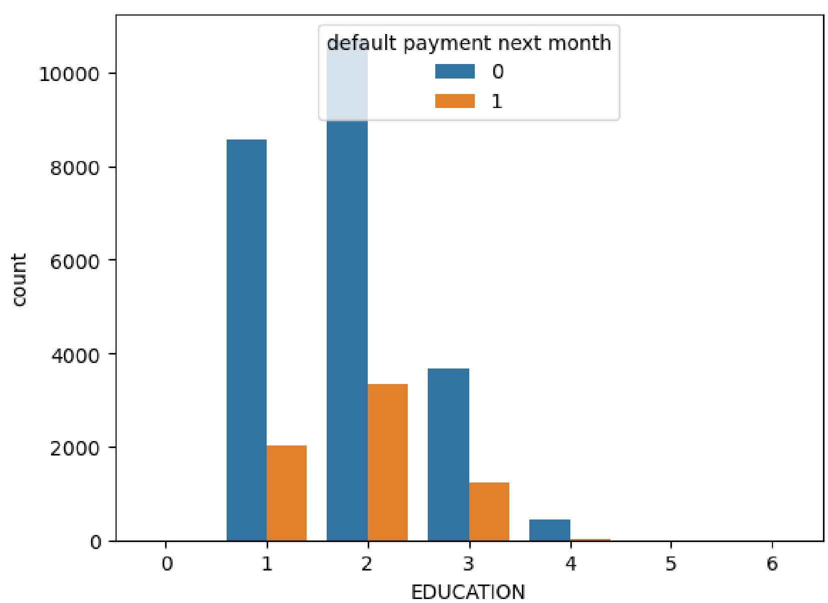
From the graph, it seems that more customers default when they are either single or married.

Countplot for individuals based on their sex



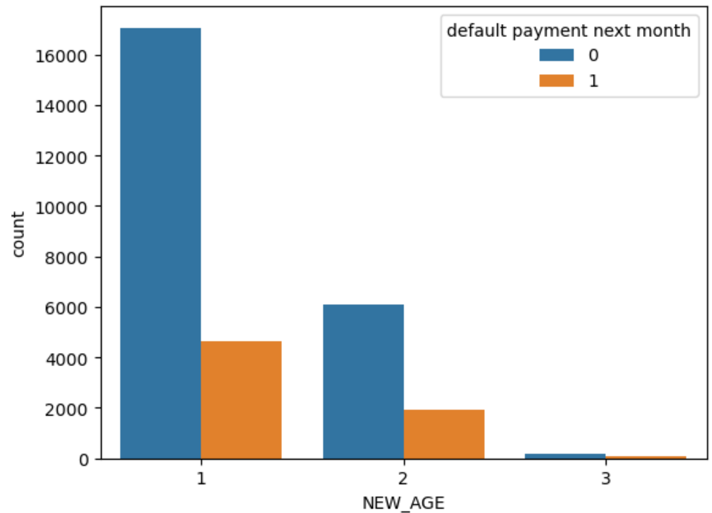
Overall, it seems that individuals who are male are more likely to default than females. Among all the males, 23.17% defaults which is more than the percentage of females(20.78%) who will default.

Countplot for individuals based on their education



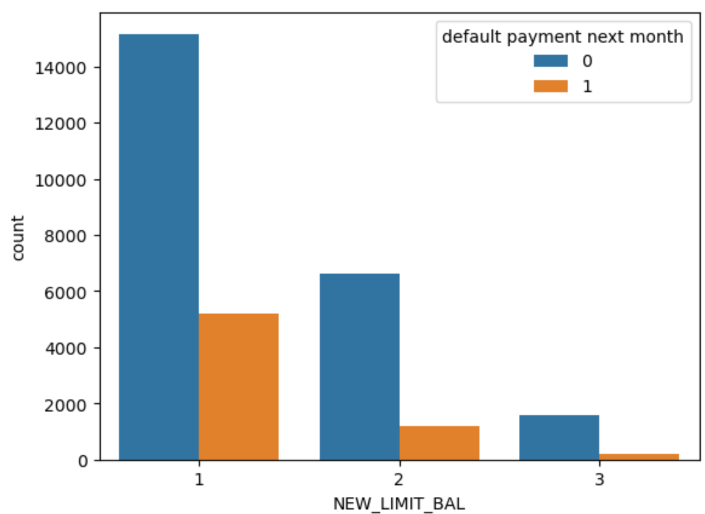
From the graph and based on our further calculations, it seems that there is a higher number of individuals with an education level of university to default.

Countplot for new Age



From the graph, we can see that most individuals who default are from the age range of [20, 60].

Countplot for new Limit Balance



From the graph, we can see that most individuals who default have a limit balance range of [0, 200 000].

Feature Selection
We conduct a Chi2 test to measure the association for categorical data.

Null Hypothesis: Feature variable and default status are independent
Alternate Hypothesis : Those two attributes are associated.
Level of significance: 95%

The features that our group is going to test are “SEX”, “LATE”, “EDUCATION” and “NEW_LIMIT_BAL”, “NEW_AGE” and “MARRIAGE”.

The p value of the Chi2 test is in the table below.

Feature Variable	p-value
SEX	4.944678999412044e-12
LATE	Very close to 0
EDUCATION	1.495064564810615e-34
NEW_LIMIT_BAL	2.3082716470487475e-100
NEW_AGE	1.6554284303886136e-05
MARRIAGE	7.790720364202813e-07

All the p-values reported are less than the p value of 0.05. Hence, the results are significant at $p < 0.05$ and we reject the H0 in favor of Ha. Hence, these feature variables may be useful input attributes in the model when we are trying to predict if the customer will default.

Hence, the features that our group has selected to train the model are “SEX”, “LATE”, “EDUCATION” and “NEW_LIMIT_BAL”, “NEW_AGE” and “MARRIAGE”. We first split our dataset into a training set and test set. 75% of our data will go to the training set and 25% of the data will go to the test set.

The classifiers that we have chosen to train our model are SVM and Logistic Regression. We use GridSearchCV to find the most suitable parameters for our model. The metric we used to decide the best parameters for the model is the area under the Receiver Operating Characteristic Curve.

Model Selection & Model Evaluation

SVM

We feel that SVM is a suitable model as it tries to maximize the margin of separation. The training data samples lying on the hyperplane are the support vectors.

We ran a 3 fold cross-validation GridSearchCV and we set the class weights as balanced so that it adjusts the weights inversely proportional to the frequencies of default. From the mean_test_score, we found that the SVM model performs the best when the kernel is linear (score is 0.712). We will specify the kernel as linear when we use it to train our training set.

After training the model, we generate a list of y_predictions based on testing the model using the test set.

Confusion Matrix

		Prediction	
Actual		No Default	Default
	No Default	5029	860
	Default	793	818

Logistic Regression

Logistic regression is frequently used in classification problems and it tries to determine the probability of the customer defaulting.

We ran a 10 fold cross-validation GridSearchCV and we found that the solver newton-cg for logistic regression performed the best(score is 0.721).
After training the model, we generate a list of y_predictions based on testing the model using the test set.
<u>Confusion Matrix</u>

		Prediction	
Actual		No Default	Default
	No Default	5443	446
	Default	1106	505

<u>Evaluation Metrics for Support Vector Machine</u>

Precision	0.48748510131108463
Recall	0.5077591558038486
Accuracy	0.7796
f1-score	0.4974156278504105
misclassification rate	0.2204
specificity	0.8539650195279335

Evaluation Metrics for Logistic Regression

Precision	0.5310199789695058
Recall	0.31346989447548107
Accuracy	0.7930666666666667
f1-score	0.3942232630757221
misclassification rate	0.20693333333333333
specificity	0.9242655798947189

Based on the evaluation metrics, our group has observed that the SVM has a higher recall, f1-score than the logistic regression model while the logistic regression has a higher precision, accuracy and specificity. The SVM also has a slightly higher misclassification rate as compared to the logistic regression. If we use accuracy as our metric to decide which model is better, then the logistic regression will be better than the SVM. However, we feel that predicting customers who will default is more important, we use the recall of the two models to compare. Since the support vector machine has a higher recall, we feel that the support vector machine is a better model for this problem

Room for improvement

The classifiers that we chose for our problem are SVM and Logistic Regression. However, there are other classifiers such as decision tree classifiers. We will get a more comprehensive view of which is the best model for the problem if we tried out other classifiers.

If we know what is the cost of wrongly classifying a customer, we may be able to set the penalty more accurately and overall improve the profits of the company.

Project code

November 16, 2022

[1]:
```
import pandas as pd
import random
import math
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
import scipy
from scipy.stats import skew
from scipy.stats import kurtosis
from scipy.stats import chi2_contingency
import numpy as np
%matplotlib inline
```

1 Introduction of dataset and data modelling problem

The dataset contains information about customers in Taiwan from April 2005 to September 2005 and their default status for the next month. Hence, our group is interested to find out how well are we able to accurately predict the customer's default status based on the attributes of the customers using a support vector machine model. To find out attributes that we feel would be relavant in predicting the default status of the model, we will conduct an exploratory data analysis first.

