ECM2423 Artificial Intelligence and Applications- Report

Ellis Mackness- 720023157

March 2024

1 Abstract

The findings highlight the potential of deep learning approaches in credit risk assessment and offer insights for practical application in the financial industry. Through extensive preprocessing, including feature scaling and addressing class imbalance, the dataset is prepared for model training. Class imbalance was especially important during the making of this report. By using random undersampling, I was able to mitigate the dataset's bias towards the majority class. Without this, the model will essentially "guess" whether a credit card will default. After addressing this issue, a comprehensive model was built and evaluated successfully.

2 Introduction

The aim of this report is to explore and evaluate deep learning techniques for predicting credit card defaults. I will be using the CCD dataset which has a reasonable amount of data to be tested. This problem holds significant importance in the financial sector, as accurate predictions of credit card defaults can aid in risk assessment for financial institutions.

One common deep learning approach employed in credit risk assessment is the use of *recurrent neural networks*(RNNs). RNNs are well-suited for processing sequential data, making them suitable for tasks involving time-series data such as credit card transaction histories. The idea is to have multiple copies of the same network, each passing a message to a successor. In this way the network can remember information.
By leveraging the temporal dependencies present in sequential data, RNNs can effectively capture patterns indicative of credit card defaults over time. Additionally, variants of RNNs such as Long Short-Term Memory (LSTM) networks have been shown to mitigate the vanishing gradient issue, further enhancing their suitability for credit risk prediction tasks [Say+18].

Through systematic experimentation and analysis, this report hopes to achieve the following objectives:

1. Assess the performance of deep learning architectures for credit card default prediction.
2. Investigate the impact of various pre-processing techniques on model performance.
3. Provide recommendations and insights for building robust and accurate credit risk assessment models using deep learning techniques.

The report follows a structured format comprising several sections. It highlights the report's achievements with a clear evaluation of the results. The Proposed Method section details the methodology, particularly focusing on the pre-processing phase. The Experimental Results section then presents a thorough analysis of the experiments, covering hyperparameter settings, evaluation processes, and the obtained results. Finally, the Summary section summarizes the main findings, discusses their significance, and suggests potential future research directions. This organized structure ensures clarity and coherence throughout the report.

3 Proposed method

My proposed methodology for predicting credit card defaults involves a systematic approach encompassing data pre-processing, model development, training, validation, and evaluation stages. This section provides a detailed description of each phase of the proposed methodology, including the pre-processing phase.

3.1 Pre-processing

Any missing values in the dataset will be addressed through appropriate imputation techniques. In my implementation, missing values will be filled using the forward-fill method. forward fill works by finding absent data values and replacing them with the most recent non-missing value that occurs previously in the dataset. This method is particularly useful for filling missing values in time-series data or datasets where the order of observations is important.

I will then need to implement a feature scaling technique. This is a step used to standardize the range of features in a dataset. Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance).
Feature scaling ensures that all features have a similar scale, preventing data with larger magnitudes from dominating the learning process. Feature scaling is crucial in machine learning algorithms that are sensitive to the scale of features, such as gradient descent-based optimization algorithms, k-nearest neighbors, and support vector machines. Before applying feature scaling, it's essential to split the dataset into training and testing sets to avoid data leakage.

Handling class imbalance is a crucial aspect of building predictive models, especially in binary classification tasks where one class significantly outnumbers the other. In the context of credit card default prediction, class imbalance occurs when the number of instances belonging to one class (e.g., non-default) is much higher than the other class (e.g., default). Failure to address class imbalance can lead to biased models that favor the majority class and perform poorly on the minority class. I

