BT2103 Project:
Prediction of Customer's Default & Non-Default Payment

Done By:

Yang JiAn: A0233196A

Yuen Jia Jie: A0233224U

Huang Renzhou: A0239579H

Diao Zhuoxuan: A0239446W

Number of Pages : 20 (excluding cover page, content page and appendix)
Page 3 to 22 (Section 1: Overview of Project to Section 10: Room for Improvements)

BT2103 Project Table of Contents

1) Overview Of Project **3**
- The Problem Statement 3
- The Dataset used 3

2) Exploratory Data Analysis **3**
- Data Cleaning 4
- Explore Data with Visualisation 5

3) Data Pre-Processing part 1 **7**
- Data Encoding for Age Variable 7
- Data transformation via Min-Max 8

4) Model Testing Before Feature Selection **8**

5) Feature Selection **9**
- Filter Method 9
- Wrapper Method 12
- Embedded Method 13
- Final Features Selected for Models Predictions 13

6) Model Testing Before Balancing the dataset **14**

7) Data Pre-Processing part 2 **15**
- Synthetic Minority Oversampling Technique (SMOTE) 15

8) Model Selection (Machine Learning Model Used) **15**
- Hyperparameter Tuning for all the models 15
- Model 1: Support Vector Machine (SVM) 16
- Model 2: Multi-Layer Perceptron Model (MLP Classifier) 16
- Model 3: K-Nearest-Neighbours (KNN) 17
- Model 4: Logistic Regression 18
- Model 5: Random Forest Classification 19

9) Model Evaluation **19**

10) Room for Improvement **22**
- Problem with the Data Set 22
- Page limit constraints 22

11) Appendix **23**
- Exploratory Data Analysis 23
- Model Selection 25

Annex **27**

1) Overview Of Project

The Problem Statement

This project attempts to **predict BOTH credit card DEFAULT payment and NON-DEFAULT payment next month** using various prediction models.

This project is taking perspective from the credit risk management entity, whose goal is to set higher bars to default customers to reduce the risk of making losses and seek more clients with good credit. Hence, it is crucial to make correct classification and prediction on both default and non-default payments.

The Dataset used

This dataset contains 30,000 credit card holders' payment information retrieved from a Taiwanese bank. Each sample consists of 23 feature attributes and one target variable, "default payment next month".

2) Exploratory Data Analysis

In section 2, we try to discover patterns, and spot anomalies, inconsistencies and missing values with the aid of summary statistics and graphical visualisations.

Summary statistics of the dataset

	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

Data Cleaning

Data Cleaning of Target Variable

- Renaming of 'default payment next month' into 'DEFAULT'
    - For the ease of viewing since the independent variable name 'default payment next month' is too lengthy. Also, it contains space in the middle, which might cause errors during coding.

Data Cleaning of Input Variables

- **Handling Education status**
    - 0: unknown
    - 4: others
    - 5: unknown
    - 6: unknown
    - Since all four statuses indicate an unclarity of education status, we can group them all as one group, assigning them a value of '4', suggesting others.
- **Handling Marriage status**
    - We have grouped the marriage status of ‘0’ with ‘3’ because ‘0’ was not defined.
- **Handling Pay status**
    - For the repayment status, we can see that there are ‘-2’, ‘-1’, ‘0’. Hence, we have subsumed them and assigned them a value of ‘0’, suggesting duly payment.
- **Renaming of variables**
    - PAY_0:
        - PAY_0: Status of Repayment in September 2005
        - PAY_2: Status of Repayment in August 2005
    - BILL_AMT1: Bill statement in September 2005
    - PAY_AMT1: Amount of previous payment in September 2005
    - Changed the name of 'PAY_0' to 'PAY_1' as it is intuitive to name the variable name in consecutive numbers if the month is consecutive.
    - Also, we changed the name of ‘PAY_0' to 'PAY_1' since it will be consistent with 'BILL_AMT1' and 'PAY_AMT1', which suggests the amount of the bill statement and amount of previous payment in September accordingly.

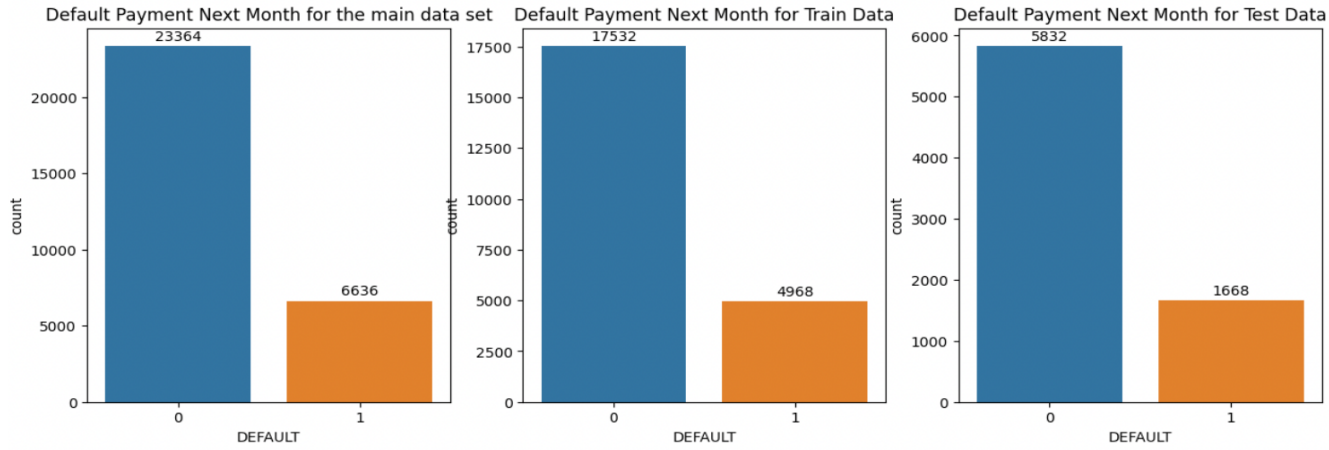
Missing Values

- There are no missing values in the dataset shown by the “count” column.

```
<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_1      30000 non-null  int64
 7   PAY_2      30000 non-null  int64
 8   PAY_3      30000 non-null  int64
 9   PAY_4      30000 non-null  int64
 10  PAY_5      30000 non-null  int64
 11  PAY_6      30000 non-null  int64
 12  BILL_AMT1  30000 non-null  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    30000 non-null  int64
dtypes: int64(25)
```

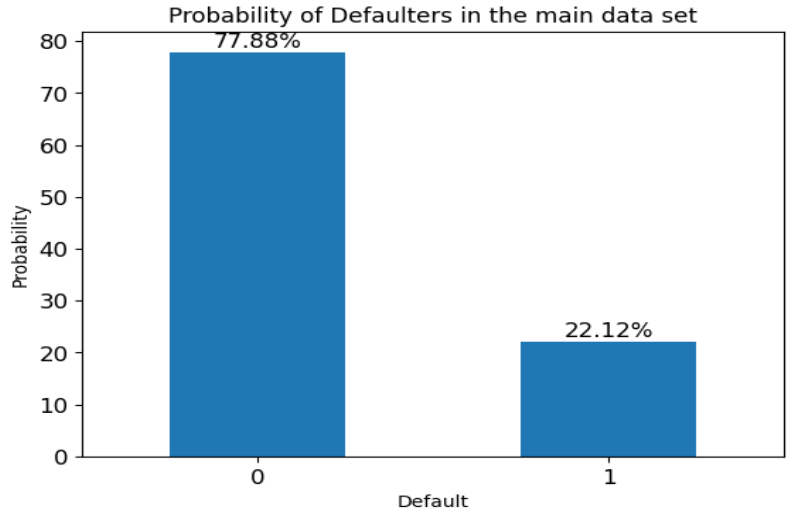
Explore Data with Visualisation

Check the composition of the target variable.



