ECM2423 Report

Student ID: 720023891

Abstract: This report explores the development of a sequential deep learning model, specifically a multilayer perceptron (MLP), to classify credit card transactions as likely to default or not default on payments. The model addresses the challenge of an 80:20 class imbalance, where non-defaulting transactions outnumber defaulting ones by a factor of 4. By employing undersampling and achieving a test accuracy of 0.78, the model demonstrates promising performance in identifying potential defaults. Classification report metrics provide further insight into the model's effectiveness for both defaulting and non-defaulting transactions. Areas for improvement through hyperparameter tuning are identified.

1. Introduction

1.1 The Aim:

This report details the development and evaluation of a deep learning model for credit card default (CCD) classification. The model aims to predict whether a given transaction is likely to default based on the given dataset.

1.2 The Problem: Credit card defaults pose a significant financial risk for lending institutions. Machine learning models can analyse historical data to identify patterns associated with defaults, enabling proactive measures. Common deep learning approaches for binary classification tasks like CCD include convolutional neural networks (CNNs) for image or sequential data and recurrent neural networks (RNNs) for temporal data. However, for smaller, less complex, volumes of data, a simpler sequential model is often suitable.

1.3 Achievements: This report demonstrates the successful development of a deep learning model for CCD classification. The model addresses class imbalance and achieves a promising test accuracy.

1.4 Report Structure: The following sections detail the dataset analysis (Section 2), methodology (Section 3), experimental results (Section 4), and a summary (Section 5).

2. Dataset Analysis

The provided dataset plays a crucial role in model development and training. Here's a breakdown of the key observations about the data:

- **Class Imbalance:** The report acknowledges an 80:20 bias in the class distribution, where non-defaulting transactions significantly outnumber defaulting ones. This imbalance can hinder the model's ability to learn from the minority class (defaulting transactions). The adopted undersampling technique (Section 3.1) helps mitigate this issue.

- **Feature Structure:** The data consists of 24 columns, categorised into two primary types:

    - **Non-temporal Features (First 5 Columns):** Static features about the customer: balance limit, sex, education, marriage and age.
    - **Temporal Features (Next 18 Columns):** These features capture repayment status, bill amount and amount of previous payment, all over a 6 month period.

2.1 Handling Temporal Features:

Temporal features introduce an additional layer of complexity compared to static features. Here are some approaches to consider when incorporating them into the model:

- **Feature Engineering:** Techniques like creating new features based on rolling window calculations (e.g., average bill amount over 6 months) or differencing past values can extract valuable insights from temporal data.
- **Recurrent Neural Networks (RNNs):** If the temporal sequence is critical for understanding default risk, RNNs like LSTMs or GRUs can be explored. These architectures are adept at learning patterns from sequential data. The choice of handling temporal features depends on the specific characteristics of the data and how each model reacts.

3. Proposed Method

3.1 Preprocessing:

- Data is loaded and features are separated from the target variable (default status).
- Train-test split allocates 80% of data for training and 20% for testing utilising Sklearn.
- StandardScaler function from Sklearn normalises features in the training set for improved model performance.
- Random undersampling (from Imblearn) tackles class imbalance by balancing the number of defaulting and non-defaulting transactions in the training set.

3.2 Model Architecture: Designed utilising Keras from Tensorflow. A sequential model leveraging L2 regularisation and dropout to prevent overfitting. Batch normalisation improves training stability. ReLU activation is used throughout hidden layers.

The final layer with a single neuron and sigmoid activation predicts the probability of a transaction defaulting.

3.3 Training: Early stopping monitors validation loss and halts training if it doesn't improve for 15 epochs, preventing overfitting.

4. Experimental Results

Bolded rows indicate usage for the final set of results.

4.1 Layers

Different types of layers were trialled to understand if some had greater understanding of the dataset. When using LSTM layers, non-temporal data was ignored.

