Tarun Dhillon

Modelling Credit Approval using Neural Networks

Introduction

Neural networks are a class of models that can be used for a wide variety of machine learning problems. One of these machine learning problems is classification. This report presents the results of applying simple neural networks to predict whether credit will be approved or not. The data used for this report is the 'Credit Approval Data Set' which contains attributes relating to credit card applications.

Data

The data has a total of 690 observations, 15 features and 1 label column representing whether the credit application was approved (positive result) or not approved (negative result). The feature names and symbols have been changed in the original data to protect confidentiality. As such, it is not possible to know what each feature represents. The feature columns were assigned names arbitrarily as A1 to A15 (inclusive) and the label column was assigned the heading A16.

Data Processing

Data processing was carried out to ensure the data was in a suitable format to be used by the neural network model. The processing steps were carried out in three steps as mentioned below.

Step 1

Some issues were found in the raw data including missing values which were recorded as "?" and some numeric data represented in string format rather than numeric (float or integers). The table on the following page summarises the issues found and the treatment of these issues.

Feature Name	Data Type	Rows with Missing Data	Treatment
A1	Nominal (2 categories)	12	These rows were removed
A2	Continuous	12	Data changed to numeric and missing values imputed to column average.
A3	Continuous	None	Data changed to numeric
A4	Nominal (3 categories)	6	These rows were removed
A5	Nominal (3 categories)	6	These rows were removed
A6	Nominal (14 categories)	9	These rows were removed
A7	Nominal (9 categories)	9	These rows were removed
A8	Continuous	None	Data changed to numeric
A9	Nominal (2 categories)	None	None
A10	Nominal (2 categories)	None	None
A11	Continuous	None	Data changed to numeric
A12	Nominal (2 categories)	None	None
A13	Nominal (3 categories)	None	None
A14	Continuous	13	Data changed to numeric and missing values imputed to column average.
A15	Continuous	None	Data changed to numeric

Step 2

The nominal data type features were re-mapped to numeric type arbitrarily. For example, the raw data for feature A1 had 2 categories, namely, “a” and “b”, these were re-mapped to 0 and 1. Similar re-mapping was carried out for the other nominal data type features.

Step 3

Finally, feature scaling was carried out using Min-Max normalisation to ensure the feature values were contained in the same [0,1] range. Without normalising the data, the neural network can potentially take a longer time (number of epochs) to train. This is due to the gradient descent algorithm oscillating between steps before it steps towards the minimum. Thus normalising the data helps the neural network train faster.

The processed data had 671 observations, 15 features and 1 label. The data can be found in the data.csv file.

