BT2103 Project Report

Table of Contents

Introduction

What causes a credit card user to default on his debts? This information is of utmost importance to banks, as it would influence the decision on whether to extend credit to a particular individual, and with what terms. This way, banks can avoid lending to those who are likely to default, thus reducing the risk of financial losses. For individuals less likely to default, banks can offer more competitive rates to attract and retain customers. In this report, we aim to predict whether a credit card holder will default on his payment in the next month.

To achieve the above, we will use a dataset containing information on 30,000 credit card holders from a bank in Taiwan. Each credit card holder is described by 23 feature attributes, which are outlined below.

Client info

- *ID*: ID of each client
- *LIMIT_BAL*: Amount of given credit in NT$ (includes individual and family/supplementary credit)
- *SEX*: Gender (1=male, 2=female)
- *EDUCATION*: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
- *MARRIAGE*: Marital status (1=married, 2=single, 3=others)
- *AGE*: Age in years

Payment

PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6

- Repayment status in Sep 2005 to Apr 2005, respectively
- Categorical variable with values:
    - -1 = Paid in full
    - 1 to 9 = payment delay for 1 to 9 months, respectively

Bill Statement

BILL_AMT1, BILL_AMT2, BILL_AMT3, BILL_AMT4, BILL_AMT5, BILL_AMT6

- Amount of bill statement in NT$ in Sep 2005 to Apr 2005, respectively

Previous Payments

PAY_AMT1, PAY_AMT2, PAY_AMT3, PAY_AMT4, PAY_AMT5, PAY_AMT6

- Amount of previous payment in NT$ in Sep 2005 to Apr 2005, respectively

Target Variable

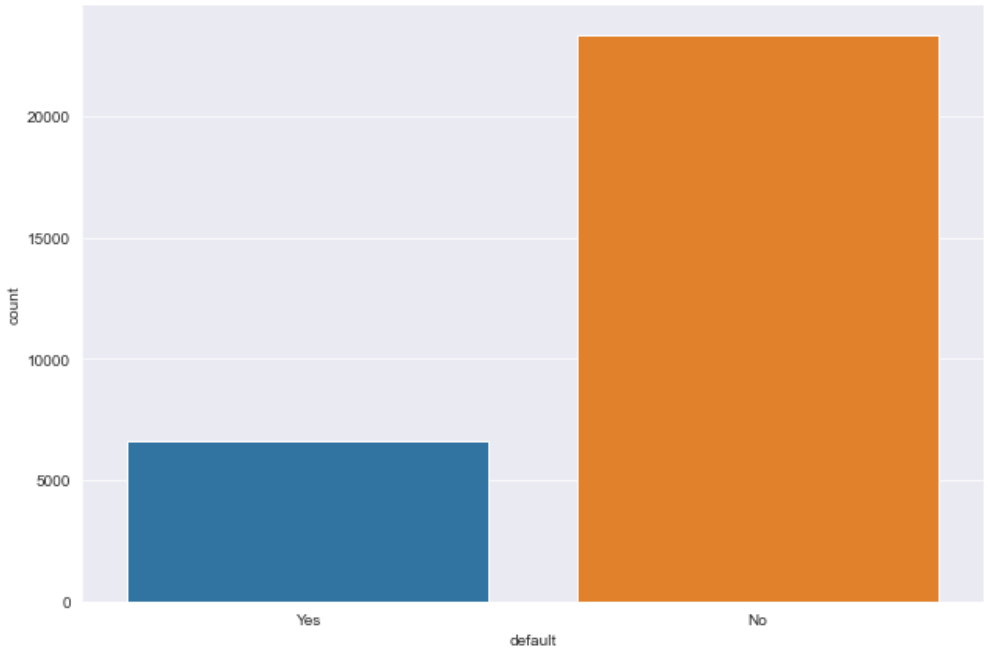
`*default payment next month*`

- 1 = Default
- 0 = No default

Exploratory Data Analysis

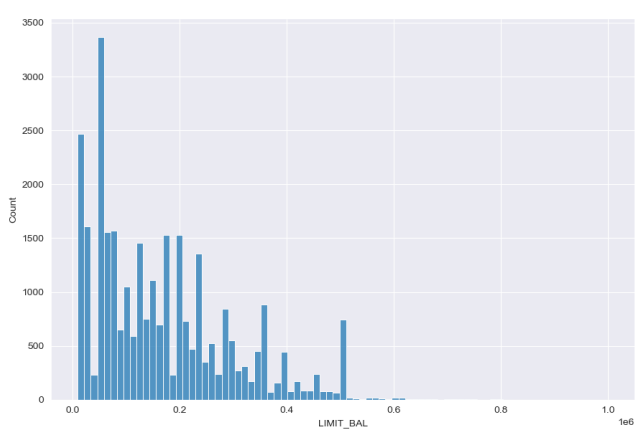
All exploratory data analysis is done on the full dataset. The distributions and conclusions should be consistent between the full, train and test datasets.

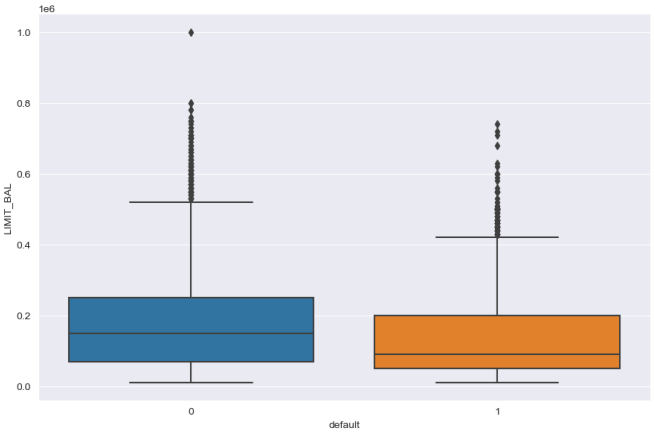
Defaults



There is a significantly lower number of defaulters compared to non-defaulters. This could result in an unbalanced dataset, since defaulting is our target variable.

