Capstone Project - 3

Credit Card Default Prediction

Zuber Ahmad

Contents :

- PREPROCESSING THE DATASHEET
- EXPLODATORY DATA ANALYSIS
  - Data Exploration
  - Data Analysis
  - Univariate Analysis
  - Bivariate Analysis
  - Smote
  - Feature Engineering
  - One Hot Encoding
- Modelling
  - Logistic Regression
  - Random Forest Classifier
  - Naïve Bayes Classifier
  - Support Vector Machine
  - XG Boost
  - Hyperparameter
- EVALUATION METRICS & ESTIMATOR
  - GridSearchCV
  - Confusion Matrix
  - AUC-ROC Curve
  - Precision
  - Recall
  - F1 Score

INTRODUCTION

- **According to the Federal Reserve economic data, the default rate on credit loans across all commercial banks is at an all-time high for the past 66 months, and it is likely to continue to climb throughout 2020.**
- **The climbing delinquencies will result in a significant amount of money lose from the lending institutions, such as commercial banks. Therefore, banks must have a risk prediction model and be able to classify the most relative characteristics that are indicative of people who have a higher probability of default on credit.**
- **Because of the risks inherent in such a large portion of the economy, building models for consumer spending behaviors to limit risk expo-sures in this sector is becoming more critical. For this to be a viable option, the predictions need to be reasonably accurate.**

PROBLEM STATEMENT

- **This project is aimed at predicting the case of customers default payments in Taiwan. From the perspective of risk management, the result of predictive accuracy of the estimated probability of default will be more valuable than the binary result of classification - credible or not credible clients. We can use the K-S chart to evaluate which customers will default on their credit card payments.**
- **We are all aware what is credit card. It is type of payment card in which charges are made against a line of credit instead of the account holder's cash deposits. When someone uses a credit card to make a purchase, that person's account accrues a balance that must be paid off each month.**
- **Credit card default happens when you have become severely delinquent on your credit card payment. Missing credit card payments once or twice does not count as a default. A payment default occurs when you fail to pay the Minimum Amount Due on the credit card for a few consecutive months.**
- **So now we know what a credit card is. Now let's see one of problems faced by companies who provide credit cards. Yes it is the people who do not clear off the credit card debt aka credit card defaulters.**
- **The research aims at developing a mechanism to predict the credit card default beforehand and to identify the potential customer base that can be offered various credit instruments so as to invite minimum default.**

OBJECTIVES

- Objective of our project is to predict which customer might default in upcoming months. Before going any further let's have a quick look on definition of what actually meant by Credit Card Default.
- First, multiple latest datasets have been used to build a machine learning model for credit risk prediction.
- Second, the data imbalance problem has been explored by comparing the different resampling techniques and evaluate the performance that which the resampling technique has given effective results with a machine learning classifier.
- Limited work was done on resampling techniques for data balancing in this domain because only a few resampling techniques were employed and also obtained less efficient results.
- Lastly, the interpretable model is also deployed on the web to ease the different stakeholders. This model will help commercial banks, financial organizations, loan institutes, and other decision-makers to predict the credit defaulter earlier.

Attributes Information

This research employed a binary variable, default payment (Yes = 1, No = 0), as the response variable. This study reviewed the literature and used the following 23 variables as explanatory variables:

- **X1: Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.**
- **X2: Gender (1 = male; 2 = female).**
- **X3: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).**
- **X4: Marital status (1 = married; 2 = single; 3 = others).**
- **X5: Age (year).**
- **X6 - X11: History of past payment. We tracked the past monthly payment records (from April to September, 2005) as follows: X6 = the repayment status in September, 2005; X7 = the repayment status in August, 2005; X11 = the repayment status in April, 2005. The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.**
- **X12-X17: Amount of bill statement (NT dollar). X12 = amount of bill statement in September, 2005; X13 = amount of bill statement in August, 2005; X17 = amount of bill statement in April, 2005.**
- **X18-X23: Amount of previous payment (NT dollar). X18 = amount paid in September, 2005; X19 = amount paid in August, 2005; X23 = amount paid in April, 2005.**

