Project Name - Credit Card Default Prediction

Project Type - Classification

Project Summary -

In today's environment, credit cards have become a lifeline for many individuals, thus banks issue credit cards to us. A credit card is a form of payment card that allows charges to be made against a credit line rather than the account holder's cash deposit. When someone uses a credit card to make a purchase, the account of that individual accumulates a balance that must be paid off each month. We now know that the most typical issue in giving these types of offers is customers being unable to pay their bills, these people are what we call ”DEFAULTERS”. The Credit Card Default Database is Taiwan's most comprehensive classified with unbalanced dataset, data set of Customer credit card default payments, which provides information on domestic around Taiwan from 6 months in 2005. There is a wealth of information available for this purpose, including payment history and bill _Amount, paid amount, nature of the target, i.e. defaulters, and so on. The primary goal of this project is to do Exploratory Data Analysis and to use Machine Learning algorithms aimed at forecasting the case of customers default payments in Taiwan accuracy of Defaulters on dataset '''.Prediction of Credit Card Defaulters Our project is to look for patterns and explanations in the context and present the findings in a dynamic and visual way.

To perform Exploratory Data Analysis on CREDIT CARD FRAUD PREDICTION, we imported Python libraries such as Numpy, Pandas, Matplotlib, Seaborn, and Plot to

display the analysis dataset, as well as Sklearn preprocessing data using standard scaler, and importing logistic regression, Random Forest Classifier, XG boost classifier for predicting Defaulter getting accuracy for finding CREDIT CARD FRAUD PREDICTION and in Graphical form through bar plot The major characteristics of "CREDIT CARD FRAUD PREDICTION" are summarised using statistical graphics and other data visualisation tools.

Problem Statement

This initiative aims to forecast customer default payments in Taiwan. From the standpoint of risk management, the predictive accuracy of the predicted chance of default will be more valuable than the binary outcome of classification - credible or not credible clients. We must determine which clients will fall behind on their credit card payments. Financial dangers are demonstrating a trend regarding commercial bank credit risk as the financial industry has improved dramatically.As a result, one of the most serious risks to commercial banks is the risk prediction of credit clients. The current project is being created in order to analyse and predict the above-mentioned database. This research aims to identify credit card consumers who are more likely to default in the next month.

General Guidelines : -

1. Well-structured, formatted, and commented code is required.
2. Exception Handling, Production Grade Code & Deployment Ready Code will be a plus. Those students will be awarded some additional credits.

   The additional credits will have advantages over other students during Star Student selection.

   ```
       [ Note: - Deployment Ready Code is defined as, the whole
   .ipynb notebook should be executable in one go
                 without a single error logged. ]
   ```
3. Each and every logic should have proper comments.
4. You may add as many number of charts you want. Make Sure for each and every chart the following format should be answered.

   ```
   # Chart visualization code
   ```

- Why did you pick the specific chart?

- What is/are the insight(s) found from the chart?
- Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

5. You have to create at least 15 logical & meaningful charts having important insights.

[Hints : - Do the Vizualization in a structured way while following "UBM" Rule.

U - Univariate Analysis,

B - Bivariate Analysis (Numerical - Categorical, Numerical - Numerical, Categorical - Categorical)

M - Multivariate Analysis]

6. You may add more ml algorithms for model creation. Make sure for each and every algorithm, the following format should be answered.

- Explain the ML Model used and it's performance using Evaluation metric Score Chart.
- Cross- Validation & Hyperparameter Tuning
- Have you seen any improvement? Note down the improvement with updates Evaluation metric Score Chart.
- Explain each evaluation metric's indication towards business and the business impact pf the ML model used.

Let's Begin !

1. Know Your Data

Import Libraries

In [10]:
```
!pip install --upgrade xlrd
```

```
Requirement already satisfied: xlrd in /Library/Frameworks/Python.framework/V
ersions/3.11/lib/python3.11/site-packages (2.0.1)

[notice] A new release of pip is available: 25.0 -> 25.0.1
[notice] To update, run: pip install --upgrade pip
```

In [11]:
```
# Import Libraries for analysis and visualisation
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import seaborn as sns
import missingno as msno
%matplotlib inline

## Library of warnings would assist in ignoring warnings issued
import warnings
warnings.filterwarnings('ignore')

#Import necessary statistical libraries
import scipy.stats as stats
import statsmodels.api as sm
from scipy.stats import norm
from scipy.stats import chisquare
from scipy.stats import chi2_contingency
from scipy.stats import ks_2samp

#import necessary libraries  for feature engineering
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler

#import libraries for  machine learning model
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier
from xgboost import XGBClassifier

#import libraries for hyperparameter tuning and metric score
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn import metrics
from sklearn.metrics import accuracy_score, f1_score,roc_auc_score,precisior
```

Dataset Loading

In [12]:
```
# Load Dataset
filepath='credit card clients.xls'
credit_df= pd.read_excel(filepath,header=1)
```

Dataset First View

In [13]:
```
# Viewing the top 5 rows to take a glimpse of the data
credit_df.head()
```

Out[13]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4
0	1	20000	2	2	1	24	2	2	-1	-1
1	2	120000	2	2	2	26	-1	2	0	0
2	3	90000	2	2	2	34	0	0	0	0
3	4	50000	2	2	1	37	0	0	0	0
4	5	50000	1	2	1	57	-1	0	-1	0

5 rows × 25 columns

Dataset Rows & Columns count

In [14]:
```
# Dataset Rows & Columns count
credit_df.shape
```

Out[14]: (30000, 25)

In [15]:
```
print(f"the number of rows {credit_df.shape[0]}, and number of columns are {
```

```
the number of rows 30000, and number of columns are 25
```

Dataset Information

In [16]:
```
# Dataset Info
credit_df.info()
```

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

All Columns are Numerical

Duplicate Values

In [17]:
```
# Dataset Duplicate Value Count
duplicate_value=len(credit_df[credit_df.duplicated()])
print("The number of duplicate values in the data set is = ",duplicate_value
```

```
The number of duplicate values in the data set is =  0
```

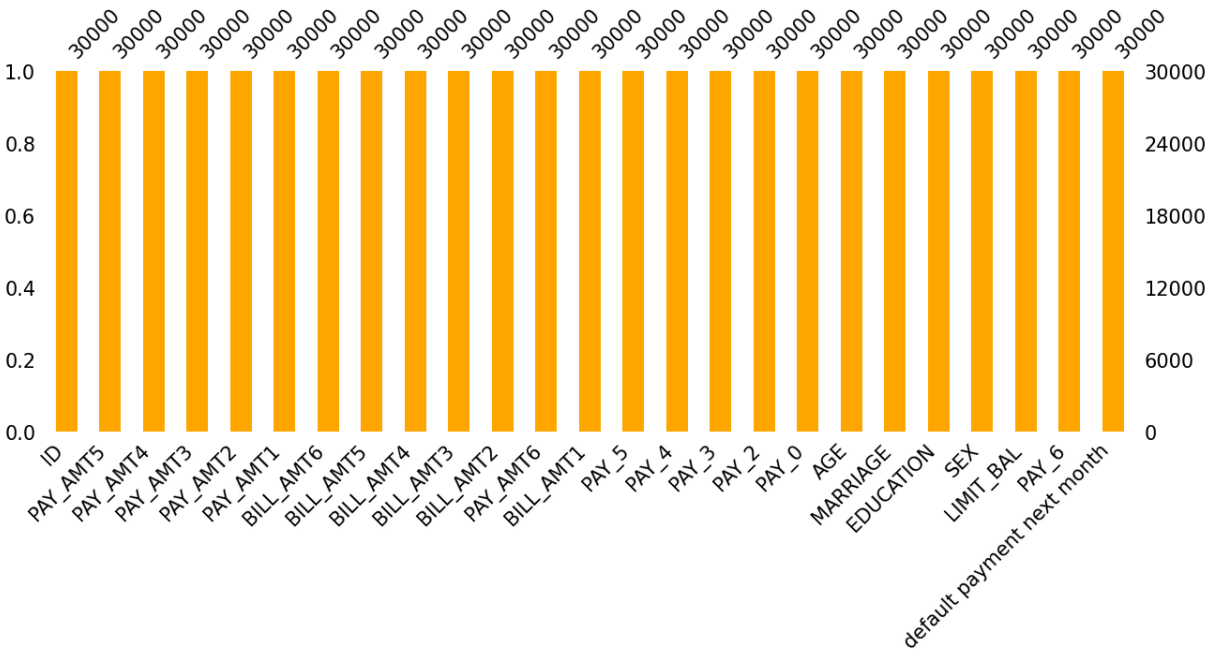
Missing Values/Null Values

In [18]:
```
# Missing Values/Null Values Count
credit_df.isna().sum()
```

Out[18]:
```
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
```

In [19]:
```
# Visualizing the missing values
msno.bar(credit_df, color='orange',sort='ascending', figsize=(15,5), fontsiz
```

Out[19]: <Axes: >



What did you know about your dataset?

A Taiwanese credit card company wants to better anticipate the chance of default for its customers and identify the primary elements that influence this likelihood. This would let the issuer decide who to offer a credit card to and what credit limit to supply. It would also help the issuer gain a better understanding of their current and potential clients, which would drive their future strategy, including their plans to offer tailored credit products to their customers.

The above dataset has 30000 rows and 25 columns. There are no mising values and duplicate values in the dataset.

2. Understanding Your Variables

In [20]:
```
# Dataset Columns
credit_df.columns
```

Out[20]:
```
Index(['ID', 'LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_0',
       'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AMT
2',
       'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1',
       'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6',
       'default payment next month'],
      dtype='object')
```

In [21]:
```
# Dataset Describe
credit_df.describe().T
```

Out[21]:

	count	mean	std	min	25%	50%
ID	30000.0	15000.500000	8660.398374	1.0	7500.75	15000.5
LIMIT_BAL	30000.0	167484.322667	129747.661567	10000.0	50000.00	140000.0
SEX	30000.0	1.603733	0.489129	1.0	1.00	2.0
EDUCATION	30000.0	1.853133	0.790349	0.0	1.00	2.0
MARRIAGE	30000.0	1.551867	0.521970	0.0	1.00	2.0
AGE	30000.0	35.485500	9.217904	21.0	28.00	34.0
PAY_0	30000.0	-0.016700	1.123802	-2.0	-1.00	0.0
PAY_2	30000.0	-0.133767	1.197186	-2.0	-1.00	0.0
PAY_3	30000.0	-0.166200	1.196868	-2.0	-1.00	0.0
PAY_4	30000.0	-0.220667	1.169139	-2.0	-1.00	0.0
PAY_5	30000.0	-0.266200	1.133187	-2.0	-1.00	0.0
PAY_6	30000.0	-0.291100	1.149988	-2.0	-1.00	0.0
BILL_AMT1	30000.0	51223.330900	73635.860576	-165580.0	3558.75	22381.5
BILL_AMT2	30000.0	49179.075167	71173.768783	-69777.0	2984.75	21200.0
BILL_AMT3	30000.0	47013.154800	69349.387427	-157264.0	2666.25	20088.5
BILL_AMT4	30000.0	43262.948967	64332.856134	-170000.0	2326.75	19052.0
BILL_AMT5	30000.0	40311.400967	60797.155770	-81334.0	1763.00	18104.5
BILL_AMT6	30000.0	38871.760400	59554.107537	-339603.0	1256.00	17071.0
PAY_AMT1	30000.0	5663.580500	16563.280354	0.0	1000.00	2100.0
PAY_AMT2	30000.0	5921.163500	23040.870402	0.0	833.00	2009.0
PAY_AMT3	30000.0	5225.681500	17606.961470	0.0	390.00	1800.0
PAY_AMT4	30000.0	4826.076867	15666.159744	0.0	296.00	1500.0
PAY_AMT5	30000.0	4799.387633	15278.305679	0.0	252.50	1500.0
PAY_AMT6	30000.0	5215.502567	17777.465775	0.0	117.75	1500.0
default payment next month	30000.0	0.221200	0.415062	0.0	0.00	0.0

Variables Description

As the response variable in this study, a binary variable, default payment (Yes = 1, No = 0), was used. This study analysed the literature and used the following variables as explanatory variables:

- **ID:** Unique ID of each client
- **LIMIT_BAL:** Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.
- **Gender:** 1 = male; 2 = female
- **Education:** 1 = graduate school; 2 = university; 3 = high school; 4 = others
- **Marital status:** 1 = married; 2 = single; 3 = others.
- **Age:** Age in years

History of past payment.

From April to September of 2005, we tracked historical monthly payment records.The payback status is measured using the following scale: -2=no spending, -1=paid in full, and 0=use of revolving credit (paid minimum only).

1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.

- **PAY_0:** Repayment status in September, 2005
- **PAY_2:** Repayment status in August, 2005
- **PAY_3:** Repayment status in July, 2005
- **PAY_4:** Repayment status in June, 2005
- **PAY_5:** Repayment status in May, 2005
- **PAY_6:** Repayment status in April, 2005

Amount of bill statement (NT dollar).

- **BILL_AMT1:** Amount of bill statement in September, 2005
- **BILL_AMT2:** Amount of bill statement in August, 2005
- **BILL_AMT3:** Amount of bill statement in July, 2005
- **BILL_AMT4:** Amount of bill statement in June, 2005
- **BILL_AMT5:** Amount of bill statement in May, 2005
- **BILL_AMT6:** Amount of bill statement in April, 2005

Amount of previous payment (NT dollar).

- **PAY_AMT1:** Amount of previous payment in September, 2005

- **PAY_AMT2:** Amount of previous payment in August, 2005
- **PAY_AMT3:** Amount of previous payment in July, 2005
- **PAY_AMT4:** Amount of previous payment in June, 2005
- **PAY_AMT5:** Amount of previous payment in May, 2005
- **PAY_AMT6:** Amount of previous payment in April, 2005
- **default.payment.next.month:** Default payment (1=yes, 0=no)

Check Unique Values for each variable.

In [22]:
```
# Check Unique Values for each variable.
# Check Unique Values for each variable.
for i in credit_df.columns.tolist():
  print("No. of unique values in ",i,"is",credit_df[i].nunique())
```

```
No. of unique values in  ID is 30000
No. of unique values in  LIMIT_BAL is 81
No. of unique values in  SEX is 2
No. of unique values in  EDUCATION is 7
No. of unique values in  MARRIAGE is 4
No. of unique values in  AGE is 56
No. of unique values in  PAY_0 is 11
No. of unique values in  PAY_2 is 11
No. of unique values in  PAY_3 is 11
No. of unique values in  PAY_4 is 11
No. of unique values in  PAY_5 is 10
No. of unique values in  PAY_6 is 10
No. of unique values in  BILL_AMT1 is 22723
No. of unique values in  BILL_AMT2 is 22346
No. of unique values in  BILL_AMT3 is 22026
No. of unique values in  BILL_AMT4 is 21548
No. of unique values in  BILL_AMT5 is 21010
No. of unique values in  BILL_AMT6 is 20604
No. of unique values in  PAY_AMT1 is 7943
No. of unique values in  PAY_AMT2 is 7899
No. of unique values in  PAY_AMT3 is 7518
No. of unique values in  PAY_AMT4 is 6937
No. of unique values in  PAY_AMT5 is 6897
No. of unique values in  PAY_AMT6 is 6939
No. of unique values in  default payment next month is 2
```

3. *Data Wrangling*

Data Wrangling Code

In [23]:
```
#renaming dependendent Variable
credit_df.rename(columns={'default payment next month' : 'IsDefaulter'}, inp
#Changing name of some columns for simplicity and better understanding
```

```
credit_df.rename(columns={'PAY_0':'PAY_SEPT','PAY_2':'PAY_AUG','PAY_3':'PAY_
credit_df.rename(columns={'BILL_AMT1':'BILL_AMT_SEPT','BILL_AMT2':'BILL_AMT_
credit_df.rename(columns={'PAY_AMT1':'PAY_AMT_SEPT','PAY_AMT2':'PAY_AMT_AUG'
```

In [24]:
```
#Dropping ID column because there is no use of ID further.
credit_df.drop('ID',axis=1,inplace=True)
```

In [25]:
```
#copy the original data to a different dataframe for better visualisation
df_credit=credit_df.copy()
```

In [26]:
```
#replacing values with there labels
df_credit.replace({'SEX': {1 : 'Male', 2 : 'Female'}}, inplace=True)
df_credit.replace({'EDUCATION' : {1 : 'Graduate School', 2 : 'University', 3
df_credit.replace({'MARRIAGE' : {1 : 'Married', 2 : 'Single', 3 : 'others',0
df_credit.replace({'IsDefaulter' : {1 : 'Yes', 0 : 'No'}}, inplace = True)
```

In [27]:
```
df_credit.head()
```

Out[27]:

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_SEPT	PAY_AUG	PAY_JUL
0	20000	Female	University	Married	24	2	2	-1
1	120000	Female	University	Single	26	-1	2	0
2	90000	Female	University	Single	34	0	0	0
3	50000	Female	University	Married	37	0	0	0
4	50000	Male	University	Married	57	-1	0	-1

5 rows × 24 columns

What all manipulations have you done and insights you found?

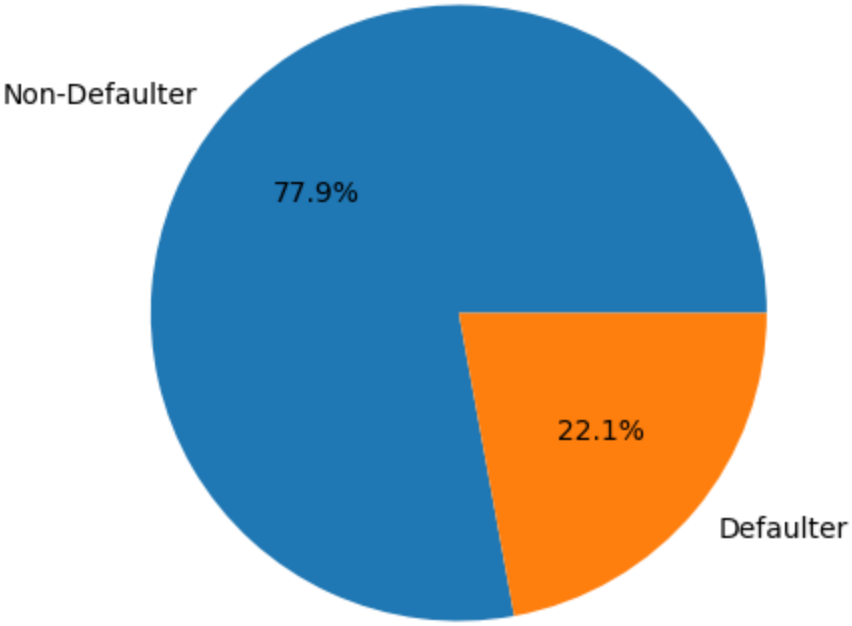
We can rename dependent variables and feature names to improve understanding of the feature.

- **converting numerical values** to categorical values for easy comprehension
- **Gender** (one = male, two = female)
- **Education** 1 = graduate school, 2 = university, 3 = high school, and 4 = others.
- **Status of marriage** (1 = married, 2 = single, 3 = other)

4. Data Vizualization, Storytelling & Experimenting with charts : Understand the relationships between variables

Chart 1- Visualization of Defaulter vs Non-Defaulter

In [28]:
```
# Visualization of IsDefaulter using pie chart
plt.figure(figsize=(10,5))
plt.pie(x=df_credit['IsDefaulter'].value_counts(),labels=['Non-Defaulter','D
plt.show()
```



1. Why did you pick the specific chart?

Pie Chart used to represent the summarizing set of nominal data or displays the different value of a given categorical variable (ex- precentage distribution).

2. What is/are the insight(s) found from the chart?

NO=Payment will not default,Yes=Payment will default

From the graph we can see that number of defaulter(22%) is less than Non defaulter(78%).

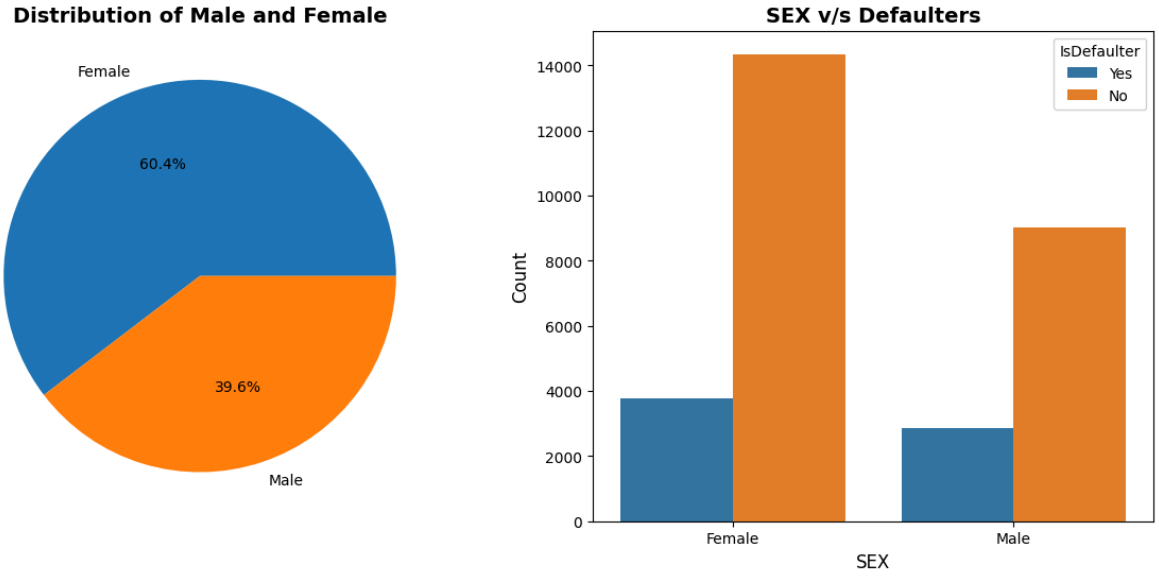
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Yes, class imbalance can have a negative impact on classification model accuracy, so we must address this issue before putting the data into training.

Chart - 2 Visualisation of Gender Column

In [29]:
```
fig,ax=plt.subplots(1,2,figsize=(15,6))
#univariate analysis-distribution of male female ratio
ax[0].pie(x=df_credit['SEX'].value_counts(),labels=['Female','Male'],autopct
ax[0].set_title('Distribution of Male and Female',fontsize=14,fontweight='bo
#bivariate analysis -Defaulter and Non-Defaulter distribution between male a
sns.countplot(x = 'SEX',hue = 'IsDefaulter',data = df_credit,ax=ax[1])
ax[1].set_title('SEX v/s Defaulters', fontsize = 14, fontweight = 'bold')
ax[1].set_xlabel('SEX',fontsize = 12)
ax[1].set_ylabel('Count',fontsize=12)
plt.show()
```



1. Why did you pick the specific chart?

- Pie Chart used to represent the summarizing set of nominal data or displays the different value of a given categorical variable.
- The countplot is used to display the number of occurrences of the observation in the categorical variable. For the visual representation, it employs the concept of a bar chart.

2. What is/are the insight(s) found from the chart?

- Number of Female credit card holder is more than Male
- Number of Female defaulter is more than Male defaulter

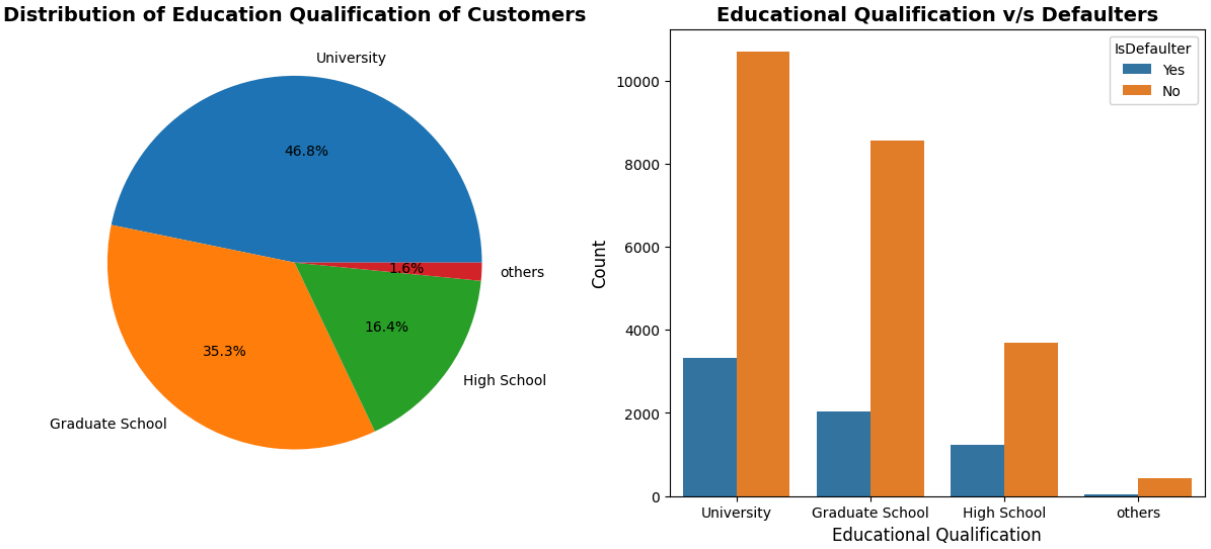
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

As we can see, female credit card holders outnumber male credit card holders, thus in order to boost male clients, banks should offer incentives to male customers while also taking care of their female clients in order to grow their business. However, some policies should be made available to male clients in order to limit the possibility of default.

Chart - 3 Visualisation of Education Column

In [30]:
```
fig,ax=plt.subplots(1,2,figsize=(15,6))
#univariate analysis-distribution of educational qualification
ax[0].pie(x=df_credit['EDUCATION'].value_counts(),labels=df_credit['EDUCATIO
ax[0].set_title('Distribution of Education Qualification of Customers',fonts
#bivariate analysis -Defaulter and Non-Defaulter distribution between male a
sns.countplot(x = 'EDUCATION',hue = 'IsDefaulter',data = df_credit,ax=ax[1])
ax[1].set_title('Educational Qualification v/s Defaulters', fontsize = 14, f
ax[1].set_xlabel('Educational Qualification',fontsize = 12)
ax[1].set_ylabel('Count',fontsize=12)
plt.show()
```



1. Why did you pick the specific chart?

- Pie Chart used to represent the summarizing set of nominal data or displays the different value of a given categorical variable.
- The countplot is used to display the number of occurrences of the observation in the categorical variable. For the visual representation, it employs the concept of a bar chart.

2. What is/are the insight(s) found from the chart?

- Most of the customer's educational qualification is university pass out followed by graduate school pass out

- Most number of defaulter is also from University but in terms of default ratio university people having less default ratio,high school have most default ratio.

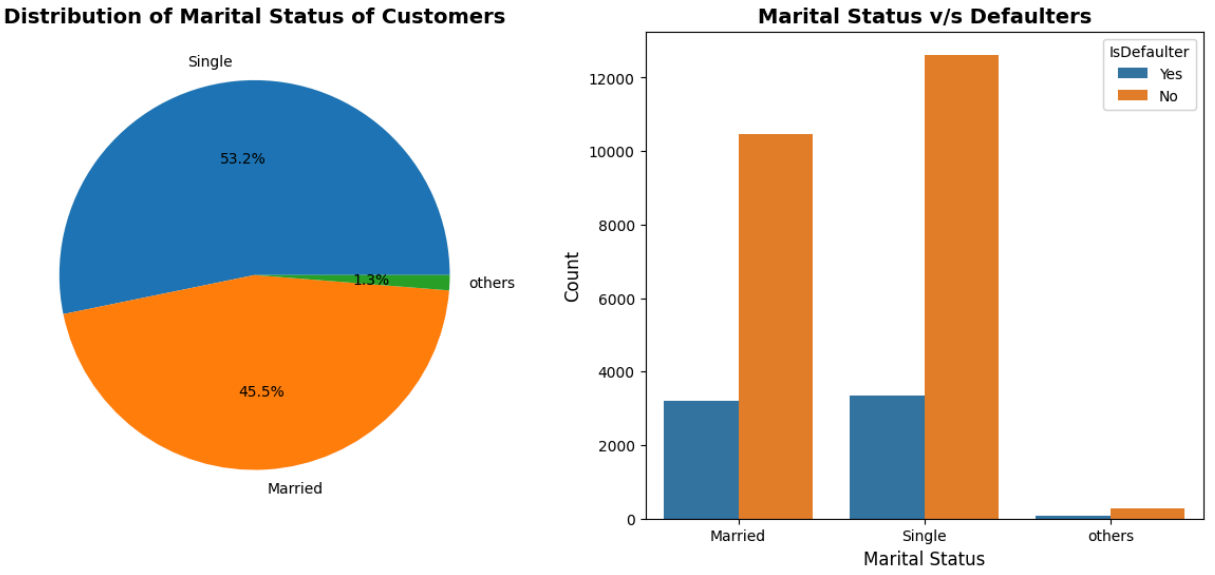
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

As we can see, the majority of credit card holders are from universities, followed by graduate schools, so banks can target these people to grow their business. As we know, the source of income for high school candidates is very low and their default ratio is also high, so there is no need to focus more on this group of people.

Chart - 4 Visualisation of Marriage Column

In [31]:
```
fig,ax=plt.subplots(1,2,figsize=(15,6))
#univariate analysis-distribution of Marriage
ax[0].pie(x=df_credit['MARRIAGE'].value_counts(),labels=df_credit['MARRIAGE'
ax[0].set_title('Distribution of Marital Status of Customers',fontsize=14,fo
#bivariate analysis -Defaulter and Non-Defaulter distribution between male a
sns.countplot(x = 'MARRIAGE',hue = 'IsDefaulter',data = df_credit,ax=ax[1])
ax[1].set_title('Marital Status v/s Defaulters', fontsize = 14, fontweight =
ax[1].set_xlabel('Marital Status',fontsize = 12)
ax[1].set_ylabel('Count',fontsize=12)
plt.show()
```



1. Why did you pick the specific chart?

- Pie Chart used to represent the summarizing set of nominal data or displays the different value of a given categorical variable.
- The countplot is used to display the number of occurrences of the observation in the categorical variable. For the visual representation, it employs the concept of a bar chart.

2. What is/are the insight(s) found from the chart?

- Maximum Card holder are single
- Married people have higher default ratio

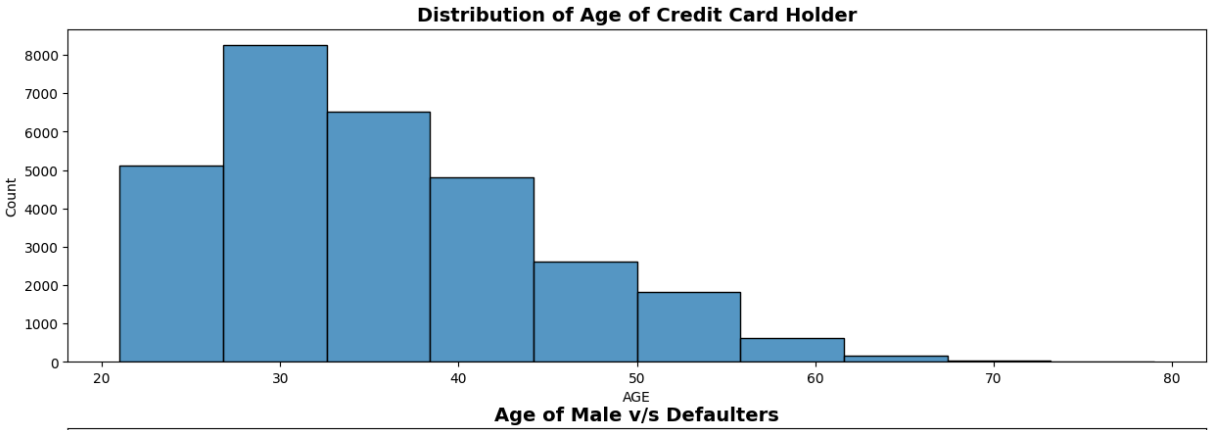
3. Will the gained insights help creating a positive business impact?

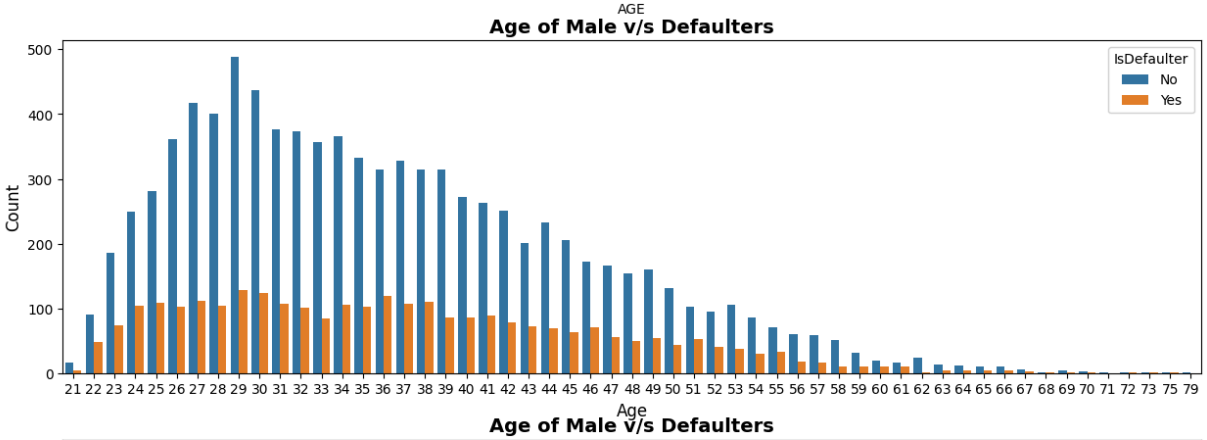
Are there any insights that lead to negative growth? Justify with specific reason.

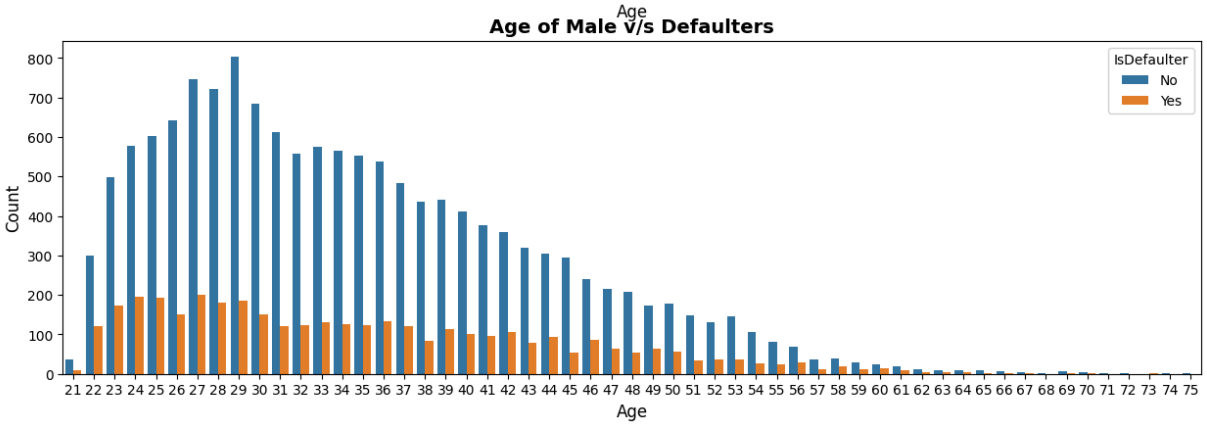
Single folks spend more than married people, thus they will use credit cards more frequently.As a result, targeting Single people will undoubtedly enhance sales.

Chart - 5 Visualisation of Age Column

In [32]:
```
fig,ax=plt.subplots(3,1,figsize=(15,15))
#univariate analysis-distribution of Marriage
age_dist=sns.histplot(x=df_credit['AGE'],ax=ax[0],bins=10)
age_dist.set_title('Distribution of Age of Credit Card Holder',fontsize=14,f
#Create two dataframe of male and female
Age_male=df_credit[df_credit['SEX']=='Male']
Age_female=df_credit[df_credit['SEX']=='Female']
sns.countplot(x = 'AGE',hue = 'IsDefaulter',data = Age_male,ax=ax[1])
#plot age of male card holder vs Defaulter
ax[1].set_title('Age of Male v/s Defaulters', fontsize = 14, fontweight = 'b
ax[1].set_xlabel('Age',fontsize = 12)
ax[1].set_ylabel('Count',fontsize=12)
#plot age of female card holder vs Defaulter
sns.countplot(x = 'AGE',hue = 'IsDefaulter',data = Age_female,ax=ax[2])
ax[2].set_title('Age of Male v/s Defaulters', fontsize = 14, fontweight = 'b
ax[2].set_xlabel('Age',fontsize = 12)
ax[2].set_ylabel('Count',fontsize=12)
plt.show()
```







1. Why did you pick the specific chart?

- A histogram is a traditional visualization tool that counts the number of data that fall into discrete bins to illustrate the distribution of one or more variables.
- The countplot is used to display the number of occurrences of the observation in the categorical variable. For the visual representation, it employs the concept of a bar chart.

2. What is/are the insight(s) found from the chart?

- Most of the credit card holder belongs to 25-30 years
- Number of credit card holder are less above 60 years.
- For equal age ratio of defualter for male and female not varied much
- Above 50 year old the ratio of defaulter has increased

3. Will the gained insights help creating a positive business impact?

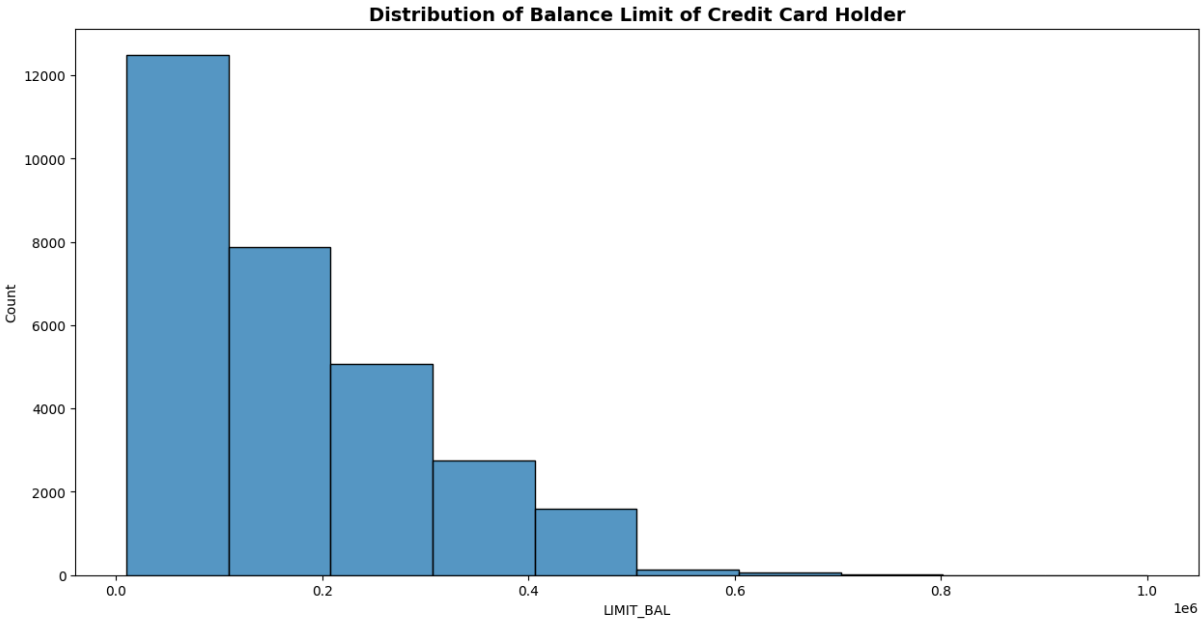
Are there any insights that lead to negative growth? Justify with specific reason.

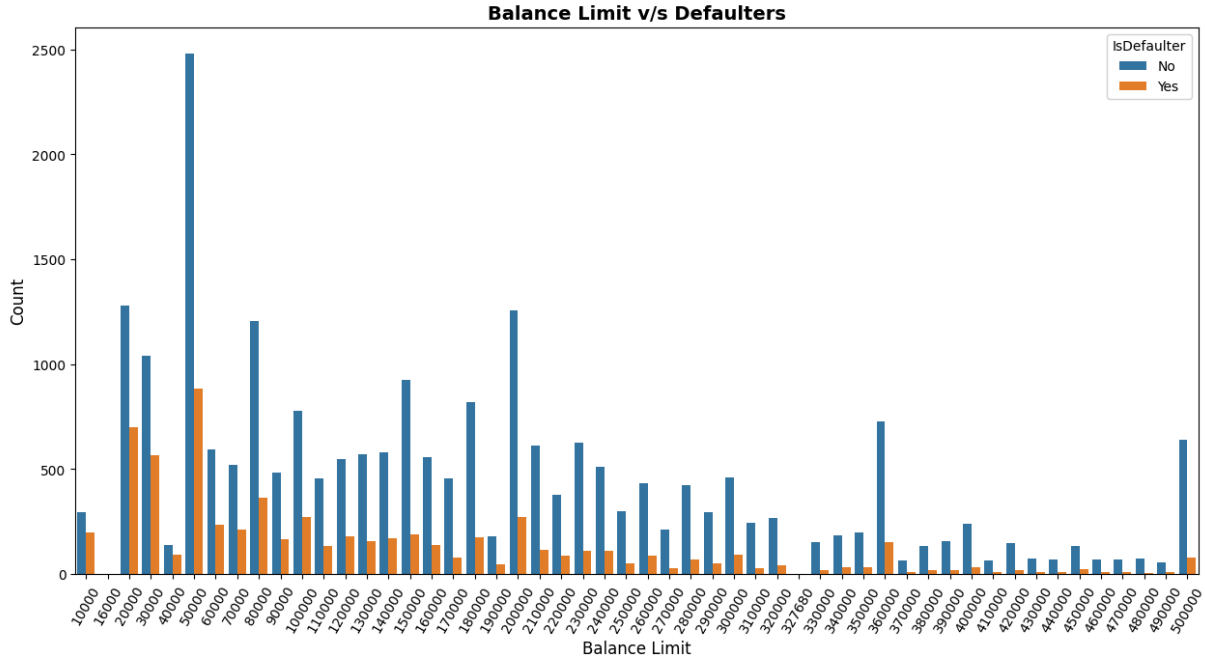
Customers are fewer as the majority of defaulters are between the ages of 20 and 25, and over the age of 60. Before issuing the credit card, it must be thoroughly reviewed. Otherwise, it may have a negative impact on corporate growth.

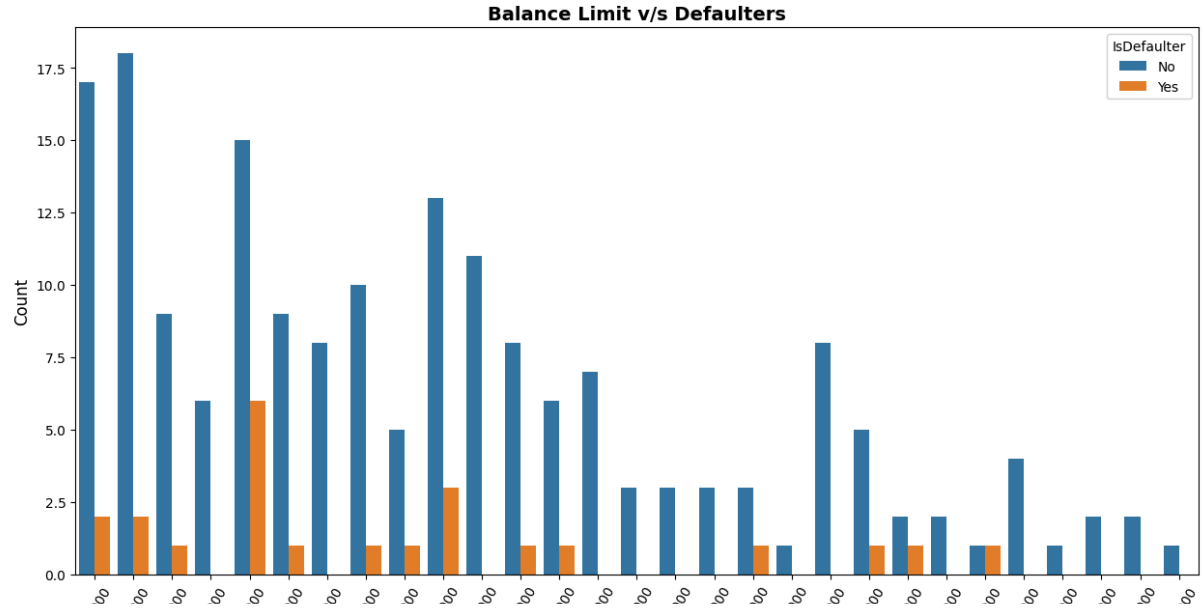
Chart - 6 Visualisation of Default payment next month Column

In [33]:
```
fig,ax=plt.subplots(3,1,figsize=(15,25))
#univariate analysis-distribution of Marriage
age_dist=sns.histplot(x=df_credit['LIMIT_BAL'],ax=ax[0],bins=10)
age_dist.set_title('Distribution of Balance Limit of Credit Card Holder',for
Limit_1=df_credit[df_credit['LIMIT_BAL']<=500000]
Limit_2=df_credit[df_credit['LIMIT_BAL']>500000]
#plot age of male card holder vs Defaulter
sns.countplot(x = 'LIMIT_BAL',hue = 'IsDefaulter',data = Limit_1,ax=ax[1])
ax[1].set_title('Balance Limit v/s Defaulters', fontsize = 14, fontweight =
ax[1].tick_params(axis='x', labelrotation= 60)
ax[1].set_xlabel('Balance Limit',fontsize = 12)
ax[1].set_ylabel('Count',fontsize=12)
#plot age of male card holder vs Defaulter
sns.countplot(x = 'LIMIT_BAL',hue = 'IsDefaulter',data = Limit_2,ax=ax[2])
ax[2].set_title('Balance Limit v/s Defaulters', fontsize = 14, fontweight =
ax[2].tick_params(axis='x', labelrotation= 60)
ax[2].set_xlabel('Balance Limit',fontsize = 12)
ax[2].set_ylabel('Count',fontsize=12)
```

Out[33]:
```
Text(0, 0.5, 'Count')
```


Distribution of Balance Limit of Credit Card Holder
Count
LIMIT_BAL
1e6


Balance Limit v/s Defaulters
IsDefaulter
No
Yes
Count
Balance Limit


Balance Limit v/s Defaulters
IsDefaulter
No
Yes
Count

1. Why did you pick the specific chart?

- A histogram is a traditional visualization tool that counts the number of data that fall into discrete bins to illustrate the distribution of one or more variables.
- The countplot is used to display the number of occurrences of the observation in the categorical variable. For the visual representation, it employs the concept of a bar chart.

2. What is/are the insight(s) found from the chart?

- Maximum Credit card limit is below 100000 .
- Very few Credit Limit are above 500000.
- When credit limit is less than equal to 50k deault ratio is high.
- Default ratio is exceptionally high when credit limit is 550000 and 600000.

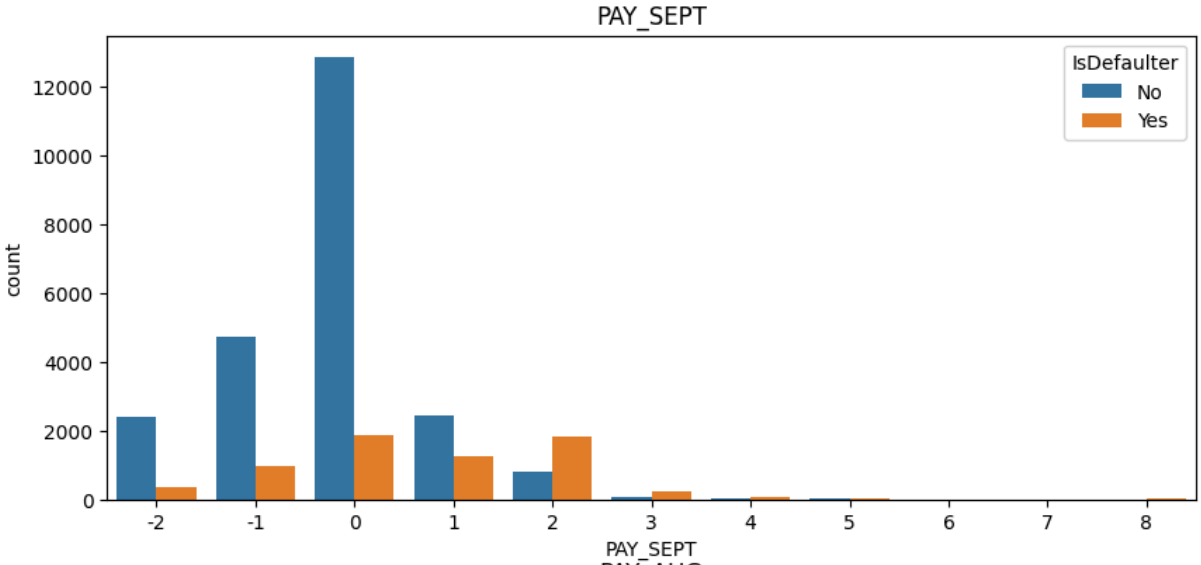
3. Will the gained insights help creating a positive business impact?

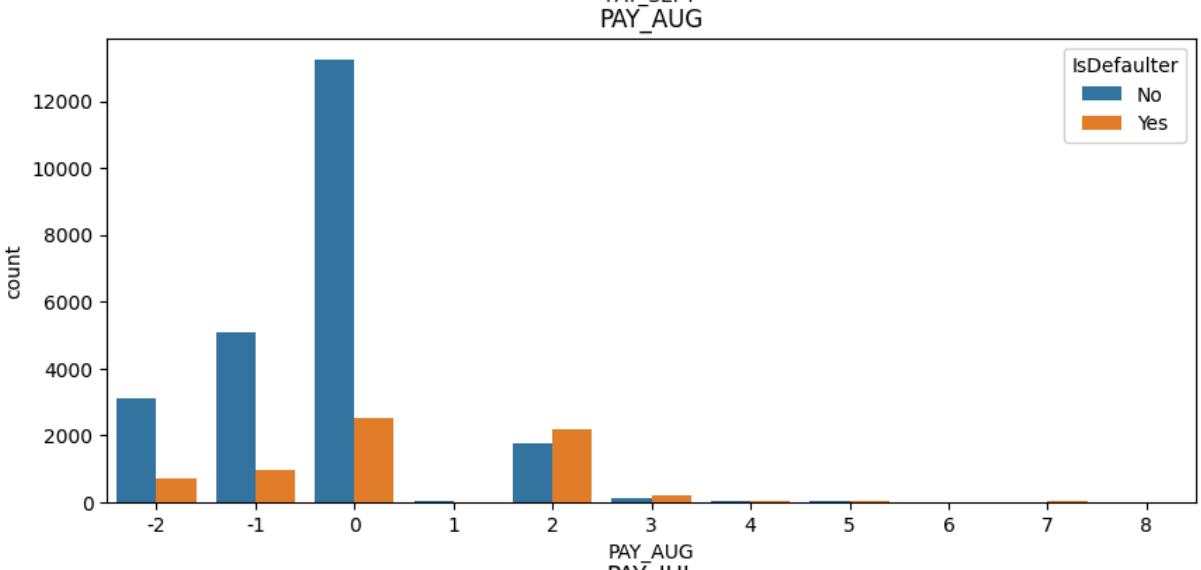
Are there any insights that lead to negative growth? Justify with specific reason.

