Neural Networks

Denis Tsvetkov and Lukas Ostrovskis

November 2022

1 Choosing the features

As it is mentioned in the assignment neural networks tend to be heavier to train and having a lot of invaluable features (for example the 50 address variables that have the same frequency of loan statuses as the rest of the data) will definitely harm the run time performance and cause the model to overfit (see Brownlee). That is why we decided to pick the 10 important features based on the regularization models from Assignment 2 and our decision tree from Assignment 3. The features we ended up with, that yield the highest AUC score and most consistent accuracy, were the following: **small_business**, **int_rate**, **fico_range_avg**, **Not Verified**, **MORTGAGE**, **avg_cur_bal**, **dti**, **term**, **acc_open_past_24mths**, and **mort_acc**. We already provided reasoning about **small_business** and **int_rate** (see Assignment 2, section 4), however, we will now examine the other features in Figures 1 to 5. Given the time constraints, we only focused on the distribution of loan statuses for those features. Note, **dti** and **avg_cur_bal** were chosen solely based on the previous 2 assignments, as they had too many unique values to be plotted in the same way as the other features.

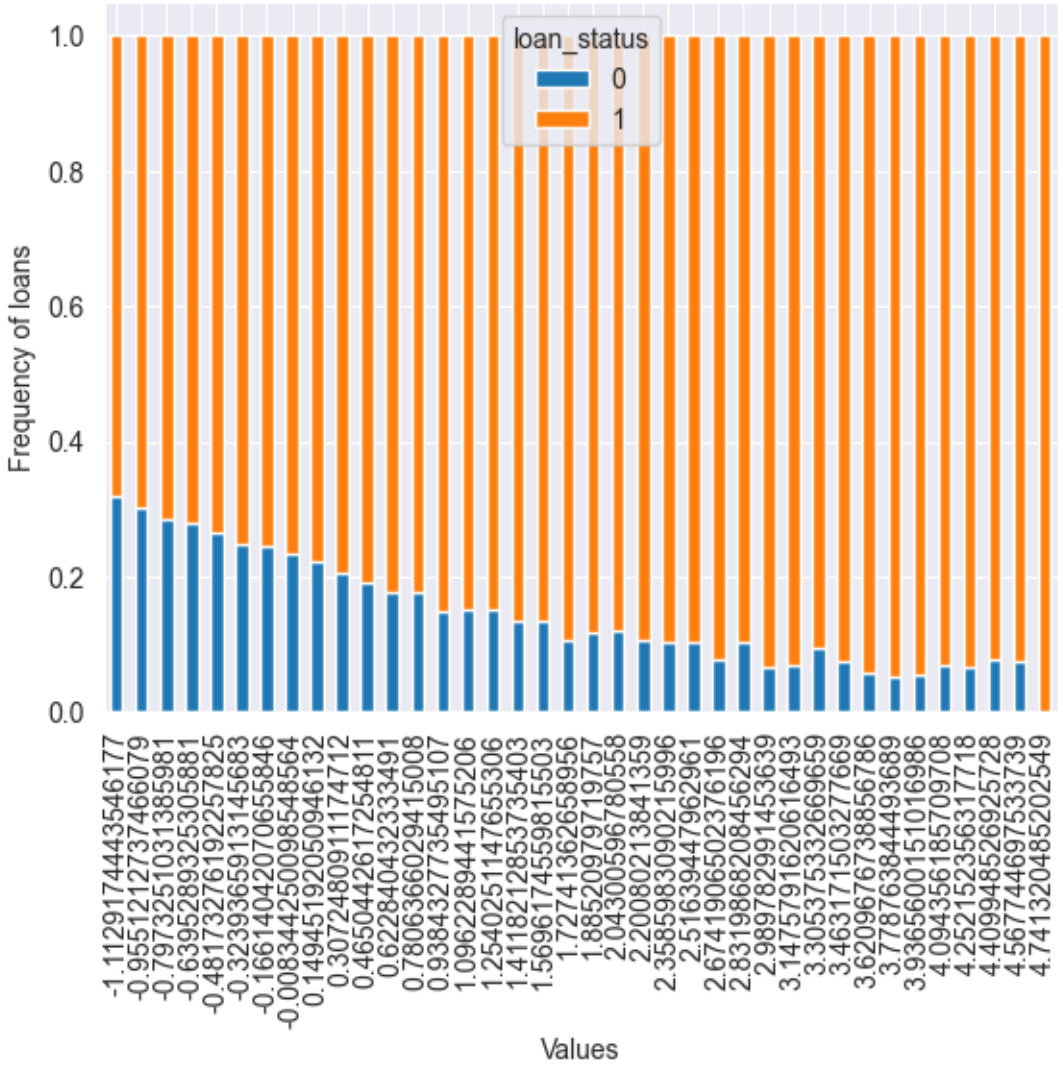

Figure 1: Distribution of loan statuses for the **fico_range_avg** feature

We can see from the plot that the larger the values get the higher the chance of a loan being paid out. Useful feature for detecting loans that will not default.

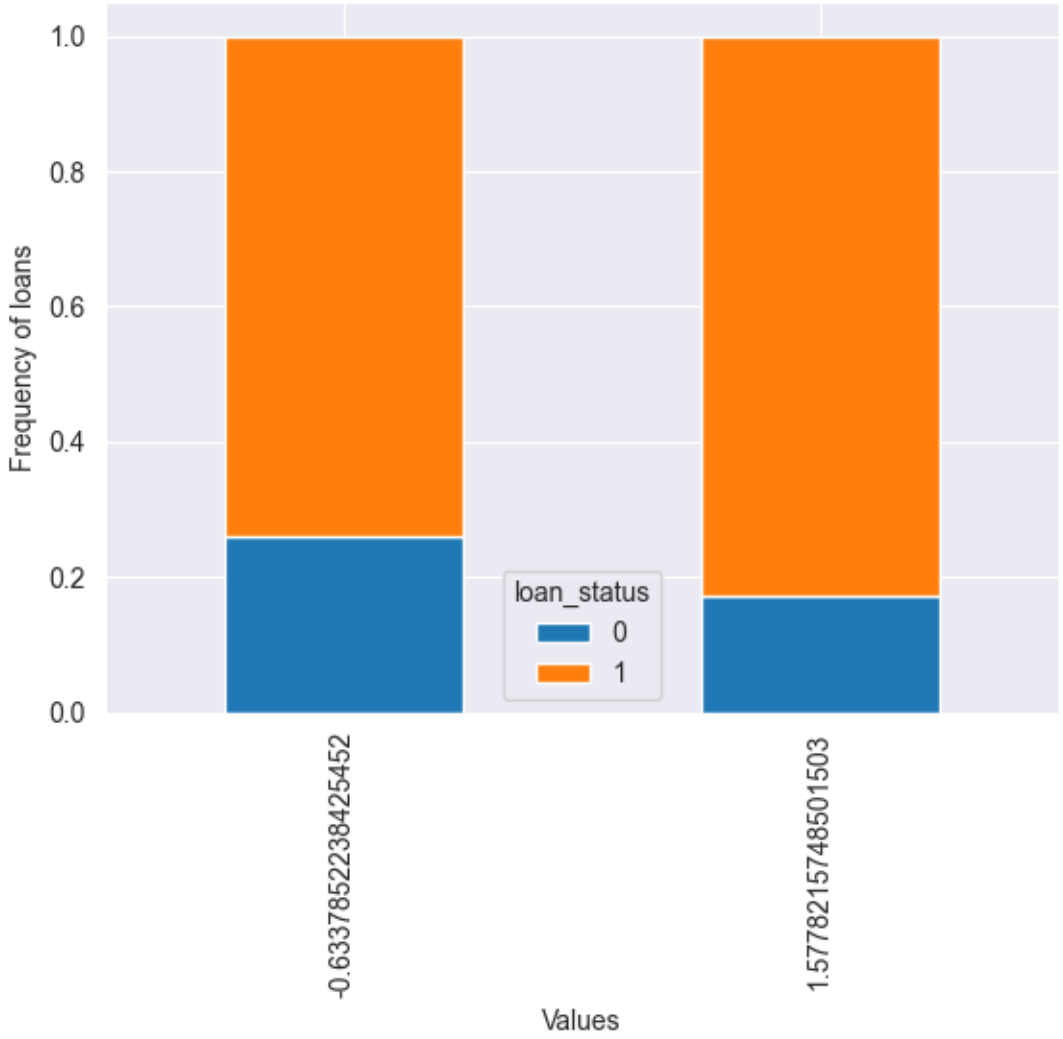

Figure 2: Distribution of loan statuses for the **Not Verified** feature

Verified applicants tend to be more likely to pay out their loans than *Not Verified*. It appeared to us as an important feature for the model. Note, similar distribution of loan statuses can be found for the **MORTGAGE** feature as well.