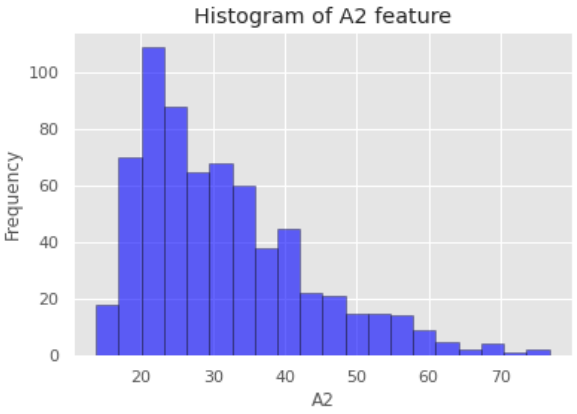
Tarun Dhillon

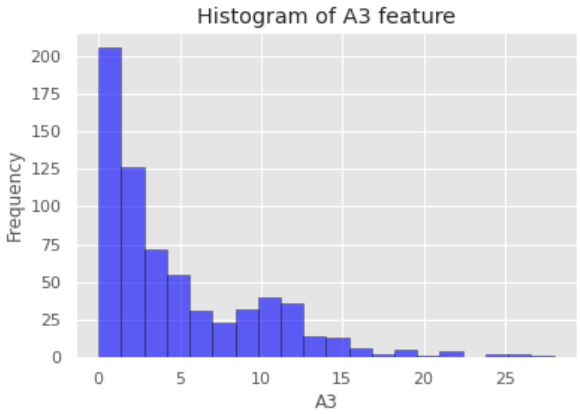
Visualisation of Data

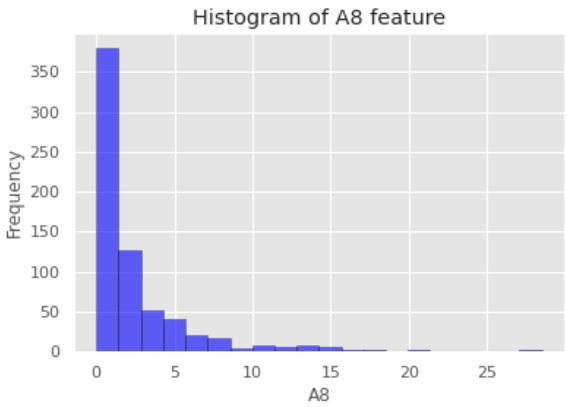
As mentioned above, some features are of nominal data type while others are continuous. As such, different visualisation tools were used to represent the features.

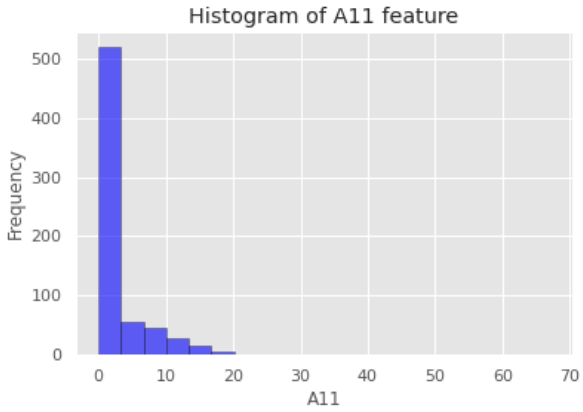
Continuous features

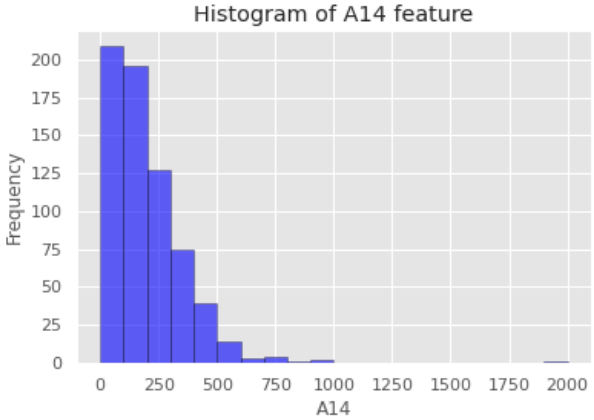
Histograms were plotted for the continuous features which are in the blue colour. Please note that the scale used was the original scale before applying the min-max normalisation as this better represents the values across which the features' values are distributed. The continuous features are A2, A3, A8, A11, A14, A15.