Data Summary

```
credit_df.head()
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	default payment next month
0	1	20000	2	2	1	24	2	2	-1	-1	...	0	0	0	0	689	0	0	0	0	1
1	2	120000	2	2	2	26	-1	2	0	0	...	3272	3455	3261	0	1000	1000	1000	0	2000	1
2	3	90000	2	2	2	34	0	0	0	0	...	14331	14948	15549	1518	1500	1000	1000	1000	5000	0
3	4	50000	2	2	1	37	0	0	0	0	...	28314	28959	29547	2000	2019	1200	1100	1069	1000	0
4	5	50000	1	2	1	57	-1	0	-1	0	...	20940	19146	19131	2000	36681	10000	9000	689	679	0

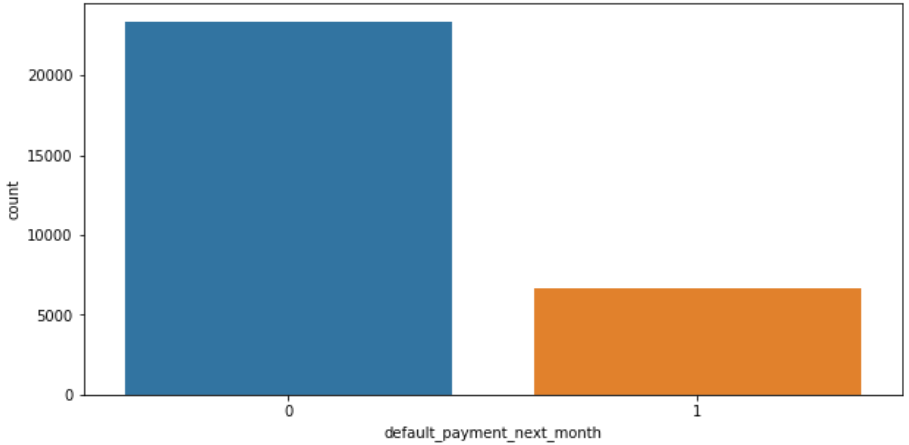
5 rows × 25 columns

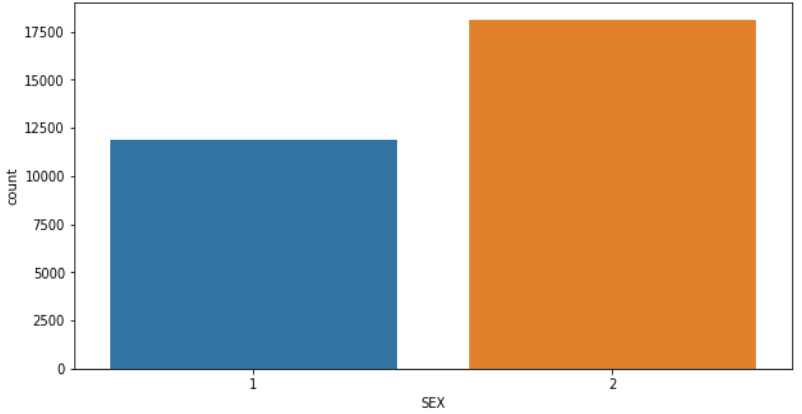
Data Summary

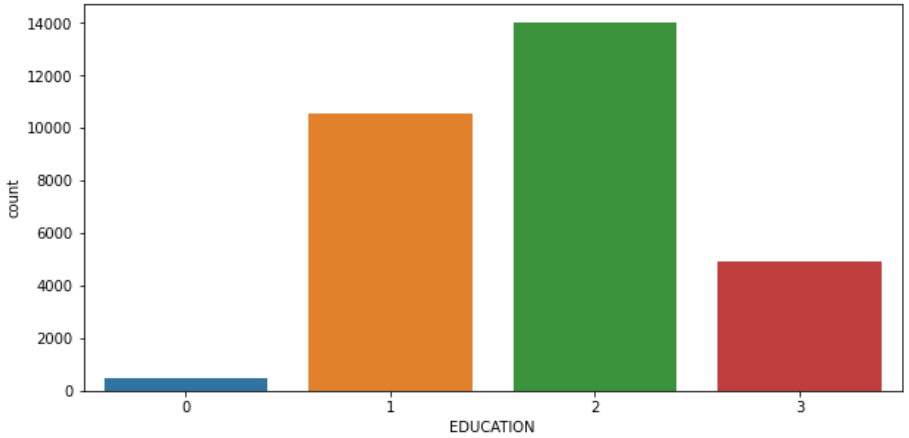
```
credit_df.describe().T
```

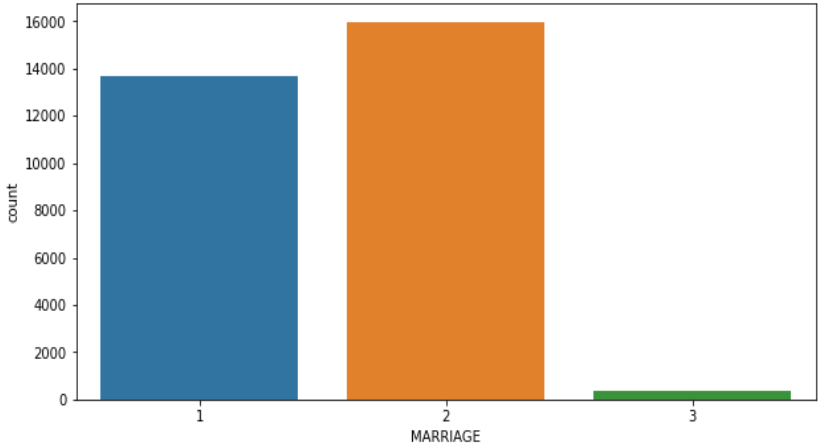
	count	mean	std	min	25%	50%	75%	max
ID	30000.0	15000.500000	8660.398374	1.0	7500.75	15000.5	22500.25	30000.0
LIMIT_BAL	30000.0	167484.322667	129747.661567	10000.0	50000.00	140000.0	240000.00	1000000.0
SEX	30000.0	1.603733	0.489129	1.0	1.00	2.0	2.00	2.0
EDUCATION	30000.0	1.853133	0.790349	0.0	1.00	2.0	2.00	6.0
MARRIAGE	30000.0	1.551867	0.521970	0.0	1.00	2.0	2.00	3.0
AGE	30000.0	35.485500	9.217904	21.0	28.00	34.0	41.00	79.0
PAY_0	30000.0	-0.016700	1.123802	-2.0	-1.00	0.0	0.00	8.0