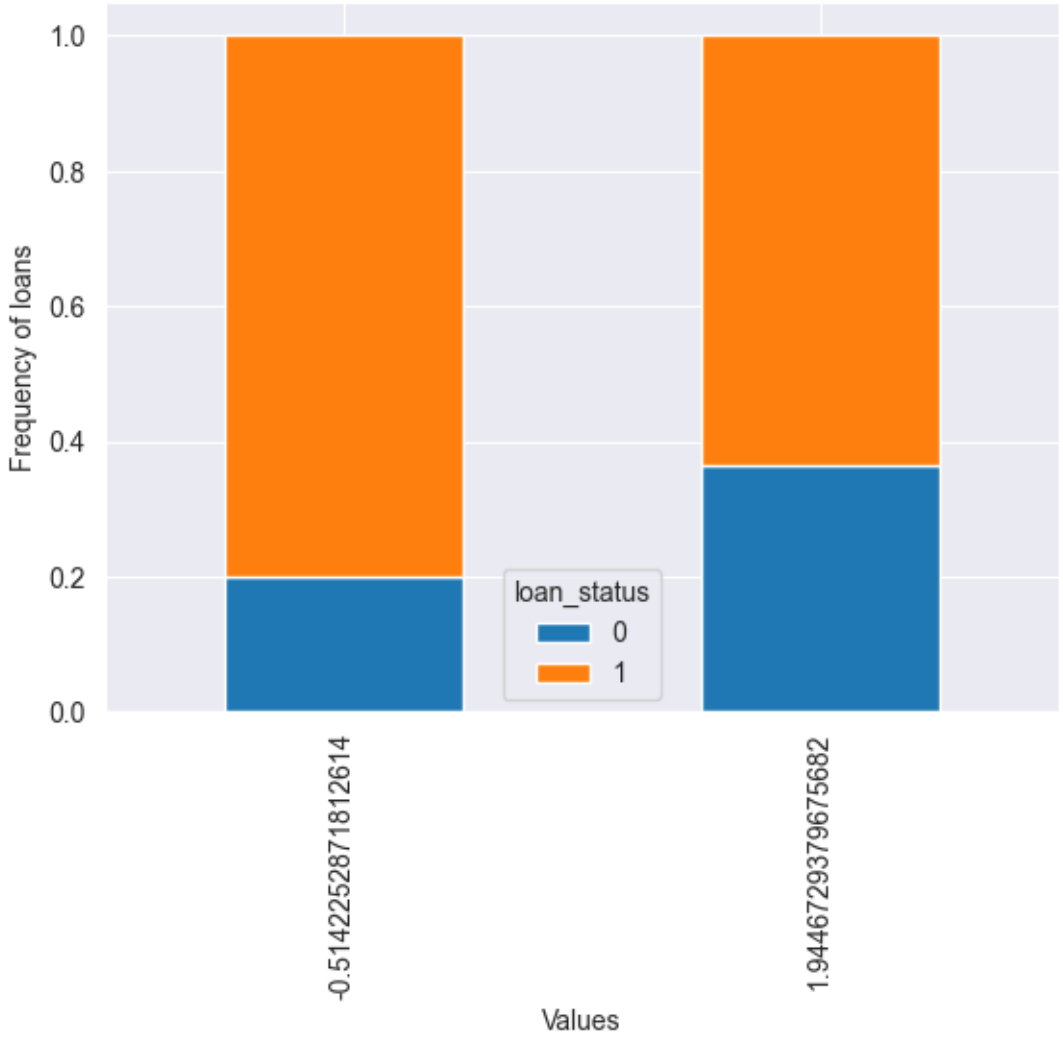

Figure 3: Distribution of loan statuses for the **term** feature

In contrast to the the previous plot, long-term loans of 60 months default more often than their alternative (36 months). Important feature for detecting loans that will default.