Layer 1	Layer 2	Layer 3	Layer 4	F1-Score (value 0)	F1-Score (value 1)	F1-Score (weighted avg)
Dense (ReLU)	**Dense (ReLU)**	**Dense (ReLU)**	**Dense (sigmoid)**	**0.87**	**0.53**	**0.79**
LSTM	LSTM	Dense (sigmoid)	–	0.87	0.51	0.79

4.2 Hyperparameters

Batch Size:

Batch Size	F1-Score (value 0)	F1-Score (value 1)	F1-Score (weighted avg)
10	0.86	0.54	0.79
20	0.85	0.53	0.78
30	**0.86**	**0.54**	**0.79**
40	0.86	0.53	0.79

Learning Rate:

Learning Rate	F1-Score (value 0)	F1-Score (value 1)	F1-Score (weighted avg)
0.01	0.88	0.52	0.79
0.001	**0.86**	**0.54**	**0.79**

0.0001	0.82	0.22	0.79
0.00001	0.86	0.53	0.79

Neurons:

Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	F1-Score (value 0)	F1-Score (value 1)	F1-Score (weighted avg)
256	128	64	32	16	0.86	0.53	0.79
128	64	32	16	–	0.85	0.53	0.78
64	32	16	–	–	0.86	0.54	0.79
32	**16**	**8**	**–**	**–**	**0.86**	**0.54**	**0.79**
96	48	24	12	–	0.86	0.53	0.78
96	48	24	–	–	0.86	0.53	0.78
23	16	10	–	–	0.86	0.54	0.80

Epochs:

Epochs	F1-Score (value 0)	F1-Score (value 1)	F1-Score (weighted avg)
32	0.86	0.53	0.79
64	**0.86**	**0.53**	**0.79**
96	0.86	0.53	0.79
128	0.86	0.53	0.79

4.3 Evaluation Process:

The model's performance is evaluated using the following metrics:

- **Test Accuracy:** This metric indicates the percentage of correctly classified transactions on unseen data.
- **Classification Report:** This report details precision, recall, F1-score, and support for each class (defaulting and non-defaulting). Precision reflects the proportion of predicted positives that are actually true positives. Recall indicates the proportion of actual

positives that are correctly identified by the model. The F1-score is a harmonic mean of precision and recall, providing a balanced view of model performance for imbalanced classes. Support refers to the total number of transactions in each class.

Two sets of graphs were used to evaluate model accuracy and loss:

1. **Graphs with validation data:** This approach replicates the class imbalance present in the training data. Evaluating similarly imbalanced validation data provides a more realistic assessment of the model's generalizability to unseen data with the same class distribution.
2. **Graphs with undersampled validation data:** Here, the validation data is artificially balanced using techniques like oversampling the minority class (defaulting transactions) or undersampling the majority class (non-defaulting transactions). This allows for evaluation on a balanced dataset and helps identify potential overfitting issues specific to the model architecture and chosen hyperparameters.