Firstly, we have divided the main data set into two subsets - train data to develop a model and a test data set to test our fitted models.

We have used the bar graph to check the proportion of default and non-default payments next month for three data sets – namely, the main data set, the train data set, and the test data set.



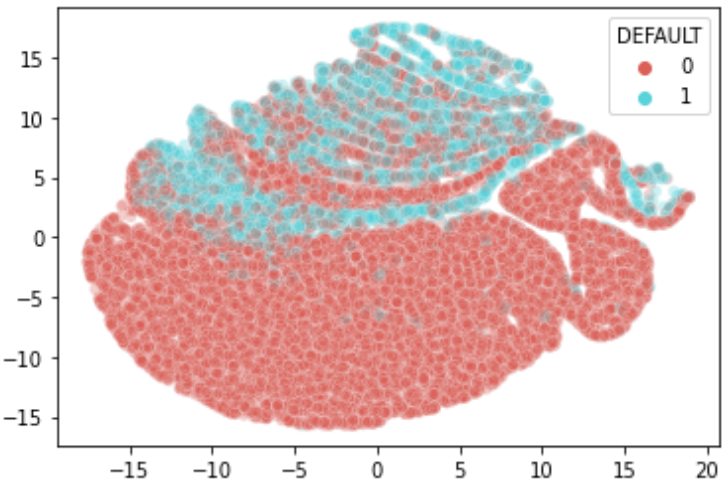
Insights:

By observation, we can conclude that the compositions of the number of default payments and non-default payments in main, train and test data are roughly the same. The percentage of default payments is around **77.9%**, while the percentage of non-fault payments is around **22.1%** for all three sets.

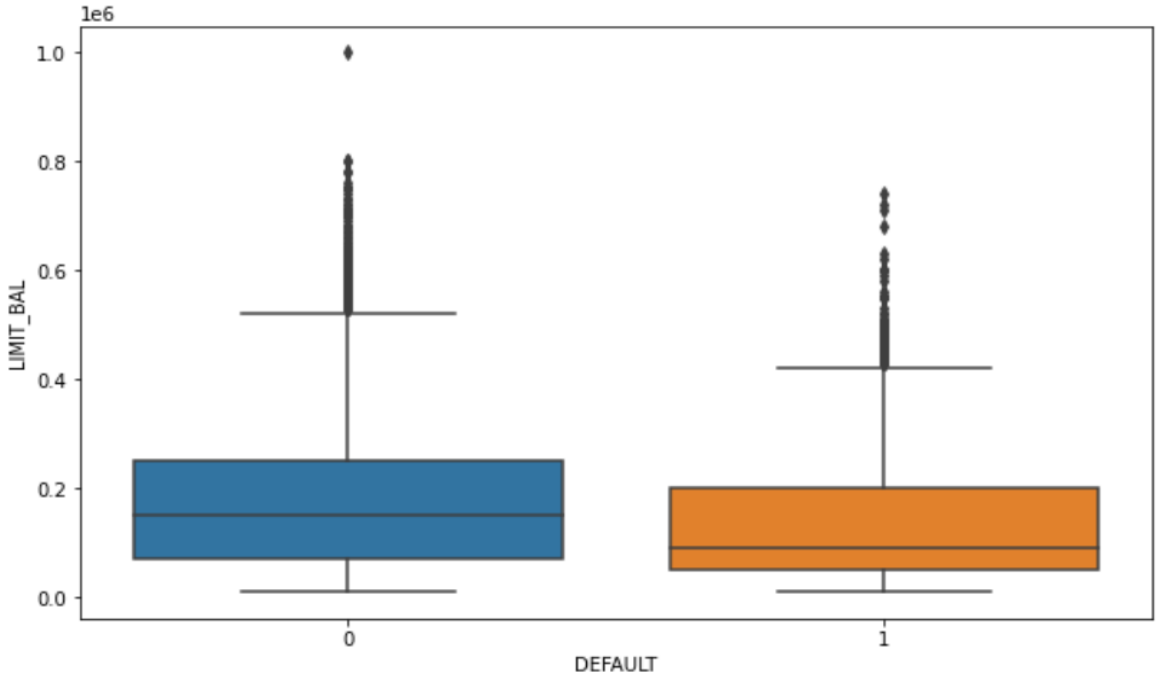
Check if target is separable using T-SNE plot

T-SNE helps us reduce the dataset's dimensionality and visualise it. From the plot, we can see that there is no clear distinct cluster. This shows that the target value in the dataset is not linearly separable.

Knowing that the target is not linearly separable, we can better pick the model and parameter to best predict our target value.



Boxplot for limit balance by default

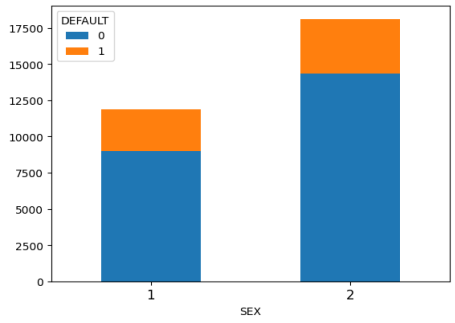


Insights

- From the boxplot, we observe that the limit balance of those who will default is slightly lower than those who will not default.
- As a result, we gain an understanding that limit balance may play a part in predicting of default payments

Check the composition of target variables in each of the categorical variables

Distribution of SEX for the main data set



SEX	DEFAULT	
1	0	75.832773
	1	24.167227
2	0	79.223719
	1	20.776281

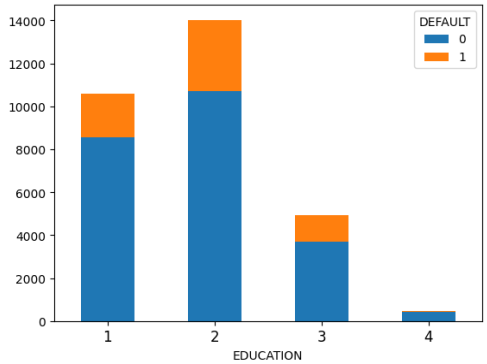
dtype: float64

Insights:

We can see that there is a marginally higher percentage of males who default than females who default.

Thus the input variable "SEX" may be a useful input attribute.

Distribution of EDUCATION status for main data set



EDUCATION	DEFAULT	
1	0	80.765234
	1	19.234766
2	0	76.265146
	1	23.734854
3	0	74.842384
	1	25.157616
4	0	92.948718
	1	7.051282

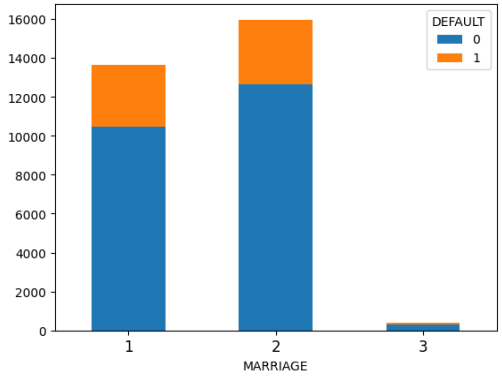
dtype: float64

Insights:

We can see that there is a higher percentage of people in high school defaulting as compared to university and graduate school.

The "EDUCATION" attribute could be a useful input attribute for our model.

Distribution of MARRIAGE status for main data set



MARRIAGE	DEFAULT	
1	0	76.528296
	1	23.471704
2	0	79.071661
	1	20.928339
3	0	76.392573
	1	23.607427

dtype: float64

Insights:
We can see that there is a higher percentage of defaulters in the married and other categories compared to the single category.
Thus, the marriage variable can be a useful input attribute.