[2]:
```
df = pd.read_csv("card.csv", sep = ",", skiprows = 1)
```

[3]:
```
df
```

[3]:
```
          ID  LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_0  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

[30000 rows x 25 columns]
```

[4]:
```python
random.seed(1234)
n = len(df.index)
index = list(range(0,n))
testindex = random.sample(index, math.trunc(n / 4))
trainindex = [x for x in index if x not in testindex]
test_data = df.loc[df.index[testindex]]
train_data = df.loc[df.index[trainindex]]
default_df = df[df["default payment next month"] == 1]
```

2 Exploratory Data Analysis

2.1 Checking for missing values

[5]:
```python
df.isnull().any()
```

[5]:
```
ID                            False
LIMIT_BAL                     False
SEX                           False
```

```
EDUCATION                     False
MARRIAGE                      False
AGE                           False
PAY_0                         False
PAY_2                         False
PAY_3                         False
PAY_4                         False
PAY_5                         False
PAY_6                         False
BILL_AMT1                     False
BILL_AMT2                     False
BILL_AMT3                     False
BILL_AMT4                     False
BILL_AMT5                     False
BILL_AMT6                     False
PAY_AMT1                      False
PAY_AMT2                      False
PAY_AMT3                      False
PAY_AMT4                      False
PAY_AMT5                      False
PAY_AMT6                      False
default payment next month    False
dtype: bool
```

[6]:
```
df.isnull().any().sum()
```

[6]:
```
0
```

After checking for null values in our dataset, we found that there are no missing values within our data set.

2.2 Checking to see how many individuals will default the next month

[7]:
```
df["default payment next month"].value_counts()
```

[7]:
```
0    23364
1     6636
Name: default payment next month, dtype: int64
```

[8]:
```
default_perc = 6636 / 30000
non_default_perc = 1 - default_perc
print("default_perc :", default_perc)
print("non-default_perc :", non_default_perc)
```

```
default_perc : 0.2212
non-default_perc : 0.7787999999999999
```

In the entire dataset, 22.1% of the customers default while 77.8% of the customers did not default.

2.3 Feature columns in the dataset

There are 24 feature columns in the dataset.

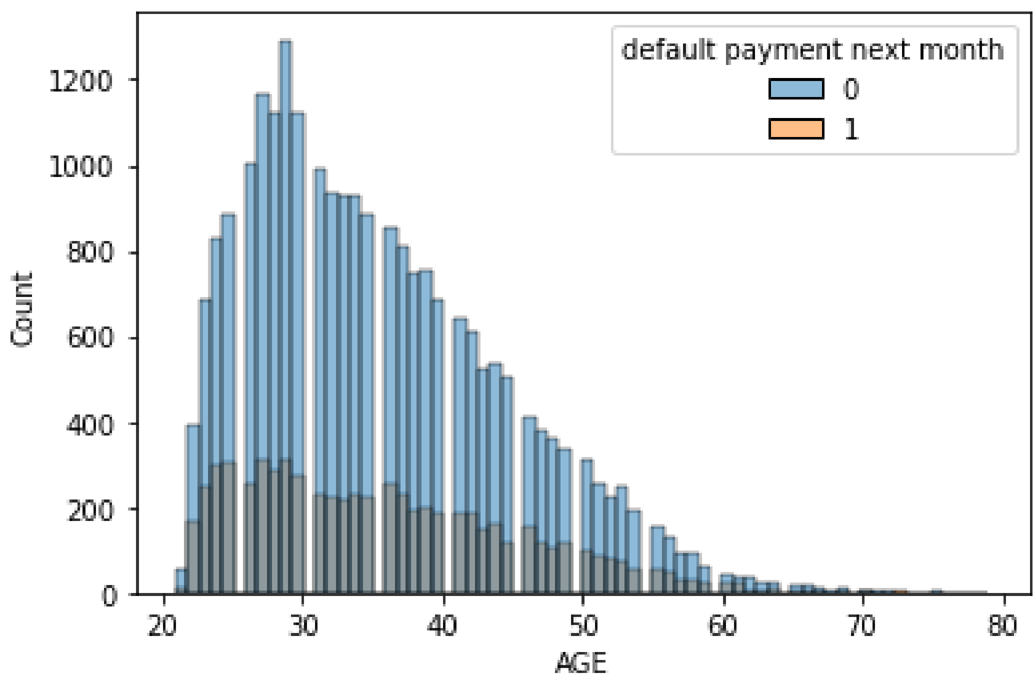
2.4 Summary Statistics for Age

[9]:
```
df["AGE"].describe()
```

[9]:
```
count    30000.000000
mean        35.485500
std          9.217904
min         21.000000
25%         28.000000
50%         34.000000
75%         41.000000
max         79.000000
Name: AGE, dtype: float64
```

[10]:
```
sns.histplot(data=df, x="AGE", hue="default payment next month")
```

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



From the histogram, it seems that there is a greater percentage of indivuals belonging to the age group of 20-40 to default. We will consider performing data pre processing on this category for the data to be easier to understand.

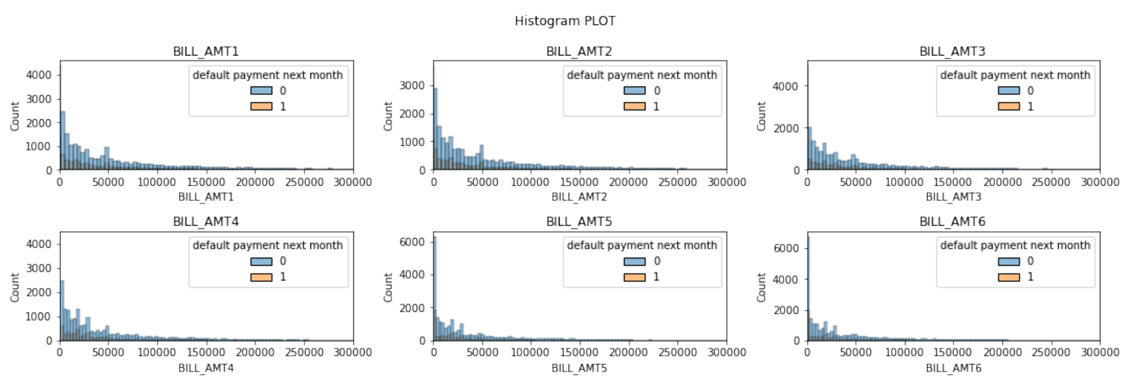
[11]:
```python
print(kurtosis(df["AGE"], fisher=False))
print(skew(df["AGE"], bias=False))
```

```
3.044096001350455
0.7322458687830563
```

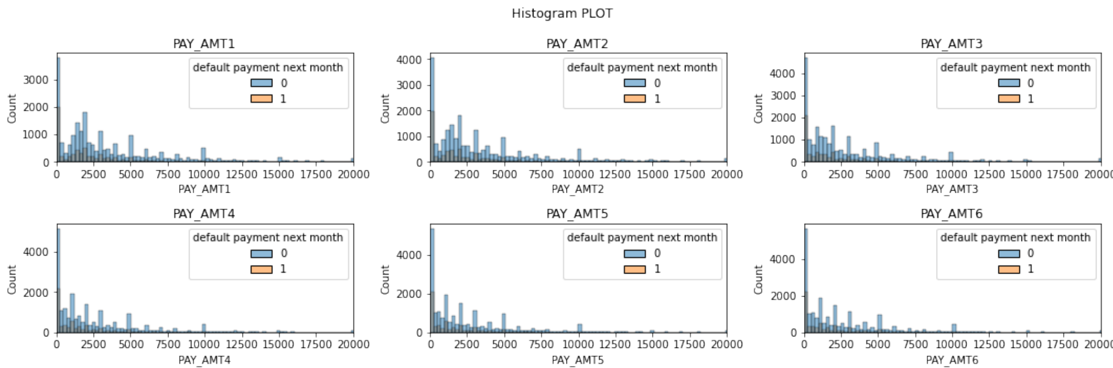
Kurtosis is greater than 3 so the distribution of AGE is leptokurtic Skew is greater than 0 so the distribution of AGE has a slightly thicker right tail

[12]:
```python
def draw_histplots(df, variables, rows, columns, max = 0):
    fig= plt.figure(figsize=(15, 5))
    fig.suptitle("Histogram PLOT")
    for i, cat_name in enumerate(variables):
        if (cat_name == "default payment next month"):
            break
        ax=fig.add_subplot(rows,columns,i+1)
        ax.set_xlim(0,max)
        sns.histplot(ax= ax, x= cat_name, hue="default payment next month",␣
 ↪data= df)
        ax.set_title(cat_name)
    fig.tight_layout()
    plt.show()
```

