Deep Learning for Predicting Credit Card Defaults

Alyssa Venter

ABSTRACT

This report describes the training and evaluation of a model that predicts credit card defaults using a dataset. The investigation revealed that preprocessing the data had a significant impact on the model's performance, due to the dataset's imbalance and skew towards people who would default on their credit cards. To address this, various resampling techniques were tested, with SMOTE proving to be the most effective at ensuring a fair representation. Futhermore, extensive research was done into finding the optimal values of different hyperparameters, it was discovered that a more complex model with a *batch size* of 64 and the *RMSprop* optimiser provided the best results. This study's findings demonstrate the importance of preprocessing methods and hyperparameter tuning in credit default prediction, and it builds on the findings of other researchers.

Keywords: Deep Learning, Credit Card Default, Model Evaluation

INTRODUCTION

Objectives

A credit card is an essential part of modern life, allowing people to borrow money when they need or want it. Credit card default prediction is extremely important in the financial sector because it informs an institution whether or not a potential client is a risk to lend to (1). If a client fails to repay a card payment, it could suggest that they are less likely to repay the money in the future (2). By predicting this about a client based on existing data, an institution can reduce financial losses and decide who to approve. A deep learning approach uses multiple layers of processing to identify meaningful relationships in a dataset. The purpose of this report is to assess the effectiveness of a deep learning model in predicting whether or not someone will default on a credit card payment, as well as to describe the process for finding an optimal model.

Related Work

Credit scoring has historically been done statistically using methods such as logistic regression and discriminant analysis (3). In 2002, Lloyds Bowmaker (a subsidiary of Lloyds Bank) reported that using neural networks improved credit default predictions by 10% (4). According to the same book, even minor improvements in credit default accuracy can have a significant impact on a company's ability to save money, demonstrating the importance of model precision, which is the main objective of this research.

More recently, a large number of studies have been conducted to look into the role of deep learning and optimisation techniques in the evolution of these early neural networks. The 2020 study (5) demonstrated the potential benefits of using recurrent neural networks to utilise historical data. Although recurrent neural networks (RNNs) have potential benefits in this area, the significant portion of non-temporal data complicates the task. Instead, a sequential model was used as it performs well for tabular and less temporal data. Alam et al. suggest using oversampling and undersampling techniques, such as SMOTE or random undersampling, to overcome the impact of imbalanced datasets on credit card default prediction (6). This was implemented in this study after it was discovered that the dataset was unbalanced.

Suitability of deep learning approaches

Deep learning is a powerful technique for handling large amounts of data. It learns hierarchical representations and employs multiple layers to extract features that other methods, such as decision trees or support vector machines, may overlook. This is particularly when there are nonlinear relationships, as seen in a lot of financial

data. In their comparative study, Bayraci and Susuz (7) found that deep learning models outperformed their conventional counterparts in predicting credit defaults, and therefore highlights their suitability for this study.

Deep learning is also highly adaptable, recognising new patterns (like complex relationships between seemingly unrelated information) over time. This is a useful feature for a credit card default prediction model because it allows for the inclusion of a diverse set of data types, and non-numerical factors must be considered when predicting whether someone will default on their payments. A deep learning model is also is continuously learning, which is significant as the model that must deal with constantly changing financial data.

Achievements

In the end, the model had an accuracy of around 0.8 and a loss of about 0.5. Although previously achieving an accuracy of around 0.85, this was traded to increase the number of true positives and negatives. This table displays the values of the confusion matrix generated during the model's final run:

Table 1. Final Confusion Matrix

Actual/Predicted	Negative	Positive
Negative	6359 (True Negatives)	671 (False Positives)
Positive	1055 (False Negatives)	915 (True Positives)

This is a good result, as a large proportion of true negatives were found, and true positives were predicted correctly the majority of the time - although there is room for improvement.

Organisation

This report is organised into two sections: the **proposed method** of the report and its **experimental results**.

The proposed method section outlines the way the model has been created, especially different preprocessing steps taken to offset the data bias that was found in the dataset. It also includes the results of tests carried out to test the effectiveness of the SMOTE method, as well as the comparison of different normalisation techniques. The experimental result sections is organised by each hyperparameter that is changed, as well as a subsection that describes the challenges faced throughout the project, and how they were dealt with.

PROPOSED METHOD

A deep learning model has been developed to solve this binary classification problem - sorting people based on whether they would default or not. Firstly, all of the necessary libraries were imported, and the data was loaded into a data frame using Pandas. The data was then split into two separate features - $'X'$ and $'y'$. $'X'$ contained all of the data that the result could be based on, and $'y'$ contained the actual boolean value that tells whether the person did default on their payments.