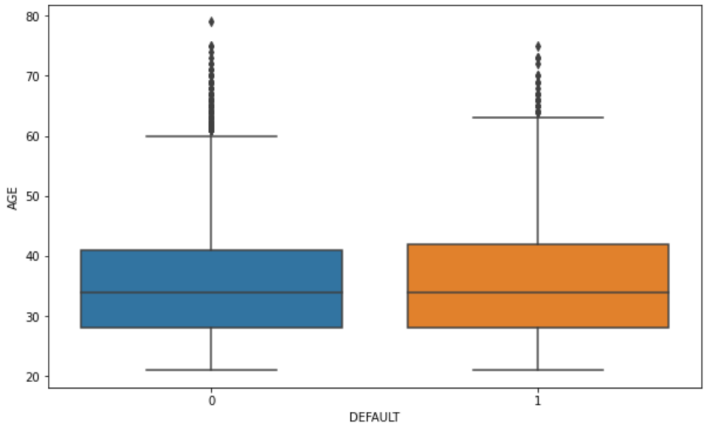
Actionable:

- In conclusion, the dataset is quite unbalanced regarding default and non-default payments distribution. Hence, we will transform the data into a more balanced one in terms of the number of default and non-default payments to enhance the model's performance.

3) Data Pre-Processing part 1

Data Encoding for Age Variable

Before Data Encoding



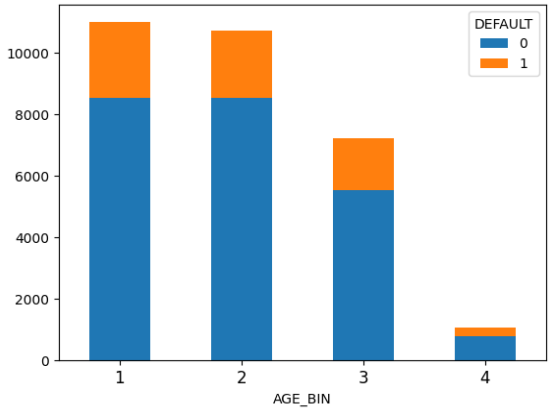
Insights
As we can see, if we were just to leave the age variable as it is, the age input attribute would not be useful as it is not distinguishable for the 2 targets. Hence, we will attempt to use Data Encoding on the Age variable to see if Age can be useful in the later part.

Minimum age is 21
Maximum age is 79

Based on statistics in Singapore, we have decided to split the age groups as

- [21, 30], [31, 40], [41, 54], [55, 80].
- **21-30** years old are young graduates who just joined the workforce and may need money to fund expenses.
- **31-40** are working adults who are more mature and have a more stable income.
- **41-54** are similarly those who worked for a very long time and should hold a stable income.
- **Aged 55 and above** are either retired individuals with retirement funds or those who need a lot of money to fund their lifestyle after retiring.

After Data Encoding




```
AGE_BIN  DEFAULT
1        0          77.562880
         1          22.437120
2        0          79.566881
         1          20.433119
3        0          76.526797
         1          23.473203
4        0          73.314340
         1          26.685660
dtype: float64
```

Insights:
From the statistics, we can see that those above 55 are more likely to default. Thus, age can be a useful input attribute for our model.

Data transformation via Min-Max

We use MinMaxScaler() to transform the data into values between 0 and 1. This is because input variables in different magnitudes will contribute differently to the feature selection and fitting of the model. This will lead to bias when larger numbers get “prioritised”. MinMaxScaler() serves to deal with such a potential problem.

MinMaxScalar seeks to rescale variables into the range 0 to 1. This can preserve the shape of the original distribution without changing the information embedded in the data.

This will allow the machine learning algorithm to perform better and converge to a minimum faster as the features are transformed closer to a Normal Distribution.

4) Model Testing Before Feature Selection

Before we move on to select the best features, we test the training data set against the testing dataset with 5 different classification models. This helps to give us a rough understanding of how well the model performs before making any feature selection, so we will know if our feature selection in the later sections indeed helped.

Elaborations on the hyperparameter chosen will be covered in greater detail in the Model Selection Section. Further evaluation on the models will be done in the model evaluation section

Model	Accuracy	Average Class Accuracy	Recall (default as positive)
SVM	81.21%	62.92%	29.98%
MLPClassifier	81.01%	63.95%	33.21%
KNN	79.81%	62.30%	30.76%
Logistic Regression	80.67%	61.16%	26.02%
Random Forest	80.37%	60.69%	25.24%

Before we do feature selection we observe some insights from the models

- The accuracy is generally around **0.798 to 0.812**, which is relatively high.
- However, accuracy cannot explain the performance of models when dealing with imbalanced dataset.
- The average class accuracy is relatively low due to the imbalance of the dataset.
- The recall is generally between **0.252 to 0.332**, in terms of predicting default. This is considered very low.

We will see if there will be improvements after making the Feature Selection.

** Explanation of the model parameter and model can be found in Section 9.*

5) Feature Selection

In feature selection, we utilise different methods to select those feature variables that contribute significantly to the target variable, and we forfeit the feature variables that have no or poor relation with target variables.

In this section, we use the

1. **Correlation plots** to find the correlation of dependent variables with target variables
2. **Chi-2 Test** to determine the independence of categorical variables with default payment
3. **Wrapper method such as Forward Selection, Backward Elimination** to find the best combination of input variables that generate the best prediction model.
4. **Embedded method such as Lasso Regularisation** to eliminate non-useful variables.

This also involves writing simple algorithms to group variables together.

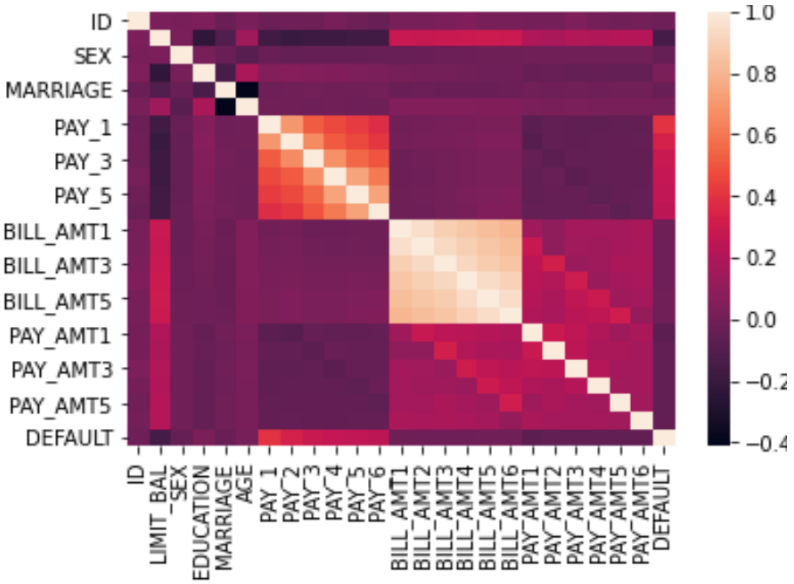
Filter Method

Analyse Correlation

A correlation matrix with all variables is plotted to visualise the correlation between variables. From the plot, no useful information was extracted due to excessive information.

We continue to analyse the correlation plot by parts.