[13]:
```python
bill_df = df[["BILL_AMT1", "BILL_AMT2", "BILL_AMT3", "BILL_AMT4", "BILL_AMT5",␣
 ↪"BILL_AMT6", "default payment next month"]]
draw_histplots(bill_df, bill_df.columns, 2,3, max = 300_000)
```



[14]:
```python
payamt_df = df[["PAY_AMT1", "PAY_AMT2", "PAY_AMT3", "PAY_AMT4", "PAY_AMT5",␣
 ↪"PAY_AMT6", "default payment next month"]]
draw_histplots(payamt_df, payamt_df.columns, 2,3, 20000)
```



[15]:
```
df["LIMIT_BAL"].describe()
```

[15]:
```
count      30000.000000
mean      167484.322667
std       129747.661567
min        10000.000000
25%        50000.000000
50%       140000.000000
75%       240000.000000
max      1000000.000000
Name: LIMIT_BAL, dtype: float64
```

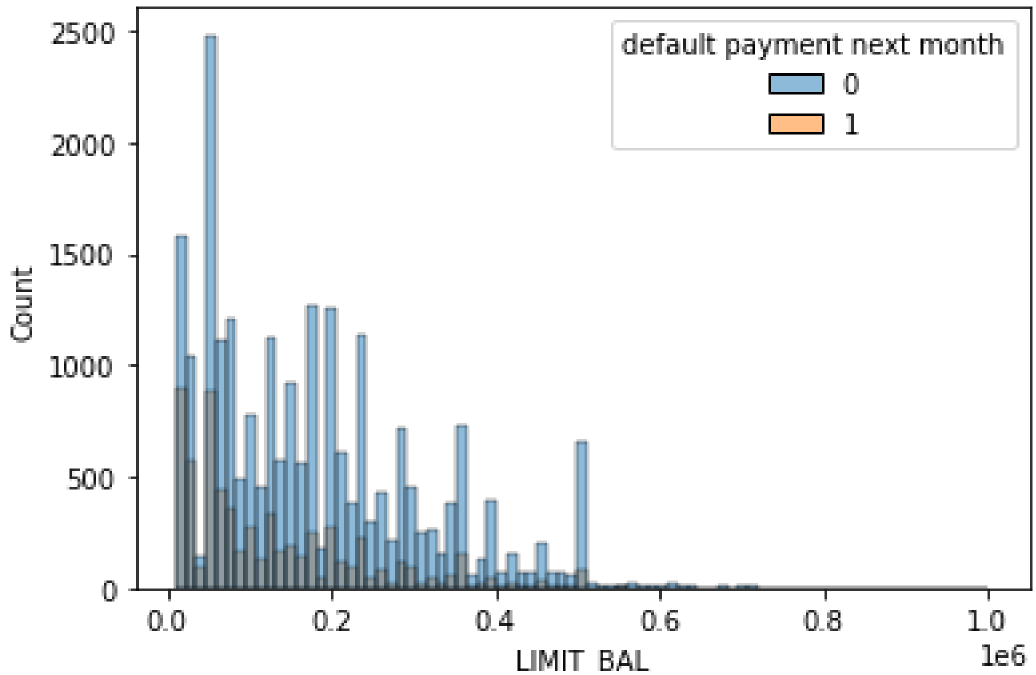
[16]:
```
print(kurtosis(df["LIMIT_BAL"], fisher=False))
print(skew(df["LIMIT_BAL"], bias = False))
```

```
3.5359735300865474
0.9928669605195439
```

Kurtosis is greater than 3 so the distribution of LIMIT_BAL is leptokurtic Skew is greater than 0 so the distribution of LIMIT_BAL has a slightly thicker right tail

[17]:
```
sns.histplot(df,x = "LIMIT_BAL", hue = "default payment next month")
```

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



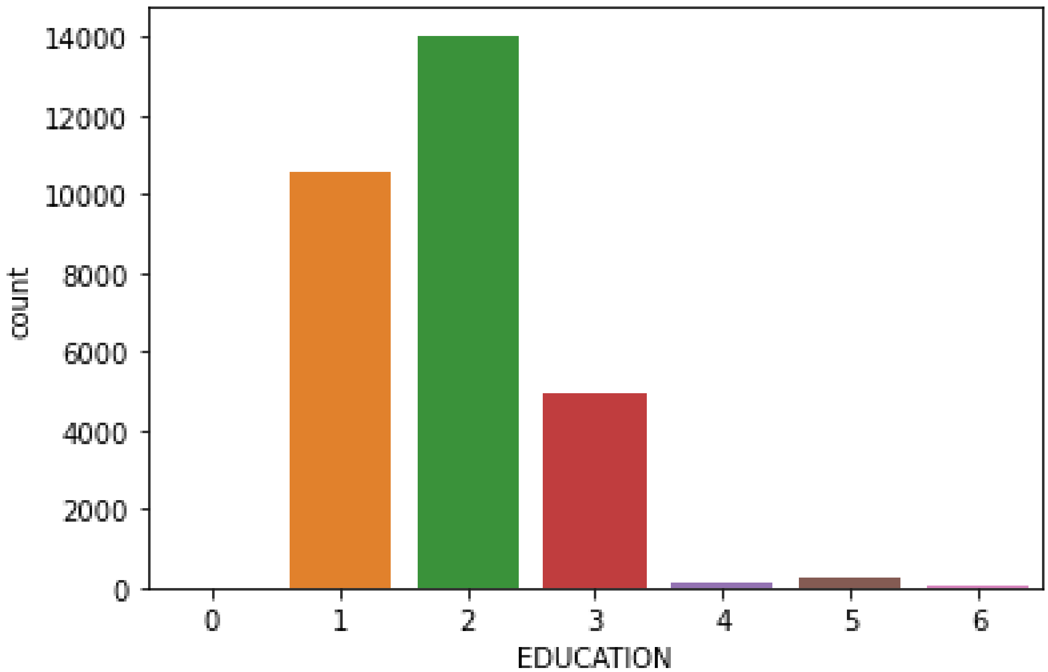
From the histogram, we can see that more people default when their limit_bal is between 0 and 0.2 * 10^6. We will conduct data discretization to reduce the number of distinct values assumed by categorical variables.

2.5 Anomalies and Inconsistencies in the dataset

2.5.1 Anomalies in dataset

[18]:
```
df["EDUCATION"].astype("category").nunique()
df['EDUCATION'] = df['EDUCATION'].astype('category')
sns.countplot(data = df, x= "EDUCATION")
```

[18]:
```
<AxesSubplot:xlabel='EDUCATION', ylabel='count'>
```



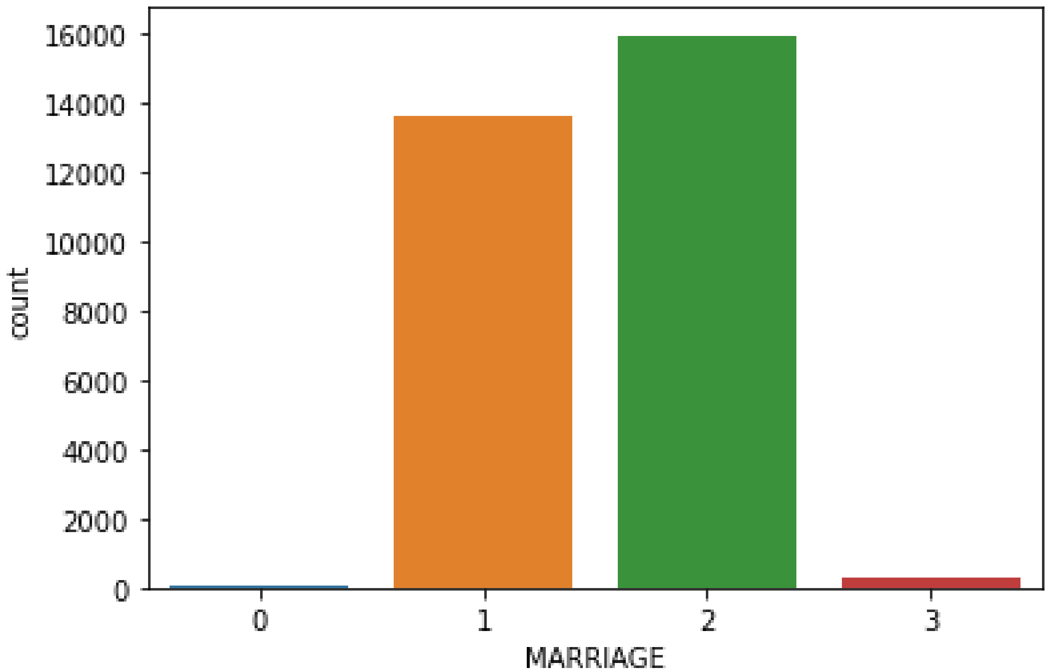
[19]:
```
len(df[df['EDUCATION'] == 0 ])
```

[19]: 14

There are 14 rows of data that have EDUCATION coded as 0 which was not explained by the author. Hence, we do not know whether the extra values was a typographical error or it means another form of education level that they author did not explain.