PAY_2	30000.0	-0.133767	1.197186	-2.0	-1.00	0.0	0.00	8.0
PAY_3	30000.0	-0.166200	1.196868	-2.0	-1.00	0.0	0.00	8.0
PAY_4	30000.0	-0.220667	1.169139	-2.0	-1.00	0.0	0.00	8.0
PAY_5	30000.0	-0.266200	1.133187	-2.0	-1.00	0.0	0.00	8.0
PAY_6	30000.0	-0.291100	1.149988	-2.0	-1.00	0.0	0.00	8.0
BILL_AMT1	30000.0	51223.330900	73635.860576	-165580.0	3558.75	22381.5	67091.00	964511.0
BILL_AMT2	30000.0	49179.075167	71173.768783	-69777.0	2984.75	21200.0	64006.25	983931.0
BILL_AMT3	30000.0	47013.154800	69349.387427	-157264.0	2666.25	20088.5	60164.75	1664089.0
BILL_AMT4	30000.0	43262.948967	64332.856134	-170000.0	2326.75	19052.0	54506.00	891586.0
BILL_AMT5	30000.0	40311.400967	60797.155770	-81334.0	1763.00	18104.5	50190.50	927171.0
BILL_AMT6	30000.0	38871.760400	59554.107537	-339603.0	1256.00	17071.0	49198.25	961664.0
PAY_AMT1	30000.0	5663.580500	16563.280354	0.0	1000.00	2100.0	5006.00	873552.0
PAY_AMT2	30000.0	5921.163500	23040.870402	0.0	833.00	2009.0	5000.00	1684259.0
PAY_AMT3	30000.0	5225.681500	17606.961470	0.0	390.00	1800.0	4505.00	896040.0
PAY_AMT4	30000.0	4826.076867	15666.159744	0.0	296.00	1500.0	4013.25	621000.0
PAY_AMT5	30000.0	4799.387633	15278.305679	0.0	252.50	1500.0	4031.50	426529.0