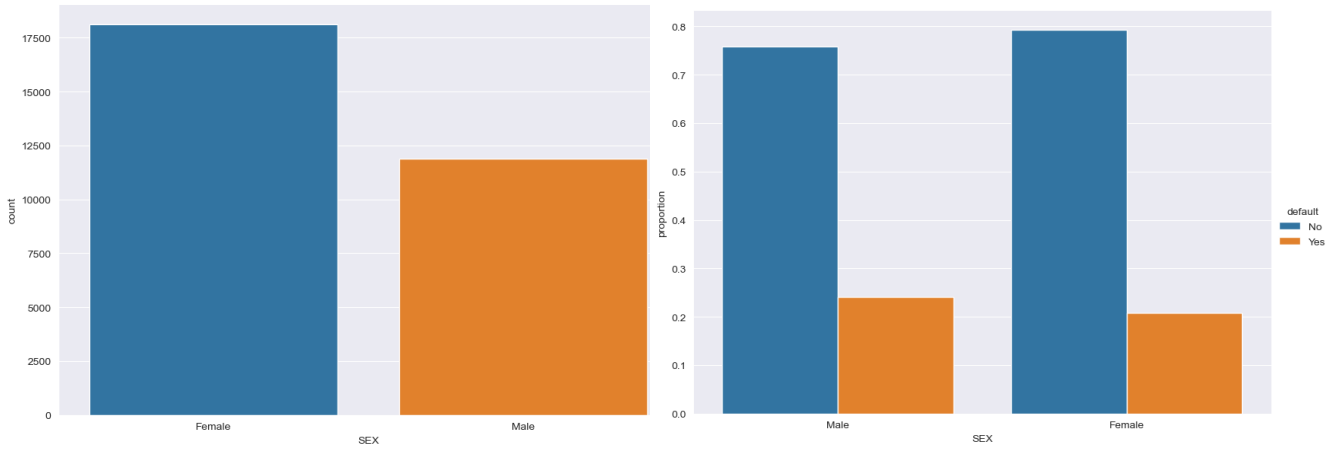
Given Credit (Limit Balances)





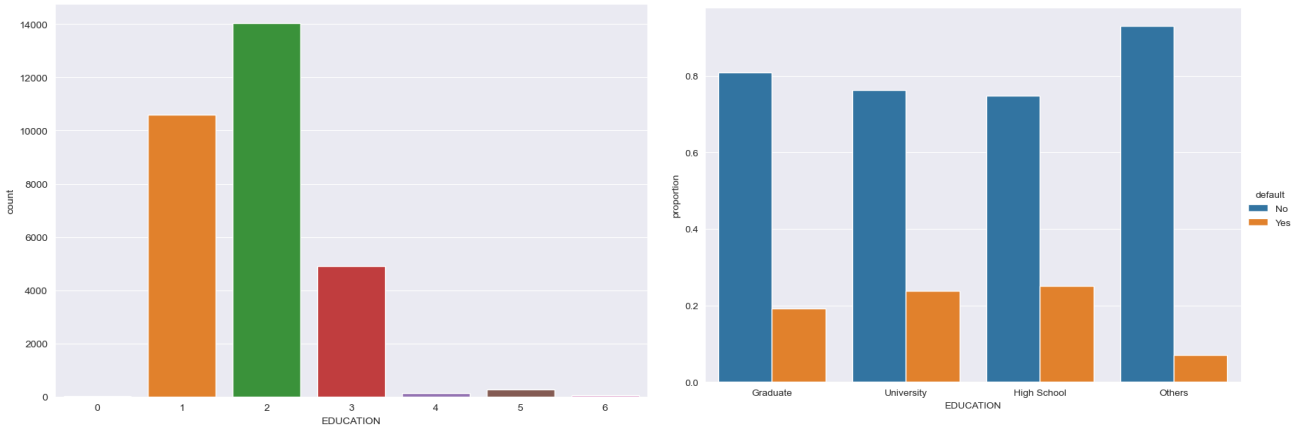
From the histogram, we can determine that the amount of credit given to card-holders is not normally distributed, but right skewed. The boxplot shows that defaulters tend to be given lower levels of credit compared to non-defaulters. This makes sense, as banks would not give risky borrowers more credit to avoid larger financial losses in the event of a default.

Sex



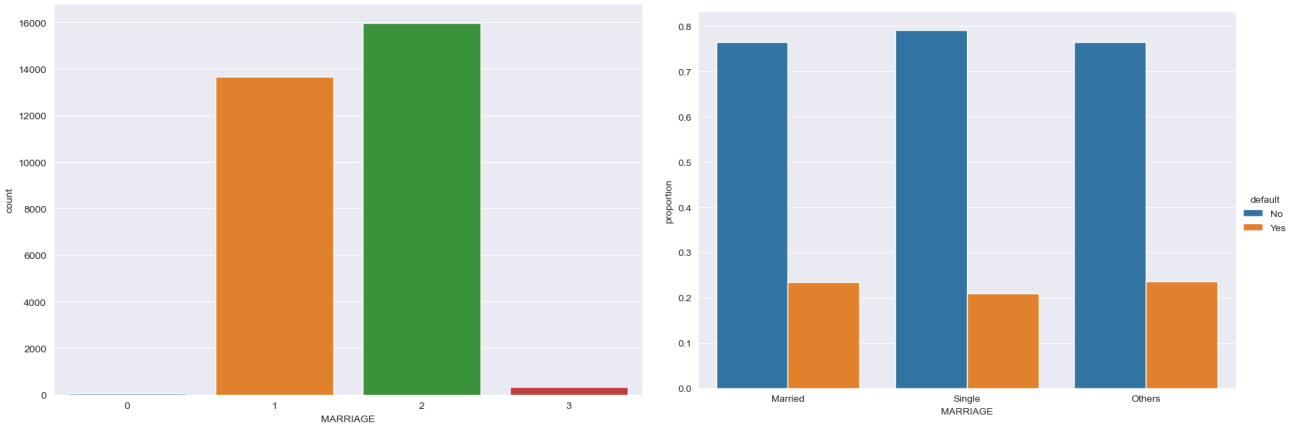
The barplot on the left shows that there are slightly more females than males in the dataset. However, the grouped barplot shows that a larger proportion of males (1) default as compared to females (2).

