# Predicting the issuing of defaults to credit users through financial and demographic factors

Module Name: COMP2261

Date: 18/1/2023

Group Number : 23

Submitted as part of the degree of MEng Computer Science G400 to the

Board of Examiners in the Department of Computer Sciences, Durham University

**Abstract** — Receiving a default notice can have a significant impact on a credit user's ability to borrow money. However, financial history may not be sufficient to determine the likelihood of receiving a default, as demographic analysis reveals signs of systemic issues which could have an unjust impact on this probability. This report investigates whether it is possible to accurately determine a customer's chances of receiving a default notice through joint consideration of financial history and demographic factors.

**Index Terms**— Binary Classification, Default credit, Machine Learning, Sci-kit Learn

## 1 Introduction

A customer's likelihood of receiving a default notice on their credit account can be difficult to predict, as the exact number of missed payments needed to instigate a default is variable, depending on the terms of the lender and their perception of the borrower [1]. As such, it is likely that both financial history and demographic factors will affect a customer's likelihood of receiving a default notice, a hypothesis confirmed by initial investigation of the dataset. As a default can stay on a customer's credit file for six years, negatively impacting their ability to borrow money [1], being able to predict its arrival would be exceptionally useful, enabling the customer to adjust their financial habits in hopes of avoiding the notice altogether.

This project therefore aims to create a machine learning model which will improve at predicting the likelihood of a customer receiving a default payment**,** with respect to the percentage of correctly predicted default payments in the test set, based on its experience learning from a 2005 Taiwanese dataset of credit customers' financial records.

The data inputted will consist of the most relevant financial information from each instance, along with 3 demographic attributes: sex, education level, and the customers marital status. These attributes will likely have an impact on the probability of receiving a default, due to gender inequality, class divides represented by education, and possible homophobia represented by marital status (as same-sex marriage was illegal at the time). The correctness of the default values outputted by our model will be evaluated through a range of metrics, hopefully proving a link between financial history, demographic attributes, and likelihood of receiving a default.

## 2 Data Preparation and Exploratory Analysis

The dataset consists of 30,000 instances, each representing a credit card customer. It covers 24 attributes, providing categorical data about the customers' demographics and financial information (account balance, monthly bills, and repayment status). The binary response variable we aim to classify is called *default payment next month*, already encoded in the given data as '0' for non-defaulters and '1' for defaulters. The dataset is notably large, meaning the risk of overfitting is lowered significantly as there are more examples to learn from, allowing the model to generalise better to new data [2]. With more data, patterns and trends can be identified, meaning the model is less likely to make predictions based on outliers in the training data and will most likely perform better on unseen data [2]. In this case, we did not need to subsample the dataset, as the Pandas library is efficient in handling data up to 1GB in size [3].

At first glance, the data appears to be already cleaned, as there exist no NULL values within the dataset. However, closer inspection reveals several irregularities. There exist values in the attributes *MARRIAGE* and *EDUCATION* that do not correspond to the description provided by the UCI website. Specifically, there were undocumented values of 0 in the *MARRIAGE* column and undocumented values of 0, 5, and 6 in the *EDUCATION* column. The presence of these unknowns can be addressed in two ways. Firstly, eliminating rows or columns that contain these values, which risks significant loss of data, introducing bias. Secondly, replacing the data with estimated values, which allows for retention of all instances but can also introduce errors and biases [4]. As rows containing unknowns comprise only 0.23% of the

dataset, as shown in Figure 1, we felt it was justified to delete these rows, as doing so should have very little impact on the models.

