ECM2423 Technical Exercise Report

[153071]
Computer Science, University of Exeter, U.K.

1 Abstract

This report details the attempts to construct a suitable neural network architecture to predict whether a person will default on their credit card at the end of the month. It explains how hyperparamter tuning affects the prediction provided and the attempt to deal with the uneven dataset. The results show that for this particular task it is difficult to get a high accuracy for this problem with a conventional neural network, owing to the imbalance of data and inability to effectively perform feature engineering. The report suggests the use of other algorithms that are more suited to class imbalance classification problems.

2 Introduction

Britain's biggest lenders expect the greatest rise in defaults on unsecured lending since 2009, according to a Bank of England survey[12]. A credit card default(CCD) is the failure of a cardholder to make required payments for their card, leading to penalties such as increased interest rates and potential legal action by the card issuer. This can be hugely damaging for both individuals and banks as people struggle to pay back the loan. Therefore it would be highly beneficial to be able to predict those that would be able to pay back these loans.

It is for this reason that I will attempt to create a machine learning frame work, using the CCD dataset as provided, and attempt to build a robust and accurate machine learning model. This is by no means a novel approach, as attempts have been made by many to tackle this problem [11] using a similar dataset.

I will carry out an analysis of the data looking at such features as sex, marital status and payment histories in order to finally classify the subjects into one of two groups, defaulter or non defaulter. I will test the relevant hyperparameters and document which of these has a significant effect on the accuracy and other metrics of this algorithm.

3 Proposed method

Within my dataset I have 23 features and a binary output 'default payment next month'. My metric of success will be the accuracy with which the testing data can be correctly predicted into one of the two groups. What is presented is a classical supervised learning binary classification problem.

3.1 Data visualisation

I ensured there was no duplicate data points, empty cells or non numerical data. I then used the describe function to look at the typical values and ranges for these columns.

After examining the dataset, I observed the significant class imbalance—with 23,364 out of 30,000 data points indicating no default payment next month, leaving only 6,636 indicating a default, as shown in Fig:1.

I then created a correlation heat map to show how much each of the features correlated with 'default payment next month'. This is shown in 1.

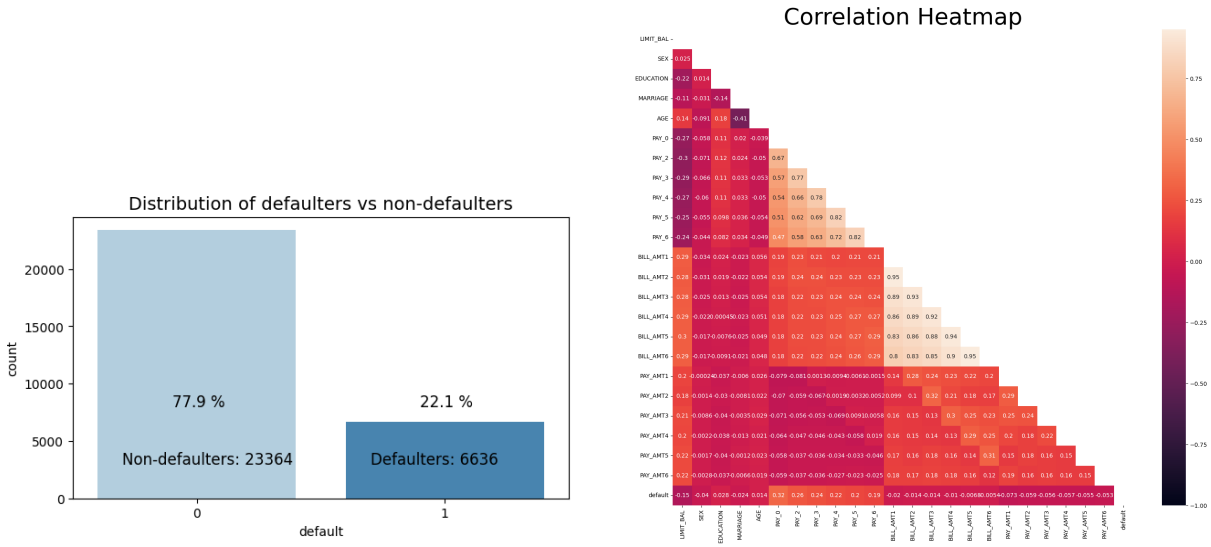

Figure 1: Initial model, showing training and validation accuracy over a number of epochs

The key takeaway is that no great feature extraction can occur as all features are important and play into deciding whether someone will default.