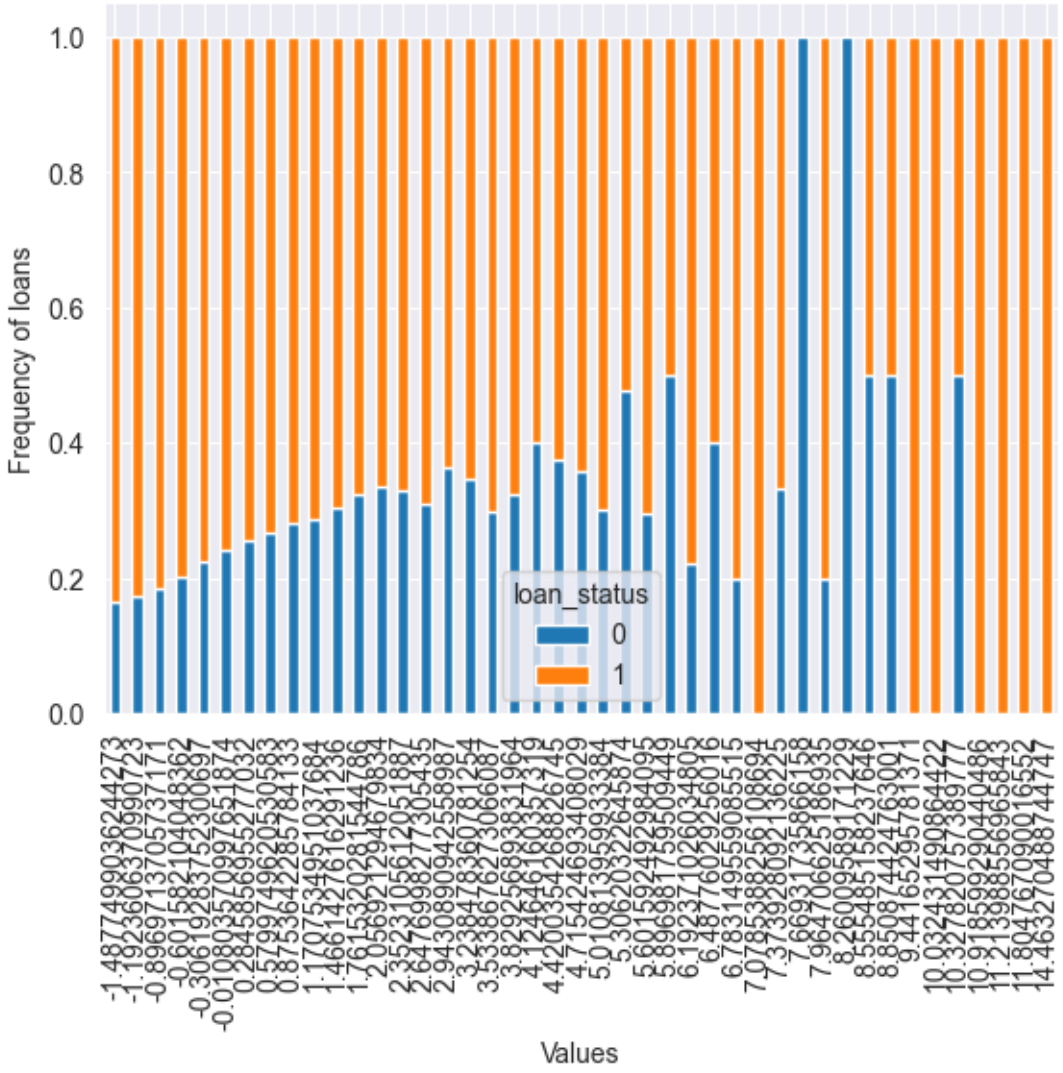

Figure 4: Distribution of loan statuses for the **acc_open_past_24mths** feature

We can see that the higher the values get the more likely it is for a loan to default (with exceptions, although the values over 8 consist of 1 - 2 instances). Vital feature for detecting loans that will default.