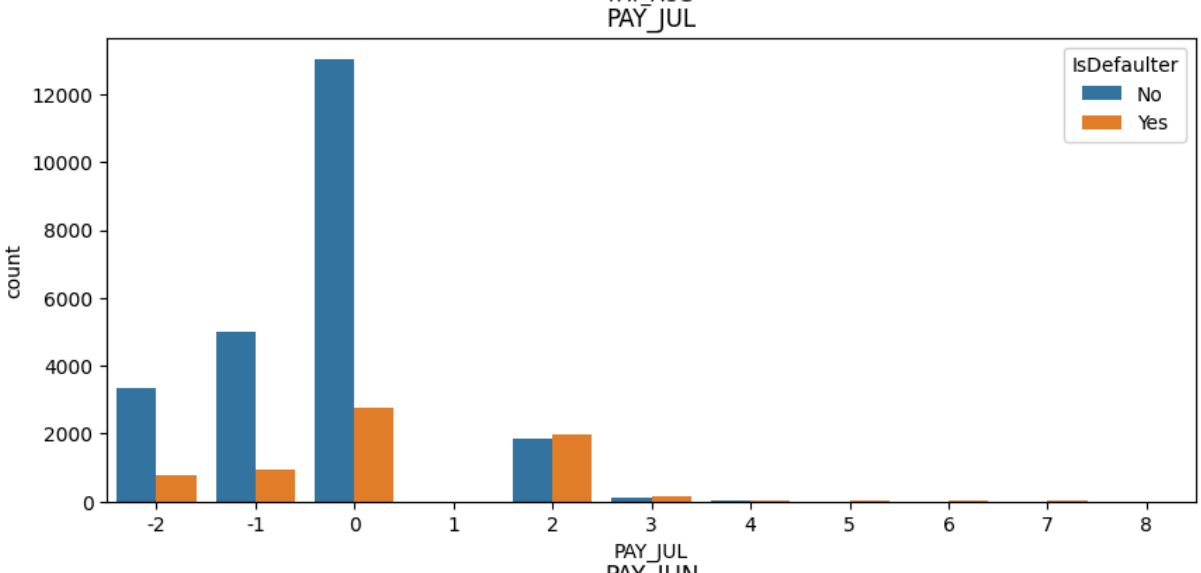
Customers with credit limits of $50,000 or more are more likely to default, so the bank should charge them in the event of a default and may cut their limit to avoid any form of damage to the company.

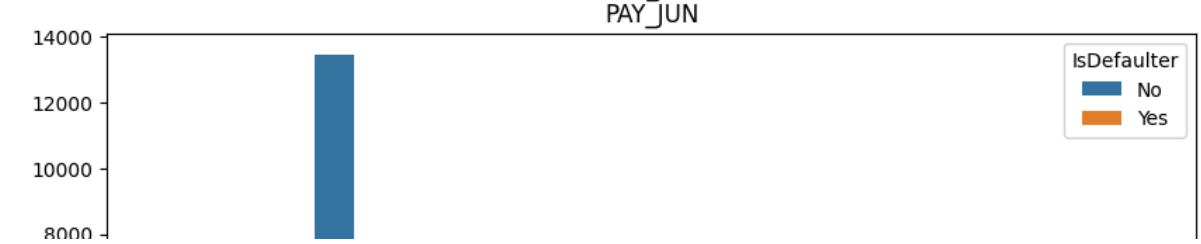
Chart - 7 Monthly wise repayment vs Default

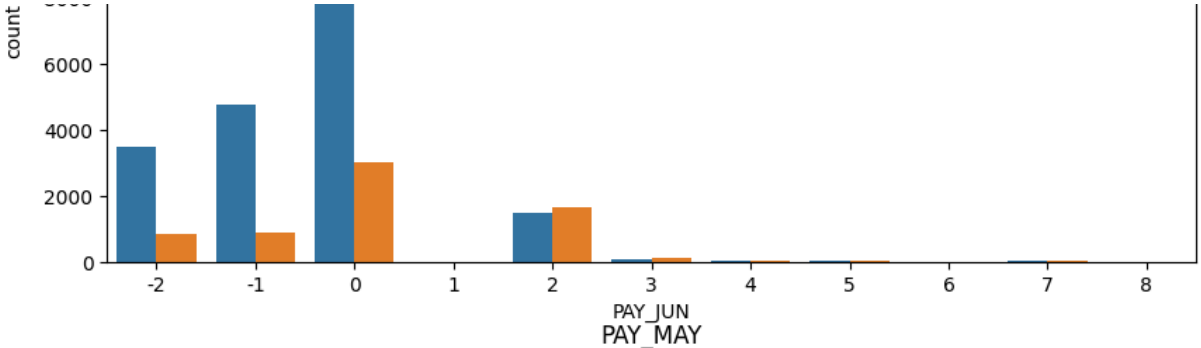
In [34]:
```
# create a list of monthly repayment columns
repay_col = ['PAY_SEPT','PAY_AUG','PAY_JUL','PAY_JUN','PAY_MAY','PAY_APR']
# plot the univariate distribution
fig,ax=plt.subplots(len(repay_col),1,figsize=(10,30))
for i,col in enumerate(repay_col):
  #plot the bivariate with isdefaulter
  sns.countplot(x = col,hue = 'IsDefaulter',data = df_credit,ax=ax[i])
  ax[i].set_title(col)
```

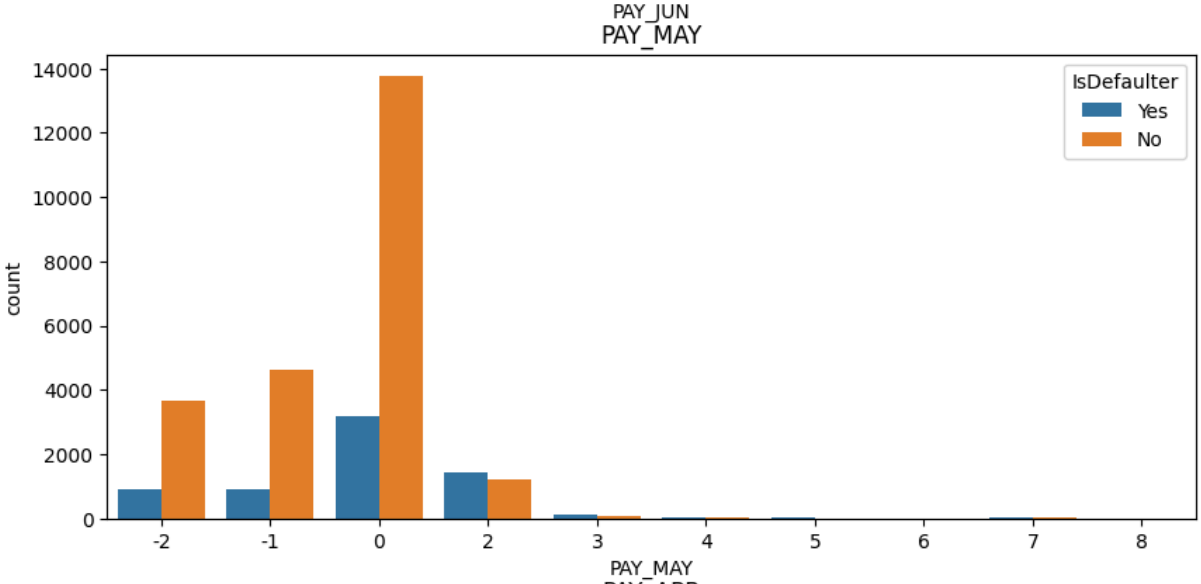

PAY_SEPT
IsDefaulter
No
Yes
count
PAY_SEPT

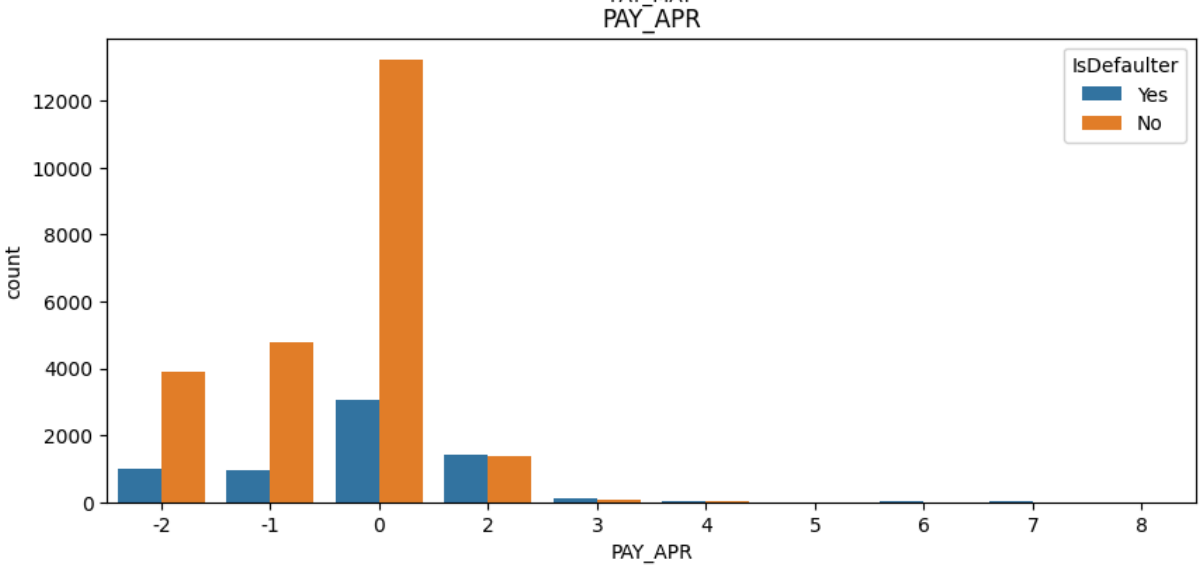

PAY_AUG
IsDefaulter
No
Yes
count
PAY_AUG


PAY_JUL
IsDefaulter
No
Yes
count
PAY_JUL


PAY_JUN
IsDefaulter
No
Yes







1. Why did you pick the specific chart?

The countplot is used to display the number of occurrences of the observation in the categorical variable.

For the visual representation, it employs the concept of a bar chart.

2. What is/are the insight(s) found from the chart?

No consumption = 2

-1 = fully paid

0 = usage of revolving credit (paid only the minimum)

1 = one-month payment delay

2 = two-month payment delay,... 8 = eight-month payment delay, 9 = nine-month payment delay and above)

We can see that customers who pay only the minimum amount have a higher risk of default.Customers in this category also mostly use credit cards.

When payment delay is greater than 2 month default ratio is very high.

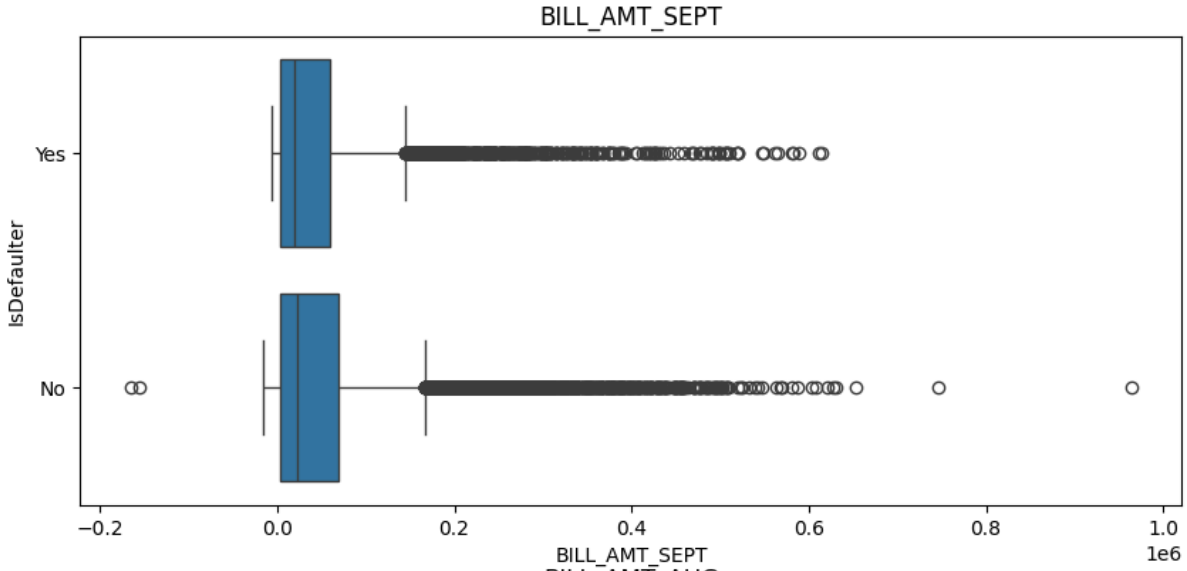
3. Will the gained insights help creating a positive business impact?

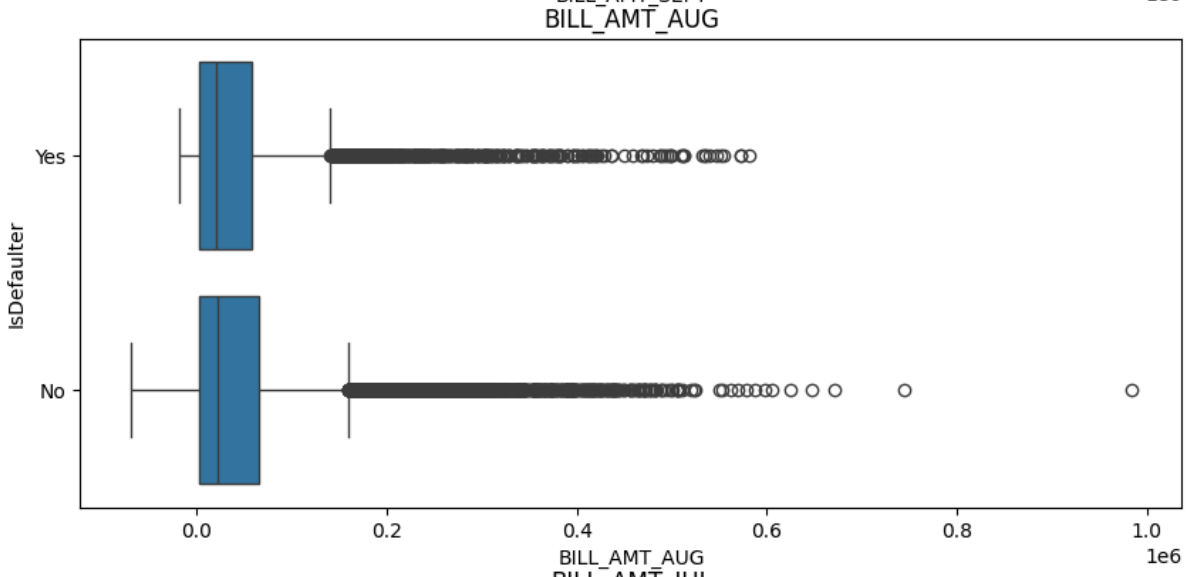
Are there any insights that lead to negative growth? Justify with specific reason.

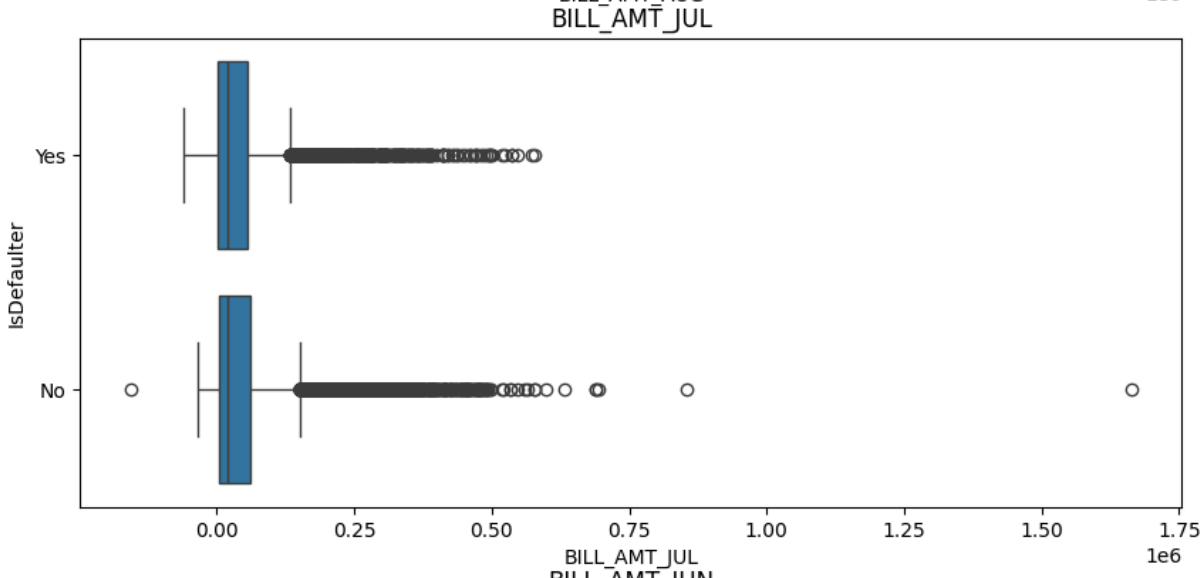
When payment delay is greater than 2 month,default ratio is high so it has negative impact on buisness. So when customer is is delaying 1 month company should concerned.

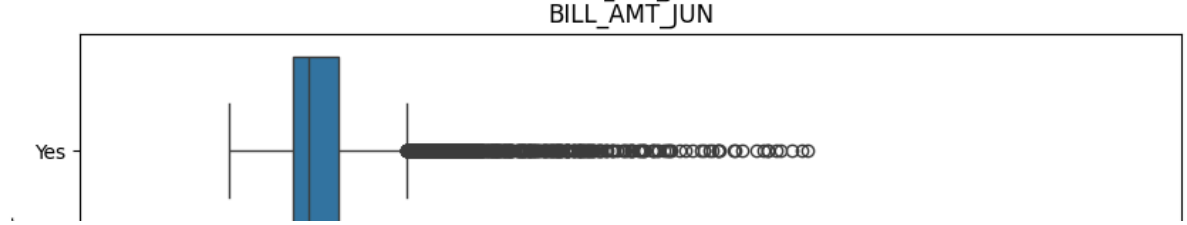
Chart - 8 Visualisation of Monthly Billing Amount

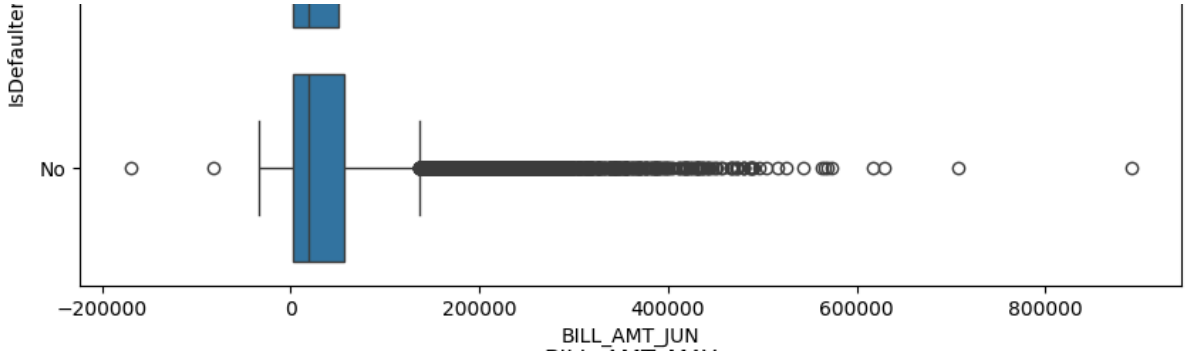
In [35]:
```
# Chart - 9 visualization code
monthlybill_col = ['BILL_AMT_SEPT','BILL_AMT_AUG', 'BILL_AMT_JUL', 'BILL_AMT
# plot the univariate distribution
fig,ax=plt.subplots(len(monthlybill_col),1,figsize=(10,30))
for i,col in enumerate(monthlybill_col):
  #plot the bivariate with isdefaulter
  sns.boxplot(x = col,y='IsDefaulter',data = df_credit,ax=ax[i])
  ax[i].set_title(col)
```

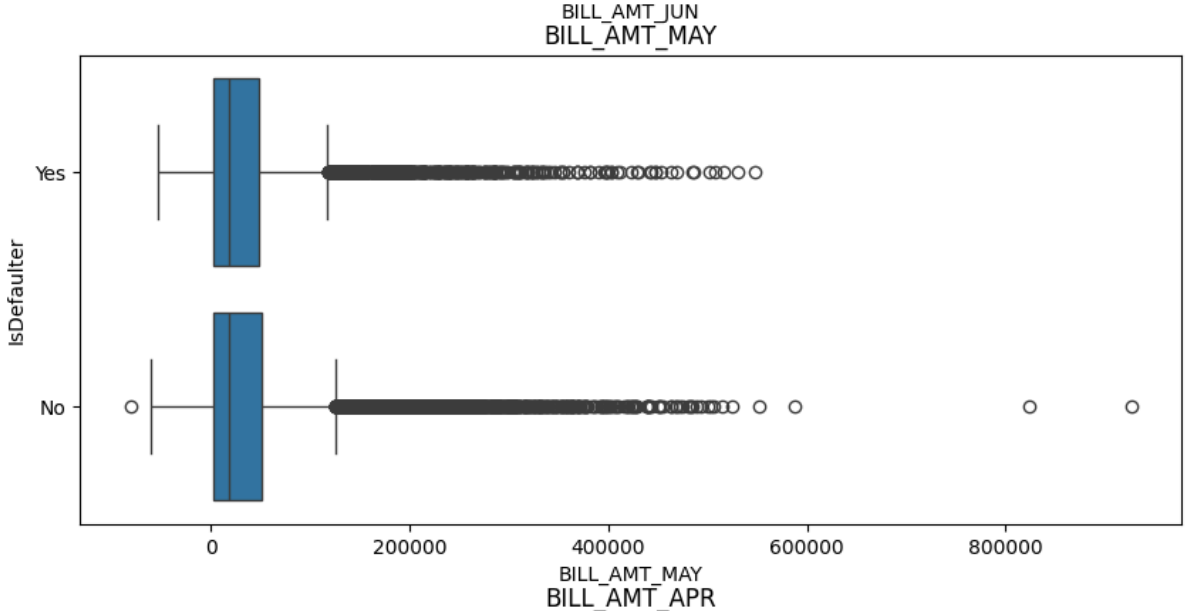

BILL_AMT_SEPT
Yes
No
IsDefaulter
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1e6
BILL_AMT_SEPT

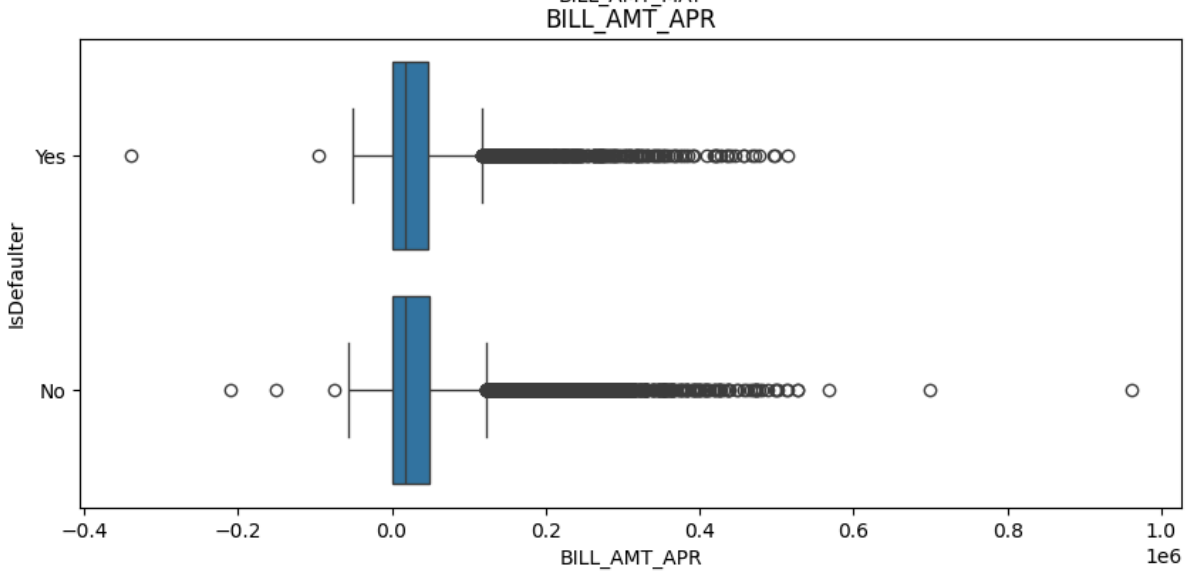

BILL_AMT_AUG
Yes
No
IsDefaulter
0.0
0.2
0.4
0.6
0.8
1.0
1e6
BILL_AMT_AUG


BILL_AMT_JUL
Yes
No
IsDefaulter
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e6
BILL_AMT_JUL


BILL_AMT_JUN
Yes







1. Why did you pick the specific chart?

Box plots provide a rapid visual assessment of a dataset's variability. They display the dataset's median, upper and lower quartiles, lowest and maximum values, and any outliers. Outliers in data might show errors or uncommon events.

2. What is/are the insight(s) found from the chart?

From the plots we can see that data distribution of monthly bill has no thick difference between defaulter and non defaulter

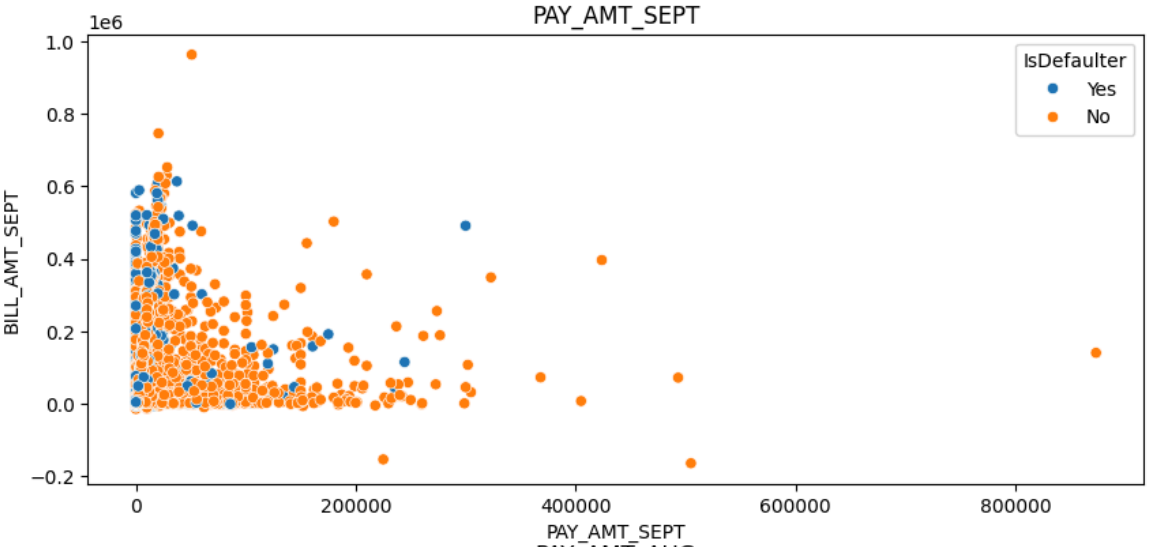
3. Will the gained insights help creating a positive business impact?

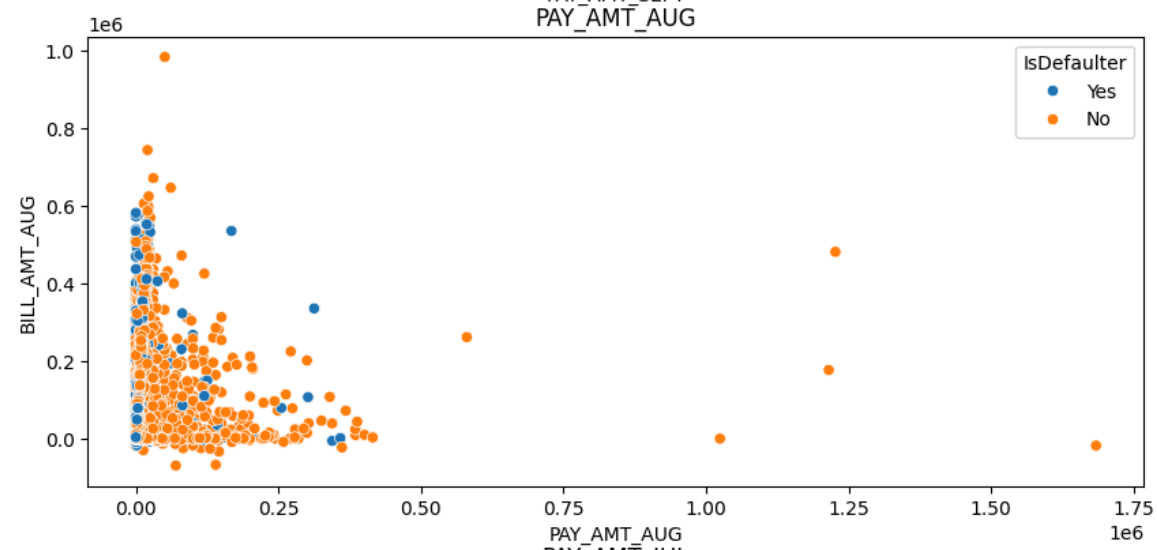
Are there any insights that lead to negative growth? Justify with specific reason.

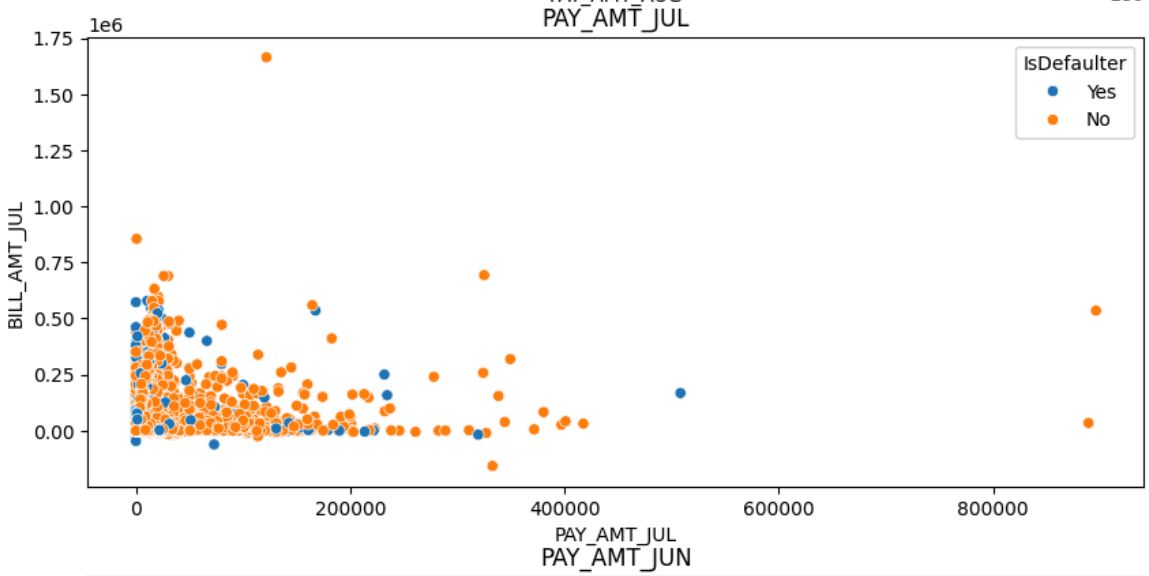
There is no insight on buisness from the figures.

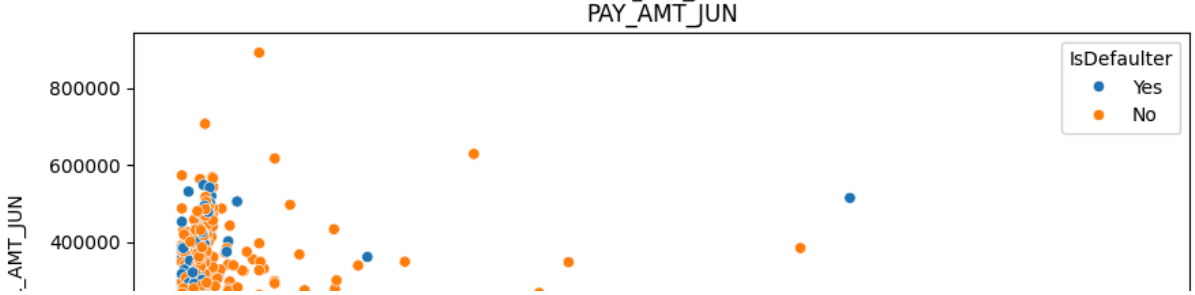
Chart - 9 Visualization of Amount of Previous Payment vs Bill Amount

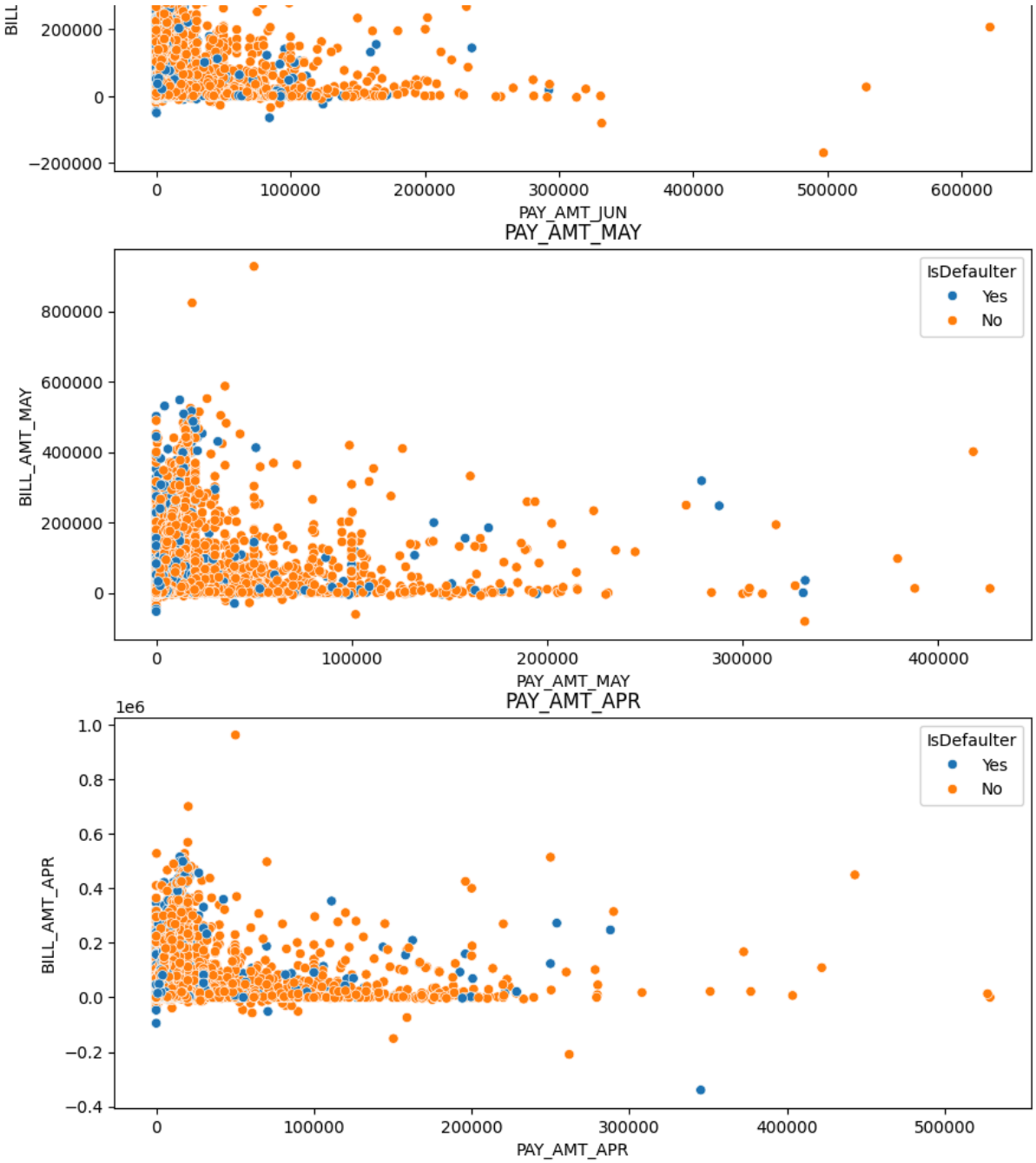
In [36]:
```
# Chart - 9 visualization code
previous_col = ['PAY_AMT_SEPT', 'PAY_AMT_AUG', 'PAY_AMT_JUL','PAY_AMT_JUN',
# plot the univariate distribution
fig,ax=plt.subplots(len(previous_col),1,figsize=(10,30))
for i,col in enumerate(previous_col):
  #plot the bivariate with isdefaulter
  sns.scatterplot(x = col,y=monthlybill_col[i],hue='IsDefaulter',data = df_c
  ax[i].set_title(col)
```


PAY_AMT_SEPT
1e6
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
BILL_AMT_SEPT
0
200000
400000
600000
800000
PAY_AMT_SEPT
IsDefaulter
Yes
No


PAY_AMT_AUG
1e6
1.0
0.8
0.6
0.4
0.2
0.0
BILL_AMT_AUG
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e6
PAY_AMT_AUG
IsDefaulter
Yes
No


PAY_AMT_JUL
1e6
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
BILL_AMT_JUL
0
200000
400000
600000
800000
PAY_AMT_JUL
IsDefaulter
Yes
No


PAY_AMT_JUN
800000
600000
400000
IsDefaulter
Yes
No



1. Why did you pick the specific chart?

A scatterplot is used to plot how much a numerical feature is affected by another numerical value.

2. What is/are the insight(s) found from the chart?

When Previous payment is low but Monthly Bill is high there is high possibility of default the payment.

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

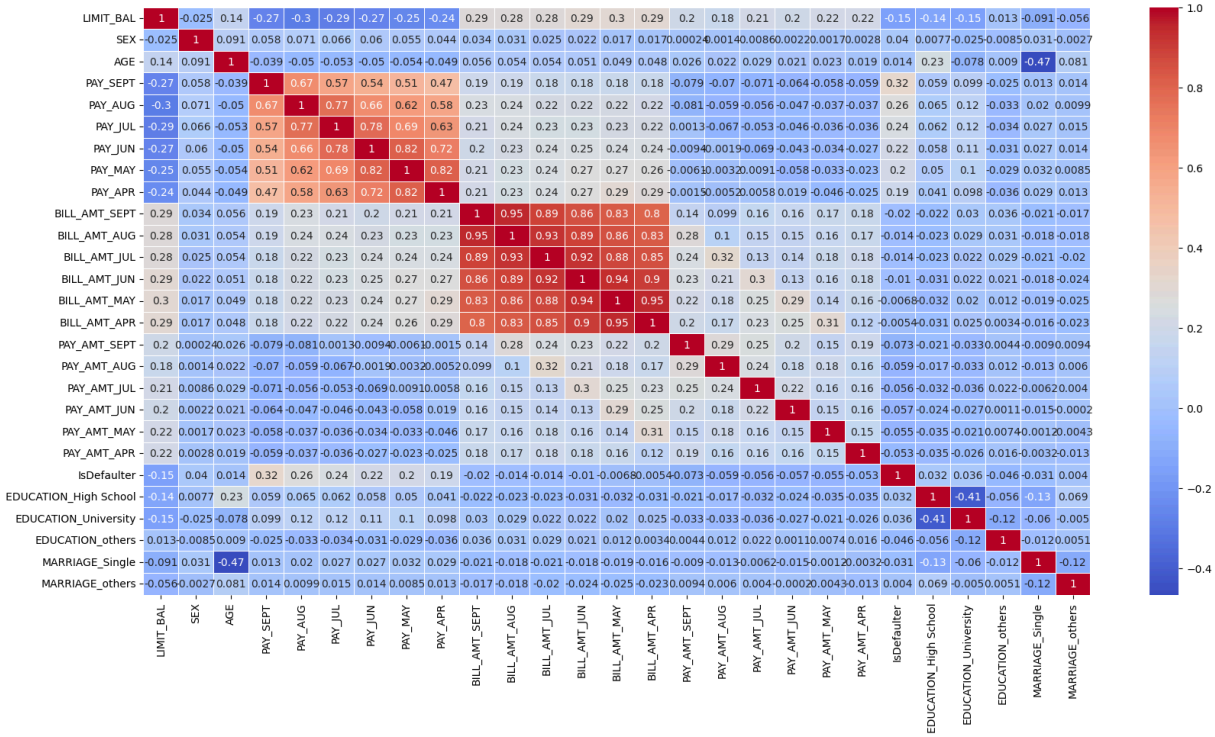
When customer is spending exceptionally high according to his his payment history company should concerned.

Chart - 10 - Correlation Heatmap

In [37]:
```
# Convert 'SEX' column (Female = 0, Male = 1)
df_credit['SEX'] = df_credit['SEX'].map({'Female': 0, 'Male': 1})

# Convert 'IsDefaulter' column (Yes = 1, No = 0)
df_credit['IsDefaulter'] = df_credit['IsDefaulter'].map({'Yes': 1, 'No': 0})

# One-Hot Encoding for 'EDUCATION' and 'MARRIAGE' (convert to numeric dummy
df_credit = pd.get_dummies(df_credit, columns=['EDUCATION', 'MARRIAGE'], drc

# Display column types after encoding
print(df_credit.dtypes)
```

```
LIMIT_BAL                int64
SEX                      int64
AGE                      int64
PAY_SEPT                 int64
PAY_AUG                  int64
PAY_JUL                  int64
PAY_JUN                  int64
PAY_MAY                  int64
PAY_APR                  int64
BILL_AMT_SEPT            int64
BILL_AMT_AUG             int64
BILL_AMT_JUL             int64
BILL_AMT_JUN             int64
BILL_AMT_MAY             int64
BILL_AMT_APR             int64
PAY_AMT_SEPT             int64
PAY_AMT_AUG              int64
PAY_AMT_JUL              int64
PAY_AMT_JUN              int64
PAY_AMT_MAY              int64
PAY_AMT_APR              int64
IsDefaulter              int64
EDUCATION_High School     bool
EDUCATION_University      bool
EDUCATION_others          bool
MARRIAGE_Single           bool
MARRIAGE_others           bool
dtype: object
```

In [38]:
```
# Correlation Heatmap visualization code
plt.figure (figsize= (20,10))
sns.heatmap(df_credit.corr(),linewidths=.5,annot=True,cmap='coolwarm')
plt.show()
```



1. Why did you pick the specific chart?

Correlation heatmaps are graphical representations of the strength of correlations between numerical data.

Correlation plots are used to determine which variables are related to one another and how strong this relationship is.

A correlation plot often has several numerical variables, each represented by a column. Each row represents the relationship between two variables.

favourable numbers imply a favourable relationship, whereas negative values suggest a negative relationship.

2. What is/are the insight(s) found from the chart?

According to the heatmap, some features (for example, PAY_MAY & PAY_APR, BILL_AMT_MAY & BILL_AMT_APR, etc.) are highly associated to each other, but we are not going to eliminate any of them because they contain client transaction details.

Chart - 11 - Pair Plot

In [39]:
```
# Pair Plot visualization code
plt.figure(figsize=(20,40))
sns.pairplot(df_credit,vars=['BILL_AMT_SEPT','BILL_AMT_AUG', 'BILL_AMT_JUL',
plt.show()
```

```
<Figure size 2000x4000 with 0 Axes>
```

In [40]:
```
# Display column types after encoding
print(df_credit.dtypes)
```

```
LIMIT_BAL                int64
SEX                      int64
AGE                      int64
PAY_SEPT                 int64
PAY_AUG                  int64
PAY_JUL                  int64
PAY_JUN                  int64
PAY_MAY                  int64
PAY_APR                  int64
BILL_AMT_SEPT            int64
BILL_AMT_AUG             int64
BILL_AMT_JUL             int64
BILL_AMT_JUN             int64
BILL_AMT_MAY             int64
BILL_AMT_APR             int64
PAY_AMT_SEPT             int64
PAY_AMT_AUG              int64
PAY_AMT_JUL              int64
PAY_AMT_JUN              int64
PAY_AMT_MAY              int64
PAY_AMT_APR              int64
IsDefaulter              int64
EDUCATION_High School     bool
EDUCATION_University      bool
EDUCATION_others          bool
MARRIAGE_Single           bool
MARRIAGE_others           bool
dtype: object
```

1. Why did you pick the specific chart?

The Seaborn Pairplot depicts pairwise relationships between variables within a dataset. This offers a good visualisation and helps us grasp the material by condensing a big quantity of data into a single image. This is critical as we explore and become acquainted with our dataset.

2. What is/are the insight(s) found from the chart?

We can see that the prior month's bill amount is linearly connected, implying that users spend a comparable amount each month. Higher billing amounts from prior months may be deafulters in the future.Where as payment amount is different each month.

5. Hypothesis Testing

Based on your chart experiments, define three hypothetical statements from the dataset. In the next three questions, perform hypothesis testing to obtain final conclusion about the statements through your code and statistical testing.

Answer Here.

Hypothetical Statement - 1

Married male defaults average age is 35 Years.

1. State Your research hypothesis as a null hypothesis and alternate hypothesis.

Null Hypothesis: N = 35

Alternate Hypothesis : N != 35

Test Type: Two Tailed Test

2. Perform an appropriate statistical test.

In [41]:
```
# Perform Statistical Test to obtain P-Value
# Ensure we are filtering correctly based on encoded categories
sample_1 = df_credit[
    (df_credit['MARRIAGE_Single'] == 0) &  # Not single (assumed to be marri
    (df_credit['MARRIAGE_others'] == 0) &  # Not in 'others' category
    (df_credit['SEX'] == 1) &  # Male
    (df_credit['IsDefaulter'] == 1)  # Defaulted payment
]

# Ensure there are enough rows to sample
sample_size = min(1000, len(sample_1))
sample_1 = sample_1.sample(sample_size)

# Hypothesis Testing
N = sample_size  # Use actual sample size
sample_mean = sample_1['AGE'].mean()
std_pop = df_credit['AGE'].std()  # Population standard deviation

# Compute Z-statistic
Z_stat = (sample_mean - 35) / (std_pop / np.sqrt(N))

# Calculate p-value
z_value = norm.cdf(Z_stat, 0, 1)
P_value = 2 * (1 - z_value) if z_value > 0.5 else 2 * z_value

print(f'P Value is {P_value:.4f}')

# Hypothesis Testing Decision
if P_value >= 0.05:
    print('Fail to reject the null hypothesis')
else:
    print('Reject the null hypothesis')
```

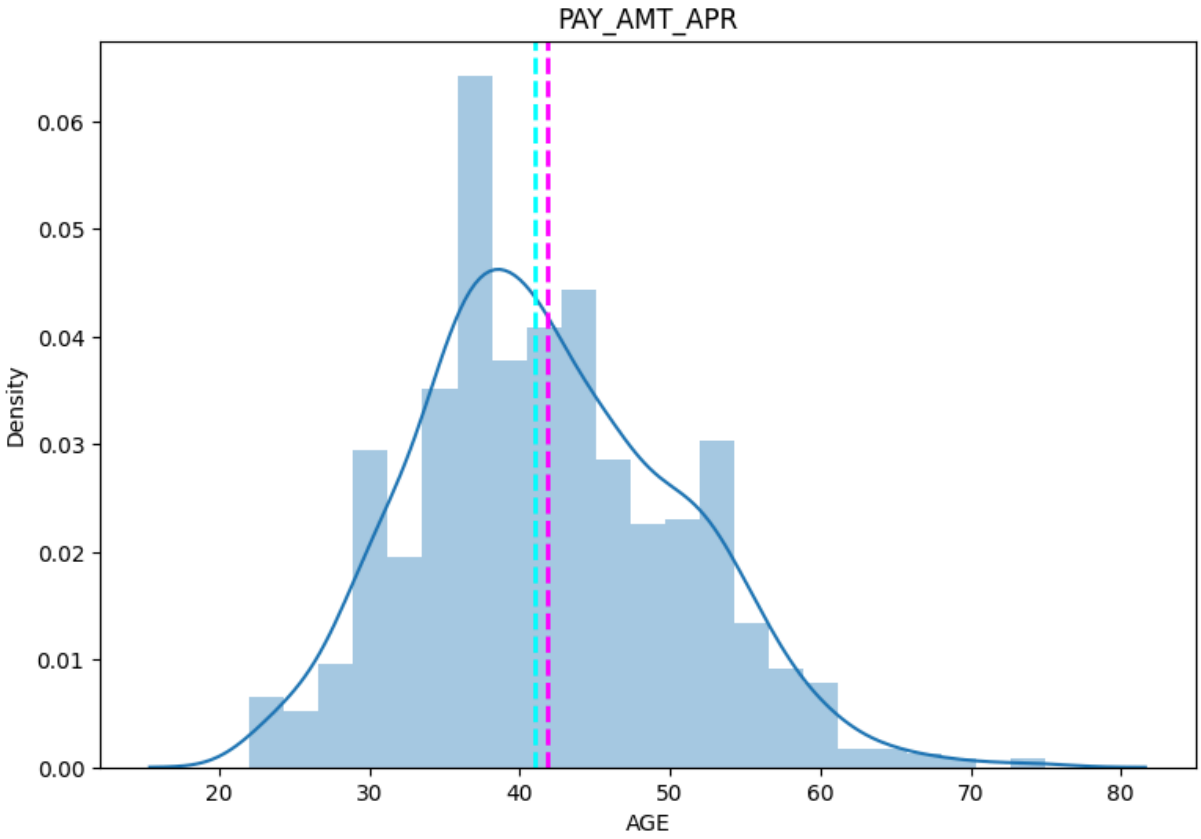
```
P Value is 0.0000
Reject the null hypothesis
```

Which statistical test have you done to obtain P-Value?

I utilised the Z-Test as a statistical test to produce P-Value and discovered that the null hypothesis was rejected and that married male customers defaulting do not have an average age of 35 years.

Why did you choose the specific statistical test?

In [42]:
```
#Distrubution plot for Age feature
fig,ax=plt.subplots(1,1,figsize=(9,6))
feature= (sample_1["AGE"])
sns.distplot(sample_1["AGE"])
ax.axvline(feature.mean(),color='magenta', linestyle='dashed', linewidth=2)
ax.axvline(feature.median(),color='cyan', linestyle='dashed', linewidth=2)
ax.set_title(col)
plt.show()
```



In [43]:
```
mean_median_difference=sample_1["AGE"].mean()- sample_1["AGE"].median()
print("Mean Median Difference is : ",mean_median_difference)
```

```
Mean Median Difference is :  0.8609999999999971
```

As illustrated above, the mean median difference is between 1 and nearly zero. The mean is roughly the same as the median. As a result, it is a Normal Distribution. As a result, I used Z-Test directly.

Hypothetical Statement - 2

Customers defaulting have an average credit limit of 90000.

1. State Your research hypothesis as a null hypothesis and alternate hypothesis.

Null Hypothesis : mean = 90000

Alternate Hypothesise : mean !=90000

Type of Test : Two Tailed test

2. Perform an appropriate statistical test.

In [44]:
```
# Perform Statistical Test to obtain P-Value
# Filter the dataset for defaulters (IsDefaulter == 1)
sample_2 = df_credit[df_credit['IsDefaulter'] == 1].sample(1000, random_stat

# Define population mean for comparison
pop_mean = 90000  # Hypothesized mean LIMIT_BAL

# Compute sample mean
sample_mean = sample_2['LIMIT_BAL'].mean()

# Perform one-sample t-test
t_statistic, P_value = stats.ttest_1samp(sample_2['LIMIT_BAL'], pop_mean)

# Print results
print(f"T-Statistic Value: {t_statistic:.4f}")
print(f"P-Value: {P_value:.4f}")

# Hypothesis Testing Decision
if P_value >= 0.05:
    print("Fail to reject the null hypothesis: No significant difference.")
else:
    print("Reject the null hypothesis: Significant difference.")
```

```
T-Statistic Value: 12.4310
P-Value: 0.0000
Reject the null hypothesis: Significant difference.
```

Which statistical test have you done to obtain P-Value?

I utilised T-Test as a statistical test to generate P-Value and discovered that the null hypothesis was rejected since the customers that defaulted had not an average credit limit of 90000.

Why did you choose the specific statistical test?

In [45]:
```
mean_median_difference=sample_2['LIMIT_BAL'].mean()- sample_2['LIMIT_BAL'].m
print("Mean Median Difference is : ",mean_median_difference)
```

```
Mean Median Difference is :  38400.0
```

The accompanying figure shows that the median is bigger than the mean over ten. As a result, the distribution is positively biassed. Z-Test cannot be conducted on skewed data.