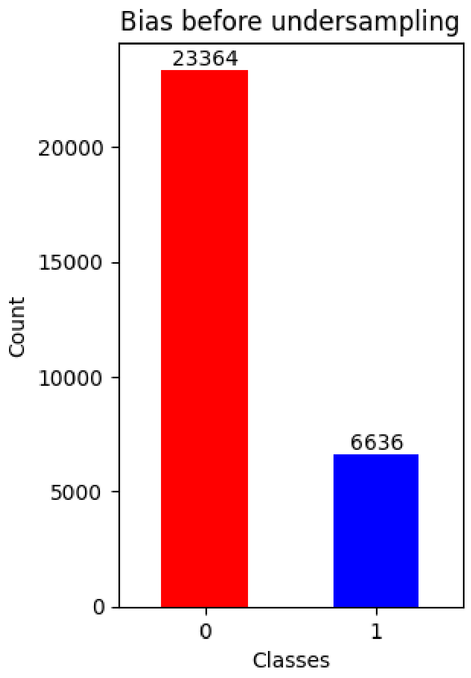

Figure 1: Before under-sampling the biased class

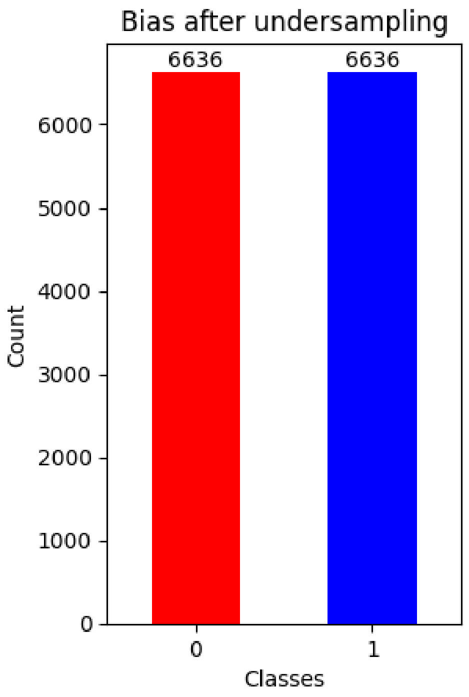

Figure 2: after under-sampling the biased class

have chosen to undersample the majority class from the CCD dataset. Undersampling involves randomly removing instances from the majority class to balance the class distribution. This approach reduces the size of the majority class to match the minority class, thus preventing the model from being biased towards the majority class. However, This can potentially lead to loss of information and may not capture the full complexity of the dataset. The sample size before was *30,000*, and after the sample size is *13,000*

After this, the preprocessed data is split into training and testing sets using a stratified splitting strategy. This strategy selects data samples at random within specific parameters, I will select a test size of 20% to provide enough data for validation while maintaning integrity. This ensures that the distribution of classes is preserved in both the training and testing sets, facilitating unbiased model evaluation.

4 Experimental results

In Section 4, I will present the results of the deep learning model training and evaluation for credit card default prediction. This section offers a concise overview of my results, highlighting key performance metrics and discussing the hyperparameters used.

4.1 Hyperparameters

1. **Optimizer**: The optimizer chosen is Adam, which is an adaptive learning rate optimization algorithm. Adam combines ideas from RMSprop and Momentum optimization and is known for its efficiency and effectiveness in training neural networks.

2. **Loss Function**: Binary crossentropy is selected as the loss function. It is commonly used for binary classification problems, such as credit card default prediction. Binary crossentropy measures the difference between the actual and predicted probability distributions for binary outcomes.

3. **Metrics**: The primary evaluation metric chosen is accuracy. Accuracy measures the proportion of correctly predicted instances among all instances. In the context of credit card default prediction, accuracy indicates the model's ability to correctly classify whether a credit card account will default or not.

4. **Number of Epochs**: The model is trained for 40 epochs. An epoch refers to one complete pass through the entire training dataset. I experimented with different epoch sizes, I found that the accuracy began to plateau around 35 epochs. Therefore 40 epochs is an adequate size.

5. **Batch Size**: The batch size is set to 128. The batch size determines the number of samples processed before the model's parameters are updated. A larger batch size may lead to faster training but will require a lot more memory

6. **Validation Split**: A validation split of 0.2 is used, meaning that 20% of the training data is reserved for validation during training. The validation set is used to monitor the model's performance on unseen data and to detect overfitting.

4.2 Model architecture

4.2.1 Activation function

In designing the architecture for the neural network model, several strategic choices were made to optimize performance and robustness. The inclusion of dense layers with varying numbers of neurons and activation functions, such as ReLU, facilitates the extraction of complex features from the input data, enabling the model to capture relevant patterns indicative of default occurrences.

4.2.2 The vanishing gradient problem

As further layers are added to a neural network, the derivative of the loss function approaches zero. This can result in decreased accuracy of a deep learning model. As the number of `n` layers increase, each `n` derivative is multiplied together. Therefore the gradient will decrease exponentially as the model propagates through the layers [Wan19].

batch normalization layers have been added after the input layer and the first hidden layer. These layers aim to address the vanishing gradient problem by normalizing the input data before it enters the subsequent layers. This normalization helps stabilize the learning process, allowing for more efficient convergence.

4.2.3 Dropout

By randomly dropping a fraction of neurons during training, dropout prevents the model from relying too heavily on specific features, thereby reducing the model's sensitivity to noise. I experimented with different dropout scenarios. one approach I carried out was to start with a relatively low dropout rate, such as 0.2 or 0.3, and gradually increase it if overfitting is observed. Conversely, if underfitting occurs, the dropout rate can be reduced. Larger models with more parameters may require higher dropout rates to prevent overfitting, while smaller models or datasets may benefit from lower dropout rates to maintain model capacity. I undersampled the majority class in the CCD dataset. This means that a smaller dropout rate is preferred to preserve model's integrity

These architectural choices were made with the goal of creating a model that not only achieves high accuracy but also demonstrates resilience against overfitting, thus ensuring its effectiveness in real-world credit risk assessment scenarios.