Namely ***PAY 1 to 6, PAY_AMOUNT 1 to 6, BILL 1 to 6*** and ***billpercredit 1 to 6*** to see the correlation to the target variable. We will then extract



useful information from the correlation plots

	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	DEFAULT
PAY_1	1.000000	0.698389	0.516970	0.460224	0.424473	0.373805	0.396019
PAY_2	0.698389	1.000000	0.663529	0.512773	0.462717	0.407086	0.327093
PAY_3	0.516970	0.663529	1.000000	0.678931	0.551430	0.492827	0.286999
PAY_4	0.460224	0.512773	0.678931	1.000000	0.745419	0.602875	0.269055
PAY_5	0.424473	0.462717	0.551430	0.745419	1.000000	0.740357	0.260850
PAY_6	0.373805	0.407086	0.492827	0.602875	0.740357	1.000000	0.244437
DEFAULT	0.396019	0.327093	0.286999	0.269055	0.260850	0.244437	1.000000

- **Observation 1:** variable PAY has a moderate positive correlation with default. This is intuitively justified because the larger the numerical value of repayment status, the longer someone delays paying their bill, and the more likely they will have default payment. From PAY_1 to PAY_6 and DEFAULT, the correlation drops as they are likely to have saved enough for payment next month.
- However, we need further feature selection techniques to determine which PAY variables to use in the model to best predict DEFAULT.

	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	DEFAULT
PAY_AMT1	1.000000	0.285576	0.252191	0.199558	0.148459	0.185735	-0.072929
PAY_AMT2	0.285576	1.000000	0.244770	0.180107	0.180908	0.157634	-0.058579
PAY_AMT3	0.252191	0.244770	1.000000	0.216325	0.159214	0.162740	-0.056250
PAY_AMT4	0.199558	0.180107	0.216325	1.000000	0.151830	0.157834	-0.056827
PAY_AMT5	0.148459	0.180908	0.159214	0.151830	1.000000	0.154896	-0.055124
PAY_AMT6	0.185735	0.157634	0.162740	0.157834	0.154896	1.000000	-0.053183
DEFAULT	-0.072929	-0.058579	-0.056250	-0.056827	-0.055124	-0.053183	1.000000

- **Observation 2:** *PAY_AMT* alone does not seem correlated to default payment since the correlation magnitude is relatively small (<0.1).
- This is intuitively justified because the amount someone pays for their bills may not be directly related to their financial ability. For example, they may be paying a lot for 1 month, but that could be due to them not paying for the previous months and just trying to pay everything they own before bankruptcy, for instance. Even so, they may not have enough to pay their full bills and can still default.

	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	DEFAULT
BILL_AMT1	1.000000	0.951484	0.892279	0.860272	0.829779	0.802650	-0.019644
BILL_AMT2	0.951484	1.000000	0.928326	0.892482	0.859778	0.831594	-0.014193
BILL_AMT3	0.892279	0.928326	1.000000	0.923969	0.883910	0.853320	-0.014076
BILL_AMT4	0.860272	0.892482	0.923969	1.000000	0.940134	0.900941	-0.010156
BILL_AMT5	0.829779	0.859778	0.883910	0.940134	1.000000	0.946197	-0.006760
BILL_AMT6	0.802650	0.831594	0.853320	0.900941	0.946197	1.000000	-0.005372
DEFAULT	-0.019644	-0.014193	-0.014076	-0.010156	-0.006760	-0.005372	1.000000

- **Observation 3:** The variable *BILL_AMT* should not be included in any of our models because *BILL_AMT1 to BILL_AMT6* are strongly correlated with each other (correlation magnitude among *BILL_AMT1 to BILL_AMT6* > 0.8). The highly correlated regressors in the

model can potentially introduce a multicollinearity problem, which undermines the variable's statistical significance or introduce overfitting to the model.

- If we wish to introduce *BILL_AMT*, we will only choose one *BILL_AMT*, which we will determine from the wrapper method in the latter part.

	billpercredit1	billpercredit2	billpercredit3	billpercredit4	billpercredit5	billpercredit6	DEFAULT
billpercredit1	1.000000	0.934032	0.862141	0.799921	0.741718	0.695778	0.086168
billpercredit2	0.934032	1.000000	0.907715	0.843031	0.780456	0.733054	0.099039
billpercredit3	0.862141	0.907715	1.000000	0.891452	0.824188	0.778303	0.103902
billpercredit4	0.799921	0.843031	0.891452	1.000000	0.905570	0.845750	0.115925
billpercredit5	0.741718	0.780456	0.824188	0.905570	1.000000	0.916978	0.119156
billpercredit6	0.695778	0.733054	0.778303	0.845750	0.916978	1.000000	0.123373
DEFAULT	0.086168	0.099039	0.103902	0.115925	0.119156	0.123373	1.000000

- **Observation 4:** billpercredit is computed by dividing BILL_AMT and LIMIT_BAL. This is used so we know the percentage of credit limit used in the month and how much it has overshot the credit limit.
- These variables have low positive correlation to DEFAULT. It makes intuitive sense because higher billpercredit is associated to overspending and overspending is associated to higher chances of defaulting.

Chi-Square Test

- H0: There is no relationship between the *DEFAULT* and categorical input variables.
- H1: There is a relationship between the *DEFAULT* and categorical input variables.

In this dataset, we have decided to perform a Chi-Square Test on commonly known categorical variables such as *SEX, EDUCATION, MARRIAGE*, *AGE* to test whether there is a relationship between these variables and *DEFAULT*, and decide whether or not these variables may be a useful input attribute in the model to predict default.

The categorical variable used in the test	P-value of Chi-Square test
SEX	4.9446e-12
EDUCATION	1.4950e-34
MARRIAGE	7.7907e-07
AGE_BIN	1.9551e-08

Insights:

We can reject the null hypothesis at **5%** level of significance that there is no relationship between each of the categorical variables and DEFAULT. Hence, we will consider adding these 4 categorical variables into our next feature selection steps, including Forward Backward regression and Regularisation. It is worth noting that EDUCATION gives the smallest p-value.

Wrapper Method

Wrapper methods use procedures to add and/or remove feature variables depending on the direction of choosing variables to find the optimal combination that maximises model performance.

We will use **Forward Selection,** and **Backward Elimination**

Forward Selection

Forward stepwise selection (or forward selection) is a regressor selection method that:

- Starts from a model that contains no regressors
- It is followed by adding the most statistically significant variables one at a step. In this case, our evaluation criterion is the r-squared value that measures the proportion of the variance of a target variable explained by a regressor in a regression model.
- Stops selection until 5 best regressors are selected.

feature_idx	cv_scores	avg_score	feature_names
(5,)	[0.11540273444382354, 0.1269922858904763, 0.18…	0.154612	(PAY_1,)
(5, 10)	[0.12139540557848905, 0.1408423861249124, 0.19…	0.165433	(PAY_1, PAY_6)
(0, 5, 10)	[0.123830783256349, 0.14410363090163247, 0.206…	0.171206	(LIMIT_BAL, PAY_1, PAY_6)
(0, 5, 7, 10)	[0.12680785812408601, 0.1468074689515605, 0.20…	0.174119	(LIMIT_BAL, PAY_1, PAY_3, PAY_6)
(0, 3, 5, 7, 10)	[0.12870963309267314, 0.1479584634615232, 0.20…	0.175167	(LIMIT_BAL, MARRIAGE, PAY_1, PAY_3, PAY_6)

Insights:

- From the Forward Regression, the top 5 features to select are LIMIT_BAL, MARRIAGE, PAY_1, PAY_3, and PAY_6

Backward Elimination

Backward stepwise selection (or backward elimination) is a regressor selection method which:

- Starts from a model that contains all regressors that we selected
- It is then followed by removing the least significant variables one step at a time
- This is repeated until only 5 regressors remain in the model

We use recursive feature elimination to determine the ranking of each regressor, 1 being the most statistically significant regressor while 25 being the least statistically significant regressor.