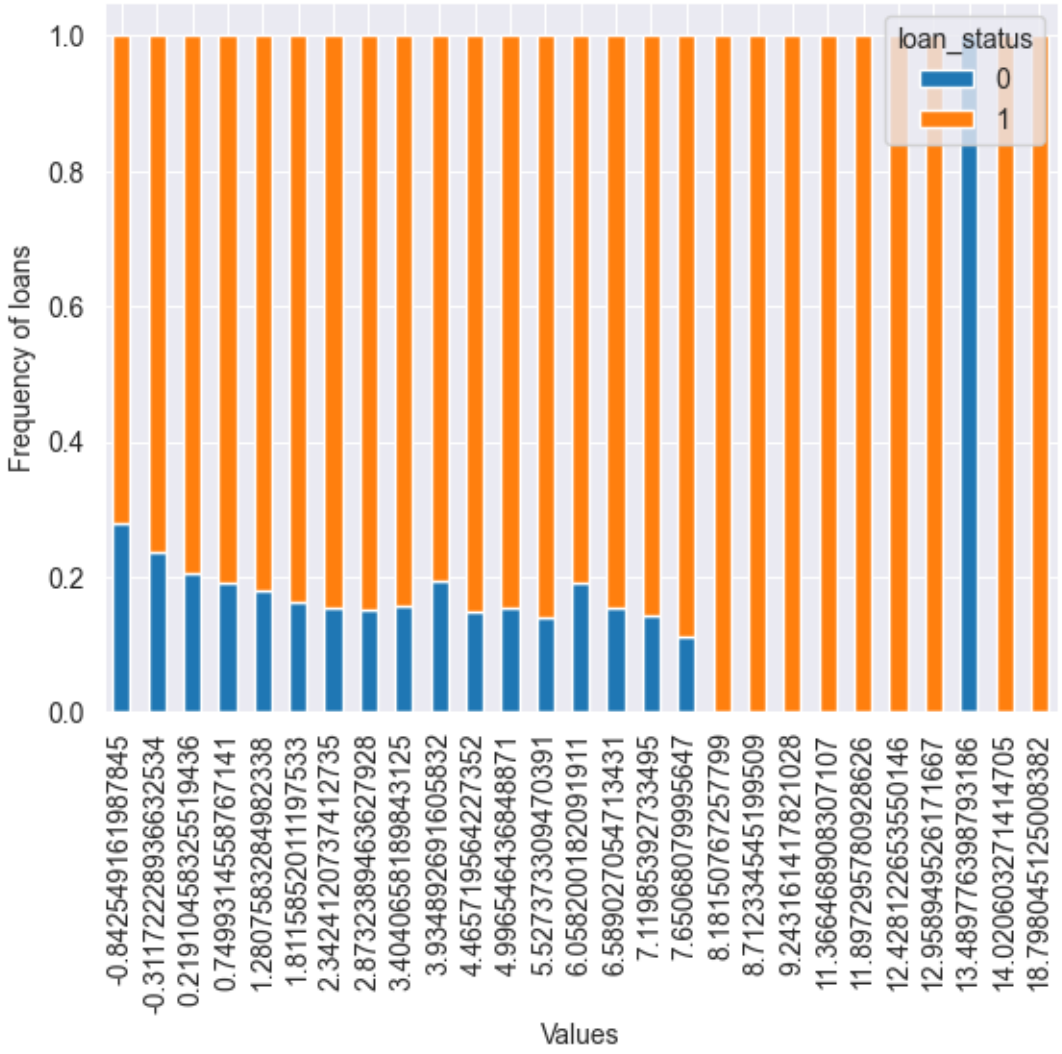

Figure 5: Distribution of loan statuses for the **mort_acc** feature

Finally, contrary to the last feature, for **mort_acc** the higher the value the more likely it is for a loan to be paid out (with one exception at 13.489, which is a single instance of a loan that defaulted). Important feature to identify loans that will not default.