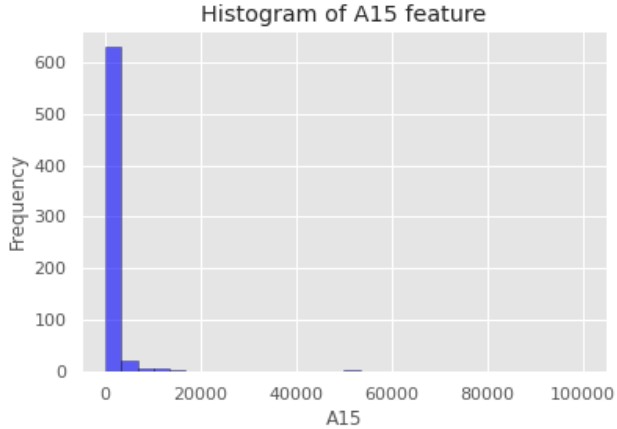








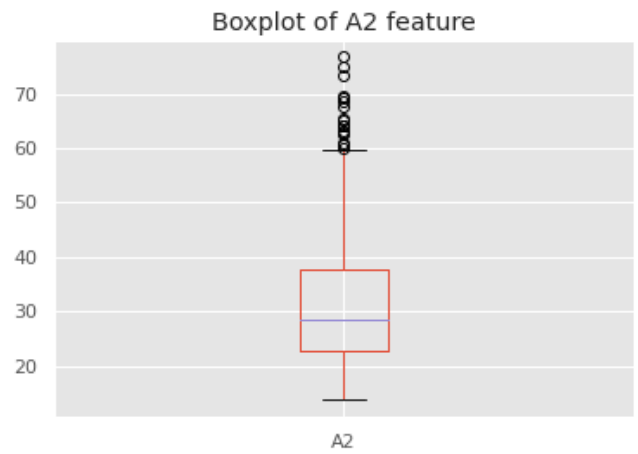


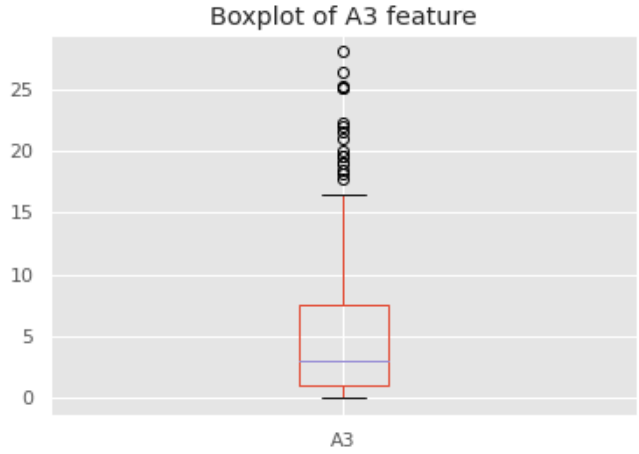


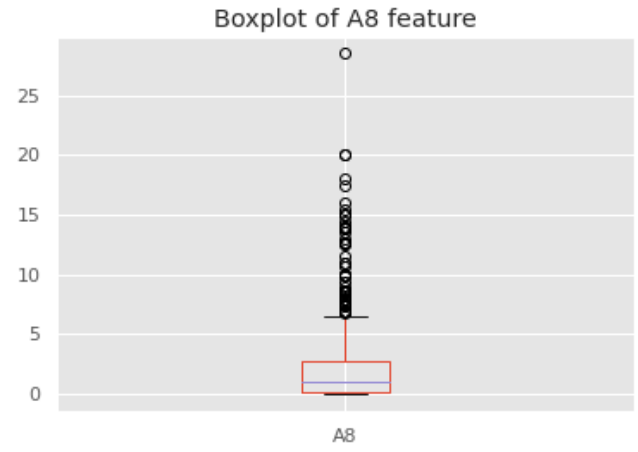
Observations

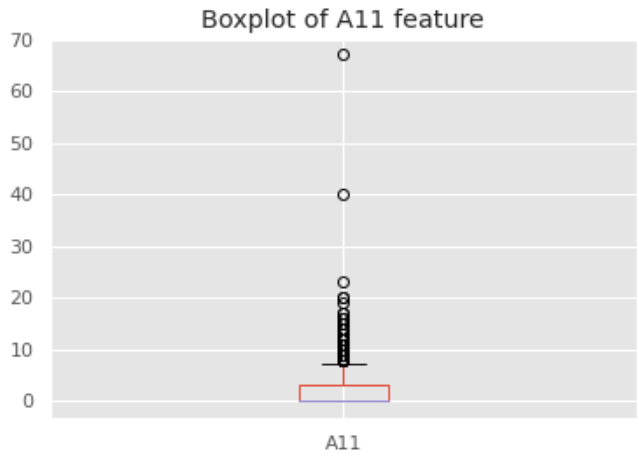
It can be clearly seen from the histograms that there is negative skew in all the continuous data type features. However, it should be noted that the apparent skew is exaggerated due to the presence of outliers, for example in the A8 feature histogram there are outliers beyond the value 20. Similarly, there is an outlier for the A14 feature around the 1900 feature and an outlier above the 50000 value in the A15 feature histogram.