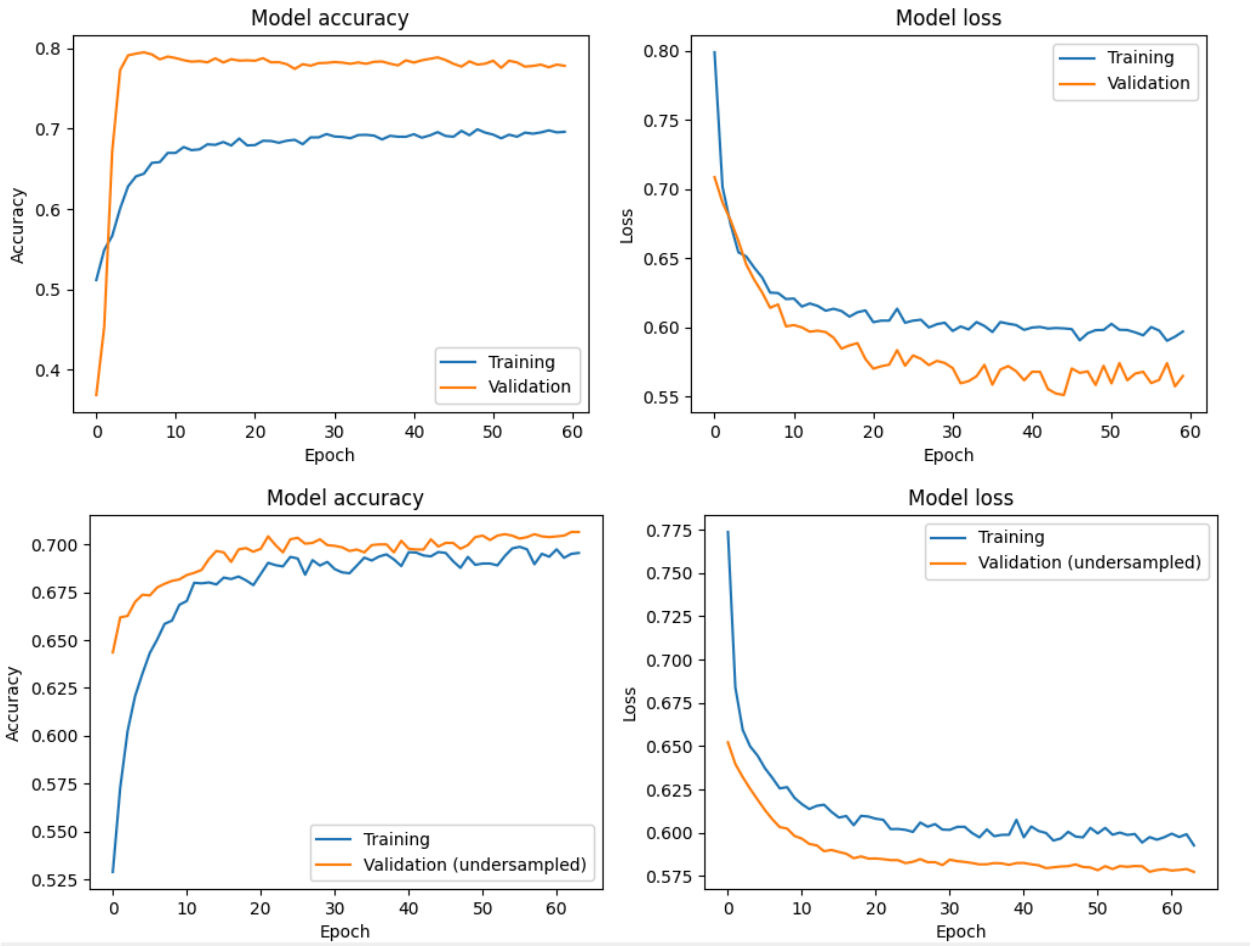
By comparing these two sets of graphs, we can gain insights into the model's ability to handle imbalanced data and potential areas for improvement.

4.4 Obtained Results:

When obtaining these results, it was important to identify what attributes held the most value. For example, the F1-score of the data class, representing a non-defaulting customer, ‘0, could be a lot higher. However, this would come at the cost of the F1-score of the data class ‘1’ (representing a defaulting customer) decreasing. To define the ideal set of results and attributes, more context would need to be given on the contextual use of the software. Below is an example set of values that meet at a good balance for the problem.

Classification report:

Data Class	Precision	Recall	F1-Score	Support
0 (non-default)	0.87	0.85	0.86	4687
1 (defaulting card)	0.51	0.56	0.53	1313
Weighted average	0.79	0.78	0.79	6000



5. Summary

This report demonstrates the development of a deep learning MLP model for CCD classification. The model addresses class imbalance through undersampling and achieves a test accuracy of 0.78. The classification report offers insights into the model's performance for both defaulting and non-defaulting transactions, with a focus on the F1-score which considers both precision and recall.

Challenges in Hyperparameter Tuning: When deciding on hyperparameter values (e.g., number of neurons, learning rate), a trade-off often exists. Improving accuracy for one class (e.g., defaulting transactions) might come at the expense of accuracy for the other class (non-defaulting transactions). Finding the optimal balance requires careful consideration of the specific business goals and potential financial implications of misclassifications.

Future Work: Further hyperparameter tuning of the MLP architecture (number of layers, neurons, learning rate, etc.) can potentially improve performance. Additionally, exploring

different deep learning architectures like CNNs for feature extraction might be beneficial since the features hold sequential significance.