PAY_AMT6	30000.0	5215.502567	17777.465775	0.0	117.75	1500.0	4000.00	528666.0
default payment next month	30000.0	0.221200	0.415062	0.0	0.00	0.0	0.00	1.0

Exploratory Data Analysis – Univariate Analysis

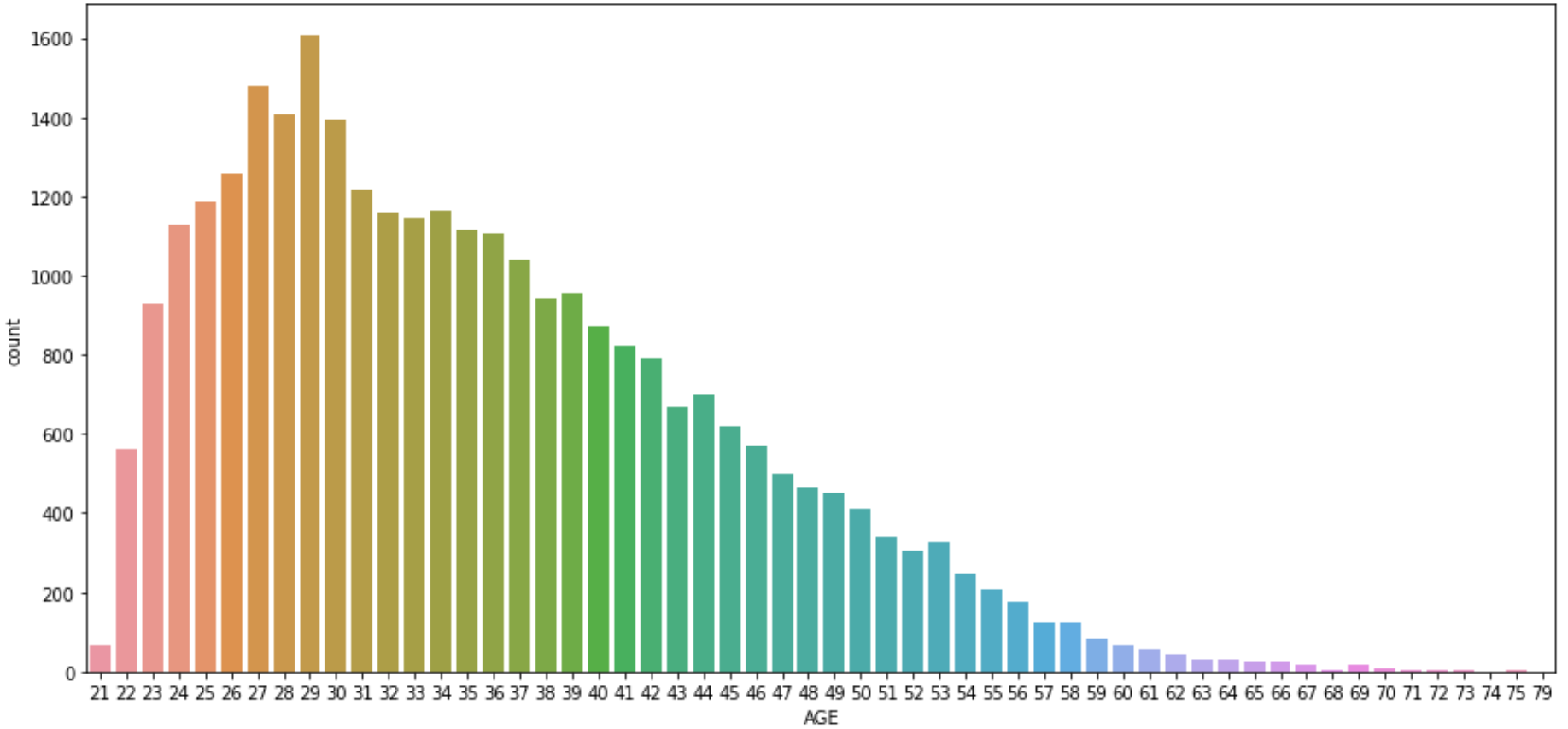




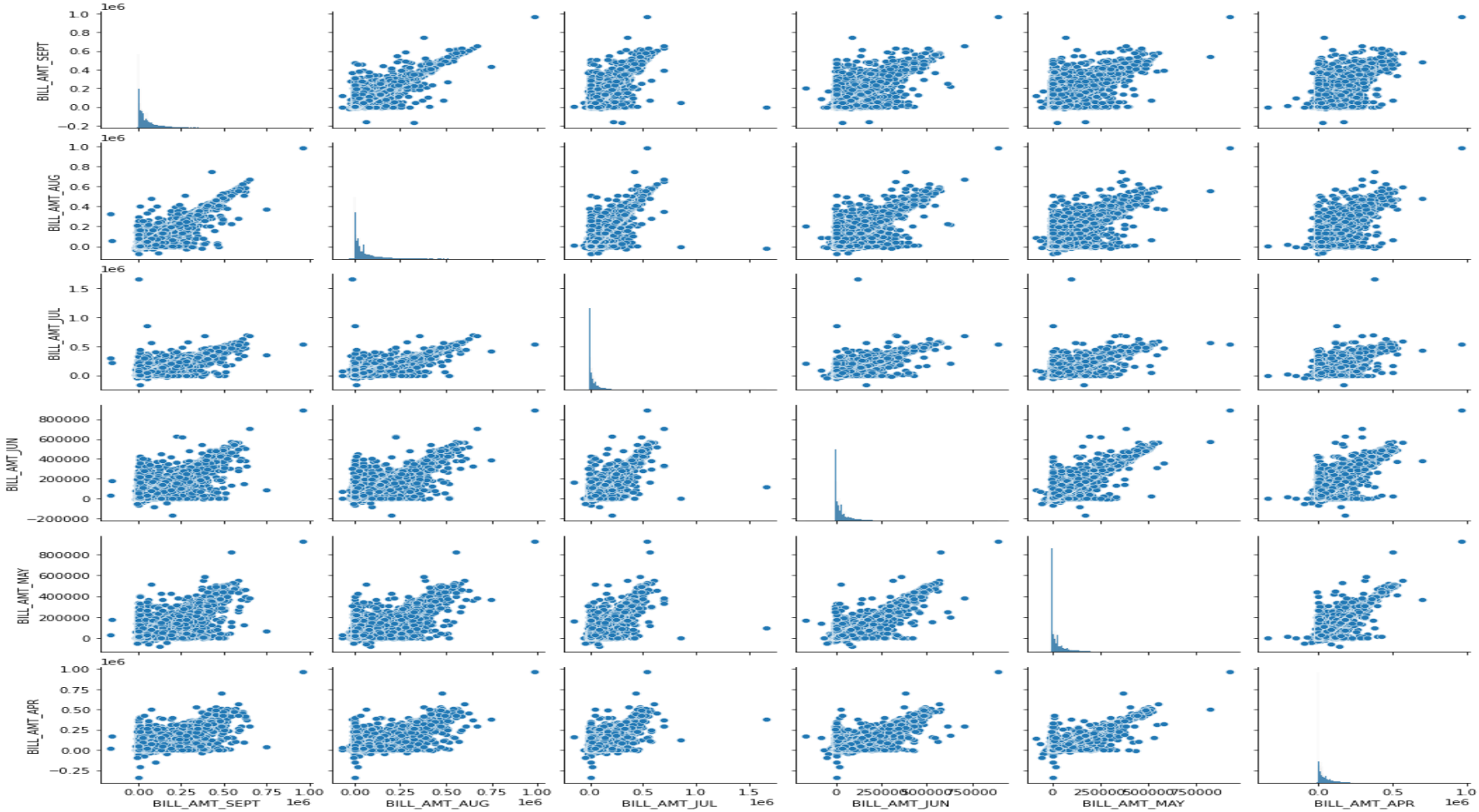




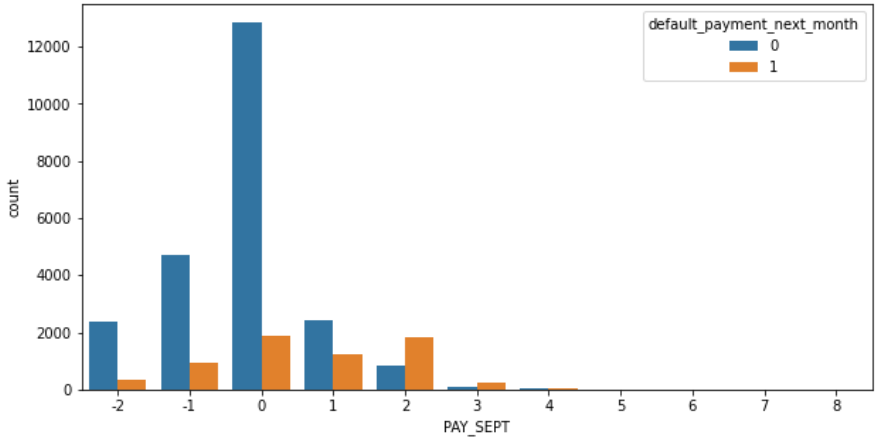
Exploratory Data Analysis – Univariate Analysis

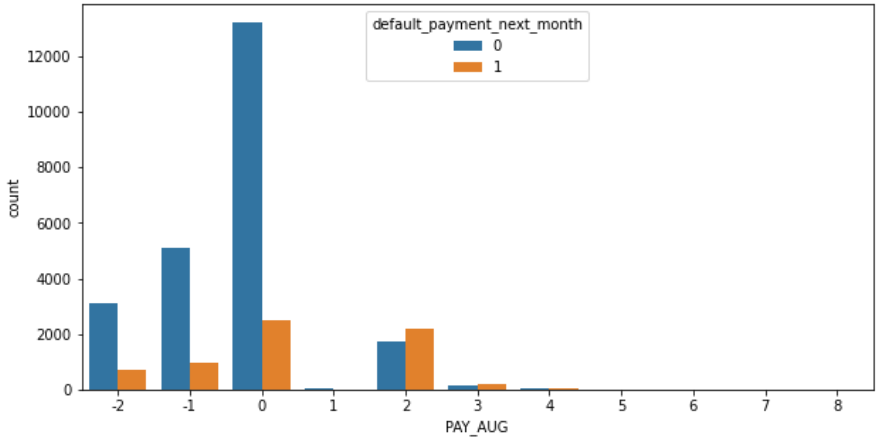


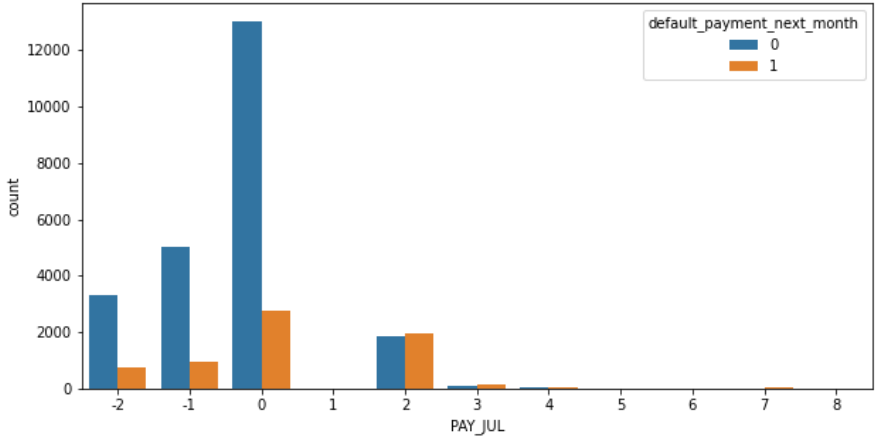
EDA - Total Bill Amount

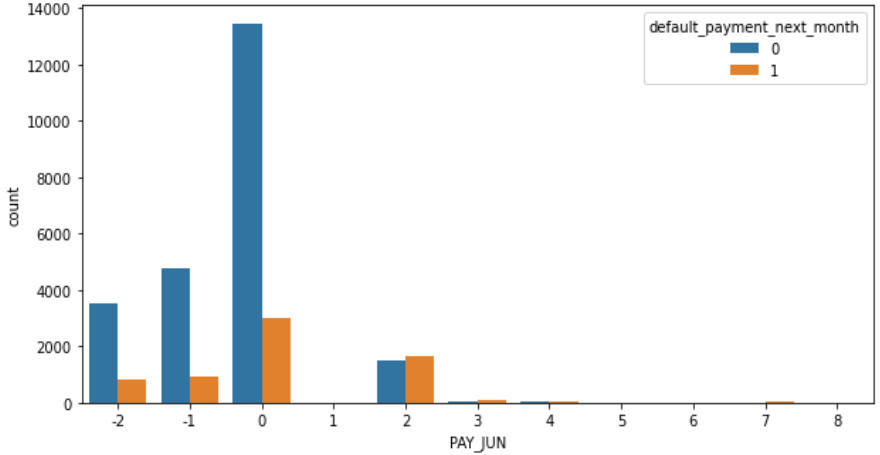


Exploratory Data Analysis – Univariate Analysis

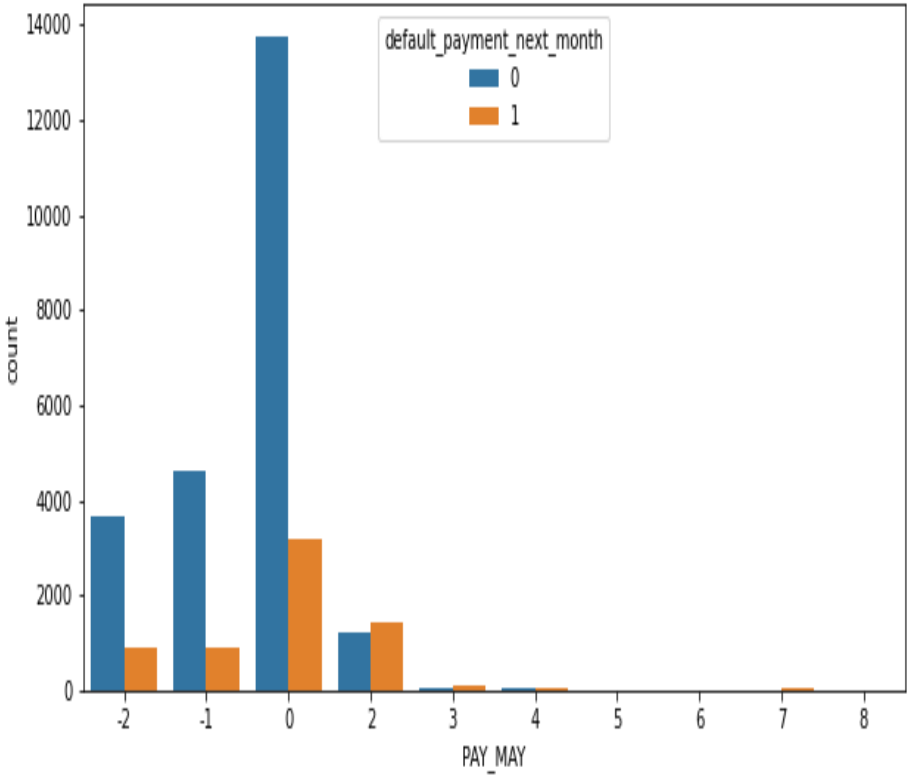


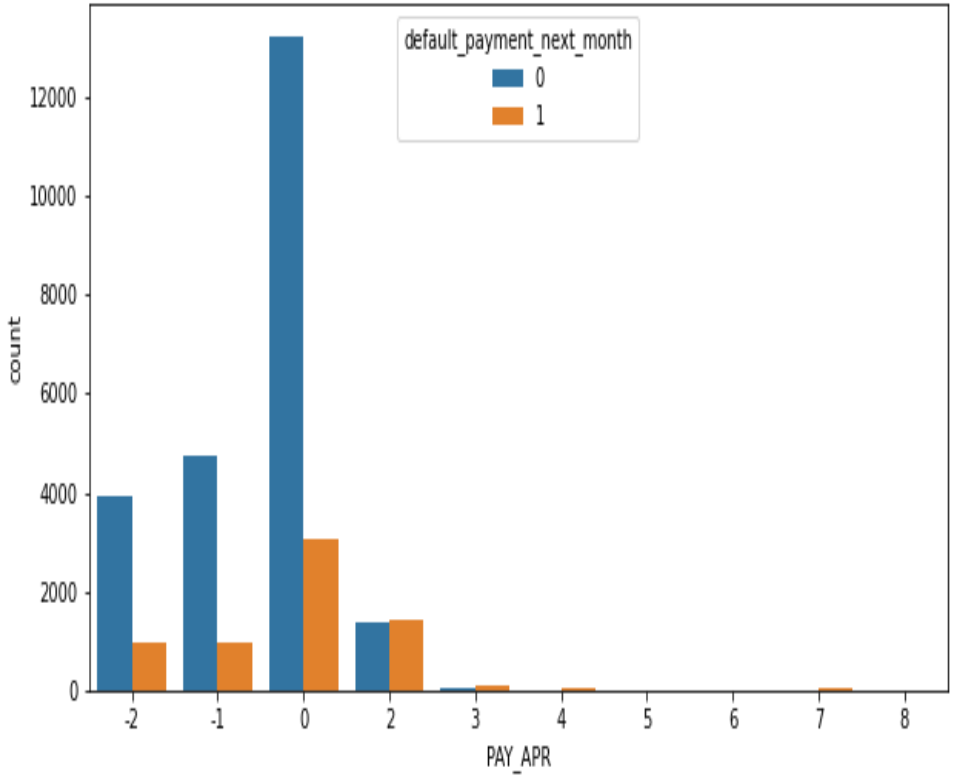




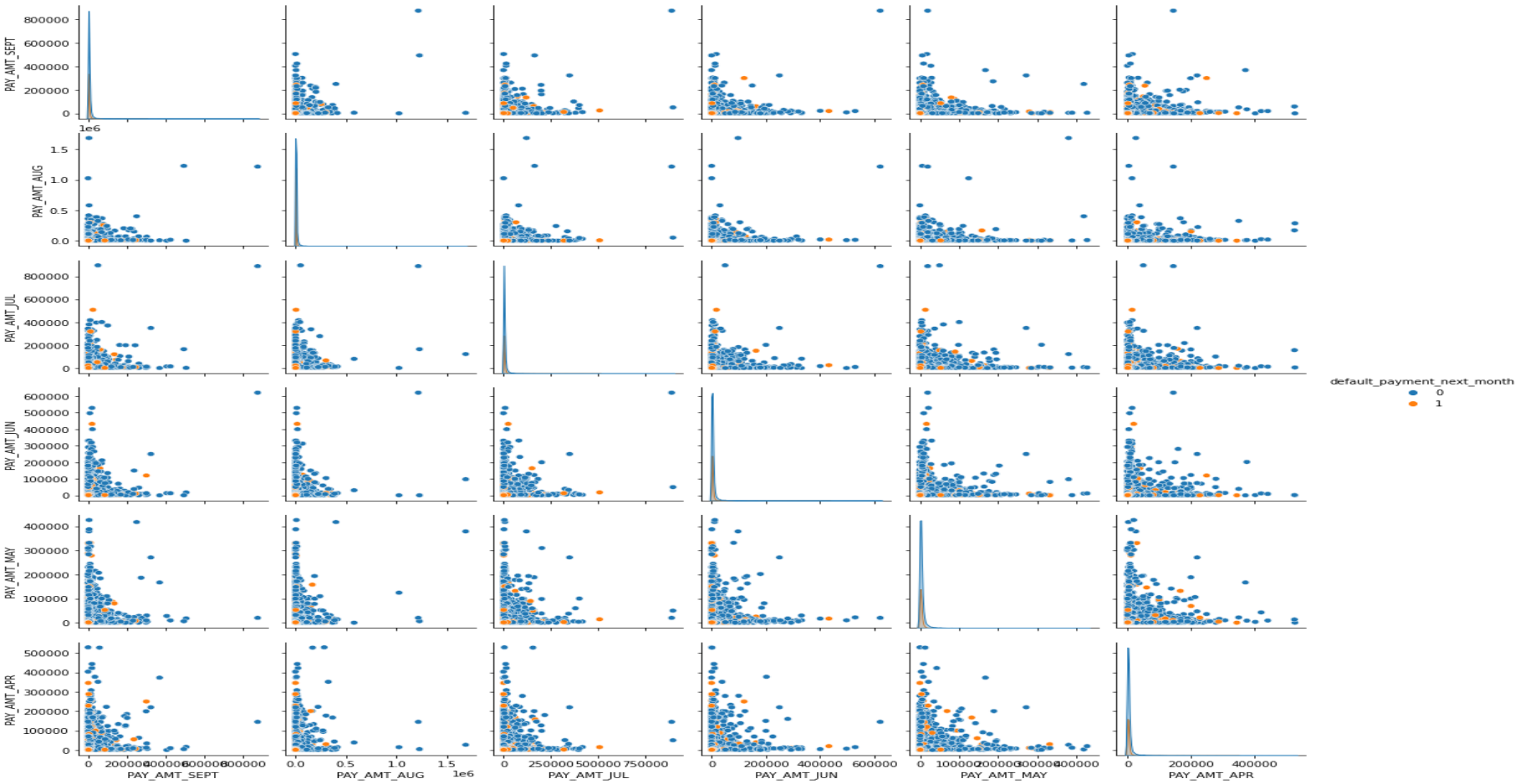


Exploratory Data Analysis – Univariate Analysis

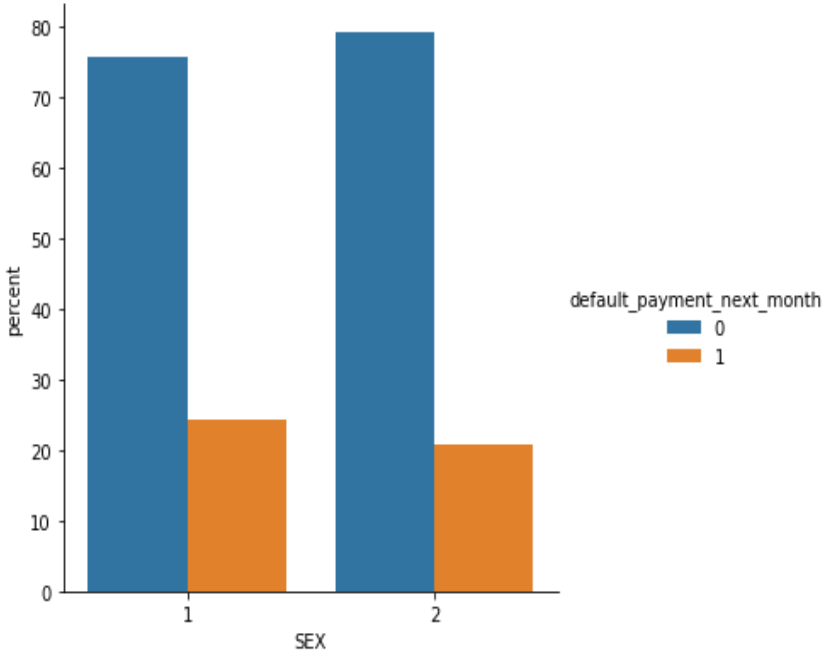


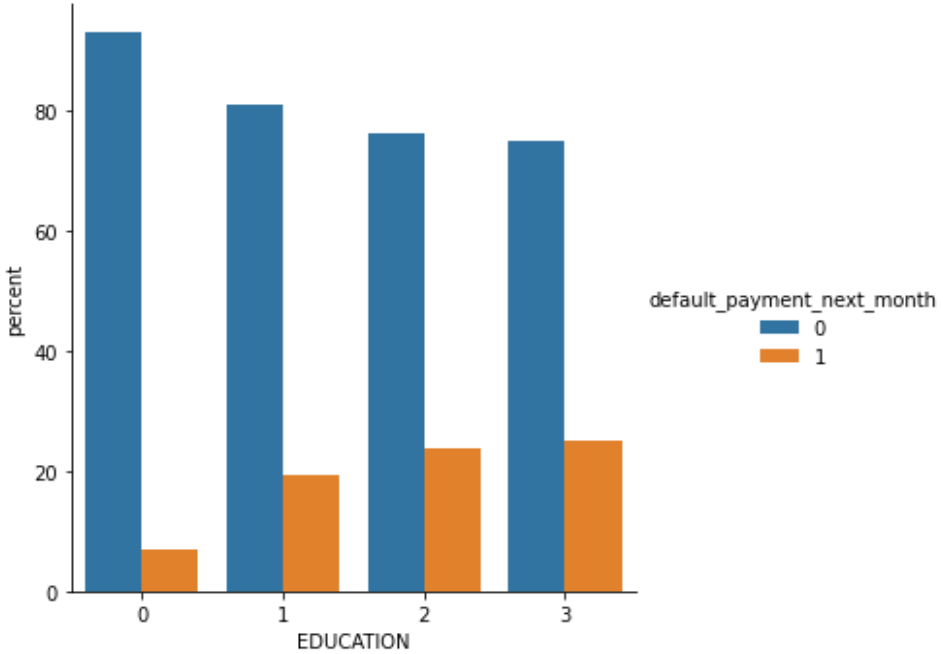


EDA - Paid Amount

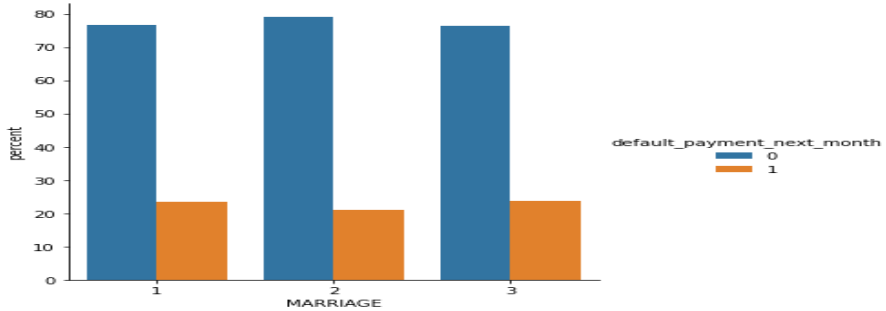


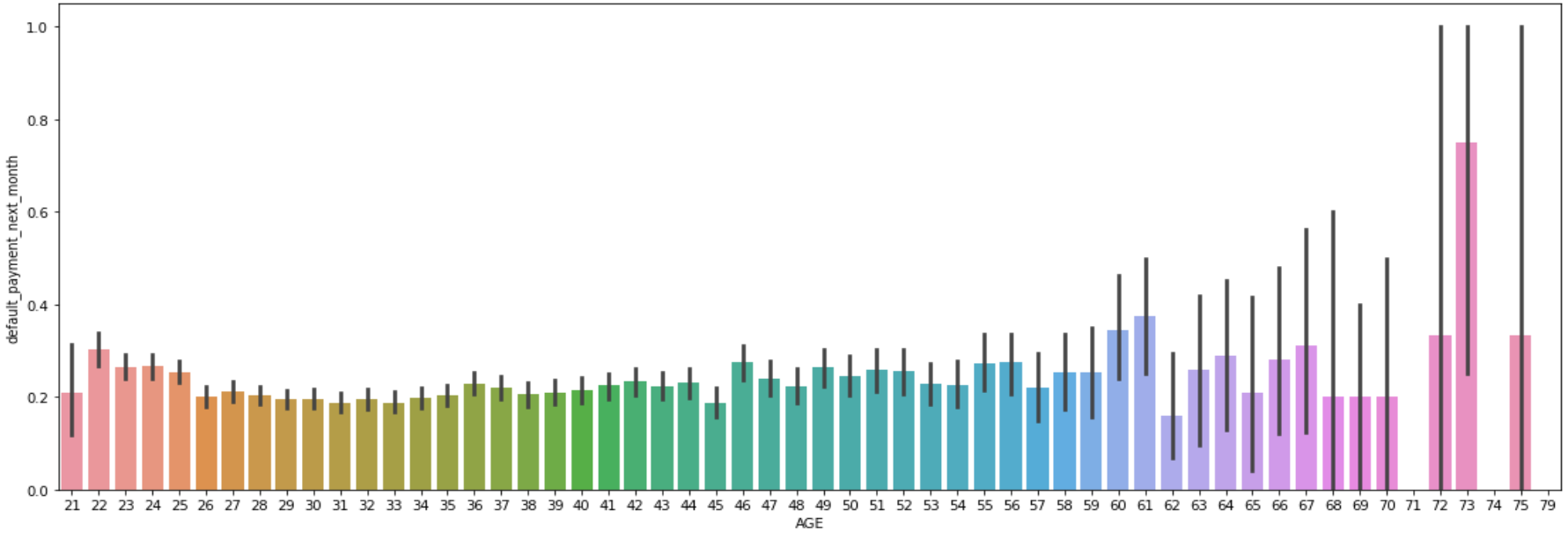
Exploratory Data Analysis – Bivariate Analysis



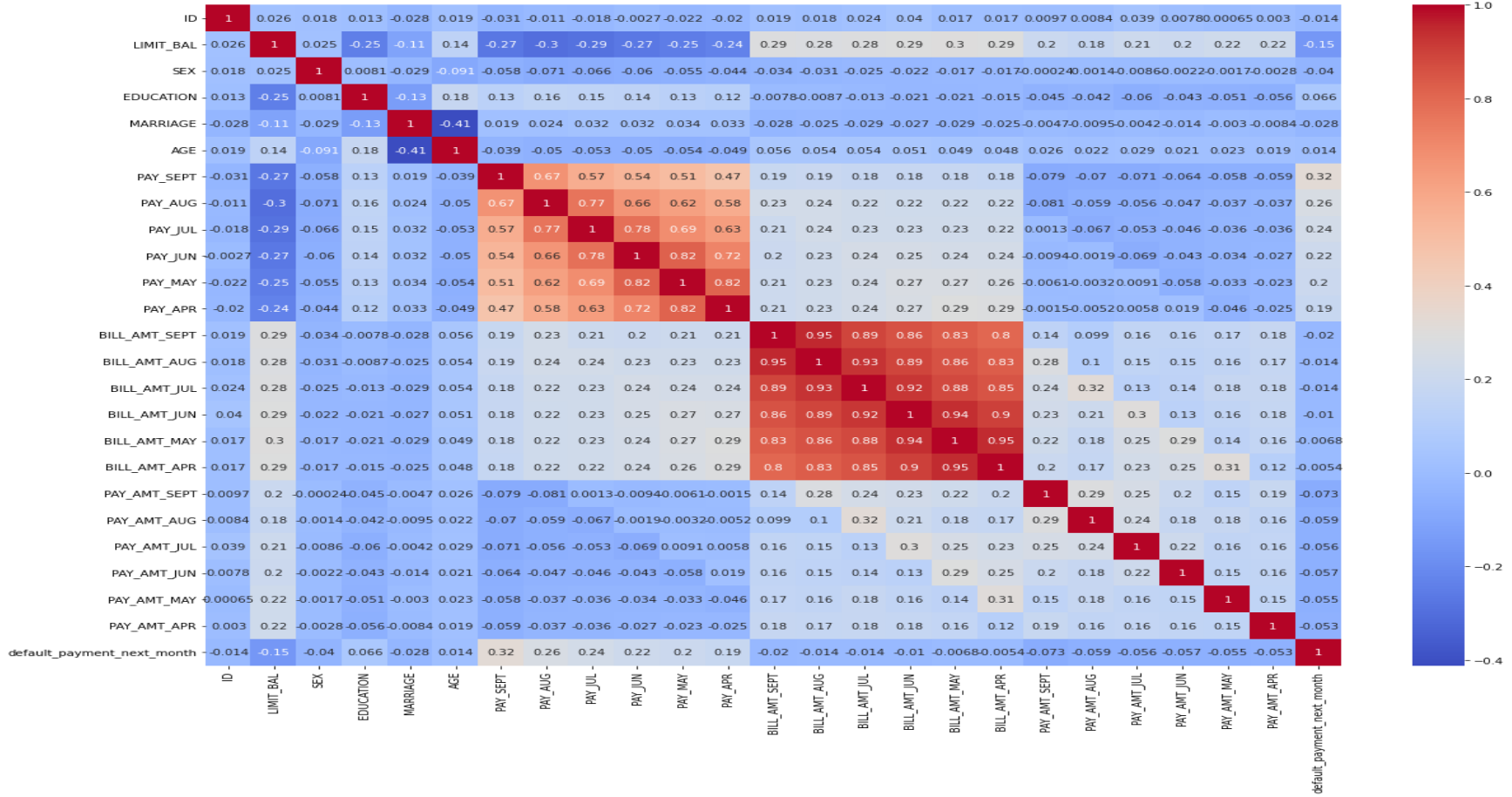


Exploratory Data Analysis – Bivariate Analysis





EDA – Correlation Between Variable



SMOTE