3.2 Data Preprocessing

The preprocessing pipeline consisted of two main steps: scaling numerical data and encoding categorical variables. I used standard scalar giving mean of 0 and standard deviations of 1. This allows for better convergence and model performance during training. I performed this by using the StandardScaler from SciKitLearn.

I also needed to one hot encode my categorical columns: Education and Marriage. This increases the size of my data-frame, creating a new binary column for each unique category in the original categorical variable. A '1' in a column indicates presence of that category, while '0' indicates its absence. I then split all features into 2 dataframes, 32 columns(X) and the target into a single column(y). I split off the SEX column first as this binary column did not need scaling.

Finally I split my data into training(20%), validation(25%) and testing datasets(55%). Then all my data was ready to be passed into a model. Splitting the data ensured the model's performance is accurately assessed and that it generalized well to new, unseen data.

3.3 Model building

My model will take the form of a typical neural network with 32 input neurons in the first layer to match the features. I then added 2 dense hidden layers 64,32 neurons wide, to build connections and terminated with a binary classifier. This first model achieved an accuracy of 0.818 and a loss of 0.433 over 10 epochs. However over 100 epochs there was clear overfitting. I then examined several key hyperparameters that could be tuned in an attempt to determine how to improve my basic model.

3.3.1 Learning rate

Before this I wanted to investigate the learning rate to choose the best optimizer. I utilised a learning rate scheduler within TensorFlow Keras framework, using the callback function LearningRateScheduler to adjust the learning rate dynamically during training based on the epoch. I plotted this result in 3,

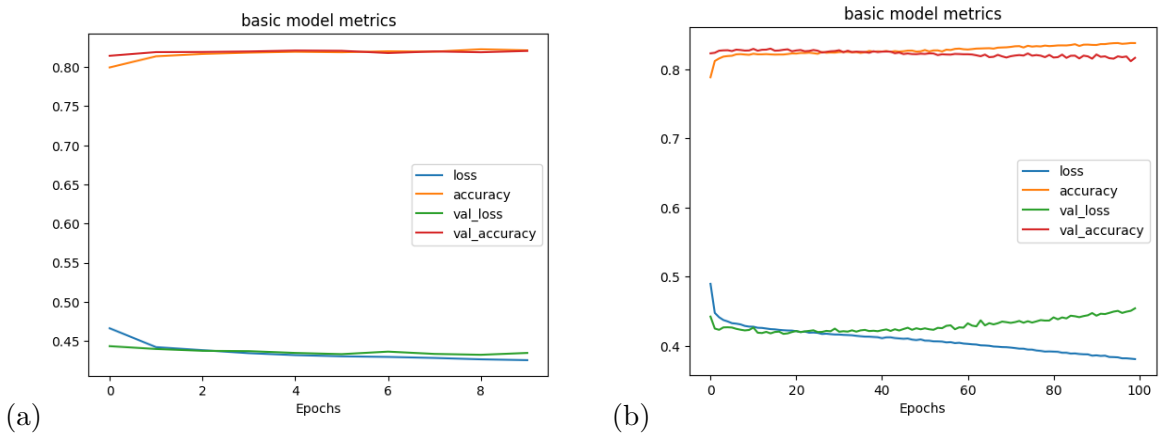

Figure 2: Initial model, showing training and validation accuracy over a number of epochs

which showed how the learning rate varied across 100 epochs.