Even with strongly skewed data, t-tests and their related confidence intervals can and should be utilised in investigations with a large sample size.

So, with skewed data, we can utilise the T-test to get a better result. As a result, I used the t - test.

Hypothetical Statement - 3

1. State Your research hypothesis as a null hypothesis and alternate hypothesis.

Null Hypothesis(HO):Defaulter does not depends on Educational qualification.

Alternate Hypothesis(Ha):Defaulter also get affected by Educational qualification.

2. Perform an appropriate statistical test.

In [46]:
```
# Identify Education columns
education_columns = ['EDUCATION_High School', 'EDUCATION_University', 'EDUCA

# Convert one-hot encoded education columns back into a single categorical v
df_credit['EDUCATION'] = df_credit[education_columns].idxmax(axis=1)

# Creating Contingency Table
cont_table = pd.crosstab(df_credit['IsDefaulter'], df_credit['EDUCATION'])

# Perform Chi-Square Test
stat, P_value, dof, expected = chi2_contingency(cont_table)

# Print Results
print(f"Chi-Square P-Value: {P_value:.4f}")
if P_value >= 0.05:
    print("Fail to reject the null hypothesis: No significant association.")
else:
    print("Reject the null hypothesis: Significant association.")
```

```
Chi-Square P-Value: 0.0000
Reject the null hypothesis: Significant association.
```

Which statistical test have you done to obtain P-Value?

I used chi-square test to generate P_value and discoverd null hypothesis was rejected since defaulter does not depend on educational qualification.

Why did you choose the specific statistical test?

Here we have to test difference between defaulter and non defaulter with educational qualification.The Chi-Square test is a statistical procedure for determining the difference between observed and expected data.

6. Feature Engineering & Data Pre-processing

In [47]:
```
#copy the original data before applying feature engineering
df=df_credit.copy()
```

1. Handling Missing Values

In [48]:
```
# Handling Missing Values & Missing Value Imputation

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

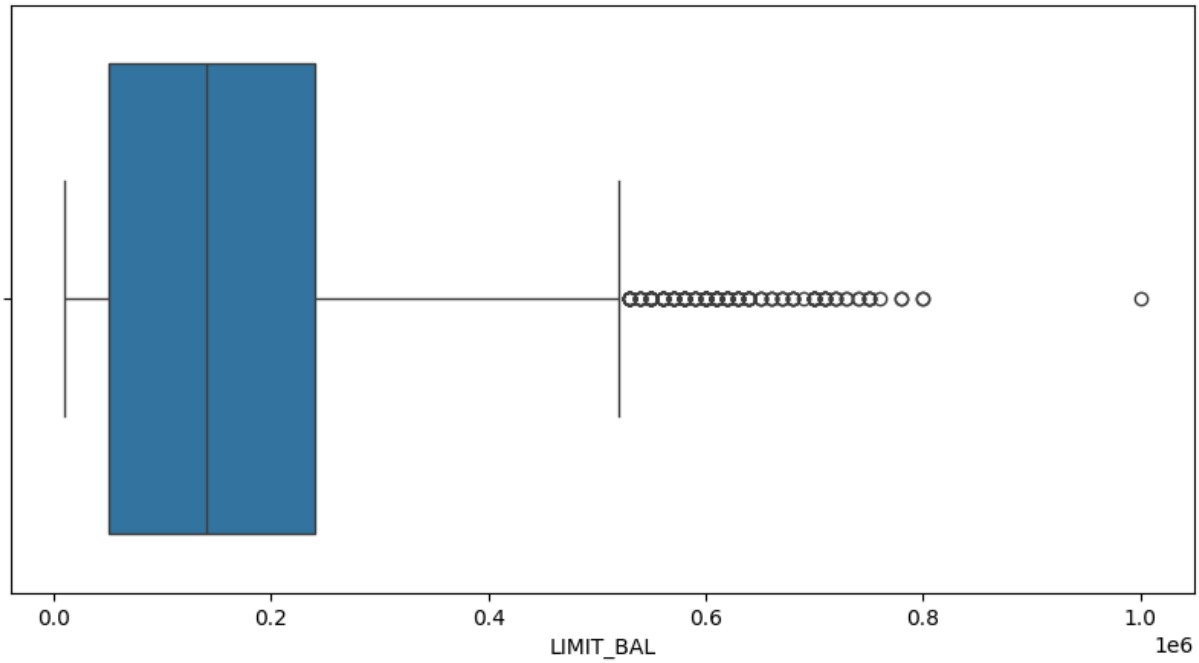
Out[48]:
```
LIMIT_BAL                 0
SEX                       0
AGE                       0
PAY_SEPT                  0
PAY_AUG                   0
PAY_JUL                   0
PAY_JUN                   0
PAY_MAY                   0
PAY_APR                   0
BILL_AMT_SEPT             0
BILL_AMT_AUG              0
BILL_AMT_JUL              0
BILL_AMT_JUN              0
BILL_AMT_MAY              0
BILL_AMT_APR              0
PAY_AMT_SEPT              0
PAY_AMT_AUG               0
PAY_AMT_JUL               0
PAY_AMT_JUN               0
PAY_AMT_MAY               0
PAY_AMT_APR               0
IsDefaulter               0
EDUCATION_High School     0
EDUCATION_University      0
EDUCATION_others          0
MARRIAGE_Single           0
MARRIAGE_others           0
EDUCATION                 0
dtype: int64
```

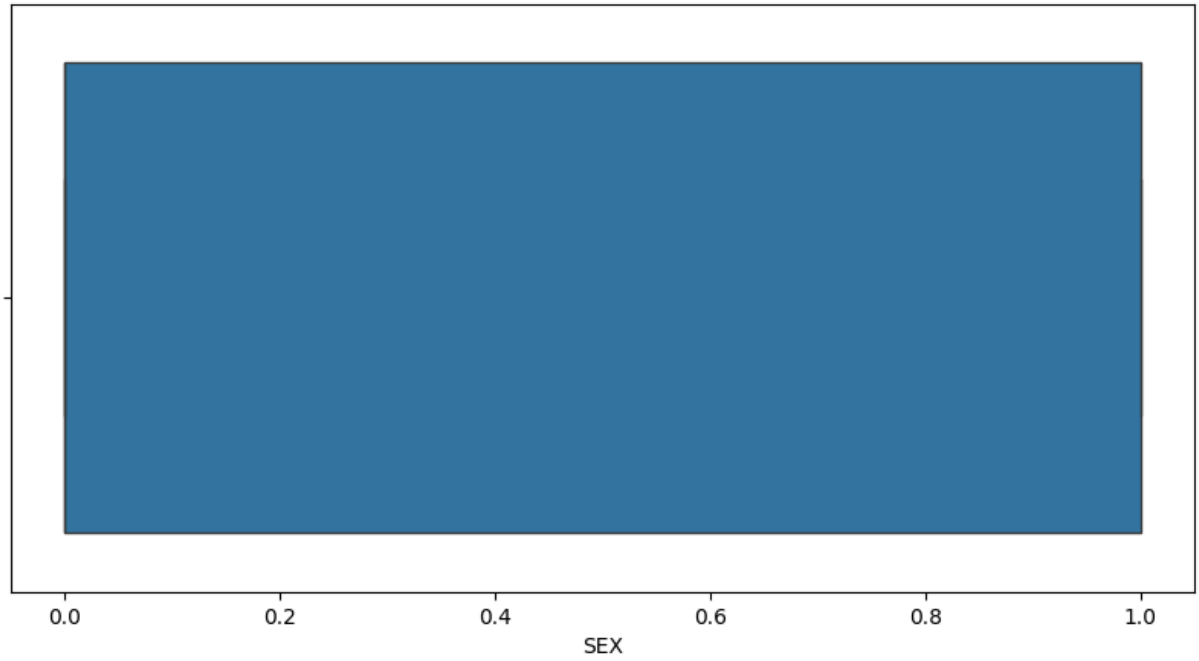
What all missing value imputation techniques have you used and why did you use those techniques?

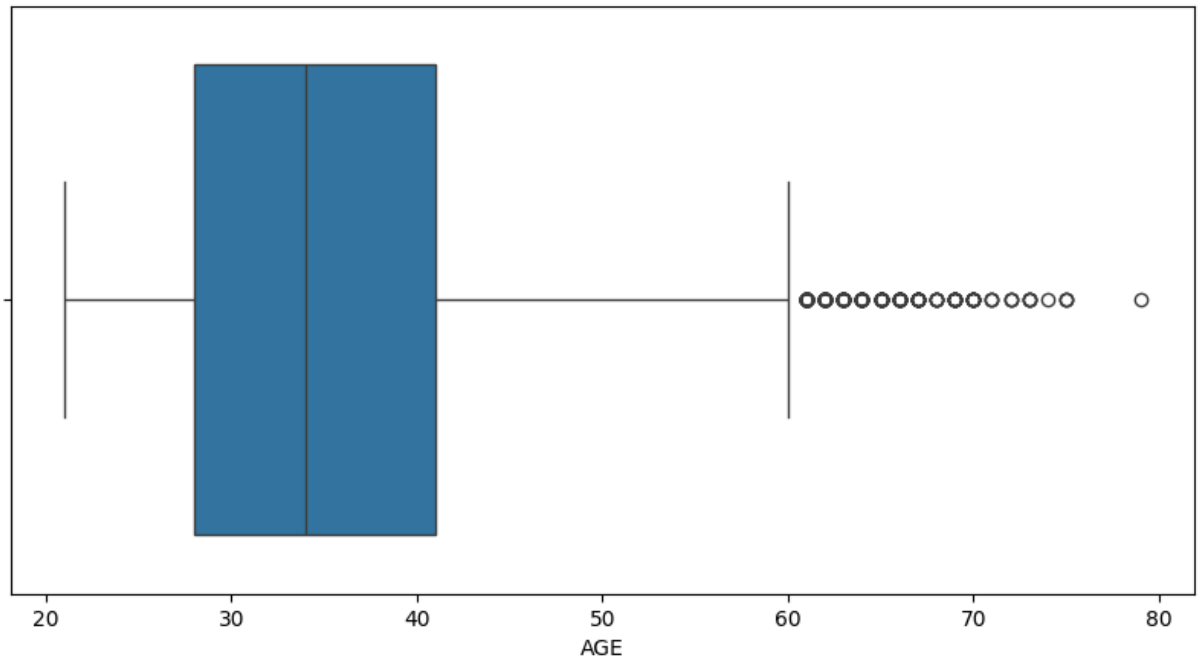
There is no missing value in the dataset to handle.

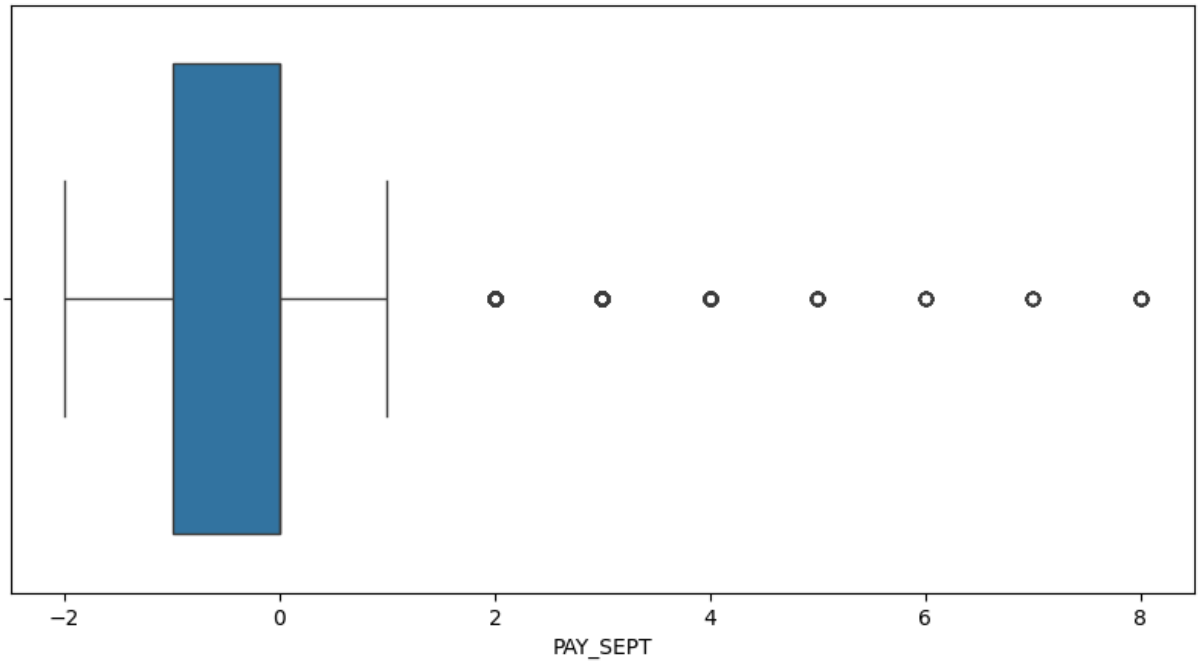
2. Handling Outliers

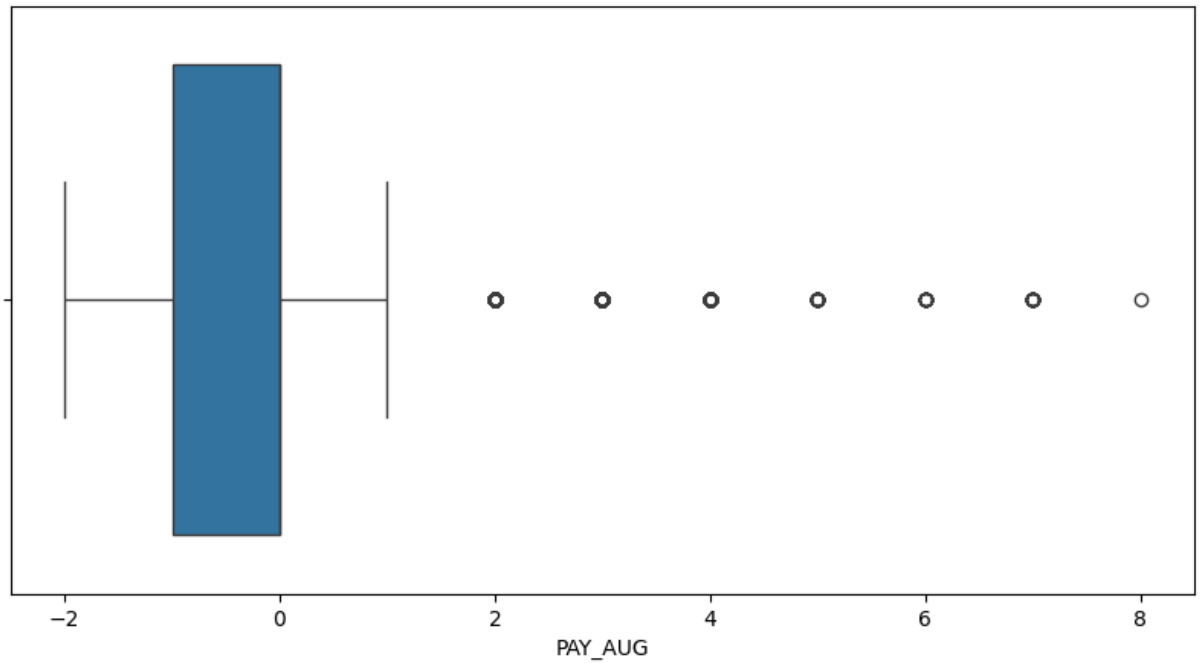
In [49]:
```
# Handling Outliers & Outlier treatments
#Checking outliers in dataset using Boxplot
for col in df.describe().columns:
  plt.figure(figsize=(10,5))
  sns.boxplot(x=df[col])
```

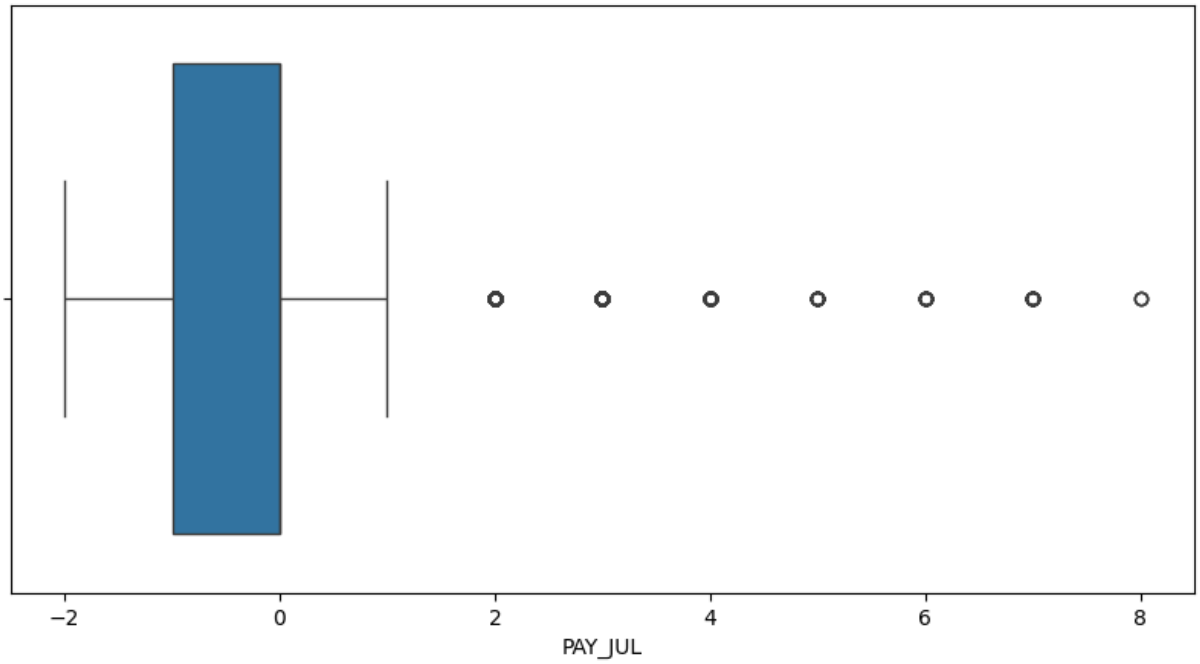


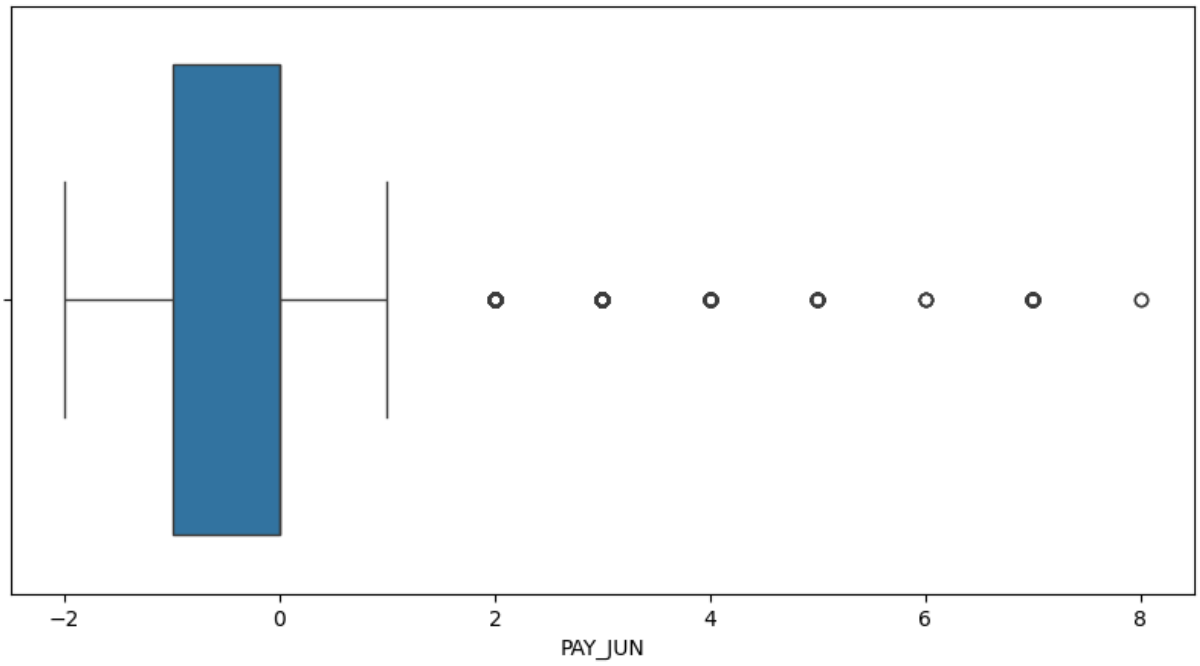


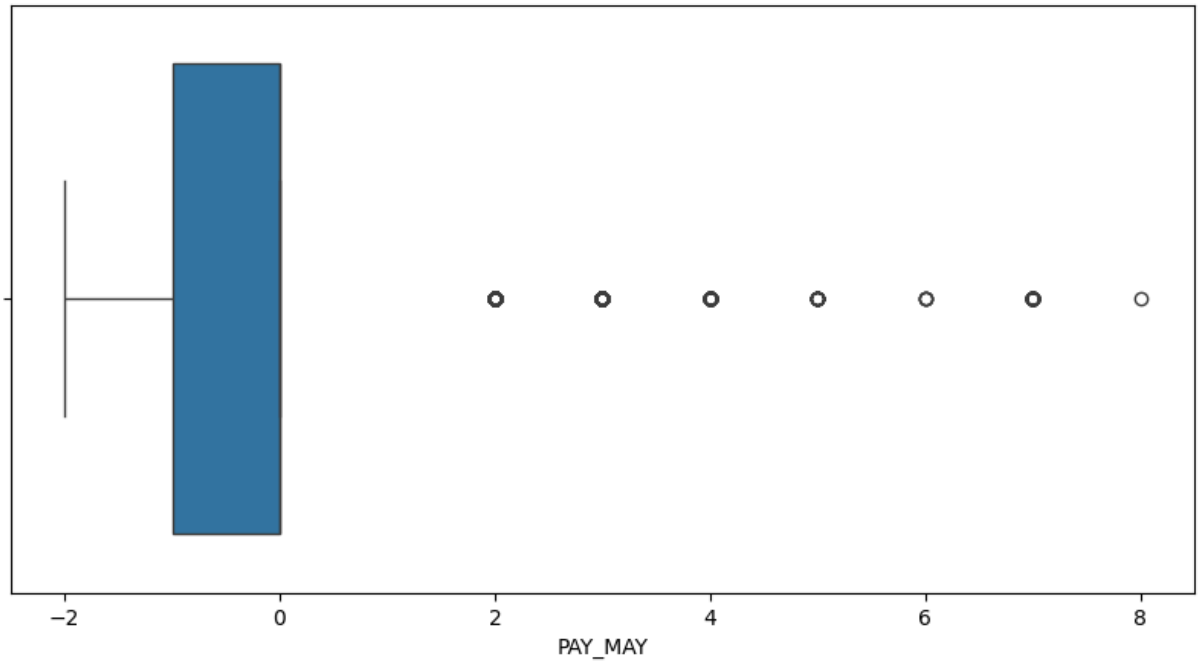

20
30
40
50
60
70
80
AGE

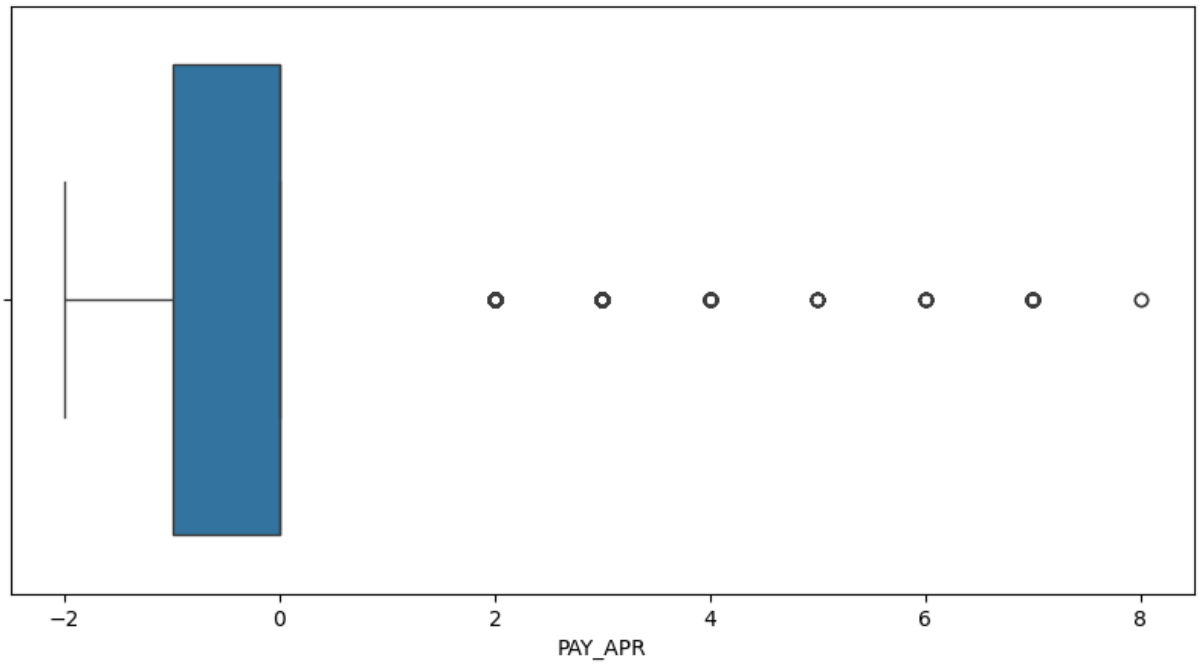

−2
0
2
4
6
8
PAY_SEPT

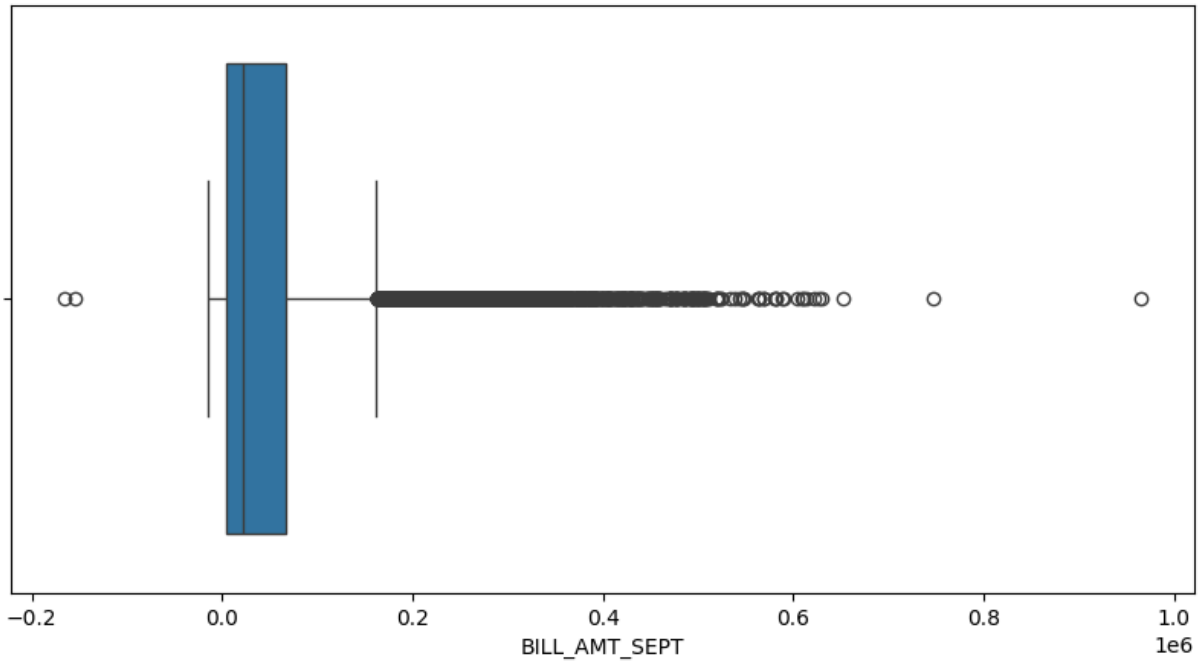

−2
0
2
4
6
8
PAY_AUG

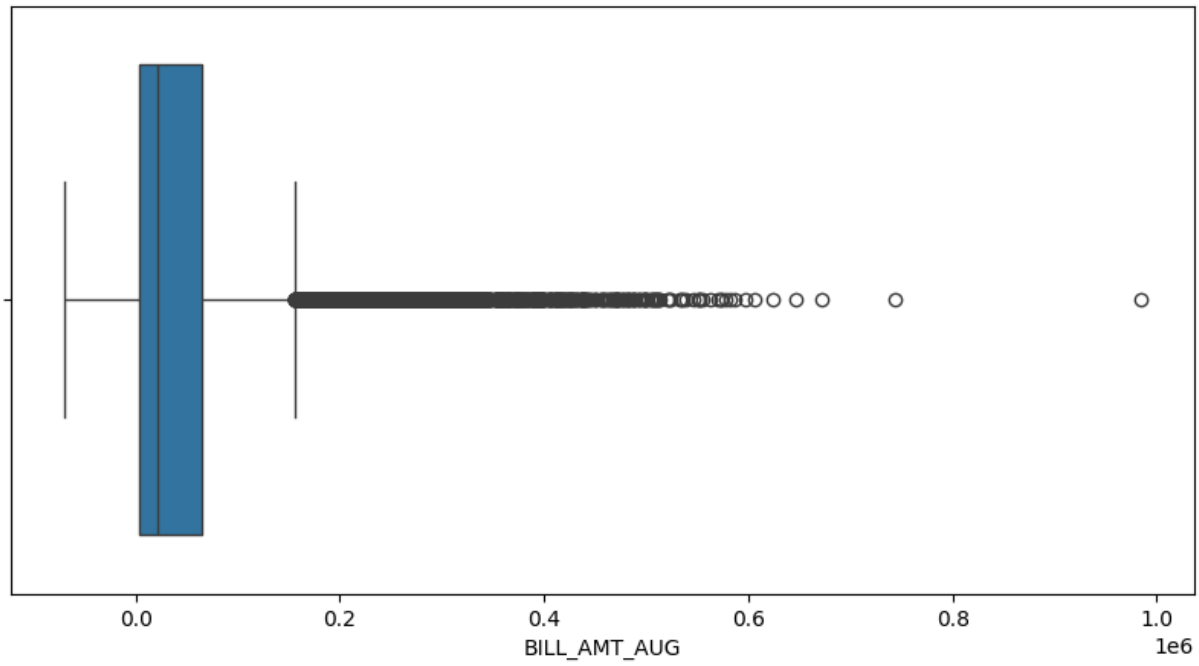

−2
0
2
4
6
8
PAY_JUL

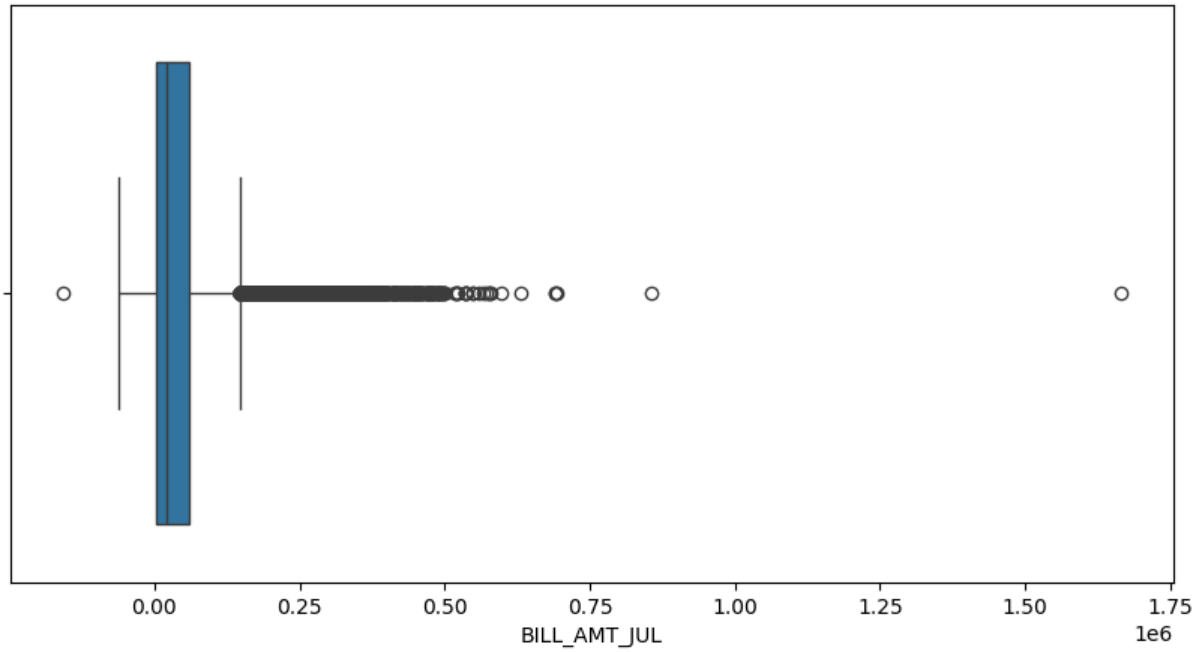

−2
0
2
4
6
8
PAY_JUN

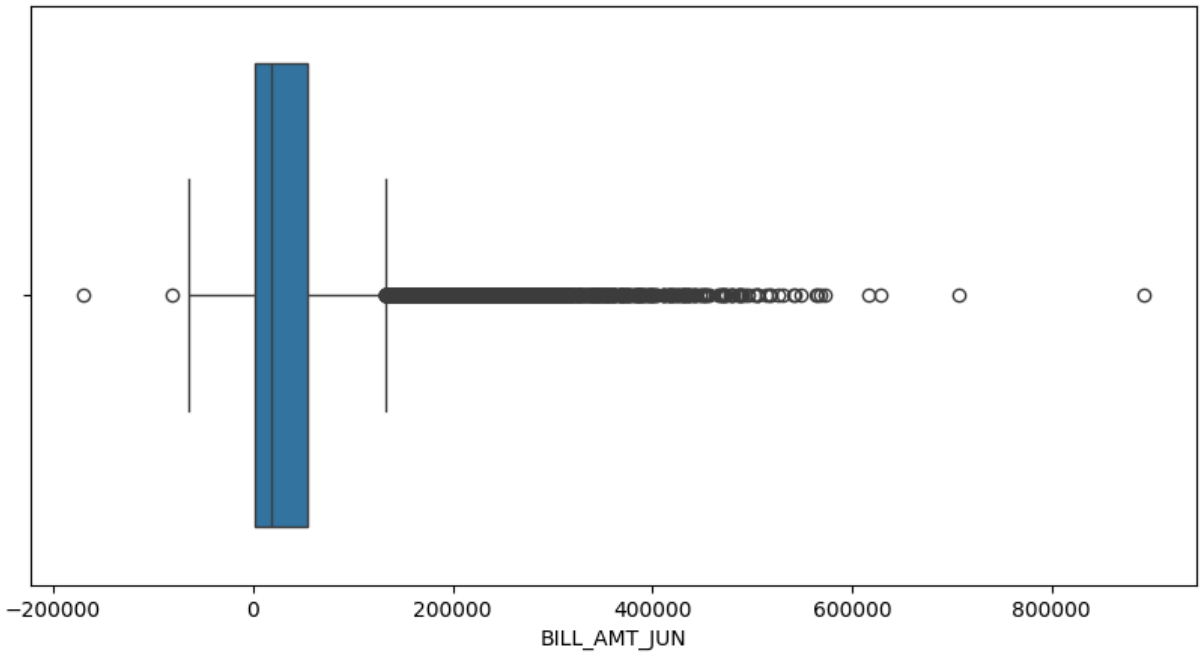

−2
0
2
4
6
8
PAY_MAY

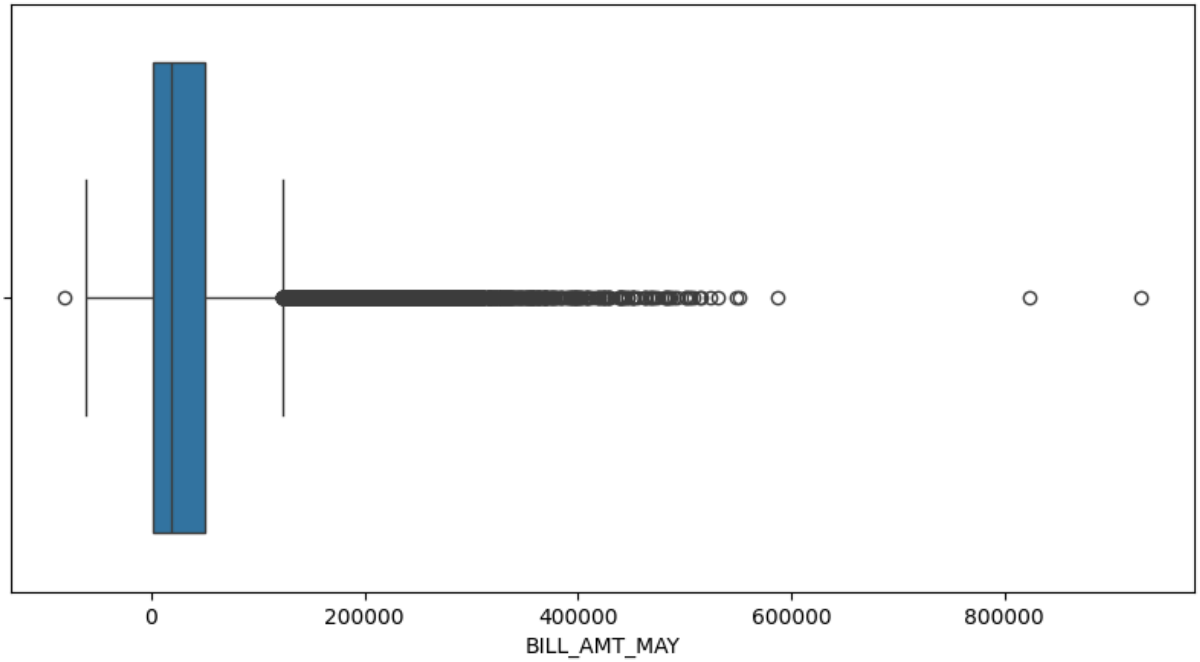

−2
0
2
4
6
8
PAY_APR

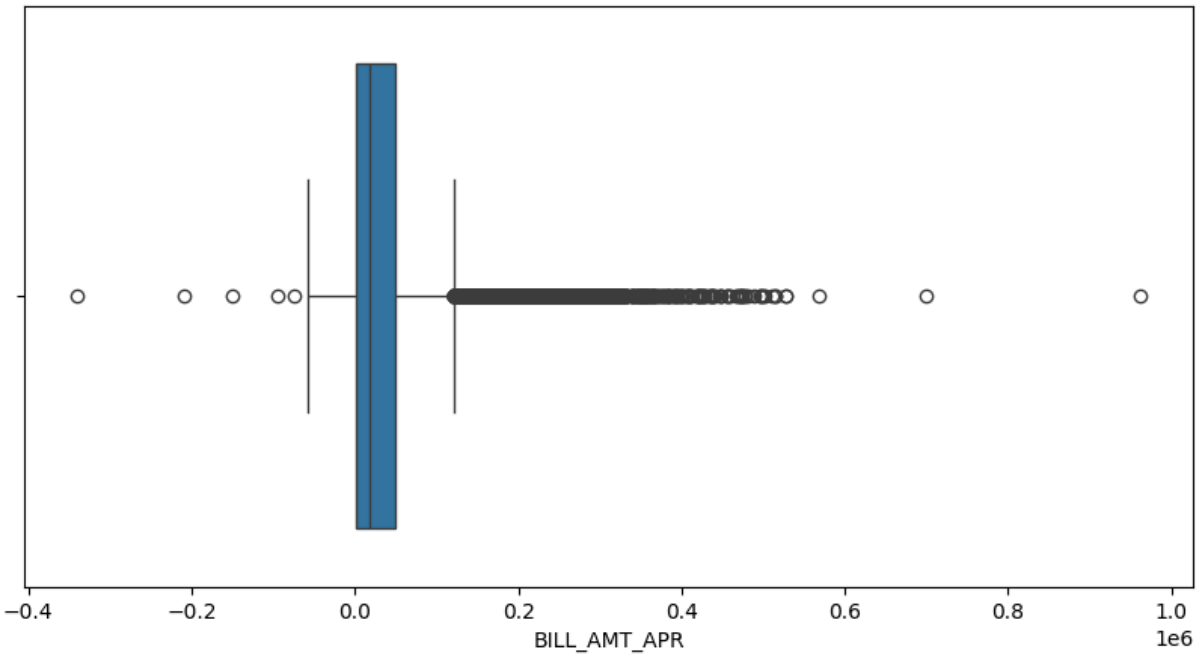

−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1e6
BILL_AMT_SEPT

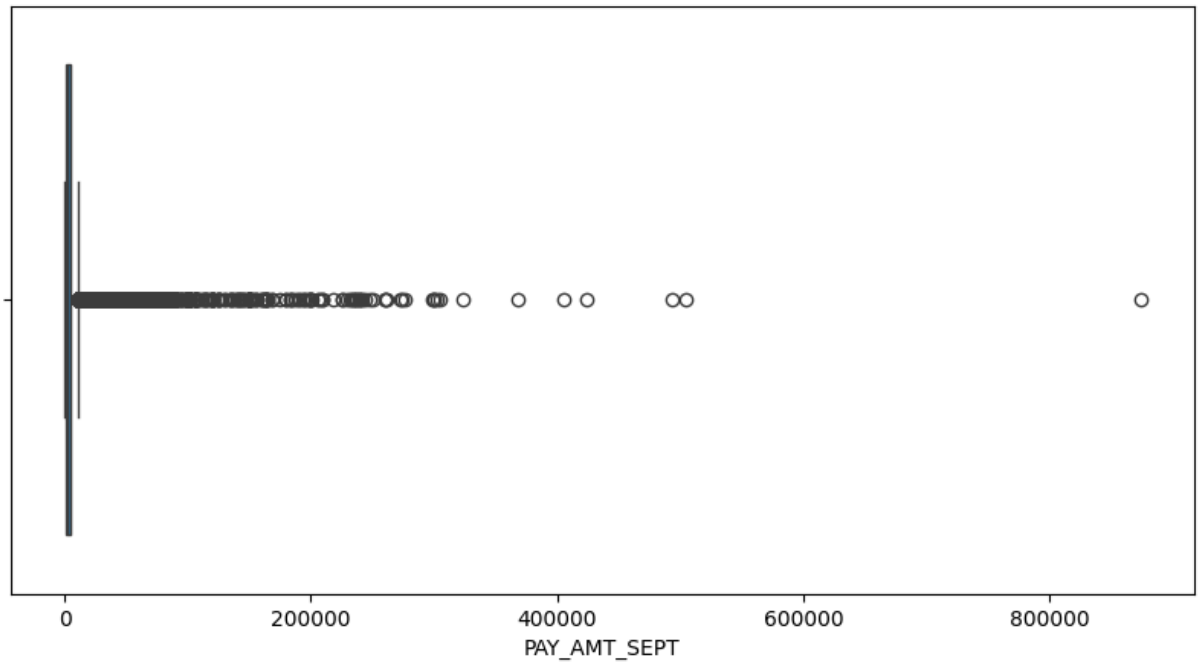

0.0
0.2
0.4
0.6
0.8
1.0
BILL_AMT_AUG
1e6

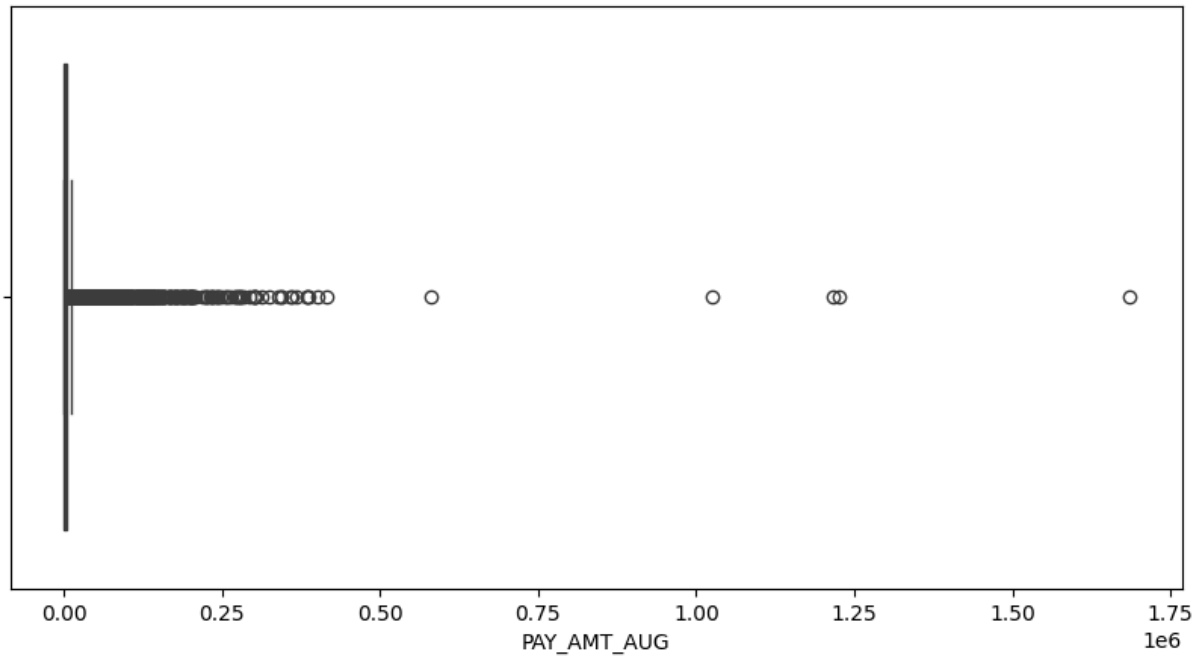

0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
BILL_AMT_JUL
1e6

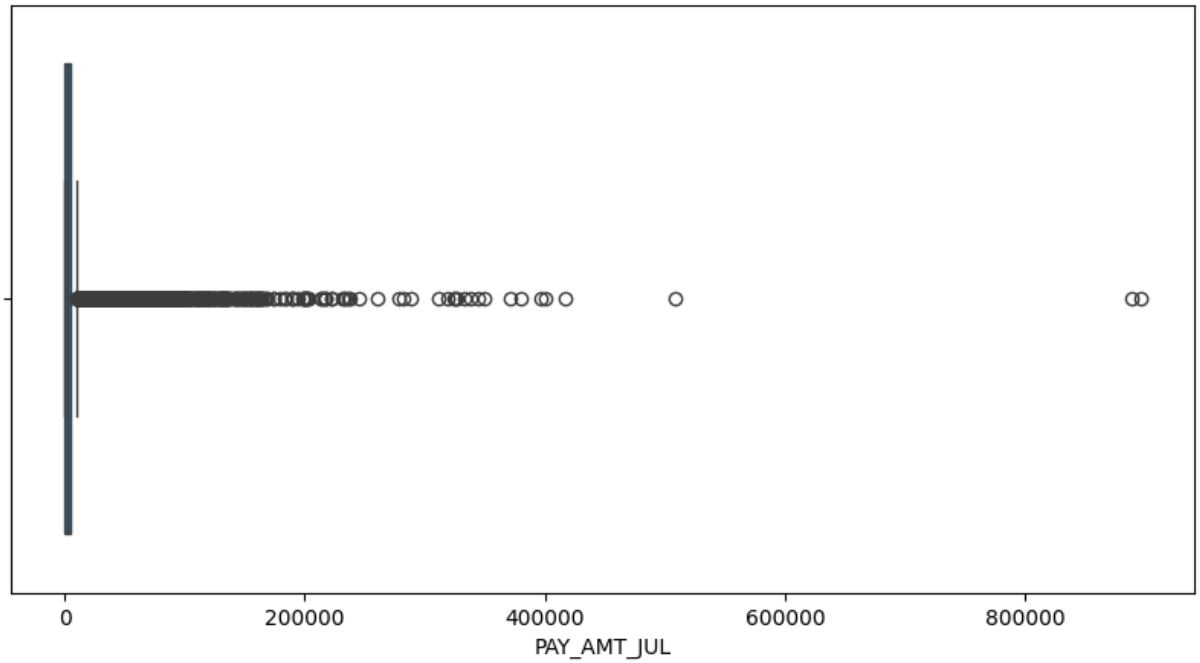

−200000
0
200000
400000
600000
800000
BILL_AMT_JUN

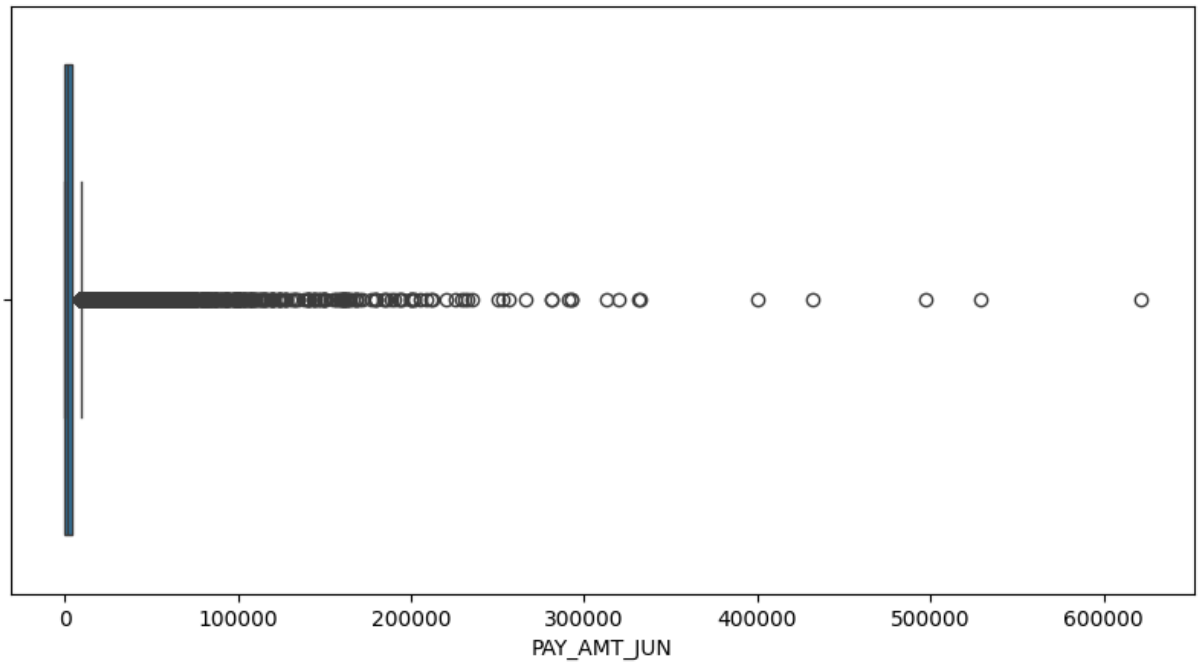

0
200000
400000
600000
800000
BILL_AMT_MAY

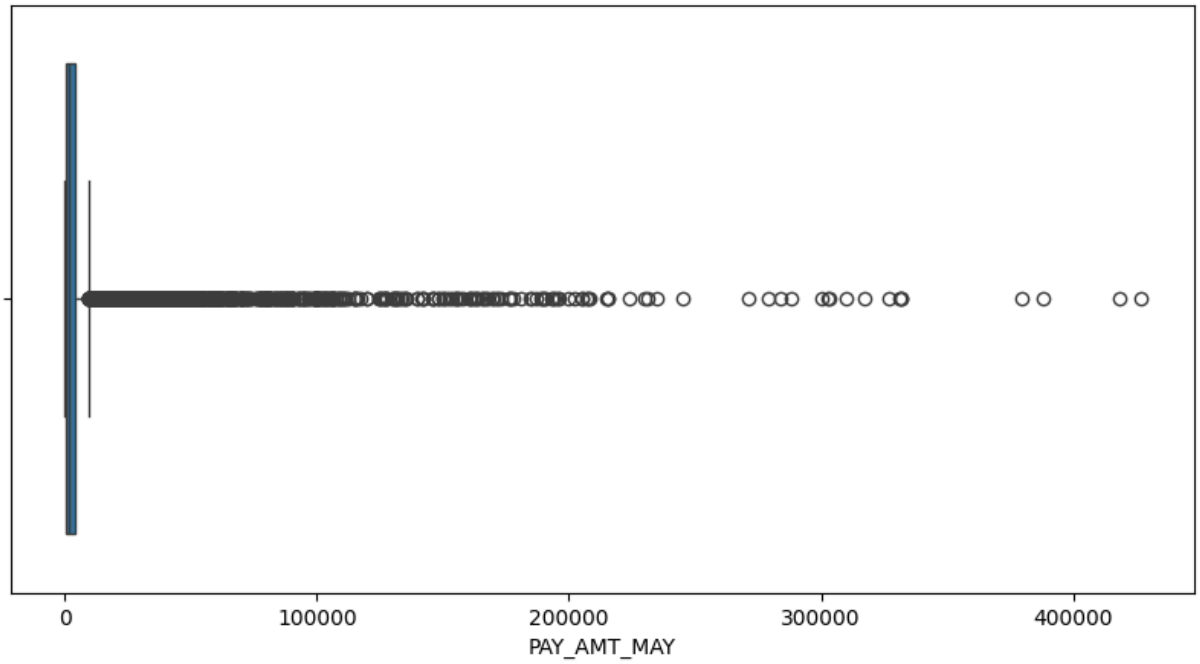

−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1e6
BILL_AMT_APR

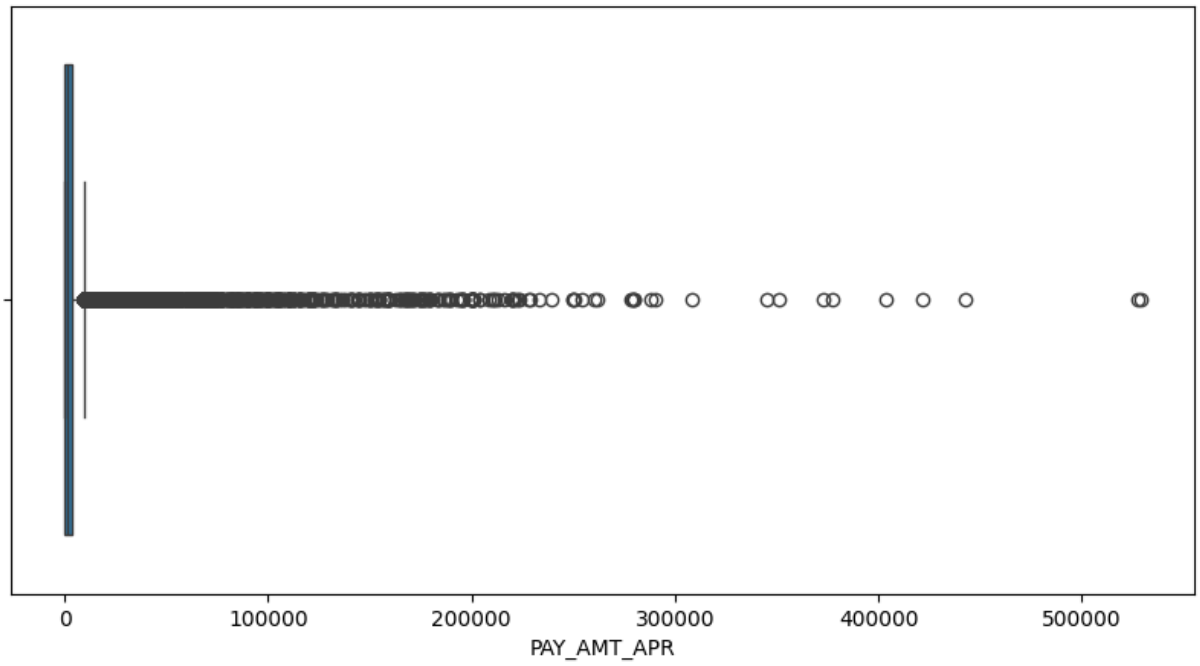

0
200000
400000
600000
800000
PAY_AMT_SEPT

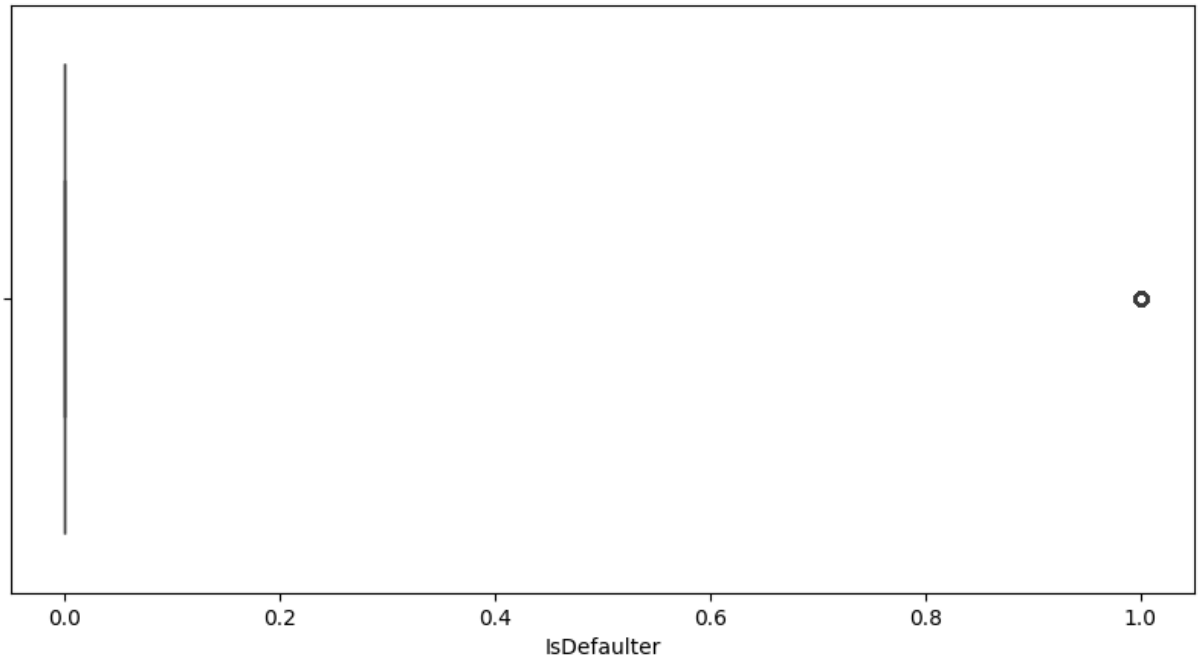

0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e6
PAY_AMT_AUG


0
200000
400000
600000
800000
PAY_AMT_JUL


0
100000
200000
300000
400000
500000
600000
PAY_AMT_JUN


0
100000
200000
300000
400000
PAY_AMT_MAY





What all outlier treatment techniques have you used and why did you use those techniques?

I used the boxplot to look for outliers in the dataset. Some points appear to be outliers in numeric features that are primarily connected to quantity, and we can use the IQR method to treat these outliers without knowing the exact limit of these columns from the bank, and addressing these outlying amounts will result in data loss.And there is a potential that the amount that was maximal and appeared to be an outlier was the true value for that characteristic, thus I did not employ any approaches to treat these outlying numbers.

3. Categorical Encoding

In [50]:
```
#make the dependable column categorical to binary
df.replace({'IsDefaulter' : {'Yes' : 1, 'No' : 0}}, inplace = True)
```

In [51]:
```
# check how many colmun contain categorical data
sum(df.dtypes==object)
```

Out[51]: 1

In [52]:
```
#Creating Dummies
df=pd.get_dummies(df)
```

In [53]:
```
df.head()
```

Out[53]:

	LIMIT_BAL	SEX	AGE	PAY_SEPT	PAY_AUG	PAY_JUL	PAY_JUN	PAY_MAY	PAY_A
0	20000	0	24	2	2	-1	-1	-2	
1	120000	0	26	-1	2	0	0	0	
2	90000	0	34	0	0	0	0	0	
3	50000	0	37	0	0	0	0	0	
4	50000	1	57	-1	0	-1	0	0	

5 rows × 30 columns

What all categorical encoding techniques have you used & why did you use those techniques?

One-shot encoding is a critical step in preparing your dataset for machine learning. Your categorical data is converted to a binary vector representation via one-shot encoding. Pandas get dummies makes this really simple. This is critical for working with many machine learning algorithms that accept only numerical inputs, such as decision trees and support vector machines. This indicates that a new column is formed for each unique value in a column. Depending on whether the value matches the column header, the values in this column are represented as 1s or 0s.

4. Textual Data Preprocessing

(It's mandatory for textual dataset i.e., NLP, Sentiment Analysis, Text Clustering etc.)

1. Expand Contraction

In [54]:
```
# Expand Contraction
```

2. Lower Casing

In [55]:
```
# Lower Casing
```

3. Removing Punctuations

In [56]:
```
# Remove Punctuations
```

4. Removing URLs & Removing words and digits contain digits.

In [57]:
```
# Remove URLs & Remove words and digits contain digits
```

5. Removing Stopwords & Removing White spaces

In [58]:
```
# Remove Stopwords
```

In [59]:
```
# Remove White spaces
```

6. Rephrase Text

In [60]:
```
# Rephrase Text
```

7. Tokenization

In [61]:
```
# Tokenization
```

8. Text Normalization

In [62]:
```
# Normalizing Text (i.e., Stemming, Lemmatization etc.)
```

Which text normalization technique have you used and why?

Answer Here.

9. Part of speech tagging

In [63]:
```
# POS Taging
```

10. Text Vectorization

In [64]:
```
# Vectorizing Text
```

Which text vectorization technique have you used and why?

Answer Here.

4. Feature Manipulation & Selection

1. Feature Manipulation

There is no need for feature manipulation or feature selection because only previous month's payment status and previous month's bill amount have a high correlation and cannot be dropped or manipulated because they are the most significant parameters of detect defaulter.

In [65]:
```
# Manipulate Features to minimize feature correlation and create new feature
```

2. Feature Selection

In [66]:
```
#create a function to check multicolinearity
from statsmodels.stats.outliers_influence import variance_inflation_factor
def calc_vif(X):