Columns	Feature_names	Selected	RFE_ranking
0	LIMIT_BAL	False	16
1	SEX	False	6
2	EDUCATION	False	12
3	MARRIAGE	False	5
4	AGE	False	11
5	PAY_1	True	1
6	PAY_2	False	8
7	PAY_3	True	1
8	PAY_4	False	9
9	PAY_5	False	3
10	PAY_6	True	1
11	BILL_AMT1	False	19
12	BILL_AMT2	False	24
13	BILL_AMT3	False	23
14	BILL_AMT4	False	20
15	BILL_AMT5	False	21
16	BILL_AMT6	False	22
17	PAY_AMT1	False	13
18	PAY_AMT2	False	15
19	PAY_AMT3	False	25
20	PAY_AMT4	False	18
21	PAY_AMT5	False	14
22	PAY_AMT6	False	17
23	billpercredit1	False	7
24	billpercredit2	True	1
25	billpercredit3	False	10
26	billpercredit4	False	4
27	billpercredit5	False	2
28	billpercredit6	True	1

Insights:

- From the Backward Elimination (Recursive Feature Elimination), the top 5 features are PAY_1, PAY_3, PAY_6, billpercredit2, and billpercredit6.

Embedded Method

Lasso Regularisation (L1)

Regularisation is a method to add a penalty to a model and seeks to prevent overfitting. The regularisation term helps to generalise a model. This is a form of Bias-variance trade-off as we add bias to reduce variance (overfitting). We acknowledge that there are both L1 and L2 Regularisation available.

However, we have decided to use L1 Regularisation instead of L2 Regularisation because we know that L1 Regularisation can shrink some of the coefficients to zero, which suggests that that particular feature variable is not useful in our model.

```
from sklearn.feature_selection import SelectFromModel
sel_ = SelectFromModel(LogisticRegression(C=1, penalty='l1', solver =␣
 ↪'liblinear'))
sel_.fit(x, y)
sel_.get_support()
```

```
array([False,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True, False, False, False, False, False, False,  True,
       False, False, False, False, False,  True,  True,  True,  True,
        True,  True])
```

```
# print the names of features that are selected
selected_feat= x.columns[(sel_.get_support())]
selected_feat
```

```
Index(['SEX', 'EDUCATION', 'MARRIAGE', 'AGE_BIN', 'PAY_1', 'PAY_2', 'PAY_3',
       'PAY_4', 'PAY_5', 'PAY_6', 'PAY_AMT1', 'billpercredit1',
       'billpercredit2', 'billpercredit3', 'billpercredit4', 'billpercredit5',
       'billpercredit6'],
      dtype='object')
```

Insights:

From the Lasso Regularisation, we know that the important features are SEX, EDUCATION, MARRIAGE, AGE_BIN, PAY_1 to PAY_6, and billpercredit1 to billpercredit6.

Final Features Selected for Models Predictions

From Forward Selection, Backward Elimination and Lasso Regularisation, *PAY_1, PAY_3, PAY_6* are chosen every time, which means that they are one of our most important features that should be added. *MARRIAGE, billpercredit2 and billpercredit6* appear 2 out of 3 times. Although *EDUCATION and LIMIT_BAL* only appeared once, they have been tested and are shown to be useful attributes in predicting default payments.

By repeatedly testing the models using different subsets of the features mentioned above, we conclude that *LIMIT_BAL, MARRIAGE, PAY_1, PAY_3, PAY_6, billpercredit2, billpercredit6, and EDUCATION* are the best subset of features that produced the highest performance in each model. However, due to the page limit constraints, we will not show all the results for the repeated testing of the models.

Predictors to be used:

- [“LIMIT_BAL”, “MARRIAGE”, “EDUCATION”, “PAY_1”, “PAY_3”, “PAY_6”, “billpercredit2”, “billpercredit6”]

6) Model Testing Before Balancing the dataset

After we selected the features, we will run all the models again and check the evaluation scores again. This helps to give us a rough understanding if the reduction in features has indeed been useful in predicting default payment

Elaboration on the hyperparameter chosen will be covered in greater detail in the Model Selection Section. Further evaluation on the models will be done in the model evaluation section

Model	Accuracy	Average Class Accuracy	Recall (default as positive)
SVM	81.32%	63.44%	31.24%
MLPClassifier	81.05%	63.44%	31.71%
KNN	80.95%	64.57%	35.07%
Logistic Regression	80.96%	62.50%	29.26%
Random Forest	81.33%	63.26%	30.70%

Legend

Value that has been **improved** after feature selection

Value that has been **worsened** after feature selection

Before we make feature selection we observe some insights from the models

- The cells highlighted in green suggest the improvement of the statistic after feature selection as compared to using all features. The cells highlighted in red suggest the worsened statistic after feature selection.
- We can see that most evaluation statistics have improved after feature selection. This means that the features we choose from the model can improve our model. For the worsening statistic, it is also a small difference.
- However, we still have yet to solve the problem of an imbalanced dataset. Having an accuracy of more than 80% may not be a reliable indicator of the matrix. If a model predicts all 0, then it can also have 80% accuracy.
- Even though all the evaluation statistics have improved, we notice that the recall and average class accuracy are still relatively low. Thus we will look to improve these after balancing the data.

To obtain a more reliable evaluation, we will balance the data and run the models again

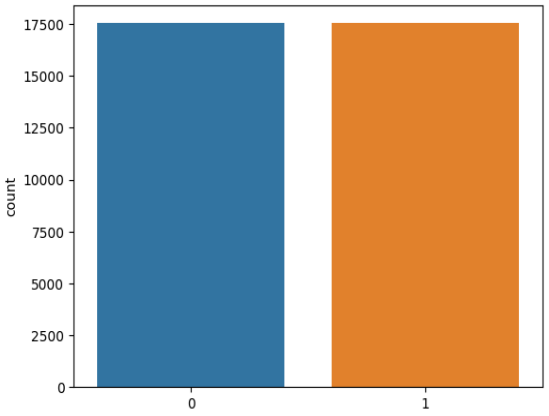
** Explanation of the model parameter and model is can be found in Section 9.*

7) Data Pre-Processing part 2

Synthetic Minority Oversampling Technique (SMOTE)

From the previous section, we found that there is an imbalance of data. The default and non-default roughly have an 8:2 ratio, which will cause misleading accuracy scores by our model. A model can predict with up to 80% accuracy even if it predicts all “0”. Thus, SMOTE is used to balance the training dataset given.

```
Before OverSampling, counts of label '1': 4968
Before OverSampling, counts of label '0': 17532

After OverSampling, the shape of train_X: (35064, 8)
After OverSampling, the shape of train_y: (35064,)

After OverSampling, counts of label '1': 17532
After OverSampling, counts of label '0': 17532
```



Thus with SMOTE (detailed code in the coding pdf), an oversampling method to produce synthetic data for minority class non-default, we can balance the initial minority dataset. This will allow us to evaluate the prediction results more accurately.

8) Model Selection (Machine Learning Model Used)

Hyperparameter Tuning for all the models

Hyperparameter tuning is essential as it gives us optimised values and parameters, which maximises our model’s predictive capabilities.

GridSearchCV is used to select the best combination of hyperparameters for each model hence finding the best possible model for the algorithm being used in the particular problem, compared based on the average class accuracy.

The cross-validation method used in GridSearchCV is k-fold cross-validation. In this project, we adopt a 5-fold cross-validation, which means we will iterate GridSearchCV 5 times to obtain the optimal hyperparameters. Any increase in the number of the fold will not improve the model much but will increase the time needed to run significantly.

Despite the time cost, this method allows the best hyperparameters to be used in each model.

Model 1: Support Vector Machine (SVM)

We first use a supervised linear model, SVM, for classification. The main goal of SVM is to distinctly classify the data points by finding a hyperplane in the N-dimensional space, where N refers to the number of features. The objective function is to minimise the difference in distance between the hyperplane and the data points that should be linearly separable.