Here the ideal learning rate was 0.0005. According to the literature [9] Adam excels at these low values.

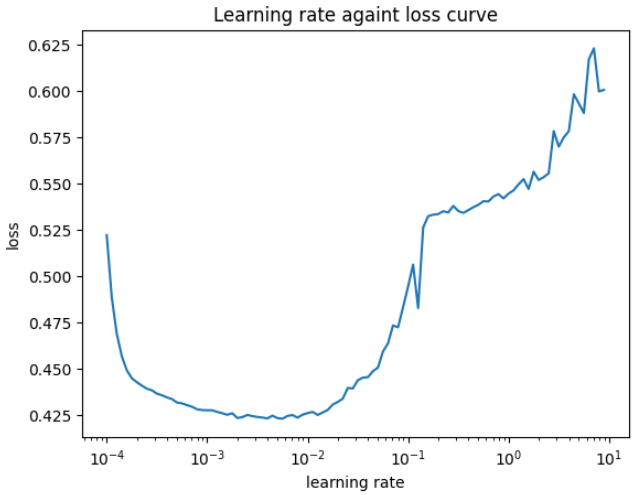

Figure 3: Graph that shows learning rate over 100 epochs on a logarithmic scale. This graph is interpreted to find the min point which is the maximum learning rate

3.3.2 Class Imbalance problem

The dataset is highly imbalanced with nearly 78% of individuals not defaulting on credit cards. To remedy this I set about using random over/undersampling from the imblearn libary, using a ratio to change the quantity between the defaulter and non defaulters. Plotted as 5.

Unfortunately, from these preliminary tests no sampling methods led to increased accuracy. When the minority class represented 50%, the size of the majority class resulted in accuracies of 0.75 for oversampling and 0.77 for undersampling. When the dataset was re-balanced to have the minority class comprise 75% of the majority class, this resulted in scores of 0.73 oversampling and 0.70 undersampling. This excessive oversampling led to significant overfitting.Dropout regularization emerges as a crucial solution to mitigate overfitting in such cases. I also applied SMOTE techniques, which yielded in similar results as oversampling.