    # Calculating VIF
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [round(variance_inflation_factor(X.values, i),2) for i in r

    return(vif)
```

In [67]:
```
#check multicolinearuty
calc_vif(df[[i for i in df.describe().columns if i not in ['IsDefaulter','MA
```

Out[67]:

	variables	VIF
10	BILL_AMT_AUG	38.21
13	BILL_AMT_MAY	35.98
11	BILL_AMT_JUL	31.78
12	BILL_AMT_JUN	29.55
14	BILL_AMT_APR	21.42
9	BILL_AMT_SEPT	20.80
7	PAY_MAY	4.99
6	PAY_JUN	4.44
0	LIMIT_BAL	3.84
5	PAY_JUL	3.73
2	AGE	3.48
8	PAY_APR	3.46
4	PAY_AUG	3.21
16	PAY_AMT_AUG	2.38
3	PAY_SEPT	1.92
15	PAY_AMT_SEPT	1.91
17	PAY_AMT_JUL	1.91
19	PAY_AMT_MAY	1.85
18	PAY_AMT_JUN	1.80
1	SEX	1.66
20	PAY_AMT_APR	1.27

In [68]:
```
#drooping sex_female column
df.drop(['SEX_Female'],axis=1,inplace=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[68], line 2
      1 #drooping sex_female column
----> 2 df.drop(['SEX_Female'],axis=1,inplace=True)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-p
ackages/pandas/core/frame.py:5581, in DataFrame.drop(self, labels, axis, inde
x, columns, level, inplace, errors)
   5433 def drop(
   5434     self,
   5435     labels: IndexLabel | None = None,
   (...)
   5442     errors: IgnoreRaise = "raise",
   5443 ) -> DataFrame | None:
   5444     """
   5445     Drop specified labels from rows or columns.
   5446
   (...)
   5579             weight  1.0     0.8
   5580     """
-> 5581     return super().drop(
   5582         labels=labels,
   5583         axis=axis,
   5584         index=index,
   5585         columns=columns,
   5586         level=level,
   5587         inplace=inplace,
   5588         errors=errors,
   5589     )

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-p
ackages/pandas/core/generic.py:4788, in NDFrame.drop(self, labels, axis, inde
x, columns, level, inplace, errors)
   4786 for axis, labels in axes.items():
   4787     if labels is not None:
-> 4788         obj = obj._drop_axis(labels, axis, level=level, errors=error
s)
   4790 if inplace:
   4791     self._update_inplace(obj)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-p
ackages/pandas/core/generic.py:4830, in NDFrame._drop_axis(self, labels, axi
s, level, errors, only_slice)
   4828         new_axis = axis.drop(labels, level=level, errors=errors)
   4829     else:
-> 4830         new_axis = axis.drop(labels, errors=errors)
   4831     indexer = axis.get_indexer(new_axis)
   4833 # Case for non-unique axis
   4834 else:

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-p
ackages/pandas/core/indexes/base.py:7070, in Index.drop(self, labels, errors)
   7068 if mask.any():
   7069     if errors != "ignore":
-> 7070         raise KeyError(f"{labels[mask].tolist()} not found in axis")
```

```
   7071     indexer = indexer[~mask]
   7072 return self.delete(indexer)

KeyError: "['SEX_Female'] not found in axis"
```

What all feature selection methods have you used and why?

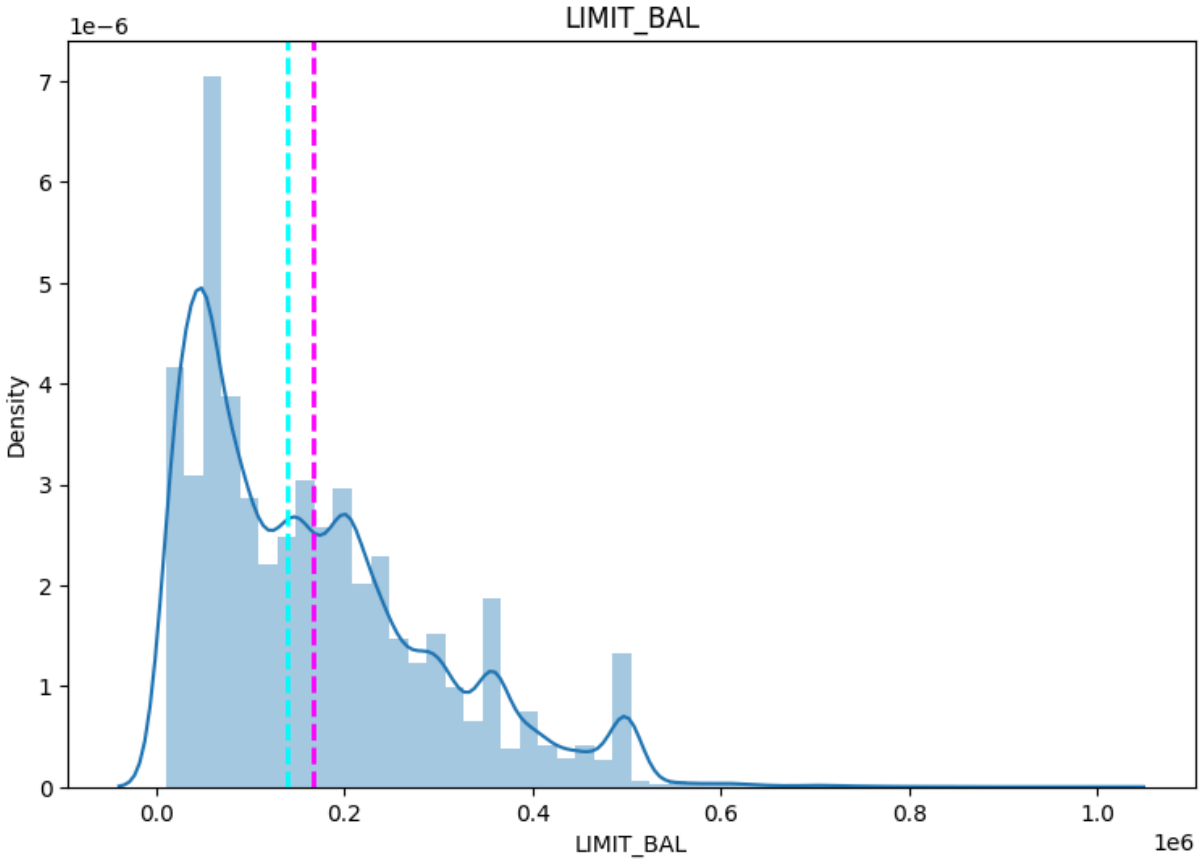
The Variance Inflation Factor (VIF) is used to detect multicollinearity. Variance inflation factors (VIF) quantify how much the variance of predicted regression coefficients is inflated when the predictor variables are not linearly connected.

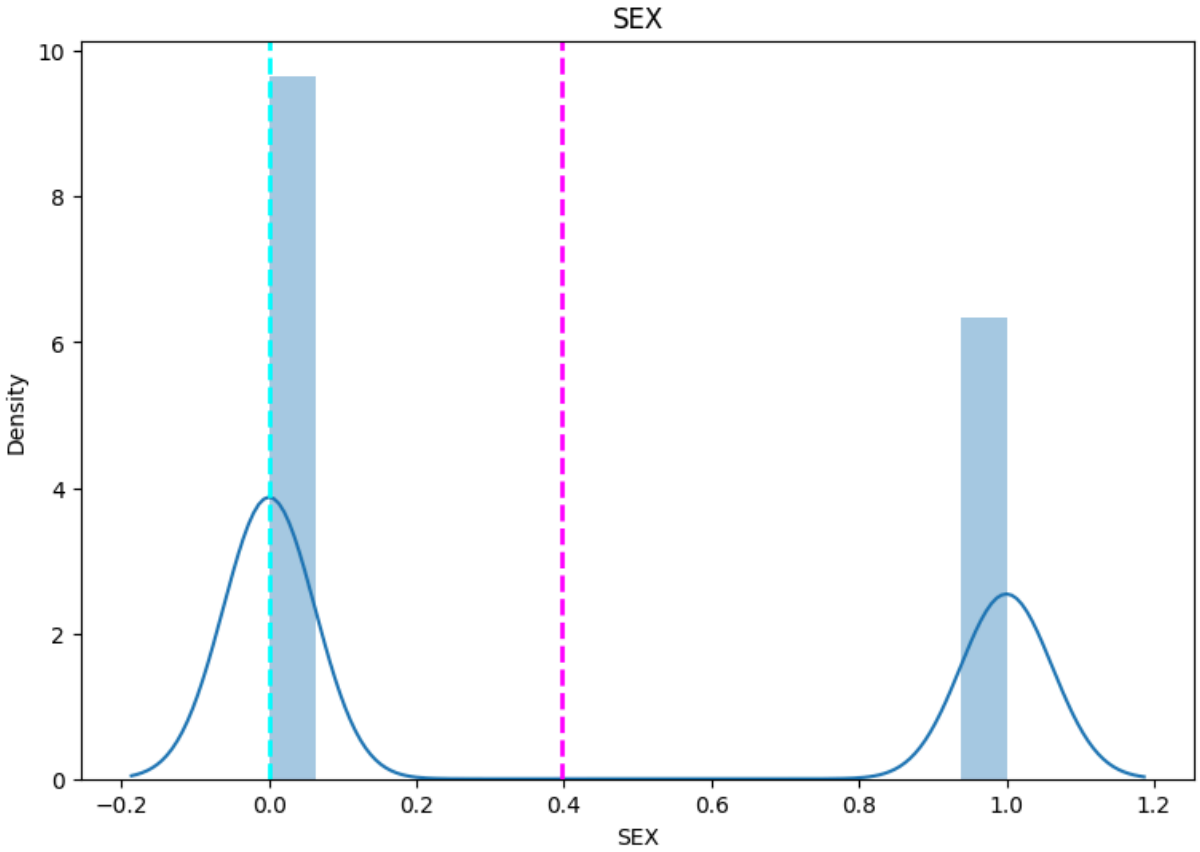
Which all features you found important and why?

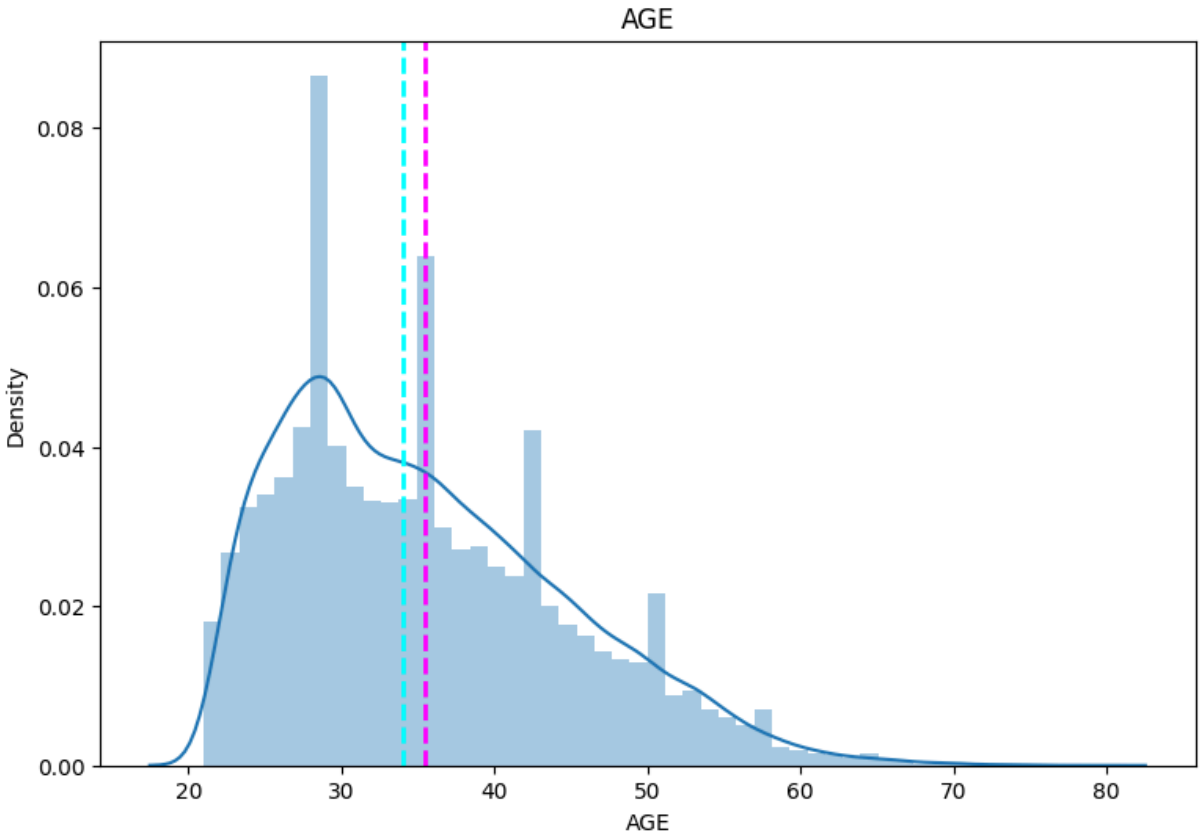
Even though the vif value of a few columns is larger than 10, we select all features since those columns are the most critical features and we cannot afford to lose them.Onle SEX_Female column we have dropped because it is unnecessary of keeping two same binary column.

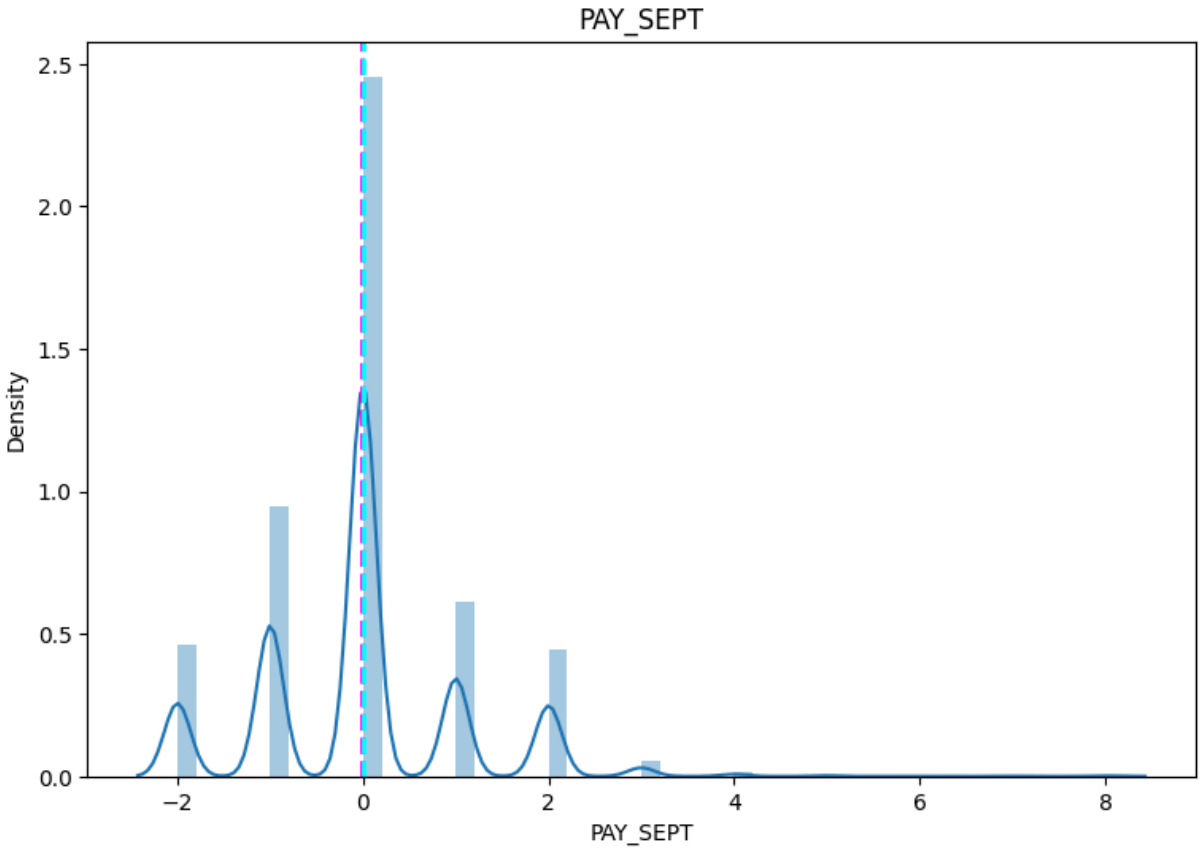
5. Data Transformation

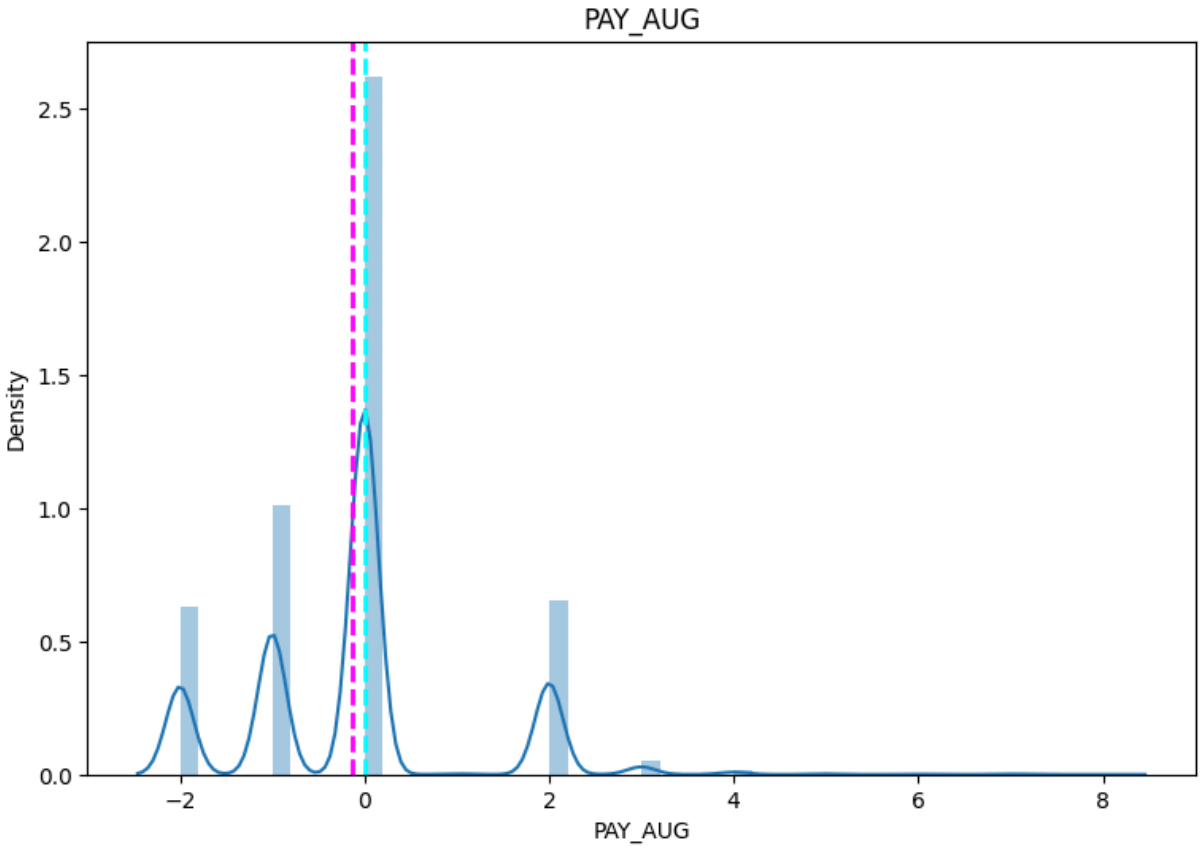
In [69]:
```
#check distribution  of all independent features
for col in df.describe().columns:
  fig=plt.figure(figsize=(9,6))
  ax=fig.gca()
  feature= (df[col])
  sns.distplot(df[col])
  ax.axvline(feature.mean(),color='magenta', linestyle='dashed', linewidth=2
  ax.axvline(feature.median(),color='cyan', linestyle='dashed', linewidth=2)
  ax.set_title(col)
plt.show()
```

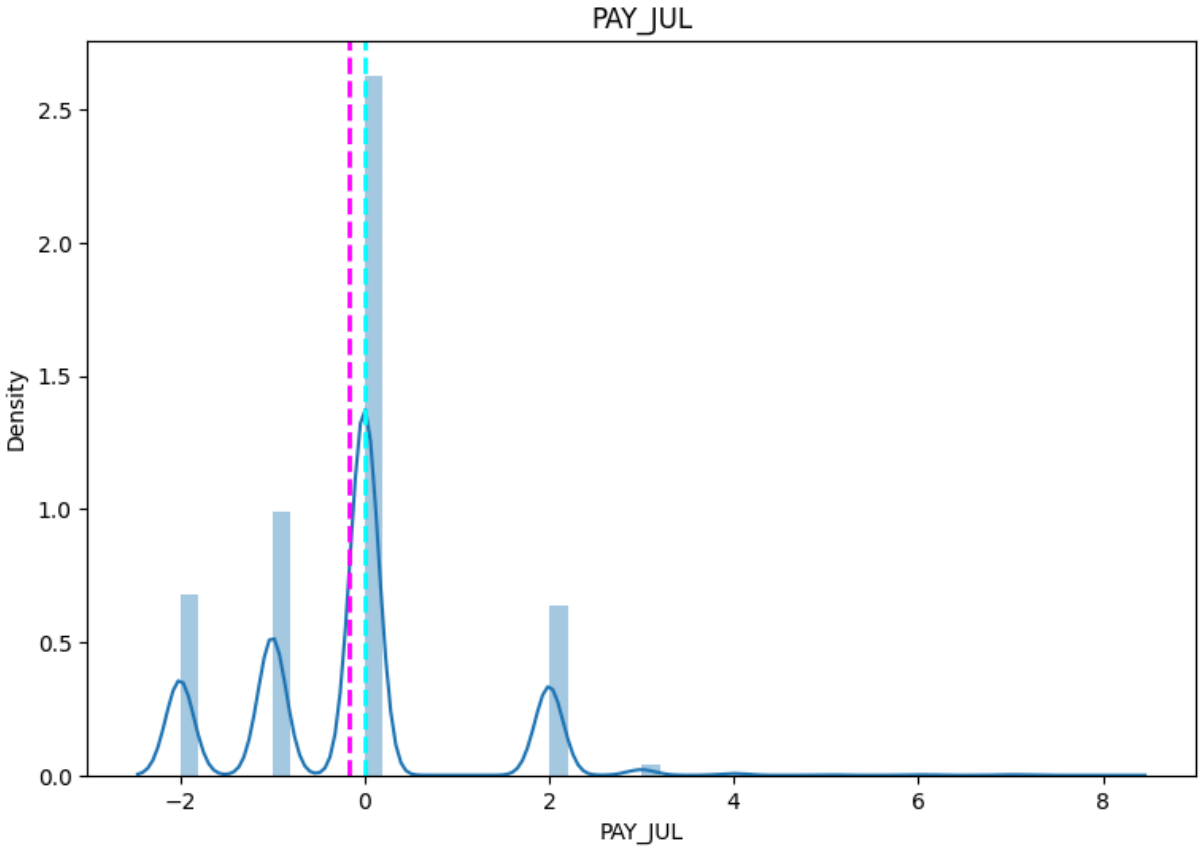

LIMIT_BAL
1e−6
Density
LIMIT_BAL
1e6

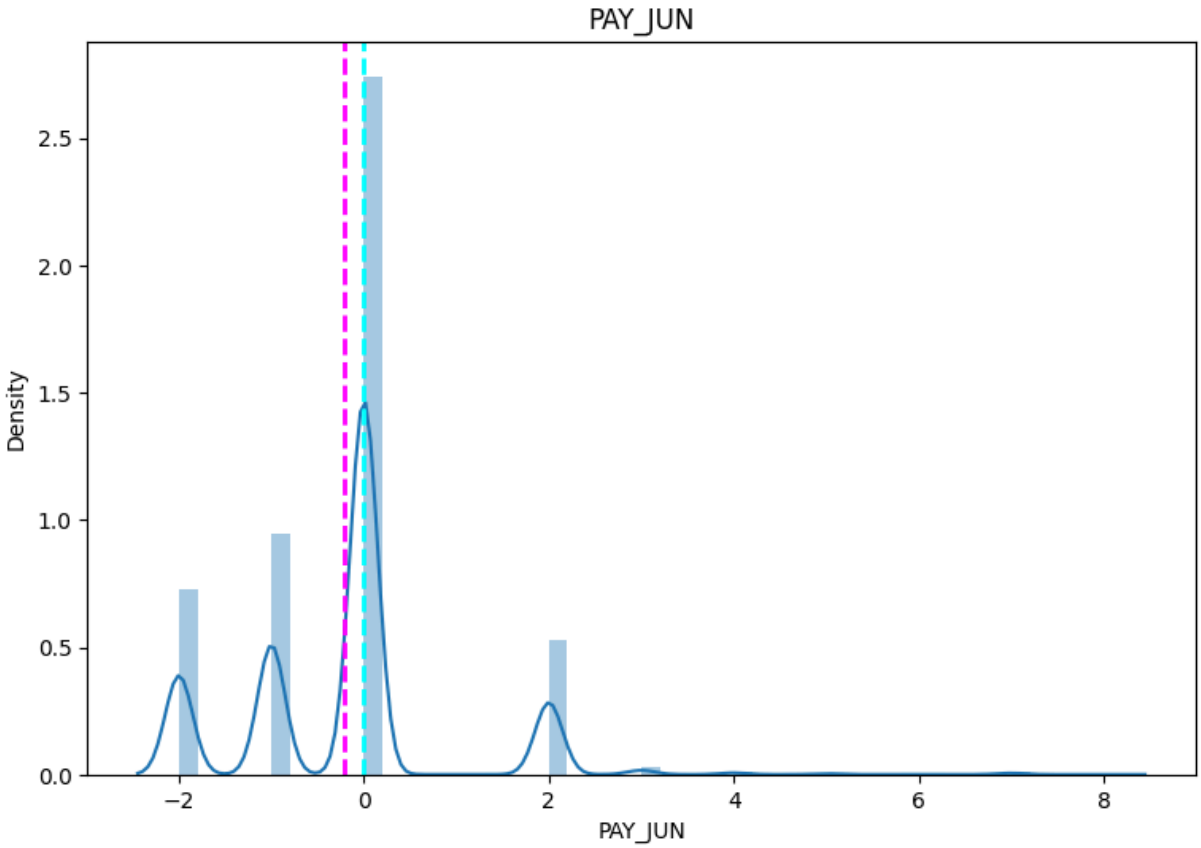

SEX
Density
SEX

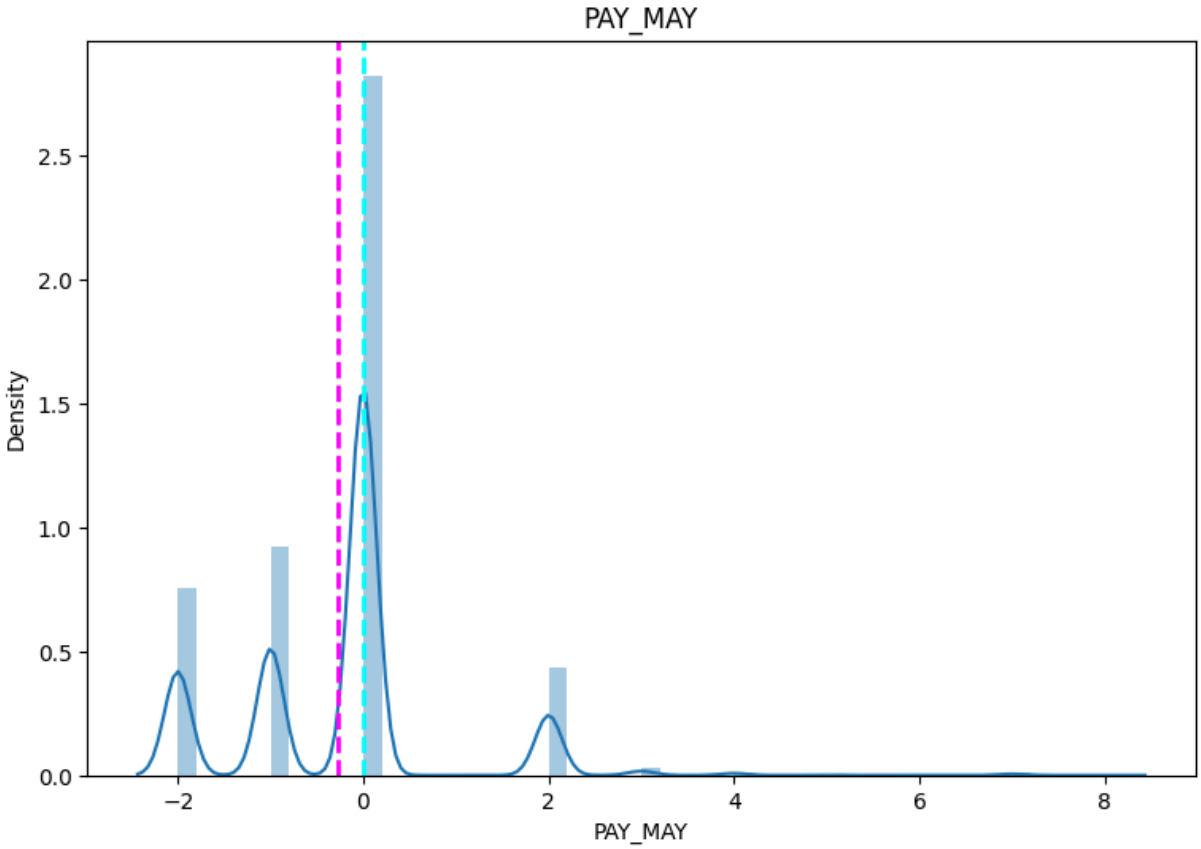

AGE
Density
0.00
0.02
0.04
0.06
0.08
20
30
40
50
60
70
80
AGE

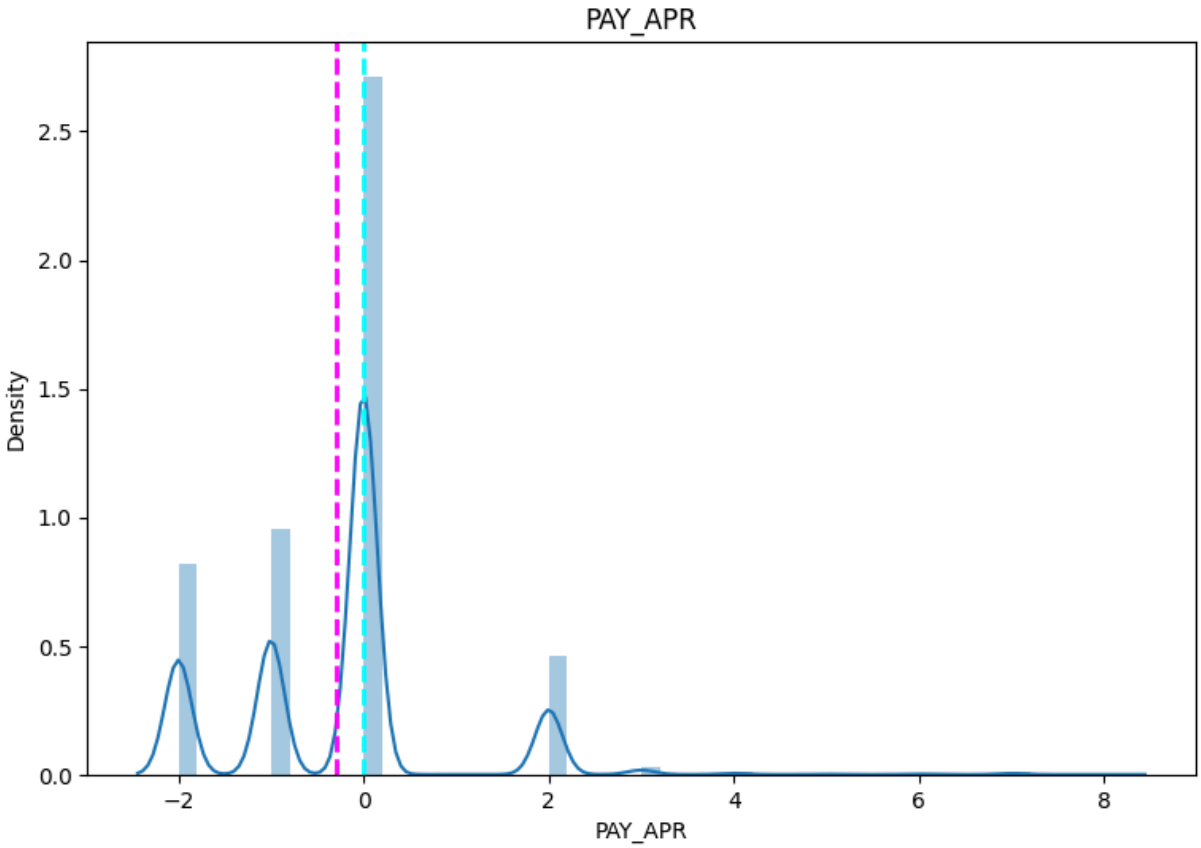

PAY_SEPT
Density
0.0
0.5
1.0
1.5
2.0
2.5
−2
0
2
4
6
8
PAY_SEPT

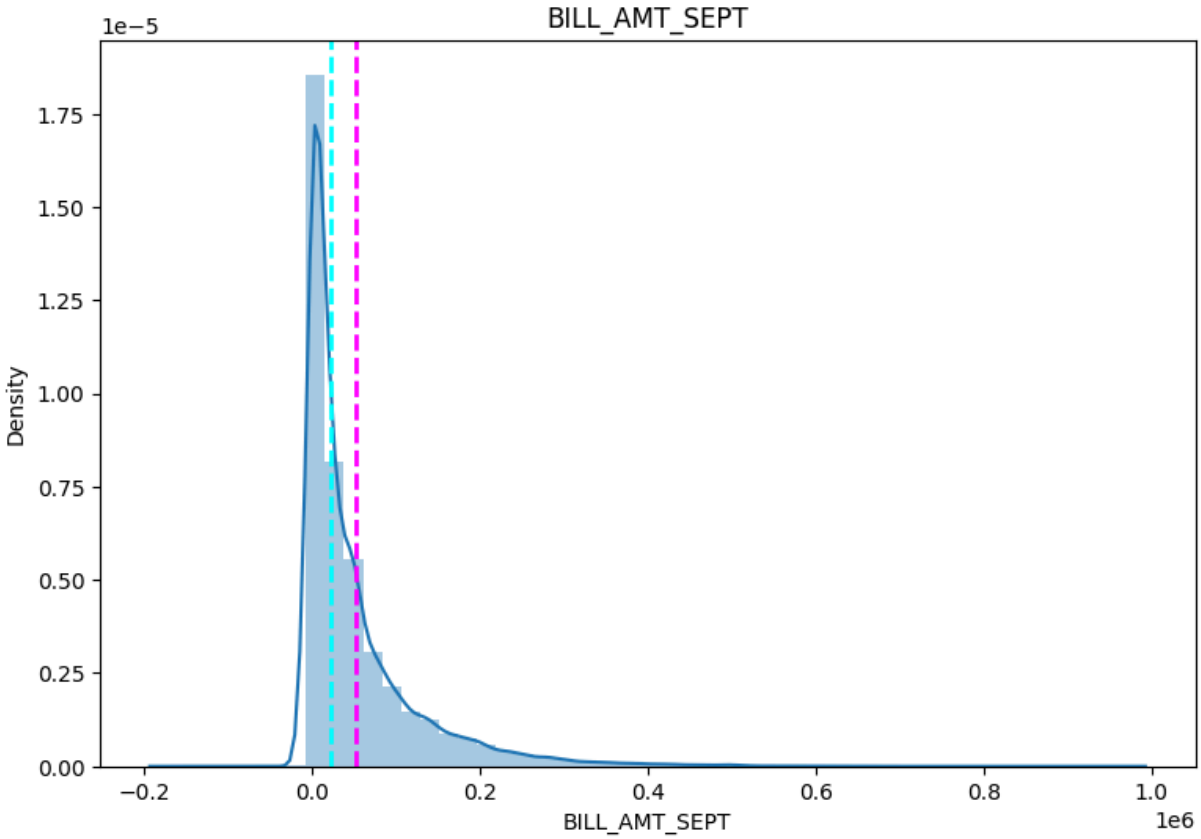

PAY_AUG
Density
PAY_AUG
0.0
0.5
1.0
1.5
2.0
2.5
−2
0
2
4
6
8

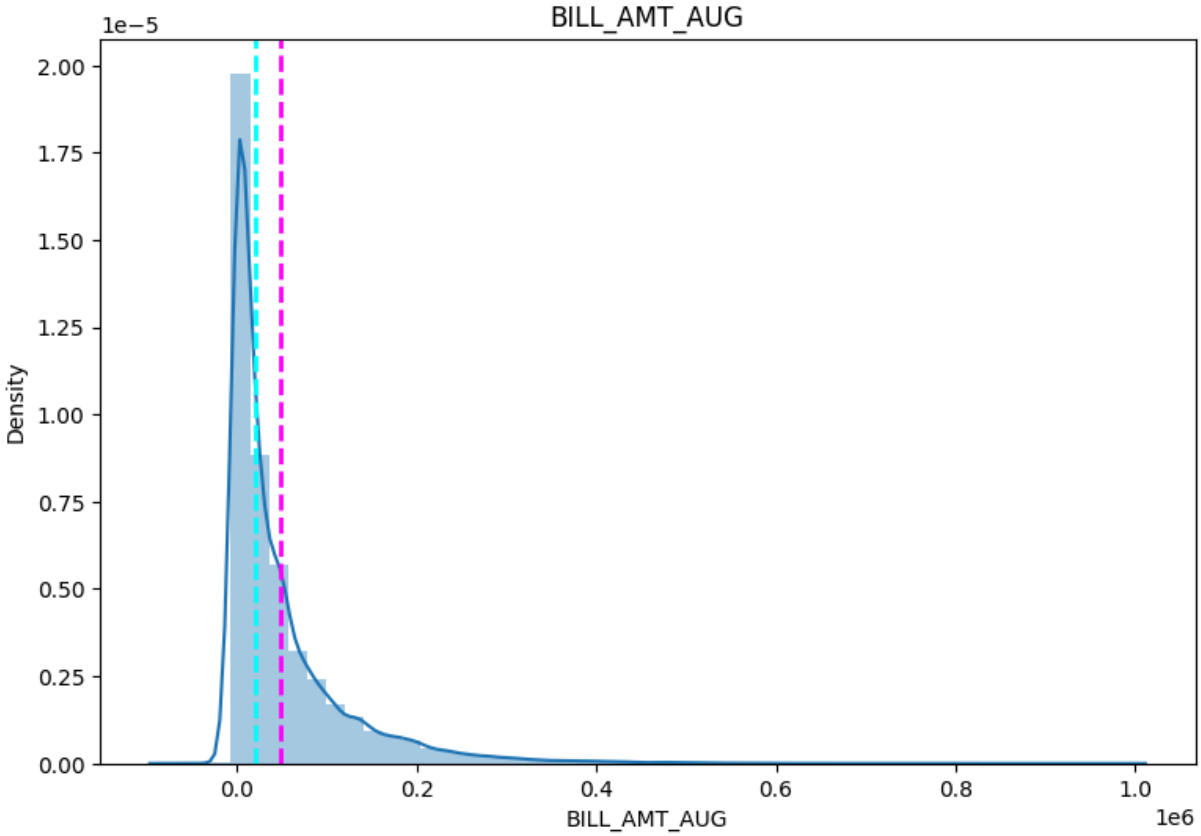

PAY_JUL
Density
PAY_JUL
0.0
0.5
1.0
1.5
2.0
2.5
−2
0
2
4
6
8

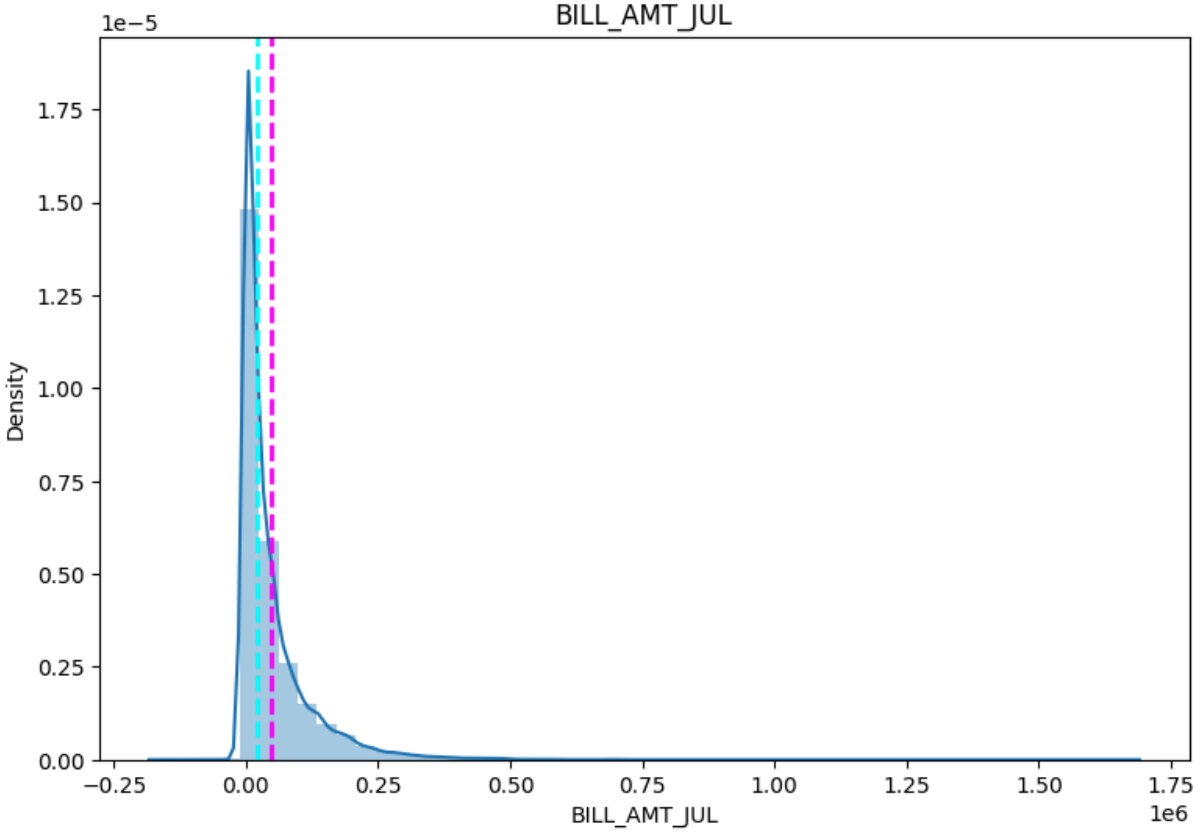

PAY_JUN
Density
PAY_JUN
0.0
0.5
1.0
1.5
2.0
2.5
−2
0
2
4
6
8

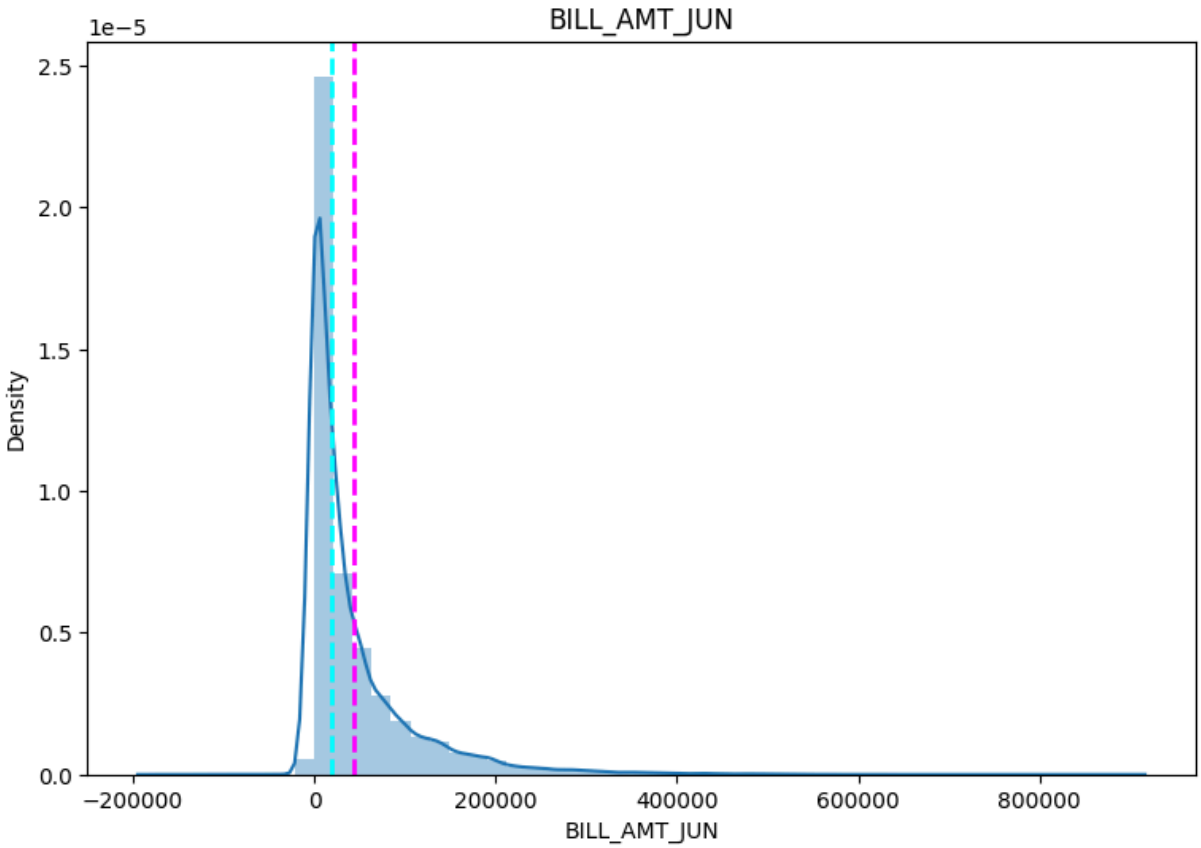

PAY_MAY
Density
PAY_MAY
0.0
0.5
1.0
1.5
2.0
2.5
−2
0
2
4
6
8

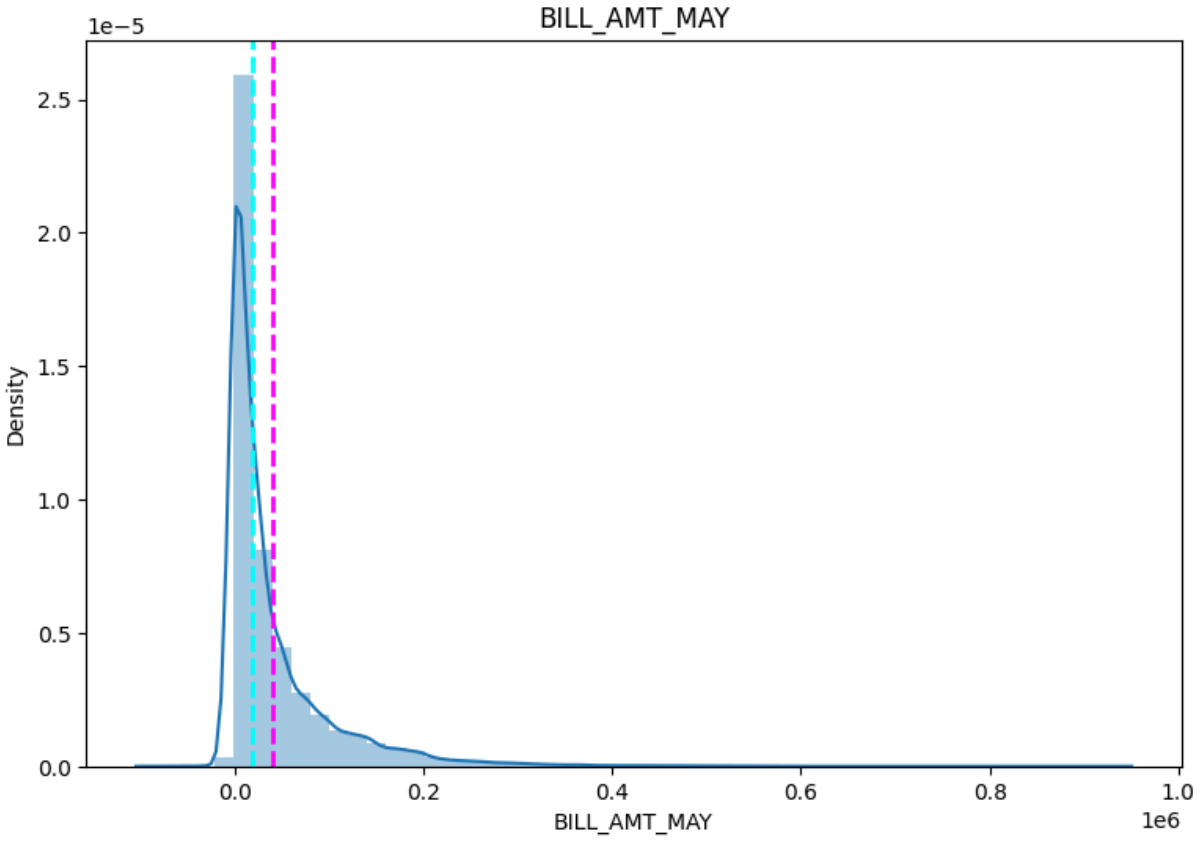

PAY_APR
Density
2.5
2.0
1.5
1.0
0.5
0.0
−2
0
2
4
6
8
PAY_APR

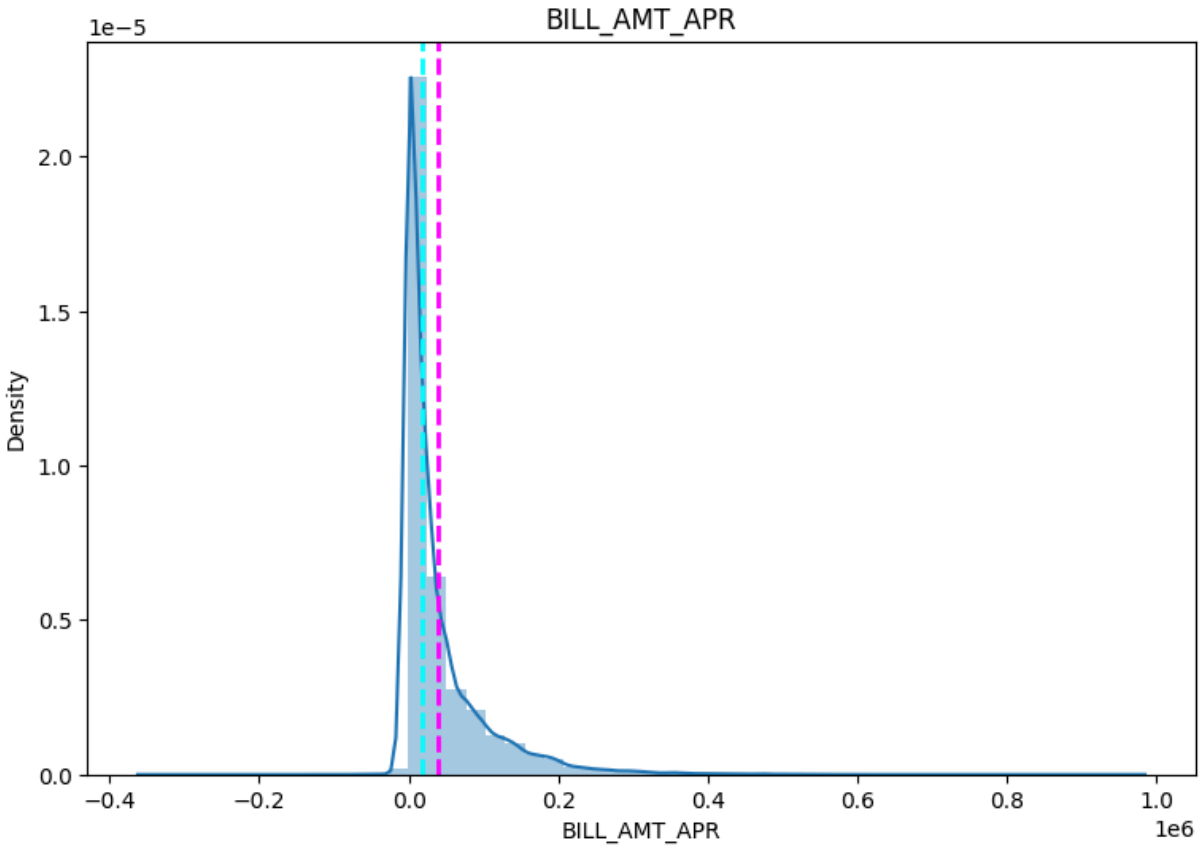

BILL_AMT_SEPT
1e−5
Density
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1e6
BILL_AMT_SEPT

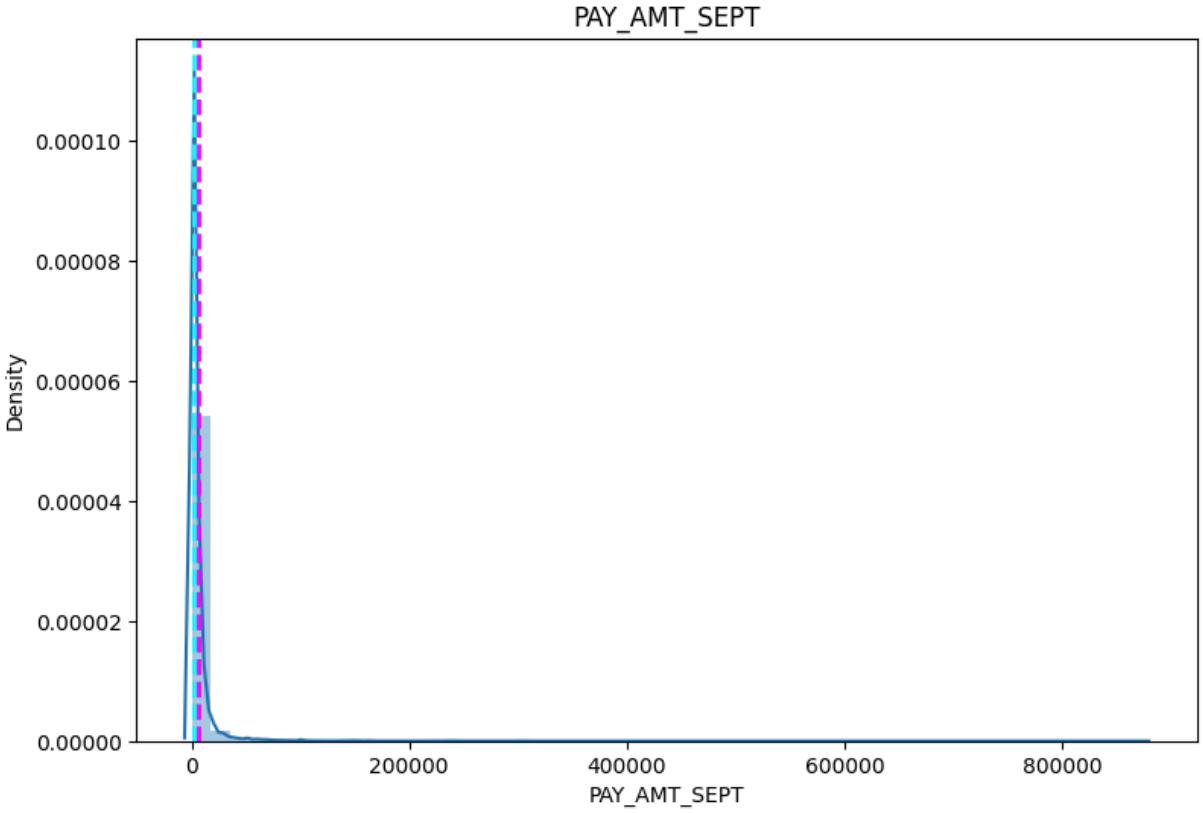

BILL_AMT_AUG
1e−5
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Density
0.0
0.2
0.4
0.6
0.8
1.0
1e6
BILL_AMT_AUG

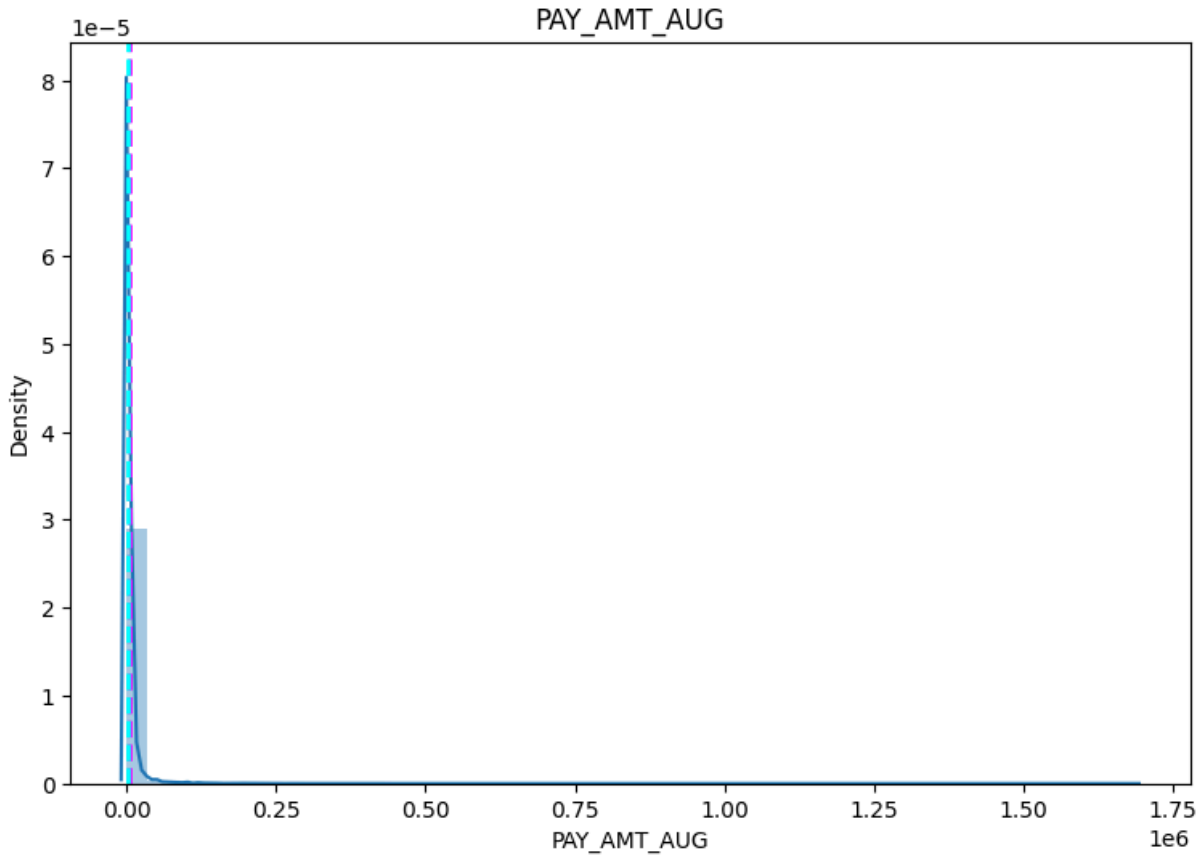

BILL_AMT_JUL
1e−5
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Density
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e6
BILL_AMT_JUL

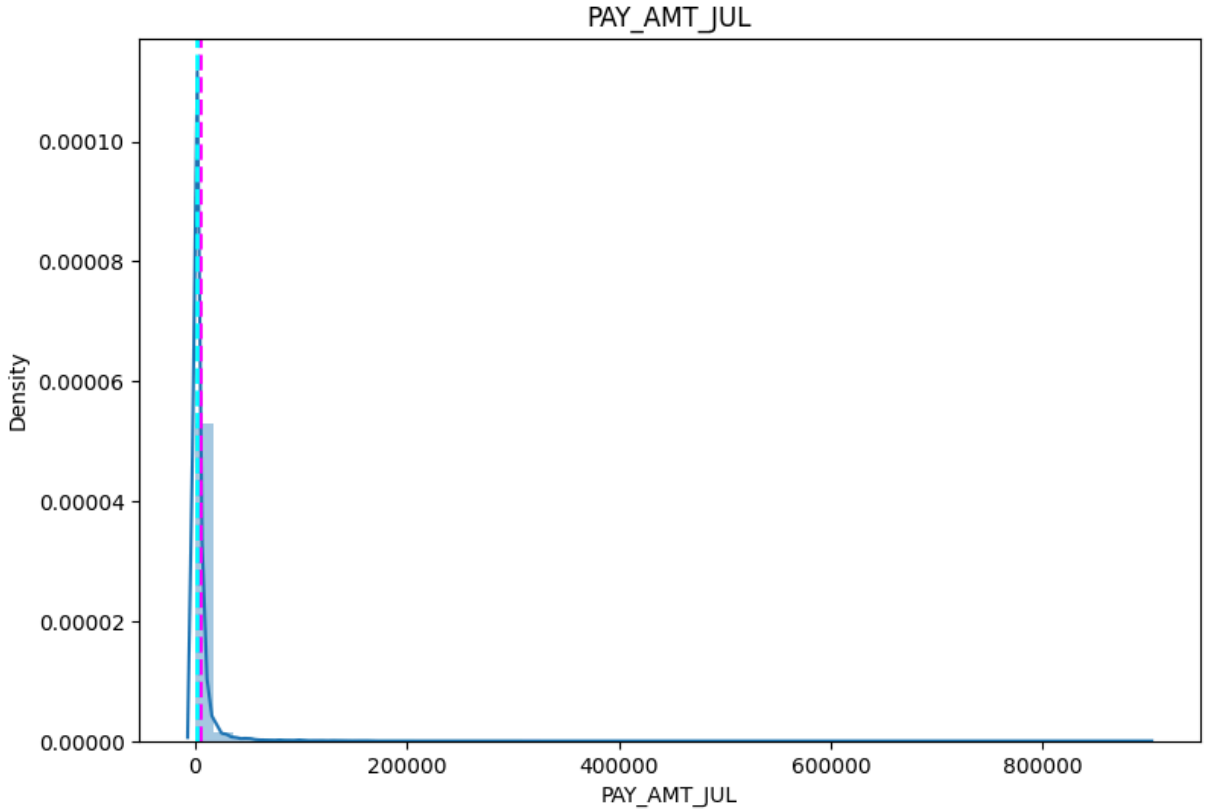

BILL_AMT_JUN
1e−5
2.5
2.0
1.5
1.0
0.5
0.0
Density
−200000
0
200000
400000
600000
800000
BILL_AMT_JUN

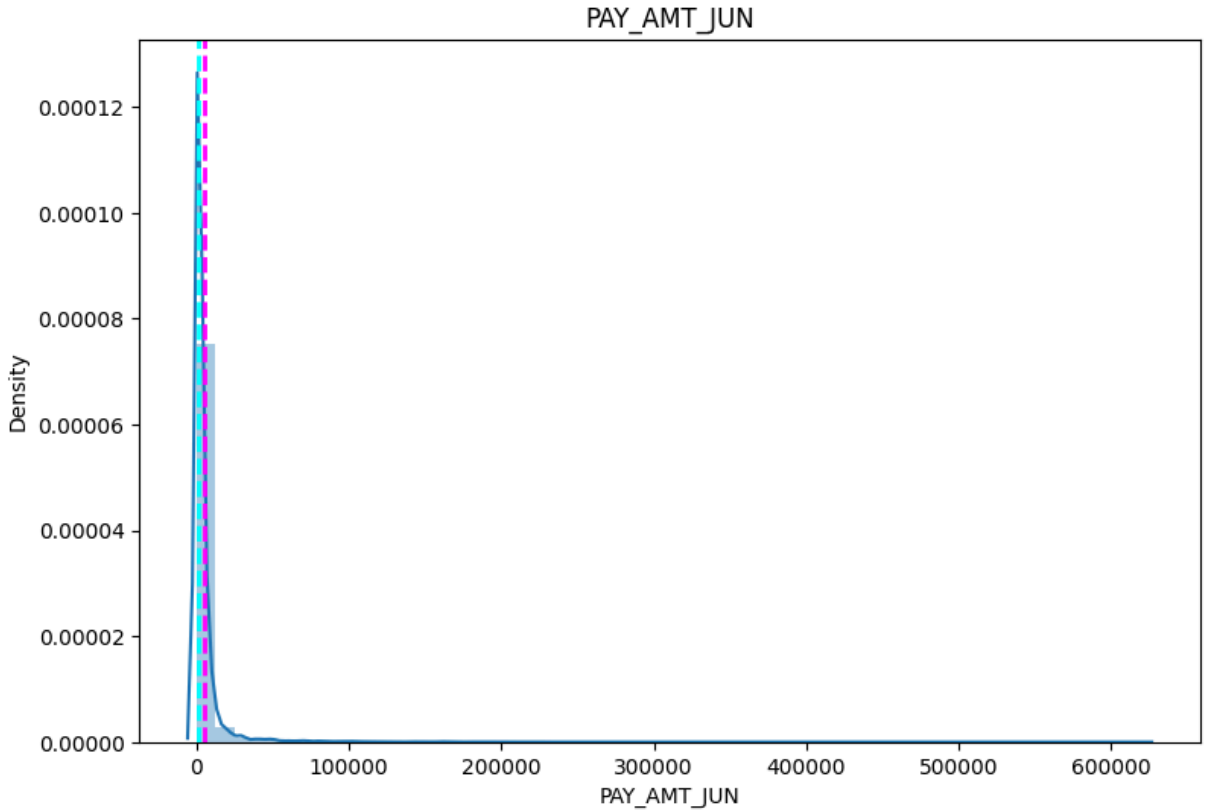

BILL_AMT_MAY
1e−5
2.5
2.0
1.5
1.0
0.5
0.0
Density
0.0
0.2
0.4
0.6
0.8
1.0
1e6
BILL_AMT_MAY

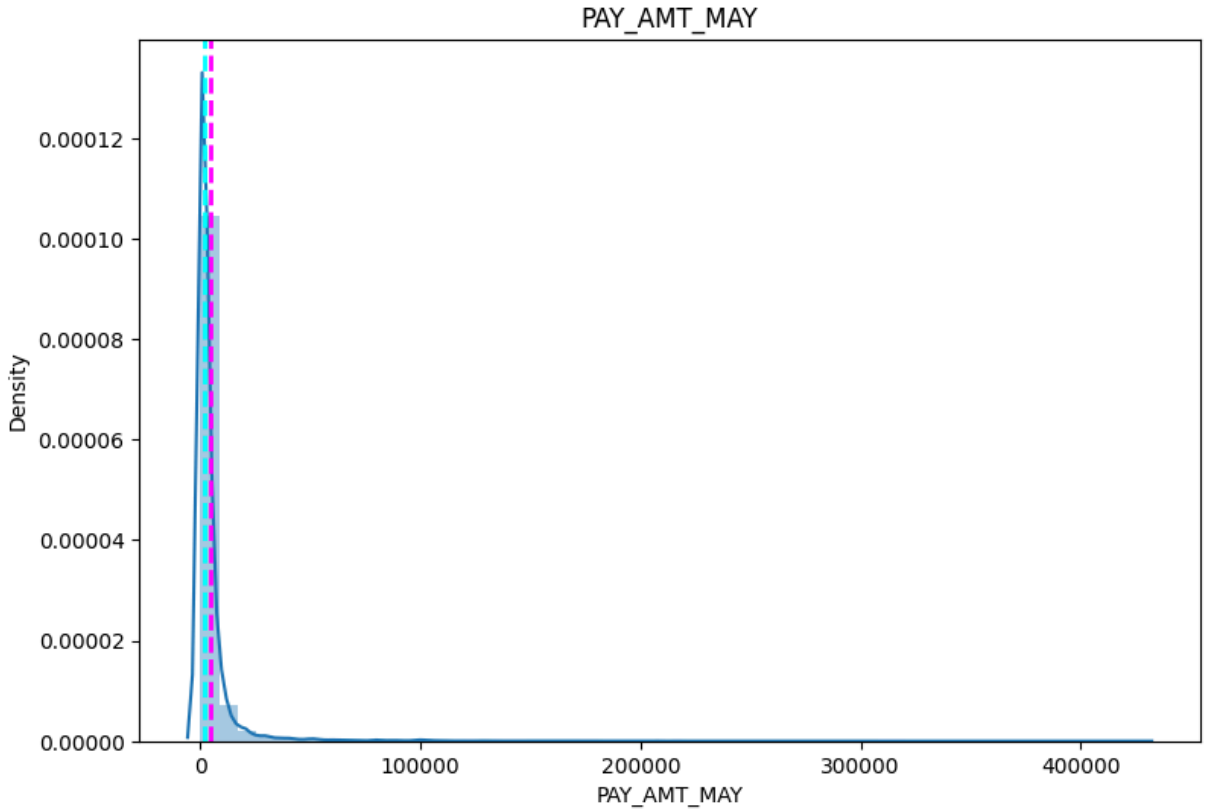

BILL_AMT_APR
1e−5
2.0
1.5
1.0
0.5
0.0
Density
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1e6
BILL_AMT_APR

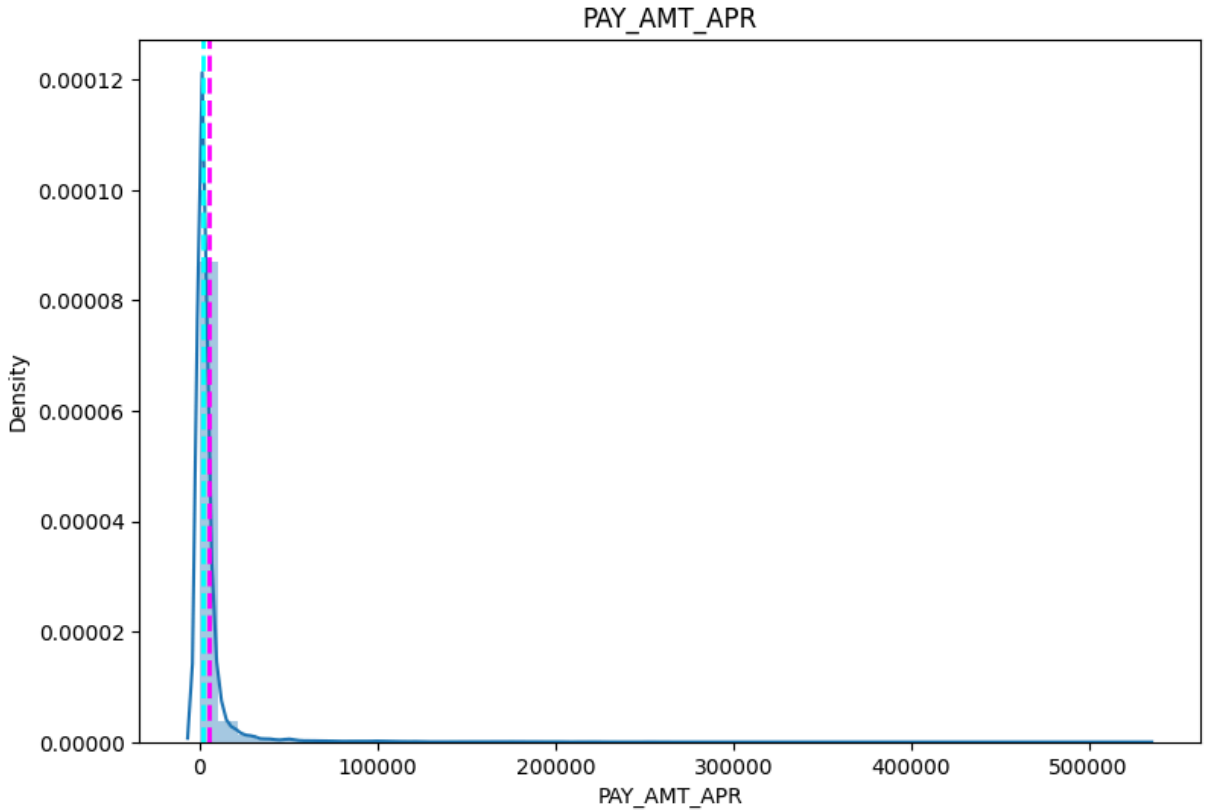

PAY_AMT_SEPT
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
Density
0
200000
400000
600000
800000
PAY_AMT_SEPT

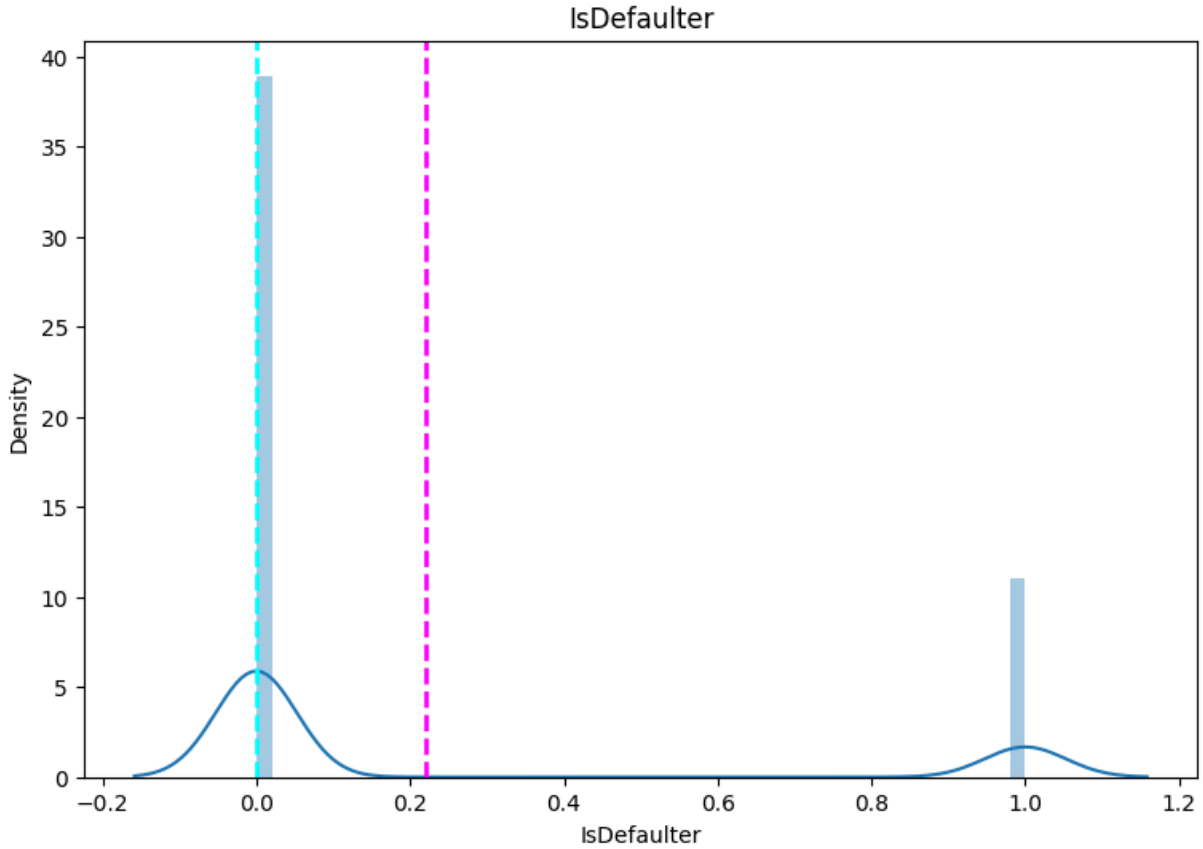

PAY_AMT_AUG
1e−5
Density
PAY_AMT_AUG
1e6


PAY_AMT_JUL
Density
PAY_AMT_JUL


PAY_AMT_JUN
Density
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0
100000
200000
300000
400000
500000
600000
PAY_AMT_JUN


PAY_AMT_MAY
Density
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0
100000
200000
300000
400000
PAY_AMT_MAY





Do you think that your data needs to be transformed? If yes, which transformation have you used. Explain Why?

There is no need to transformed any features.

In [ ]:
```
# Transform Your data
```

6. Dimesionality Reduction

No there is no need for dimensionality reduction in our dataset.

Do you think that dimensionality reduction is needed? Explain Why?

Answer Here.

In [ ]:
```
# DImensionality Reduction (If needed)
```

Which dimensionality reduction technique have you used and why? (If dimensionality reduction done on dataset.)

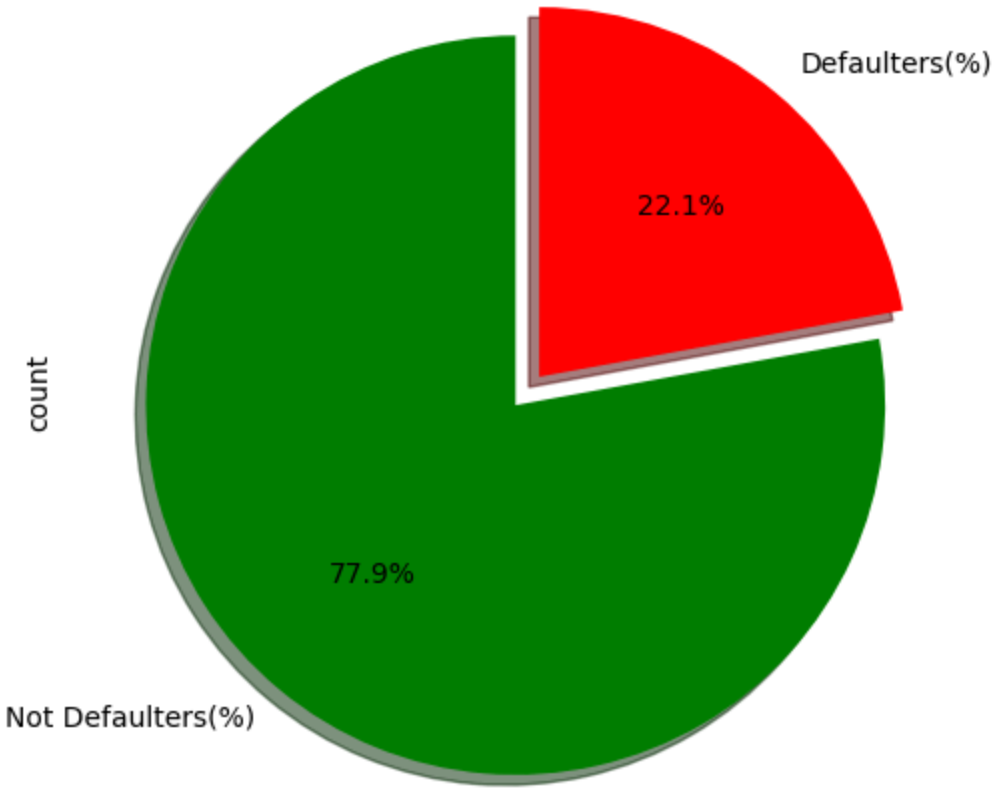
Answer Here.

9. Handling Imbalanced Dataset

In [70]:
```
#split dependent and indipendent variable
y=df['IsDefaulter']
X=df.drop(['IsDefaulter'],axis=1)
```

Do you think the dataset is imbalanced? Explain Why.

In [71]:
```
#Visualisation of class imbalance
y.value_counts().plot(kind='pie',
                              figsize=(15,6),
                               autopct="%1.1f%%",
                               startangle=90,
                               shadow=True,
                               labels=['Not Defaulters(%)','Defaulters(%)'],
                               colors=['green','red'],
                               explode=[.05,.05]
                              )
plt.show()
```



As we can see, 77.9% of the data belongs to the Not defaulters category, while only 22.1% belongs to the Defaulters category, implying that our dataset is unbalanced.

In a dataset with highly imbalanced classes, the classifier will always "predict" the most common class without performing any feature analysis and will have a high accuracy rate, despite the fact that it is plainly not the correct one.

The balanced data would imply that each class would receive 50% of the available points. Little imbalance is not a concern for most machine learning techniques. So, if one class has 60% of the points and the other has 40%, there should be no noticeable performance reduction. Only when the class imbalance is extreme, such as 90% for one class and 10% for the other, would typical optimisation criteria or performance metrics be ineffective and require adjustment.

In [72]:
```
# Handling Imbalanced Dataset (If needed)
# importing SMOTE
from imblearn.over_sampling import SMOTE