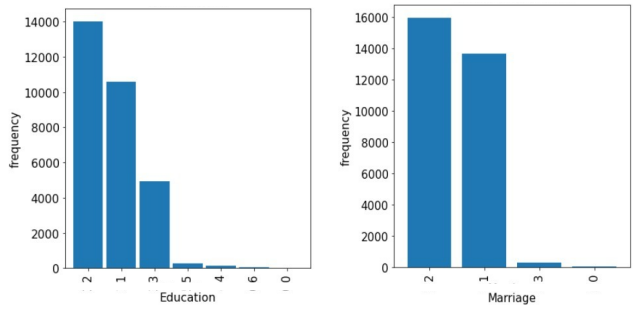


**Fig 1.** Distribution of discrete attribute values

The *PAY_AMT* attributes are described on the UCI website as being on a scale of -1 to 9, excluding 0. However, the observed minimum and maximum values were found to be -2 and 8, respectively. Therefore rescaling is necessary. In this instance, we added 1 to each attribute and combined 0 and -1 values in order to align the data with the intended measurement scale, preserving the relationships between values. We made this decision under the assumption that 0 and -1 values represent overlapping groups. By reducing the number of categories, we were able to simplify the data, improving the robustness of results. Additionally, for clarity, we have renamed *PAY_0* to *PAY_1* and our target variable *default payment next month* to *Default*.

The ranges of our features vary significantly. This risks skewing the results of our algorithms, especially for scale variant models such as Support Vector Machines. Therefore, we applied a Min-Max Scaler to all numerical attributes in the range of 0-1. We selected normalisation over standardisation as it maintains the distribution of the data. [5] The equation used to apply this Scalar is below:

$$X_{\text{scaled}} = (X - X_{\min}) \,/\, (X_{\max} - X_{\min})$$

One-Hot Encoding restructures categorical data into a format suitable for machine learning models, leading to improved prediction accuracy [6]. The features *SEX*, *EDUCATION*, and *MARRIAGE* were originally integer-coded, but this can introduce nominal or ordinal relationships in the data that would not otherwise exist [6]. One-hot encoding was used to convert these integers into binary values, with each column representing a true/false state for each possible attribute value.

Figure 2 shows box plots with scaled data for each continuous attribute. This enables easy comparison, aiding in the process of feature selection. It makes evident that the dataset contains many outliers, particularly for the *PAY_AMT* feature set. We have decided to retain the outliers rather than remove them from the data, for several reasons. Outliers are often valuable sources of information, providing insights into unexpected patterns in the data. Removing a large number of outliers may result in the loss of a significant amount of information, which would undermine the validity of the analysis.

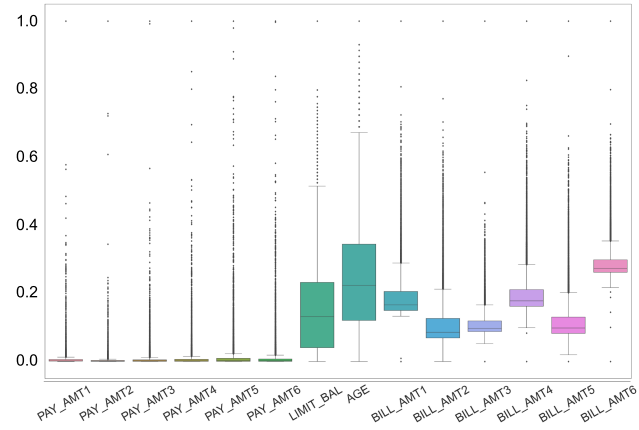


**Fig 2.** Box plot mapping for scaled attributes

The dataset was divided into a training set and a testing set using an **80/20** split, which is commonly used in the machine learning industry [7]. The random state was set to 42 to allow for reproducible results when testing different models. To prevent data leakage from the testing set, the dataset was split using a stratified sampling method, ensuring that the class balance was preserved in both sets. The training and testing set were preprocessed separately when implementing the scaler, encoding and feature selection, to avoid data leakage.

Figure 3 presents a correlation matrix. It measures the linear relationship between pairs of variables, calculated using **Pearson's correlation coefficient (PCC)**:

$$P_{x,y} = COV(X, Y)/\sigma_x \, \sigma_y \in [1, \; -1] \subset \Re$$

Where COV is the covariance and σ is the standard deviation. We used PCC over the Chi-Squared metric as the latter struggles with a large number of attributes [8].