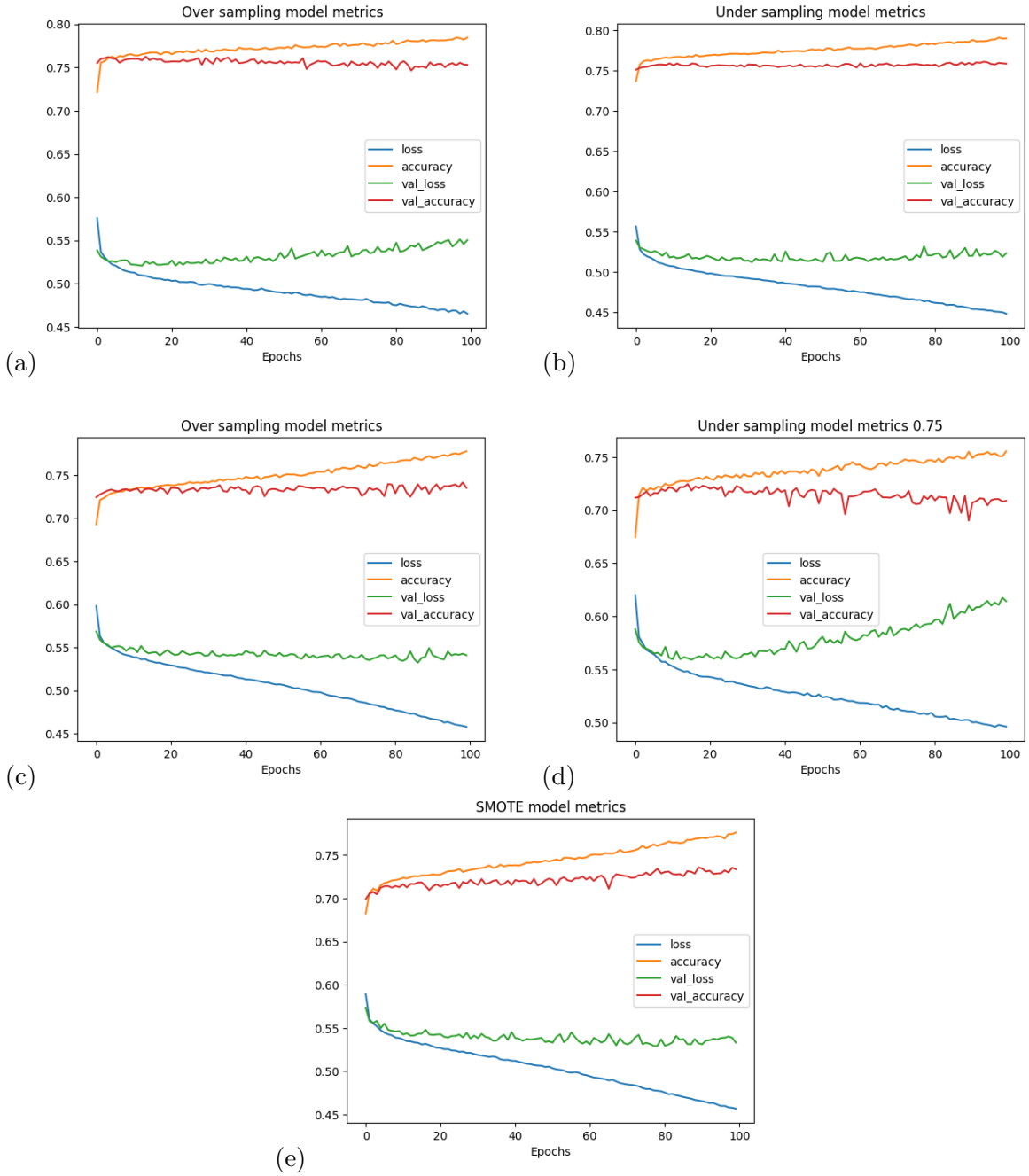

Figure 4: graphs showing how undersampling, oversampling and smote affect several key performance metrics.

3.3.3 Dropout

Dropout is a regularization technique used in neural networks to prevent over fitting[4] by randomly dropping a proportion of neurons during training. A model learns to memorize training data instead of generalizing to unseen data.

I implemented this using the tensor flow library where I imported dropout and added it after dense layers. I will investigated the optimal value of dropout.

3.3.4 Batch Size

Batch size is a key hyperparameter. It determines the number of data points processed by the model before updating its parameters through backpropagation, and has been shown to lead to changes in

performance [7]. Therefore I will be investigating how varying batch size affects the models' performance.

3.3.5 Number of epochs

Training for multiple epochs allows the model to gradually learn complex patterns in the data by adjusting its parameters to minimize the loss function. Higher epochs give the model more opportunities to learn from the data and improve its performance.

However, training for too many epochs can lead again to overfitting[5], where the model learns to memorize the training data instead of generalizing well to unseen data. Therefore, the number of epochs needs to be tuned.

3.3.6 Network Depth & breadth

Depth and breadth significantly impact a network's performance and generalization capacity[8]. While deeper networks can potentially enhance performance, they may suffer from vanishing gradients[1], leading to decreased training efficiency. Moreover, deeper networks are computationally intensive and demand larger datasets for effective training.

3.3.7 Class weighting

Class weighting is crucial in neural networks for handling imbalanced datasets. It modifies the current training algorithm to correct for skewed distributions. The different weights attached to groups penalizes miss-classification. This helps mitigate bias towards majority classes, leading to better overall performance and generalization.