print('Dataset Before:\n', y.value_counts())
smt = SMOTE(random_state=40)

# fit predictor and target variable
X_smt, y_smt = smt.fit_resample(X,y)
```

```
print('\nDataset After:\n',y_smt.value_counts())
```

```
Dataset Before:
 IsDefaulter
0    23364
1     6636
Name: count, dtype: int64

Dataset After:
 IsDefaulter
1    23364
0    23364
Name: count, dtype: int64
```

What technique did you use to handle the imbalance dataset and why? (If needed to be balanced)

SMOTE (Synthetic Minority Over-sampling technique) was employed to balance the dataset.

SMOTE is a machine learning technique for coping with problems that arise while working with an imbalanced data collection. In practise, imbalanced data sets are common, and most ML algorithms are vulnerable to them, so we must improve their performance by employing approaches such as SMOTE.

The SMOTE algorithm operates in four simple steps:

1)Select a minority group as the input vector.

2)Determine its k nearest neighbours (k_neighbors is an input to the SMOTE() method).

3)Select one of these neighbours and draw a synthetic point somewhere on the line connecting the point under consideration and its selected neighbour.

4)Repeat the process until the data is balanced.

7. Data Splitting

In [73]:
```
# Split your data to train and test. Choose Splitting ratio wisely.
X_train,X_test,y_train,y_test=train_test_split(X_smt,y_smt,test_size=0.2,rar
```

In [74]:
```
#Checking the shape of
print(X_train.shape,y_train.shape)
# Check the shape of test dataset
print(X_test.shape, y_test.shape)
```

```
(37382, 29) (37382,)
(9346, 29) (9346,)
```

What data splitting ratio have you used and why?

I used an 80:20 split for the train-test split. We can see that we have 37382 data for training and 9346 data for testing, which is a reasonable split to begin with because we have kept a substantial amount of data for training our model.

8. Data Scaling

In [75]:
```
# Scaling your data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Which method have you used to scale you data and why?

I chose Standard Scaler to scale my data since it adjusts the data so that the mean is 0 and the standard deviation is 1. In a nutshell, it standardises data. Standardisation is beneficial for data with negative values. The data is arranged in a conventional normal distribution. It's better for classification than regression.

7. ML Model Implementation

ML Model - 1 Logistic Regression

In [76]:
```
# ML Model - 1 Implementation
model_lr=LogisticRegression(fit_intercept=True,max_iter=1000)

# Fit the Algorithm
model_lr.fit(X_train_scaled,y_train)
```

Out[76]:
```
LogisticRegression(max_iter=1000)
```

In [77]:
```
#Cheking model coefficient
print("The Coefficients obtain from logisticregression model",model_lr.coef_
# Checking the intercept value
print("The Coefficients obtain from logisticregression model",model_lr.inter
```

```
The Coefficients obtain from logisticregression model [[-0.15041894 -0.160332
54 -0.03429755  0.62152259  0.11656494  0.07584934
   0.02572093  0.0789968  -0.020382   -0.5075592   0.28156893  0.15795979
  -0.07765115 -0.04808135  0.12654957 -0.3101838  -0.2814535  -0.02662904
  -0.07268588 -0.10158572 -0.03618437 -0.25768263 -2.00451376 -0.50200246
  -0.45827685 -0.11183331 -3.85328465 -2.00451376 -0.50200246]]
The Coefficients obtain from logisticregression model [0.97043618]
```

In [78]:
```
#predicting the probabilities of test data
lr_train_proba=model_lr.predict_proba(X_train_scaled)
lr_test_proba=model_lr.predict_proba(X_test_scaled)
```

In [79]:
```
#predicting the values of y from x via model
y_pred_test = model_lr.predict(X_test_scaled)
y_pred_train = model_lr.predict(X_train_scaled)
```

1. Explain the ML Model used and it's performance using Evaluation metric Score Chart.

Logistic regression is a classification algorithm that uses supervised learning to predict the likelihood of a target variable. Because the nature of the target or dependent variable is dichotomous, there are only two viable classes.

Simply put, the dependent variable is binary in nature, with data represented as either 1 (for success/yes) or 0 (for failure/no).

A logistic regression model predicts P(Y=1) as a function of X mathematically. It is one of the most basic ML techniques that may be used to solve a variety of classification issues such as spam identification, diabetes prediction, cancer diagnosis, and so on.

Evaluation Matrics

Accuracy: Accuracy = number of correct forecasts. Total number of forecasts.

Precision: Precision is defined as the number of genuine positives divided by the number of anticipated positives given a label.When the cost of false positives (FP) is high, precision is a good statistic to utilise.

Precision = TP / (TP + FP).

Recall: Recall is defined as the number of true positives divided by the total number of actual positives for a label.When the cost of false negative (FN) is high, recall is an excellent statistic to employ.

recall=TP / (TP + FN)

F1-score: The harmonic mean of precision and recall is defined as the F1-score.

AUC-ROC: The Receiver Operator Characteristic (ROC) curve is a binary classification issue evaluation metric. It is a probability curve that plots the TPR against FPR at various threshold levels and essentially separates the 'signal' from the 'noise'. The Area Under the Curve (AUC) is a measure of a classifier's ability to distinguish between classes and is used to summarise the ROC curve.

I'll be using Recall primarily for model evaluation because False Negative indicates that a person will not default when they actually do. And recognising defaulter clients as non-defaulters will result in a large loss for the bank, thus we must reduce False Negative, and as False Negative decreases, Recall will grow.

KS-plot: Kolmogorov-Smirnov chart measures performance of classification models. K-S is a measure of the degree of separation between the positive and negative distributions.

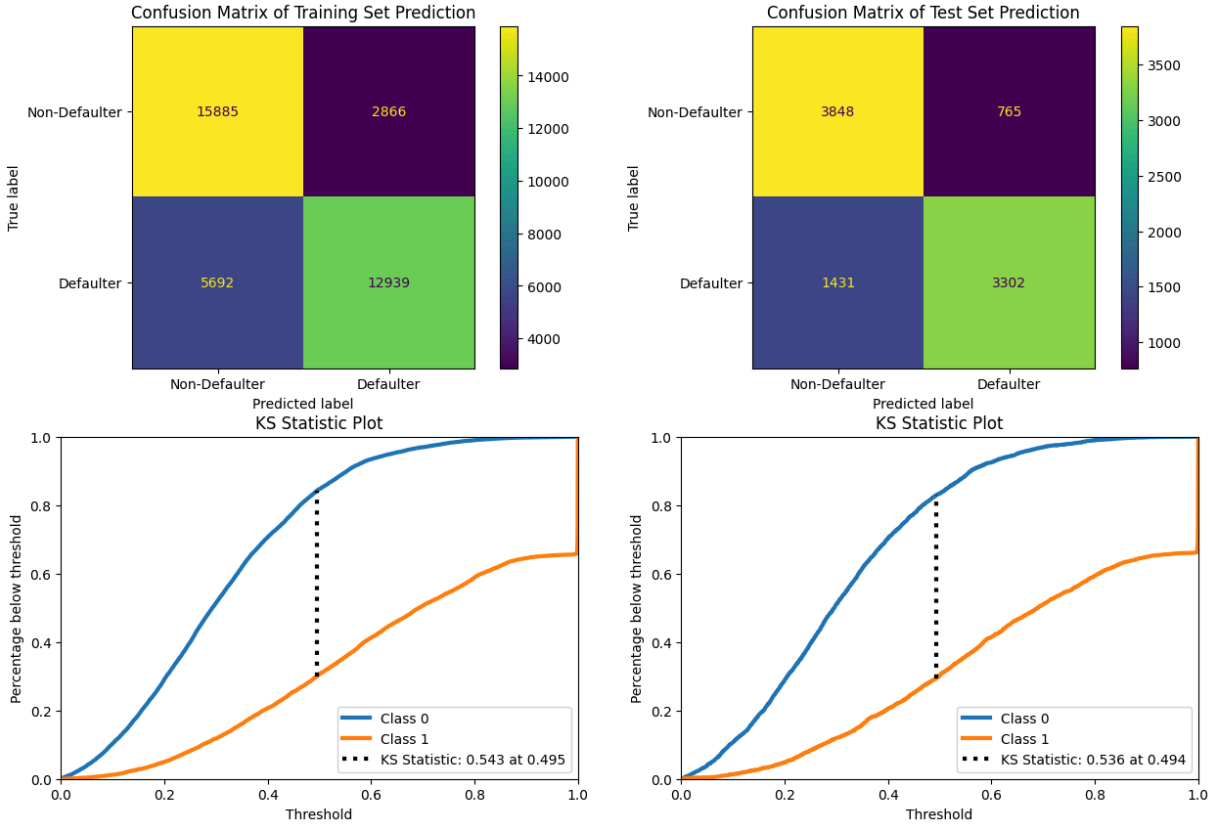
In [80]:
```
#create a function to visualize the model performence
def model_vis(y_train,y_test,y_pred_train,y_pred_test,y_train_proba,y_test_p
  '''
  This Function helps to visualize confusion matrix of train set and test se
  '''
  fig,ax=plt.subplots(2,2,figsize=(15,10))
  cm = metrics.confusion_matrix(y_train, y_pred_train)
  cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm, display
  cm_display.plot(ax=ax[0,0])
  ax[0,0].set_title('Confusion Matrix of Training Set Prediction')
  #Confusion matrics for Test
  cm = confusion_matrix(y_test, y_pred_test)
  cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm, display
  cm_display.plot(ax=ax[0,1])
  ax[0,1].set_title('Confusion Matrix of Test Set Prediction')

  #ks chart plot
  skplt.metrics.plot_ks_statistic(y_train, y_train_proba,ax=ax[1,0])
  #remove the last subplots
  skplt.metrics.plot_ks_statistic(y_test, y_test_proba,ax=ax[1,1])
  plt.show()
```

In [81]:
```
!pip install --upgrade scikit-plot
print(dir())  # Lists all available variables
```

```
Requirement already satisfied: scikit-plot in /Library/Frameworks/Python.fram
ework/Versions/3.11/lib/python3.11/site-packages (0.3.7)
Requirement already satisfied: matplotlib>=1.4.0 in /Library/Frameworks/Pytho
n.framework/Versions/3.11/lib/python3.11/site-packages (from scikit-plot) (3.
10.0)
Requirement already satisfied: scikit-learn>=0.18 in /Library/Frameworks/Pyth
on.framework/Versions/3.11/lib/python3.11/site-packages (from scikit-plot)
(1.6.1)
Requirement already satisfied: scipy>=0.9 in /Library/Frameworks/Python.frame
work/Versions/3.11/lib/python3.11/site-packages (from scikit-plot) (1.9.3)
Requirement already satisfied: joblib>=0.10 in /Library/Frameworks/Python.fra
mework/Versions/3.11/lib/python3.11/site-packages (from scikit-plot) (1.4.2)
Requirement already satisfied: contourpy>=1.0.1 in /Library/Frameworks/Pytho
n.framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib>=1.4.
0->scikit-plot) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /Library/Frameworks/Python.fra
mework/Versions/3.11/lib/python3.11/site-packages (from matplotlib>=1.4.0->sc
ikit-plot) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /Library/Frameworks/Pytho
n.framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib>=1.4.
0->scikit-plot) (4.40.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /Library/Frameworks/Pytho
n.framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib>=1.4.
0->scikit-plot) (1.4.4)
Requirement already satisfied: numpy>=1.23 in /Library/Frameworks/Python.fram
ework/Versions/3.11/lib/python3.11/site-packages (from matplotlib>=1.4.0->sci
kit-plot) (1.25.2)
Requirement already satisfied: packaging>=20.0 in /Library/Frameworks/Python.
framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib>=1.4.0-
>scikit-plot) (23.1)
Requirement already satisfied: pillow>=8 in /Library/Frameworks/Python.framew
ork/Versions/3.11/lib/python3.11/site-packages (from matplotlib>=1.4.0->sciki
t-plot) (9.5.0)
Requirement already satisfied: pyparsing>=2.3.1 in /Library/Frameworks/Pytho
n.framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib>=1.4.
0->scikit-plot) (3.1.0)
Requirement already satisfied: python-dateutil>=2.7 in /Library/Frameworks/Py
thon.framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib>=
1.4.0->scikit-plot) (2.8.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /Library/Frameworks/Py
thon.framework/Versions/3.11/lib/python3.11/site-packages (from scikit-learn>
=0.18->scikit-plot) (3.5.0)
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.framewo
rk/Versions/3.11/lib/python3.11/site-packages (from python-dateutil>=2.7->mat
plotlib>=1.4.0->scikit-plot) (1.16.0)

[notice] A new release of pip is available: 25.0 -> 25.0.1
[notice] To update, run: pip install --upgrade pip
['AdaBoostClassifier', 'Age_female', 'Age_male', 'ConfusionMatrixDisplay', 'D
ecisionTreeClassifier', 'GridSearchCV', 'In', 'KNeighborsClassifier', 'Limit_
1', 'Limit_2', 'LogisticRegression', 'N', 'Out', 'P_value', 'RandomForestClas
sifier', 'RandomizedSearchCV', 'RocCurveDisplay', 'SMOTE', 'StandardScaler',
'X', 'XGBClassifier', 'X_smt', 'X_test', 'X_test_scaled', 'X_train', 'X_train
_scaled', 'Z_stat', '_', '_13', '_14', '_18', '_19', '_20', '_21', '_27', '_3
3', '_48', '_51', '_53', '_67', '_76', '__', '___', '__builtin__', '__builtin
s__', '__doc__', '__loader__', '__name__', '__package__', '__spec__', '__vsc_
```

```
ipynb_file__', '_dh', '_exit_code', '_i', '_i1', '_i10', '_i11', '_i12', '_i1
3', '_i14', '_i15', '_i16', '_i17', '_i18', '_i19', '_i2', '_i20', '_i21', '_
i22', '_i23', '_i24', '_i25', '_i26', '_i27', '_i28', '_i29', '_i3', '_i30',
'_i31', '_i32', '_i33', '_i34', '_i35', '_i36', '_i37', '_i38', '_i39', '_i
4', '_i40', '_i41', '_i42', '_i43', '_i44', '_i45', '_i46', '_i47', '_i48',
'_i49', '_i5', '_i50', '_i51', '_i52', '_i53', '_i54', '_i55', '_i56', '_i5
7', '_i58', '_i59', '_i6', '_i60', '_i61', '_i62', '_i63', '_i64', '_i65', '_
i66', '_i67', '_i68', '_i69', '_i7', '_i70', '_i71', '_i72', '_i73', '_i74',
'_i75', '_i76', '_i77', '_i78', '_i79', '_i8', '_i80', '_i81', '_i9', '_ih',
'_ii', '_iii', '_oh', 'accuracy_score', 'age_dist', 'ax', 'calc_vif', 'chi2_c
ontingency', 'chisquare', 'classification_report', 'col', 'confusion_matrix',
'cont_table', 'credit_df', 'df', 'df_credit', 'dof', 'duplicate_value', 'educ
ation_columns', 'exit', 'expected', 'f1_score', 'feature', 'fig', 'filepath',
'get_ipython', 'i', 'ks_2samp', 'lr_test_proba', 'lr_train_proba', 'mean_medi
an_difference', 'metrics', 'model_lr', 'model_vis', 'monthlybill_col', 'msn
o', 'norm', 'np', 'open', 'pd', 'plt', 'pop_mean', 'precision_score', 'previo
us_col', 'quit', 'recall_score', 'repay_col', 'roc_auc_score', 'sample_1', 's
ample_2', 'sample_mean', 'sample_size', 'scaler', 'skplt', 'sm', 'smt', 'sn
s', 'stat', 'stats', 'std_pop', 't_statistic', 'train_test_split', 'variance_
inflation_factor', 'warnings', 'y', 'y_pred_test', 'y_pred_train', 'y_smt',
'y_test', 'y_train', 'z_value']
```

In [82]:
```
import scikitplot as skplt

# Now call the function
model_vis(y_train, y_test, y_pred_train, y_pred_test, lr_train_proba, lr_tes
```



- There is similar outcome of training set and test set.

