Michael Luan (z5160351)

MATH5836 Assignment 2: Option 1 – Report

Introduction

This report details a neural network modelling approach in modelling the instances of people who were and were not granted credit. The model involves one target/output variable (named A16), which is binary, and 15 input variables (named A1 – A15). The first section will delve into the introductory insights from processing the data itself, and the second section will summarise the findings in our neural network model.

Data Processing

Procedure

The dataset itself was first cleaned, and every row with at least one missing value was discarded, as modelling in the presence of missing data can cause misleading results. After this procedure, there remained 665 observations/rows. Next, the non-numerical data was either binary or integer encoded, dependent on the number of unique values, and the target class (A16) was binary encoded. Further, the input features were normalised, and the associated data frame was saved into a file named 'data.csv'. From now on, all explorations and analysis will be using the normalised features.

Distributions of Features

As an initial exploration, histograms and boxplots were created for all 16 features. Figure 1 and figure 2 below show the plotted graphs, respectively.

First, it is important to note that A2, A3, A8, A11, A14 and A15 are the only continuous features, and all other features are categorical. As such, we will refrain from exploring the histograms associated with the categorical features, as they may result in misleading conclusions, and only focus on the numerical variables. To retain as much information as possible, normalisation was not used to create the histogram plots. In figure 1, we can see that A2, A3, A8, A11 and A15 are all right-skewed, while A6 appears to be somewhat right-skewed and bimodal. Feature A14 also is extremely centred on 0.

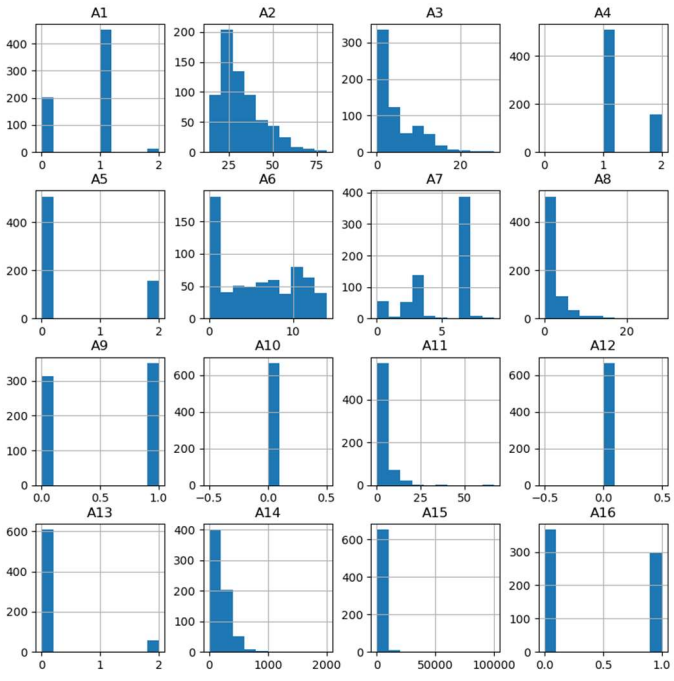

Figure 1: Histogram of all 16 features.

As for the boxplots, since we are interested more in the distribution, normalisation was applied to restrict every feature within [0, 1]. Figure 2 depicts the boxplots for all the features.

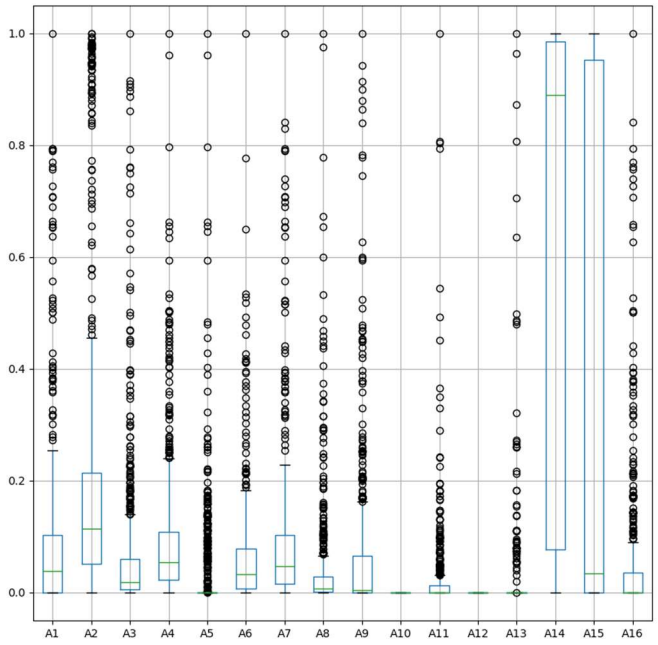

Figure 2: Boxplot of all 16 features.