Education



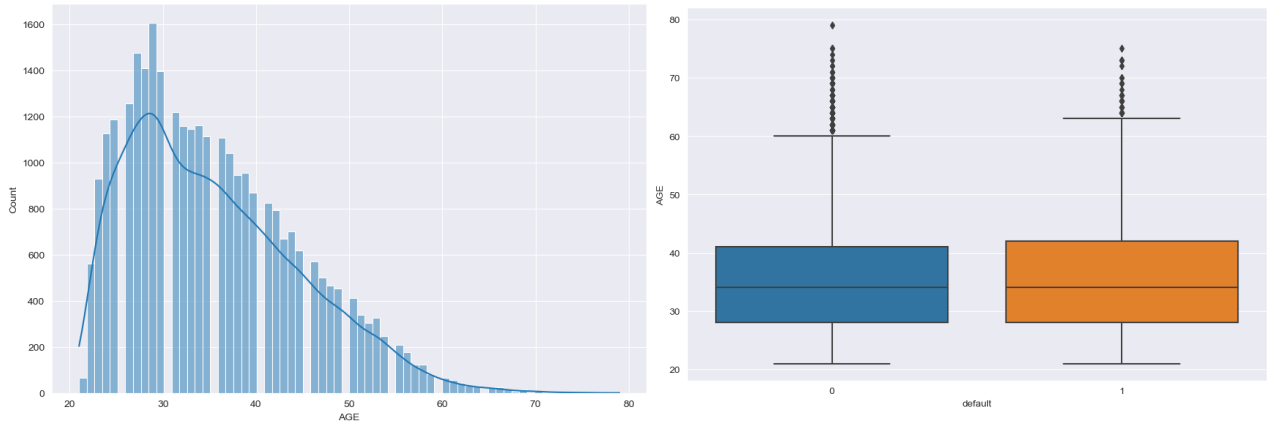
Most of the observations are university students, followed closely by graduate school students. Comparing between the three known education levels (1, 2 and 3), there is a higher rate of default for people with a lower level of education. Thus, education could be a significant independent variable for us to examine.

Marriage



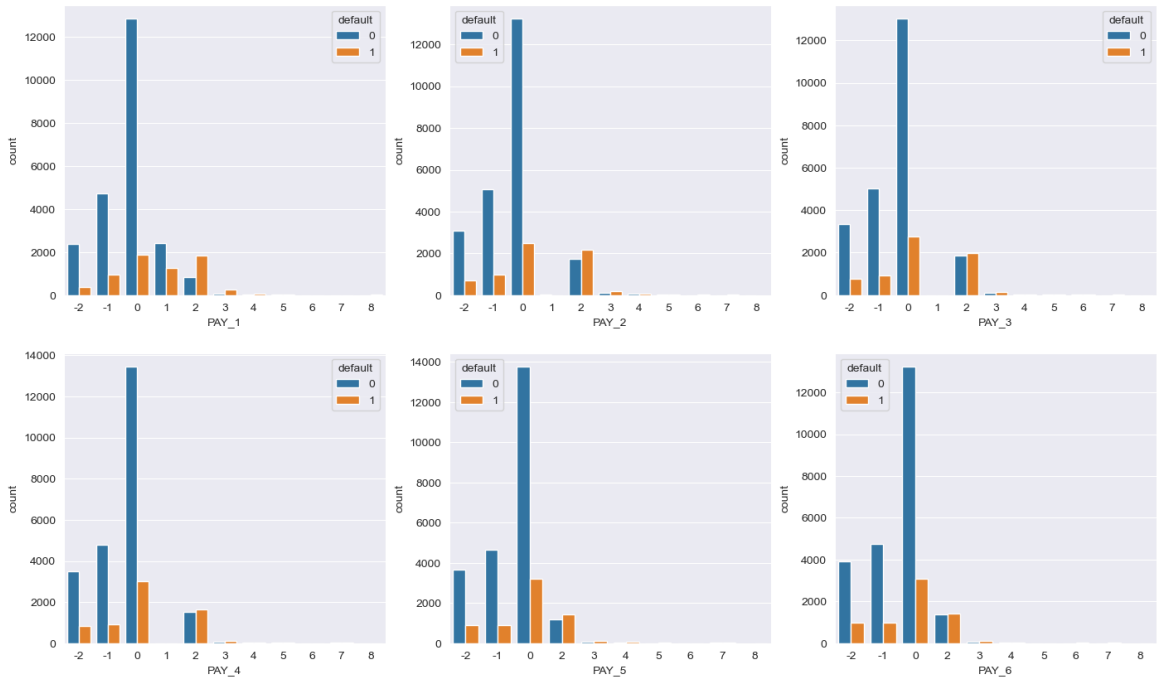
There are slightly more singles in the dataset compared to those who are married. However, those who are married are slightly more likely to default on their payments.

Age



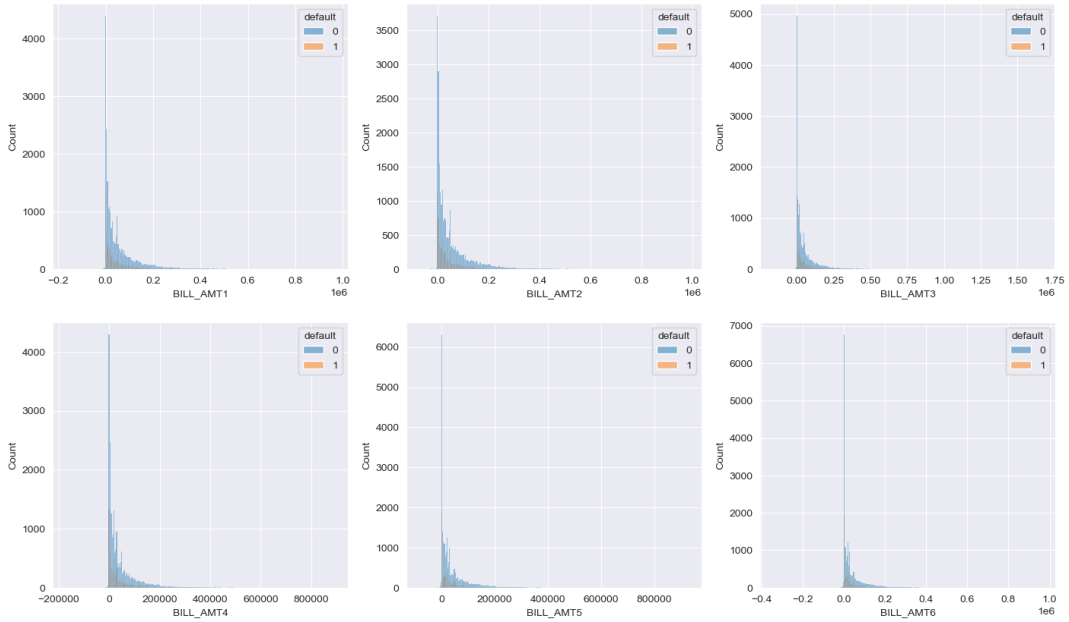
Age seems to be approximately normally distributed, with a slight right skew. Most people are in their late 20s. According to the boxplot, there is no significant difference in distribution of ages for people who default for those who don't.