- According to the test set, around 90% of all defaulters are correctly anticipated, and approximately 70% of all defaulters are successfully predicted.
- 0.655 is going to be our threshold. We begin the solution of the business case by considering as potential defaulters all the observations with a predicted probability of defaulting greater than 0.655

In [ ]:
```
#y_pred_new_train = (model_lr.predict_proba(X_train_scaled)[:,1]>=0.65).asty
```

In [ ]:
```
#y_pred_new_test=(model_lr.predict_proba(X_test_scaled)[:,1]>=0.35).astype(i
```

In [83]:
```
#create an empty dataframe with columns name of metric chart
metrics_df=pd.DataFrame(columns = ['Model', 'Accuracy', 'Precision', 'Recall
def metric_score(model_name,y_test,y_pred_test,y_proba):
  '''
  This Function will detremine model evalation metrics with a dataframe
  '''
  #accuracy
  acc = accuracy_score(y_test, y_pred_test)
  #precision
  prec = precision_score(y_test, y_pred_test)
  #recall
  rec = recall_score(y_test,y_pred_test)
  #f1-score
  f1 = f1_score(y_test, y_pred_test)
  #create a dataframe for ks stat calculation
  df = pd.DataFrame()
  df['real'] = y_test
  df['proba'] = y_proba[:, 1]

   # Recover each class
  class0 = df[df['real'] == 0]
  class1 = df[df['real'] == 1]

  ks = ks_2samp(class0['proba'], class1['proba'])
  stat=round(ks[0],3)
  #dataframe of all metric score

  eval_metrics_list =[model_name, acc,prec,rec, f1,stat]
  metrics_df.loc[len(metrics_df)]=eval_metrics_list
  return metrics_df
```

In [84]:
```
#view all the metric score in a dataframe
metric_score('Logistic_Regression',y_test,y_pred_test,lr_test_proba)
```

Out[84]:

	Model	Accuracy	Precision	Recall	F1 Score	KS_Stat
0	Logistic_Regression	0.765033	0.811901	0.697655	0.750455	0.536

2. Cross- Validation & Hyperparameter Tuning

In [85]:
```
# ML Model - 1 Implementation with hyperparameter optimization techniques (i
lr=LogisticRegression()
```

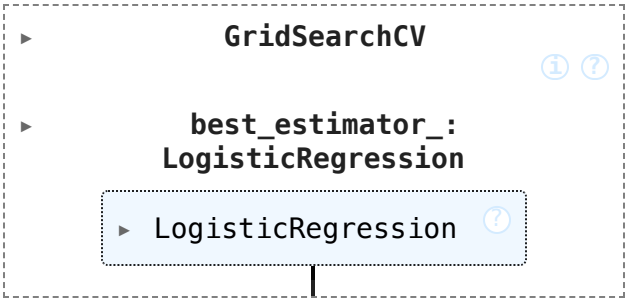
```
param_grid = {'penalty':['l1','l2'], 'C' : [0.0001,0.001, 0.01, 0.1, 0.2, 0.
lr_cv=GridSearchCV(lr, param_grid, scoring = 'recall',n_jobs = -1, verbose =
# Fit the Algorithm
lr_cv.fit(X_train_scaled,y_train)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
[CV 2/5] END .................C=0.0001, penalty=l1;, score=nan total time=
0.0s
[CV 5/5] END .................C=0.0001, penalty=l1;, score=nan total time=
0.0s
[CV 4/5] END .................C=0.0001, penalty=l1;, score=nan total time=
0.0s
[CV 1/5] END .................C=0.0001, penalty=l1;, score=nan total time=
0.0s
[CV 3/5] END .................C=0.0001, penalty=l1;, score=nan total time=
0.0s
[CV 2/5] END ...............C=0.0001, penalty=l2;, score=0.699 total time=
0.1s
[CV 1/5] END ...............C=0.0001, penalty=l2;, score=0.719 total time=
0.1s
[CV 1/5] END ..................C=0.001, penalty=l1;, score=nan total time=
0.0s
[CV 4/5] END ...............C=0.0001, penalty=l2;, score=0.717 total time=
0.0s
[CV 3/5] END ...............C=0.0001, penalty=l2;, score=0.703 total time=
0.1s
[CV 2/5] END ..................C=0.001, penalty=l1;, score=nan total time=
0.0s
[CV 5/5] END ...............C=0.0001, penalty=l2;, score=0.707 total time=
0.0s
[CV 3/5] END ..................C=0.001, penalty=l1;, score=nan total time=
0.0s
[CV 4/5] END ..................C=0.001, penalty=l1;, score=nan total time=
0.0s
[CV 5/5] END ..................C=0.001, penalty=l1;, score=nan total time=
0.0s
[CV 1/5] END ...................C=0.01, penalty=l1;, score=nan total time=
0.0s
[CV 2/5] END ...................C=0.01, penalty=l1;, score=nan total time=
0.0s
[CV 3/5] END ...................C=0.01, penalty=l1;, score=nan total time=
0.0s
[CV 1/5] END ................C=0.001, penalty=l2;, score=0.716 total time=
0.0s
[CV 2/5] END ................C=0.001, penalty=l2;, score=0.704 total time=
0.0s
[CV 4/5] END ...................C=0.01, penalty=l1;, score=nan total time=
0.0s
[CV 5/5] END ................C=0.001, penalty=l2;, score=0.705 total time=
0.0s
[CV 3/5] END ................C=0.001, penalty=l2;, score=0.694 total time=
0.0s
[CV 5/5] END ...................C=0.01, penalty=l1;, score=nan total time=
0.0s
[CV 4/5] END ................C=0.001, penalty=l2;, score=0.717 total time=
0.0s
[CV 1/5] END ....................C=0.1, penalty=l1;, score=nan total time=
0.0s
[CV 2/5] END ....................C=0.1, penalty=l1;, score=nan total time=
0.0s
[CV 3/5] END ....................C=0.1, penalty=l1;, score=nan total time=
```

```
0.0s
[CV 4/5] END ..................C=0.1, penalty=l1;, score=nan total time=
0.0s
[CV 5/5] END ..................C=0.1, penalty=l1;, score=nan total time=
0.0s
[CV 2/5] END ................C=0.01, penalty=l2;, score=0.694 total time=
0.1s
[CV 1/5] END ................C=0.01, penalty=l2;, score=0.704 total time=
0.1s
[CV 3/5] END ................C=0.01, penalty=l2;, score=0.685 total time=
0.1s
[CV 5/5] END ................C=0.01, penalty=l2;, score=0.699 total time=
0.1s
[CV 4/5] END ................C=0.01, penalty=l2;, score=0.708 total time=
0.1s
[CV 2/5] END ..................C=0.2, penalty=l1;, score=nan total time=
0.0s
[CV 1/5] END ..................C=0.2, penalty=l1;, score=nan total time=
0.0s
[CV 3/5] END ..................C=0.2, penalty=l1;, score=nan total time=
0.0s
[CV 4/5] END ..................C=0.2, penalty=l1;, score=nan total time=
0.0s
[CV 5/5] END ..................C=0.2, penalty=l1;, score=nan total time=
0.0s
[CV 1/5] END ................C=0.1, penalty=l2;, score=0.702 total time=
0.1s
[CV 3/5] END ................C=0.1, penalty=l2;, score=0.682 total time=
0.1s
[CV 2/5] END ................C=0.1, penalty=l2;, score=0.693 total time=
0.1s
[CV 4/5] END ................C=0.1, penalty=l2;, score=0.706 total time=
0.1s
[CV 5/5] END ................C=0.1, penalty=l2;, score=0.697 total time=
0.1s
[CV 1/5] END ..................C=0.3, penalty=l1;, score=nan total time=
0.0s
[CV 3/5] END ..................C=0.3, penalty=l1;, score=nan total time=
0.0s
[CV 2/5] END ..................C=0.3, penalty=l1;, score=nan total time=
0.0s
[CV 4/5] END ..................C=0.3, penalty=l1;, score=nan total time=
0.0s
[CV 5/5] END ..................C=0.3, penalty=l1;, score=nan total time=
0.0s
[CV 1/5] END ..................C=0.4, penalty=l1;, score=nan total time=
0.0s
[CV 2/5] END ..................C=0.4, penalty=l1;, score=nan total time=
0.0s
[CV 1/5] END ................C=0.2, penalty=l2;, score=0.702 total time=
0.1s
[CV 2/5] END ................C=0.2, penalty=l2;, score=0.693 total time=
0.1s
[CV 3/5] END ..................C=0.4, penalty=l1;, score=nan total time=
0.0s
[CV 4/5] END ..................C=0.4, penalty=l1;, score=nan total time=
```

```
0.0s
[CV 3/5] END .................C=0.2, penalty=l2;, score=0.682 total time=
0.1s
[CV 5/5] END ...................C=0.4, penalty=l1;, score=nan total time=
0.0s
[CV 1/5] END .................C=0.3, penalty=l2;, score=0.701 total time=
0.1s
[CV 2/5] END .................C=0.3, penalty=l2;, score=0.691 total time=
0.1s
[CV 4/5] END .................C=0.2, penalty=l2;, score=0.706 total time=
0.1s
[CV 1/5] END ...................C=0.5, penalty=l1;, score=nan total time=
0.0s
[CV 2/5] END ...................C=0.5, penalty=l1;, score=nan total time=
0.0s
[CV 4/5] END .................C=0.3, penalty=l2;, score=0.706 total time=
0.1s
[CV 3/5] END ...................C=0.5, penalty=l1;, score=nan total time=
0.0s
[CV 4/5] END ...................C=0.5, penalty=l1;, score=nan total time=
0.0s
[CV 5/5] END ...................C=0.5, penalty=l1;, score=nan total time=
0.0s
[CV 4/5] END .................C=0.4, penalty=l2;, score=0.706 total time=
0.1s
[CV 1/5] END .................C=0.4, penalty=l2;, score=0.701 total time=
0.1s
[CV 2/5] END .................C=0.4, penalty=l2;, score=0.692 total time=
0.1s
[CV 3/5] END .................C=0.3, penalty=l2;, score=0.682 total time=
0.1s
[CV 1/5] END .................C=0.5, penalty=l2;, score=0.700 total time=
0.1s
[CV 5/5] END .................C=0.2, penalty=l2;, score=0.697 total time=
0.1s
[CV 2/5] END .................C=0.5, penalty=l2;, score=0.692 total time=
0.1s
[CV 5/5] END .................C=0.3, penalty=l2;, score=0.696 total time=
0.1s
[CV 4/5] END .................C=0.5, penalty=l2;, score=0.706 total time=
0.1s
[CV 3/5] END .................C=0.4, penalty=l2;, score=0.682 total time=
0.1s
[CV 5/5] END .................C=0.4, penalty=l2;, score=0.697 total time=
0.1s
[CV 3/5] END .................C=0.5, penalty=l2;, score=0.682 total time=
0.1s
[CV 5/5] END .................C=0.5, penalty=l2;, score=0.696 total time=
0.1s
```

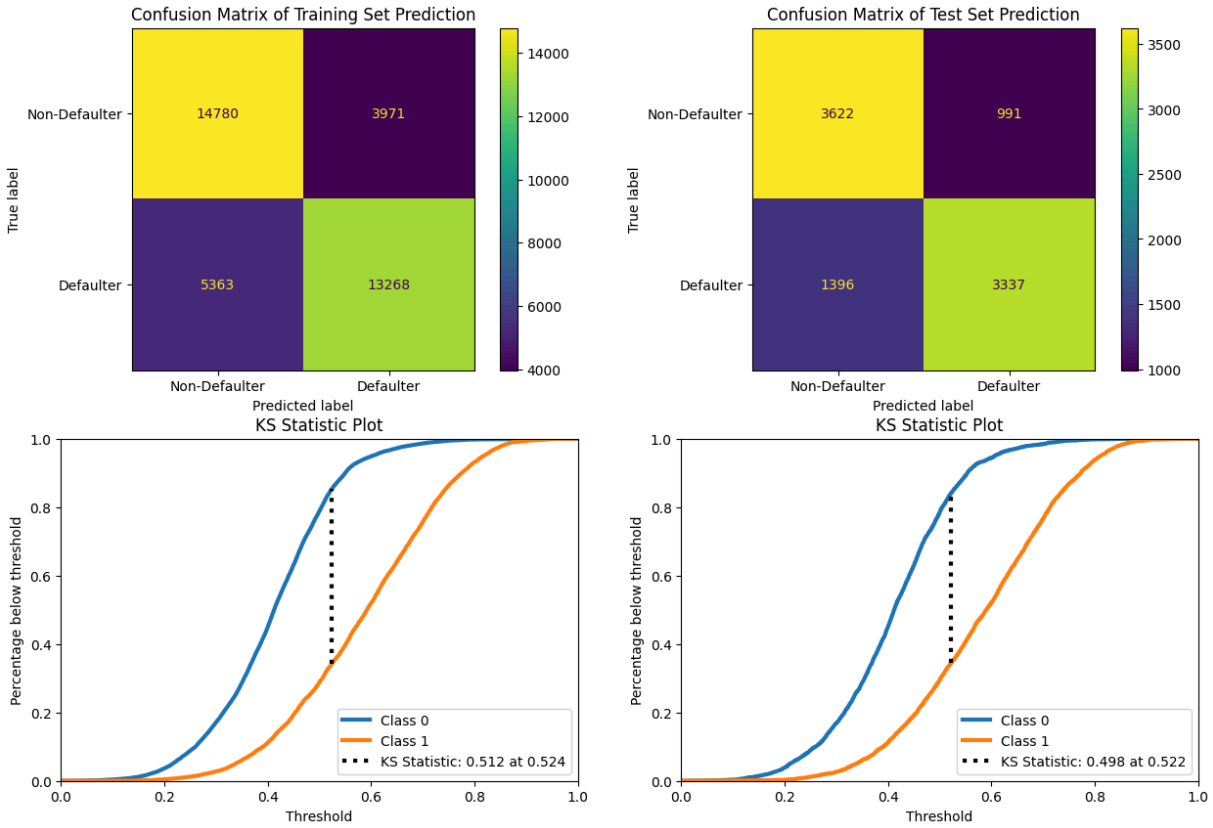
Out[85]:



In [86]:
```
#Best parameters
print('Best parameters:', lr_cv.best_params_)
lr_best=lr_cv.best_estimator_

#predicting the probabilities of test data
lr_best_test_proba=lr_best.predict_proba(X_test_scaled)
lr_best_train_proba=lr_best.predict_proba(X_train_scaled)

 #predicting the values of y from x via model
y_best_pred_test = lr_best.predict(X_test_scaled)
y_best_pred_train = lr_best.predict(X_train_scaled)
```

```
Best parameters: {'C': 0.0001, 'penalty': 'l2'}
```

In [87]:
```
#visualize the model performence
model_vis(y_train,y_test,y_best_pred_train,y_best_pred_test,lr_best_train_pr
```



In [88]:
```
#view all the metric score in a dataframe
metric_score('Tuned Logistic_Regression',y_test,y_best_pred_test,lr_best_tes
```

Out[88]:

	Model	Accuracy	Precision	Recall	F1 Score	KS_Stat
0	Logistic_Regression	0.765033	0.811901	0.697655	0.750455	0.536
1	Tuned Logistic_Regression	0.744597	0.771026	0.705050	0.736563	0.498

Which hyperparameter optimization technique have you used and why?

I used GridSearch CV to discover the best parameters for the model in order to improve its accuracy. GridSearchCV is the process of tweaking hyperparameters to determine the best values for a particular model.The value of hyperparameters has a substantial impact on model performance.

GridSearchCV examines the model for each combination of the values supplied in the dictionary using the Cross-Validation technique. As a result of utilising this function, we can calculate the accuracy/loss for each combination of hyperparameters and select the one with the best performance.

Have you seen any improvement? Note down the improvement with updates Evaluation metric Score Chart.

We can observe from the above that, with the exception of the ks statictics value, all evaluation parameters have changed. Only precision has been significantly reduced, while the rest have been slightly increased.Because the model dependant data is unbalanced and our target population is small in contrast to the non-target population, we prioritised recall value improvement.We can say that model performance has not improved significantly because the recall value has increased slightly after adjustment.

ML Model - 2 Decision Tree Classifier

In [89]:
```
# ML Model - 1 Implementation
model_dt=DecisionTreeClassifier(criterion="gini",random_state=100,min_sample

# Fit the Algorithm
model_dt.fit(X_train_scaled,y_train)
```

Out[89]:

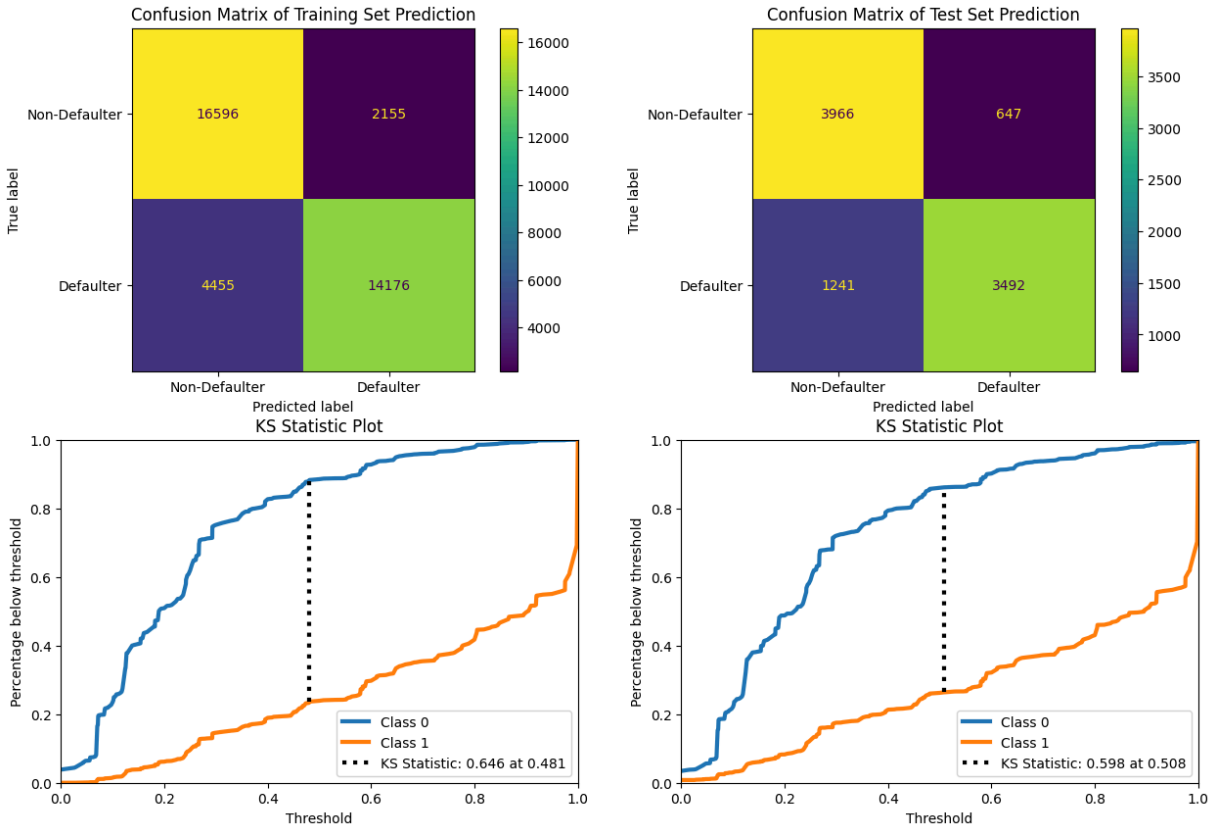
DecisionTreeClassifier

```
DecisionTreeClassifier(max_depth=10, min_samples_leaf=10, random_st
ate=100)
```

In [90]:
```
#predicting the probabilities of test data
dt_test_proba=model_dt.predict_proba(X_test_scaled)
dt_train_proba=model_dt.predict_proba(X_train_scaled)
 #predicting the values of y from x via model
y_pred_test = model_dt.predict(X_test_scaled)
y_pred_train = model_dt.predict(X_train_scaled)
```

1. Explain the ML Model used and it's performance using Evaluation metric Score Chart.

In [91]:
```
#visualize the model performence
model_vis(y_train,y_test,y_pred_train,y_pred_test,dt_train_proba,dt_test_pro
```



In [92]:
```
#view all the metric score in a dataframe
metric_score('Decision_Tree_Classifier',y_test,y_pred_test,dt_test_proba)
```

Out[92]:

	Model	Accuracy	Precision	Recall	F1 Score	KS_Stat
0	Logistic_Regression	0.765033	0.811901	0.697655	0.750455	0.536
1	Tuned Logistic_Regression	0.744597	0.771026	0.705050	0.736563	0.498
2	Decision_Tree_Classifier	0.797988	0.843682	0.737798	0.787196	0.598

2. Cross- Validation & Hyperparameter Tuning

In [93]:
```
# ML Model - 1 Implementation with hyperparameter optimization techniques (i
dt=DecisionTreeClassifier()
param_grid = param_dict = {'max_depth': [20,25,30,35],
              'min_samples_split':[50,100,150],
              'min_samples_leaf': [40,50,60]}
dt_cv=GridSearchCV(dt, param_grid, scoring = 'recall',n_jobs = -1, verbose =
# Fit the Algorithm
dt_cv.fit(X_train_scaled,y_train)
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
[CV 4/5] END max_depth=20, min_samples_leaf=40, min_samples_split=50;, score=
0.754 total time=   0.4s
[CV 3/5] END max_depth=20, min_samples_leaf=40, min_samples_split=50;, score=
0.729 total time=   0.4s
[CV 1/5] END max_depth=20, min_samples_leaf=40, min_samples_split=50;, score=
0.760 total time=   0.4s
[CV 2/5] END max_depth=20, min_samples_leaf=40, min_samples_split=100;, score
=0.737 total time=   0.4s
[CV 2/5] END max_depth=20, min_samples_leaf=40, min_samples_split=50;, score=
0.742 total time=   0.5s
[CV 3/5] END max_depth=20, min_samples_leaf=40, min_samples_split=100;, score
=0.732 total time=   0.4s
[CV 1/5] END max_depth=20, min_samples_leaf=40, min_samples_split=100;, score
=0.756 total time=   0.4s
[CV 5/5] END max_depth=20, min_samples_leaf=40, min_samples_split=50;, score=
0.744 total time=   0.4s
[CV 4/5] END max_depth=20, min_samples_leaf=40, min_samples_split=100;, score
=0.760 total time=   0.5s
[CV 5/5] END max_depth=20, min_samples_leaf=40, min_samples_split=100;, score
=0.740 total time=   0.5s
[CV 2/5] END max_depth=20, min_samples_leaf=40, min_samples_split=150;, score
=0.740 total time=   0.5s
[CV 1/5] END max_depth=20, min_samples_leaf=40, min_samples_split=150;, score
=0.754 total time=   0.5s
[CV 3/5] END max_depth=20, min_samples_leaf=40, min_samples_split=150;, score
=0.728 total time=   0.5s
[CV 4/5] END max_depth=20, min_samples_leaf=40, min_samples_split=150;, score
=0.759 total time=   0.5s
[CV 5/5] END max_depth=20, min_samples_leaf=40, min_samples_split=150;, score
=0.747 total time=   0.5s
[CV 1/5] END max_depth=20, min_samples_leaf=50, min_samples_split=50;, score=
0.757 total time=   0.5s
[CV 3/5] END max_depth=20, min_samples_leaf=50, min_samples_split=50;, score=
0.724 total time=   0.4s
[CV 5/5] END max_depth=20, min_samples_leaf=50, min_samples_split=50;, score=
0.739 total time=   0.4s
[CV 4/5] END max_depth=20, min_samples_leaf=50, min_samples_split=50;, score=
0.749 total time=   0.5s
[CV 1/5] END max_depth=20, min_samples_leaf=50, min_samples_split=100;, score
=0.757 total time=   0.4s
[CV 3/5] END max_depth=20, min_samples_leaf=50, min_samples_split=100;, score
=0.725 total time=   0.4s
[CV 2/5] END max_depth=20, min_samples_leaf=50, min_samples_split=50;, score=
0.734 total time=   0.5s
[CV 2/5] END max_depth=20, min_samples_leaf=50, min_samples_split=100;, score
=0.734 total time=   0.5s
[CV 4/5] END max_depth=20, min_samples_leaf=50, min_samples_split=100;, score
=0.749 total time=   0.4s
[CV 5/5] END max_depth=20, min_samples_leaf=50, min_samples_split=100;, score
=0.739 total time=   0.5s
[CV 1/5] END max_depth=20, min_samples_leaf=50, min_samples_split=150;, score
=0.756 total time=   0.5s
[CV 2/5] END max_depth=20, min_samples_leaf=50, min_samples_split=150;, score
=0.739 total time=   0.5s
[CV 3/5] END max_depth=20, min_samples_leaf=50, min_samples_split=150;, score
```

```
=0.722 total time=   0.5s
[CV 4/5] END max_depth=20, min_samples_leaf=50, min_samples_split=150;, score
=0.752 total time=   0.5s
[CV 5/5] END max_depth=20, min_samples_leaf=50, min_samples_split=150;, score
=0.742 total time=   0.5s
[CV 2/5] END max_depth=20, min_samples_leaf=60, min_samples_split=50;, score=
0.737 total time=   0.5s
[CV 1/5] END max_depth=20, min_samples_leaf=60, min_samples_split=50;, score=
0.759 total time=   0.6s
[CV 3/5] END max_depth=20, min_samples_leaf=60, min_samples_split=50;, score=
0.724 total time=   0.5s
[CV 4/5] END max_depth=20, min_samples_leaf=60, min_samples_split=50;, score=
0.750 total time=   0.5s
[CV 3/5] END max_depth=20, min_samples_leaf=60, min_samples_split=100;, score
=0.724 total time=   0.4s
[CV 2/5] END max_depth=20, min_samples_leaf=60, min_samples_split=100;, score
=0.737 total time=   0.5s
[CV 5/5] END max_depth=20, min_samples_leaf=60, min_samples_split=50;, score=
0.733 total time=   0.5s
[CV 1/5] END max_depth=20, min_samples_leaf=60, min_samples_split=100;, score
=0.759 total time=   0.5s
[CV 4/5] END max_depth=20, min_samples_leaf=60, min_samples_split=100;, score
=0.750 total time=   0.5s
[CV 5/5] END max_depth=20, min_samples_leaf=60, min_samples_split=100;, score
=0.733 total time=   0.5s
[CV 1/5] END max_depth=20, min_samples_leaf=60, min_samples_split=150;, score
=0.759 total time=   0.4s
[CV 3/5] END max_depth=20, min_samples_leaf=60, min_samples_split=150;, score
=0.723 total time=   0.4s
[CV 2/5] END max_depth=20, min_samples_leaf=60, min_samples_split=150;, score
=0.744 total time=   0.5s
[CV 4/5] END max_depth=20, min_samples_leaf=60, min_samples_split=150;, score
=0.745 total time=   0.5s
[CV 1/5] END max_depth=25, min_samples_leaf=40, min_samples_split=50;, score=
0.760 total time=   0.5s
[CV 5/5] END max_depth=20, min_samples_leaf=60, min_samples_split=150;, score
=0.735 total time=   0.5s
[CV 2/5] END max_depth=25, min_samples_leaf=40, min_samples_split=50;, score=
0.742 total time=   0.6s
[CV 3/5] END max_depth=25, min_samples_leaf=40, min_samples_split=50;, score=
0.728 total time=   0.6s
[CV 4/5] END max_depth=25, min_samples_leaf=40, min_samples_split=50;, score=
0.753 total time=   0.5s
[CV 5/5] END max_depth=25, min_samples_leaf=40, min_samples_split=50;, score=
0.744 total time=   0.5s
[CV 1/5] END max_depth=25, min_samples_leaf=40, min_samples_split=100;, score
=0.756 total time=   0.5s
[CV 2/5] END max_depth=25, min_samples_leaf=40, min_samples_split=100;, score
=0.737 total time=   0.4s
[CV 3/5] END max_depth=25, min_samples_leaf=40, min_samples_split=100;, score
=0.731 total time=   0.4s
[CV 4/5] END max_depth=25, min_samples_leaf=40, min_samples_split=100;, score
=0.760 total time=   0.4s
[CV 1/5] END max_depth=25, min_samples_leaf=40, min_samples_split=150;, score
=0.754 total time=   0.4s
[CV 5/5] END max_depth=25, min_samples_leaf=40, min_samples_split=100;, score
```

```
=0.740 total time=   0.5s
[CV 2/5] END max_depth=25, min_samples_leaf=40, min_samples_split=150;, score
=0.740 total time=   0.4s
[CV 3/5] END max_depth=25, min_samples_leaf=40, min_samples_split=150;, score
=0.728 total time=   0.4s
[CV 4/5] END max_depth=25, min_samples_leaf=40, min_samples_split=150;, score
=0.759 total time=   0.4s
[CV 5/5] END max_depth=25, min_samples_leaf=40, min_samples_split=150;, score
=0.747 total time=   0.4s
[CV 1/5] END max_depth=25, min_samples_leaf=50, min_samples_split=50;, score=
0.757 total time=   0.4s
[CV 2/5] END max_depth=25, min_samples_leaf=50, min_samples_split=50;, score=
0.734 total time=   0.4s
[CV 3/5] END max_depth=25, min_samples_leaf=50, min_samples_split=50;, score=
0.725 total time=   0.4s
[CV 4/5] END max_depth=25, min_samples_leaf=50, min_samples_split=50;, score=
0.749 total time=   0.4s
[CV 1/5] END max_depth=25, min_samples_leaf=50, min_samples_split=100;, score
=0.757 total time=   0.4s
[CV 5/5] END max_depth=25, min_samples_leaf=50, min_samples_split=50;, score=
0.739 total time=   0.4s
[CV 3/5] END max_depth=25, min_samples_leaf=50, min_samples_split=100;, score
=0.726 total time=   0.4s
[CV 2/5] END max_depth=25, min_samples_leaf=50, min_samples_split=100;, score
=0.734 total time=   0.4s
[CV 4/5] END max_depth=25, min_samples_leaf=50, min_samples_split=100;, score
=0.748 total time=   0.4s
[CV 5/5] END max_depth=25, min_samples_leaf=50, min_samples_split=100;, score
=0.739 total time=   0.4s
[CV 1/5] END max_depth=25, min_samples_leaf=50, min_samples_split=150;, score
=0.756 total time=   0.4s
[CV 2/5] END max_depth=25, min_samples_leaf=50, min_samples_split=150;, score
=0.739 total time=   0.4s
[CV 3/5] END max_depth=25, min_samples_leaf=50, min_samples_split=150;, score
=0.722 total time=   0.4s
[CV 4/5] END max_depth=25, min_samples_leaf=50, min_samples_split=150;, score
=0.752 total time=   0.4s
[CV 5/5] END max_depth=25, min_samples_leaf=50, min_samples_split=150;, score
=0.742 total time=   0.4s
[CV 1/5] END max_depth=25, min_samples_leaf=60, min_samples_split=50;, score=
0.759 total time=   0.4s
[CV 3/5] END max_depth=25, min_samples_leaf=60, min_samples_split=50;, score=
0.725 total time=   0.4s
[CV 2/5] END max_depth=25, min_samples_leaf=60, min_samples_split=50;, score=
0.737 total time=   0.4s
[CV 4/5] END max_depth=25, min_samples_leaf=60, min_samples_split=50;, score=
0.747 total time=   0.4s
[CV 5/5] END max_depth=25, min_samples_leaf=60, min_samples_split=50;, score=
0.733 total time=   0.5s
[CV 1/5] END max_depth=25, min_samples_leaf=60, min_samples_split=100;, score
=0.759 total time=   0.4s
[CV 2/5] END max_depth=25, min_samples_leaf=60, min_samples_split=100;, score
=0.737 total time=   0.4s
[CV 3/5] END max_depth=25, min_samples_leaf=60, min_samples_split=100;, score
=0.724 total time=   0.4s
[CV 1/5] END max_depth=25, min_samples_leaf=60, min_samples_split=150;, score
```

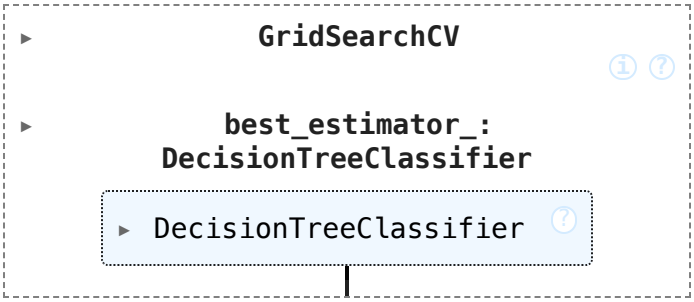
```
=0.759 total time=   0.4s
[CV 5/5] END max_depth=25, min_samples_leaf=60, min_samples_split=100;, score
=0.733 total time=   0.4s
[CV 4/5] END max_depth=25, min_samples_leaf=60, min_samples_split=100;, score
=0.750 total time=   0.5s
[CV 2/5] END max_depth=25, min_samples_leaf=60, min_samples_split=150;, score
=0.744 total time=   0.4s
[CV 3/5] END max_depth=25, min_samples_leaf=60, min_samples_split=150;, score
=0.723 total time=   0.4s
[CV 4/5] END max_depth=25, min_samples_leaf=60, min_samples_split=150;, score
=0.745 total time=   0.4s
[CV 5/5] END max_depth=25, min_samples_leaf=60, min_samples_split=150;, score
=0.735 total time=   0.4s
[CV 3/5] END max_depth=30, min_samples_leaf=40, min_samples_split=50;, score=
0.729 total time=   0.4s
[CV 2/5] END max_depth=30, min_samples_leaf=40, min_samples_split=50;, score=
0.742 total time=   0.4s
[CV 1/5] END max_depth=30, min_samples_leaf=40, min_samples_split=50;, score=
0.760 total time=   0.5s
[CV 4/5] END max_depth=30, min_samples_leaf=40, min_samples_split=50;, score=
0.754 total time=   0.4s
[CV 5/5] END max_depth=30, min_samples_leaf=40, min_samples_split=50;, score=
0.744 total time=   0.4s
[CV 1/5] END max_depth=30, min_samples_leaf=40, min_samples_split=100;, score
=0.756 total time=   0.4s
[CV 2/5] END max_depth=30, min_samples_leaf=40, min_samples_split=100;, score
=0.737 total time=   0.4s
[CV 3/5] END max_depth=30, min_samples_leaf=40, min_samples_split=100;, score
=0.731 total time=   0.4s
[CV 4/5] END max_depth=30, min_samples_leaf=40, min_samples_split=100;, score
=0.760 total time=   0.4s
[CV 5/5] END max_depth=30, min_samples_leaf=40, min_samples_split=100;, score
=0.740 total time=   0.5s
[CV 1/5] END max_depth=30, min_samples_leaf=40, min_samples_split=150;, score
=0.754 total time=   0.5s
[CV 2/5] END max_depth=30, min_samples_leaf=40, min_samples_split=150;, score
=0.740 total time=   0.4s
[CV 3/5] END max_depth=30, min_samples_leaf=40, min_samples_split=150;, score
=0.728 total time=   0.4s
[CV 4/5] END max_depth=30, min_samples_leaf=40, min_samples_split=150;, score
=0.759 total time=   0.4s
[CV 5/5] END max_depth=30, min_samples_leaf=40, min_samples_split=150;, score
=0.747 total time=   0.4s
[CV 1/5] END max_depth=30, min_samples_leaf=50, min_samples_split=50;, score=
0.757 total time=   0.5s
[CV 2/5] END max_depth=30, min_samples_leaf=50, min_samples_split=50;, score=
0.734 total time=   0.4s
[CV 3/5] END max_depth=30, min_samples_leaf=50, min_samples_split=50;, score=
0.726 total time=   0.4s
[CV 4/5] END max_depth=30, min_samples_leaf=50, min_samples_split=50;, score=
0.748 total time=   0.4s
[CV 5/5] END max_depth=30, min_samples_leaf=50, min_samples_split=50;, score=
0.739 total time=   0.4s
[CV 2/5] END max_depth=30, min_samples_leaf=50, min_samples_split=100;, score
=0.734 total time=   0.4s
[CV 3/5] END max_depth=30, min_samples_leaf=50, min_samples_split=100;, score
```

```
=0.724 total time=   0.4s
[CV 1/5] END max_depth=30, min_samples_leaf=50, min_samples_split=100;, score
=0.757 total time=   0.5s
[CV 5/5] END max_depth=30, min_samples_leaf=50, min_samples_split=100;, score
=0.739 total time=   0.4s
[CV 4/5] END max_depth=30, min_samples_leaf=50, min_samples_split=100;, score
=0.748 total time=   0.4s
[CV 2/5] END max_depth=30, min_samples_leaf=50, min_samples_split=150;, score
=0.739 total time=   0.4s
[CV 1/5] END max_depth=30, min_samples_leaf=50, min_samples_split=150;, score
=0.756 total time=   0.4s
[CV 3/5] END max_depth=30, min_samples_leaf=50, min_samples_split=150;, score
=0.722 total time=   0.4s
[CV 4/5] END max_depth=30, min_samples_leaf=50, min_samples_split=150;, score
=0.752 total time=   0.4s
[CV 5/5] END max_depth=30, min_samples_leaf=50, min_samples_split=150;, score
=0.742 total time=   0.4s
[CV 1/5] END max_depth=30, min_samples_leaf=60, min_samples_split=50;, score=
0.759 total time=   0.4s
[CV 3/5] END max_depth=30, min_samples_leaf=60, min_samples_split=50;, score=
0.724 total time=   0.4s
[CV 4/5] END max_depth=30, min_samples_leaf=60, min_samples_split=50;, score=
0.747 total time=   0.4s
[CV 1/5] END max_depth=30, min_samples_leaf=60, min_samples_split=100;, score
=0.759 total time=   0.4s
[CV 2/5] END max_depth=30, min_samples_leaf=60, min_samples_split=50;, score=
0.737 total time=   0.5s
[CV 5/5] END max_depth=30, min_samples_leaf=60, min_samples_split=50;, score=
0.733 total time=   0.4s
[CV 2/5] END max_depth=30, min_samples_leaf=60, min_samples_split=100;, score
=0.737 total time=   0.4s
[CV 3/5] END max_depth=30, min_samples_leaf=60, min_samples_split=100;, score
=0.724 total time=   0.4s
[CV 4/5] END max_depth=30, min_samples_leaf=60, min_samples_split=100;, score
=0.747 total time=   0.4s
[CV 3/5] END max_depth=30, min_samples_leaf=60, min_samples_split=150;, score
=0.723 total time=   0.3s
[CV 2/5] END max_depth=30, min_samples_leaf=60, min_samples_split=150;, score
=0.744 total time=   0.3s
[CV 5/5] END max_depth=30, min_samples_leaf=60, min_samples_split=100;, score
=0.733 total time=   0.4s
[CV 4/5] END max_depth=30, min_samples_leaf=60, min_samples_split=150;, score
=0.744 total time=   0.4s
[CV 1/5] END max_depth=30, min_samples_leaf=60, min_samples_split=150;, score
=0.759 total time=   0.4s
[CV 5/5] END max_depth=30, min_samples_leaf=60, min_samples_split=150;, score
=0.735 total time=   0.3s
[CV 1/5] END max_depth=35, min_samples_leaf=40, min_samples_split=50;, score=
0.760 total time=   0.4s
[CV 2/5] END max_depth=35, min_samples_leaf=40, min_samples_split=50;, score=
0.742 total time=   0.5s
[CV 3/5] END max_depth=35, min_samples_leaf=40, min_samples_split=50;, score=
0.729 total time=   0.4s
[CV 5/5] END max_depth=35, min_samples_leaf=40, min_samples_split=50;, score=
0.744 total time=   0.4s
[CV 4/5] END max_depth=35, min_samples_leaf=40, min_samples_split=50;, score=
```

```
0.753 total time=   0.4s
[CV 1/5] END max_depth=35, min_samples_leaf=40, min_samples_split=100;, score
=0.756 total time=   0.4s
[CV 2/5] END max_depth=35, min_samples_leaf=40, min_samples_split=100;, score
=0.737 total time=   0.5s
[CV 3/5] END max_depth=35, min_samples_leaf=40, min_samples_split=100;, score
=0.732 total time=   0.4s
[CV 4/5] END max_depth=35, min_samples_leaf=40, min_samples_split=100;, score
=0.760 total time=   0.4s
[CV 5/5] END max_depth=35, min_samples_leaf=40, min_samples_split=100;, score
=0.740 total time=   0.5s
[CV 3/5] END max_depth=35, min_samples_leaf=40, min_samples_split=150;, score
=0.728 total time=   0.4s
[CV 4/5] END max_depth=35, min_samples_leaf=40, min_samples_split=150;, score
=0.759 total time=   0.4s
[CV 1/5] END max_depth=35, min_samples_leaf=40, min_samples_split=150;, score
=0.754 total time=   0.4s
[CV 2/5] END max_depth=35, min_samples_leaf=40, min_samples_split=150;, score
=0.740 total time=   0.4s
[CV 5/5] END max_depth=35, min_samples_leaf=40, min_samples_split=150;, score
=0.747 total time=   0.4s
[CV 1/5] END max_depth=35, min_samples_leaf=50, min_samples_split=50;, score=
0.757 total time=   0.5s
[CV 2/5] END max_depth=35, min_samples_leaf=50, min_samples_split=50;, score=
0.734 total time=   0.5s
[CV 3/5] END max_depth=35, min_samples_leaf=50, min_samples_split=50;, score=
0.725 total time=   0.6s
[CV 4/5] END max_depth=35, min_samples_leaf=50, min_samples_split=50;, score=
0.748 total time=   0.6s
[CV 1/5] END max_depth=35, min_samples_leaf=50, min_samples_split=100;, score
=0.757 total time=   0.7s
[CV 5/5] END max_depth=35, min_samples_leaf=50, min_samples_split=50;, score=
0.739 total time=   0.7s
[CV 2/5] END max_depth=35, min_samples_leaf=50, min_samples_split=100;, score
=0.734 total time=   0.7s
[CV 4/5] END max_depth=35, min_samples_leaf=50, min_samples_split=100;, score
=0.749 total time=   0.5s
[CV 3/5] END max_depth=35, min_samples_leaf=50, min_samples_split=100;, score
=0.725 total time=   0.7s
[CV 5/5] END max_depth=35, min_samples_leaf=50, min_samples_split=100;, score
=0.739 total time=   0.6s
[CV 1/5] END max_depth=35, min_samples_leaf=50, min_samples_split=150;, score
=0.756 total time=   0.4s
[CV 2/5] END max_depth=35, min_samples_leaf=50, min_samples_split=150;, score
=0.739 total time=   0.5s
[CV 3/5] END max_depth=35, min_samples_leaf=50, min_samples_split=150;, score
=0.722 total time=   0.4s
[CV 4/5] END max_depth=35, min_samples_leaf=50, min_samples_split=150;, score
=0.752 total time=   0.5s
[CV 5/5] END max_depth=35, min_samples_leaf=50, min_samples_split=150;, score
=0.742 total time=   0.4s
[CV 1/5] END max_depth=35, min_samples_leaf=60, min_samples_split=50;, score=
0.759 total time=   0.4s
[CV 2/5] END max_depth=35, min_samples_leaf=60, min_samples_split=50;, score=
0.737 total time=   0.5s
[CV 3/5] END max_depth=35, min_samples_leaf=60, min_samples_split=50;, score=
```

```
0.724 total time=   0.4s
[CV 4/5] END max_depth=35, min_samples_leaf=60, min_samples_split=50;, score=
0.750 total time=   0.4s
[CV 5/5] END max_depth=35, min_samples_leaf=60, min_samples_split=50;, score=
0.733 total time=   0.4s
[CV 1/5] END max_depth=35, min_samples_leaf=60, min_samples_split=100;, score
=0.759 total time=   0.4s
[CV 3/5] END max_depth=35, min_samples_leaf=60, min_samples_split=100;, score
=0.724 total time=   0.4s
[CV 2/5] END max_depth=35, min_samples_leaf=60, min_samples_split=100;, score
=0.737 total time=   0.4s
[CV 4/5] END max_depth=35, min_samples_leaf=60, min_samples_split=100;, score
=0.747 total time=   0.4s
[CV 5/5] END max_depth=35, min_samples_leaf=60, min_samples_split=100;, score
=0.733 total time=   0.4s
[CV 1/5] END max_depth=35, min_samples_leaf=60, min_samples_split=150;, score
=0.759 total time=   0.4s
[CV 2/5] END max_depth=35, min_samples_leaf=60, min_samples_split=150;, score
=0.744 total time=   0.4s
[CV 3/5] END max_depth=35, min_samples_leaf=60, min_samples_split=150;, score
=0.722 total time=   0.3s
[CV 4/5] END max_depth=35, min_samples_leaf=60, min_samples_split=150;, score
=0.745 total time=   0.3s
[CV 5/5] END max_depth=35, min_samples_leaf=60, min_samples_split=150;, score
=0.735 total time=   0.3s
```

Out[93]:



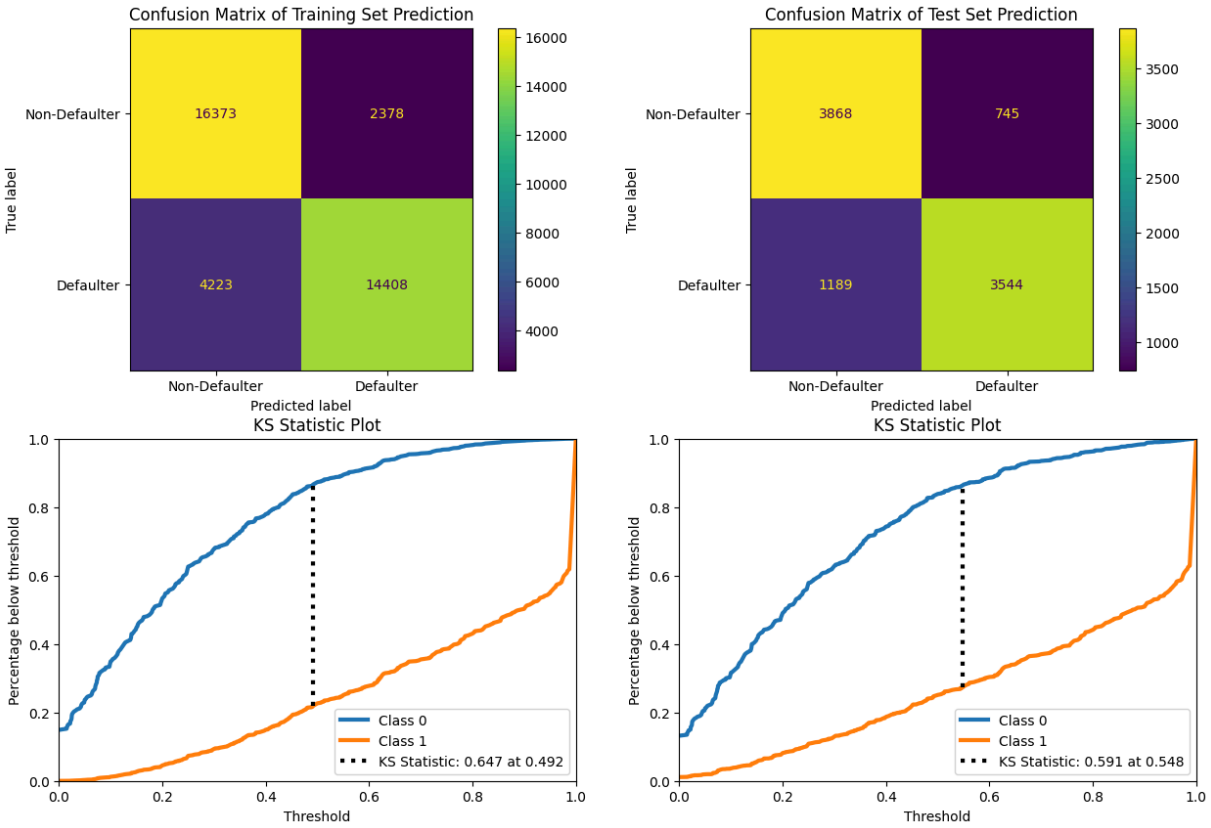
In [94]:

```
#Best parameters
print('Best parameters:', dt_cv.best_params_)
dt_best=dt_cv.best_estimator_

#predicting the probabilities of test data
dt_best_test_proba=dt_best.predict_proba(X_test_scaled)
dt_best_train_proba=dt_best.predict_proba(X_train_scaled)

 #predicting the values of y from x via model
y_best_pred_test = dt_best.predict(X_test_scaled)
y_best_pred_train = dt_best.predict(X_train_scaled)
```

```
Best parameters: {'max_depth': 30, 'min_samples_leaf': 40, 'min_samples_spli
t': 50}
```

In [95]:

```
#visualize the model performence
model_vis(y_train,y_test,y_best_pred_train,y_best_pred_test,dt_best_train_pr
```



In [96]:
```
#view all the metric score in a dataframe
metric_score('Tuned Decision_Tree_Classifier',y_test,y_best_pred_test,dt_bes
```

Out[96]:

	Model	Accuracy	Precision	Recall	F1 Score	KS_Stat
0	Logistic_Regression	0.765033	0.811901	0.697655	0.750455	0.536
1	Tuned Logistic_Regression	0.744597	0.771026	0.705050	0.736563	0.498
2	Decision_Tree_Classifier	0.797988	0.843682	0.737798	0.787196	0.598
3	Tuned Decision_Tree_Classifier	0.793067	0.826300	0.748785	0.785635	0.591

Which hyperparameter optimization technique have you used and why?

I used GridSearch CV to discover the best parameters for the model in order to improve its accuracy. GridSearchCV is the process of tweaking hyperparameters to determine the best values for a particular model.The value of hyperparameters has a substantial impact on model performance.

GridSearchCV examines the model for each combination of the values supplied in the dictionary using the Cross-Validation technique. As a result of utilising this function, we can calculate the accuracy/loss for each combination of hyperparameters and select the one with the best performance.