SMOTE (Synthetic Minority Oversampling Technique) – Oversampling is one of the most commonly used oversampling methods to solve the imbalance problem. It aims to balance class distribution by randomly increasing minority class examples by replicating them.

- **In our data set we have Imbalanced Data Distribution in our dependent variable, it generally happens when observations in one of the class are much higher i.e not defaulter or lower than the other classes i.e defaulter.**
- **As Machine Learning algorithms tend to increase accuracy by reducing the error, they do not consider the class distribution.**
- **Standard ML techniques such as Decision Tree and Logistic Regression have a bias towards the majority class, and they tend to ignore the minority class. They tend only to predict the majority class, hence, having major misclassification of the minority class in comparison with the majority class. In more technical words, if we have imbalanced data distribution in our dataset then our model becomes more prone to the case when the minority class has a negligible or very lesser recall.**

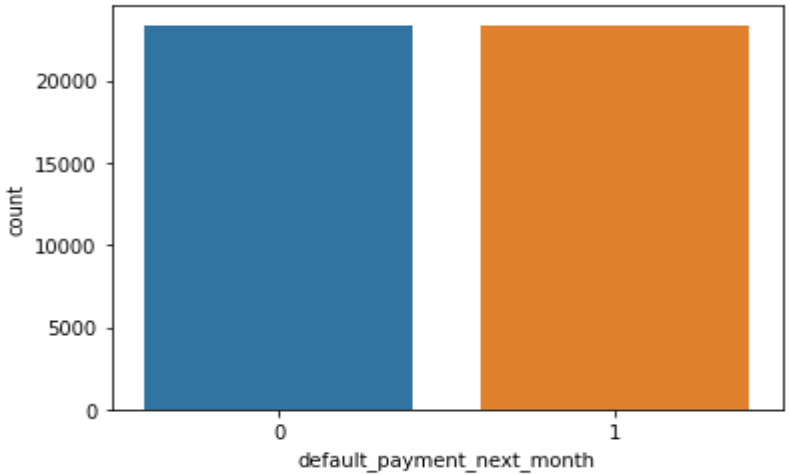
```
#import the module
from imblearn.over_sampling import SMOTE