Payment Timeliness



Majority of non-defaulters pay on time indicated by the 0 value. For each plot, the number of defaults (orange bar) is equal to or higher than the number of non-defaults (blue bar) when the value of the variable is above 0. The opposite happens for values 0 or below. This means that those who pay late are much more likely to be defaulters. As such, we will merge payments that are 1, 2 ... 8 as late, indicated by 1, and payments that are on time , -2, -1, 0, as on time, indicated by 0.

Bill Statements



The distribution of defaults and non-defaults bill statements is approximately even across all months.

Data Pre-Preprocessing

Missing Values

- There are no missing values in the dataset, as seen from the table below.

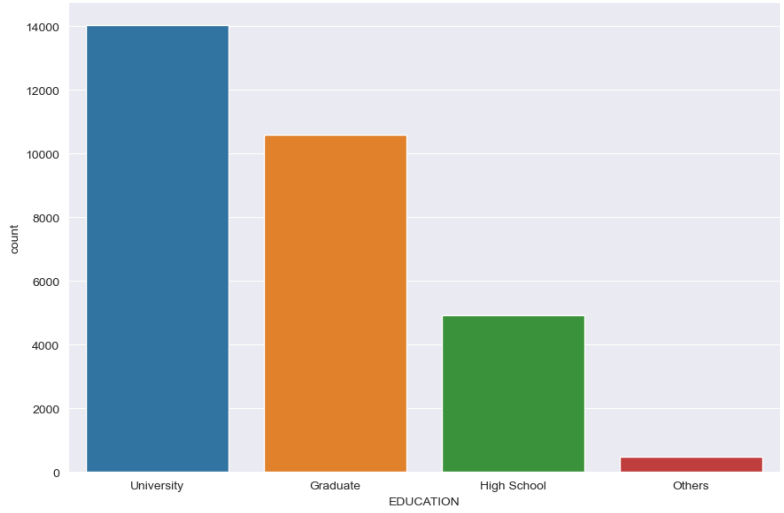
#	Column	Non-Null Count	Dtype
0	LIMIT_BAL	30000 non-null	int64
1	SEX	30000 non-null	int64
2	EDUCATION	30000 non-null	int64
3	MARRIAGE	30000 non-null	int64
4	AGE	30000 non-null	int64
5	PAY_0	30000 non-null	int64
6	PAY_2	30000 non-null	int64
7	PAY_3	30000 non-null	int64
8	PAY_4	30000 non-null	int64
9	PAY_5	30000 non-null	int64
10	PAY_6	30000 non-null	int64
11	BILL_AMT1	30000 non-null	int64
12	BILL_AMT2	30000 non-null	int64
13	BILL_AMT3	30000 non-null	int64
14	BILL_AMT4	30000 non-null	int64
15	BILL_AMT5	30000 non-null	int64
16	BILL_AMT6	30000 non-null	int64
17	PAY_AMT1	30000 non-null	int64
18	PAY_AMT2	30000 non-null	int64
19	PAY_AMT3	30000 non-null	int64
20	PAY_AMT4	30000 non-null	int64
21	PAY_AMT5	30000 non-null	int64
22	PAY_AMT6	30000 non-null	int64
23	default	30000 non-null	int64

Modification of Variables

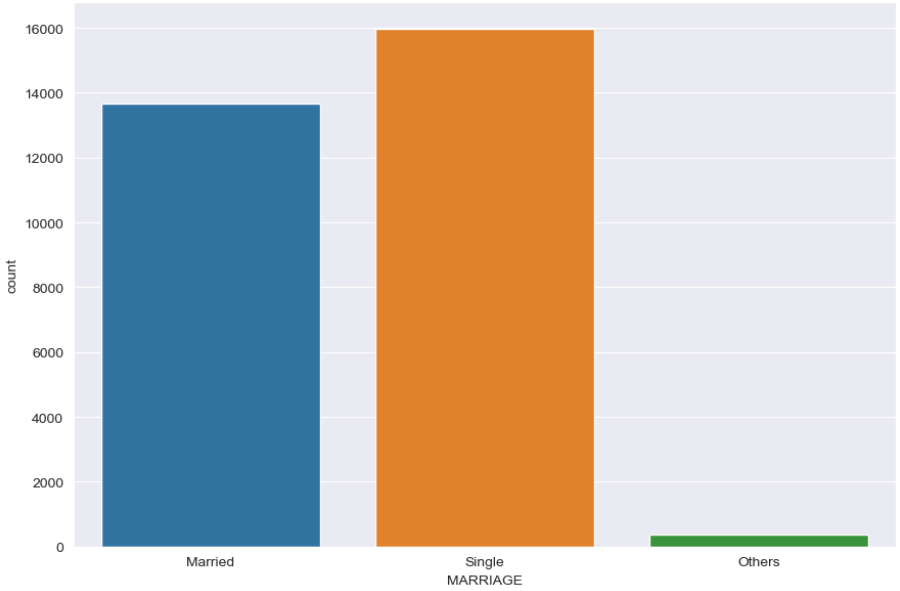
- Renamed target variable `default payment next month` to `default`, for ease of typing, and to prevent errors stemming from the spaces within the variable.
- Renamed `PAY_0` to `PAY_1` for consistency with `BILL_AMT1` and `PAY_AMT1`.
- Grouped payment status into 0 (on time) and 1 (late), as mentioned earlier.