[20]:
```
df['MARRIAGE'] = df['MARRIAGE'].astype('category')
sns.countplot(data = df, x= "MARRIAGE")
```

[20]: <AxesSubplot:xlabel='MARRIAGE', ylabel='count'>



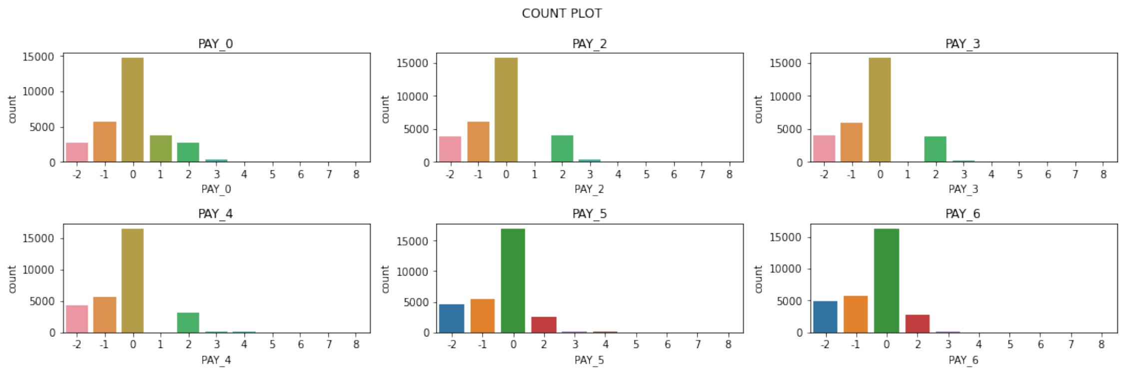
[21]:
```
df["MARRIAGE"].value_counts()
```

[21]:
```
2    15964
1    13659
3      323
0       54
Name: MARRIAGE, dtype: int64
```

Similarly, the MARRIAGE variable also has an extra possible value that is 0 which was also not explained by the author. There are 54 rows of data that have MARRIAGE encoded as 0.

[22]:
```
def draw_countplots(df, variables, rows, columns):
    fig= plt.figure(figsize=(15, 5))
    fig.suptitle("COUNT PLOT")
    for i, cat_name in enumerate(variables):
        ax=fig.add_subplot(rows,columns,i+1)
        sns.countplot(ax= ax, x= cat_name, data= df)
        ax.set_title(cat_name)
    fig.tight_layout()
    plt.show()
```

[23]:
```
pay_df = df[["PAY_0", "PAY_2", "PAY_3", "PAY_4", "PAY_5", "PAY_6"]]
draw_countplots(pay_df, pay_df.columns, 2,3)
```



[24]:
```
len(df[df["PAY_0"] == 0])
```

[24]: 14737

[25]:
```
len(df[df["PAY_0"] == 0]) / len(df)
```

[25]: 0.49123333333333336

We can observe that there are extra values that the repayment status in our dataset that can take on. Specifically, we see that the repayment status can be 0 or -2 which was not explained in the dataset description. Furthermore, there are many rows that have these extra values. For example in PAY_0 alone there are already 14737 rows of data that have PAY_0 being encoded as 0 which is already about 50% of the entire dataset. Therefore, we should not filter out all the data that have these extra values encoded since that would mean that our the size of our dataset will reduce dramatically. Given that there are many rows of data that contains values that were not explained, we need to perform data cleaning and pre-processing later before starting to train our model.

[26]:
```
df["SEX"].astype("category").value_counts()
```

[26]:
```
2    18112
1    11888
Name: SEX, dtype: int64
```

No unexplained encodings observed for "Sex"

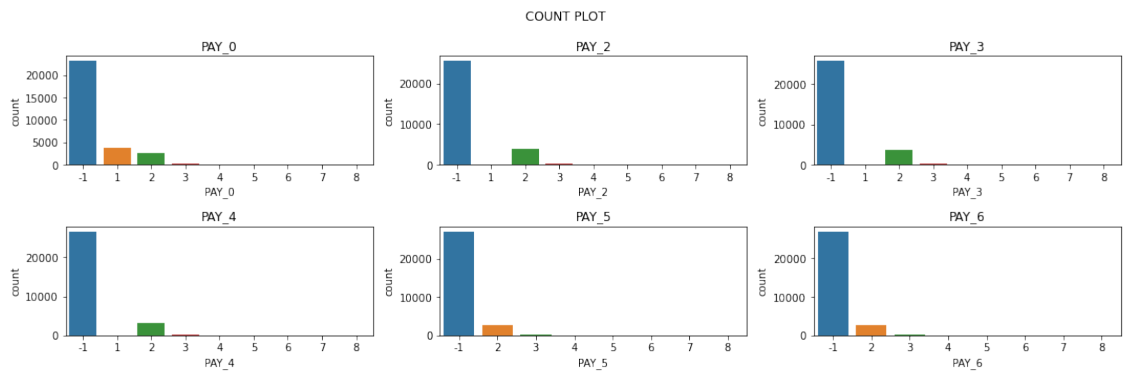
2.6 Data cleaning and preprocessing

We have seen from our exploratory data analysis that there are several values that were not explained and many rows in the dataset contain such values. Hence, we decide to see if we are able to encode these values to another value that makes sense for our dataset.

2.6.1 Repayment Status

Since the value in repayment status represents the number of months for payment delay, our group feels that 0 and -2 should also means that there 0 months in payment delay. Hence, our group decides to enocde 0 and -2 to -1 which represents duly payments.

[27]:
```
repayment_list = [df.PAY_0,df.PAY_2,df.PAY_3,df.PAY_4,df.PAY_5,df.PAY_6]
filter_list = []
names = ["PAY_0","PAY_2","PAY_3","PAY_4","PAY_5","PAY_6"]
for r in repayment_list:
    filter_list.append((r == -2) | (r  == -1) | (r == 0))
i = 0
for filter_obj in filter_list:
    df.loc[filter_obj, names[i]] = -1
    i+=1
pay_df = df[["PAY_0", "PAY_2", "PAY_3", "PAY_4", "PAY_5", "PAY_6"]]
draw_countplots(pay_df, pay_df.columns, 2,3)
```



[28]:
```
df["LATE"] = df["PAY_0"] > -1
```

[29]:
```
df["LATE"] = df["LATE"].astype(int)
```

Our group decides to create a feature labelled as "LATE" to separate customers into two groups for simpler understanding. The two main groups are individuals who have been late in their payment before and those who have never been late before. We feel that there is a greater tendency for individuals who have been late in their payments to default as they may be experiencing some financial difficulties in their life.

2.6.2 Education and Marriage

[30]:
```
filtered_education = (df.EDUCATION == 0)| (df.EDUCATION == 6) | (df.EDUCATION␣
 ↪== 5)
df.loc[filtered_education, 'EDUCATION'] = 4
filtered_marriage = (df.MARRIAGE == 0)
df.loc[filtered_marriage, 'MARRIAGE'] = 3
```

Since the value 5 and 6 for Education is unknown and value of 0 for Education is not explained, our group decides to group all of them under value 4 which is 'others'. For the value 0 under Marriage, we decide to group them under value 3 which is "others".

2.6.3 Age

[31]:
```
def conditions_age(df):
    if (df['AGE'] > 20) and (df["AGE"] <= 40):
        return 1
    elif (df["AGE"] > 40 and df["AGE"] <= 60):
        return 2
    else:
        return 3

df["NEW_AGE"] = df.apply(conditions_age, axis=1)
```

Age > 20 and Age <= 40 : 1 Age > 40 and Age <= 60 : 2 Age > 60 3 this is for data discretization

2.6.4 Limit Bal

[32]:
```
def conditions_limit(df):
    if (df['LIMIT_BAL'] > 0) and (df["LIMIT_BAL"] <= 200_000):
        return 1
    elif (df["LIMIT_BAL"] > 200_000 and df["LIMIT_BAL"] <= 400_000):
        return 2
    else:
        return 3

df["NEW_LIMIT_BAL"] = df.apply(conditions_limit, axis=1)
```