Once again for the same reasons, we are only concerned with the numerical variables A2, A3, A8, A11, A14 and A15. As seen in figure 2, similar trends can be seen across all features besides A14 and A15, where we see that a lot of observations lie beyond the 3rd quartile, indicating its skew. A14 and A15 showed little variation around 0 in the histograms, and their boxplots also reflected this fact.

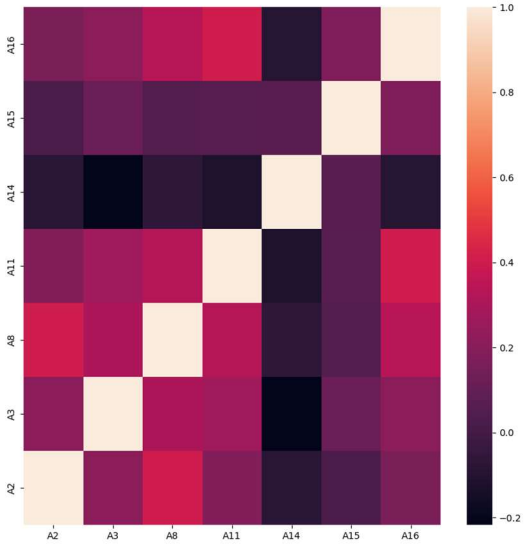

Figure 3: Correlation Heatmap.

To gain more of an understanding into the relationship between these numerical variables, alongside the target variable, a correlation heatmap is constructed, as shown in figure 3. Here, we see that the target feature is not strongly correlated to any of the other numerical variables, with the maximum correlation observed to be approximately 0.5, with variable A11. There appear to be no other major relationships and variable A14 seems to be rather uncorrelated with all the other variables. In the next section, results and findings from the neural network model is detailed.

Neural Network Model

The first step is to select the number of hidden neurons in the hidden layer, which is done by using the equation $N_h = \frac{N_s}{\alpha(N_i+N_o)}$. Here, $N_i = 15$ is the number of input neurons, $N_o = 1$ is the number of output neurons, $N_s = 665 \times 50\% \approx 333$ (take ceiling for training set) is the number of samples in the training dataset, and α is a scaling factor ranging from 2-10. For our analysis, we will take $\alpha = 3$ arbitrarily, resulting in 7 hidden neurons after calculating with the given formula. In moving forward, 7 hidden neurons will be used in all models. Further, a learning rate of 0.01 is applied to all models used, and each model is ran

experimentally a total of 10 times, with a 50%/50% in train/test data split, with 500 epochs based on trial and error. Next, we perform several comparisons with these models, which the following sections will detail.

Adam vs. SGD (no regularisation)

Firstly, we use a single hidden layer with 7 hidden neurons, without any regularisation. The table below compares the performance of the training and test dataset in terms of accuracy between using the Adam and SGD method.

	Train Dataset		**Test Dataset**		
	Mean	SD	Mean	SD	Average AUC
Adam	0.97	0.0038	0.97	0.0020	0.96
SGD	0.93	0.0041	0.92	0.0053	0.92

Table 1: Accuracy of Adam and SGD, for the train and test datasets. All values are correct to 2 significant figures.

As seen in table 1, the Adam learning algorithm outperforms stochastic gradient descent for both the training and testing dataset. It has both a higher mean accuracy of prediction, while achieving lower standard deviations. The average area under curve displays similar results, with Adam outperforming SGD. Further, we can look at the model performance graphs, given in the figures below.

