**INTRODUCTION**

The aim of this report is to investigate the development and performance of a deep learning model for predicting credit card defaults from a credit card default dataset.

The dataset contains 23 variables such as education, age, and bill amounts. The target variable is whether someone defaults on their credit card, defaulting being when someone does not make the minimum payment required for one hundred and eighty days.

When it comes to deep learning in the finance and banking industry, this can be split into two areas of application: banking and credit risk and financial market investment (Huang, 2020).

For financial market investment you can have AI traders which use algorithms and machine learning techniques to identify patterns and trends in the market, reducing the risk of human error and increasing the accuracy of trades. (Prosper Trading Academy, 2023)

For banking and credit risk you can check to see if people are going to default. This is useful to credit card companies and financial institutions as they can risk assess people before they get a credit card to be able to tell if they are likely to default or not, which helps them mitigate the risk of financial losses due to unpaid debts.

However, there are challenges of using deep learning models in the finance industry as these models learn through observations, which means these systems learn from what they see in training data. The extent of their knowledge hinges on the volume and diversity of data they encounter. If a user’s dataset is small or derived from a single source, the resulting models may lack the generalizability needed for broader applications. Another significant concern that arises is data bias. This raises crucial questions about the ethical implications of their predictions. If the data used for training contains biases, the models will repeat those biases in their predictions. (Nguyen, 2023)

I am going to look at different methods to creating this deep learning model to see if someone is likely to default and look at what variables can be changed to improve performance, such as sampling types and different hyperparameters.

**PROPOSED METHOD**

The key steps to creating this model are going to be data processing, model architecture design and training.

The data processing stage will include opening the initial csv and converting it to a 2-dimensional array. I will then need to remove any useless headers and split it into the input data and the target data. In this case the last column, which is default payment next month, is our target class, so we need to split it into that and the rest of the data which is the input data.

Next we will try and find the categorical variables in the input data which can be one hot encoded to make it easier to understand and process. These variables are sex, education, marriage status, and all six of the pay categories.

Before splitting into training and testing I need to scale all the other categories down to values between one and zero to improve performance using a scaler. Finally, I will then split into training and testing data before using a method of sampling to counter bias in the data.

For the model I want to use a deep neural network with multiple hidden ReLU layers which is suited the task due to DNN’s ability to automatically learn hierarchical representations of data, and non-linearity, making it good for binary classification. It will also need dropout layers to help prevent overfitting, and the final layer will need to be an output layer consisting of a single neuron with a sigmoid activation function, as sigmoid is good for binary classification.

I will use binary cross entropy as my loss model as it is good for binary classification, and for my optimizer I will use Adam due to its adaptive learning rate, momentum, and bias correction mechanisms.

I will test my model with multiple things such as test accuracy, test loss, graphs tracking the training loss and accuracy, and the validation loss and accuracy. I will also use an F1 score to analyse precision and recall and get a more detailed performance analysis and get insight into the model’s weaknesses.

**EXPERIMENTAL RESULTS**

Countering Bias with Sampling

Looking at the data, the target class (default payment next month), the class 0 was significantly more prevalent than the class 1, with approximately 22.12% of results being class 1, and the other 77.88% of results being of class 0. This could lead to a poor model performance due to the model struggling to learn meaningful patterns from class 1 due to the lack of samples compared to class 0. For example, there would be a likelihood of bias in the model with the model predicting class 0 more frequently resulting in a lower true positive rate for class 1.

To counter this I looked at using both undersampling and oversampling to negate the bias and recorded the information. I was using the same scaled data for each of them. I also did it with no sampling to try and prove that there would be bias.

Undersampling:

A graph of a train and validation

Description automatically generated with medium confidence

Test Loss: 0.6930719614028931

Test Accuracy: 0.7663333415985107

Test F1 Score: 0.497250776954339

For the undersampling there seemed to be very clear signs of overfitting happening as there is a massive difference between the training data and the validation data. As the model continues to train, it becomes specialized in predicting the training data, which leads to a large gap between the training and validation performance. Therefore, this model will not perform well on unseen data. This could be caused by the lack of data as the underfitting will remove data, leaving there to be less for it to use to train.

Oversampling:

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

Test Loss: 0.5952760577201843

Test Accuracy: 0.7526666522026062

Test F1 Score: 0.4811188811188811

For the oversampling there seems to no signs of overfitting, which is shown by the close alignment between the training and validation performance. This suggests that the model is learning to generalize well to new samples. There are also no signs of bias due to the balanced representation of classes caused by oversampling. However, it has a similar F1 score to the undersampling which shows that it may not completely address the challenges of class imbalance.