The datasets were then split into four variables: *X_train, X_test, y_train,* and *y_test*; these are used to contain the features for testing and training the model and test set. The test size was set to a default 0.2, which means that 20% of the dataset will be used to test, and 80% will be used for training purposes, but was then later changed to 0.3. A random seed was also used, which ensures that the data will be split every time the model was run. Data bias was found to be an issue in this dataset, shown in Figure 1.

SMOTE (Synthetic Minority Oversampling Technique) was used in this model in order to reduce the data bias, and works well as it creates manufactured data to reduce this. This was in contrast to under-sampling, which could get rid of data that may be useful to find complex relationships - and was compared later on in the report. A scaler was also used to normalise the features of the dataset. When comparing two common scalers, the *Standard Scaler* was seen to be a better choice over the *MinMax Scaler*. This was seen in Figure 2, where the effects of the *Standard Scaler* are much more consistent for the accuracy, and provides a lower loss value.

The model was defined and had different amounts of layers throughout the efficiency improving process. Different optimisers were also compared in Figure 2, but the loss function used consistently throughout was *Binary Cross-Entropy*, which is the best for a binary classification problem (8). The model is then compiled and trained with various hyperparameters.

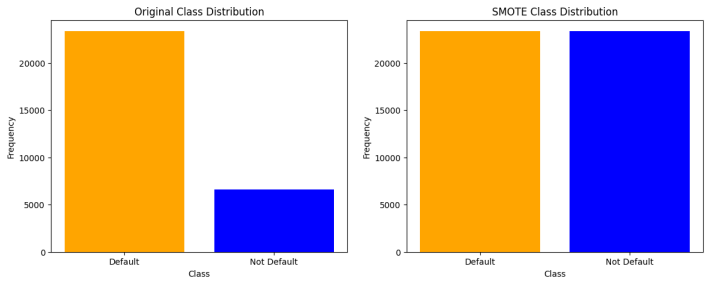


Figure 1. Bar chart showing data bias in the CCD dataset before and after SMOTE is applied

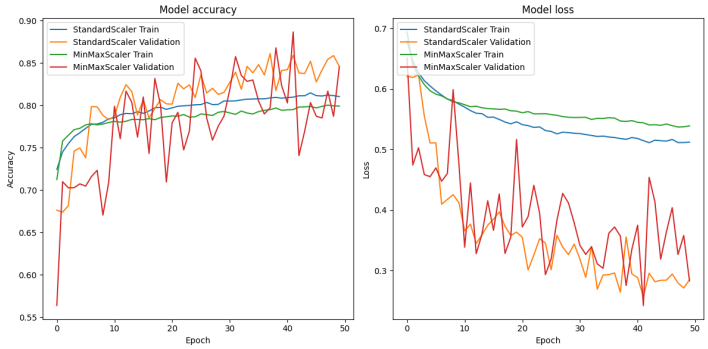


Figure 2. Comparison of MinMaxScaler and StandardScaler

EXPERIMENTAL RESULTS

Throughout this report, many different hyperparameters were tested to determine the best results. When designing the mode, the primary goal was to improve accuracy. However, later on, finding hyperparameters that increased the number of true negatives and positives became a priority because it indicated the true number of people who did and did not default on their payments, as well as whether the model was correct or not. The main evaluation process was completed by testing different intervals of different hyperparameters. As it would be unrealistic to test every single possibility, the hyperparameters were tested with current optimal values for other hyperparameters.

Undersampling and oversampling

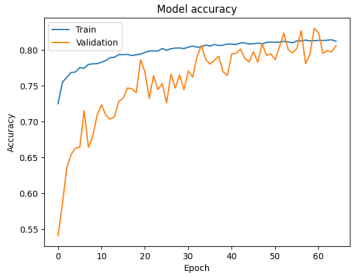
(a) Accuracy of model using undersampling

(b) Loss of model using undersampling

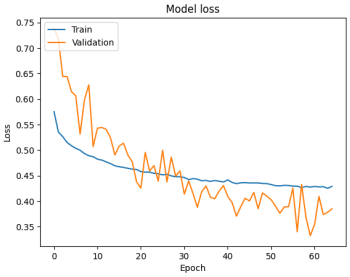
Figure 3. Figures showing accuracy and loss using undersampling

As can be seen from the figures, undersampling techniques had a big effect on the loss and accuracy values

of the model; in these figures, a random undersampler was used. Due to the big difference in the training and validation, it can be determined that the model was underfitting - potentially due to the loss of data. Normally, underfitting suggests that the model needs to be more complex, however this was tested on a relatively complex model with 6 layers and between 32-128 neurons per layer. Therefore, undersampling was not a good choice for this dataset and instead **SMOTE** was used for oversampling. This produced much more stable and fitting accuracy and loss graphs.