The candidate HyperParameters selected are ***'rbf', 'linear', 'poly' and 'sigmoid'.*** Using GridSearchCV, the best HyperParameter found is ***'rbf'*** (**Gaussian Kernel Radial Basis Function).**

This makes sense in theory as rbf, or Radial Kernel, can deal with overlapping data. It can find Support Vector Classifiers with infinite dimensions.

Model Performance

Evaluation Metrics	Value
Accuracy	**76.89%**
Precision (default as positive)	**48.24%**
Recall (default as positive)	**53.30%**
Precision (non-default as positive)	**86.23%**
Recall (non-default as positive)	**83.64%**
Average Class Accuracy	**68.47%**
F1 Score (default as positive)	**0.5064**
F1 Score (non-default as positive)	**0.8492**
AUC / ROC*	**0.7100**

Confusion Matrix

	Default	Non – default
Default	**889**	**779**
Non – default	**954**	**4878**

*Row represents **actual** default / non-default

*Column represents **predicted** default / non-default

*Further evaluation of models will be done in Section 6, Model Evaluation below.

* Details of the confusion matrix and ROC curve can be found in the Appendix section

Model 2: Multi-Layer Perceptron Model (MLP Classifier)

We then used the MLP classifier which is a feedforward artificial neural network model, a type of supervised learning model.

The candidate parameters for penalty term alpha are 0.2, 0.205, 0.21, 0.215 and 0.22. Using GridSearchCV, the best penalty strength selected is **0.2**. A relatively high alpha as compared to default 0.001 may solve the problem of high variance by encouraging smaller weights so that the probability of overfitting is reduced.

The candidate maximum iterations are between 220, 225 and 230. Using GridSearchCV, the best maximum iteration is **220**. This means that the solver will only iterate up to 220 times until convergence.

Model Performance

Evaluation Metrics	Value
Accuracy	**77.60%**
Precision (default as positive)	**49.67%**
Recall (default as positive)	**53.54%**
Precision (non-default as positive)	**86.41%**
Recall (non-default as positive)	**84.48%**
Average Class Accuracy	**69.01%**
F1 Score (default as positive)	**0.5153**
F1 Score (non-default as positive)	**0.8544**
AUC / ROC*	**0.7500**

Confusion Matrix

	Default	Non – default
Default	**893**	**775**
Non – default	**905**	**4927**

*Row represents **actual** default / non-default

*Column represents **predicted** default / non-default

*Further evaluation of models will be done in Section 6, Model Evaluation below.

* Details of the confusion matrix and ROC curve can be found in the Appendix section

Model 3: K-Nearest-Neighbours (KNN)

The K-Nearest Neighbours algorithm is a supervised learning classifier. The algorithm uses distance to make predictions about data input. The 'K' in KNN refers to the number of nearest neighbours.

The candidate's closest number of neighbours is ranged from 1 to 21. Using GridSearchCV, the best estimator number of neighbours selected is **7**, which will produce the highest accuracy among all options.

The candidate distance metrics for finding the distance between neighbours are Euclidean Distance, Manhattan Distance, and Minkowski Distance. Using GridSearchCV, the best distance metric is Manhattan Distance, which has a formula of $|x_i – y_i|$.

Model Performance

Evaluation Metrics	Value
Accuracy	**76.65%**
Precision (default as positive)	**47.39%**
Recall (default as positive)	**45.08%**
Precision (non-default as positive)	**84.51%**

Confusion Matrix

	Default	Non – default
Default	**752**	**916**
Non – default	**835**	**4997**

*Row represents **actual** default / non-default

Recall (non-default as positive)	**85.68%**
Average Class Accuracy	**65.38%**
F1 Score (default as positive)	**0.4621**
F1 Score (non-default as positive)	**0.8509**
AUC / ROC*	**0.6900**

*Column represents **predicted** default / non-default

*Further evaluation of models will be done in Section 6, Model Evaluation below.

* Details of the confusion matrix and ROC curve can be found in the Appendix section

Model 4: Logistic Regression

We also used a Logistic Regression model, which is a supervised learning algorithm that is used to predict a target variable based on regressors that we chose earlier in feature selection. The logistic regression model computes a sum of the input features by multiplying a suitable coefficient and calculates the logistic of the result.

The candidate C value – penalty strength selected are 0.1,0.5,1.0,2.0, and 3.0. Using GridSearchCV, the best penalty strength selected is **0.1**. A low value of C tells the model to give lower weight to the training data to prevent overfitting.

Model Performance

Evaluation Metrics	Value
Accuracy	**77.57%**
Precision (default as positive)	**49.61%**
Recall (default as positive)	**53.18%**
Precision (non-default as positive)	**86.33%**
Recall (non-default as positive)	**84.55%**
Average Class Accuracy	**68.86%**
F1 Score (default as positive)	**0.5133**
F1 Score (non-default as positive)	**0.8543**
AUC / ROC*	**0.7400**

Confusion Matrix

	Default	Non – default
Default	**887**	**781**
Non – default	**901**	**4931**

*Row represents **actual** default / non-default

*Column represents **predicted** default / non-default

*Further evaluation of models will be done in Section 6, Model Evaluation below.

* Details of the confusion matrix and ROC curve can be found in the Appendix section

Model 5: Random Forest Classification

We also used Random Forest Classification, a supervised learning algorithm. Random Forest contains several decision trees, which are combinations of different classifiers on various subsets. It then takes the average, which reduces variance.

Random Forest was used instead of trees to average out the irregular patterns of simply just using Decision Trees. In Decision Trees, trees are grown to learn many patterns, and there tends to be overfitting, i.e. having a low bias but very high variance. This is not what we want in our model. Random Forest seeks to reduce the variance by averaging out multiple Decision Trees. Although Random Forest leads to an increase in bias, the significantly reduced variance is what is desirable and greatly prevents overfitting.

The candidate estimator sizes selected are 10,50,80,100,150,200,250, and 300. Using GridSearchCV, the best estimator size (number of trees in the random forest) selected is **50**.

The performance of Random Forest drastically increases when tree sizes increase from 10 to 50 and remains almost the same when tree sizes continue to increase. Hence, to prevent overfitting, we choose 50 as the estimator size.

Model Performance

Evaluation Metrics	Value
Accuracy	**76.05%**
Precision (default as positive)	**46.82%**
Recall (default as positive)	**56.41%**
Precision (non-default as positive)	**86.76%**
Recall (non-default as positive)	**81.67%**
Average Class Accuracy	**69.04%**
F1 Score (default as positive)	**0.5117**
F1 Score (non-default as positive)	**0.8414**
AUC / ROC*	**0.7400**

Confusion Matrix

	Default	Non – default
Default	**941**	**727**
Non – default	**1069**	**4763**

*Row represents **actual** default / non-default

*Column represents **predicted** default / non-default

*Further evaluation of models will be done in Section 6, Model Evaluation below.

* Details of the confusion matrix and ROC curve can be found in the Appendix section

9) Model Evaluation

Evaluation metrics quantifies the performance of the model. By comparing evaluation metrics across different models, we can conclude the best and worst-performing models depending on the characteristics of the evaluation metrics and the aim of the project.