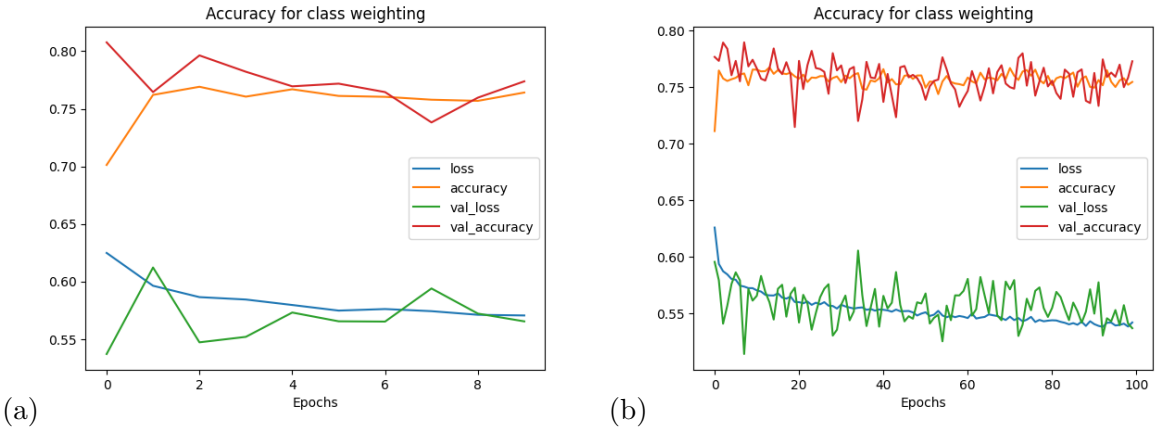

Figure 5: graphs showing how undersampling, oversampling and smote affect several key performance metrics.

3.3.8 Final plan

Once I had established an initial model, my plan is to investigate the effect of varying hyperparameters. I did this through the use of the function systematic_tuning which will iterate through a number of predefined hyperparameters: epochs, dropout, batch size and depth. I picked these as as I had already outlined that they needed to be fine-tuned, and are easy to vary. After running the presets through this function I returned a number of metrics to accurately represent how these parameters changed the performance. I exported this data to a CSV, then I plotted and analysed the trends.

4 Experimental results

Results from systematic_tuning runs varying epochs, dropout rate, batch size, and depth systematically, were analyzed. Multiple loops iterated over arrays of these values applied to the model, with results appended to an array. This process was repeated for different data types (unedited, under/over sampled, and SMOTE) and various depths of architecture. The resulting data was collected in a CSV and plotted (see Figure 6). Mean values for each group of independent variables were calculated, and lines of best fit were plotted for each data type.

4.1 Metrics

Metrics crucial for understanding model effectiveness and the impact of hyperparameters were examined. Test accuracy, indicating the proportion of correctly classified instances in the test set, is a primary metric but can be misleading in imbalanced datasets due to false positives and negatives [3]. Precision, measuring true positives among all positive predictions, is essential for minimizing false positives [6]. Recall, measuring true positives among all positive occurrences and contributing to the F1-score, is crucial for balancing false positives and negatives. ROC-AUC evaluated the sensitivity-specificity trade-off, while PR-AUC focuses on retrieving positive instances, especially in imbalanced datasets, by considering precision and recall simultaneously.

All of these metrics were plotted for each independent variable in 6.

4.2 Analysis

As can be seen immediately in a vast majority of these plots, the unedited data performs astonishingly badly compared to the augmented data in all categories bar accuracy. The model guessed far too many positives, shown by the precision being far lower than all others as well as a very low value for recall. The unedited dataset model effectively ignored the possibility that a person could default on their credit card.

The other resampled data was much closer in performance, so it was worth examining the individual parameter tuning that show the differences in result.

The number of epoch had no huge discernible effect on many metrics. In many cases, it led to simultaneous increases and decreases in performance between datasets. However in general it seemed like increasing the number of epochs did not increase the metric. But these increases were typically not large enough to push the models into good performance.