smote = SMOTE()

# fit predictor and target variable
x_smote, y_smote = smote.fit_resample(credit_df.iloc[:,0:-1], credit_df['default_payment_next_month'])

print('Original dataset shape', len(credit_df))
print('Resampled dataset shape', len(y_smote))
```

```
Original dataset shape 30000
Resampled dataset shape 46728
```



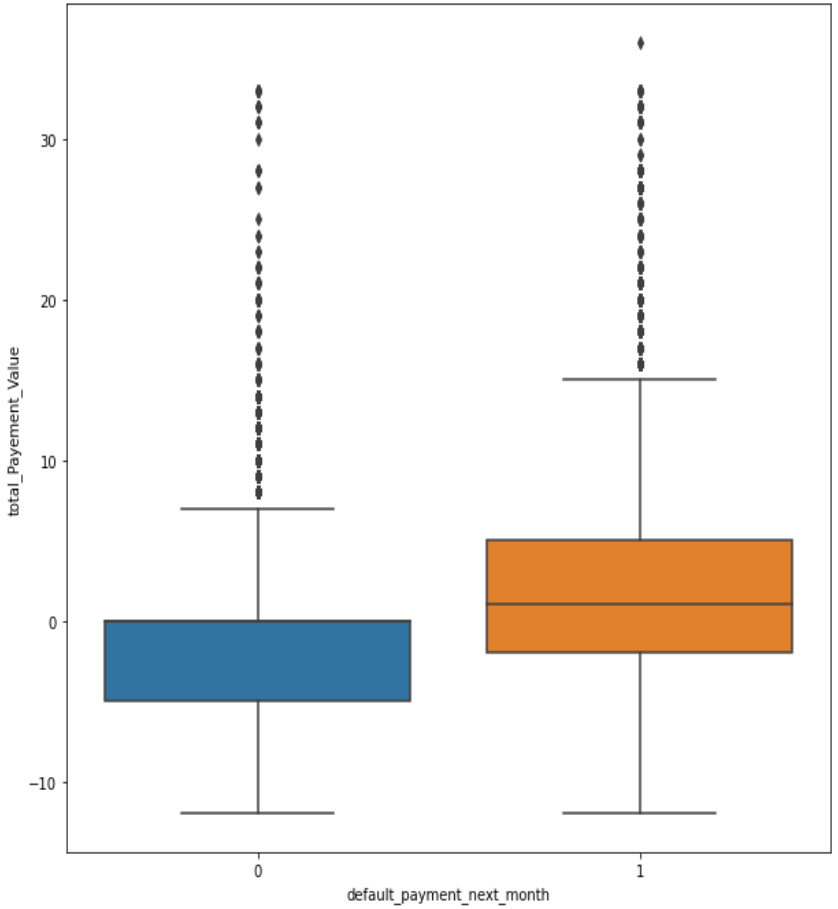
Feature Engineering

Feature engineering is the process that takes raw data and transforms it into features that can be used to create a predictive model using machine learning or statistical modeling, such as deep learning.

The intention of feature engineering is to achieve two primary goals:

- **Preparing an input dataset that is compatible with and best fits the machine learning algorithm.**
- **Improving the performance of machine learning models.**

When feature engineering processes are executed well, the resulting dataset will be optimal and contain all the essential factors that bear an impact on the business problem. These datasets in turn result in best possible predictive models and most beneficial insights.




```
#check the mean to calculate the correletion
credit_df_copy.groupby('default_payment_next_month')['Dues'].mean()
```

```
default_payment_next_month
0    187742.051532
1    195482.891243
Name: Dues, dtype: float64
```

```
credit_df_copy.replace({'SEX': {1 : 'MALE', 2 : 'FEMALE'},
                        'EDUCATION' : {1 : 'graduate school', 2 : 'university', 3 : 'high school', 0 : 'others'},
                        'MARRIAGE' : {1 : 'married', 2 : 'single', 3 : 'others'}}, inplace = True)
```

One Hot Encoding

- One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction.
- One Hot Encoding is used to convert numerical categorical variables into binary vectors. Before implementing this algorithm. Make sure the categorical values must be label encoded as one hot encoding takes only numerical categorical values.
- In our model we perform one hot encoding on 'EDUCATION','MARRIAGE','PAY_SEPT','PAY_AUG','PAY_JUL','PAY_JUN','PAY_MAY', 'PAY_APR'.

```
# LABEL ENCODING FOR SEX
encoders_nums = {
                "SEX":{"FEMALE": 0, "MALE": 1}
}
credit_df_copy = credit_df_copy.replace(encoders_nums)
```

```
credit_df_copy.drop('ID',axis = 1, inplace = True)
```

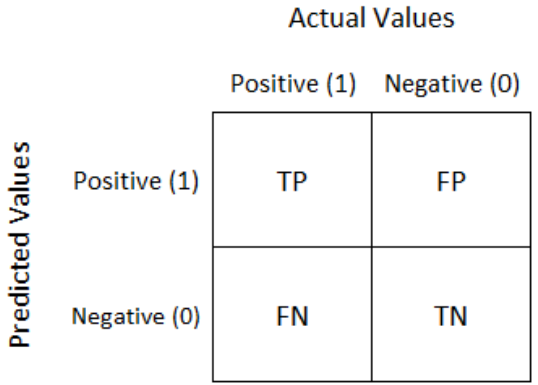
```
credit_df_copy.columns
```