Unknown Values

- For `EDUCATION`, the values of 5 and 6 are written as “unknown” in the dataset description. There also exists values of 0, which is not mentioned. We have classified these three values into 4 (“others”). The updated distribution for the `EDUCATION` variable is as shown:



- For `MARRIAGE`, the value of 0 exists in the dataset, but is not described in the data description. We have classified that into 3 (“others”). The updated distribution for the `MARRIAGE` variable is as shown:



In Preparation for Modelling

- Carried out one hot encoding for the variables with more than two categories, namely `EDUCATION`, and `MARRIAGE`.
- Scaled all variables to be within the range of 0 and 1, as the range of values for each variable differs greatly, which may affect the models’ accuracy.
- Created a training and test dataset with a ratio of 80:20.

Naive Models

As this is a classification problem, we have chosen to focus on these models:

1. Logistic Regression
2. Support Vector Machine
3. Random Forest
4. XGBoost
5. Neural Network

Before feature selection and hyperparameter tuning, we created naive models with the five classification models as shown above. This was done using the pre-processed training dataset with arbitrarily selected hyperparameters. This provides us with a basic understanding of each model, and their respective requirements, in terms of data and hyperparameters. We would also be able to determine whether feature selection and hyperparameter tuning was useful. Elaboration on our models will be under our Model Selection section. For our metrics evaluation, we will be taking Default as the positive class (default = 1) and Non-default as the negative class (default = 0).

Naive Models	Logistic Regression	SVM	Random Forest	XGBoost	Neural Network
Accuracy	81.18%	81.08%	81.55%	81.20%	81.37%
Precision	64.38%	63.46%	62.53%	61.08%	61.64%
Recall	28.99%	29.45%	36.55%	35.93%	36.55%
F1-Score	39.98%	40.23%	46.13%	45.24%	45.89%

Although accuracy is relatively high at above 0.8, we feel that those values do not accurately represent the model performances due to the dataset being imbalanced. Precision values are mediocre at around 0.6, and recall values are very low at around 0.3.

Feature Selection

Using the training dataset, we used 3 feature selection methods to identify important features for each. From these three methods, a final set of features will be chosen to be in our new training dataset. The methods implemented are as follows:

- (1) Filter Method: Chi-2 Dependency Test
- (2) Wrapper Method: Forward Selection
- (3) Embedded Method: Random Forest

(1) Filter Method

These are our hypotheses for the chi-square test, conducted at a 5% level of significance:

- H_0: The target variable ("default") is independent of the categorical variables.
- H_1: The target variable is not independent of the categorical variables.

	feature	score	pvalue
0	SEX	19.328592	0.00001
1	PAY_1	2504.407508	0.00000
2	PAY_2	2336.386495	0.00000
3	PAY_3	1783.040210	0.00000
4	PAY_4	1596.935908	0.00000
5	PAY_5	1567.964929	0.00000
6	PAY_6	1352.401621	0.00000
7	EDUCATION_1	39.102150	0.00000
8	EDUCATION_2	20.157046	0.00001
9	EDUCATION_3	14.052988	0.00018
10	EDUCATION_4	49.763611	0.00000
11	MARRIAGE_1	18.253826	0.00002
12	MARRIAGE_2	16.289254	0.00005
13	MARRIAGE_3	0.415601	0.51914

Based on the chi-square values and the p-values, we choose all categorical features whose p-values are smaller than 0.05.

The total number of features selected is 13, with MARRIAGE_3 being the only variable whose p-value is greater than 0.05.

As all of these features have a p-value of less than 0.05, we can reject the null hypothesis that the target variable ("default") is independent of these 13 categorical features.

Hence, we will consider choosing them as our features for our models.

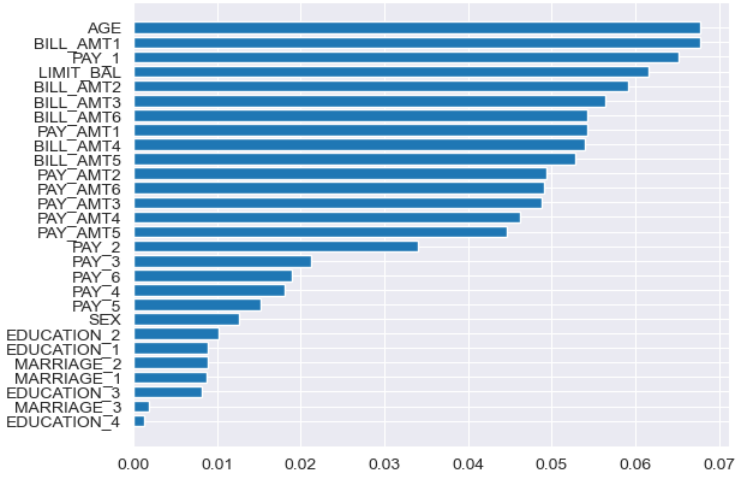
(2) Wrapper Method

The wrapper method that we used is the Forward Selection method using linear regression. The model starts with no predictors and iteratively adds the most contributive predictors. The performance of the model was determined by the r^2 value. We ran the model until the top 10 predictors were selected.