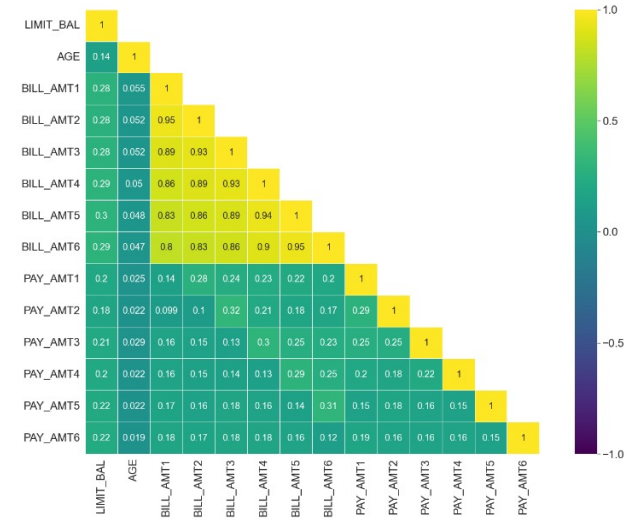


**Fig 3.** Pearson correlation coefficient matrix of each attribute

As shown in Figure 3, all *BILL_AMT* features were found to be strongly linearly correlated with a high PCC > 0.92. As binary classification algorithms assume independence between features [9], we decided to remove the *BILL_AMT2 - BILL_AMT6* feature set.

The dataset is still highly dimensional, posing a risk of the 'curse of dimensionality' problem, hindering generalisation performance and model training speed. To reduce the dimensionality of the dataset, we have applied **Principal Component Analysis (PCA)**. This technique is commonly used to reduce multicollinearity among features, mitigating the risk of overfitting. PCA is applied only to the training data to avoid data leakage. Figure 4 outlines the amount of variance captured by the first 13 **Principal Components (PC)** out of 29. As Figure 4 shows, the first 8 PCs capture almost the entirety (97.5%) of the variance in the data. Therefore we have selected 8 PCs to maximise retention of information, whilst minimising the risk of overfitting.

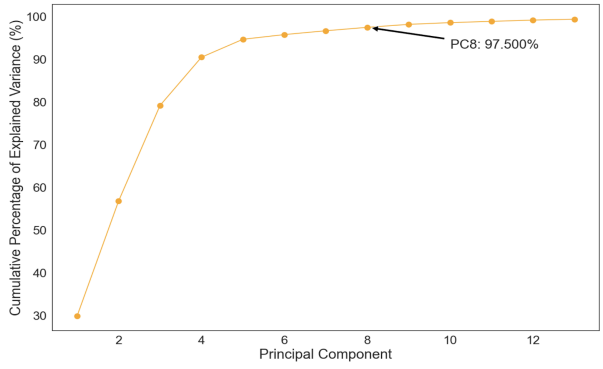


**Fig 4.** Cumulative Percentage Variance for the first 13 PCs

The dataset is significantly imbalanced, with only 22.1% of the instances invoking a default. This imbalance can make it difficult for machine learning algorithms to effectively learn all classes, leading to a biassed training process. To address this issue, we have applied the **Synthetic Minority Over-sampling Technique (SMOTE)** to the dataset. SMOTE synthesises new minority class samples in the feature space rather than simply oversampling the existing minority class samples. This helps reduce the risk of overfitting, improving the model's generalizability.

## 3 Learning Algorithm Selection

The algorithms selected need to be suitable for a supervised binary classification task - in our case, the prediction of a default payment. It is also important to avoid overfitting, as the finished model will be used for predicting new, unknown values. This rules out Logistic Regression and Decision Trees as candidates, due to their tendency to overfit when trained with large datasets such as ours. [10] Multinomial Naive Bayes was also rejected, as it performs best with discrete data and many of our attributes are continuous. Two models we decided on are Support Vector Machine and Random Forest. These are known to score well with regards to cost functions on a test set, and are relatively fast which will afford us more time to fine-tune the parameters. The final model will be K-Nearest Neighbors, which is also suitable for binary classification, but is said to show overfitting tendencies with large datasets [10]. Our inclusion of this model should serve as a test of its true capabilities, through comparing its results with those of our other models.