```
Index(['LIMIT_BAL', 'SEX', 'AGE', 'BILL_AMT_SEPT', 'BILL_AMT_AUG',
       'BILL_AMT_JUL', 'BILL_AMT_JUN', 'BILL_AMT_MAY', 'BILL_AMT_APR',
       'PAY_AMT_SEPT', 'PAY_AMT_AUG', 'PAY_AMT_JUL', 'PAY_AMT_JUN',
       'PAY_AMT_MAY', 'PAY_AMT_APR', 'default_payment_next_month',
       'total_Payement_Value', 'Dues', 'EDUCATION_graduate school',
       'EDUCATION_high school', 'EDUCATION_others', 'EDUCATION_university',
       'MARRIAGE_married', 'MARRIAGE_others', 'MARRIAGE_single', 'PAY_SEPT_-1',
       'PAY_SEPT_0', 'PAY_SEPT_1', 'PAY_SEPT_2', 'PAY_SEPT_3', 'PAY_SEPT_4',
       'PAY_SEPT_5', 'PAY_SEPT_6', 'PAY_SEPT_7', 'PAY_SEPT_8', 'PAY_AUG_-1',
       'PAY_AUG_0', 'PAY_AUG_1', 'PAY_AUG_2', 'PAY_AUG_3', 'PAY_AUG_4',
       'PAY_AUG_5', 'PAY_AUG_6', 'PAY_AUG_7', 'PAY_AUG_8', 'PAY_JUL_-1',
       'PAY_JUL_0', 'PAY_JUL_1', 'PAY_JUL_2', 'PAY_JUL_3', 'PAY_JUL_4',
       'PAY_JUL_5', 'PAY_JUL_6', 'PAY_JUL_7', 'PAY_JUL_8', 'PAY_JUN_-1',
       'PAY_JUN_0', 'PAY_JUN_1', 'PAY_JUN_2', 'PAY_JUN_3', 'PAY_JUN_4',
       'PAY_JUN_5', 'PAY_JUN_6', 'PAY_JUN_7', 'PAY_JUN_8', 'PAY_MAY_-1',
       'PAY_MAY_0', 'PAY_MAY_1', 'PAY_MAY_2', 'PAY_MAY_3', 'PAY_MAY_4',
       'PAY_MAY_5', 'PAY_MAY_6', 'PAY_MAY_7', 'PAY_MAY_8', 'PAY_APR_-1',
       'PAY_APR_0', 'PAY_APR_1', 'PAY_APR_2', 'PAY_APR_3', 'PAY_APR_4',
       'PAY_APR_5', 'PAY_APR_6', 'PAY_APR_7', 'PAY_APR_8'],
      dtype='object')
```

Evaluation Metrics & Estimator

- **Precision** is a good metric to use when the costs of false positive(FP) is high.
- **Precision** = TP / (TP + FP)
- **Recall** is a good metric to use when the cost associated with false negative(FN) is high.
- **Recall** = TP / (TP + FN)
- **F1-score** is a weighted average of precision and recall. Thus, it considers FP and FN. This metric is very useful when we have uneven class distribution, as it seeks a balance between precision and recall. F1 score is defined as the harmonic mean between precision and recall. It is used as a statistical measure to rate performance. This means a statistical measure of the accuracy of a test or an individual.
- F1-score = 2 * (precision * recall) / (precision + recall)
- **Confusion Matrix** is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.

- True Positive: This combination tells us how many times a model correctly classifies a positive sample as Positive?
- False Negative: This combination tells us how many times a model incorrectly classifies a positive sample as Negative?
- False Positive: This combination tells us how many times a model incorrectly classifies a negative sample as Positive?
- True Negative: This combination tells us how many times a model correctly classifies a negative sample as Negative?



Evaluation Metrics & Estimator

- **AUC-ROC curve:** The Receiver Operator Characteristic (ROC) curve is an evaluation metric for binary classification problems. It is a probability curve that plots the TPR against FPR at various threshold values and essentially separates the ‘signal’ from the ‘noise’. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.
- **GridSearchCV:** GridSearchCV is a library function that is a member of sklearn's model_selection package. It helps to loop through predefined hyperparameters and fit our estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.
- **GridSearchcv** classification is an important step in classification machine learning projects for model select and hyper Parameter Optimization. It takes a dictionary that describes the parameters that could be tried on a model to train it. The grid of parameters is defined as a dictionary, where the keys are the parameters and the values are the settings to be tested.
- One of the great things about **GridSearchCV** is that it is a meta-estimator. It takes an estimator like SVC and creates a new estimator, that behaves exactly the same – in this case, like a classifier. You should add refit=True and choose verbose to whatever number you want, the higher the number, the more verbose (verbose just means the text output describing the process).

Logistic Regression

AI

- Logistic regression is a classification algorithm that predicts the probability of an outcome that can only have two values (i.e. a dichotomy). Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1.
- Logistic regression is an excellent tool to know for classification problems, which are problems where the output value that we wish to predict only takes on only a small number of discrete values. Here we'll focus on the binary classification problem, where the output can take on only two distinct classes.
- In Logistic Regression, the log-odds of a categorical response being "true" (1) is modeled as a linear combination of the features:

$$\log\left(\frac{p}{1-p}\right) = w_0 + w_1 x_1, \ldots, w_j x_j$$
$$= w^T x$$

where:

w0 is the intercept term, and w1 to wj represents the parameters for all the other features (a total of j features).

By convention of we can assume that x0=1, so that we can re-write the whole thing using the matrix notation wTx.

This is called the logit function. The equation can be re-arranged into the logistic function:

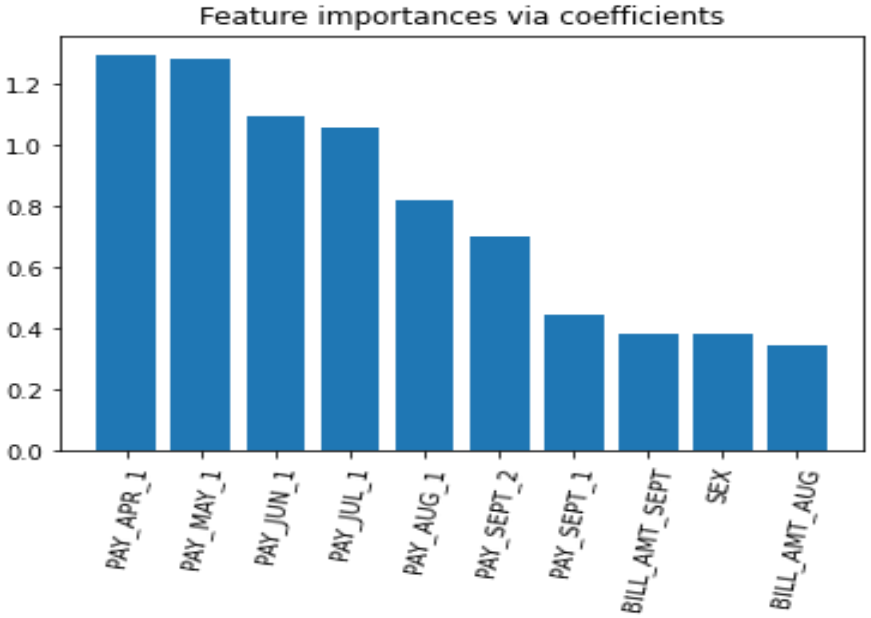
$$p = \frac{e^{w^T x}}{1 + e^{w^T x}}$$

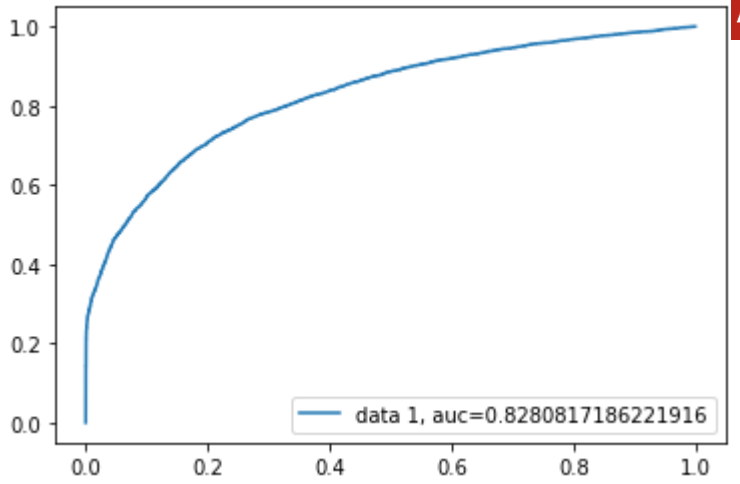
Or in the more commonly seen form:

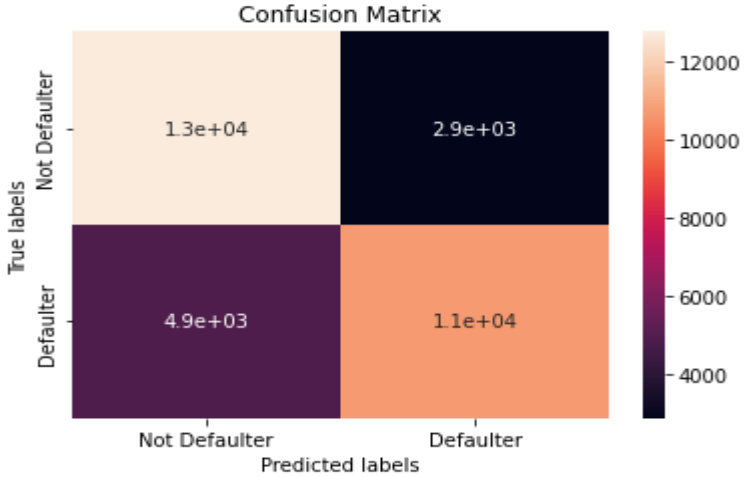
$$h_w(x) = \frac{1}{1 + e^{-w^T x}}$$

Logistic Regression

- The accuracy on test data is **0.7553984825886778**
- The precision on test data is **0.6936446173800259**
- The recall on test data is **0.7913583900562297**
- The f1 on test data is **0.7392867016864806**
- The roc_score on test data is **0.7593522874903104**





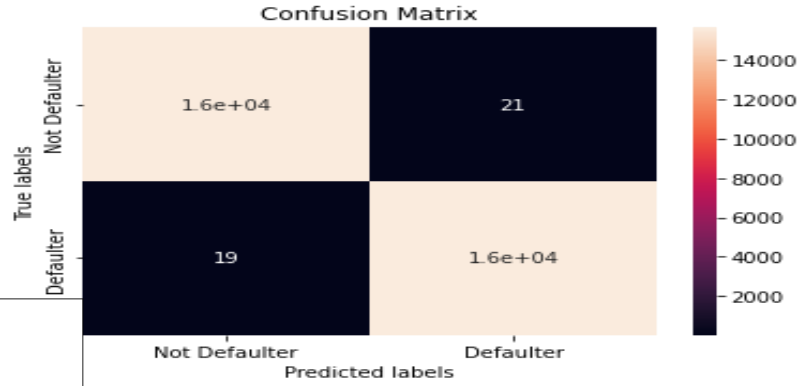


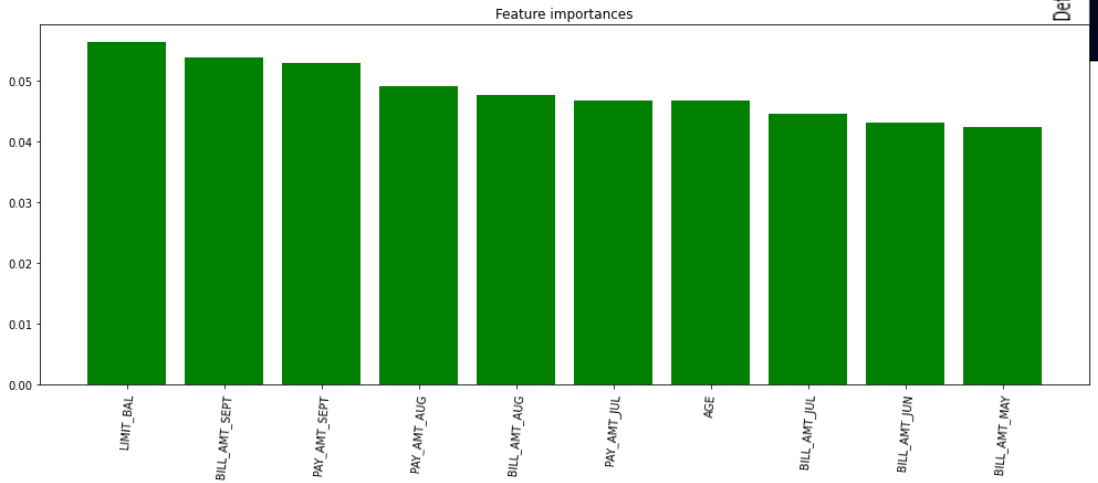
Random Forest Classifier

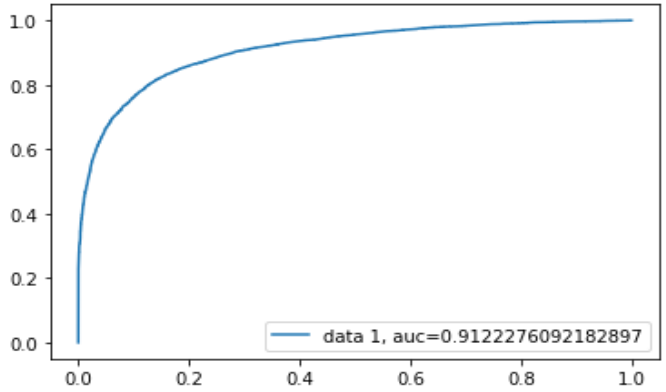
A random forest classifier. A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

GridSearchCV Evaluation

- The accuracy on test data is **0.8337332209324947**
- The precision on test data is **0.8020752269779508**
- The recall on test data is **0.856272500692329**
- The f1 on test data is **0.8282882400214305**
- The roc_score on test data is **0.8350761210621055**







Naïve Bayes Classifier

Naive Bayes models are a group of extremely fast and simple classification algorithms that are often suitable for very high-dimensional datasets. Because they are so fast and have so few tunable parameters, they end up being very useful as a quick-and-dirty baseline for a classification problem.

The posterior probability can be written as : $P(Y \mid X) = P(y)P(x_1|y)P(x_2|y,x_1)\ldots P(x_n|y,x_1,\ldots,x_{n-1})$

Assuming all the X are conditionally independent

$$P(Y=1|X) = \frac{P(y=1)P(x_1|y=1)\ldots P(x_n|y=1)}{P(X)}$$