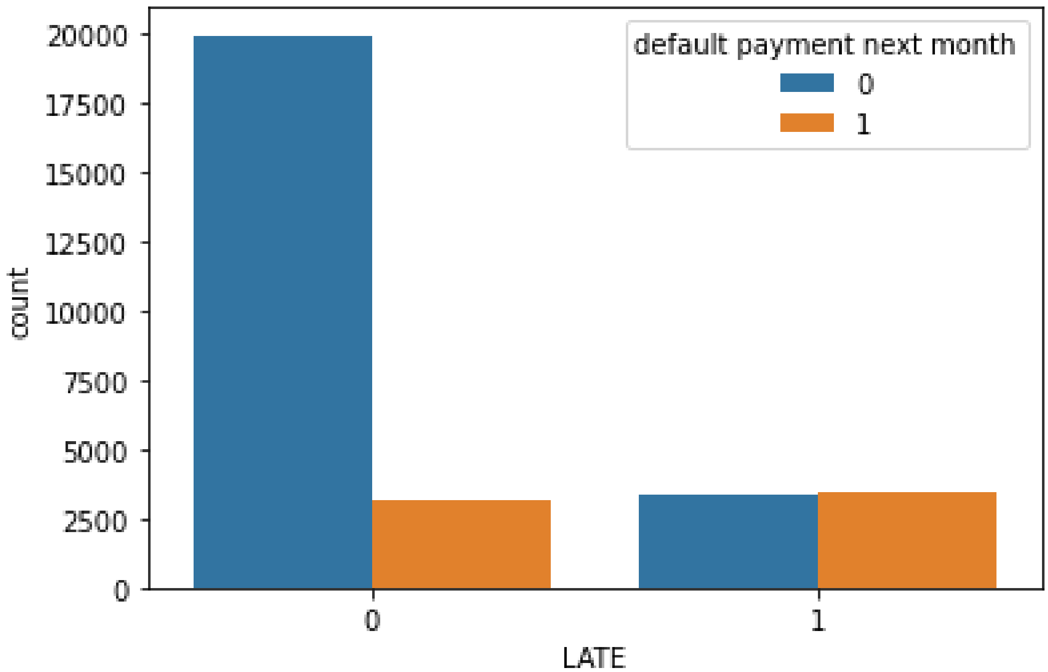
LIMIT_BAL > 0 and LIMIT_BAL <= 200 000 : 1 LB > 200 000 and LB <= 400 000 : 2 LB > 400 000 3 same as age

2.7 Checking for trends after data preprocessing

2.7.1 Countplot for individuals who have been late in their payment before

[33]:
```
sns.countplot(data = df, x = "LATE", hue = "default payment next month")
```

[33]:
```
<AxesSubplot:xlabel='LATE', ylabel='count'>
```



[34]:
```
# late is 0 and defaults
LATE_0_defaults = len(df[(df["LATE"] == 0) & (df["default payment next month"]␣
↪== 1)])
LATE_0_defaults_perc = LATE_0_defaults / len(df[(df["LATE"] == 0)])
print(LATE_0_defaults_perc)


# late is 1 and defaults
LATE_1_defaults = len(df[(df["LATE"] == 1) & (df["default payment next month"]␣
↪== 1)])
LATE_1_defaults_perc = LATE_1_defaults / len(df[(df["LATE"] == 1)])
print(LATE_1_defaults_perc)
```

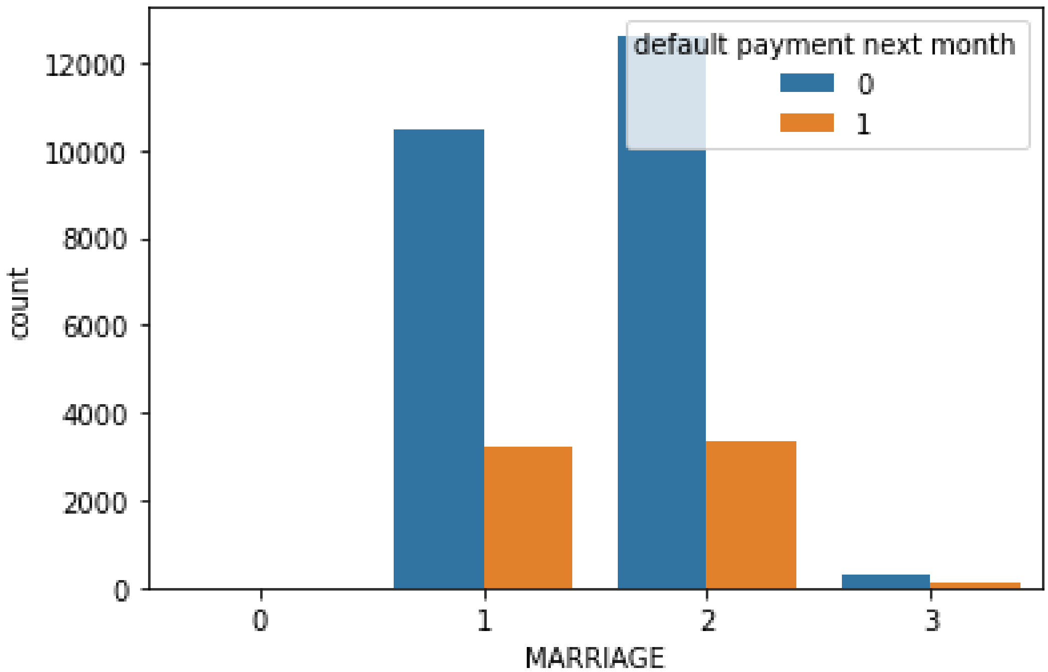
```
0.13834009145026313
0.5029334115576415
```

From the graph we can see that a greater percentage of those who have been late in their payments before to default in the next month. It seems that this feature will useful in predicting whether customer will default.

2.7.2 Countplot for individuals based on their marriage status

[35]:
```
sns.countplot(data = df, x = "MARRIAGE", hue = "default payment next month")
```

[35]:
```
<AxesSubplot:xlabel='MARRIAGE', ylabel='count'>
```



[36]:

```
# marriage is 1 and defaults
MARRIAGE_1_defaults = len(df[(df["MARRIAGE"] == 1) & (df["default payment next␣
↪month"] == 1)])
MARRIAGE_1_defaults_perc = MARRIAGE_1_defaults / len(df[(df["MARRIAGE"] == 1)])
print(MARRIAGE_1_defaults_perc)


# marriage is 2 and defaults
MARRIAGE_2_defaults = len(df[(df["MARRIAGE"] == 2) & (df["default payment next␣
↪month"] == 1)])
MARRIAGE_2_defaults_perc = MARRIAGE_2_defaults / len(df[(df["MARRIAGE"] == 2)])
print(MARRIAGE_2_defaults_perc)

# marriage is 3 and defaults
MARRIAGE_3_defaults = len(df[(df["MARRIAGE"] == 3) & (df["default payment next␣
↪month"] == 1)])
MARRIAGE_3_defaults_perc = MARRIAGE_3_defaults / len(df[(df["MARRIAGE"] == 3)])
print(MARRIAGE_3_defaults_perc)
```

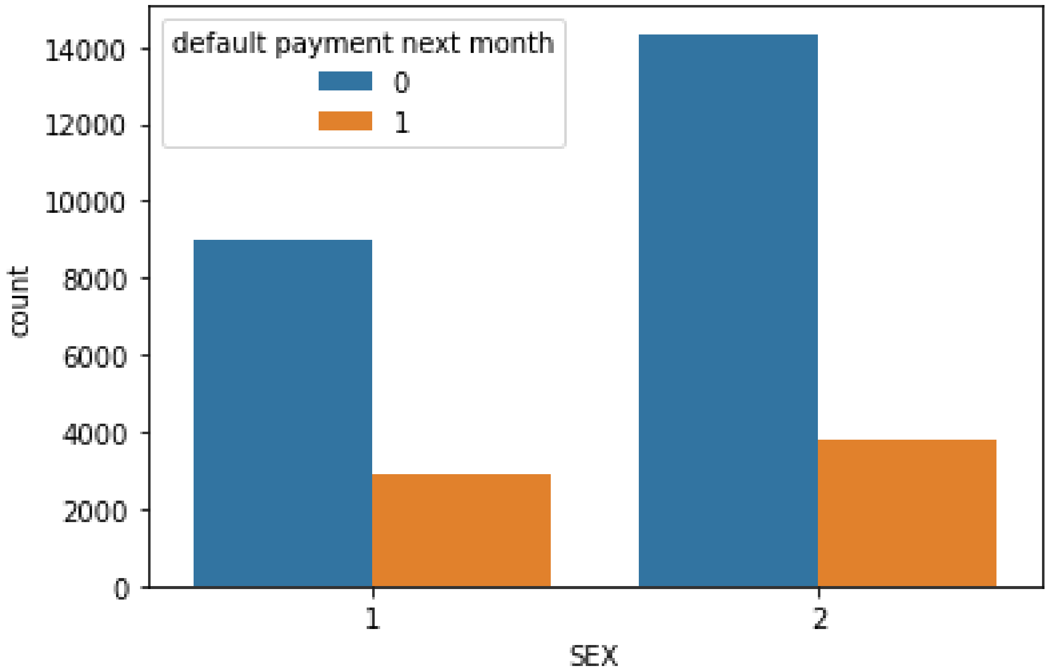
```
0.23471703638626545
0.20928338762214985
0.23607427055702918
```

Not much differences

2.7.3 Countplot for individuals based on their sex

[37]:
```
sns.countplot(data = df, x = "SEX", hue = "default payment next month")
```