## 4 Model Training and Evaluation

We will follow a systematic process of training and evaluating each model using only the training data set. **Grid search** will be used to investigate a wide range of hyperparameters, and we will plot the training data vs test data to reveal whether the model is overfitting or underfitting the data. Grid search was chosen over random search as it is more exhaustive, and therefore more likely to find the optimal set of hyperparameters. By analysing the training and testing scores in relation to various hyperparameters, we can determine which ones are most significant and adjust them as needed. We will also examine the range of values associated with each hyperparameter in greater detail to identify the behaviour, signs of convergence or any interactions between them. We will utilise a **10-fold stratified cross-validation** method widely used in machine learning. This involves dividing the training set into 10 stratified folds, using 9 of them for training and 1 for testing. This approach is more effective than a singular training/validation/test split, as it provides better generalisation to new data and reduces overfitting.

Given our imbalanced dataset, evaluating the performance of a classifier using accuracy is misleading, as the classifier may achieve high accuracy by simply predicting the majority class all the time. This would not provide a reliable indication of the classifier's performance on the minority class. Which in our case is the focus of the classification task; to accurately identify clients who will default. Instead, precision, recall, and F1 score provide a more comprehensive evaluation. With identifying defaults, it is important to minimise false negatives, as failing to correctly identify customers who will default can have serious consequences for the credit card issuer. Therefore, we will prioritise recall over precision. As we aim to minimise both false positive and false negative predictions, we will tune each model using the **AUC-ROC metric**. It is a useful evaluation metric for imbalanced datasets, where one class is rare, as it allows us to visualise the model's performance at different classification thresholds. This helps us to understand how well the classifier is able to distinguish between positive and negative instances, and to identify the optimal classification threshold for making predictions.

### 4.1 Support Vector Machines (SVM)

Support Vector Machines are a machine learning model widely used for classification tasks. The goal of the SVM is to find the hyperplane with the greatest margin between the different classes, which is achieved through the use of a cost function and an optimization algorithm. To avoid overfitting in SVMs, we can use soft margins, which allow for some errors in classification using Lagrangian optimization. By adjusting the values of the Lagrange multipliers, SVM is able to find a hyperplane that maximises the margin while still allowing for some errors in classification. The hyperparameters we will tune will be the kernel type (linear, polynomial, or radial basis function) and kernel parameters (such as gamma value), as well as the C regularisation parameter. These

hyperparameters control complexity and balance the trade-off between the margin and misclassified points.

From Figure 5, it is clear that using a kernel allows for a wider range of scores and more flexibility. However, all box plots show some degree of overfitting, with the highest occurring for polynomial kernels.

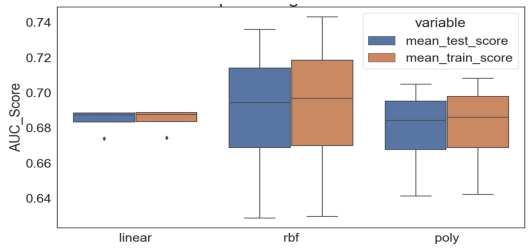


**Fig 5.** Boxplots displaying Kernels plotted against AUC score

The 'linear' kernel shows the least improvement or change in performance when varying hyperparameters. Therefore, we have chosen to use the 'rbf' kernel, as it demonstrates the greatest improvement in scores, ranging from 0.69 to 0.71. Despite being more prone to overfitting, we can fine-tune the 'rbf' kernel using the hyperparameter *gamma*, as shown in Figure 6. As the value of *gamma* increases, both training and test scores improve. However, the rate of overfitting between the two scores increases as well. This trend does not converge within the tested range of *gamma* values, therefore it may be worthwhile to investigate even higher values to find values of convergence and maximise performance.