	feature_idx	cv_scores	avg_score	feature_names
1	(3,)	[0.1493453338652141, 0.15603158985458587, 0.10...	0.134645	(PAY_1,)
2	(3, 7)	[0.16689715979474085, 0.18346492163381334, 0.1...	0.157067	(PAY_1, PAY_5)
3	(3, 5, 7)	[0.1776180954004405, 0.1919190131535382, 0.143...	0.166702	(PAY_1, PAY_3, PAY_5)
4	(0, 3, 5, 7)	[0.1816902129537168, 0.195047685075718, 0.1462...	0.170809	(LIMIT_BAL, PAY_1, PAY_3, PAY_5)
5	(0, 3, 5, 7, 8)	[0.18244146637156478, 0.19761495156268338, 0.1...	0.172928	(LIMIT_BAL, PAY_1, PAY_3, PAY_5, PAY_6)
6	(0, 3, 5, 7, 8, 26)	[0.18471736060559452, 0.19664098349476122, 0.1...	0.174311	(LIMIT_BAL, PAY_1, PAY_3, PAY_5, PAY_6, MARRIA...
7	(0, 3, 4, 5, 7, 8, 26)	[0.18419494977466977, 0.19493017522635647, 0.1...	0.175213	(LIMIT_BAL, PAY_1, PAY_2, PAY_3, PAY_5, PAY_6,...
8	(0, 3, 4, 5, 7, 8, 24, 26)	[0.18395368107918797, 0.19614342955272046, 0.1...	0.175969	(LIMIT_BAL, PAY_1, PAY_2, PAY_3, PAY_5, PAY_6,...
9	(0, 1, 3, 4, 5, 7, 8, 24, 26)	[0.18523891652939084, 0.19726953839960093, 0.1...	0.176697	(LIMIT_BAL, SEX, PAY_1, PAY_2, PAY_3, PAY_5, P...
10	(0, 1, 3, 4, 5, 7, 8, 15, 24, 26)	[0.18600923529533686, 0.19801381812057617, 0.1...	0.177418	(LIMIT_BAL, SEX, PAY_1, PAY_2, PAY_3, PAY_5, P...

The top 10 features are LIMIT_BAL, SEX, PAY_1, PAY_2, PAY_3, PAY_5, PAY_6, PAY_AMT1, EDUCATION_4, MARRIAGE_2.

(3) Embedded Method



For the embedded method, we will be using Random Forest to help us select the top 10 features. We plotted the results of the Random Forest. Here, the most important feature is “AGE” and the least important feature is “EDUCATION_4”.

The top 10 features are 'AGE', 'BILL_AMT1', 'PAY_1', 'LIMIT_BAL', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT6', 'BILL_AMT4', 'PAY_AMT1', 'BILL_AMT5'.

Final Selection

To decide on the features to be used in our models, we will be using feature selection method (2) as the tie breaker.

For categorical variables, we will select features that appear in both (1) and (2). These features are 'SEX', 'PAY_1', 'PAY_2', 'PAY_3', 'PAY_5', 'PAY_6', 'MARRIAGE_2', 'EDUCATION_4'.

For continuous variables, we will select features that appear in both (2) and (3). These features are 'LIMIT_BAL', 'PAY_AMT1'.

As such, our final set of selected features are 'SEX', 'PAY_1', 'PAY_2', 'PAY_3', 'PAY_5', 'PAY_6', 'MARRIAGE_2', 'EDUCATION_4', 'LIMIT_BAL', 'PAY_AMT1'.

Models After Feature Selection

The five models were run again using the same hyperparameters after the above features were selected, and their performance evaluated. The summary table is as follows:

	Logistic Regression	SVM	Random Forest	XGBoost	Neural Network
Accuracy	81.05%	80.97%	76.77%	80.82%	81.03%
Precision	63.33%	62.73%	45.08%	60.61%	61.99%
Recall	29.30%	29.45%	34.23%	32.15%	31.69%
F1-Score	40.06%	40.08%	38.91%	42.02%	41.94%

The cells highlighted in red show that a performance metric had worsened, while those in white show no change. Across the board, most metrics have worsened. However, the fall in performance is comparatively small. Furthermore, we have managed to reduce the feature space significantly, which would improve the time required for model training.

SMOTE

We have noticed that the dataset is imbalanced, with a lot less people defaulting compared to those that do not. The skewed class proportions could affect the accuracy and parameter learning of our models. To balance the dataset, SMOTE was employed. Synthetic samples were generated for the minority class (non-defaulters), to balance our training data and allow us to evaluate our model predictions with greater accuracy. The final counts for each class in the target variable is shown here.

```
y_SMOTE.value_counts()

0.0    18661
1.0    18661
```

Model Selection

Following the feature selection, we applied Synthetic Minority Oversampling Technique (SMOTE) before tuning the hyperparameters of each of our models. Hyperparameter tuning was done using GridSearchCV in the sklearn library, and the best hyperparameters were selected using the accuracy metric.

The final models (after feature selection, SMOTE, and hyperparameter tuning) were used to determine the ability of each model type in predicting the target variable. A confusion matrix was created for each model, and a summary of evaluation metrics were provided for each model across every improvement stage. A light green cell denotes a slight increase of 2% or less in model performance over the previous stage, while a dark green cell denotes a strong increase of more than 2% instead. Red cells are similar, except for decreases in model performance.

Logistic Regression

Logistic Regression is used to predict an outcome (default or non-default) based on selected independent variables. The logistic regression model computes the sum of the selected features by estimating their coefficients, and calculates the log-odds of the result using the Sigmoid function.

Hyperparameter Tuning

The hyperparameters tuned are `C`, `penalty`, and `solver`, of which 10, “l2”, and “newton-cg” were selected respectively. `C` represents the inverse of regularisation strength, and in this case, increasing it specifies weaker regularisation. `penalty` represents the penalty norm, which does not change from the default, but our `solver`, which is the algorithm to use in the optimization problem, is tuned to be “newton-cg”.

	Predicted Default	Predicted Non-default
Actual Default	759	538
Actual Non-default	849	3854

Overall Evaluation Metrics

	Naive	Selected Features	SMOTE	Tuned
Accuracy	81.18%	81.05%	76.97%	76.88%
Precision	64.38%	63.33%	47.33%	47.20%
Recall	28.99%	29.30%	58.13%	58.52%
F1-Score	39.98%	40.06%	52.18%	52.26%

With every additional improvement methodology done, we see a general trend in the performance metrics. Both accuracy and precision have a decreasing trend, with the latter having a greater decrease. However, recall has increased greatly, especially after SMOTE. This is expected, as the imbalanced dataset has overemphasised the negative class (non-default), as explained earlier, resulting in skewed metrics (precision being too high while recall being too low). The balancing done through SMOTE gives us a better picture of the model performance across all classes. The F1 metric, being a combination of both precision and recall, has increased, showing that the model displayed an overall improvement despite having a lower precision level.

With regards to feature selection and hyperparameter tuning, we see only slight changes, showing that they are not of much importance in this logistic regression model.