[37]: <AxesSubplot:xlabel='SEX', ylabel='count'>



[38]:
```
# sex is male and defaults
sex_1_defaults = len(df[(df["SEX"] == 1) & (df["default payment next month"] ==␣
↪1)])
sex_1_defaults_perc = sex_1_defaults / len(df[(df["SEX"] == 1)])
print(sex_1_defaults_perc)

# sex is female and defaults
sex_2_defaults = len(df[(df["SEX"] == 2) & (df["default payment next month"] ==␣
↪1)])
sex_2_defaults_perc = sex_2_defaults / len(df[(df["SEX"] == 2)])
print(sex_2_defaults_perc)
```

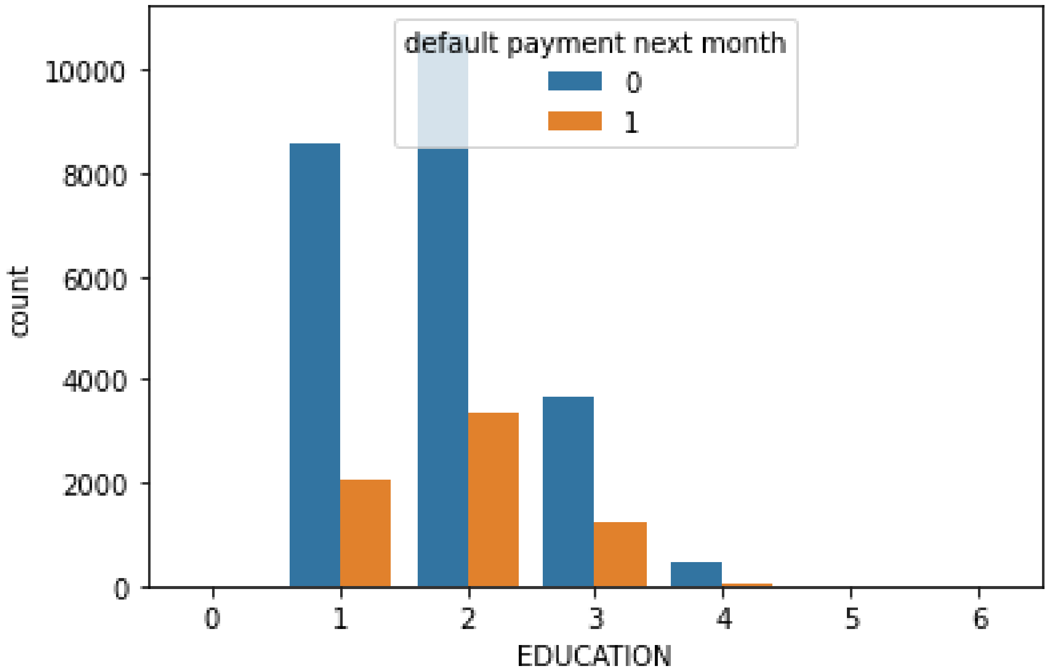
```
0.2416722745625841
0.20776280918727916
```

Overall, it seems that individuals who are male are more likely to default than females. Hence, the gender of the customer could be useful in predicting customer default status.

2.7.4 Countplot for individuals based on their education

[39]:
```
sns.countplot(data = df, x = "EDUCATION", hue = "default payment next month")
```

[39]: <AxesSubplot:xlabel='EDUCATION', ylabel='count'>



[40]:
```
# education is 1 and defaults
edu_1_defaults = len(df[(df["EDUCATION"] == 1) & (df["default payment next␣
 ↪month"] == 1)])
edu_1_defaults_perc = edu_1_defaults / len(df[(df["EDUCATION"] == 1)])
print(edu_1_defaults_perc)


# education is 2 and defaults
edu_2_defaults = len(df[(df["EDUCATION"] == 2) & (df["default payment next␣
 ↪month"] == 1)])
edu_2_defaults_perc = edu_2_defaults / len(df[(df["EDUCATION"] == 2)])
print(edu_2_defaults_perc)

# education is 3 and defaults
edu_3_defaults = len(df[(df["EDUCATION"] == 3) & (df["default payment next␣
 ↪month"] == 1)])
edu_3_defaults_perc = edu_3_defaults / len(df[(df["EDUCATION"] == 3)])
print(edu_3_defaults_perc)
```

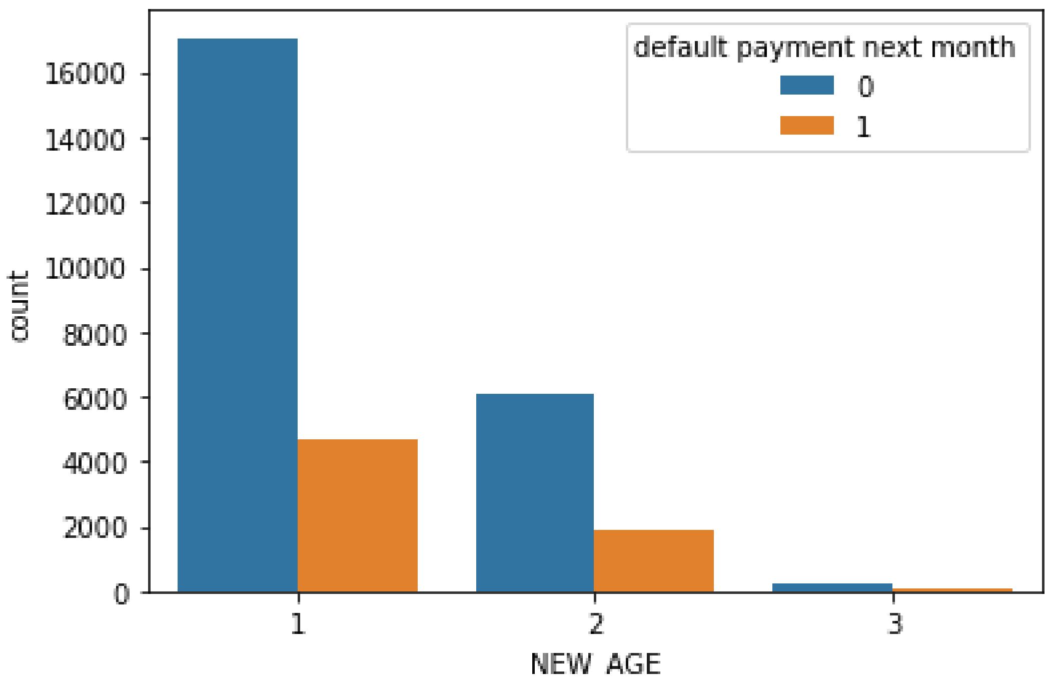
```
# education is 4 and defaults
edu_4_defaults = len(df[(df["EDUCATION"] == 4) & (df["default payment next␣
 ↪month"] == 1)])
edu_4_defaults_perc = edu_4_defaults / len(df[(df["EDUCATION"] == 4)])
print(edu_4_defaults_perc)
```

```
0.19234766178554558
0.23734853884533144
0.2515761643278422
0.07051282051282051
```

From the graph and based on our further calculations, it seems that there is a higher percentage of individuals with an education level of high school to default. This feature could also be useful for our model.

[41]:
```
sns.countplot(data = df, x = "NEW_AGE", hue = "default payment next month")
```

[41]: <AxesSubplot:xlabel='NEW_AGE', ylabel='count'>



[42]:
```
# AGE is 1 and defaults
age_1_defaults = len(df[(df["NEW_AGE"] == 1) & (df["default payment next␣
 ↪month"] == 1)])
age_1_defaults_perc = age_1_defaults / len(df[(df["NEW_AGE"] == 1)])
print(age_1_defaults_perc)
```

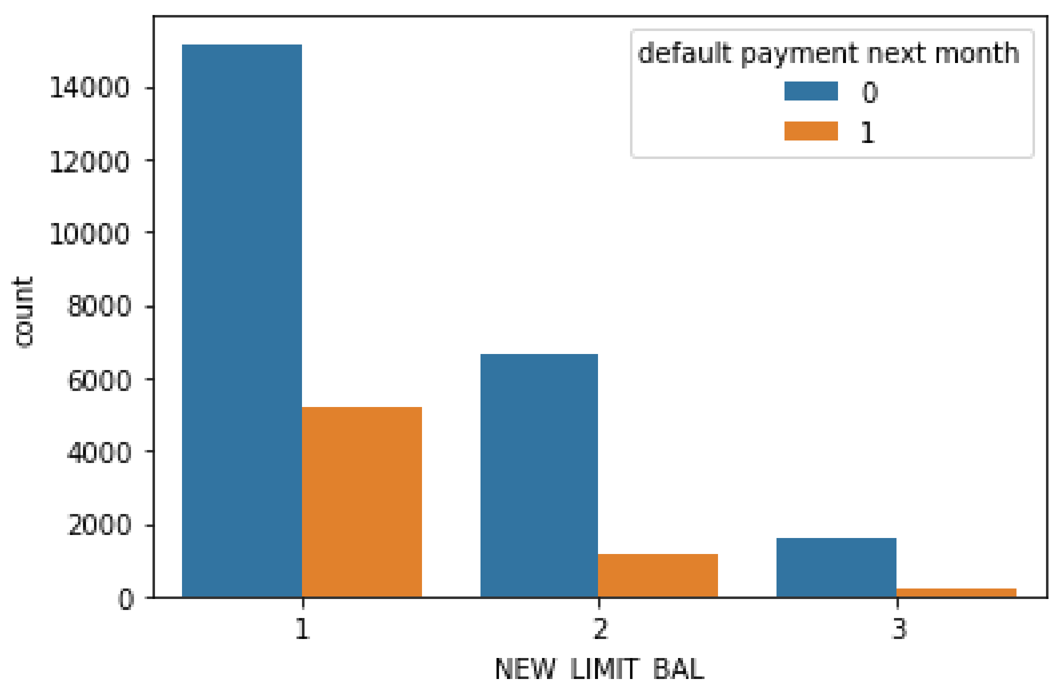
```
# AGE is 2 and defaults
age_2_defaults = len(df[(df["NEW_AGE"] == 2) & (df["default payment next␣
↪month"] == 1)])
age_2_defaults_perc = age_2_defaults / len(df[(df["NEW_AGE"] == 2)])
print(age_2_defaults_perc)

# AGE is 3 and defaults
age_3_defaults = len(df[(df["NEW_AGE"] == 3) & (df["default payment next␣
↪month"] == 1)])
age_3_defaults_perc = age_3_defaults / len(df[(df["NEW_AGE"] == 3)])
print(age_3_defaults_perc)
```

```
0.2144895516892203
0.23781554611347164
0.26838235294117646
```