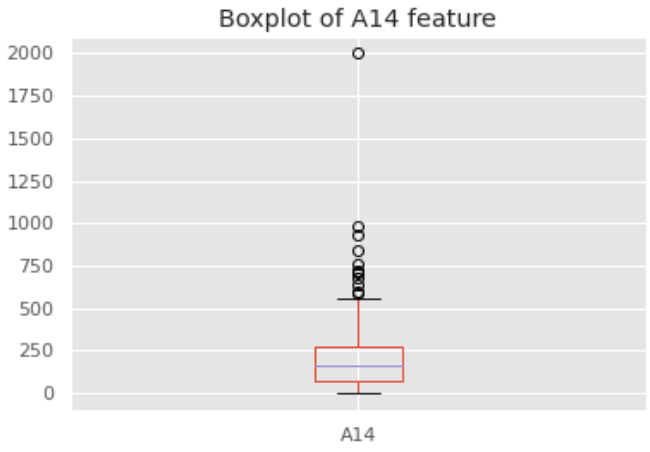
Further, the range of values for the A15 histogram go up to 100000, which indicates that there is at least 1 value around 100000 however due to it having a low frequency, it is not visible on the histogram. As such, to better visualise the outliers, boxplots were also plotted for the continuous type features which are shown below.

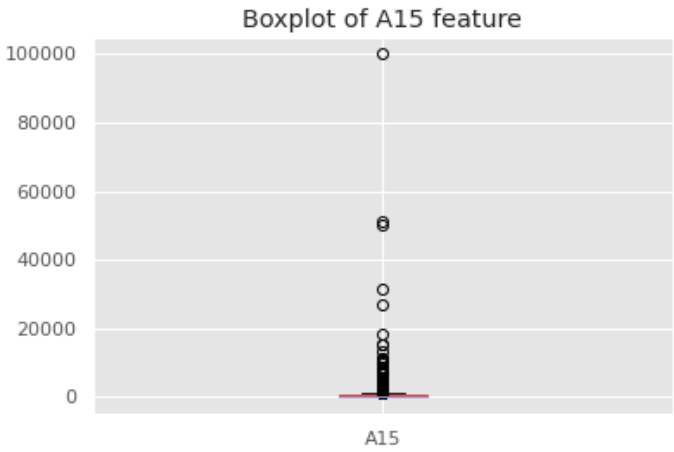












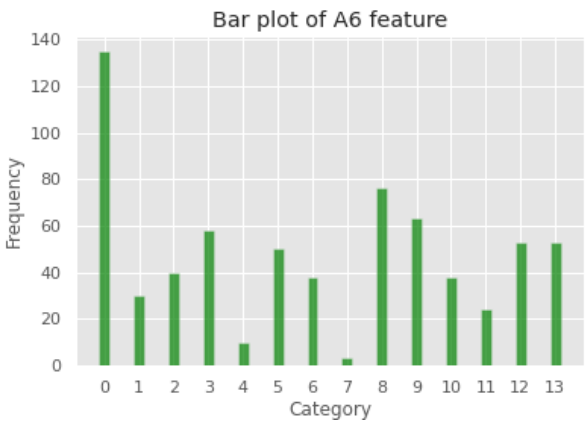
Observations

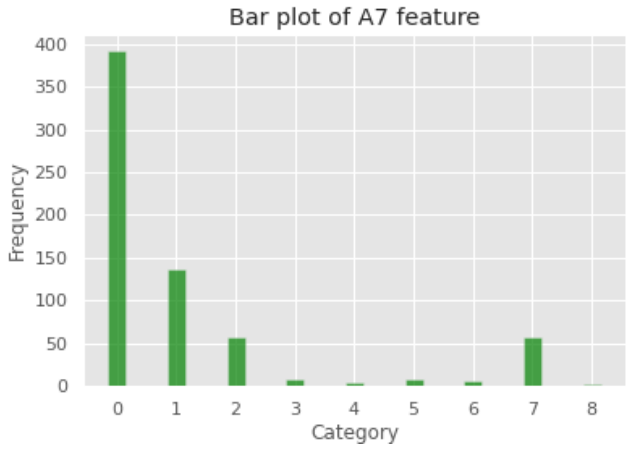
The outliers suspected in the histograms are clearly visible in the boxplots for the continuous features, most notably for the A11 feature. These outliers are causing the continuous data type features to have large positive skew in the data. There are numerous treatments that can be applied on any given dataset to deal with outlier values. The treatment selected however, depends on the nature of the feature. For example, if one of the features in this dataset was the age of the applicant, it is impossible