2 Training a neural network with a single hidden layer

We chose the following architecture for this simple neural network:

- **sigmoid as activation function** - a good choice in our case as its output is in the range of 0 and 1, meaning that it can be treated as the probability, and given some threshold we can directly map it to one of the 2 classes (0 or 1);
- **35 neurons in the hidden layer** - from other projects we had reached to the conclusion that numbers between 1.5 and 4 times the number of input features worked best. We decided to use 35 at random for this, however this value will be further optimized in section 3;
- **randomly generated weights and biases;**

- **50 epochs of training;**
- **batches of size 10000** - for a model with a lot of samples as ours a large batch size is to be expected. We opted for a roughly 8 percent batch size (compared to the number of samples) as it is enough for learning progress to be made, but not enough for the exceptions in the data to affect the weights (as the changes would be averaged out by the other samples in the batch);
- **constant learning rate of 0.1** - we chose a *larger* learning rate as we opted for fewer iterations, but we still wanted to have an accurate and well-trained model at the end;
- **early stopping** - to prevent overfitting we decided to implement early stopping - after each epoch we would check the accuracy of the model against a validation set. Then, if we have reached *5* consecutive epochs with no increase in the accuracy we would terminate the training.

The model yielded the following results: it took 22 epochs to be trained (with the early stopping) and it managed to increase its accuracy against the validation set from roughly 38 percent to around 70 - see Figure 6.

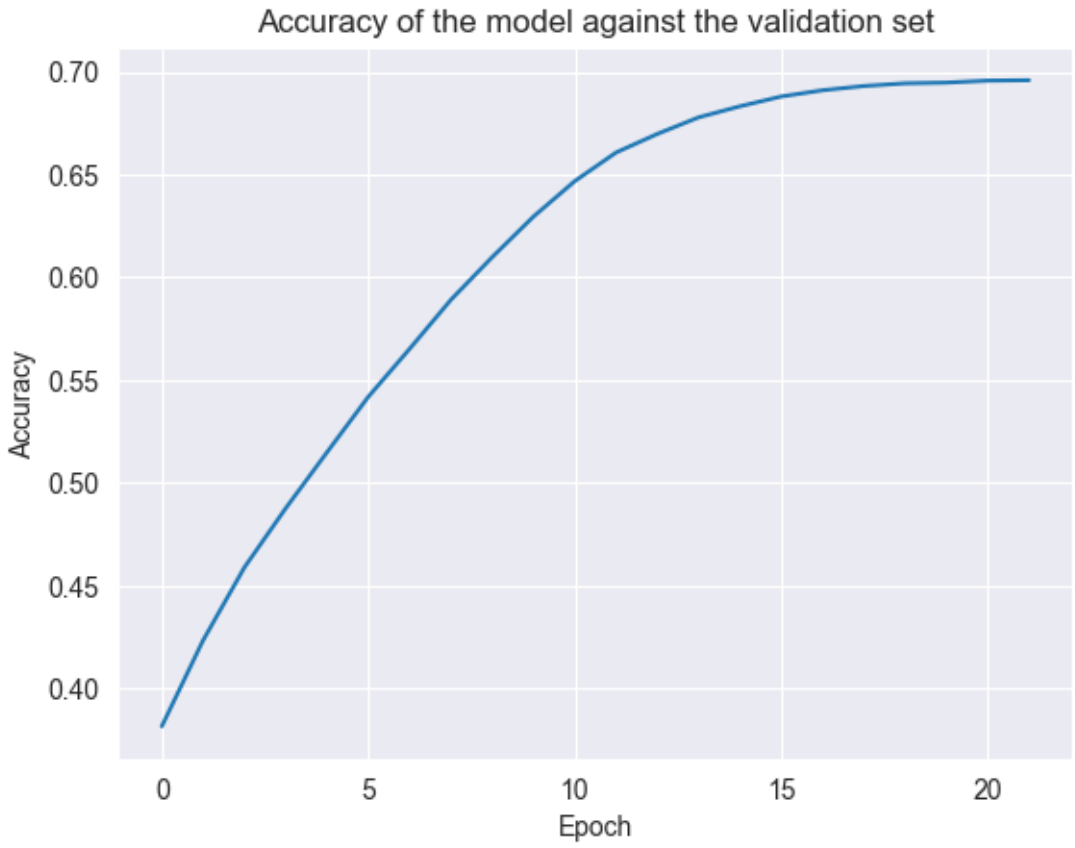

Figure 6: The accuracy of the model during training

The accuracy of our model is lower than that, because for our test set we are using the optimal threshold from the previous assignments (0.8) and that one only yields an accuracy of 0.54. Furthermore, the ROC curve can be seen in figure 2 with an AUC equal to 0.57. The neural network performed poorly compared to our models from the previous 2 sections, as for those we had AUCs of around 0.71 - 0.72. Based on this alone, we can conclude that the models from the previous 2 assignments might be better suited for the data than this simple neural network.