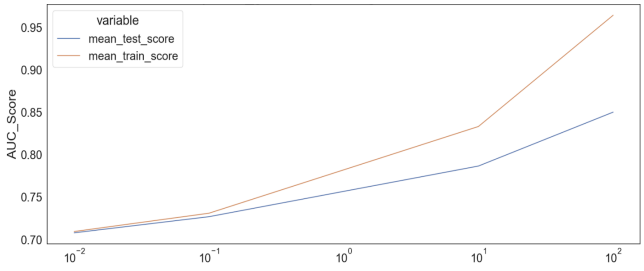


**Fig 6.** Value of *gamma* plotted against AUC score

It is necessary to consider the trade-offs between optimising the model's performance, the computational cost, and risk of overfitting. Thus, we will first examine the effect of adjusting the hyperparameter *C* before considering further increases in the value of *gamma*.

Figure 7 shows that as the value of *C* increases, both training and test scores also increase, albeit at a slower rate compared to the increase in *gamma*. However, there is less overfitting at higher values of *C*. The hyperparameter *C* controls the regularisation strength in an SVM model, with larger values of *C* resulting in a more complex model that is more prone to overfitting. It is important to note that increasing both *C* and *gamma* values add more complexity and require more computational resources. In order to strike a balance between model optimization and computational cost, our chosen values were *C*=1000 and *gamma*=10.

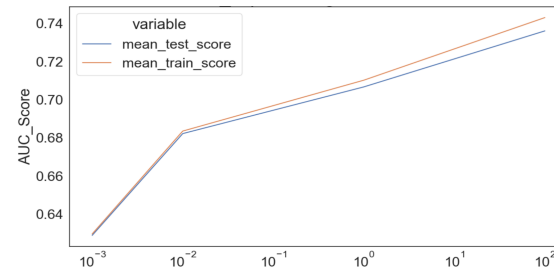


**Fig 7.** Value of *C* plotted against AUC score

## 4.2 K-NEAREST NEIGHBORS (KNN)

The K-Nearest Neighbors algorithm is a memory-based learning algorithm which uses a distance metric to compare data instances. When used for classification tasks, an unlabelled instance is assigned the label most common among its *k* nearest neighbors [11].

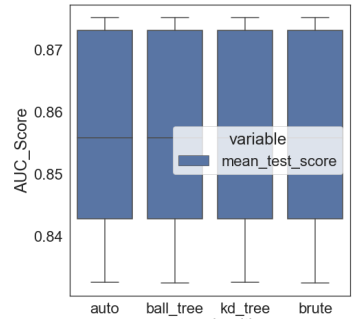


**Fig 8.1.** Values for *algorithm*

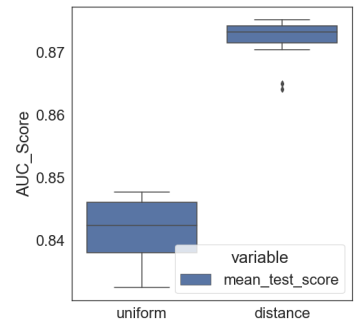


**Fig 8.2.** Values for *weights*

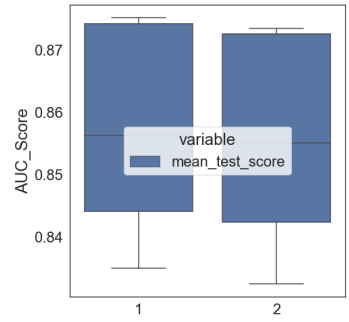


**Fig 8.3.** Values for *p*

Its most relevant hyperparameter is *n_neighbors*, which determines the value of *k*. Before fine-tuning a continuous hyperparameter such as this one, grid search was first used with a variety of *n_neighbors* values to determine the best values for other, discrete hyperparameters. These discrete hyperparameters are *algorithms* (determines the data structure used to search through the *k* neighbors), *weights* (adjusts the influence each neighbor has on the final instance label), and *p* (power parameter for the Minkowski metric used for distance computation).