(a) Accuracy of model using SMOTE

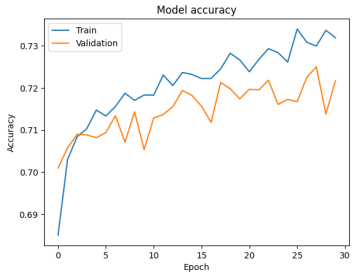


(b) Loss of model using SMOTE

Figure 4. Figures showing accuracy and loss using SMOTE

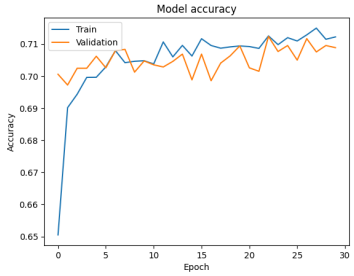
Dropout

To reduce overfitting in the data and ensure that the model was learning, certain nodes were removed using dropout. Different values for adjusting the number of nodes dropped were tested, and it was determined that 0.3 (lower dropout) was the best value. Using a value of 0.5 indicated that the model started slightly underfitting, implying that it was too intense.

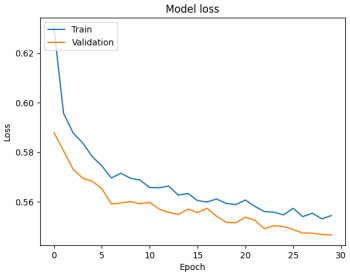


(a) Accuracy of model with 0.3

(b) Loss of model with 0.3



(c) Accuracy of model with 0.5



(d) Loss of model with 0.5

Figure 5. Figures showing accuracy and loss using different dropout values

Optimisers

Two different optimisers were tested in order to see which would produce the most true positives and negatives, which importance was previously discussed. Although there is not a big difference between the values (only around 30 value difference for true negatives), RMSprop outperformed the Adam optimiser in 3 categories, even with two different learning rates.

Table 2. Testing different optimisers

Optimiser	True Negatives	False Positives	False Negatives	True Positives
RMSprop	5373	508	878	761
Adam (lr = 0.001)	5343	528	877	752
Adam (lr = 0.01)	5320	551	871	758

Complexity of the model

Table 3 shows that the majority of models had a similar number of true positives and negatives. However, the complex model had slightly more true positives and enables the discovery of more complex relationships. This was significant because credit default is a complex problem with numerous variables.

Table 3. Model Evaluation Results with Different Architectures

Model	True Negatives	False Positives	False Negatives	True Positives
Simple ($32 \rightarrow 16 \rightarrow 1$)	4218	496	640	646
Complex ($128 \rightarrow 64 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 1$)	4298	426	680	606
Medium ($64 \rightarrow 32 \rightarrow 32 \rightarrow 16 \rightarrow 1$)	4164	550	632	654

Batch size

When testing from this point, it was discovered that the model's validation loss was significantly lower than its training loss. As a result, the model received more data, and the ratio of validation to testing data was increased from 0.2 to 0.3. This enabled the model to collect more data and learn new patterns while validating. The model's batch size was then changed, and it was discovered that, while a batch size of 16 had the highest F1 score, it had a significantly lower number of true negatives than the other, and the accuracy/loss graphs indicated high instability. As a result, the final model used a batch rate of 64, which was much more stable, fast, and efficient with resources.