4.3 Results

I ran my model three times to obtain an average for my evaluation metrics. The metrics I am using are a follows:

1. **Accuracy**: Accuracy measures the overall correctness of the model's predictions across all classes. It's calculated as the ratio of correctly predicted instances to the total number of instances.

2. **Loss Score**: Loss score, often represented by the loss function used during training, quantifies the error between the true labels and the predicted probabilities. The lower the loss score, the better the model's predictions align with the truth.

3. **Precision**: Precision focuses on the quality of positive predictions made by the model. It measures the ratio of true positive predictions to the total number of positive predictions.

4. **Recall**: Recall, also known as sensitivity or true positive rate, assesses the model's ability to capture all positive instances in the dataset. It measures the ratio of true positive predictions to the total number of actual positive instances.

5. **F1 Score**: The F1 score is the harmonic mean of precision and recall. It provides a balanced measure of a model's performance by considering both false positives (precision) and false negatives (recall). The F1 score ranges from 0 to 1, with higher values indicating better model performance. It's particularly useful when there is an imbalance between the classes.

	Test 1	Test 2	Test 3	Average
Accuracy	0.695	0.695	0.700	0.697
Loss Score	0.583	0.581	0.581	0.582
Precision	0.768	0.775	0.763	0.769
Recall	0.560	0.551	0.576	0.562
F1 score	0.648	0.644	0.657	0.650

Table 1: Model findings (3 significant figures)

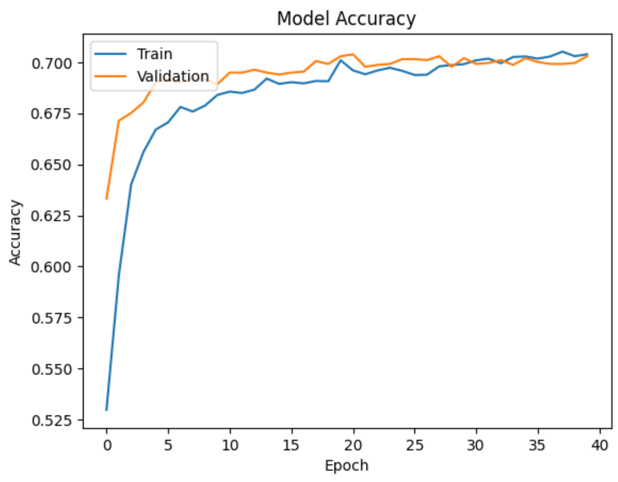

Figure 3: Model accuracy

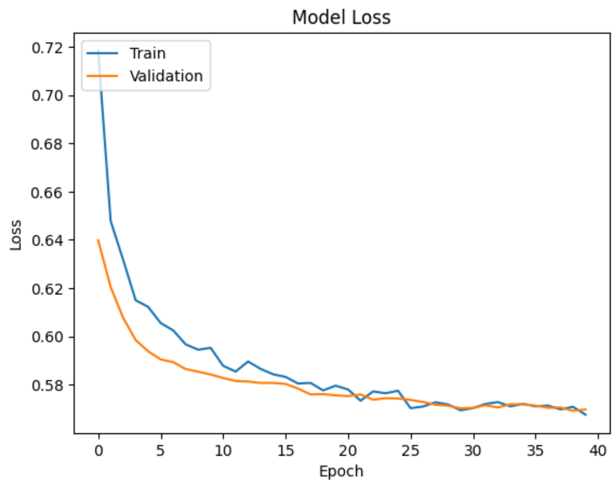

Figure 4: Model loss

Across multiple tests, the model demonstrates consistent accuracy. This demonstrates that the model maintains a stable level of correctness in its predictions across all classes. The calculated loss scores are consistently low. This means that the model's predictions closely align with the ground truth, with minimal error in the estimation process.

Precision and recall metrics exhibit slight variability across different experiments. While precision values range from 0.763 to 0.775, recall values range from 0.551 to 0.576. This suggests that the model's ability to avoid false positives (precision) and capture all positive instances (recall) may exhibit slight variations under different conditions. The F1 scores are relatively balanced across experiments. This indicates that the model achieves a satisfactory balance between precision and recall, effectively considering both false positives and false negatives in its predictions.

5 Summary

In summary, the experimental results indicate that the trained model performs well across various metrics. While there may be slight variations in performance across different experiments, the model demonstrates consistent accuracy, low loss scores, and a balanced trade-off between precision and recall. These findings highlight the model's effectiveness in making predictions on credit card default dataset. Furthermore, it demonstrates the importance of measuring different metrics. This allows a more precise evaluation of the model, whereas simply measuring accuracy on it's own may prove to be misleading.

References

[Say+18] Yashna Sayjadah et al. “Credit Card Default Prediction using Machine Learning Techniques”. In: *2018 Fourth International Conference on Advances in Computing, Communication Automation (ICACCA)* (2018).

[Wan19] Chi-Feng Wang. *The vanishing gradient problem*. Jan. 2019. URL: `https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484`.