that a value for this feature can be negative or above 100 say. In this case, it is reasonable to remove any rows from the dataset for which this feature has a negative value as it is clearly incorrect.

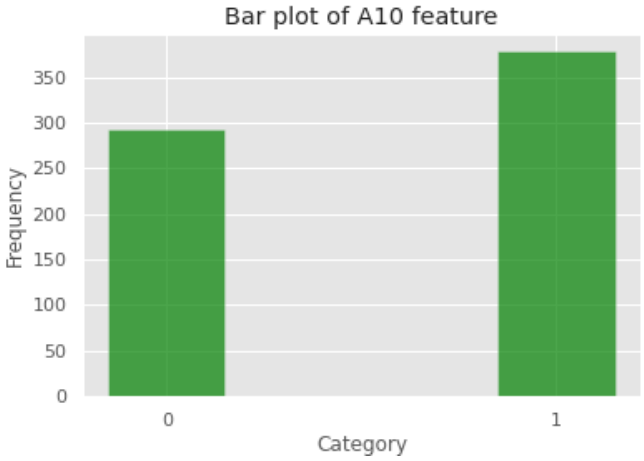
However, as mentioned above, the information on what each feature represents has been concealed for confidentiality reasons and as such it is not possible to be sure about the legitimate range of values that these features can take. For this reason, the outliers, while being clearly visible, have not been removed or treated in any way for the purposes of this modelling exercise.

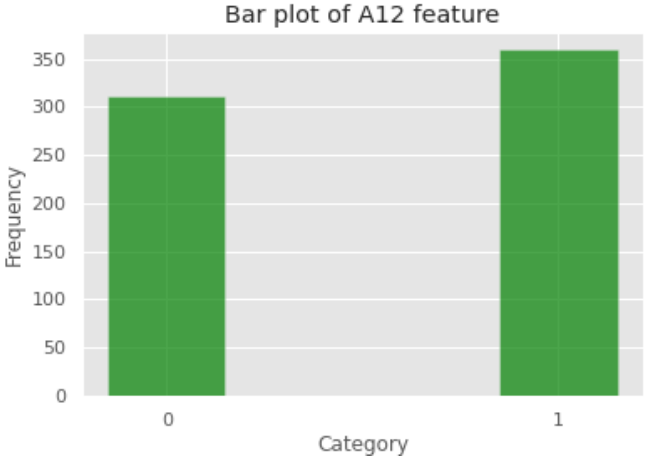
Nominal features

Bar plots were used for the nominal data types to reflect the frequencies of each of the categories within the nominal data type features, the bar plots are shown in green**.** Please note that the scale used was the original scale before applying the min-max normalisation. The nominal features are A1, A4, A5, A6, A7, A9, A10, A12, A13.