Support Vector Machine (SVM)

SVM aims to classify observations by finding an optimal hyperplane in the N-dimensional space (N refers to the number of features), that maximises the margin of separation.

Hyperparameter Tuning

The hyperparameters tuned are `C` and `kernel`, of which 100 and “rbf” were selected respectively. `C` represents the regularisation parameter, where the strength of regularisation is inversely proportional to it. An increase in `C` indicates weaker regularisation. `kernel` is the kernel type used by the algorithm. It did not change.

	Predicted Default	Predicted Non-default
Actual Default	788	509
Actual Non-default	962	3741

Overall Evaluation Metrics

	Naive	Selected Features	SMOTE	Tuned
Accuracy	81.08%	80.97%	75.43%	75.48%
Precision	63.46%	62.73%	45.01%	45.03%
Recall	29.45%	29.45%	61.60%	60.76%
F1-Score	40.23%	40.08%	52.02%	51.72%

Feature selection had slight effects on model performance, with small decreases across the board except for recall which stayed the same. SMOTE followed the trend seen in logistic regression, with large decreases in accuracy and precision, with recall increasing greatly. Overall, F1-score increased, showing that the model improved in general. After hyperparameter tuning, model performance changed slightly, but we deem it to be insignificant, possibly due to randomness.

Random Forest

Random Forests uses a large number of decision trees, each built with a subset of the data and features. The model then aggregates the predictions made by each tree, which reduces variance. Thus, Random Forests is less prone to overfitting, therefore increasing the accuracy of its predictions.

Hyperparameter Tuning

The hyperparameters tuned are `max_depth` and `n_estimators`, of which 5 and 500 were selected respectively. `max_depth` is the maximum depth of the tree, and the parameter 5 was chosen from a range of [2, 5]. `n_estimators` indicates the number of trees in the forest, and 500 was chosen from {10, 50, 100, 250, 500}, an increase from the default of 100.

	Predicted Default	Predicted Non-default
Actual Default	768	529
Actual Non-default	877	3826

Overall Evaluation Metrics

	Naive	Selected Features	SMOTE	Tuned
Accuracy	81.55%	76.77%	68.78%	76.57%
Precision	62.53%	45.08%	35.14%	46.69%
Recall	36.55%	34.23%	52.51%	59.21%
F1-Score	46.13%	38.91%	42.10%	52.21%

Feature selection decreased model performance across the board. SMOTE caused large decreases in accuracy and precision, with recall increasing greatly. Overall, F1-score increased, showing that the model improved in general. Hyperparameter tuning greatly improved the final model, with large metric increases across the board.

XGBoost

Extreme Gradient Boosting (XGBoost), like Random Forests, also uses a large number of decision trees to make predictions. However, each decision tree is weak, but is iteratively added to the model to improve its overall performance. Regularisation techniques are used to prevent overfitting and improve accuracy.

Hyperparameter Tuning

The hyperparameters tuned are `max_depth` and `n_estimators`, of which 5 and 500 were selected respectively. Similar to the parameters of Random Forest, `max_depth` is the maximum depth of the tree, and `n_estimators` indicates the number of trees in the forest. The chosen parameters were chosen from the same range of values given.

	Predicted Default	Predicted Non-default
Actual Default	693	604
Actual Non-default	968	3735

Overall Evaluation Metrics

	Naive	Selected Features	SMOTE	Tuned
Accuracy	81.20%	80.82%	74.73%	73.80%
Precision	61.08%	60.61%	43.42%	41.72%
Recall	35.93%	32.15%	55.74%	53.43%
F1-Score	45.24%	42.02%	48.82%	46.86%

Feature selection made model performance worse off, with decreases across the board. SMOTE caused large decreases in accuracy and precision, with recall increasing greatly. Overall, F1-score increased, showing that the model improved in general. After hyperparameter tuning, model performance decreased slightly.

Neural Network

The neural network used here is a Multilayer Perceptron (MLP). The MLP model used has two hidden layers, with binary cross entropy as the loss function and Adam as the optimiser.

Hyperparameter Tuning

The hyperparameters tuned are `learning_rate`, `neurons_1` and `neurons_2`, of which 0.01, 8 and 8 were selected respectively. ‘learning_rate’ represents the rate at which the model updates itself every iteration, ‘neurons_1’ and ‘neurons_2’ represent the number of hidden units in the first and second hidden layers of our neural network respectively.

	Predicted Default	Predicted Non-default
Actual Default	749	548
Actual Non-default	882	3821

Overall Evaluation Metrics

	Naive	Selected Features	SMOTE	Tuned
Accuracy	81.37%	81.03%	75.45%	76.17%
Precision	61.64%	61.99%	44.95%	45.92%
Recall	36.55%	31.69%	60.37%	57.75%
F1-Score	45.89%	41.94%	51.53%	51.16%

Feature selection made model performance worse off, with decreases across the board except for precision which increased slightly. SMOTE caused large decreases in accuracy and precision, with recall increasing greatly. Overall, F1-score increased, showing that the model improved in general. After hyperparameter tuning, accuracy and precision increased slightly, but recall decreased to a larger degree. This resulted in a slight drop in F1-Score.