Figure 8.1 shows little-to-no difference in model performance for each search algorithm. Therefore, the default value of 'auto' was kept.

Figure 8.2 shows a notable difference in performance between the two options for *weights*. 'distance', which gives greater influence to the query point's closer neighbors, will be used.

Figure 8.3 shows a minor difference in performance between the usage of Manhattan distance and euclidean distance, represented by integers 1 and 2 respectively. As

using Manhattan distance marginally improves performance, option 1 was selected over default value 2.

The parameter *leaf_size* is passed to the BallTree and KDTree search algorithms, so has no impact when these are not used. As 'auto' has been selected, these two algorithms may occasionally be used, but as shown in Figure 9, adjusting *leaf_size* has expectedly very little effect on the performance of the model. The *best_params_* attribute of grid search suggested a size of 75, which we have implemented as a relatively large leaf size such as this can also decrease model runtime. [11]

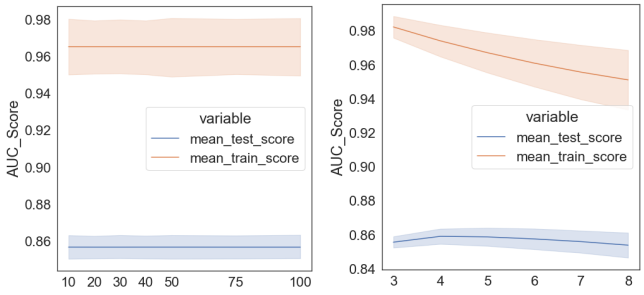


**Fig 9.** Grid search results for *leaf_size* (L) and *n_neighbors* (R)

Finally, with every secondary hyperparameter decided, grid search was run with a range of values for *n_neighbors*. An initial search garnered the data shown in Figure 9. They show a decrease in overfitting as *n_neighbors* increases, as the mean training score decreases while the mean test score stays relatively uniform. As a result of these findings, manual testing was done with larger values of *n_neighbors*, and it was determined that *n_neighbors* ≈ 1000 resulted in the best overall score.

## 4.3 Random Forest (RF)

Random Forest is a machine learning algorithm that can be used for classification tasks. It works by training a large number of decision trees on a labelled dataset, where each tree 'works' on the classification of a given sample. The final classification for the sample is determined by a majority vote among all trees. Decision trees in a random forest are trained using a technique called bagging which involves training each tree in a random subset of data. Training in such a manner helps reduce overfitting as each tree is only seeing a part of the overall data, and so is therefore less likely to learn patterns that are specific to the training set. This helps to not generalise new data.

Figures 10 and 11 show the results of testing *min_samples_split* and *min_samples_leaf* with ranges (0, 20, 4). It was found that an increase in the number of leaves and the number of splits both result in a decrease in the overall AUC score. This shows a risk of underfitting with higher values. The value of *min_samples_leaf* had a much greater effect, with the AUC score dropping dramatically with a higher leaf count. We found that the optimal value was 6 for each.

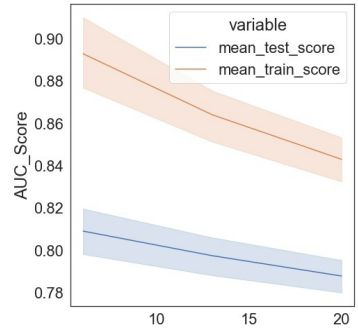


**Fig 10.** Grid search results for *min_samples_leaf*

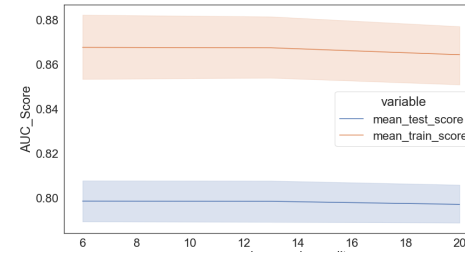


**Fig 11.** Grid search results for *min_samples_split*

We also tested *max_depth*, which is the maximum depth an individual tree can go, shown tested with values 10, 20, and 30 in Figure 12.1. Knowing the risks of overfitting here, the range tested was kept to a minimum, and results came back with 30 being the optimal. Any higher values would have had the potential to severely overfit the data.