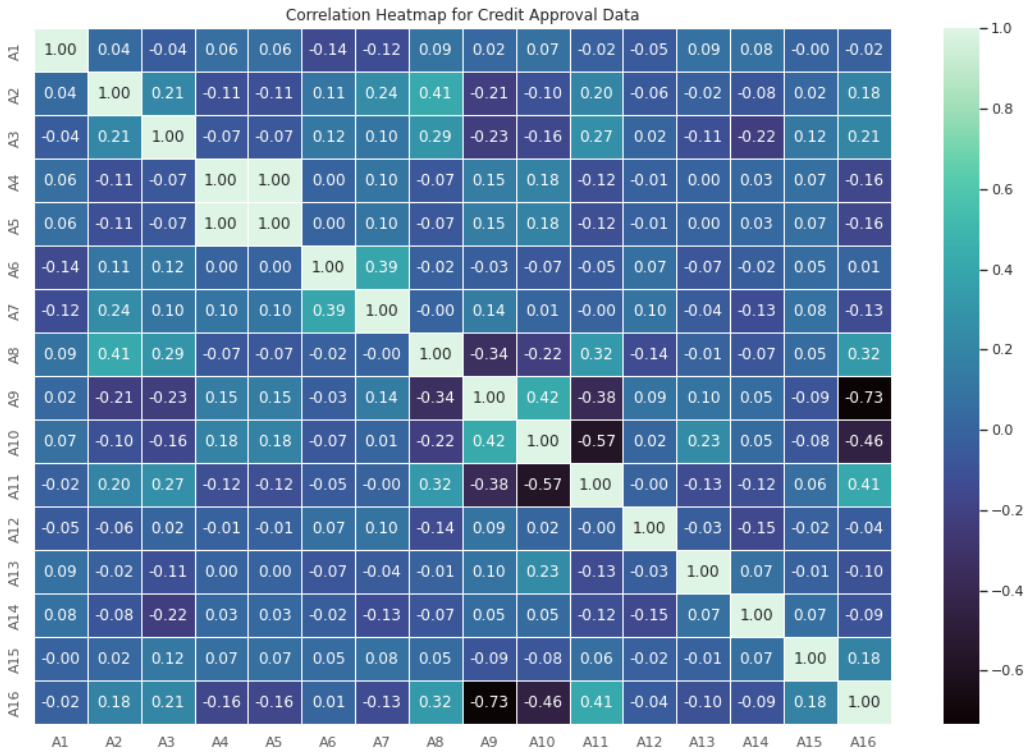


Observations

It can be seen that there is unevenness in the distribution of all the nominal features. As in the case of the continuous type data, there seem to be outliers in some nominal features as well, for example in A13, the category mapped as “1” has only 1 or 2 observations. Similarly, this is the case with A4 and A5 for category mapped as “2”. These could be outliers, or incorrect values however without knowing the nature of the attributes this judgement is not possible to be made legitimately. For the purposes of modelling, these were left as such.

Further, a bar plot is also shown for the label feature A16. There are more credit non-approvals (371 observations with label 0) than approvals (300 observations with label 1). This suggested that when splitting the dataset into train and test data, stratified sampling should be used to ensure the proportion of approvals to non-approvals in maintained both in the test and training set.

Correlation Heatmap



Observations

It can be seen from the correlation heatmap that none of the features are significantly correlated with the target label except feature A9 which has a high negative correlation with the target label. This suggests that a simple linear model is not appropriate for this task, a more complex model such as a neural network is needed with non-linearity incorporated into its design (through activation functions and hidden layers). A further observation can be made that feature A4 and A5 are perfectly correlated. Upon further investigation, it became apparent that the original values for these features exhibit the same pattern in the sense that whenever the raw value of feature A4 is “u”, the value for feature A5 is “g” and so on for all the categories. However, as the nature of the attributes is not known, no treatment was given to these features (for example, removing one of them) and both features were kept in the data as in.