	SVM	MLP Classifier	K–Nearest Neighbours	Logistic Regression	Random Forest
Accuracy	76.89%	77.60%	76.65%	77.57%	76.05%
Precision (default as positive)	48.23%	49.67%	47.39%	49.61%	46.82%
Recall (default as positive)	53.29%	53.54%	45.08%	53.18%	56.41%
Precision (non default as positive)	86.23%	86.41%	84.51%	86.33%	86.76%
Recall (non default as positive)	83.64%	84.48%	85.68%	84.55%	81.67%
Average Class Accuracy	68.47%	69.01%	65.38%	68.86%	69.04%
F1 Score (default as positive)	0.5064	0.5153	0.4621	0.5133	0.5117
F1 Score (non default as positive)	0.8509	0.8544	0.8509	0.8543	0.8414
AUC / ROC*	0.7100	0.7500	0.6900	0.7400	0.7400

Legend

Highest Value for that Evaluation Matrix

Lowest Value for that Evaluation Matrix

Insights

- Even though the accuracy has decreased as compared to evaluation metrics before balancing of data and feature selection, this is due to balancing of data which makes the default and non-default entries in the test data equal in proportion.
- After balancing the dataset and feature selection, all the recall metrics when we set default as positive has increased drastically by around **20 – 25%,** which in turn leads to an overall increase of average class accuracy, by around **8 – 10%**, which is what we desire as our problem statement indicates that we prioritise average class accuracy when we determine which is the model with the best performance.

Model Comparison

Best Models: MLP Classifier and Random Forest

- These are the best models to predict for default and non default in terms of average class accuracy and F1 score.
- These are the models that we want to consider using for default and non default payment
- **MLP Classifier:** It has the highest accuracy, F1 score and recall. It also have the second highest average class accuracy
- **Random Forest**: It has the highest average class accuracy. Even though it perform poorly for other metrics, since we are trying to find the model that have the best average class accuracy, we will consider random forest as one of the better models
- Potential Reason:
    - Both model are able to work well with large datasets well

Worst Model: K – Nearest – Neighbour

- This is the worst model to predict for default and non default in terms of average class accuracy and F1 score.
- This model performs the worst in terms of average class accuracy, F1 score and ROC.
- Thus, we will not want to use this model to predict for default and non default payments
- Potential Reasons:
    - Does not work well with the dataset as it is too huge
    - There is noise in the data or outlier which KNN is very sensitive to.

** Read explanation of evaluation metric below to know what each metrics imply*

Explanation of evaluation metrics

- **True Positive (TP):**
    - The proportion of correct predictions among all default payments next month.
- **True Negative (TN):**
    - The proportion of correct predictions among all non-default payments next month.
- **Precision:**
    - The proportion of actual positive instances among all positive predictions.
- **Recall:**
    - The proportion of actual positive instances among all actual positive instances.
- **Accuracy:**
    - Proportion of correct predictions in the entire dataset.
    - For all models, the overall accuracy varies slightly from **76%** to **78%**.
    - This is similar to the case where we predict everything as positive, where the accuracy is **77.8%**.
    - Hence, we use other evaluation metrics to complement our decision of which model has the best performance.
- **Average Class accuracy:**
    - Average class accuracy combines the recall when we set default as positive and when we set non default as positive, and divide it by 2.
    - This is one of the **most important metrics** we value when determine the best performance model as it is one of the best way to determine “accuracy”

- **F1 score:**
  - F1 score combines the recall and precision of the model, also known as the harmonic mean of the model's precision and recall.
  - Another indication of "accuracy" that helps us to determine the best-performing models in this project.
- **AUC for ROC curve:**
  - Area under the curve for ROC curve, which is a graph that is plotted with TPR against FPR.
  - AUC for the ROC curve indicates how well the model is as compared to random classifiers in distinguishing positive classes against negative classes.
  - The higher the value, the better the model is.
  - AUC for the ROC curve of range [0.7,0.8] is considered acceptable as the specific model should perform better than random classifiers.

10) Room for Improvement

Problem with the Data Set

The dataset is imbalanced as the default to non-default ratio is roughly 8:2, which is very difficult to come up with a reliable model that can produce a high average class accuracy. Hence, we have decided to use other metrics, such as F1 score and Recall, which can be more useful when evaluating our project since the main aim is to predict the customers who will default.

Even though we have used SMOTE to balance the data, SMOTE itself has some limitations. Firstly, SMOTE does not consider the fact that the data points chosen can be from other classes. Hence, this may result in the increase of overlapping classes and causes more noise in the data. SMOTE may also oversample samples that are not as useful. To put things into perspective in our project, SMOTE could have oversampled samples that are far away from the decision boundary, which plays an insignificant role in the prediction of default.

Page limit constraints

Due to the 20-page restriction, we are unable to show a lot more of our testing process

Machine Learning is not a linear process. It requires a lot of trial and error, repeating the process. We demonstrated this by running the model 3 times, namely before feature selection, before balancing of data and after 1 final run for evaluation.

However, we actually did a lot more testing behind the scenes. For example, we tested a lot of different combinations of the features selected from all the feature selection techniques and arrived at the set with the best outcome. However, due to the page limit, we cannot include all the code and thought process in.

11) Appendix

Exploratory Data Analysis

Checking for null values

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

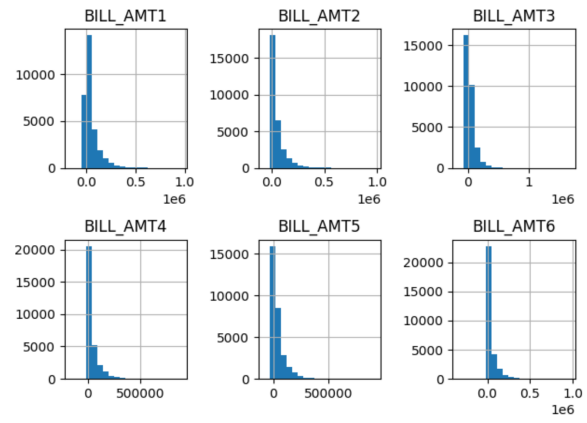
- We see that all 25 columns have 30,000 rows of data. The data is clean and does not have any missing values

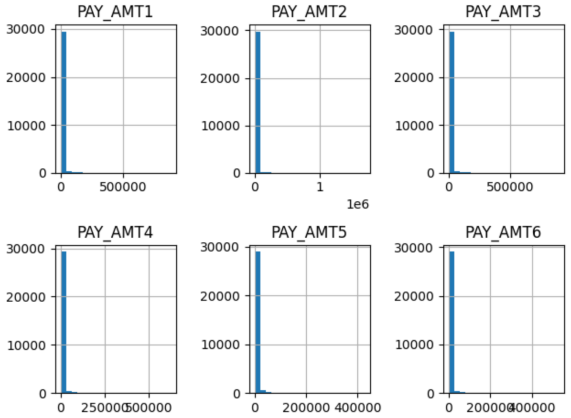
Check inconsistencies in variables

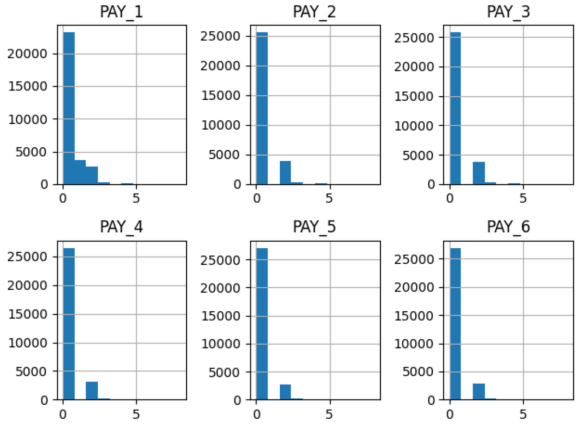
```
EDUCATION [1, 2, 3, 4]

MARRIAGE [1, 2, 3]

PAY_1 [0, 1, 2, 3, 4, 5, 6, 7, 8]
PAY_2 [0, 1, 2, 3, 4, 5, 6, 7, 8]
PAY_3 [0, 1, 2, 3, 4, 5, 6, 7, 8]
PAY_4 [0, 1, 2, 3, 4, 5, 6, 7, 8]
PAY_5 [0, 2, 3, 4, 5, 6, 7, 8]
PAY_6 [0, 2, 3, 4, 5, 6, 7, 8]
```