percentage is quite similar for all age groups.

[43]:
```
sns.countplot(data = df, x = "NEW_LIMIT_BAL", hue = "default payment next␣
↪month")
```

[43]: `<AxesSubplot:xlabel='NEW_LIMIT_BAL', ylabel='count'>`



[44]:
```
# NLB is 1 and defaults
nlb_1_defaults = len(df[(df["NEW_LIMIT_BAL"] == 1) & (df["default payment next␣
↪month"] == 1)])
nlb_1_defaults_perc = nlb_1_defaults / len(df[(df["NEW_LIMIT_BAL"] == 1)])
print(nlb_1_defaults_perc)


# NLB is 2 and defaults
nlb_2_defaults = len(df[(df["NEW_LIMIT_BAL"] == 2) & (df["default payment next␣
↪month"] == 1)])
nlb_2_defaults_perc = nlb_2_defaults / len(df[(df["NEW_LIMIT_BAL"] == 2)])
print(nlb_2_defaults_perc)

# NLB is 3 and defaults
nlb_3_defaults = len(df[(df["NEW_LIMIT_BAL"] == 3) & (df["default payment next␣
↪month"] == 1)])
nlb_3_defaults_perc = nlb_3_defaults / len(df[(df["NEW_LIMIT_BAL"] == 3)])
print(nlb_3_defaults_perc)
```

```
0.2561095298851703
0.15349194167306215
0.12028824833702882
```

People who have limit balance of 0 to 200 000 have a much higher percentage of default compared to other groups.

2.8 Feature Selection

[45]:
```
# SEX
print(df["SEX"].value_counts())
print(df[(df["SEX"] == 1) & (df["default payment next month"] == 1)]["SEX"].
↪value_counts())
print(df[(df["SEX"] == 2) & (df["default payment next month"] == 1)]["SEX"].
↪value_counts())
print(df[(df["SEX"] == 1) & (df["default payment next month"] == 0)]["SEX"].
↪value_counts())
print(df[(df["SEX"] == 2) & (df["default payment next month"] == 0)]["SEX"].
↪value_counts())
```

```
2    18112
1    11888
Name: SEX, dtype: int64
1    2873
Name: SEX, dtype: int64
2    3763
Name: SEX, dtype: int64
1    9015
Name: SEX, dtype: int64
```

```
2    14349
Name: SEX, dtype: int64
```

2873 -> SEX = 1 and default 3763 -> SEX = 2 and default 9015 -> SEX = 1 and no default 14349 -> SEX = 2 and no default

[46]:
```python
tab_data = [[2873, 3763], [9015, 14349]]
chi2_contingency(tab_data)
```

[46]:
```
(47.70879689062111,
 4.944678999412044e-12,
 1,
 array([[ 2629.6256,  4006.3744],
        [ 9258.3744, 14105.6256]]))
```

[47]:
```python
# EDUCATION
print(df["EDUCATION"].value_counts())
print(len(df[(df["EDUCATION"] == 1) & (df["default payment next month"] == 1)]))
print(len(df[(df["EDUCATION"] == 2) & (df["default payment next month"] == 1)]))
print(len(df[(df["EDUCATION"] == 3) & (df["default payment next month"] == 1)]))
print(len(df[(df["EDUCATION"] == 4) & (df["default payment next month"] == 1)]))


print(len(df[(df["EDUCATION"] == 1) & (df["default payment next month"] == 0)]))
print(len(df[(df["EDUCATION"] == 2) & (df["default payment next month"] == 0)]))
print(len(df[(df["EDUCATION"] == 3) & (df["default payment next month"] == 0)]))
print(len(df[(df["EDUCATION"] == 4) & (df["default payment next month"] == 0)]))
```

```
2    14030
1    10585
3     4917
4      468
0        0
5        0
6        0
Name: EDUCATION, dtype: int64
2036
3330
1237
33
8549
10700
3680
435
```

[48]:
```python
tab_data_EDUCATION = [[2036,3330,1237,33], [8549,10700,3680,435]]
chi2, p, dof, ex =  chi2_contingency(tab_data_EDUCATION)
p
```

[48]: 1.495064564810615e-34

[49]:
```
# MARRIAGE
print(df["MARRIAGE"].value_counts())
print(len(df[(df["MARRIAGE"] == 1) & (df["default payment next month"] == 1)]))
print(len(df[(df["MARRIAGE"] == 2) & (df["default payment next month"] == 1)]))
print(len(df[(df["MARRIAGE"] == 3) & (df["default payment next month"] == 1)]))

print(len(df[(df["MARRIAGE"] == 1) & (df["default payment next month"] == 0)]))
print(len(df[(df["MARRIAGE"] == 2) & (df["default payment next month"] == 0)]))
print(len(df[(df["MARRIAGE"] == 3) & (df["default payment next month"] == 0)]))
```

```
2    15964
1    13659
3      377
0        0
Name: MARRIAGE, dtype: int64
3206
3341
89
10453
12623
288
```

[50]:
```
tab_data_MARRIAGE = [[3206,3341,89], [10453,12623,288]]
chi2, p, dof, ex =  chi2_contingency(tab_data_MARRIAGE)
p
```

[50]: 7.790720364202813e-07

[51]:
```
# NEW_LIMIT_BAL
print(df["NEW_LIMIT_BAL"].value_counts())
print(len(df[(df["NEW_LIMIT_BAL"] == 1) & (df["default payment next month"] ==␣
 ↪1)]))
print(len(df[(df["NEW_LIMIT_BAL"] == 2) & (df["default payment next month"] ==␣
 ↪1)]))
print(len(df[(df["NEW_LIMIT_BAL"] == 3) & (df["default payment next month"] ==␣
 ↪1)]))

print(len(df[(df["NEW_LIMIT_BAL"] == 1) & (df["default payment next month"] ==␣
 ↪0)]))
print(len(df[(df["NEW_LIMIT_BAL"] == 2) & (df["default payment next month"] ==␣
 ↪0)]))
print(len(df[(df["NEW_LIMIT_BAL"] == 3) & (df["default payment next month"] ==␣
 ↪0)]))
```

```
1    20378
2     7818
```

```
3     1804
Name: NEW_LIMIT_BAL, dtype: int64
5219
1200
217
15159
6618
1587
```

[52]:
```
tab_data_NLB = [[5219,1200,217], [15159,6618,1587]]
chi2, p, dof, ex =  chi2_contingency(tab_data_NLB)
p
```

[52]: 2.3082716470487475e-100

[53]:
```
# LATE
print(df["LATE"].value_counts())
print(len(df[(df["LATE"] == 0) & (df["default payment next month"] == 1)]))
print(len(df[(df["LATE"] == 1) & (df["default payment next month"] == 1)]))

print(len(df[(df["LATE"] == 0) & (df["default payment next month"] == 0)]))
print(len(df[(df["LATE"] == 1) & (df["default payment next month"] == 0)]))
```

```
0    23182
1     6818
Name: LATE, dtype: int64
3207
3429
19975
3389
```

[54]:
```
tab_data_LATE = [[3207,3429], [19975,3389]]
chi2, p, dof, ex = chi2_contingency(tab_data_LATE)
p
```