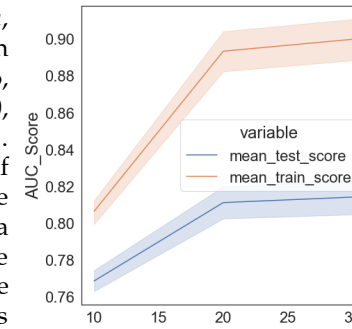


**Fig 12.1.** Values for *max_depth*

A larger value of *n_estimators* reduces overfitting, but drastically increases training times. In the range 1 - 200, Figure 12.2 shows that 200 was the most optimal number of estimators when tuning this hyperparameter, with there being a dramatic increase in performance from 15, converging shortly after.

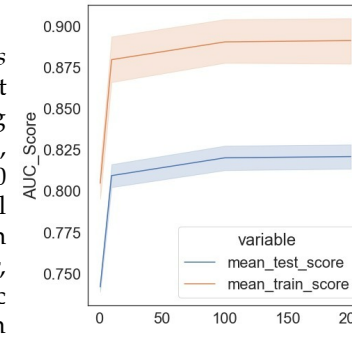


**Fig 12.2.** Values for *n_estimators*

As shown by Figure 13, varying *criterion* resulted in subtle changes to the score. 'entropy' provided a better score at the expense of more computational complexity, therefore we chose to use 'entropy' in the model.

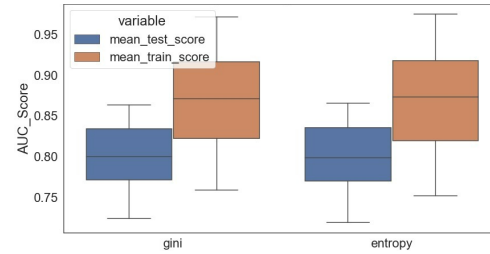


**Fig 13.** Grid search results for *criterion*

Overall, *max_depth* and *n_estimators* had a large increase in the output accuracy, with the positive effect on the score gradually converging at values 20 and 100 respectively. Results were hardly affected by the value of *min_samples_split*. With *min_samples_leaf*, an increase in leaves decreased performance.

## 5 MODEL COMPARISON

Various evaluation metrics were utilised to assess the performance of each model and their optimised parameters on the test set. This comparison analyses the performance of each model to determine their success in correctly identifying customers who default. As shown in Figure 14, all models exhibited similar AUC scores, with values slightly above 0.5. This suggests that the models were able to achieve some level of success in distinguishing between defaults and non-defaults.

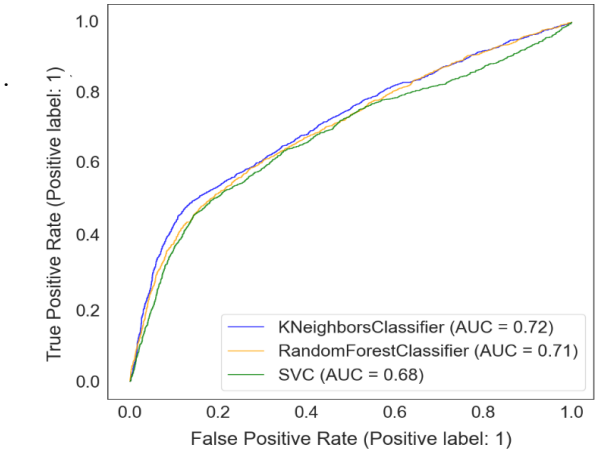


**Fig 14.** AUC-ROC curve for each model

Upon closer examination, it is apparent that the KNN model performed slightly better than the other models, achieving the highest AUC score of 0.72. This indicates that the KNN model is the most successful in separating default and non default classes. This may be because the decision boundary between classes is complex and nonlinear, and a non-parametric model such as KNN (and Random Forest to a lesser extent) is more effective at capturing complex decision boundaries. SVM had the lowest AUC score of 0.68 among the models tested, indicating that it may not be as effective in classifying defaults. This may be due to limitations in investigating a range of values for the parameters *C* and *gamma*, which control the complexity of the decision boundary. Despite using a kernel to capture non-linear decision boundaries, the model performed the worst.