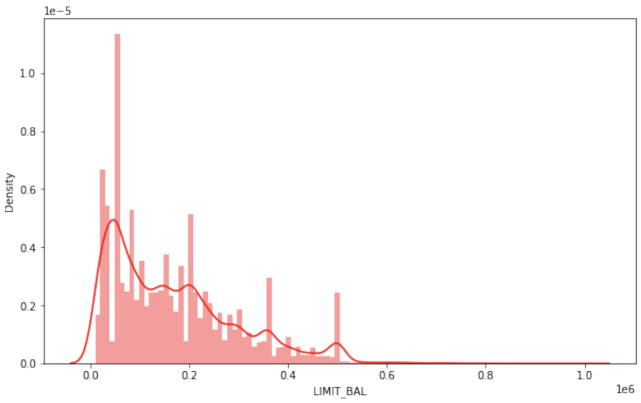
There is no inconsistencies

Get the Histogram of PAY, BILL and PAY_AMT and Limit Balance



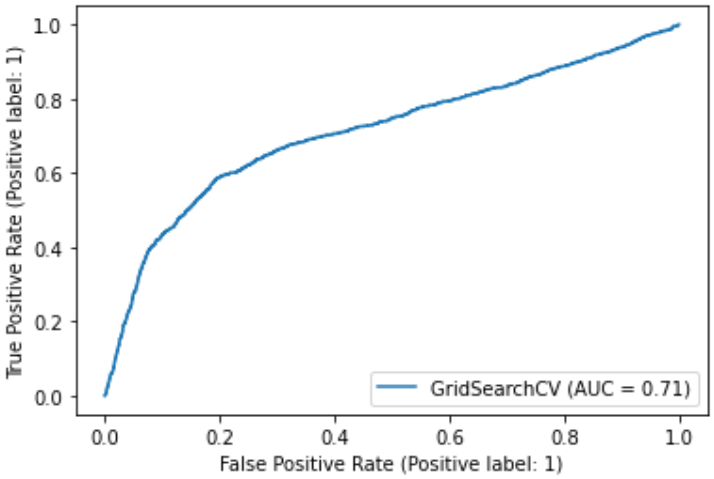




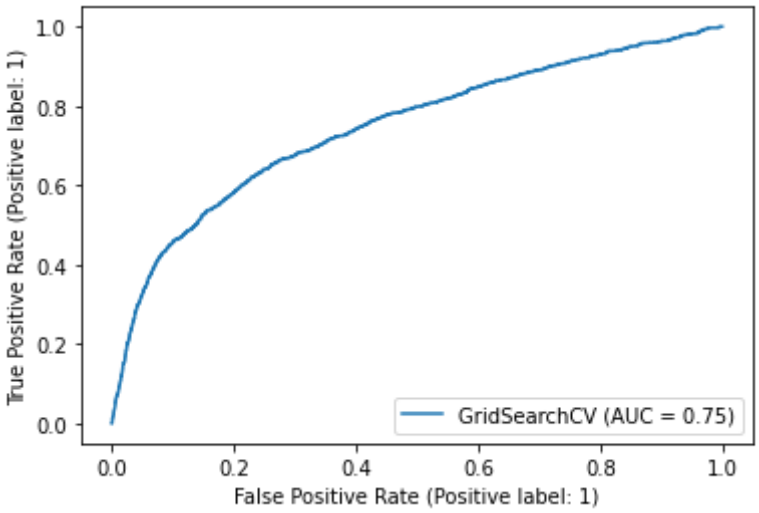


Model Selection

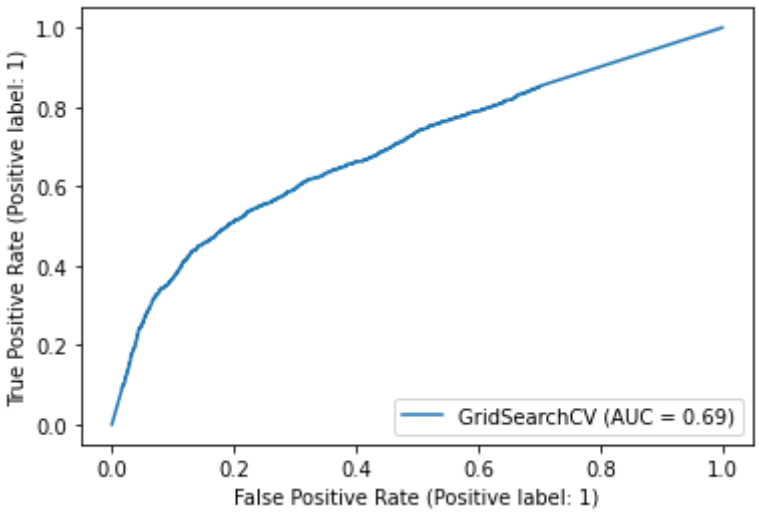
Support Vector Machine



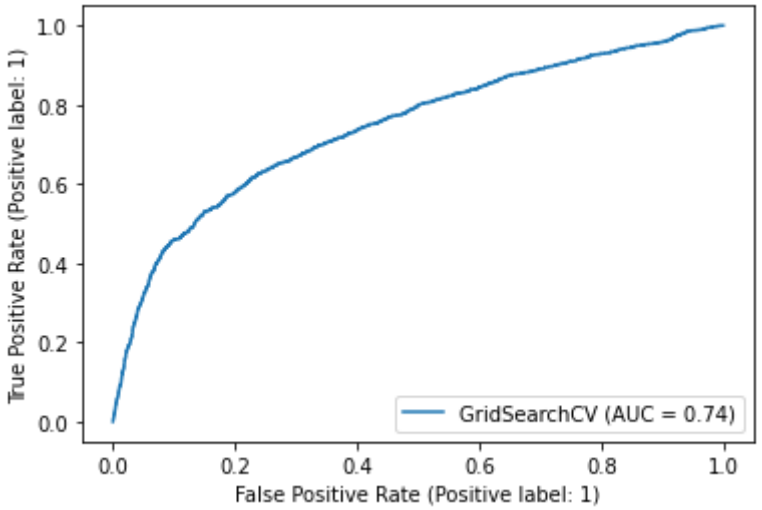
Multi-Layer Perceptron Model (MLP classifier)



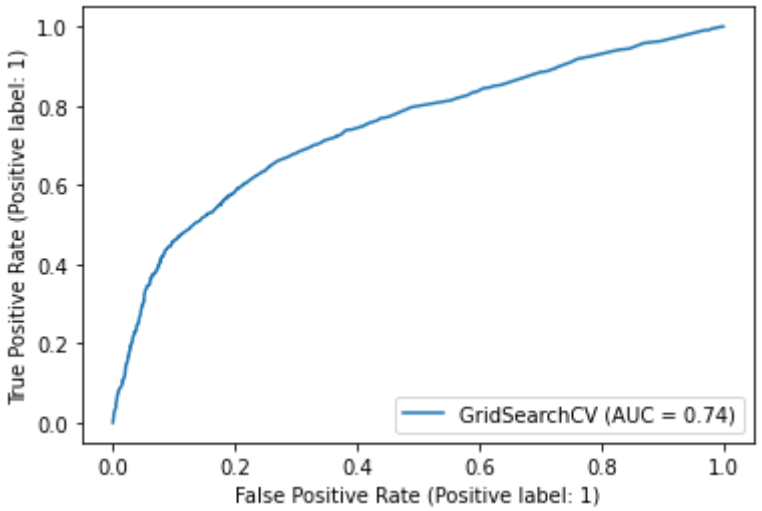
K-Nearest-Neightbour



Logistic Regression



Random Forest



Annex

CODE USED IS ON NEXT PAGE