[54]: 0.0

[55]:
```
# NEW_AGE
print(df["NEW_AGE"].value_counts())
print(len(df[(df["NEW_AGE"] == 1) & (df["default payment next month"] == 1)]))
print(len(df[(df["NEW_AGE"] == 2) & (df["default payment next month"] == 1)]))
print(len(df[(df["NEW_AGE"] == 3) & (df["default payment next month"] == 1)]))


print(len(df[(df["NEW_AGE"] == 1) & (df["default payment next month"] == 0)]))
print(len(df[(df["NEW_AGE"] == 2) & (df["default payment next month"] == 0)]))
print(len(df[(df["NEW_AGE"] == 3) & (df["default payment next month"] == 0)]))
```

```
1    21726
```

```
2     8002
3      272
Name: NEW_AGE, dtype: int64
4660
1903
73
17066
6099
199
```

[56]:
```
tab_data_NEW_AGE = [[4660,1903,73], [17066,6099,199]]
chi2, p, dof, ex = chi2_contingency(tab_data_NEW_AGE)
p
```

[56]:
```
1.6554284303886136e-05
```

2.9 Creating training set and test set for our model

[57]:
```
from sklearn import svm, metrics
from sklearn.model_selection import GridSearchCV, RepeatedStratifiedKFold
from sklearn.linear_model import LogisticRegression
```

[58]:
```
random.seed(1234)
n = len(df.index)
index = list(range(0,n))
testindex = random.sample(index, math.trunc(n / 4))
trainindex = [x for x in index if x not in testindex]
test_data = df.loc[df.index[testindex]]
train_data = df.loc[df.index[trainindex]]
```

[59]:
```
train_x = train_data[["SEX", "LATE", "EDUCATION","NEW_LIMIT_BAL", "NEW_AGE",␣
 ↪"MARRIAGE"]]
train_y = train_data["default payment next month"]
test_x = test_data[["SEX", "LATE","EDUCATION","NEW_LIMIT_BAL",  "NEW_AGE",␣
 ↪"MARRIAGE"]]
test_y = test_data["default payment next month"]
```

2.10 Model Selection

2.11 SVM

[60]:
```
param_grid = {'kernel': ('linear', 'rbf')}
```

[61]:
```
svc = svm.SVC(random_state=0, class_weight= "balanced")
search = GridSearchCV(estimator=svc, param_grid=param_grid, scoring="roc_auc",␣
 ↪cv = 3)
search.fit(train_x, train_y)
```

[61]:
```
GridSearchCV(cv=3, estimator=SVC(class_weight='balanced', random_state=0),
             param_grid={'kernel': ('linear', 'rbf')}, scoring='roc_auc')
```

Setting class_weights = 'balanced', the model assigns the class weights inversely proportional to their respective frequencies.

[62]:
```
results_df = pd.DataFrame(search.cv_results_)
results_df = results_df.sort_values(by=["rank_test_score"])
```

[63]:
```
results_df["mean_test_score"]
```

[63]:
```
0    0.712730
1    0.696309
Name: mean_test_score, dtype: float64
```

[64]:
```
results_df
```

[64]:
```
   mean_fit_time  std_fit_time  mean_score_time  std_score_time param_kernel  \
0       1.718400      0.010447         0.570098        0.008824       linear
1       2.894331      0.150738         4.249627        0.054925          rbf

                 params  split0_test_score  split1_test_score  \
0  {'kernel': 'linear'}           0.697327           0.713879
1     {'kernel': 'rbf'}           0.683023           0.704573

   split2_test_score  mean_test_score  std_test_score  rank_test_score
0           0.726984         0.712730        0.012134                1
1           0.701331         0.696309        0.009487                2
```

Based on the GridSearch result, we can conclude that the best parameter to choose is the linear kernel.

2.11.1 Model evaluation

[65]:
```
clf = svm.SVC(kernel='linear', class_weight='balanced') # Linear Kernel
clf.fit(train_x, train_y)
```

[65]:
```
SVC(class_weight='balanced', kernel='linear')
```

[66]:
```
y_pred = clf.predict(test_x)
```

[67]:
```
confusion_matrix = metrics.confusion_matrix(test_y, y_pred)
tn, fp, fn, tp = confusion_matrix.ravel()
confusion_matrix
```

[67]:
```
array([[5029,  860],
       [ 793,  818]])
```

[68]:
```
# Model Precision
print("Precision: ",metrics.precision_score(test_y, y_pred))

# Model Recall
print("Recall: ",metrics.recall_score(test_y, y_pred))

#Model Accuracy
print("Accuracy: ",metrics.accuracy_score(test_y, y_pred))

#f1-score
print("f1-score: ", metrics.f1_score(test_y, y_pred))

#misclassification rate
misclassification_rate = (fp + fn) / (fp + fn + tp + tn)
print("misclassification rate: ", misclassification_rate)

#sensitivity
sensitivity = (tp) / (fn + tp)
print("sensitivity: ", sensitivity)

#specificity
specificity = (tn) / (tn + fp)
print("specificity: ", specificity)
```

```
Precision:  0.48748510131108463
Recall:  0.5077591558038486
Accuracy:  0.7796
f1-score:  0.4974156278504105
misclassification rate:  0.2204
sensitivity:  0.5077591558038486
specificity:  0.8539650195279335
```

2.12 Logistic Regression

[69]:
```
parameters = {
    'solver'  : ['newton-cg', 'lbfgs', 'liblinear']
}
```

[70]:
```
logreg = LogisticRegression(class_weight = "balanced")
search_logreg = GridSearchCV(logreg,
                   param_grid = parameters,
                   scoring="roc_auc",
                   cv=10)
search_logreg.fit(train_x, train_y)
```

[70]:
```
GridSearchCV(cv=10, estimator=LogisticRegression(class_weight='balanced'),
             param_grid={'solver': ['newton-cg', 'lbfgs', 'liblinear']},
             scoring='roc_auc')
```

[71]:
```
results_logreg_df = pd.DataFrame(search_logreg.cv_results_)
results_logreg_df = results_logreg_df.sort_values(by=["rank_test_score"])
```

[72]:
```
results_logreg_df["mean_test_score"]
results_logreg_df
```

[72]:
```
   mean_fit_time  std_fit_time  mean_score_time  std_score_time param_solver  \
0       0.040238      0.004099         0.001592        0.000848    newton-cg
1       0.027204      0.003500         0.001736        0.000892        lbfgs
2       0.016287      0.002733         0.001294        0.000648    liblinear

                   params  split0_test_score  split1_test_score  \
0  {'solver': 'newton-cg'}           0.700115           0.711203
1      {'solver': 'lbfgs'}           0.700115           0.711203
2  {'solver': 'liblinear'}           0.700110           0.711203

   split2_test_score  split3_test_score  split4_test_score  split5_test_score  \
0           0.710299           0.695224           0.713758           0.743361
1           0.710299           0.695224           0.713758           0.743361
2           0.710299           0.695138           0.713734           0.743361

   split6_test_score  split7_test_score  split8_test_score  split9_test_score  \
0           0.735079           0.732856           0.738173           0.733928
1           0.735079           0.732856           0.738173           0.733928
2           0.735078           0.732856           0.737945           0.733916

   mean_test_score  std_test_score  rank_test_score
0         0.721400        0.016318                1
1         0.721400        0.016318                1
2         0.721364        0.016309                3
```

[73]:
```
logreg = LogisticRegression(solver = "newton-cg")
logreg.fit(train_x,train_y)
y_pred_log = logreg.predict(test_x)
print("Accuracy:",metrics.accuracy_score(test_y, y_pred_log))
```

```
Accuracy: 0.7930666666666667
```

[74]:
```
confusion_matrix = metrics.confusion_matrix(test_y, y_pred_log)
tn, fp, fn, tp = confusion_matrix.ravel()
confusion_matrix
```

[74]:
```
array([[5443,  446],
       [1106,  505]])
```

[75]:
```
# Model Precision
print("Precision: ",metrics.precision_score(test_y, y_pred_log))
```

```
# Model Recall
print("Recall: ",metrics.recall_score(test_y, y_pred_log))

#Model Accuracy
print("Accuracy: ",metrics.accuracy_score(test_y, y_pred_log))

#f1-score
print("f1-score: ", metrics.f1_score(test_y, y_pred_log))

#misclassification rate
misclassification_rate = (fp + fn) / (fp + fn + tp + tn)
print("misclassification rate: ", misclassification_rate)

#sensitivity
sensitivity = (tp) / (fn + tp)
print("sensitivity: ", sensitivity)

#specificity
specificity = (tn) / (tn + fp)
print("specificity: ", specificity)
```

```
Precision:  0.5310199789695058
Recall:  0.31346989447548107
Accuracy:  0.7930666666666667
f1-score:  0.3942232630757221
misclassification rate:  0.20693333333333333
sensitivity:  0.31346989447548107
specificity:  0.9242655798947189
```

svm gives higher recall so higher true positive rate so should choose it since we want to determine those who defaults.

2.13 Room for improvement

The classifiers that we chose for our problem are SVM and Logistic Regression. However, there are other classifiers such as decision tree classifiers. We will get a more comprehensive view of which is the best model for the problem if we tried out other classifiers.

If we know what is the cost of wrongly classifying a customer, we may be able to set the penalty more accurately and overall improve the profits of the company.