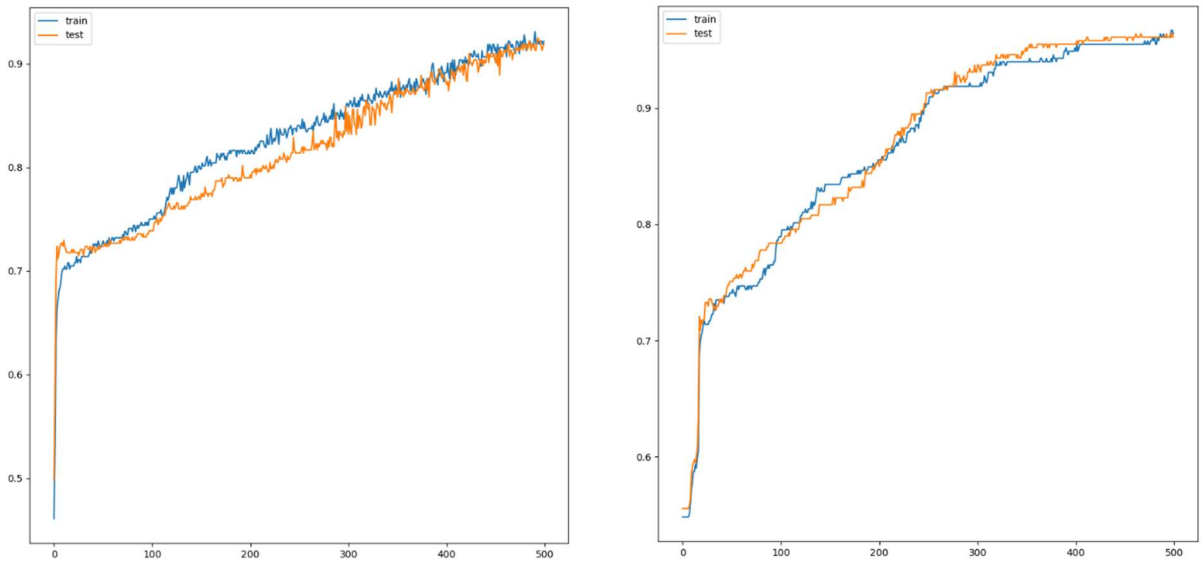

Figure 4: Performance curve of SGD (Left) and Adam (Right) over 500 epochs for one run

In the performance, we can see that Adam is able to achieve a greater accuracy, but the increase in accuracy is more inconsistent compared to SGD. It is also apparent that SGD is likely tend to work better for more generalised situations, given the stochastic nature of its curve compared to Adam. For this particular dataset, Adam has achieved better accuracy

faster. Lastly, looking at the ROC curves in figure 5, they reflect similar outcomes to our previous findings based on accuracy.

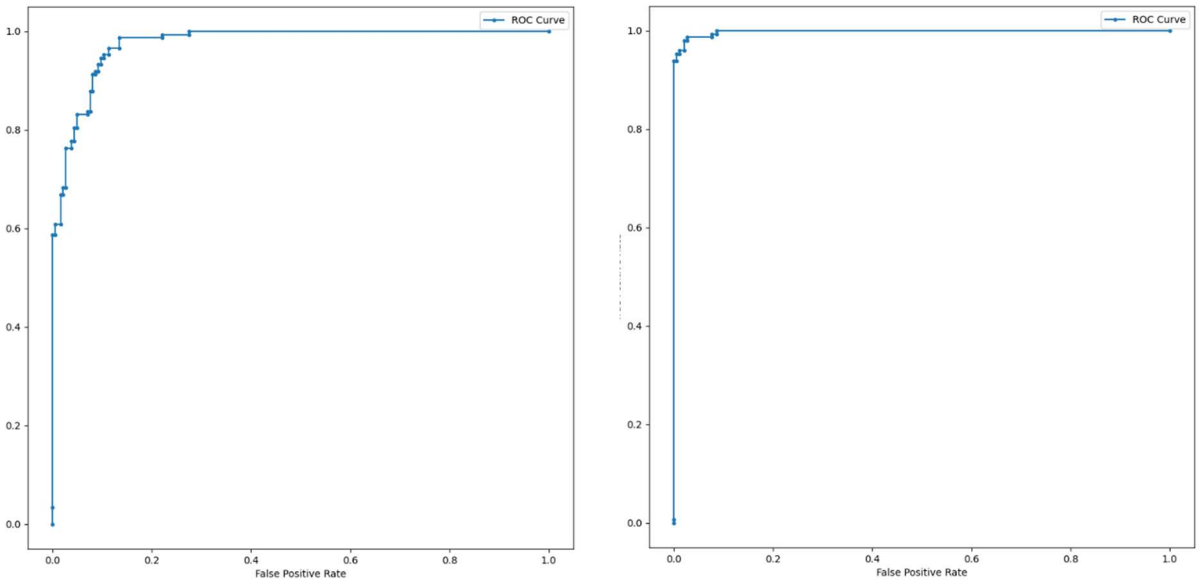

Figure 5: ROC curve of SGD (left) and Adam (right) over 500 epochs for one run.