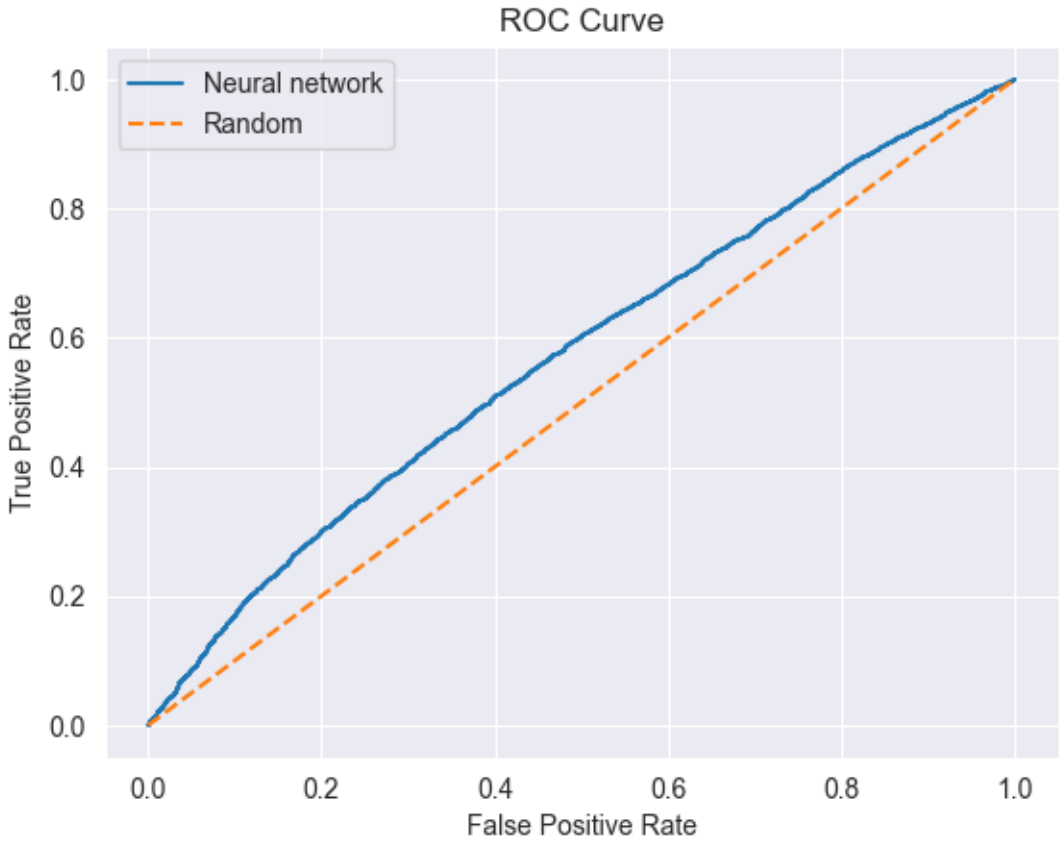

Figure 7: The ROC curve for the neural network

Finally, the precision of the neural network was 0.80 and the true positive rate (TPR), derived from the confusion matrix below, is 0.52.

13.79	9.82
36.52	39.87

Table 1: Confusion matrix of the neural network's predictions (using a threshold of 0.8)

3 Refining the model

The hyperparameters that needed to be optimized were the following four: the activation function, the size of the hidden layer, the learning rate, and the batch size [1]. So we decided to perform the so called grid search to find the optimal set of parameters. The values that we chose to test against were the following:

- **activation function** - identity, logistic (sigmoid), tanh, and relu (these were the options offered by the *MLPClassifier* class);

[1]Note that we decided not to optimize the number of iterations, as the maximum (50) was not reachable given our early stopping condition.