Next, dropout rate was varied between 0.1 and 0.25. It had no great effect on accuracy or PR-AUC, but it led to an increase in precision. This is good as it shows that the dropout is preventing the issue of overfitting to the training data. Unfortunately the increasing dropout led to decreased values of Recall, F1 and ROC-AUC across the board. This could be due to the fact that it exceeded the suggested amount and too many connections were being dropped, so the optimum value likely sits around 0.125.

Batch size was unusual amongst these as lower values typically garnered better results. This was the case for all but precision. The unedited data showed the largest change in precision at the lower batch sizes. Finally, increasing the number of layers showed surprisingly little increase in the model's performance. This was surprising as it was well documented that the depth of a network had a large impact on performance- to a greater extent than width [10].

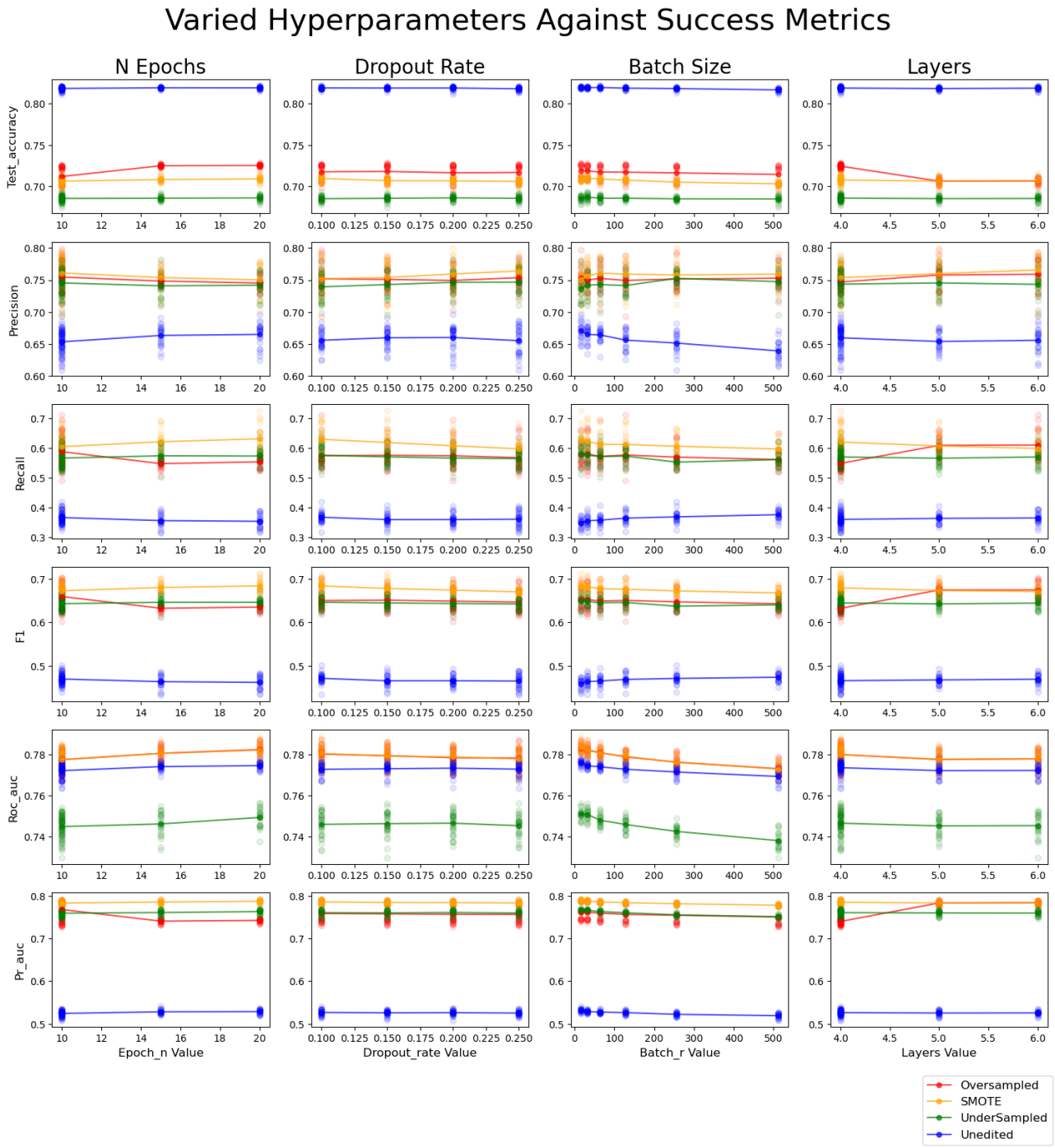

Figure 6: Details plots of how metrics: Test Accuracy, Precision, Recall, F1-score, ROC-AUC and PR-AUC vary as the hyper-parameters: number of epochs, dropout rate, batch size and layers are systematically altered, This is done for all datatypes: unedited, oversampled, undersampled and SMOTE

5 Conclusion

In conclusion, this report highlights how test accuracy alone is not a sufficient metric for analysing the effectiveness of a neural network. By systematically changing the parameters, I have shown that decreasing batch size and increasing the number of epochs have across the board benefits for these algorithms. Other factors such as dropout have clear optimal values which could be discovered with further research. The depth and width of the network also need to be further examined.

However, the flaws of this work highlight that implementing this typical neural network structure has

its limitations as the test accuracy is still relatively low despite the size of the dataset. I therefore believe that it would be worthwhile to examine other algorithms aside from neural network, such as random forest or logistic regression [2] which have been shown to work despite large class imbalance.

References