Adam: L2 (weight decay) vs. Dropout vs. No Regularisation

In this section, we compare L2 regularisation using weight decay to using dropout, as well as using no regularisation. In our analysis, we select the dropout rate to be 0.1 and the weight decay to be 0.01 (based on combination 1 in the questions) and base our comparison around these hyperparameters. Table 2 below showcases the results:

	Train Dataset		**Test Dataset**		
	Mean	SD	Mean	SD	Average AUC
Adam (no regularisation)	0.97	0.0038	0.97	0.0020	0.96
Adam (L2 weight decay)	0.74	0.0035	0.75	0.0036	0.75
Adam (dropout)	0.95	0.0038	0.96	0.0015	0.95

Table 2: Accuracy for different regularisation methods for Adam, for the train and test datasets. All values are correct to 2 significant figures.

As seen from table 2, Adam with L2 weight decay performs the poorest out of the three, and there does not seem to be any major differences between Adam using dropout and Adam using no regularisation, based on the accuracy metrics. We further our comparison by looking at the performance curves as well as the ROC curves of each one. Figure 6 below displays these curves. Note that the performance and ROC curve of Adam with no regularisation is shown in figure 4 previously.

Looking at the performance curves of all three methods, we see that Adam with L2 regularisation improves its accuracy very slowly compared to Adam with dropout and Adam