Table 1:
SUMMARY OF PAGE LENGTHS FOR SECTIONS

| Evaluation Metric | Support Vector Machines | K-Nearest Neighbors | Random Forest |
|---|---|---|---|
| Class 0 Recall | 0.781 | 0.835 | 0.786 |
| Class 1 Recall | 0.533 | 0.520 | 0.541 |
| F1 Score (*Macro avg*) | 0.640 | 0.672 | 0.647 |

Results in Table 1 below indicate RF performed the best, achieving the highest Class 1 recall score of 0.541, which is a measure of the proportion of positive cases that were correctly identified. As our focus was on correctly identifying defaults, we prioritise this metric. All models performed poorly in this aspect as our dataset was imbalanced, even with the application of SMOTE. KNN achieved the highest F1 score of 0.672, However, it also had the lowest Class 1 recall score, 0.520. This suggests that while the KNN model was able to correctly classify a majority of the instances in the dataset, it struggled to correctly identify default cases, leading to inadequate generalisation. Despite the small variations in performance, it is clear that all models struggled to identify positives due to the imbalance of data with RF demonstrating the best resilience to this flaw.

## 6 EVALUATION AND CONCLUSION

The aim for this supervised machine learning task was to predict the issuing of defaults to credit users based on financial and demographic data. To achieve this, we trained three different binary classification models on the same dataset and compared their performances. Ultimately, the performance of each model was extremely similar, with each yielding many false negatives and relatively fewer false positives indicating difficulty in correctly identifying defaults (true positives). This was a result of our dataset being imbalanced, and while we attempted to address this issue using SMOTE, we would most likely have seen better results if we had synthesised even more minority class samples or explored other sampling techniques such as cluster centroid. Regardless, the positive impact of our initial SMOTE application demonstrates that a more balanced dataset would result in even more effective machine learning models, proving the utility of such models to those credit users who wish to avoid receiving a default.

Regarding our procedure, results may have been improved by tuning more hyperparameters using a wider range of suitable values. This is especially true for SVM, due to its large number of hyperparameters which greatly increase the flexibility of the model. We limited our use of grid search due to the time cost of hyperparameter tuning, but we have since learned that comparable results can be obtained using a randomised search, which would have enabled us to explore possible combinations in greater depth. Our procedure has also highlighted the importance of feature engineering. The application of PCA to the data had minimal impact on model performance, as PCA is not designed for categorical data. In this case, techniques such as **Multiple Correspondence Analysis (MCA)**, designed for categorical data may have been more appropriate. In light of this, we have learned that careful consideration should be given to the selection of techniques used in the preprocessing stage.

## REFERENCES

[1] Experian, *How Can a Default Impact My Credit Profile*
[2] Analytics Vidhya, *30 Questions to test a data scientist on Linear Regression* (2017)
[3] Fazer Pergunta, *How many rows can Python pandas handle?* (2021)
[4] Abhay Parashar, *Ways to Handle Missing Values in Machine Learning* (2020)
[5] Tani N Prabhu, *Normalization vs Standardization, which one is better* (2020)
[6] Deepchecks, *What is One-hot Encoding*
[7] V. Roshan Joseph, *Optimal ratio for data splitting* (2022)
[8] University of Washington, *Inappropriate use of Chi Square* (1998)
[9] Vishal Mendekar, *Machine Learning – it's all about assumptions*
[10] Alex Ortner, *Top 10 Binary Classification Algorithms* (2020)
[11] Shubh Tripathi, *KNN* (2021)