Table 4. Model Performance with Different Batch Sizes

Batch Size	True Negatives	False Positives	False Negatives	True Positives	F1 Score
16	6197	833	982	988	0.521
32	6453	577	1109	861	0.505
64	6359	671	1055	915	0.515

Class Weights

Another parameter that was adjusted was the class weights, which changed the model's bias towards the minority class. A test was carried out, and from Figure 6 it is clear that 1.35 is the optimal value:

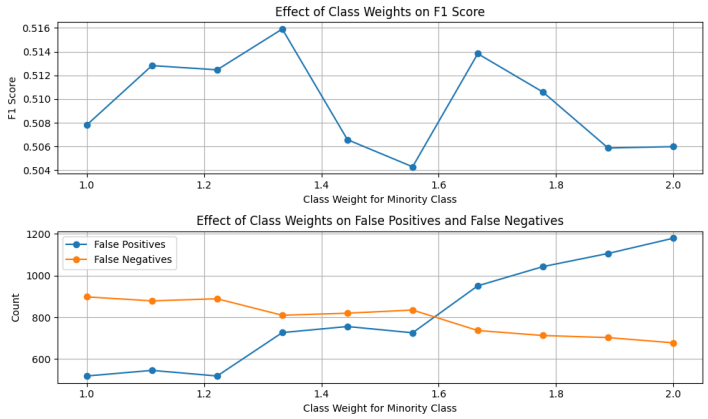


Figure 6. Graphs comparing the effects of class weights on F1 score and True positives/negatives

The main deep learning challenging problem and attempts to cope with the problem

There were numerous challenges encountered during the course of this report, particularly given the number of possible combinations for different hyperparameters. To overcome this, I went through them one by one to determine the most optimal values for the current model. A different strategy would have been to use a grid search method, but I decided against it due to concerns about resource allocation and time constraints. Another big issue that was faced was overfitting and underfitting. One hot encoding was used during the experimentation phase with different approaches that were not included in the final project, which caused the model to overfit excessively. This was also apparent when testing different class weights and oversampling techniques. Underfitting occurred when the data was too regularised; however, techniques such as increasing model complexity were implemented to combat this.

One of the biggest challenges was finding out which metric to use in order to assess the quality of the model. Initially, accuracy seemed to be the best option as it represented as it provides a good general overview of how the model is doing. However, using true positives and negatives was later discovered to be the most accurate method of determining this, particularly when investigating credit card defaults. It has the most real-world impact because it predicts whether a customer will default, which carries a significant financial risk for the company as well as customer trust. If an institution rejects a loan when someone is unlikely to default, it risks losing their trust and, as a result, losing business.

Another metric that was considered was the F1 score. However, it was discovered that the F1 score ranged from 0.5 to 0.52 for each model created, indicating a minor difference. Furthermore, the achieve score in this case is less reliable due to the dataset's imbalance. Even using SMOTE to balance the dataset may not improve the F1 score. Therefore, most results were considered based on the accuracy and true positives and negatives.

SUMMARY

There is definitely room for improvement in the model I developed; while it correctly predicted true positives most of the time, it has the potential to achieve even higher accuracy. Despite continuing attempts to optimise the model, more refinements are required; incorporating non-deep learning techniques into the deep learning model, such as random forest, may increase these values. The proportion of true negatives and the final model accuracy of 0.8 is encouraging, as even though it is not 100% accurate, credit defaulting can be unpredictable; many factors influence it, and even someone with the lowest risk may default for unknown reasons. Therefore, this is promising for future development, and shows the clear effects deep learning can have on the financial industry and predicting credit card defaulting.

REFERENCES

[1] F. Butaru, Q. Chen, B. Clark, S. Das, A. W. Lo, and A. Siddique, "Risk and risk management in the credit card industry," *Journal of Banking and Finance*, vol. 72, pp. 218–239, 2016.

[2] B. K. Wong and Y. Selvi, "Neural network applications in finance: A review and analysis of literature (1990–1996)," *Information & management*, vol. 34, no. 3, pp. 129–139, 1998.

[3] L. C. Thomas, D. B. Edelman, and J. N. Crook, "Credit scoring and its applications," 2002.

[4] S. Goonatilake and P. C. Treleaven, *Intelligent systems for finance and business*. John Wiley & Sons, Inc., 1995.

[5] J. M. Clements, D. Xu, N. Yousefi, and D. Efimov, "Sequential deep learning for credit risk monitoring with tabular financial data," *arXiv preprint arXiv:2012.15330*, 2020.

[6] T. M. Alam, K. Shaukat, I. A. Hameed, S. Luo, M. U. Sarwar, S. Shabbir, J. Li, and M. Khushi, "An investigation of credit card default prediction in the imbalanced datasets," *IEEE Access*, 2020.

[7] S. Bayraci, O. Susuz, *et al.*, "A deep neural network (dnn) based classification model in application to loan default prediction," *Theoretical and Applied Economics*, vol. 4, no. 621, pp. 75–84, 2019.

[8] D. Godoy, "Understanding binary cross-entropy/log loss: a visual explanation," *towards data science*, vol. 21, 2018.