Have you seen any improvement? Note down the improvement with updates Evaluation metric Score Chart.

We can observe from the above that, with the exception of the ks statictics value, all evaluation parameters have changed. Only precision has been significantly reduced, while the rest have been slightly increased.We can say that model performance has not improved significantly because the recall value has slightly increased after adjustment.

3. Explain each evaluation metric's indication towards business and the business impact pf the ML model used.

I'll be using Recall primarily for model evaluation because False Negative indicates that a person will not default when they actually do. And acknowledging defaulter clients as non-defaulters will result in a large loss for the bank, thus we must reduce False Negative, and as False Negative decreases, Recall will grow.

ML Model - 3 Random Forest

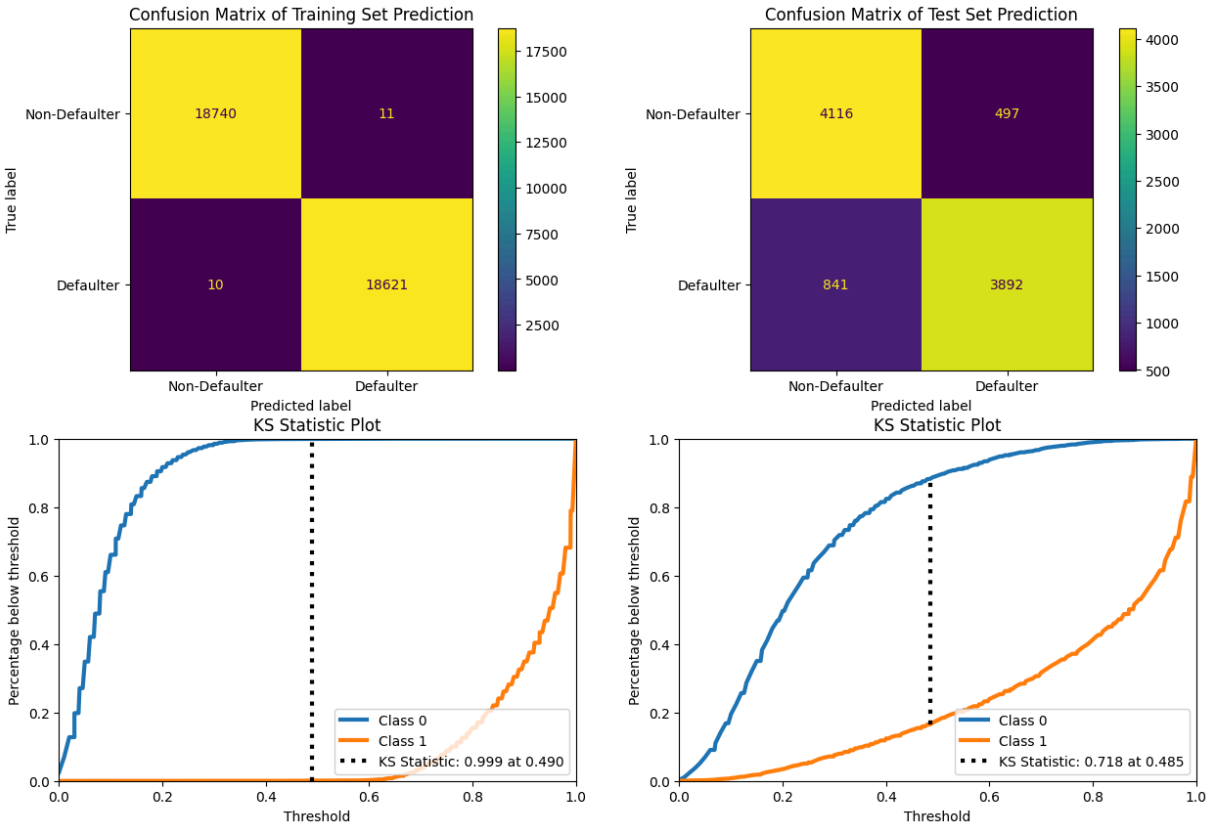
In [97]:
```
# ML Model - 1 Implementation
model_rf= RandomForestClassifier(n_estimators = 100,criterion = 'entropy',ra
# Fit the Algorithm
model_rf.fit(X_train_scaled,y_train)
```

Out[97]:
```
RandomForestClassifier(criterion='entropy', random_state=0)
```

In [98]:
```
#predicting the probabilities of test data
rf_test_proba=model_rf.predict_proba(X_test_scaled)
rf_train_proba=model_rf.predict_proba(X_train_scaled)
 #predicting the values of y from x via model
y_pred_test = model_rf.predict(X_test_scaled)
y_pred_train = model_rf.predict(X_train_scaled)
```

1. Explain the ML Model used and it's performance using Evaluation metric Score Chart.

In [99]:
```
#visualize the model performence
model_vis(y_train,y_test,y_pred_train,y_pred_test,rf_train_proba,rf_test_prc
```



In [100…
```
#view all the metric score in a dataframe
metric_score('Random_Forrest_Classifier',y_test,y_pred_test,rf_test_proba)
```

Out[100]:

	Model	Accuracy	Precision	Recall	F1 Score	KS_Stat
0	Logistic_Regression	0.765033	0.811901	0.697655	0.750455	0.536
1	Tuned Logistic_Regression	0.744597	0.771026	0.705050	0.736563	0.498
2	Decision_Tree_Classifier	0.797988	0.843682	0.737798	0.787196	0.598
3	Tuned Decision_Tree_Classifier	0.793067	0.826300	0.748785	0.785635	0.591
4	Random_Forrest_Classifier	0.856837	0.886762	0.822311	0.853322	0.718

2. Cross- Validation & Hyperparameter Tuning

In [101…
```
# ML Model - 1 Implementation with hyperparameter optimization techniques (i
rf=RandomForestClassifier()
param_grid = {'n_estimators' : [100,250],
              'max_depth' : [None,80,100],
              'min_samples_split' : [10,2],
              'min_samples_leaf' : [1,10]}
rf_cv=GridSearchCV(rf, param_grid, scoring = 'recall',n_jobs = -1, verbose =
# Fit the Algorithm
rf_cv.fit(X_train_scaled,y_train)
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
[CV 5/5] END max_depth=None, min_samples_leaf=1, min_samples_split=10, n_esti
mators=100;, score=0.798 total time=   7.3s
[CV 4/5] END max_depth=None, min_samples_leaf=1, min_samples_split=10, n_esti
mators=100;, score=0.807 total time=   7.3s
[CV 1/5] END max_depth=None, min_samples_leaf=1, min_samples_split=10, n_esti
mators=100;, score=0.811 total time=   7.4s
[CV 3/5] END max_depth=None, min_samples_leaf=1, min_samples_split=10, n_esti
mators=100;, score=0.786 total time=   7.5s
[CV 2/5] END max_depth=None, min_samples_leaf=1, min_samples_split=10, n_esti
mators=100;, score=0.798 total time=   7.7s
[CV 2/5] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estim
ators=100;, score=0.808 total time=   7.7s
[CV 1/5] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estim
ators=100;, score=0.826 total time=   8.3s
[CV 3/5] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estim
ators=100;, score=0.805 total time=   8.2s
[CV 1/5] END max_depth=None, min_samples_leaf=1, min_samples_split=10, n_esti
mators=250;, score=0.813 total time=  18.7s
[CV 2/5] END max_depth=None, min_samples_leaf=1, min_samples_split=10, n_esti
mators=250;, score=0.797 total time=  18.9s
[CV 3/5] END max_depth=None, min_samples_leaf=1, min_samples_split=10, n_esti
mators=250;, score=0.788 total time=  19.3s
[CV 4/5] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estim
ators=100;, score=0.824 total time=   8.1s
[CV 5/5] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estim
ators=100;, score=0.809 total time=   8.1s
[CV 5/5] END max_depth=None, min_samples_leaf=1, min_samples_split=10, n_esti
mators=250;, score=0.799 total time=  18.7s
[CV 4/5] END max_depth=None, min_samples_leaf=1, min_samples_split=10, n_esti
mators=250;, score=0.804 total time=  19.4s
[CV 1/5] END max_depth=None, min_samples_leaf=10, min_samples_split=10, n_est
imators=100;, score=0.786 total time=   6.5s
[CV 2/5] END max_depth=None, min_samples_leaf=10, min_samples_split=10, n_est
imators=100;, score=0.770 total time=   6.2s
[CV 3/5] END max_depth=None, min_samples_leaf=10, min_samples_split=10, n_est
imators=100;, score=0.767 total time=   6.6s
[CV 1/5] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estim
ators=250;, score=0.831 total time=  20.5s
[CV 4/5] END max_depth=None, min_samples_leaf=10, min_samples_split=10, n_est
imators=100;, score=0.782 total time=   7.2s
[CV 2/5] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estim
ators=250;, score=0.812 total time=  20.0s
[CV 5/5] END max_depth=None, min_samples_leaf=10, min_samples_split=10, n_est
imators=100;, score=0.773 total time=   7.4s
[CV 3/5] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estim
ators=250;, score=0.810 total time=  20.8s
[CV 4/5] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estim
ators=250;, score=0.821 total time=  20.7s
[CV 5/5] END max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estim
ators=250;, score=0.814 total time=  19.8s
[CV 1/5] END max_depth=None, min_samples_leaf=10, min_samples_split=2, n_esti
mators=100;, score=0.788 total time=   7.1s
[CV 2/5] END max_depth=None, min_samples_leaf=10, min_samples_split=2, n_esti
mators=100;, score=0.770 total time=   7.1s
[CV 3/5] END max_depth=None, min_samples_leaf=10, min_samples_split=2, n_esti
```

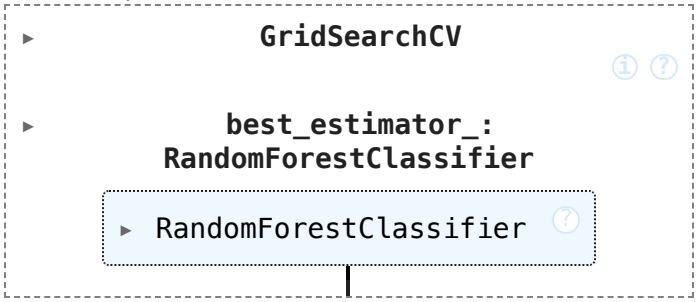
```
mators=100;, score=0.764 total time=   7.7s
[CV 1/5] END max_depth=None, min_samples_leaf=10, min_samples_split=10, n_est
imators=250;, score=0.787 total time=  18.3s
[CV 4/5] END max_depth=None, min_samples_leaf=10, min_samples_split=2, n_esti
mators=100;, score=0.784 total time=   7.1s
[CV 2/5] END max_depth=None, min_samples_leaf=10, min_samples_split=10, n_est
imators=250;, score=0.772 total time=  18.2s
[CV 5/5] END max_depth=None, min_samples_leaf=10, min_samples_split=2, n_esti
mators=100;, score=0.770 total time=   7.9s
[CV 4/5] END max_depth=None, min_samples_leaf=10, min_samples_split=10, n_est
imators=250;, score=0.783 total time=  17.5s
[CV 3/5] END max_depth=None, min_samples_leaf=10, min_samples_split=10, n_est
imators=250;, score=0.766 total time=  18.8s
[CV 5/5] END max_depth=None, min_samples_leaf=10, min_samples_split=10, n_est
imators=250;, score=0.773 total time=  17.1s
[CV 1/5] END max_depth=80, min_samples_leaf=1, min_samples_split=10, n_estima
tors=100;, score=0.809 total time=   7.1s
[CV 3/5] END max_depth=80, min_samples_leaf=1, min_samples_split=10, n_estima
tors=100;, score=0.791 total time=   7.0s
[CV 2/5] END max_depth=80, min_samples_leaf=1, min_samples_split=10, n_estima
tors=100;, score=0.795 total time=   7.6s
[CV 2/5] END max_depth=None, min_samples_leaf=10, min_samples_split=2, n_esti
mators=250;, score=0.771 total time=  16.6s
[CV 1/5] END max_depth=None, min_samples_leaf=10, min_samples_split=2, n_esti
mators=250;, score=0.788 total time=  17.4s
[CV 3/5] END max_depth=None, min_samples_leaf=10, min_samples_split=2, n_esti
mators=250;, score=0.765 total time=  16.4s
[CV 4/5] END max_depth=80, min_samples_leaf=1, min_samples_split=10, n_estima
tors=100;, score=0.803 total time=   7.6s
[CV 4/5] END max_depth=None, min_samples_leaf=10, min_samples_split=2, n_esti
mators=250;, score=0.784 total time=  17.7s
[CV 5/5] END max_depth=80, min_samples_leaf=1, min_samples_split=10, n_estima
tors=100;, score=0.795 total time=   8.9s
[CV 5/5] END max_depth=None, min_samples_leaf=10, min_samples_split=2, n_esti
mators=250;, score=0.773 total time=  18.1s
[CV 2/5] END max_depth=80, min_samples_leaf=1, min_samples_split=2, n_estimat
ors=100;, score=0.809 total time=   8.3s
[CV 1/5] END max_depth=80, min_samples_leaf=1, min_samples_split=2, n_estimat
ors=100;, score=0.826 total time=   8.8s
[CV 3/5] END max_depth=80, min_samples_leaf=1, min_samples_split=2, n_estimat
ors=100;, score=0.804 total time=   8.2s
[CV 1/5] END max_depth=80, min_samples_leaf=1, min_samples_split=10, n_estima
tors=250;, score=0.812 total time=  20.9s
[CV 3/5] END max_depth=80, min_samples_leaf=1, min_samples_split=10, n_estima
tors=250;, score=0.789 total time=  20.3s
[CV 4/5] END max_depth=80, min_samples_leaf=1, min_samples_split=2, n_estimat
ors=100;, score=0.826 total time=   8.1s
[CV 5/5] END max_depth=80, min_samples_leaf=1, min_samples_split=2, n_estimat
ors=100;, score=0.808 total time=   8.1s
[CV 2/5] END max_depth=80, min_samples_leaf=1, min_samples_split=10, n_estima
tors=250;, score=0.798 total time=  21.5s
[CV 5/5] END max_depth=80, min_samples_leaf=1, min_samples_split=10, n_estima
tors=250;, score=0.797 total time=  19.6s
[CV 4/5] END max_depth=80, min_samples_leaf=1, min_samples_split=10, n_estima
tors=250;, score=0.801 total time=  20.3s
[CV 1/5] END max_depth=80, min_samples_leaf=10, min_samples_split=10, n_estim
```

```
ators=100;, score=0.789 total time=   6.3s
[CV 2/5] END max_depth=80, min_samples_leaf=10, min_samples_split=10, n_estim
ators=100;, score=0.773 total time=   6.2s
[CV 3/5] END max_depth=80, min_samples_leaf=10, min_samples_split=10, n_estim
ators=100;, score=0.763 total time=   6.4s
[CV 1/5] END max_depth=80, min_samples_leaf=1, min_samples_split=2, n_estimat
ors=250;, score=0.828 total time=  21.2s
[CV 4/5] END max_depth=80, min_samples_leaf=10, min_samples_split=10, n_estim
ators=100;, score=0.783 total time=   9.0s
[CV 5/5] END max_depth=80, min_samples_leaf=10, min_samples_split=10, n_estim
ators=100;, score=0.773 total time=   8.9s
[CV 2/5] END max_depth=80, min_samples_leaf=1, min_samples_split=2, n_estimat
ors=250;, score=0.807 total time=  21.4s
[CV 4/5] END max_depth=80, min_samples_leaf=1, min_samples_split=2, n_estimat
ors=250;, score=0.824 total time=  20.4s
[CV 3/5] END max_depth=80, min_samples_leaf=1, min_samples_split=2, n_estimat
ors=250;, score=0.809 total time=  21.0s
[CV 5/5] END max_depth=80, min_samples_leaf=1, min_samples_split=2, n_estimat
ors=250;, score=0.815 total time=  20.9s
[CV 1/5] END max_depth=80, min_samples_leaf=10, min_samples_split=10, n_estim
ators=250;, score=0.789 total time=  19.5s
[CV 2/5] END max_depth=80, min_samples_leaf=10, min_samples_split=2, n_estima
tors=100;, score=0.773 total time=   9.2s
[CV 1/5] END max_depth=80, min_samples_leaf=10, min_samples_split=2, n_estima
tors=100;, score=0.787 total time=   9.4s
[CV 3/5] END max_depth=80, min_samples_leaf=10, min_samples_split=2, n_estima
tors=100;, score=0.769 total time=   8.5s
[CV 2/5] END max_depth=80, min_samples_leaf=10, min_samples_split=10, n_estim
ators=250;, score=0.771 total time=  19.5s
[CV 3/5] END max_depth=80, min_samples_leaf=10, min_samples_split=10, n_estim
ators=250;, score=0.765 total time=  19.2s
[CV 4/5] END max_depth=80, min_samples_leaf=10, min_samples_split=10, n_estim
ators=250;, score=0.782 total time=  20.0s
[CV 4/5] END max_depth=80, min_samples_leaf=10, min_samples_split=2, n_estima
tors=100;, score=0.784 total time=   9.2s
[CV 5/5] END max_depth=80, min_samples_leaf=10, min_samples_split=10, n_estim
ators=250;, score=0.772 total time=  21.4s
[CV 5/5] END max_depth=80, min_samples_leaf=10, min_samples_split=2, n_estima
tors=100;, score=0.773 total time=   9.8s
[CV 1/5] END max_depth=100, min_samples_leaf=1, min_samples_split=10, n_estim
ators=100;, score=0.812 total time=   9.7s
[CV 2/5] END max_depth=100, min_samples_leaf=1, min_samples_split=10, n_estim
ators=100;, score=0.798 total time=   9.1s
[CV 3/5] END max_depth=100, min_samples_leaf=1, min_samples_split=10, n_estim
ators=100;, score=0.787 total time=   8.8s
[CV 1/5] END max_depth=80, min_samples_leaf=10, min_samples_split=2, n_estima
tors=250;, score=0.789 total time=  20.0s
[CV 2/5] END max_depth=80, min_samples_leaf=10, min_samples_split=2, n_estima
tors=250;, score=0.775 total time=  20.8s
[CV 3/5] END max_depth=80, min_samples_leaf=10, min_samples_split=2, n_estima
tors=250;, score=0.764 total time=  19.7s
[CV 4/5] END max_depth=80, min_samples_leaf=10, min_samples_split=2, n_estima
tors=250;, score=0.782 total time=  19.5s
[CV 4/5] END max_depth=100, min_samples_leaf=1, min_samples_split=10, n_estim
ators=100;, score=0.805 total time=   8.2s
[CV 5/5] END max_depth=80, min_samples_leaf=10, min_samples_split=2, n_estima
```

```
tors=250;, score=0.769 total time=  18.6s
[CV 5/5] END max_depth=100, min_samples_leaf=1, min_samples_split=10, n_estim
ators=100;, score=0.795 total time=   7.8s
[CV 1/5] END max_depth=100, min_samples_leaf=1, min_samples_split=2, n_estima
tors=100;, score=0.827 total time=   7.2s
[CV 2/5] END max_depth=100, min_samples_leaf=1, min_samples_split=2, n_estima
tors=100;, score=0.809 total time=   7.1s
[CV 3/5] END max_depth=100, min_samples_leaf=1, min_samples_split=2, n_estima
tors=100;, score=0.808 total time=   7.1s
[CV 1/5] END max_depth=100, min_samples_leaf=1, min_samples_split=10, n_estim
ators=250;, score=0.813 total time=  17.8s
[CV 2/5] END max_depth=100, min_samples_leaf=1, min_samples_split=10, n_estim
ators=250;, score=0.800 total time=  17.5s
[CV 3/5] END max_depth=100, min_samples_leaf=1, min_samples_split=10, n_estim
ators=250;, score=0.785 total time=  17.9s
[CV 4/5] END max_depth=100, min_samples_leaf=1, min_samples_split=2, n_estima
tors=100;, score=0.819 total time=   7.9s
[CV 5/5] END max_depth=100, min_samples_leaf=1, min_samples_split=2, n_estima
tors=100;, score=0.806 total time=   8.1s
[CV 4/5] END max_depth=100, min_samples_leaf=1, min_samples_split=10, n_estim
ators=250;, score=0.803 total time=  18.7s
[CV 5/5] END max_depth=100, min_samples_leaf=1, min_samples_split=10, n_estim
ators=250;, score=0.798 total time=  18.1s
[CV 1/5] END max_depth=100, min_samples_leaf=10, min_samples_split=10, n_esti
mators=100;, score=0.786 total time=   7.1s
[CV 2/5] END max_depth=100, min_samples_leaf=10, min_samples_split=10, n_esti
mators=100;, score=0.767 total time=   7.9s
[CV 3/5] END max_depth=100, min_samples_leaf=10, min_samples_split=10, n_esti
mators=100;, score=0.768 total time=   7.5s
[CV 1/5] END max_depth=100, min_samples_leaf=1, min_samples_split=2, n_estima
tors=250;, score=0.827 total time=  22.0s
[CV 4/5] END max_depth=100, min_samples_leaf=10, min_samples_split=10, n_esti
mators=100;, score=0.786 total time=   9.2s
[CV 5/5] END max_depth=100, min_samples_leaf=10, min_samples_split=10, n_esti
mators=100;, score=0.772 total time=   8.4s
[CV 2/5] END max_depth=100, min_samples_leaf=1, min_samples_split=2, n_estima
tors=250;, score=0.812 total time=  22.6s
[CV 3/5] END max_depth=100, min_samples_leaf=1, min_samples_split=2, n_estima
tors=250;, score=0.809 total time=  22.3s
[CV 4/5] END max_depth=100, min_samples_leaf=1, min_samples_split=2, n_estima
tors=250;, score=0.822 total time=  23.2s
[CV 5/5] END max_depth=100, min_samples_leaf=1, min_samples_split=2, n_estima
tors=250;, score=0.812 total time=  22.2s
[CV 1/5] END max_depth=100, min_samples_leaf=10, min_samples_split=2, n_estim
ators=100;, score=0.789 total time=   6.5s
[CV 2/5] END max_depth=100, min_samples_leaf=10, min_samples_split=2, n_estim
ators=100;, score=0.767 total time=   7.4s
[CV 1/5] END max_depth=100, min_samples_leaf=10, min_samples_split=10, n_esti
mators=250;, score=0.789 total time=  19.3s
[CV 3/5] END max_depth=100, min_samples_leaf=10, min_samples_split=2, n_estim
ators=100;, score=0.765 total time=   8.7s
[CV 2/5] END max_depth=100, min_samples_leaf=10, min_samples_split=10, n_esti
mators=250;, score=0.768 total time=  27.7s
[CV 4/5] END max_depth=100, min_samples_leaf=10, min_samples_split=2, n_estim
ators=100;, score=0.781 total time=  17.2s
[CV 3/5] END max_depth=100, min_samples_leaf=10, min_samples_split=10, n_esti
```

```
mators=250;, score=0.766 total time=  28.8s
[CV 4/5] END max_depth=100, min_samples_leaf=10, min_samples_split=10, n_esti
mators=250;, score=0.780 total time=  29.3s
[CV 5/5] END max_depth=100, min_samples_leaf=10, min_samples_split=2, n_estim
ators=100;, score=0.769 total time=  18.1s
[CV 5/5] END max_depth=100, min_samples_leaf=10, min_samples_split=10, n_esti
mators=250;, score=0.772 total time=  28.8s
[CV 1/5] END max_depth=100, min_samples_leaf=10, min_samples_split=2, n_estim
ators=250;, score=0.789 total time=  25.0s
[CV 2/5] END max_depth=100, min_samples_leaf=10, min_samples_split=2, n_estim
ators=250;, score=0.771 total time=  24.2s
[CV 3/5] END max_depth=100, min_samples_leaf=10, min_samples_split=2, n_estim
ators=250;, score=0.765 total time=  14.7s
[CV 4/5] END max_depth=100, min_samples_leaf=10, min_samples_split=2, n_estim
ators=250;, score=0.784 total time=  15.7s
[CV 5/5] END max_depth=100, min_samples_leaf=10, min_samples_split=2, n_estim
ators=250;, score=0.772 total time=  13.6s
```

Out[101]:



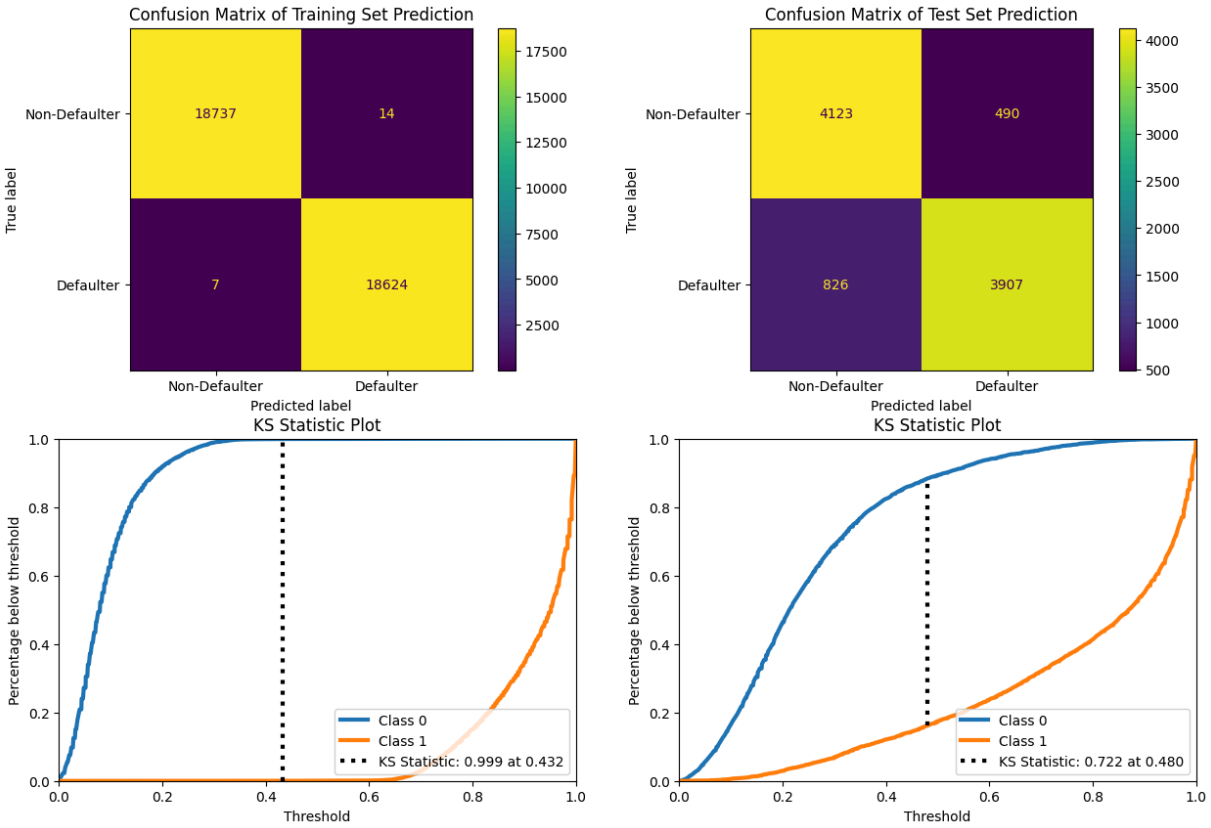
In [103…

```
#Best parameters
print('Best parameters:', rf_cv.best_params_)
rf_best=rf_cv.best_estimator_

#predicting the probabilities of test data
rf_best_test_proba=rf_best.predict_proba(X_test_scaled)
rf_best_train_proba=rf_best.predict_proba(X_train_scaled)

 #predicting the values of y from x via model
y_best_pred_test = rf_best.predict(X_test_scaled)
y_best_pred_train = rf_best.predict(X_train_scaled)
```

```
Best parameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_spli
t': 2, 'n_estimators': 250}
```

In [104…

```
#visualize the model performence
model_vis(y_train,y_test,y_best_pred_train,y_best_pred_test,rf_best_train_pr
```



In [105…
```
#view all the metric score in a dataframe
metric_score('Tuned Random_Forrest_Classifier',y_test,y_best_pred_test,rf_be
```

Out[105]:

	Model	Accuracy	Precision	Recall	F1 Score	KS_Stat
0	Logistic_Regression	0.765033	0.811901	0.697655	0.750455	0.536
1	Tuned Logistic_Regression	0.744597	0.771026	0.705050	0.736563	0.498
2	Decision_Tree_Classifier	0.797988	0.843682	0.737798	0.787196	0.598
3	Tuned Decision_Tree_Classifier	0.793067	0.826300	0.748785	0.785635	0.591
4	Random_Forrest_Classifier	0.856837	0.886762	0.822311	0.853322	0.718
5	Tuned Random_Forrest_Classifier	0.859191	0.888560	0.825481	0.855860	0.722

Which hyperparameter optimization technique have you used and why?

I used GridSearch CV to discover the best parameters for the model in order to improve its accuracy. GridSearchCV is the process of tweaking hyperparameters to determine the best values for a particular model.The value of hyperparameters has a substantial impact on model performance.

GridSearchCV examines the model for each combination of the values supplied in the dictionary using the Cross-Validation technique. As a result of utilising this function, we

can calculate the accuracy/loss for each combination of hyperparameters and select the one with the best performance.

Have you seen any improvement? Note down the improvement with updates Evaluation metric Score Chart.

Performance of model doesn't improved after cross validation so we can say that our base model was performing good with Recall of 84%.

ML Model - 4 **K - Nearest Neighbors (KNN)**

In [106…
```
# ML Model - 3 Implementation
model_knn = KNeighborsClassifier(n_neighbors=7)
# Fit the Algorithm
model_knn.fit(X_train_scaled,y_train)
```

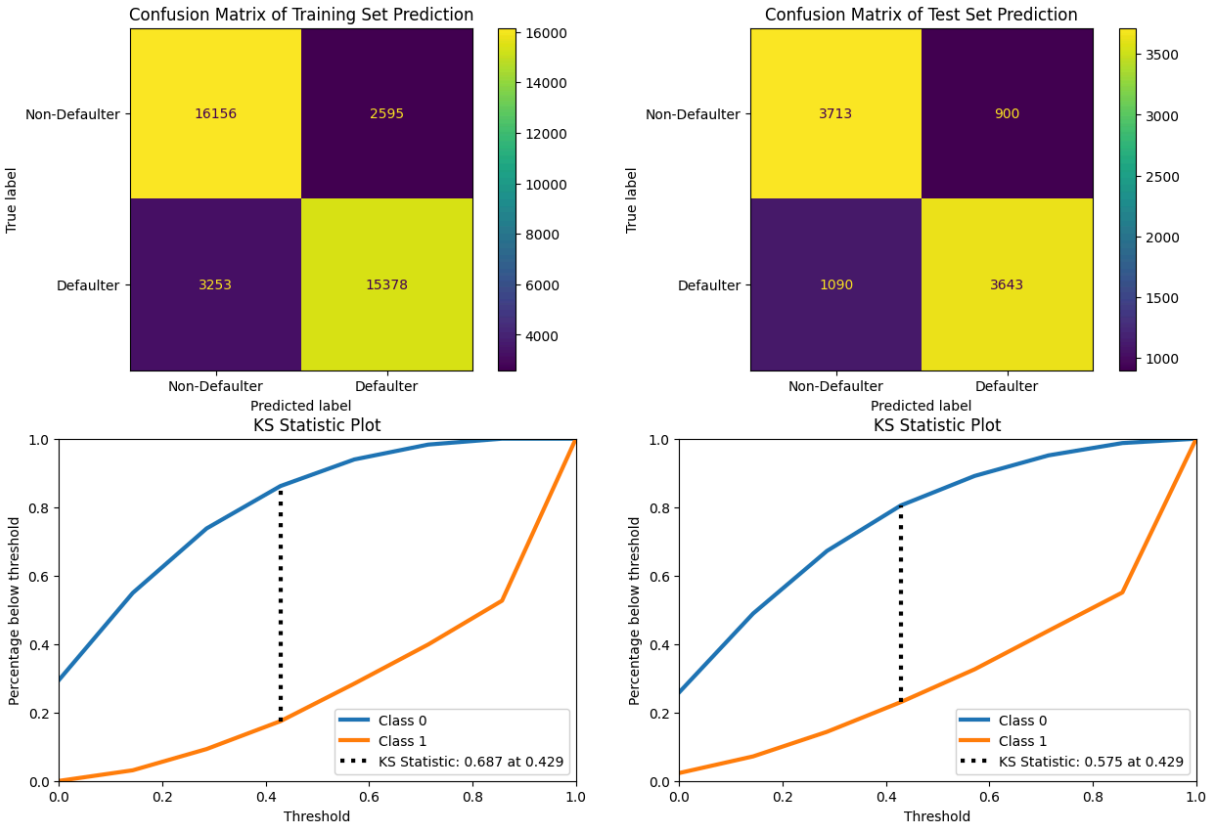
Out[106]:
```
KNeighborsClassifier(n_neighbors=7)
```

In [107…
```
#predicting the probabilities of test data
knn_test_proba=model_knn.predict_proba(X_test_scaled)
knn_train_proba=model_knn.predict_proba(X_train_scaled)
 #predicting the values of y from x via model
y_pred_test = model_knn.predict(X_test_scaled)
y_pred_train = model_knn.predict(X_train_scaled)
```

1. Explain the ML Model used and it's performance using Evaluation metric Score Chart.

The k-nearest neighbours algorithm, often known as KNN or k-NN, is a non-parametric, supervised learning classifier that employs proximity to classify or predict the grouping of a single data point. While it can be used for either regression or classification issues, it is most commonly utilised as a classification algorithm, based on the idea that similar points can be discovered nearby.

In [108…
```
#visualize the model performence
model_vis(y_train,y_test,y_pred_train,y_pred_test,knn_train_proba,knn_test_p
```



In [109…
```
#view all the metric score in a dataframe
metric_score('KNN Classifier',y_test,y_pred_test,knn_test_proba)
```

Out[109]:

	Model	Accuracy	Precision	Recall	F1 Score	KS_Stat
0	Logistic_Regression	0.765033	0.811901	0.697655	0.750455	0.536
1	Tuned Logistic_Regression	0.744597	0.771026	0.705050	0.736563	0.498
2	Decision_Tree_Classifier	0.797988	0.843682	0.737798	0.787196	0.598
3	Tuned Decision_Tree_Classifier	0.793067	0.826300	0.748785	0.785635	0.591
4	Random_Forrest_Classifier	0.856837	0.886762	0.822311	0.853322	0.718
5	Tuned Random_Forrest_Classifier	0.859191	0.888560	0.825481	0.855860	0.722
6	KNN Classifier	0.787075	0.801893	0.769702	0.785468	0.575

2. Cross- Validation & Hyperparameter Tuning

In [110…
```
# ML Model - 1 Implementation with hyperparameter optimization techniques (i
knn=KNeighborsClassifier()
param_grid = {
    'n_neighbors': [3,5,7,9,11],
    'weights': ['uniform', 'distance'],
    'p': [1, 2]
}
knn_cv=GridSearchCV(knn, param_grid, scoring = 'recall',n_jobs = -1, verbose
```

```
# Fit the Algorithm
knn_cv.fit(X_train_scaled,y_train)
```

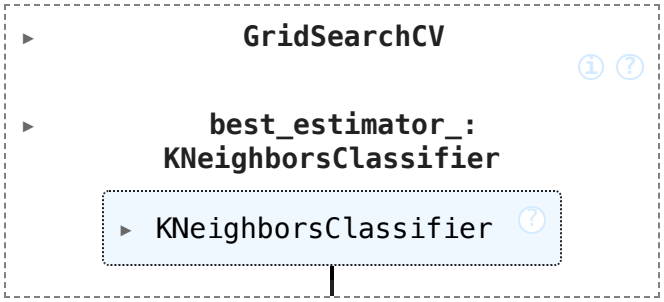
```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
[CV 1/5] END n_neighbors=3, p=1, weights=uniform;, score=0.815 total time=
4.4s
[CV 2/5] END n_neighbors=3, p=1, weights=uniform;, score=0.799 total time=
4.4s
[CV 4/5] END n_neighbors=3, p=1, weights=uniform;, score=0.823 total time=
4.5s
[CV 3/5] END n_neighbors=3, p=1, weights=uniform;, score=0.794 total time=
4.5s
[CV 1/5] END n_neighbors=3, p=1, weights=distance;, score=0.833 total time=
4.5s
[CV 3/5] END n_neighbors=3, p=1, weights=distance;, score=0.811 total time=
4.4s
[CV 5/5] END n_neighbors=3, p=1, weights=uniform;, score=0.802 total time=
4.6s
[CV 2/5] END n_neighbors=3, p=1, weights=distance;, score=0.822 total time=
4.5s
[CV 1/5] END n_neighbors=3, p=2, weights=uniform;, score=0.790 total time=
1.0s
[CV 4/5] END n_neighbors=3, p=2, weights=uniform;, score=0.792 total time=
0.8s
[CV 2/5] END n_neighbors=3, p=2, weights=uniform;, score=0.778 total time=
1.0s
[CV 1/5] END n_neighbors=3, p=2, weights=distance;, score=0.806 total time=
0.9s
[CV 3/5] END n_neighbors=3, p=2, weights=uniform;, score=0.763 total time=
0.9s
[CV 5/5] END n_neighbors=3, p=2, weights=uniform;, score=0.785 total time=
1.0s
[CV 3/5] END n_neighbors=3, p=2, weights=distance;, score=0.782 total time=
0.9s
[CV 2/5] END n_neighbors=3, p=2, weights=distance;, score=0.799 total time=
1.0s
[CV 4/5] END n_neighbors=3, p=2, weights=distance;, score=0.813 total time=
1.0s
[CV 5/5] END n_neighbors=3, p=2, weights=distance;, score=0.802 total time=
1.1s
[CV 4/5] END n_neighbors=3, p=1, weights=distance;, score=0.841 total time=
4.4s
[CV 5/5] END n_neighbors=3, p=1, weights=distance;, score=0.824 total time=
4.3s
[CV 1/5] END n_neighbors=5, p=1, weights=uniform;, score=0.797 total time=
4.4s
[CV 2/5] END n_neighbors=5, p=1, weights=uniform;, score=0.790 total time=
4.6s
[CV 4/5] END n_neighbors=5, p=1, weights=uniform;, score=0.804 total time=
4.2s
[CV 3/5] END n_neighbors=5, p=1, weights=uniform;, score=0.772 total time=
4.4s
[CV 5/5] END n_neighbors=5, p=1, weights=uniform;, score=0.792 total time=
4.5s
[CV 1/5] END n_neighbors=5, p=1, weights=distance;, score=0.829 total time=
4.6s
[CV 1/5] END n_neighbors=5, p=2, weights=uniform;, score=0.778 total time=
0.9s
[CV 2/5] END n_neighbors=5, p=2, weights=uniform;, score=0.764 total time=
```

```
0.9s
[CV 3/5] END n_neighbors=5, p=2, weights=uniform;, score=0.756 total time=
1.0s
[CV 4/5] END n_neighbors=5, p=2, weights=uniform;, score=0.774 total time=
1.0s
[CV 5/5] END n_neighbors=5, p=2, weights=uniform;, score=0.769 total time=
0.8s
[CV 1/5] END n_neighbors=5, p=2, weights=distance;, score=0.804 total time=
0.9s
[CV 2/5] END n_neighbors=5, p=2, weights=distance;, score=0.793 total time=
0.8s
[CV 3/5] END n_neighbors=5, p=1, weights=distance;, score=0.804 total time=
4.3s
[CV 3/5] END n_neighbors=5, p=2, weights=distance;, score=0.779 total time=
0.9s
[CV 2/5] END n_neighbors=5, p=1, weights=distance;, score=0.819 total time=
4.5s
[CV 4/5] END n_neighbors=5, p=2, weights=distance;, score=0.804 total time=
0.9s
[CV 5/5] END n_neighbors=5, p=2, weights=distance;, score=0.795 total time=
0.8s
[CV 4/5] END n_neighbors=5, p=1, weights=distance;, score=0.835 total time=
4.2s
[CV 5/5] END n_neighbors=5, p=1, weights=distance;, score=0.820 total time=
4.1s
[CV 2/5] END n_neighbors=7, p=1, weights=uniform;, score=0.780 total time=
3.9s
[CV 3/5] END n_neighbors=7, p=1, weights=uniform;, score=0.760 total time=
3.9s
[CV 1/5] END n_neighbors=7, p=1, weights=uniform;, score=0.788 total time=
4.1s
[CV 4/5] END n_neighbors=7, p=1, weights=uniform;, score=0.797 total time=
3.9s
[CV 5/5] END n_neighbors=7, p=1, weights=uniform;, score=0.769 total time=
3.9s
[CV 1/5] END n_neighbors=7, p=1, weights=distance;, score=0.822 total time=
3.9s
[CV 2/5] END n_neighbors=7, p=2, weights=uniform;, score=0.757 total time=
0.8s
[CV 1/5] END n_neighbors=7, p=2, weights=uniform;, score=0.767 total time=
1.0s
[CV 2/5] END n_neighbors=7, p=1, weights=distance;, score=0.814 total time=
3.9s
[CV 3/5] END n_neighbors=7, p=2, weights=uniform;, score=0.745 total time=
0.9s
[CV 4/5] END n_neighbors=7, p=2, weights=uniform;, score=0.771 total time=
0.9s
[CV 3/5] END n_neighbors=7, p=1, weights=distance;, score=0.798 total time=
4.0s
[CV 1/5] END n_neighbors=7, p=2, weights=distance;, score=0.798 total time=
0.9s
[CV 2/5] END n_neighbors=7, p=2, weights=distance;, score=0.789 total time=
0.9s
[CV 5/5] END n_neighbors=7, p=2, weights=uniform;, score=0.756 total time=
1.0s
[CV 3/5] END n_neighbors=7, p=2, weights=distance;, score=0.777 total time=
```

```
0.9s
[CV 5/5] END n_neighbors=7, p=2, weights=distance;, score=0.787 total time=
1.0s
[CV 4/5] END n_neighbors=7, p=2, weights=distance;, score=0.801 total time=
1.0s
[CV 5/5] END n_neighbors=7, p=1, weights=distance;, score=0.806 total time=
4.0s
[CV 4/5] END n_neighbors=7, p=1, weights=distance;, score=0.831 total time=
4.1s
[CV 1/5] END n_neighbors=9, p=1, weights=uniform;, score=0.774 total time=
7.7s
[CV 3/5] END n_neighbors=9, p=1, weights=uniform;, score=0.751 total time=
7.9s
[CV 1/5] END n_neighbors=9, p=1, weights=distance;, score=0.812 total time=
7.7s
[CV 4/5] END n_neighbors=9, p=1, weights=uniform;, score=0.780 total time=
8.0s
[CV 2/5] END n_neighbors=9, p=1, weights=uniform;, score=0.770 total time=
8.4s
[CV 5/5] END n_neighbors=9, p=1, weights=uniform;, score=0.761 total time=
8.3s
[CV 1/5] END n_neighbors=9, p=2, weights=uniform;, score=0.759 total time=
1.1s
[CV 2/5] END n_neighbors=9, p=2, weights=uniform;, score=0.749 total time=
1.0s
[CV 3/5] END n_neighbors=9, p=2, weights=uniform;, score=0.741 total time=
1.2s
[CV 4/5] END n_neighbors=9, p=2, weights=uniform;, score=0.763 total time=
0.9s
[CV 5/5] END n_neighbors=9, p=2, weights=uniform;, score=0.743 total time=
1.1s
[CV 1/5] END n_neighbors=9, p=2, weights=distance;, score=0.788 total time=
1.1s
[CV 2/5] END n_neighbors=9, p=1, weights=distance;, score=0.808 total time=
8.5s
[CV 3/5] END n_neighbors=9, p=1, weights=distance;, score=0.792 total time=
8.4s
[CV 2/5] END n_neighbors=9, p=2, weights=distance;, score=0.787 total time=
1.1s
[CV 3/5] END n_neighbors=9, p=2, weights=distance;, score=0.775 total time=
1.1s
[CV 4/5] END n_neighbors=9, p=2, weights=distance;, score=0.792 total time=
1.1s
[CV 5/5] END n_neighbors=9, p=2, weights=distance;, score=0.782 total time=
1.1s
[CV 4/5] END n_neighbors=9, p=1, weights=distance;, score=0.819 total time=
5.3s
[CV 5/5] END n_neighbors=9, p=1, weights=distance;, score=0.805 total time=
5.7s
[CV 4/5] END n_neighbors=11, p=1, weights=uniform;, score=0.777 total time=
5.4s
[CV 1/5] END n_neighbors=11, p=1, weights=uniform;, score=0.765 total time=
5.7s
[CV 2/5] END n_neighbors=11, p=1, weights=uniform;, score=0.760 total time=
5.7s
[CV 3/5] END n_neighbors=11, p=1, weights=uniform;, score=0.747 total time=
```

```
5.6s
[CV 1/5] END n_neighbors=11, p=1, weights=distance;, score=0.807 total time=
5.3s
[CV 5/5] END n_neighbors=11, p=1, weights=uniform;, score=0.758 total time=
5.4s
[CV 1/5] END n_neighbors=11, p=2, weights=uniform;, score=0.749 total time=
0.9s
[CV 2/5] END n_neighbors=11, p=2, weights=uniform;, score=0.745 total time=
0.9s
[CV 2/5] END n_neighbors=11, p=1, weights=distance;, score=0.804 total time=
4.6s
[CV 3/5] END n_neighbors=11, p=2, weights=uniform;, score=0.731 total time=
0.8s
[CV 4/5] END n_neighbors=11, p=2, weights=uniform;, score=0.756 total time=
1.0s
[CV 3/5] END n_neighbors=11, p=1, weights=distance;, score=0.791 total time=
4.4s
[CV 1/5] END n_neighbors=11, p=2, weights=distance;, score=0.782 total time=
0.9s
[CV 5/5] END n_neighbors=11, p=2, weights=uniform;, score=0.738 total time=
0.9s
[CV 3/5] END n_neighbors=11, p=2, weights=distance;, score=0.768 total time=
0.8s
[CV 2/5] END n_neighbors=11, p=2, weights=distance;, score=0.779 total time=
0.8s
[CV 4/5] END n_neighbors=11, p=2, weights=distance;, score=0.788 total time=
0.8s
[CV 5/5] END n_neighbors=11, p=2, weights=distance;, score=0.779 total time=
0.7s
[CV 5/5] END n_neighbors=11, p=1, weights=distance;, score=0.803 total time=
3.6s
[CV 4/5] END n_neighbors=11, p=1, weights=distance;, score=0.816 total time=
3.7s
```

Out[110]:

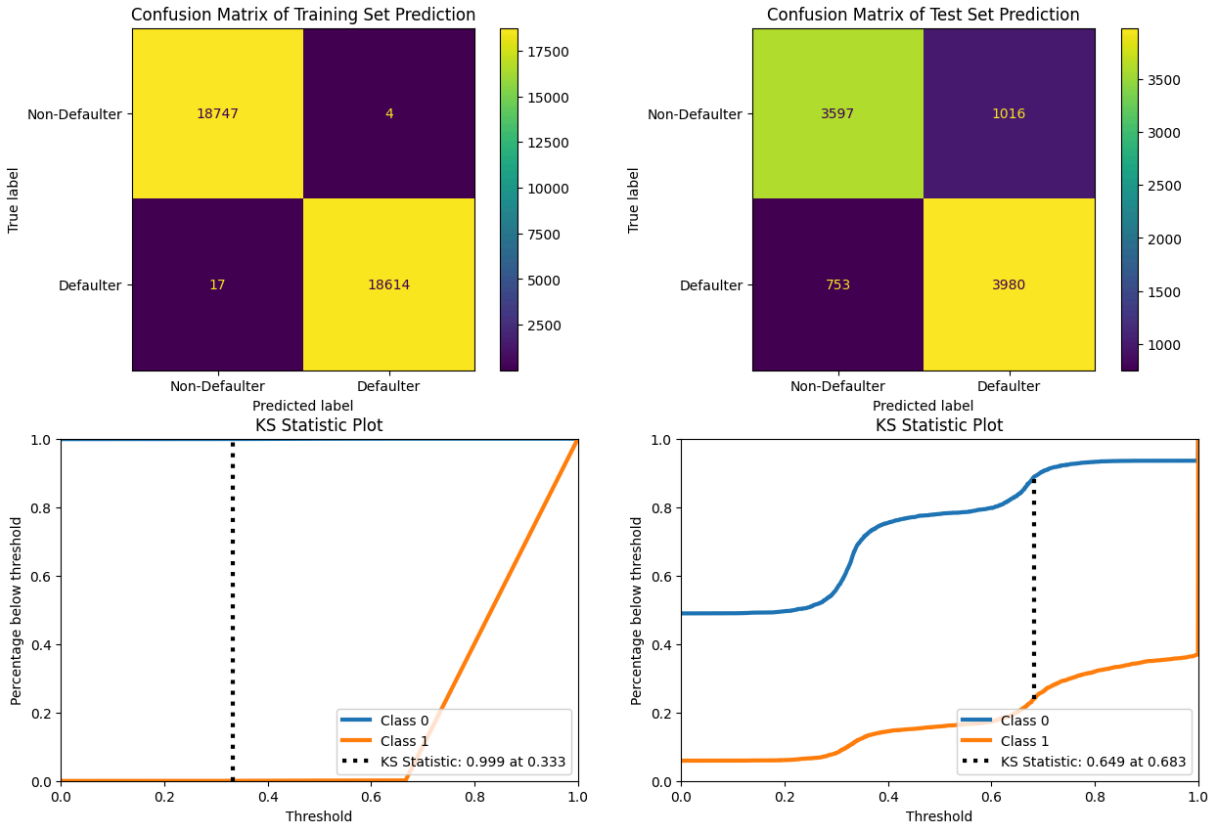


In [111…

```python
#Best parameters
print('Best parameters:', knn_cv.best_params_)
knn_best=knn_cv.best_estimator_

#predicting the probabilities of test data
knn_best_test_proba=knn_best.predict_proba(X_test_scaled)
knn_best_train_proba=knn_best.predict_proba(X_train_scaled)

 #predicting the values of y from x via model
y_best_pred_test = knn_best.predict(X_test_scaled)
y_best_pred_train = knn_best.predict(X_train_scaled)
```

```
Best parameters: {'n_neighbors': 3, 'p': 1, 'weights': 'distance'}
```

In [112…
```
#visualize the model performence
model_vis(y_train,y_test,y_best_pred_train,y_best_pred_test,knn_best_train_p
```



In [ ]:
```
#view all the metric score in a dataframe
metric_score('Tuned KNN_Classifier',y_test,y_best_pred_test,knn_best_test_pr
```

Which hyperparameter optimization technique have you used and why?

I used GridSearch CV to discover the best parameters for the model in order to improve its accuracy. GridSearchCV is the process of tweaking hyperparameters to determine the best values for a particular model.The value of hyperparameters has a substantial impact on model performance.

GridSearchCV examines the model for each combination of the values supplied in the dictionary using the Cross-Validation technique. As a result of utilising this function, we can calculate the accuracy/loss for each combination of hyperparameters and select the one with the best performance.

Have you seen any improvement? Note down the improvement with updates Evaluation metric Score Chart.

Yes performance of our model has been improved with improvmenet in all evaluation matrics.After cross validation recall becomes around 85%.