- **hidden layer size** - 15, 25, 35, 45 (we tried to stay between the 1.5 to 4 times the input range), and we decided to also try 1 neural network with 2 hidden layers (the first one having 25 neurons and the second one - 10);
- **learning rate** - apart from the constant learning rate, we tried using an adaptive one with initial values of 0.01, 0.1, and 0.25;
- **batch size** - 6000, 10000, and 12000.

The grid search identified this set of parameters as optimal: relu as the activation function, a single hidden layer with 45 neurons, adaptive learning rate with initial value of 0.01, and finally a batch size of 6000. The new model was able to increase its accuracy to 58 percent (not that much but still to be expected with this threshold). The AUC under the ROC plot below is 0.71, which means that this improved model is just as optimal at predicting the loan statuses as the ones from the previous assignments. Finally, the precision and the TPR improved to 0.88 and 0.53 respectively (the confusion matrix can be found in table 2).

18.1	5.51
35.66	40.73

Table 2: Confusion matrix of the optimized neural network's predictions (using a threshold of 0.8)

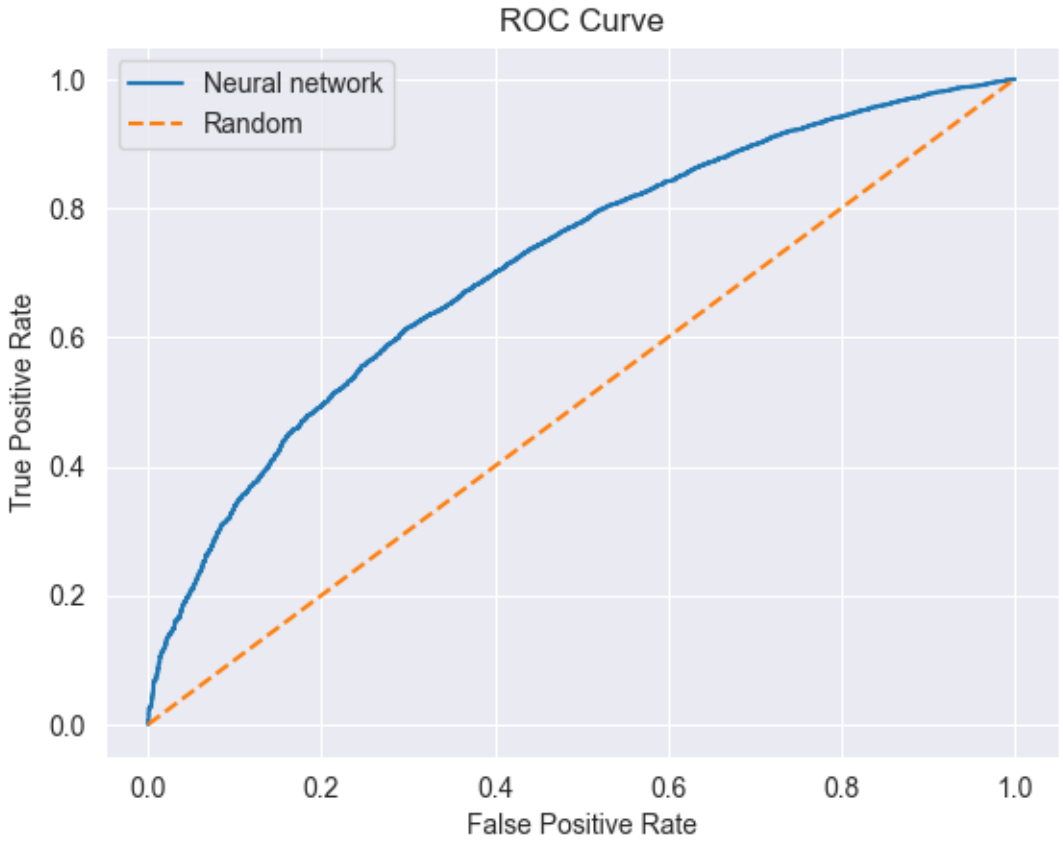

Figure 8: The ROC curve for the optimized neural network

References

[1] Brownlee, J. (2020). Data Preparation for Machine Learning.