No Sampling:

A graph of a train and validation

Description automatically generated with medium confidence

Test Loss: 0.5672410130500793

Test Accuracy: 0.804111123085022

Test F1 Score: 0.4519738887161952

For no sampling there is a clear divergence between the training and validation performance which indicated there may be signs of overfitting to the training data, and there is a probability of bias due to the class imbalance. Despite it have a higher test accuracy than sampling, it has a lower F1 score, which indicates a poorer performance identifying positive and negative instances. The reason the accuracy is so high is probably due to the bias towards the majority class, as the percentage of class 0 is very close to the accuracy.

In conclusion, I decided to go with oversampling because it counters the bias to an acceptable level, however, does not struggle with overfitting like undersampling does.

Using early stopping to optimise epochs.

Initially I used a level of 252 Epochs which seemed to have a good level of learning without becoming specialized for the training data (overfitting). However, I wanted to see if implementing a system such as early stopping would improve some of the metrics like F1 score and test accuracy, and I will also look at a confusion matrix to see what the model is getting right and wrong. To check this, I ran the model twice, once with early stopping and once without and compared the results.

With early stopping:

The early stopping model I used monitored the val\_loss variable with a patience of ten. When I ran the model it stopped after 48 epochs.

Test Loss: 0.54237961769104

Test Accuracy: 0.7535555362701416

Test F1 Score: 0.5064530485091233

A blue and white diagram

Description automatically generated

Without early stopping:

Test Loss: 0.5852118134498596

Test Accuracy: 0.7617777585983276

Test F1 Score: 0.47425208435507604

**A blue and white graph

Description automatically generated**

What we can see from this is that the early stopping model drastically decreased the number of epochs down from 252 to 48, and this did impact the results. The test loss was significantly lower, and the f1 has an increase of around 0.03. The test accuracy slightly decreased however this was minimal.

The confusion matrix also shows us that without early stopping the model’s accuracy when true label is class 1 is poor at under 50% which may show signs of the model just guessing 0 because most of the data is 1, while with early stopping it got 57% of the 1’s right, which is a nice increase. This may mean that the model is starting to overfit going into later epochs. The model without early stopping did however get more 0’s correctly guessed, but I am putting this down to the majority class prediction that that model seems to suffer from.

To conclude early stopping seems to stop majority class prediction more than without early stopping, leading to a higher F1 score.

Hyperparameters

The hyperparameters I used and changed around were the following: learning rate, Epochs (already covered with early stopping), batch size and validation split.

After playing around with the learning rate the two best learning rates I found were 0.001 and 0.01, which gave the following results:

0.001: Test Loss: 0.5374593734741211, Test Accuracy: 0.75855553150177, Test F1 Score: 0.5268887437404747

0.01: Test Loss: 0.5483982563018799, Test Accuracy: 0.7762222290039062, Test F1 Score: 0.5149325626204239

Considering these were very similar results I went for the learning rate with the higher F1 score, as that is the best variable for how good its precision and recall is.

For batch size I ended up going for 32, as compared to all other ones I tried, it was around that number that gave me the best results.

Initially I used a high validation split of around 0.5 due to it giving me a really high test accuracy of around 0.82, however after plotting graphs and getting an F1 score implemented I realised that it was only doing this due to majority class prediction, as it was predicting twice as many incorrect as correct when the true label was 1. Because of this I lowered it to 0.1 which while lowering the test accuracy, gave me a much higher F1 score and graphs that showed it was learning.

SUMMARY

To summarise, the main findings in the report are about the development of a deep learning model to predict credit card defaults. The report discusses the challenges of using deep learning models in the finance and banking industry due to data bias, and the importance of countering this.

The report also details the making of the model, and the encountering of class bias in the data set used, with there being over three times the amount of class 0 as there is class 1, leading to a model which could be biased towards class 0. The report discusses how the bias was countered with oversampling to replicate samples from the minority class. The report also discusses how early stopping can be used to prevent overfitting, leading to an improved F1 score. It also discusses the best hyperparameters for the deep learning model created, being a learning rate of 0.0001, a batch size of 32, and a validation split of 0.1.

Overall, the report found that you can use a deep neural network to predict credit card defaults, however there are lots of hurdles, such as data bias, which must be overcome.

# Bibliography

Huang, J. C. (2020). *Deep learning in finance and banking: A literature review and classification*. Retrieved from https://doi.org/10.1186/s11782-020-00082-6

Nguyen, T. (2023, August). Retrieved from neurond: https://www.neurond.com/blog/deep-learning-in-finance#:~:text=Uses%20of%20Deep%20Learning%20in%20Finance%201%20Algorithmic,Risk%20Assessment%20...%205%20Security%20Breach%20Detection%20

Prosper Trading Academy. (2023, December). Retrieved from Nasdaq: https://www.nasdaq.com/articles/ai-trading-what-is-ai-trading-how-its-used-in-stock-trading