Model Evaluation

Summary Table

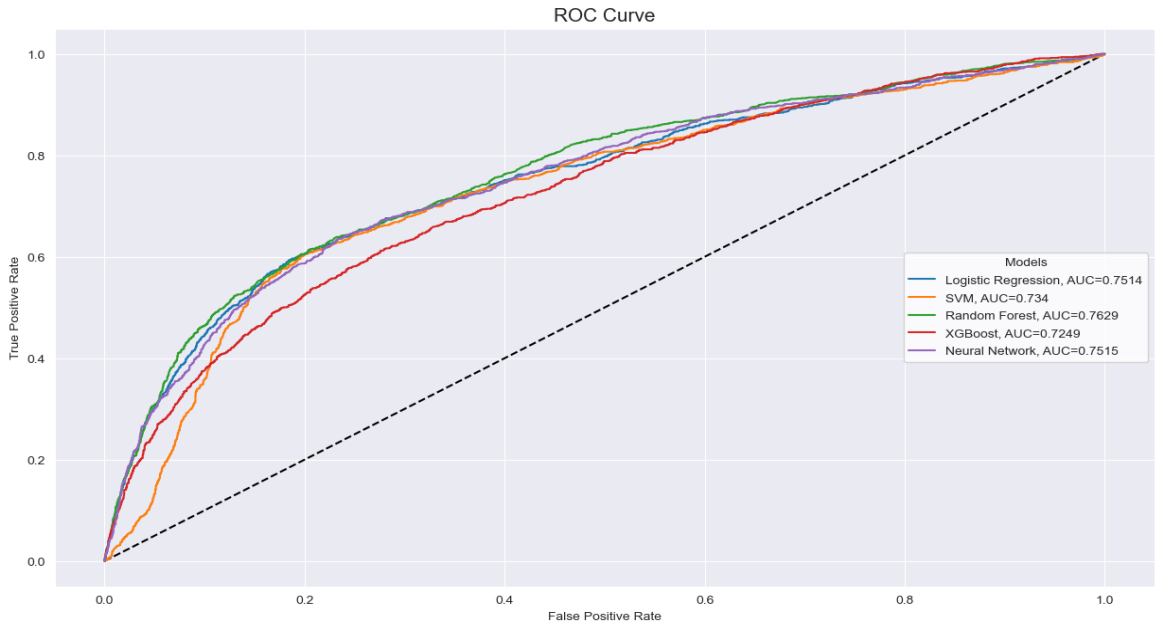
Below is a summary table of all final models (after feature selection, SMOTE, and hyperparameter tuning). An expanded list of metrics were used to evaluate them. For each metric, the best model is highlighted in green, while the worst model is highlighted in red.

	Logistic Regression	SVM	Random Forest	XGBoost	Neural Network
Accuracy	76.88%	75.48%	76.57%	73.80%	76.17%
Precision	47.20%	45.03%	46.69%	41.72%	45.92%
Recall	58.52%	60.76%	59.21%	53.43%	57.75%
F1	52.26%	51.72%	52.21%	46.86%	51.16%
AUC	75.14%	73.40%	76.29%	72.49%	75.15%
Average Class Accuracy	70.23%	70.15%	70.28%	66.42%	69.50%

Recall is generally higher than precision across the board. Assuming that SMOTE is unbiased, this means that there are more false positives than false negatives by each model. With regards to the dataset, more people are predicted to default on their debts but actually do not, compared to people predicted to not default but end up defaulting.

ROC Curve

The ROC curve can be seen below. The area under the ROC curve (AUC) has been provided in the summary table above.



Model Comparison

Best Model: Random Forest

The best model would be either logistic regression or Random Forest, depending on what the most important metric is, be it F1, AUC, or average class accuracy respectively. All three are robust metrics for overall model performances, and they are rather similar across these two models.

We have chosen AUC and Average Class Accuracy to be the metrics that determine our final chosen model. This is because AUC represents the area under the ROC curve and is easily interpretable, wherein a higher AUC represents a model with higher classification power. On the other hand, Average Class Accuracy is a powerful metric that evaluates the model's performance based on the recall of each target class, giving insights to specific classes in which the model may be struggling to make accurate predictions.

We have decided to choose Random Forest as the best model, because it has the highest AUC and Average Class Accuracy value. This means that it performs the best on average for both classes.

Besides taking business needs into account, Random Forest is also highly scalable, in terms of larger amounts of data, or higher number of variables - both categorical and continuous. This is important in today's digital age, where information is collected in larger quantities.

Worst Model: XGBoost

We can easily tell that XGBoost seems to be the worst model to use, with the poorest performance across all metrics by far.

Areas of Improvement

Dataset Issues

There are various issues within the dataset provided.

With regards to the target variable, the dataset is imbalanced, with a default:non-default ratio of around 1:4, which may lead to various difficulties in creating a robust model. Firstly, the models might be biased towards the majority class (non-default), and thus perform poorly on the underrepresented minority class. This will lead to overfitting to the non-default samples, and poor performance on the defaults, as shown in the low recall scores of the naive models.

There are also missing or improperly labelled values within the dataset. For example, with regards to payment timeliness, there are unlabelled values of -2 and 0. Unknown values are also in `EDUCATION` (5 and 6) and `MARRIAGE` (0). These values might have provided meaningful information had they been encoded properly.

Use of SMOTE

To resolve the issue of an imbalanced dataset, we have resorted to using SMOTE. However, SMOTE may create noise when synthetic samples are created that do not accurately represent the minority class, which leads to poorer model performance. Should the minority class (default) be highly skewed, SMOTE may have oversampled those outlier samples, resulting in lower predictive accuracy.

Alternative Evaluation Metrics

We used accuracy as the main metric to do hyperparameter tuning, as we believe that issues regarding the imbalanced dataset have been mostly resolved with SMOTE. However, as mentioned earlier, SMOTE does have some limitations, which may prompt us to use more robust metrics such as F1 score instead. In the real world, false positives and false negatives may have different costs to the bank, thus our models have to be adjusted to that fact as well. For evaluation, we could focus on recall, as the costs incurred by a defaulting customer predicted as non-defaulting would be much higher than vice versa. Models can also be updated with a loss matrix stating the weights put on false positives or negatives.

Improvement of Data Collection

Other relevant variables can be collected to aid prediction, especially time-invariant user-specific data. Such variables could include credit score, debt levels, income level or housing type. They could have high explainability when predicting the chances of defaulting, which is likely to increase the predictive power of our models.

Conclusion

The default risk of any individual is an important piece of information for lenders, especially banks, when offering credit. It is crucial for those financial entities to make the correct predictions, and even a percentage decrease in accuracy would cost them millions of dollars in lost profit. To create a model to do so accurately, we have first pre-processed the data, ensuring that all variables are encoded properly, carried out scaling and one-hot encoding. Secondly, we have selected relevant features that have high predictive power, and disregarded those that do not. Third, the dataset was balanced using SMOTE. Fourth, hyperparameter tuning was done on each of the five models. Finally, the performances of all the five final models were evaluated and compared. We believe that Random Forests is the best model, as it is comparable to the other two better models, yet is highly scalable with regards to more observations and variables.