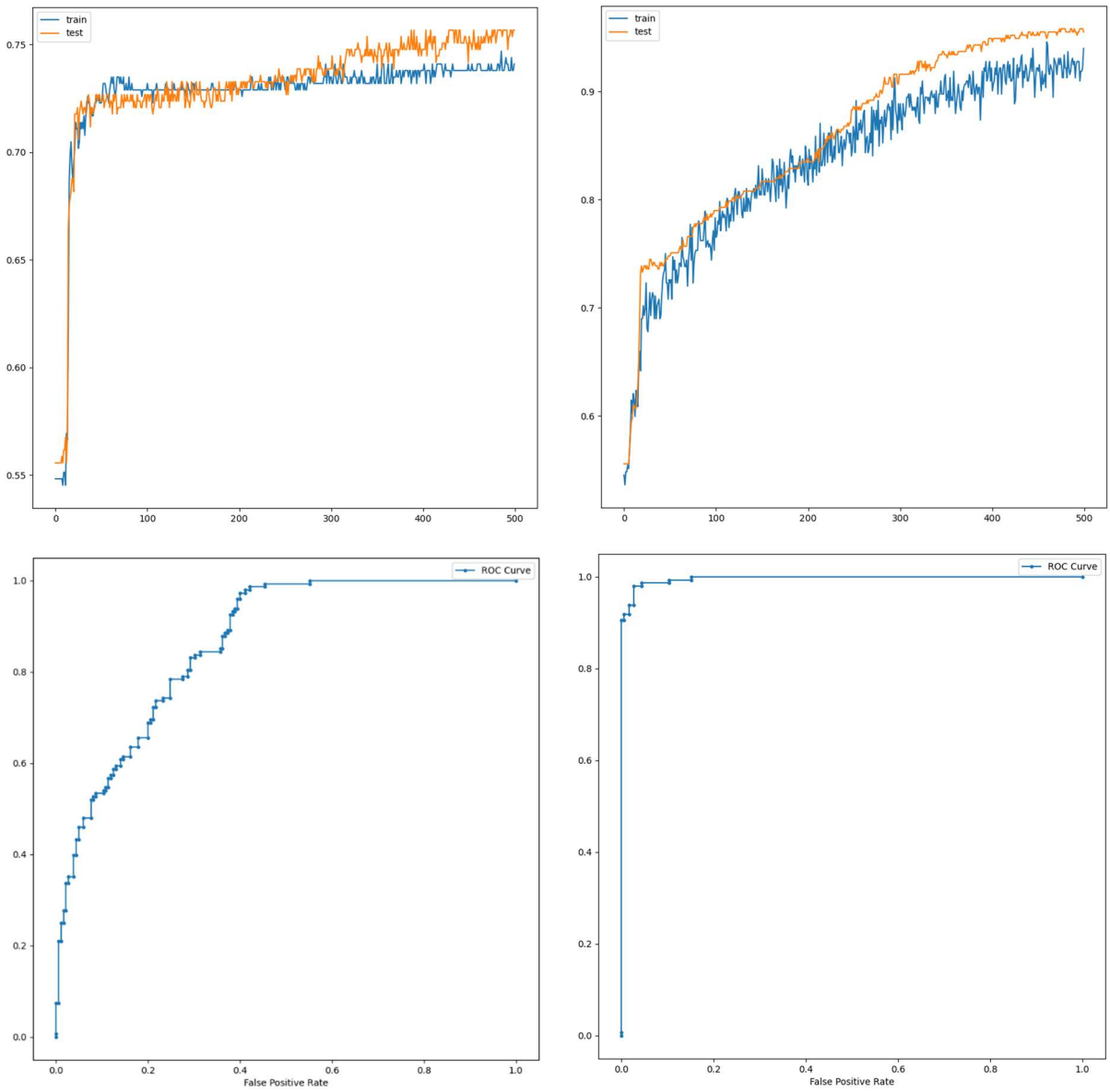

Figure 6: Performance and ROC curve of Adam with L2 (weight decay) (left) and Adam with dropout (right) over 500 epochs for one run.

without regularisation. This is likely due to the fact that L2 regularisation tends to reduce overfitting very drastically, causing the time to reach a desired accuracy to be higher. In comparing between Adam with dropout and Adam without regularisation, we see a higher degree of stochasticity in Adam with dropout, and this is expected as Adam without dropout increases the generalised performance of the model. Next, we can see that the performance curves for Adam with L2 weight decay and with dropout both display stochasticity, meaning that both methods are able to improve for more generalised data and are aimed at reducing overfitting. In terms of predictive accuracy, Adam with dropout is the standout, as it is on par with Adam without regularisation in regards to accuracy itself, but is able to account for generalisation.

Looking at the ROC curves, we see that the ROC curve for Adam with L2 weight decay deviates drastically from the top left corner, as opposed to the ROC curves the other two methods, suggesting a large difference in predictive accuracy. The ROC curves is similar between Adam with dropout and Adam without regularisation, which parallels our findings based on the accuracy metrics provided in table 2.

Holistically, Adam with dropout is the best model to use, as it is able to reach a high level of predictive accuracy, together with higher generalisation performance.

Discussion of Limitations and Future Directions

In regards to Adam without regularisation, the major problem is that it does not generalise well, and can easily overfit, which the performance curves were suggestive of. Ways of mitigating this problem include the use of regularisation methods such as L1 and L2 (with or without weight decay), as well as methods to improve generalisation such as dropout.

Next, in regards to Adam with L2 weight decay, one potential issue is that Adam with L2 weight decay is implemented suboptimally in known algorithms and methods, which leads to suboptimal results [2]. Another issue found in [2] was that the optimal weight decay was dependent on the number of epochs, and this is often neglected in training the model. All of this can cause the model to perform suboptimally. [2] also offers some possible solutions, revolving around altering the way that weight updates are performed, as well as normalising the values of weight decay.

Furthermore, in regards to Adam with dropout, a problem was identified in [3] in that dropout regularisation typically increases the training time by 2-3 fold, and this lowers the overall efficiency of the model. To combat this, it proposes to use more efficient regularisation methods that are equivalent to a dropout layer.

In summary, future considerations involve improvements on the methods used in this report, as well as applying new models, such as a Bayesian Neural Network, or utilising SGD with momentum for comparison. Other optimisers should also be considered, including AdaGrad, Ada-delta, and RMSprop. Finally, other types of regularisation can also be considered, including early stopping as well as data augmentation [4].

Conclusion

In this report, we have explored a neural network model utilising SGD and Adam optimisation methods, together with L2 weight decay and dropout methods in combatting overfitting, and this was fitted to a credit data set. Our findings were suggestive of Adam with dropout being the best model. However, this could still be improved, and future considerations include using more efficient optimiser algorithms, together with more robust and efficient methods of regularisations.

Michael Luan (z5160351)

References

1. Exploring Optimizers in Machine Learning - https://heartbeat.comet.ml/exploring-optimizers-in-machine-learning-7f18d94cd65b
2. Fixing Weight Decay Regularisation in Adam - https://openreview.net/pdf?id=rk6qdGgCZ
3. A Simple Introduction to Dropout Regularization (With Code!) - https://medium.com/analytics-vidhya/a-simple-introduction-to-dropout-regularization-with-code-5279489dda1e
4. Types of Regularization Techniques To Avoid Overfitting In Learning Models - https://analyticsindiamag.com/types-of-regularization-techniques-to-avoid-overfitting-in-learning-models/
5. Code adapted from the following:
   Exercise 2.1 solution - https://edstem.org/au/courses/6212/lessons/13873/slides/107809
   Exercise 3.1 solution - https://edstem.org/au/courses/6212/lessons/13874/slides/100348
   Exercise 4.2 solution - https://edstem.org/au/courses/6212/lessons/13875/slides/100361