Modelling

The neural network was built using the Keras API in Python.

Splitting the dataset

- The data was split into 50% training set and 50% test set. As such, there were 335 observations in the training set and 336 observations in the test set.
- The data was shuffled before splitting
- Stratified sampling was used as mentioned above.

Neural Network Architecture

The neural network used for the modelling had the following architecture:

- An input layer to receive 15 inputs, one from each of the features
- A single hidden layer with the ReLu activation function.
- The number of neurons used in the hidden layer were 10. This calculation was done using the following formula:

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

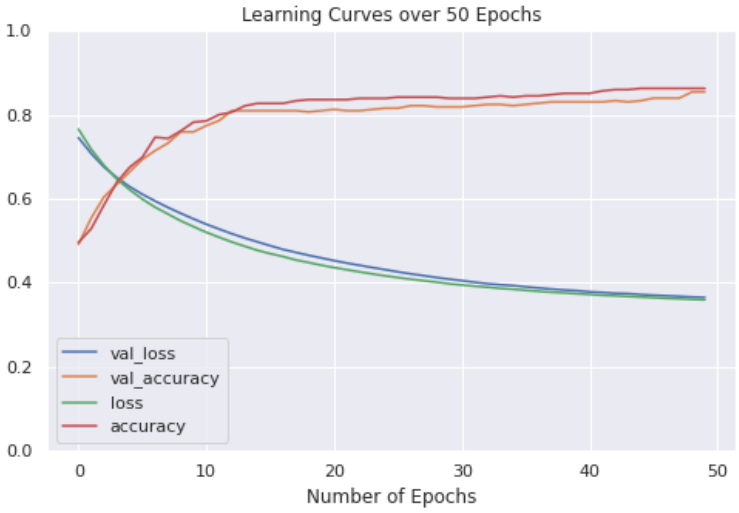
Ni = number of input neurons.
No = number of output neurons.
Ns = number of samples in training data set.
α = an arbitrary scaling factor usually 2-10.

  - Ni = 15
  - No = 1
  - Ns = 335
  - α = 2
  - As such, Nh = 335/(2*(15+1)) ~= 10
- As this modelling exercise is a binary classification exercise, the activation function used in the output layer was the sigmoid function.
- As this modelling exercise is a binary classification exercise, the loss function used is the binary cross-entropy function.

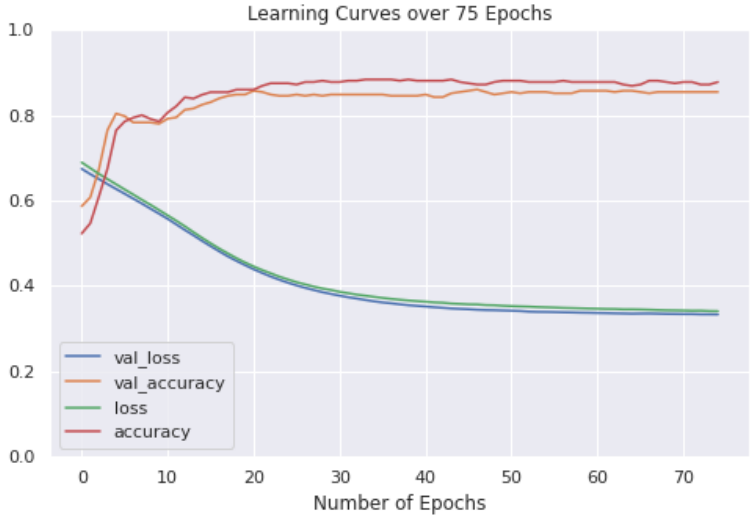
To determine the training time, that is, number of epochs, a number of neural network models were fitted with different number of epochs. The learning curves for these networks showing the training and validation loss and accuracy are presented below.