```
[ ]  y_pred = model_gnb.predict(X_test)
     #To measure Accuracy
     from sklearn.metrics import accuracy_score
     print(f'Model accuracy score: {100*accuracy_score(y_test, y_pred):0.2f}%')
     print()
```

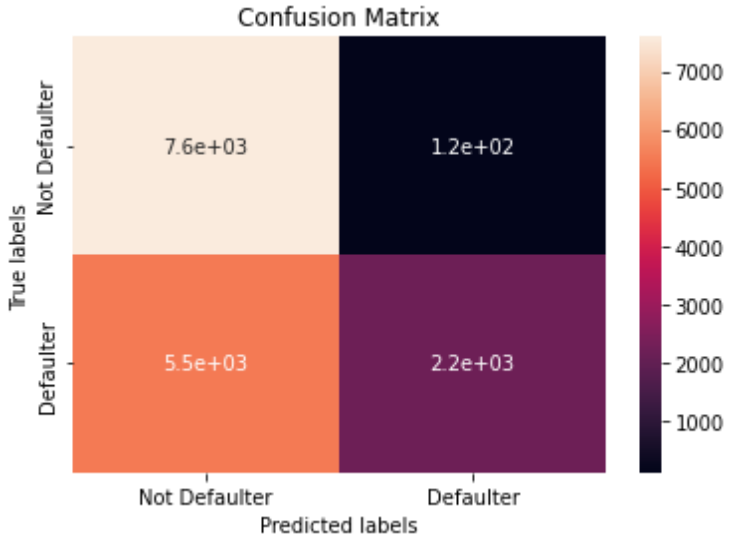
```
Model accuracy score: 63.45%
```

This model gives 63.5% accuracy and we can improve the accuracy by using hyperparameter optimization method.

1) The probability of the hypothesis multiplies the prior probability of a hypothesis, given the training data.

2) The output is not only classification but also a probability distribution representing all classes as well.

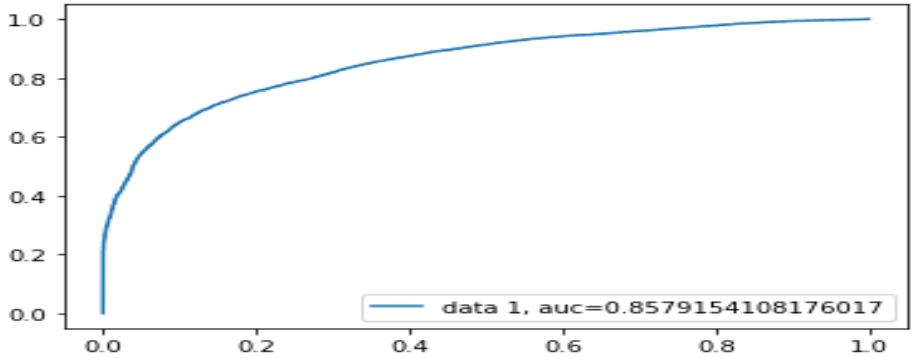
3) With each data instance, the prior and the likelihood can be updated incrementally.

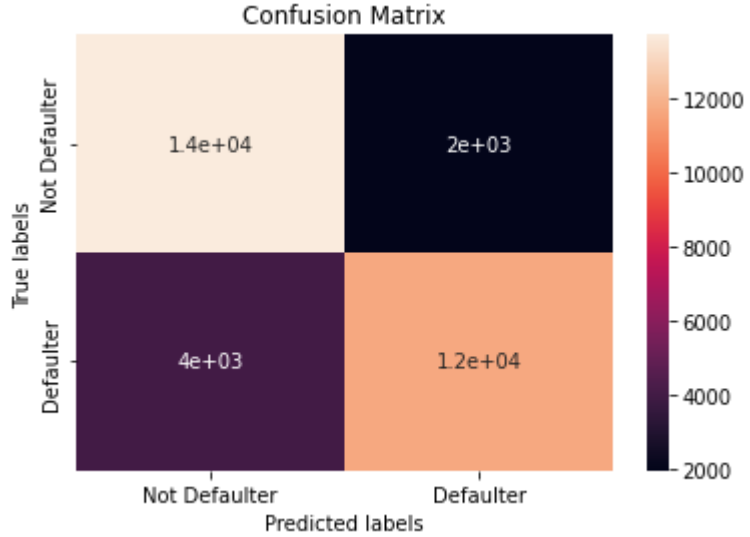


Support Vector Machine (SVM) or Support Vector Classifier (SVC)

- **The Linear Support Vector Classifier (SVC) method applies a linear kernel function to perform classification and it performs well with a large number of samples. If we compare it with the SVC model, the Linear SVC has additional parameters such as penalty normalization which applies 'L1' or 'L2' and loss function.**
- **Support Vector Machines (SVMs in short) are machine learning algorithms that are used for classification and regression purposes. SVMs are one of the powerful machine learning algorithms for classification, regression and outlier detection purposes.**
- **SVMs can be used for linear classification purposes known as Support Vector Classifiers (SVCs). These are generally discriminative models. Discriminative models, also called conditional models, tend to learn the boundary between classes/labels in a dataset.**

- The accuracy on test data is **0.766746644186499**
- The precision on test data is **0.6900129701686122**
- The recall on test data is **0.8150758388233492**
- The f1 on test data is **0.7473484582426073**
- The roc_score on test data is **0.7731776765513193**

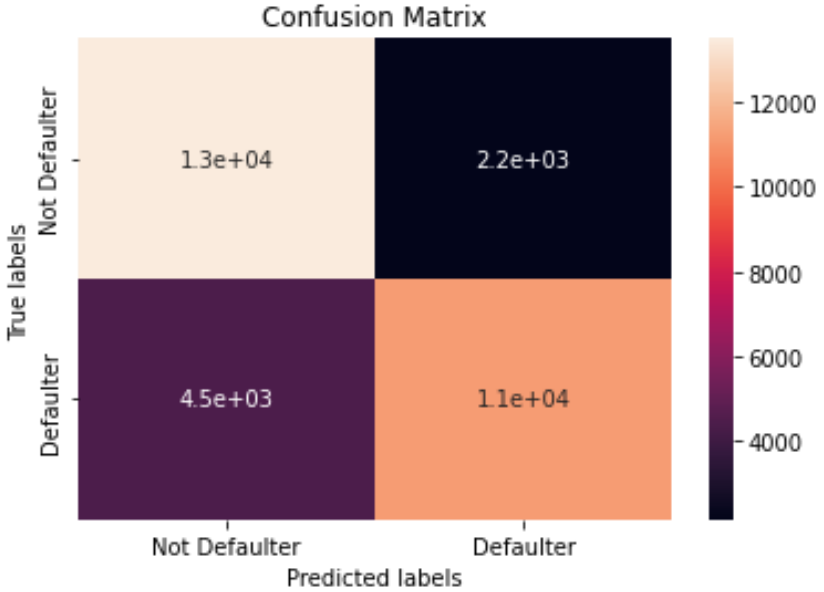




XGBoost Classifier

- XGBoost provides a wrapper class to allow models to be treated like classifiers or regressors in the scikit-learn framework.
- The XGBoost model for classification is called XGBClassifier. We can create and and fit it to our training dataset. Models are fit using the scikit-learn API and the model.fit() function.
- **Unlike many other algorithms, XGBoost is an ensemble learning algorithm meaning that it combines the results of many models, called base learners to make a prediction.**
- **Just like in Random Forests, XGBoost uses Decision Trees as base learners**

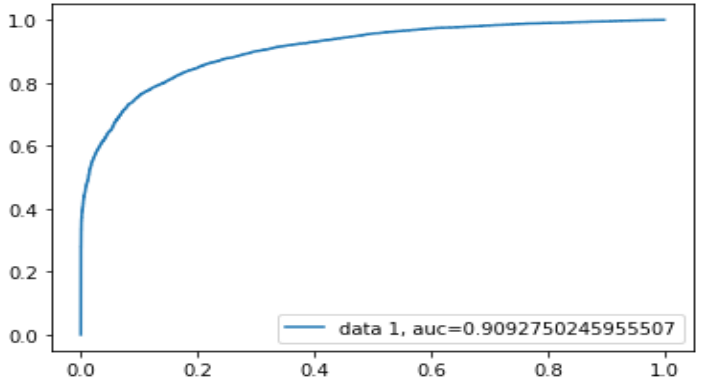
- The accuracy on test data is **0.7531288502691136**
- The precision on test data is **0.677561608300908**
- The recall on test data is **0.79816653934301**
- The f1 on test data is **0.732935811995791**
- The roc_score on test data is **0.7590427108612301**

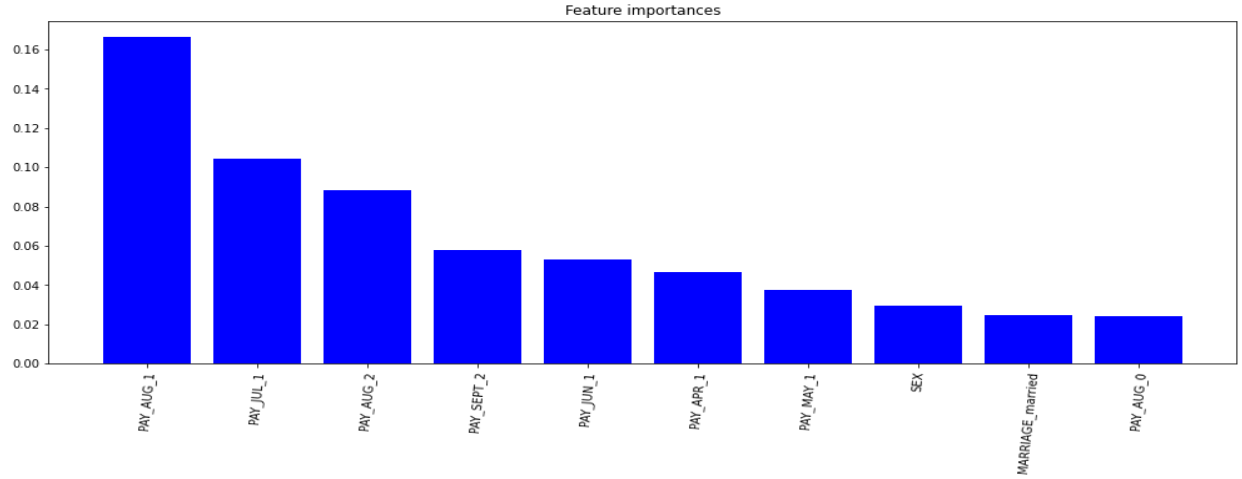


Machine Learning Modelling

7. Hyperparameter Tuning

- Hyperparameters are crucial as they control the overall behavior of a machine learning model. The ultimate goal is to find an optimal combination of hyperparameters that minimizes a predefined loss function to give better results.
- Hyperparameters are crucial as they control the overall behavior of a machine learning model. The ultimate goal is to find an optimal combination of hyperparameters that minimizes a predefined loss function to give better results.





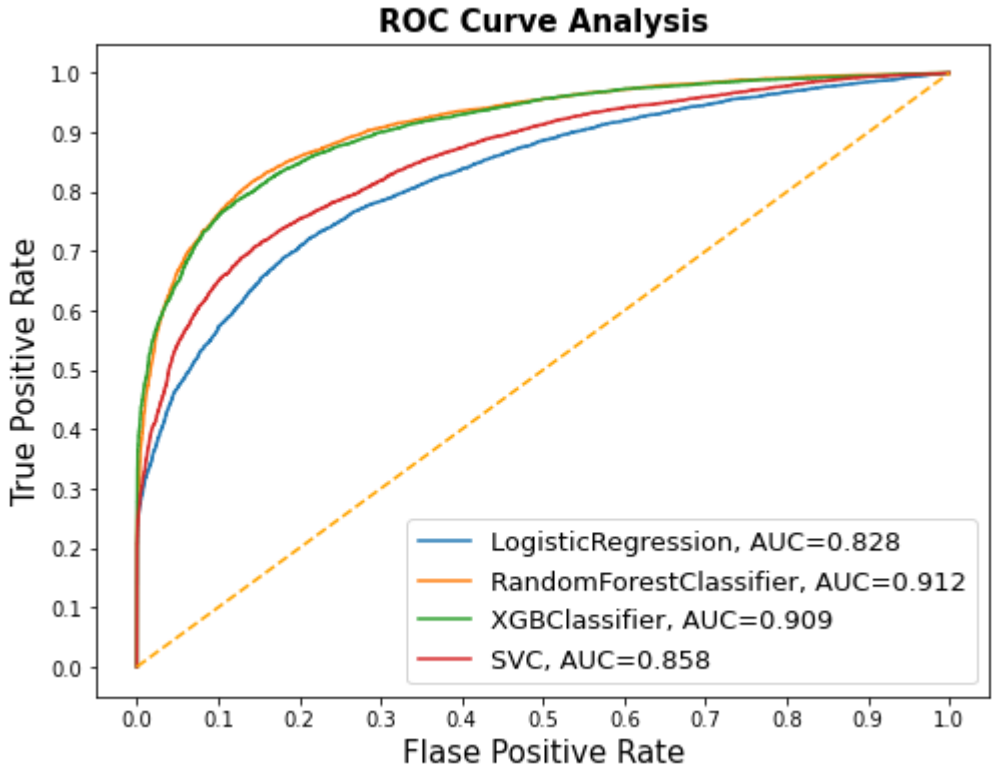
- The accuracy on test data is **0.8299721159457882**
- The precision on test data is **0.7932555123216601**
- The recall on test data is **0.8561030235162373**
- The f1 on test data is **0.8234818903998922**
- The roc_score on test data is **0.8317605850938684**

Evaluating Models

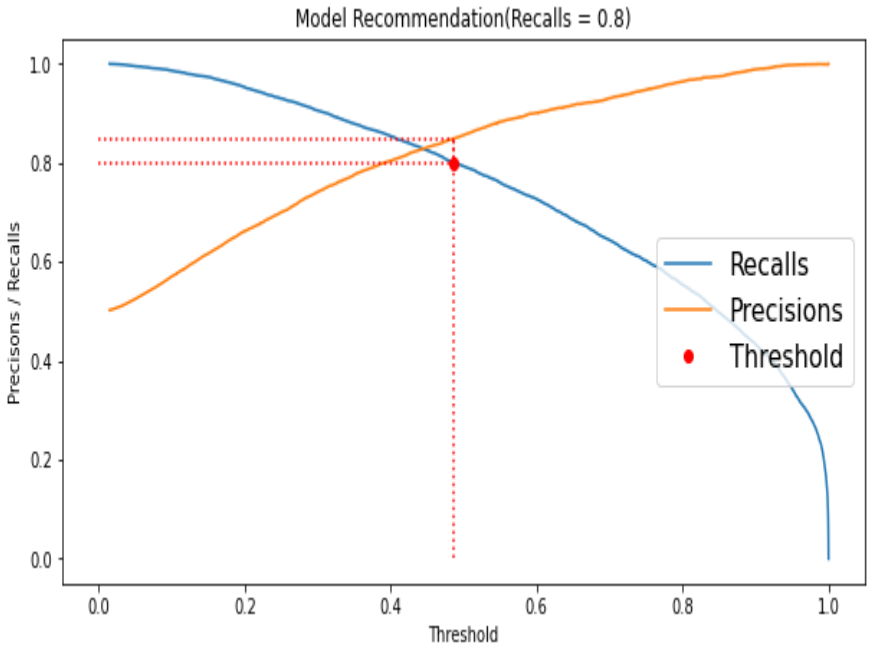
	Classifier	Train Accuracy	Test Accuracy	Precision Score	Recall Score	F1 Score
0	Logistic Regression	0.753921	0.754037	0.690791	0.790794	0.737418
1	SVC	0.809404	0.780883	0.715305	0.823257	0.765494
2	Random Forest CLf	0.998722	0.836976	0.805966	0.859237	0.831749
3	Xgboost Clf	0.916185	0.829972	0.793256	0.856103	0.823482

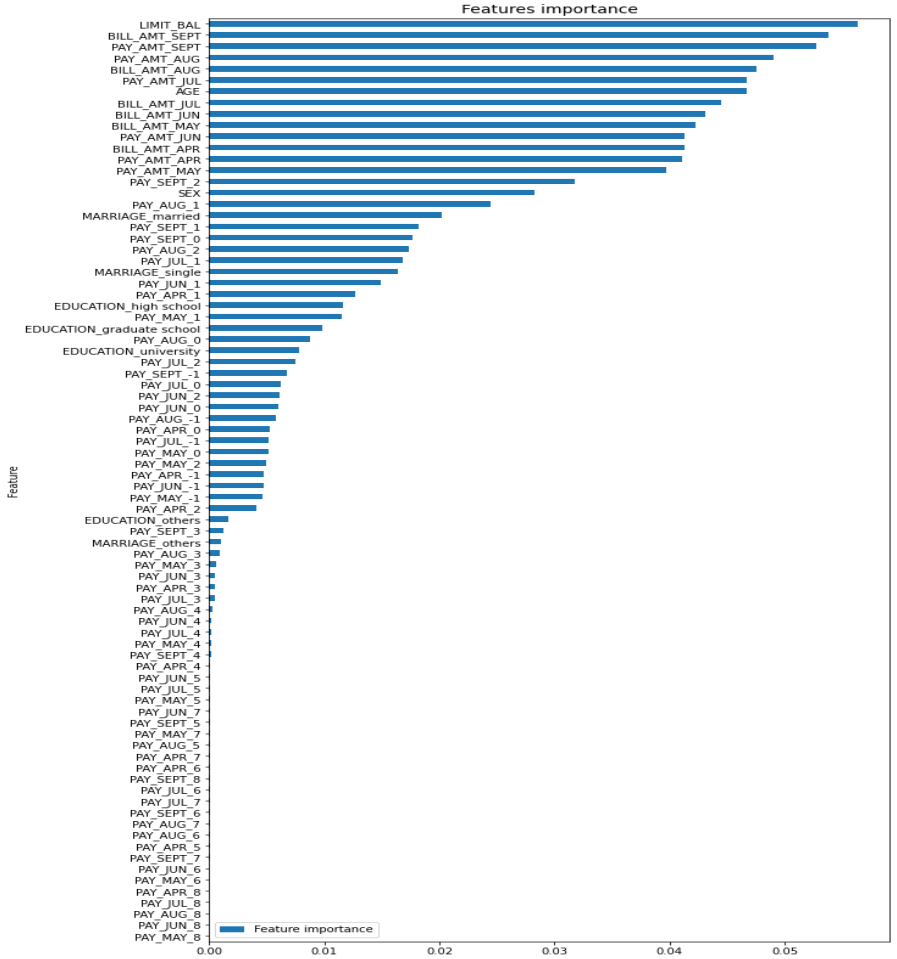
classifiers	fpr	tpr	auc
LogisticRegression	[0.0, 0.0, 0.0, 0.00012968486577616392, 0.0001...	[0.0, 0.00012970168612191958, 0.10894941634241...	0.828082
RandomForestClassifier	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.036575875486381325, 0.037224383916990...	0.912228
XGBClassifier	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.00012970168612191958, 0.00324254215304...	0.909275
SVC	[0.0, 0.0, 0.0, 0.00012968486577616392, 0.0001...	[0.0, 0.00012970168612191958, 0.13852140077821...	0.857915

Evaluating Models



Recommending Model





Conclusion

XGBoost model has the highest recall, if the business cares recall the most, then this model is the best candidate. If the balance of recall and precision is the most important metric, then Random Forest is the ideal model. Since Random Forest has slightly lower recall but much higher precision than Logistic Regression, I would recommend Random Forest.

- Data categorical variables had minority classes which were added to their closest majority class
- There were not huge gap but female clients tended to default the most.
- Labels of the data were imbalanced and had a significant difference.
- Gradient boost gave the highest accuracy of 82% on test dataset.
- Repayment in the month of September tended to be the most important feature for our machine learning model.
- The best accuracy is obtained for the Random forest and XGBoost classifier.
- In general, all models have comparable accuracy. Nevertheless, because the classes are imbalanced (the proportion of non-default credit cards is higher than default) this metric is misleading.
- Furthermore, accuracy does not consider the rate of false positives (non-default credits cards that were predicted as default) and false negatives (default credit cards that were incorrectly predicted as non-default).
- Both cases have negative impact on the bank, since false positives leads to unsatisfied customers and false negatives leads to financial loss.
- We can see that XGBoost Classifier having Recall, F1-score, and ROC Score values equals 85%, 82%, and 91% and Random forest Classifier having Recall, F1-score, and ROC Score values equals 86%, 83%, and 91%.
- XGBoost Classifier and Decision Tree Classifier are giving us the best Recall, F1-score, and ROC Score among other algorithms. We can conclude that these two algorithms are the best to predict whether the credit card is default or not default according to our analysis.

References

1. Stacksoverflow.com
2. GeeksforGeeks
3. towardsdatascience.com/
4. Kaggle.com
5. analyticsvidhya.com
6. scikit-learn.org