[1] Sepp Hochreiter. "The vanishing gradient problem during learning recurrent neural nets and problem solutions". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02 (1998), pp. 107–116.

[2] P. Jeatrakul and K.W. Wong. "Comparing the performance of different neural networks for binary classification problems". In: *2009 Eighth International Symposium on Natural Language Processing*. 2009, pp. 111–115. DOI: 10.1109/SNLP.2009.5340935.

[3] T Ryan Hoens and Nitesh V Chawla. "Imbalanced datasets: from sampling to classifiers". In: *Imbalanced learning: Foundations, algorithms, and applications* (2013), pp. 43–59.

[4] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[5] Andreas S Weigend. "On overfitting and the effective number of hidden units". In: *Proceedings of the 1993 connectionist models summer school*. Psychology Press. 2014, pp. 335–342.

[6] Takaya Saito and Marc Rehmsmeier. "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets". In: *PloS one* 10.3 (2015), e0118432.

[7] Pavlo M Radiuk. "Impact of training set batch size on the performance of convolutional neural networks for diverse datasets". In: *Information Technology and Management Science* 20.1 (2017), pp. 20–24.

[8] Itay Safran and Ohad Shamir. "Depth-Width Tradeoffs in Approximating Natural Functions with Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 2979–2987. URL: https://proceedings.mlr.press/v70/safran17a.html.

[9] Liyuan Liu et al. "On the variance of the adaptive learning rate and beyond". In: *arXiv preprint arXiv:1908.03265* (2019).

[10] Gal Vardi, Gilad Yehudai, and Ohad Shamir. "Width is Less Important than Depth in ReLU Neural Networks". In: *Proceedings of Thirty Fifth Conference on Learning Theory*. Ed. by Po-Ling Loh and Maxim Raginsky. Vol. 178. Proceedings of Machine Learning Research. PMLR, Feb. 2022, pp. 1249–1281. URL: https://proceedings.mlr.press/v178/vardi22a.html.

[11] Kanishk Bakshi and S Vinila Jinny. "Predicting Credit Card Defaults with Machine Learning". In: *2023 Annual International Conference on Emerging Research Areas: International Conference on Intelligent Systems (AICERA/ICIS)*. IEEE. 2023, pp. 1–6.

[12] Bank Of England. *Credit Conditions Survey - 2023 Q4*. Accessed: 09/03/2024. 2023.