ML Model - 5 **AdaBoost Classifier**

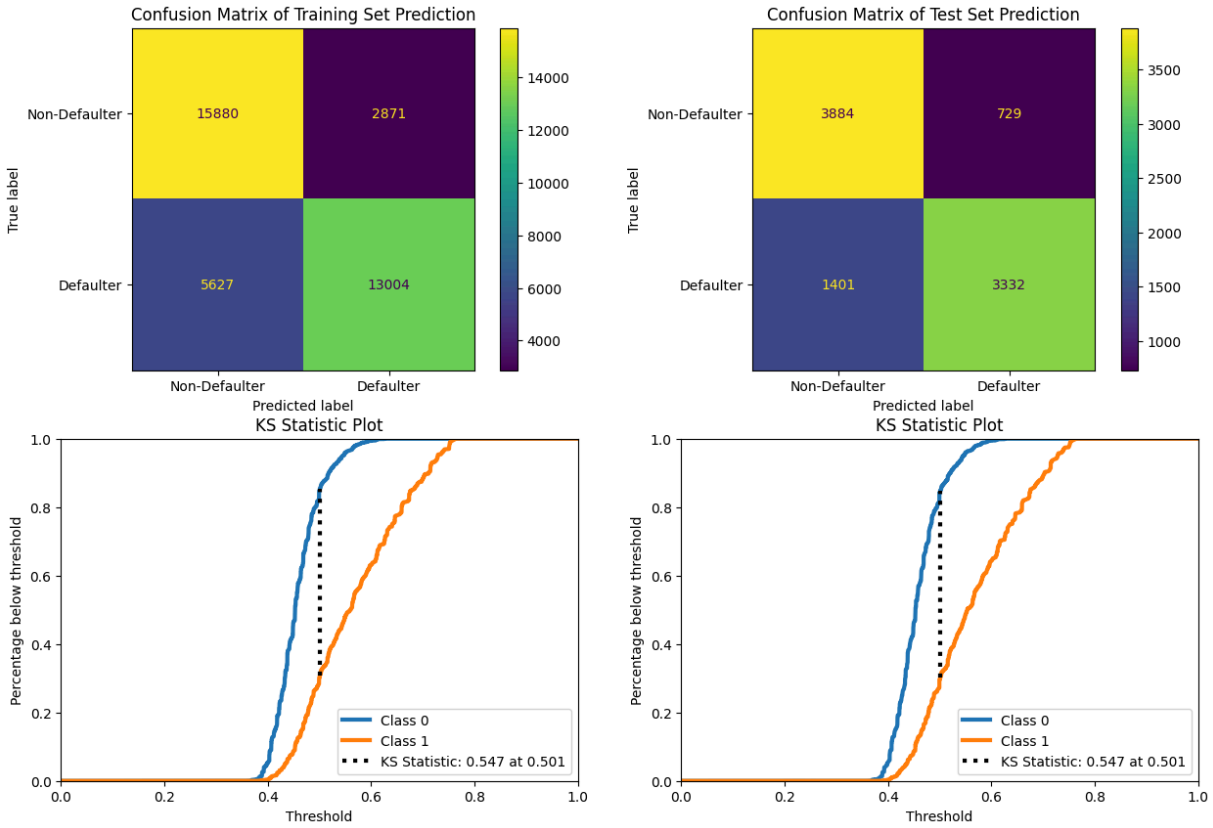
In [113…
```
# ML Model - 3 Implementation
model_ada = AdaBoostClassifier(n_estimators=100)
# Fit the Algorithm
model_ada.fit(X_train_scaled,y_train)
```

Out[113]:
```
AdaBoostClassifier(n_estimators=100)
```

In [114…
```
#predicting the probabilities of test data
ada_test_proba=model_ada.predict_proba(X_test_scaled)
ada_train_proba=model_ada.predict_proba(X_train_scaled)
 #predicting the values of y from x via model
y_pred_test = model_ada.predict(X_test_scaled)
y_pred_train = model_ada.predict(X_train_scaled)
```

1. Explain the ML Model used and it's performance using Evaluation metric Score Chart.

In [115…
```
#visualize the model performence
model_vis(y_train,y_test,y_pred_train,y_pred_test,ada_train_proba,ada_test_p
```



In [116…
```
#view all the metric score in a dataframe
metric_score('AdaBoost Classifier',y_test,y_pred_test,ada_test_proba)
```

Out[116]:

	Model	Accuracy	Precision	Recall	F1 Score	KS_Stat
0	Logistic_Regression	0.765033	0.811901	0.697655	0.750455	0.536
1	Tuned Logistic_Regression	0.744597	0.771026	0.705050	0.736563	0.498
2	Decision_Tree_Classifier	0.797988	0.843682	0.737798	0.787196	0.598
3	Tuned Decision_Tree_Classifier	0.793067	0.826300	0.748785	0.785635	0.591
4	Random_Forrest_Classifier	0.856837	0.886762	0.822311	0.853322	0.718
5	Tuned Random_Forrest_Classifier	0.859191	0.888560	0.825481	0.855860	0.722
6	KNN Classifier	0.787075	0.801893	0.769702	0.785468	0.575
7	AdaBoost Classifier	0.772095	0.820488	0.703993	0.757789	0.547

2. Cross- Validation & Hyperparameter Tuning

In [117…

```
# ML Model - 1 Implementation with hyperparameter optimization techniques (i
ada=AdaBoostClassifier()
param_grid = {
    'n_estimators' : [50, 90, 120, 180, 200],
    'learning_rate' : [0.001, 0.01, 0.1, 1, 10]
}
ada_cv=GridSearchCV(ada, param_grid, scoring = 'recall',n_jobs = -1, verbose
# Fit the Algorithm
ada_cv.fit(X_train_scaled,y_train)
```

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
[CV 4/5] END learning_rate=0.001, n_estimators=50;, score=0.463 total time=
3.2s
[CV 5/5] END learning_rate=0.001, n_estimators=50;, score=0.466 total time=
3.1s
[CV 2/5] END learning_rate=0.001, n_estimators=50;, score=0.461 total time=
3.2s
[CV 1/5] END learning_rate=0.001, n_estimators=50;, score=0.467 total time=
3.2s
[CV 3/5] END learning_rate=0.001, n_estimators=50;, score=0.456 total time=
3.5s
[CV 2/5] END learning_rate=0.001, n_estimators=90;, score=0.461 total time=
5.5s
[CV 1/5] END learning_rate=0.001, n_estimators=90;, score=0.467 total time=
5.9s
[CV 3/5] END learning_rate=0.001, n_estimators=90;, score=0.456 total time=
6.0s
[CV 4/5] END learning_rate=0.001, n_estimators=90;, score=0.463 total time=
5.7s
[CV 5/5] END learning_rate=0.001, n_estimators=90;, score=0.466 total time=
5.7s
[CV 1/5] END learning_rate=0.001, n_estimators=120;, score=0.467 total time=
7.2s
[CV 2/5] END learning_rate=0.001, n_estimators=120;, score=0.461 total time=
7.7s
[CV 3/5] END learning_rate=0.001, n_estimators=120;, score=0.456 total time=
7.5s
[CV 4/5] END learning_rate=0.001, n_estimators=120;, score=0.463 total time=
7.7s
[CV 5/5] END learning_rate=0.001, n_estimators=120;, score=0.466 total time=
7.5s
[CV 1/5] END learning_rate=0.001, n_estimators=180;, score=0.467 total time=
11.1s
[CV 3/5] END learning_rate=0.001, n_estimators=180;, score=0.456 total time=
11.1s
[CV 2/5] END learning_rate=0.001, n_estimators=180;, score=0.461 total time=
11.4s
[CV 4/5] END learning_rate=0.001, n_estimators=180;, score=0.463 total time=
10.8s
[CV 5/5] END learning_rate=0.001, n_estimators=180;, score=0.466 total time=
11.1s
[CV 1/5] END learning_rate=0.001, n_estimators=200;, score=0.467 total time=
11.9s
[CV 1/5] END learning_rate=0.01, n_estimators=50;, score=0.467 total time=
2.7s
[CV 2/5] END learning_rate=0.01, n_estimators=50;, score=0.461 total time=
2.6s
[CV 3/5] END learning_rate=0.01, n_estimators=50;, score=0.456 total time=
3.2s
[CV 2/5] END learning_rate=0.001, n_estimators=200;, score=0.461 total time=
12.3s
[CV 3/5] END learning_rate=0.001, n_estimators=200;, score=0.456 total time=
12.6s
[CV 4/5] END learning_rate=0.01, n_estimators=50;, score=0.463 total time=
3.1s
[CV 5/5] END learning_rate=0.01, n_estimators=50;, score=0.466 total time=
```

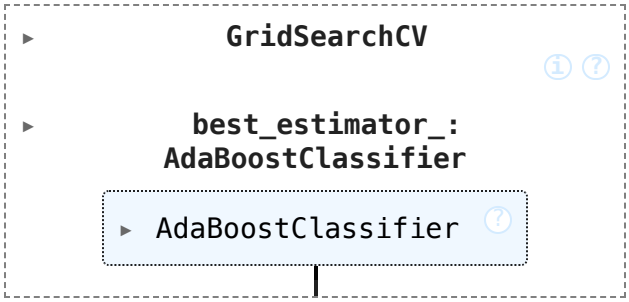
```
3.3s
[CV 4/5] END learning_rate=0.001, n_estimators=200;, score=0.463 total time=
11.9s
[CV 1/5] END learning_rate=0.01, n_estimators=90;, score=0.467 total time=
5.6s
[CV 3/5] END learning_rate=0.01, n_estimators=90;, score=0.456 total time=
5.2s
[CV 2/5] END learning_rate=0.01, n_estimators=90;, score=0.461 total time=
5.8s
[CV 4/5] END learning_rate=0.01, n_estimators=90;, score=0.463 total time=
5.3s
[CV 5/5] END learning_rate=0.01, n_estimators=90;, score=0.363 total time=
5.2s
[CV 5/5] END learning_rate=0.001, n_estimators=200;, score=0.466 total time=
12.2s
[CV 1/5] END learning_rate=0.01, n_estimators=120;, score=0.493 total time=
7.0s
[CV 2/5] END learning_rate=0.01, n_estimators=120;, score=0.506 total time=
6.8s
[CV 3/5] END learning_rate=0.01, n_estimators=120;, score=0.422 total time=
6.8s
[CV 4/5] END learning_rate=0.01, n_estimators=120;, score=0.502 total time=
6.9s
[CV 5/5] END learning_rate=0.01, n_estimators=120;, score=0.462 total time=
6.7s
[CV 2/5] END learning_rate=0.01, n_estimators=180;, score=0.507 total time=
9.8s
[CV 1/5] END learning_rate=0.01, n_estimators=180;, score=0.465 total time=
10.0s
[CV 3/5] END learning_rate=0.01, n_estimators=180;, score=0.462 total time=
9.8s
[CV 4/5] END learning_rate=0.01, n_estimators=180;, score=0.463 total time=
10.5s
[CV 1/5] END learning_rate=0.1, n_estimators=50;, score=0.520 total time=
2.7s
[CV 5/5] END learning_rate=0.01, n_estimators=180;, score=0.470 total time=
10.0s
[CV 2/5] END learning_rate=0.1, n_estimators=50;, score=0.498 total time=
3.6s
[CV 1/5] END learning_rate=0.01, n_estimators=200;, score=0.514 total time=
12.4s
[CV 3/5] END learning_rate=0.1, n_estimators=50;, score=0.495 total time=
3.9s
[CV 2/5] END learning_rate=0.01, n_estimators=200;, score=0.487 total time=
13.1s
[CV 4/5] END learning_rate=0.1, n_estimators=50;, score=0.504 total time=
5.1s
[CV 3/5] END learning_rate=0.01, n_estimators=200;, score=0.462 total time=
13.8s
[CV 5/5] END learning_rate=0.1, n_estimators=50;, score=0.506 total time=
4.2s
[CV 4/5] END learning_rate=0.01, n_estimators=200;, score=0.511 total time=
14.0s
[CV 5/5] END learning_rate=0.01, n_estimators=200;, score=0.471 total time=
13.8s
[CV 2/5] END learning_rate=0.1, n_estimators=90;, score=0.639 total time=
```

```
6.5s
[CV 1/5] END learning_rate=0.1, n_estimators=90;, score=0.643 total time=
6.7s
[CV 3/5] END learning_rate=0.1, n_estimators=90;, score=0.629 total time=
5.6s
[CV 4/5] END learning_rate=0.1, n_estimators=90;, score=0.640 total time=
5.7s
[CV 5/5] END learning_rate=0.1, n_estimators=90;, score=0.646 total time=
5.8s
[CV 1/5] END learning_rate=0.1, n_estimators=120;, score=0.662 total time=
7.5s
[CV 3/5] END learning_rate=0.1, n_estimators=120;, score=0.654 total time=
7.3s
[CV 2/5] END learning_rate=0.1, n_estimators=120;, score=0.659 total time=
7.4s
[CV 4/5] END learning_rate=0.1, n_estimators=120;, score=0.666 total time=
7.3s
[CV 5/5] END learning_rate=0.1, n_estimators=120;, score=0.669 total time=
8.0s
[CV 2/5] END learning_rate=0.1, n_estimators=180;, score=0.668 total time=  1
1.7s
[CV 1/5] END learning_rate=0.1, n_estimators=180;, score=0.669 total time=  1
2.4s
[CV 3/5] END learning_rate=0.1, n_estimators=180;, score=0.662 total time=  1
1.7s
[CV 4/5] END learning_rate=0.1, n_estimators=180;, score=0.678 total time=  1
1.8s
[CV 1/5] END ..learning_rate=1, n_estimators=50;, score=0.698 total time=
3.0s
[CV 5/5] END learning_rate=0.1, n_estimators=180;, score=0.678 total time=  1
1.6s
[CV 2/5] END ..learning_rate=1, n_estimators=50;, score=0.677 total time=
3.0s
[CV 1/5] END learning_rate=0.1, n_estimators=200;, score=0.668 total time=  1
2.4s
[CV 2/5] END learning_rate=0.1, n_estimators=200;, score=0.667 total time=  1
2.4s
[CV 3/5] END ..learning_rate=1, n_estimators=50;, score=0.682 total time=
3.1s
[CV 3/5] END learning_rate=0.1, n_estimators=200;, score=0.660 total time=  1
2.9s
[CV 4/5] END ..learning_rate=1, n_estimators=50;, score=0.705 total time=
3.9s
[CV 5/5] END ..learning_rate=1, n_estimators=50;, score=0.721 total time=
4.1s
[CV 2/5] END ..learning_rate=1, n_estimators=90;, score=0.699 total time=
6.8s
[CV 4/5] END learning_rate=0.1, n_estimators=200;, score=0.679 total time=  1
3.3s
[CV 1/5] END ..learning_rate=1, n_estimators=90;, score=0.721 total time=
7.1s
[CV 3/5] END ..learning_rate=1, n_estimators=90;, score=0.710 total time=
7.0s
[CV 5/5] END learning_rate=0.1, n_estimators=200;, score=0.680 total time=  1
3.6s
[CV 4/5] END ..learning_rate=1, n_estimators=90;, score=0.705 total time=
```

```
6.7s
[CV 5/5] END ..learning_rate=1, n_estimators=90;, score=0.710 total time=
5.5s
[CV 1/5] END .learning_rate=1, n_estimators=120;, score=0.702 total time=
7.6s
[CV 2/5] END .learning_rate=1, n_estimators=120;, score=0.688 total time=
7.7s
[CV 3/5] END .learning_rate=1, n_estimators=120;, score=0.689 total time=
7.7s
[CV 5/5] END .learning_rate=1, n_estimators=120;, score=0.711 total time=
7.5s
[CV 4/5] END .learning_rate=1, n_estimators=120;, score=0.705 total time=
8.0s
[CV 1/5] END .learning_rate=1, n_estimators=180;, score=0.715 total time=  1
1.5s
[CV 2/5] END .learning_rate=1, n_estimators=180;, score=0.709 total time=  1
1.3s
[CV 3/5] END .learning_rate=1, n_estimators=180;, score=0.708 total time=  1
1.0s
[CV 4/5] END .learning_rate=1, n_estimators=180;, score=0.725 total time=  1
0.7s
[CV 1/5] END .learning_rate=10, n_estimators=50;, score=0.467 total time=
3.0s
[CV 2/5] END .learning_rate=10, n_estimators=50;, score=0.461 total time=
2.8s
[CV 5/5] END .learning_rate=1, n_estimators=180;, score=0.712 total time=  1
0.4s
[CV 3/5] END .learning_rate=10, n_estimators=50;, score=0.456 total time=
2.9s
[CV 1/5] END .learning_rate=1, n_estimators=200;, score=0.722 total time=  1
1.3s
[CV 2/5] END .learning_rate=1, n_estimators=200;, score=0.716 total time=  1
1.9s
[CV 3/5] END .learning_rate=1, n_estimators=200;, score=0.701 total time=  1
1.7s
[CV 4/5] END .learning_rate=10, n_estimators=50;, score=0.463 total time=
2.7s
[CV 5/5] END .learning_rate=10, n_estimators=50;, score=0.466 total time=
2.8s
[CV 1/5] END .learning_rate=10, n_estimators=90;, score=0.467 total time=
4.6s
[CV 4/5] END .learning_rate=1, n_estimators=200;, score=0.734 total time=  1
1.2s
[CV 5/5] END .learning_rate=1, n_estimators=200;, score=0.720 total time=  1
1.1s
[CV 2/5] END .learning_rate=10, n_estimators=90;, score=0.461 total time=
5.0s
[CV 3/5] END .learning_rate=10, n_estimators=90;, score=0.456 total time=
6.7s
[CV 4/5] END .learning_rate=10, n_estimators=90;, score=0.463 total time=
6.6s
[CV 5/5] END .learning_rate=10, n_estimators=90;, score=0.466 total time=
6.9s
[CV 1/5] END learning_rate=10, n_estimators=120;, score=0.467 total time=
9.4s
[CV 2/5] END learning_rate=10, n_estimators=120;, score=0.461 total time=
```

```
9.4s
[CV 3/5] END learning_rate=10, n_estimators=120;, score=0.456 total time=
9.2s
[CV 4/5] END learning_rate=10, n_estimators=120;, score=0.463 total time=
9.1s
[CV 5/5] END learning_rate=10, n_estimators=120;, score=0.466 total time=
9.4s
[CV 1/5] END learning_rate=10, n_estimators=180;, score=0.467 total time=  1
1.0s
[CV 3/5] END learning_rate=10, n_estimators=180;, score=0.456 total time=  1
0.9s
[CV 2/5] END learning_rate=10, n_estimators=180;, score=0.461 total time=  1
1.5s
[CV 4/5] END learning_rate=10, n_estimators=180;, score=0.463 total time=
9.7s
[CV 5/5] END learning_rate=10, n_estimators=180;, score=0.466 total time=
9.7s
[CV 1/5] END learning_rate=10, n_estimators=200;, score=0.467 total time=  1
0.3s
[CV 2/5] END learning_rate=10, n_estimators=200;, score=0.461 total time=  1
0.3s
[CV 3/5] END learning_rate=10, n_estimators=200;, score=0.456 total time=  1
0.0s
[CV 4/5] END learning_rate=10, n_estimators=200;, score=0.463 total time=
9.3s
[CV 5/5] END learning_rate=10, n_estimators=200;, score=0.466 total time=
9.0s
```

Out[117]:



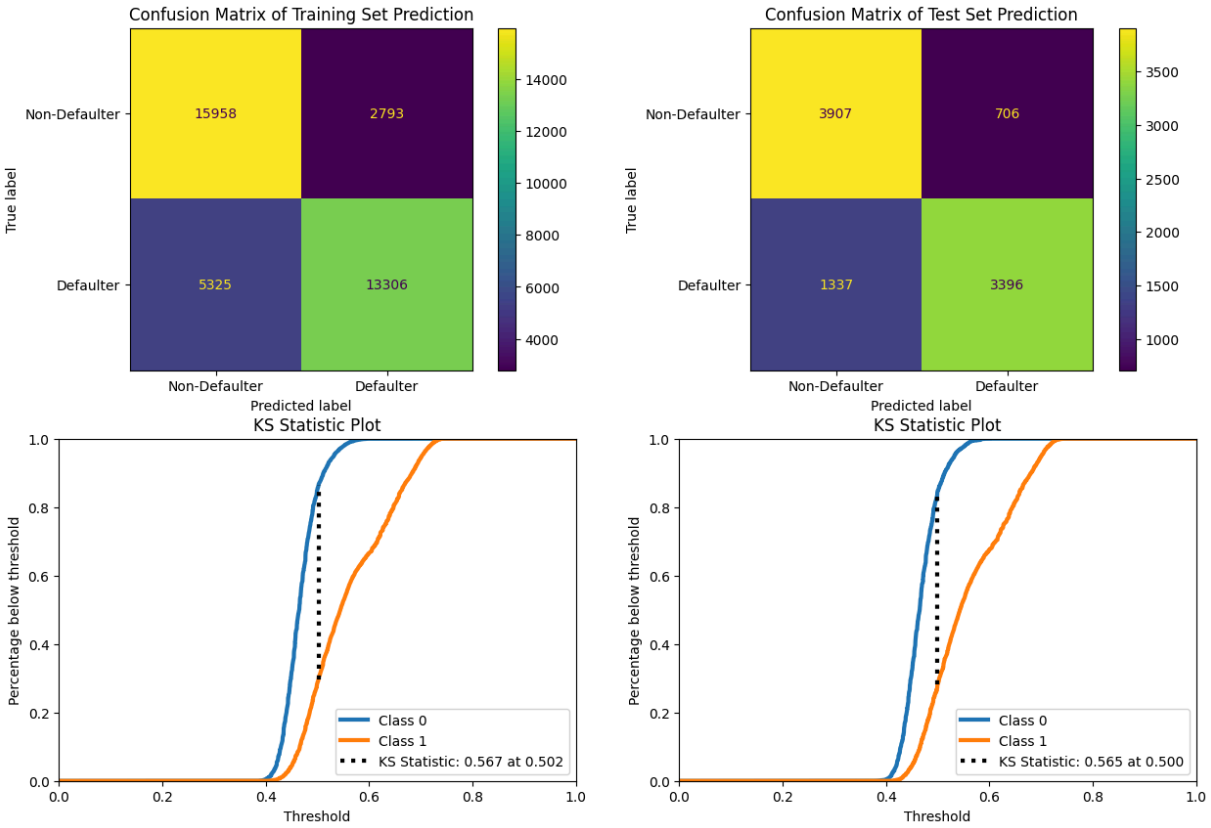
In [118…

```
#Best parameters
print('Best parameters:', ada_cv.best_params_)
ada_best=ada_cv.best_estimator_

#predicting the probabilities of test data
ada_best_test_proba=ada_best.predict_proba(X_test_scaled)
ada_best_train_proba=ada_best.predict_proba(X_train_scaled)

 #predicting the values of y from x via model
y_best_pred_test = ada_best.predict(X_test_scaled)
y_best_pred_train = ada_best.predict(X_train_scaled)
```

```
Best parameters: {'learning_rate': 1, 'n_estimators': 200}
```

In [119…

```
#visualize the model performence
model_vis(y_train,y_test,y_best_pred_train,y_best_pred_test,ada_best_train_p
```



In [120…
```
#view all the metric score in a dataframe
metric_score('Tuned AdaBoost_Classifier',y_test,y_best_pred_test,ada_best_te
```

Out[120]:

	Model	Accuracy	Precision	Recall	F1 Score	KS_Stat
0	Logistic_Regression	0.765033	0.811901	0.697655	0.750455	0.536
1	Tuned Logistic_Regression	0.744597	0.771026	0.705050	0.736563	0.498
2	Decision_Tree_Classifier	0.797988	0.843682	0.737798	0.787196	0.598
3	Tuned Decision_Tree_Classifier	0.793067	0.826300	0.748785	0.785635	0.591
4	Random_Forrest_Classifier	0.856837	0.886762	0.822311	0.853322	0.718
5	Tuned Random_Forrest_Classifier	0.859191	0.888560	0.825481	0.855860	0.722
6	KNN Classifier	0.787075	0.801893	0.769702	0.785468	0.575
7	AdaBoost Classifier	0.772095	0.820488	0.703993	0.757789	0.547
8	Tuned AdaBoost_Classifier	0.781404	0.827889	0.717515	0.768761	0.565

Which hyperparameter optimization technique have you used and why?

Because GridsearchCV was taking too long, I utilised RandomizedSearchCV, which solves the problems of GridSearchCV by going through only a predetermined number of hyperparameter values. It moves randomly throughout the grid to discover the best

collection of hyperparameters. This method eliminates unnecessary computation.It uses the value distribution.

Have you seen any improvement? Note down the improvement with updates Evaluation metric Score Chart.

There is very little improvement in performance of our model,recall value increase to 0.75 from 0.73

ML Model-5 **XGboost**

In [ ]:
```
!pip install --upgrade xgboost scikit-learn
from xgboost import XGBClassifier
```

```
Requirement already satisfied: xgboost in /Library/Frameworks/Python.framewor
k/Versions/3.11/lib/python3.11/site-packages (2.1.3)
Collecting xgboost
  Using cached xgboost-2.1.4-py3-none-macosx_12_0_arm64.whl.metadata (2.1 kB)
Requirement already satisfied: scikit-learn in /Library/Frameworks/Python.fra
mework/Versions/3.11/lib/python3.11/site-packages (1.6.1)
Requirement already satisfied: numpy in /Library/Frameworks/Python.framework/
Versions/3.11/lib/python3.11/site-packages (from xgboost) (1.25.2)
Requirement already satisfied: scipy in /Library/Frameworks/Python.framework/
Versions/3.11/lib/python3.11/site-packages (from xgboost) (1.9.3)
Requirement already satisfied: joblib>=1.2.0 in /Library/Frameworks/Python.fr
amework/Versions/3.11/lib/python3.11/site-packages (from scikit-learn) (1.4.
2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /Library/Frameworks/Py
thon.framework/Versions/3.11/lib/python3.11/site-packages (from scikit-learn)
(3.5.0)
Using cached xgboost-2.1.4-py3-none-macosx_12_0_arm64.whl (1.9 MB)
Installing collected packages: xgboost
  Attempting uninstall: xgboost
    Found existing installation: xgboost 2.1.3
    Uninstalling xgboost-2.1.3:
      Successfully uninstalled xgboost-2.1.3
Successfully installed xgboost-2.1.4

[notice] A new release of pip is available: 25.0 -> 25.0.1
[notice] To update, run: pip install --upgrade pip
```

In [127…
```
from xgboost import XGBClassifier

# Initialize and train the model
model_xg = XGBClassifier()
model_xg.fit(X_train_scaled, y_train)

print("XGBoost model trained successfully!")
```

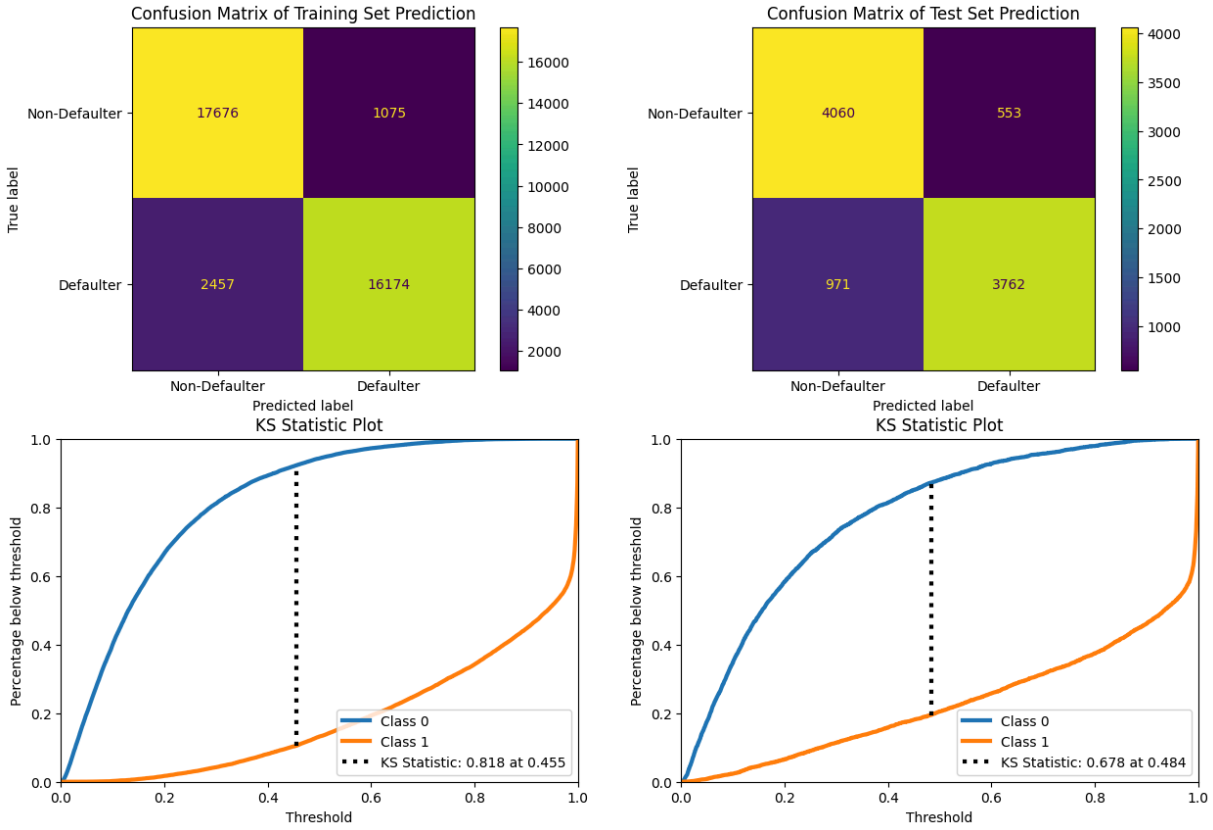
```
XGBoost model trained successfully!
```

In [128…
```
#predicting the probabilities of test data
xg_test_proba=model_xg.predict_proba(X_test_scaled)
xg_train_proba=model_xg.predict_proba(X_train_scaled)
 #predicting the values of y from x via model
```

```
y_pred_test = model_xg.predict(X_test_scaled)
y_pred_train = model_xg.predict(X_train_scaled)
```

1. Explain the ML Model used and it's performance using Evaluation metric Score Chart.

In [129…
```
#visualize the model performence
model_vis(y_train,y_test,y_pred_train,y_pred_test,xg_train_proba,xg_test_prc
```



In [130…
```
#view all the metric score in a dataframe
metric_score('XGB_Classifier',y_test,y_pred_test,xg_test_proba)
```

Out[130]:

	Model	Accuracy	Precision	Recall	F1 Score	KS_Stat
0	Logistic_Regression	0.765033	0.811901	0.697655	0.750455	0.536
1	Tuned Logistic_Regression	0.744597	0.771026	0.705050	0.736563	0.498
2	Decision_Tree_Classifier	0.797988	0.843682	0.737798	0.787196	0.598
3	Tuned Decision_Tree_Classifier	0.793067	0.826300	0.748785	0.785635	0.591
4	Random_Forrest_Classifier	0.856837	0.886762	0.822311	0.853322	0.718
5	Tuned Random_Forrest_Classifier	0.859191	0.888560	0.825481	0.855860	0.722
6	KNN Classifier	0.787075	0.801893	0.769702	0.785468	0.575
7	AdaBoost Classifier	0.772095	0.820488	0.703993	0.757789	0.547
8	Tuned AdaBoost_Classifier	0.781404	0.827889	0.717515	0.768761	0.565
9	XGB_Classifier	0.836936	0.871842	0.794845	0.831565	0.678

2. Cross- Validation & Hyperparameter Tuning

In [154…

```
!pip uninstall -y scikit-learn xgboost

!pip install xgboost scikit-learn==1.3.0
```

```
WARNING: Skipping scikit-learn as it is not installed.
WARNING: Skipping xgboost as it is not installed.
Collecting xgboost
  Using cached xgboost-2.1.4-py3-none-macosx_12_0_arm64.whl.metadata (2.1 kB)
Collecting scikit-learn==1.3.0
  Using cached scikit_learn-1.3.0-cp311-cp311-macosx_12_0_arm64.whl.metadata
(11 kB)
Requirement already satisfied: numpy>=1.17.3 in /Library/Frameworks/Python.fr
amework/Versions/3.11/lib/python3.11/site-packages (from scikit-learn==1.3.0)
(1.25.2)
Requirement already satisfied: scipy>=1.5.0 in /Library/Frameworks/Python.fra
mework/Versions/3.11/lib/python3.11/site-packages (from scikit-learn==1.3.0)
(1.9.3)
Requirement already satisfied: joblib>=1.1.1 in /Library/Frameworks/Python.fr
amework/Versions/3.11/lib/python3.11/site-packages (from scikit-learn==1.3.0)
(1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Library/Frameworks/Py
thon.framework/Versions/3.11/lib/python3.11/site-packages (from scikit-learn=
=1.3.0) (3.5.0)
Using cached scikit_learn-1.3.0-cp311-cp311-macosx_12_0_arm64.whl (9.4 MB)
Using cached xgboost-2.1.4-py3-none-macosx_12_0_arm64.whl (1.9 MB)
Installing collected packages: xgboost, scikit-learn
Successfully installed scikit-learn-1.3.0 xgboost-2.1.4

[notice] A new release of pip is available: 25.0 -> 25.0.1
[notice] To update, run: pip install --upgrade pip
```

In [ ]:
```
# ML Model - 1 Implementation with hyperparameter optimization techniques (i
xg=XGBClassifier()
param_grid={'n_estimators': [50,100,150],'max_depth': [3,5,9]}
xg_cv=GridSearchCV(xg, param_grid, scoring = 'recall',n_jobs = -1, verbose =
# Fit the Algorithm
xg_cv.fit(X_train_scaled,y_train)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[155], line 23
     13 xg_cv = GridSearchCV(
     14     estimator=xg,
     15     param_grid=param_grid,
   (...)
     19     cv=5
     20 )
     22 # Fit the model
---> 23 xg_cv.fit(X_train_scaled, y_train)
     25 # Print best parameters
     26 print("Best parameters found:", xg_cv.best_params_)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/sklearn/base.py:1389, in _fit_context.<locals>.decorator.<locals>.wrapper(estimator, *args, **kwargs)
   1382     estimator._validate_params()
   1384 with config_context(
   1385     skip_parameter_validation=(
   1386         prefer_skip_nested_validation or global_skip_validation
   1387     )
   1388 ):
-> 1389     return fit_method(estimator, *args, **kwargs)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/sklearn/model_selection/_search.py:933, in BaseSearchCV.fit(self, X, y, **params)
    929 params = _check_method_params(X, params=params)
    931 routed_params = self._get_routed_params_for_fit(params)
--> 933 cv_orig = check_cv(self.cv, y, classifier=is_classifier(estimator))
    934 n_splits = cv_orig.get_n_splits(X, y, **routed_params.splitter.split)
    936 base_estimator = clone(self.estimator)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/sklearn/base.py:1237, in is_classifier(estimator)
   1230     warnings.warn(
   1231         f"passing a class to {print(inspect.stack()[0][3])} is deprecated and "
   1232         "will be removed in 1.8. Use an instance of the class instead.",
   1233         FutureWarning,
   1234     )
   1235     return getattr(estimator, "_estimator_type", None) == "classifier"
-> 1237 return get_tags(estimator).estimator_type == "classifier"

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/sklearn/utils/_tags.py:430, in get_tags(estimator)
    428 for klass in reversed(type(estimator).mro()):
    429     if "__sklearn_tags__" in vars(klass):
--> 430         sklearn_tags_provider[klass] = klass.__sklearn_tags__(estimator)  # type: ignore[attr-defined]
    431         class_order.append(klass)
    432     elif "_more_tags" in vars(klass):
```

```
File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/sklearn/base.py:540, in ClassifierMixin.__sklearn_tags__(self)
    539 def __sklearn_tags__(self):
--> 540     tags = super().__sklearn_tags__()
    541     tags.estimator_type = "classifier"
    542     tags.classifier_tags = ClassifierTags()

AttributeError: 'super' object has no attribute '__sklearn_tags__'
```

In [156…
```
#Best parameters
print('Best parameters:', xg_cv.best_params_)
xg_best=xg_cv.best_estimator_

#predicting the probabilities of test data
xg_best_test_proba=xg_best.predict_proba(X_test_scaled)
xg_best_train_proba=xg_best.predict_proba(X_train_scaled)

 #predicting the values of y from x via model
y_best_pred_test = xg_best.predict(X_test_scaled)
y_best_pred_train = xg_best.predict(X_train_scaled)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[156], line 2
      1 #Best parameters
----> 2 print('Best parameters:', xg_cv.best_params_)
      3 xg_best=xg_cv.best_estimator_
      5 #predicting the probabilities of test data

AttributeError: 'GridSearchCV' object has no attribute 'best_params_'
```

In [157…
```
#visualize the model performence
model_vis(y_train,y_test,y_best_pred_train,y_best_pred_test,xg_best_train_pr
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[157], line 2
      1 #visualize the model performence
----> 2 model_vis(y_train,y_test,y_best_pred_train,y_best_pred_test,xg_best_train_proba,xg_best_test_proba)

NameError: name 'xg_best_train_proba' is not defined
```

In [158…
```
#view all the metric score in a dataframe
metric_score('Tuned XGB_Classifier',y_test,y_best_pred_test,xg_best_test_prc
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[158], line 2
      1 #view all the metric score in a dataframe
----> 2 metric_score('Tuned XGB_Classifier',y_test,y_best_pred_test,xg_best_test_proba)

NameError: name 'xg_best_test_proba' is not defined
```

In [161…

```
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV

# Define XGBClassifier
xg = XGBClassifier(use_label_encoder=False, eval_metric="logloss")

# Define parameter grid for tuning
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [3, 5, 9]
}

# Set up GridSearchCV
xg_cv = GridSearchCV(
    estimator=xg,
    param_grid=param_grid,
    scoring='recall',
    n_jobs=1,  # Use single thread to avoid multiprocessing issues
    verbose=3,
    cv=5
)

# Fit the model
xg_cv.fit(X_train_scaled, y_train)

# Print best parameters
print("Best parameters found:", xg_cv.best_params_)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[161], line 24
     14 xg_cv = GridSearchCV(
     15     estimator=xg,
     16     param_grid=param_grid,
   (...)
     20     cv=5
     21 )
     23 # Fit the model
---> 24 xg_cv.fit(X_train_scaled, y_train)
     26 # Print best parameters
     27 print("Best parameters found:", xg_cv.best_params_)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/sklearn/base.py:1389, in _fit_context.<locals>.decorator.<locals>.wrapper(estimator, *args, **kwargs)
   1382     estimator._validate_params()
   1384 with config_context(
   1385     skip_parameter_validation=(
   1386         prefer_skip_nested_validation or global_skip_validation
   1387     )
   1388 ):
-> 1389     return fit_method(estimator, *args, **kwargs)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/sklearn/model_selection/_search.py:933, in BaseSearchCV.fit(self, X, y, **params)
    929 params = _check_method_params(X, params=params)
    931 routed_params = self._get_routed_params_for_fit(params)
--> 933 cv_orig = check_cv(self.cv, y, classifier=is_classifier(estimator))
    934 n_splits = cv_orig.get_n_splits(X, y, **routed_params.splitter.split)
    936 base_estimator = clone(self.estimator)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/sklearn/base.py:1237, in is_classifier(estimator)
   1230     warnings.warn(
   1231         f"passing a class to {print(inspect.stack()[0][3])} is deprecated and "
   1232         "will be removed in 1.8. Use an instance of the class instead.",
   1233         FutureWarning,
   1234     )
   1235     return getattr(estimator, "_estimator_type", None) == "classifier"
-> 1237 return get_tags(estimator).estimator_type == "classifier"

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/sklearn/utils/_tags.py:430, in get_tags(estimator)
    428 for klass in reversed(type(estimator).mro()):
    429     if "__sklearn_tags__" in vars(klass):
--> 430         sklearn_tags_provider[klass] = klass.__sklearn_tags__(estimator)  # type: ignore[attr-defined]
    431         class_order.append(klass)
    432     elif "_more_tags" in vars(klass):
```

```
File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/sklearn/base.py:540, in ClassifierMixin.__sklearn_tags__(self)
    539 def __sklearn_tags__(self):
--> 540     tags = super().__sklearn_tags__()
    541     tags.estimator_type = "classifier"
    542     tags.classifier_tags = ClassifierTags()

AttributeError: 'super' object has no attribute '__sklearn_tags__'
```

1. Which Evaluation metrics did you consider for a positive business impact and why?

I'll be using Recall primarily for model evaluation because False Negative indicates that a person will not default when they actually do. And recognising defaulter clients as non-defaulters will result in a large loss for the bank, thus we must reduce False Negative, and as False Negative decreases, Recall will grow.

2. Which ML model did you choose from the above created models as your final prediction model and why?

I will use Random Forest Classifier as my final prediction model because its recall is approximately 83.89%, which is greater than other models in both cases.

KNN is not a memory-efficient algorithm. As the model needs to store all of the data points, it gets exceedingly sluggish as the number of data points increases. It is computationally intensive since the algorithm must calculate the distance between all datapoints and determine the nearest neighbours for each datapoint.

3. Explain the model which you have used and the feature importance using any model explainability tool?

For Model Explaination we will use LIME method.

LIME stands for Local Interpretable Model-agnostic Explanations. LIME focuses on training local surrogate models to explain individual predictions. Local surrogate models are interpretable models that are used to explain individual predictions of black box machine learning models. Surrogate models are trained to approximate the predictions of the underlying black box model. Instead of training a global surrogate model, LIME focuses on training local surrogate models.

In [159…

```
! pip install lime
```

```
Collecting lime
  Downloading lime-0.2.0.1.tar.gz (275 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: matplotlib in /Library/Frameworks/Python.frame
work/Versions/3.11/lib/python3.11/site-packages (from lime) (3.10.0)
Requirement already satisfied: numpy in /Library/Frameworks/Python.framework/
Versions/3.11/lib/python3.11/site-packages (from lime) (1.25.2)
Requirement already satisfied: scipy in /Library/Frameworks/Python.framework/
Versions/3.11/lib/python3.11/site-packages (from lime) (1.9.3)
Requirement already satisfied: tqdm in /Library/Frameworks/Python.framework/V
ersions/3.11/lib/python3.11/site-packages (from lime) (4.65.0)
Requirement already satisfied: scikit-learn>=0.18 in /Library/Frameworks/Pyth
on.framework/Versions/3.11/lib/python3.11/site-packages (from lime) (1.3.0)
Collecting scikit-image>=0.12 (from lime)
  Downloading scikit_image-0.25.2-cp311-cp311-macosx_12_0_arm64.whl.metadata
(14 kB)
Collecting scipy (from lime)
  Using cached scipy-1.15.2-cp311-cp311-macosx_14_0_arm64.whl.metadata (61 k
B)
Requirement already satisfied: networkx>=3.0 in /Library/Frameworks/Python.fr
amework/Versions/3.11/lib/python3.11/site-packages (from scikit-image>=0.12->
lime) (3.2.1)
Collecting pillow>=10.1 (from scikit-image>=0.12->lime)
  Downloading pillow-11.1.0-cp311-cp311-macosx_11_0_arm64.whl.metadata (9.1 k
B)
Requirement already satisfied: imageio!=2.35.0,>=2.33 in /Library/Frameworks/
Python.framework/Versions/3.11/lib/python3.11/site-packages (from scikit-imag
e>=0.12->lime) (2.35.1)
Collecting tifffile>=2022.8.12 (from scikit-image>=0.12->lime)
  Downloading tifffile-2025.2.18-py3-none-any.whl.metadata (31 kB)
Requirement already satisfied: packaging>=21 in /Library/Frameworks/Python.fr
amework/Versions/3.11/lib/python3.11/site-packages (from scikit-image>=0.12->
lime) (23.1)
Collecting lazy-loader>=0.4 (from scikit-image>=0.12->lime)
  Downloading lazy_loader-0.4-py3-none-any.whl.metadata (7.6 kB)
Requirement already satisfied: joblib>=1.1.1 in /Library/Frameworks/Python.fr
amework/Versions/3.11/lib/python3.11/site-packages (from scikit-learn>=0.18->
lime) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Library/Frameworks/Py
thon.framework/Versions/3.11/lib/python3.11/site-packages (from scikit-learn>
=0.18->lime) (3.5.0)
Requirement already satisfied: contourpy>=1.0.1 in /Library/Frameworks/Pytho
n.framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib->lim
e) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /Library/Frameworks/Python.fra
mework/Versions/3.11/lib/python3.11/site-packages (from matplotlib->lime) (0.
11.0)
Requirement already satisfied: fonttools>=4.22.0 in /Library/Frameworks/Pytho
n.framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib->lim
e) (4.40.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /Library/Frameworks/Pytho
n.framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib->lim
e) (1.4.4)
Requirement already satisfied: pyparsing>=2.3.1 in /Library/Frameworks/Pytho
n.framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib->lim
e) (3.1.0)
```

```
Requirement already satisfied: python-dateutil>=2.7 in /Library/Frameworks/Py
thon.framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib->l
ime) (2.8.2)
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.framewo
rk/Versions/3.11/lib/python3.11/site-packages (from python-dateutil>=2.7->mat
plotlib->lime) (1.16.0)
Downloading scikit_image-0.25.2-cp311-cp311-macosx_12_0_arm64.whl (13.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.2/13.2 MB 45.2 MB/s eta 0:00:0
0 0:00:01
Using cached scipy-1.15.2-cp311-cp311-macosx_14_0_arm64.whl (22.4 MB)
Downloading lazy_loader-0.4-py3-none-any.whl (12 kB)
Downloading pillow-11.1.0-cp311-cp311-macosx_11_0_arm64.whl (3.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.1/3.1 MB 45.1 MB/s eta 0:00:00
Downloading tifffile-2025.2.18-py3-none-any.whl (226 kB)
Building wheels for collected packages: lime
  Building wheel for lime (setup.py) ... done
  Created wheel for lime: filename=lime-0.2.0.1-py3-none-any.whl size=283835
sha256=b28e40c2ceaf7f1746d5fc13ee2f154d7bc4e25061d64806d06f5bbcdf6ad764
  Stored in directory: /Users/sahilbhagat/Library/Caches/pip/wheels/85/fa/a3/
9c2d44c9f3cd77cf4e533b58900b2bf4487f2a17e8ec212a3d
Successfully built lime
Installing collected packages: tifffile, scipy, pillow, lazy-loader, scikit-i
mage, lime
  Attempting uninstall: scipy
    Found existing installation: scipy 1.9.3
    Uninstalling scipy-1.9.3:
      Successfully uninstalled scipy-1.9.3
  Attempting uninstall: pillow
    Found existing installation: Pillow 9.5.0
    Uninstalling Pillow-9.5.0:
      Successfully uninstalled Pillow-9.5.0
Successfully installed lazy-loader-0.4 lime-0.2.0.1 pillow-11.1.0 scikit-imag
e-0.25.2 scipy-1.15.2 tifffile-2025.2.18

[notice] A new release of pip is available: 25.0 -> 25.0.1
[notice] To update, run: pip install --upgrade pip
```

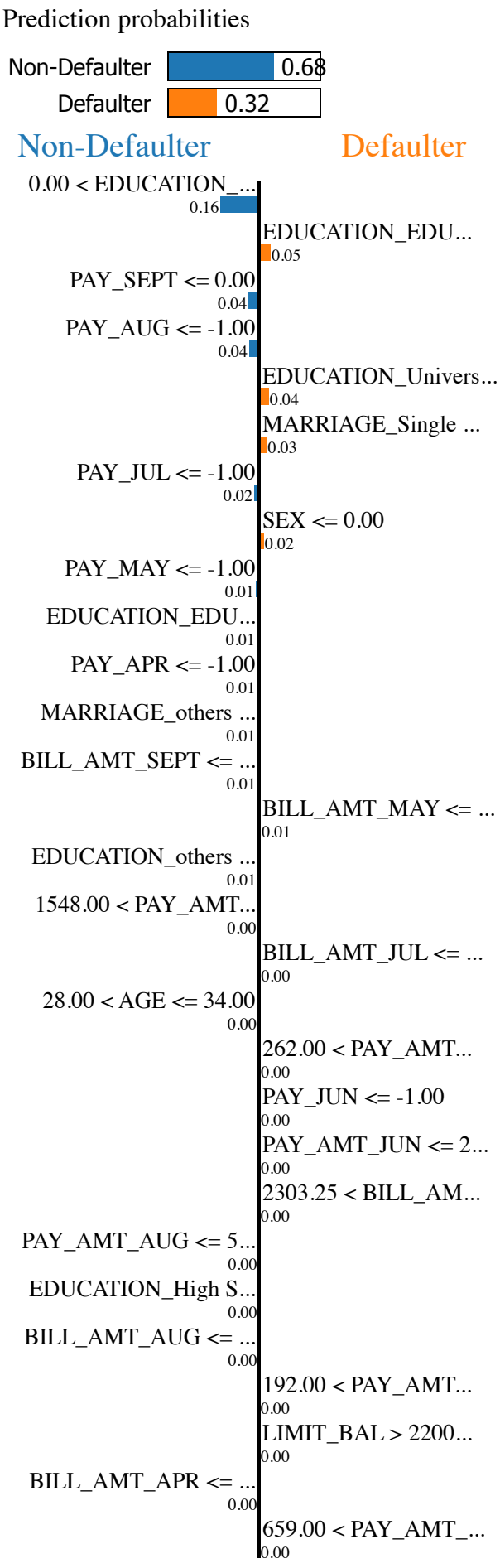
In [162…
```
#import lime library
import lime
import lime.lime_tabular
```

In [163…
```
# LIME has one explainer for all the models
explainer = lime.lime_tabular.LimeTabularExplainer(X_train.values, feature_r
                                                   class_names=['Non-Defaulte
```

In [164…
```
# Choose the 5th instance and use it to predict the results
j = 5
exp = explainer.explain_instance(X_test.values[j], rf_best.predict_proba, nu
```

```
Intercept 0.5668156068695205
Prediction_local [0.39149621]
Right: 0.316
```

In [165…
```
# Show the predictions
exp.show_in_notebook(show_table=True)
```


Prediction probabilities
Non-Defaulter
0.68
Defaulter
0.32
Non-Defaulter
Defaulter
0.00 < EDUCATION_...
0.16
EDUCATION_EDU...
0.05
PAY_SEPT <= 0.00
0.04
PAY_AUG <= -1.00
0.04
EDUCATION_Univers...
0.04
MARRIAGE_Single ...
0.03
PAY_JUL <= -1.00
0.02
SEX <= 0.00
0.02
PAY_MAY <= -1.00
0.01
EDUCATION_EDU...
0.01
PAY_APR <= -1.00
0.01
MARRIAGE_others ...
0.01
BILL_AMT_SEPT <= ...
0.01
BILL_AMT_MAY <= ...
0.01
EDUCATION_others ...
0.01
1548.00 < PAY_AMT...
0.00
BILL_AMT_JUL <= ...
0.00
28.00 < AGE <= 34.00
0.00
262.00 < PAY_AMT...
0.00
PAY_JUN <= -1.00
0.00
PAY_AMT_JUN <= 2...
0.00
2303.25 < BILL_AM...
0.00
PAY_AMT_AUG <= 5...
0.00
EDUCATION_High S...
0.00
BILL_AMT_AUG <= ...
0.00
192.00 < PAY_AMT...
0.00
LIMIT_BAL > 2200...
0.00
BILL_AMT_APR <= ...
0.00
659.00 < PAY_AMT_...
0.00

Feature	Value
EDUCATION_EDUCATION_High School	1.00
EDUCATION_EDUCATION_University	0.00
PAY_SEPT	-2.00
PAY_AUG	-2.00
EDUCATION_University	0.00
MARRIAGE_Single	0.00
PAY_JUL	-1.00
SEX	0.00
PAY_MAY	-1.00

8. Future Work (Optional)

1. Save the best performing ml model in a pickle file or joblib file format for deployment process.

In [ ]:
```
# Save the File
```

2. Again Load the saved model file and try to predict unseen data for a sanity check.

In [ ]:
```
# Load the File and predict unseen data.
```

Congrats! Your model is successfully created and ready for deployment on a live server for a real user interaction !!!

Conclusion

- We saw that our dataset had an uneven class.
- The majority of credit card holders were female, while male clients had a high default ratio.
- Higher educated people are less likely to default, whereas fewer educated people are more likely to default.
- The likelihood of default increases after the age of 60.

- In this scenario, 'Recall' is a significant parameter to compare all of the algorithms. Because the corporation cannot afford to anticipate a false negative, that is, to predict a defaulter as a non-defaulter. Since the corporation is the one who would deliver money to consumers, if for any reason, sending money to defaulter increases the risk of obtaining the investment returned. As a result, recognising false negatives is critical in this case.
- The Random Forest Classifier model has a high recall (83.89%).
- I will not use KNN as my basic model because it has a similar recall to Random Forest Classisieer, but its bestparams n_neighbors is 1 and can be noisy, leading to outlier effects in the model.

Hurrah! You have successfully completed your Machine Learning Capstone Project !!!