Tarun Dhillon

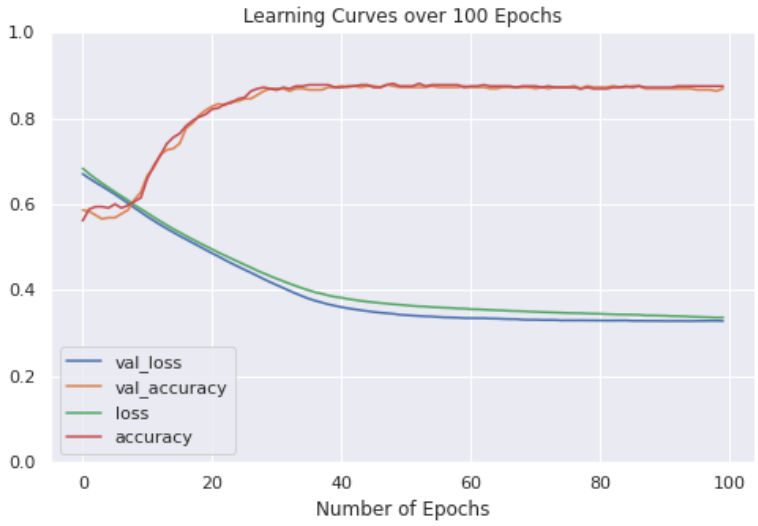
50 Epochs



75 Epochs

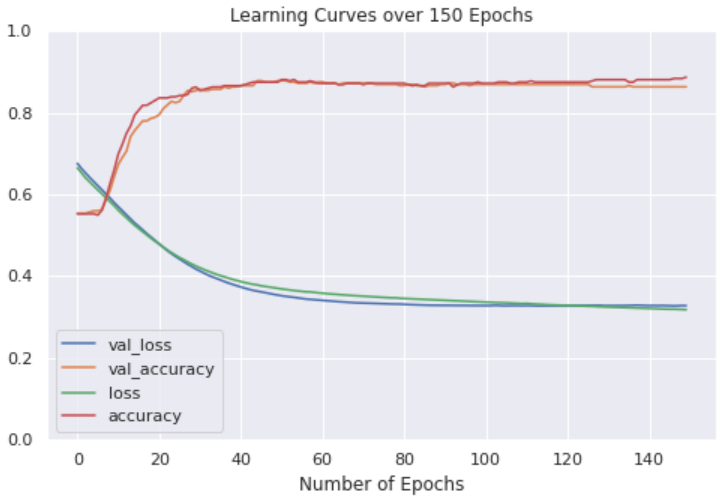


100 Epochs

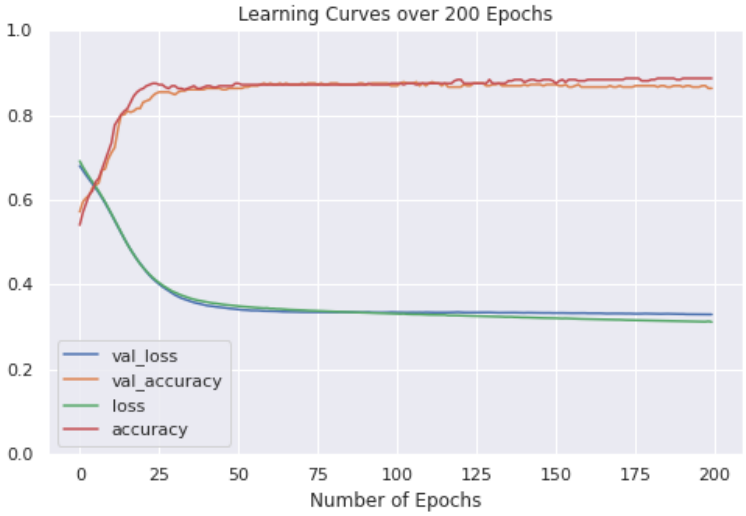


Tarun Dhillon

150 Epochs

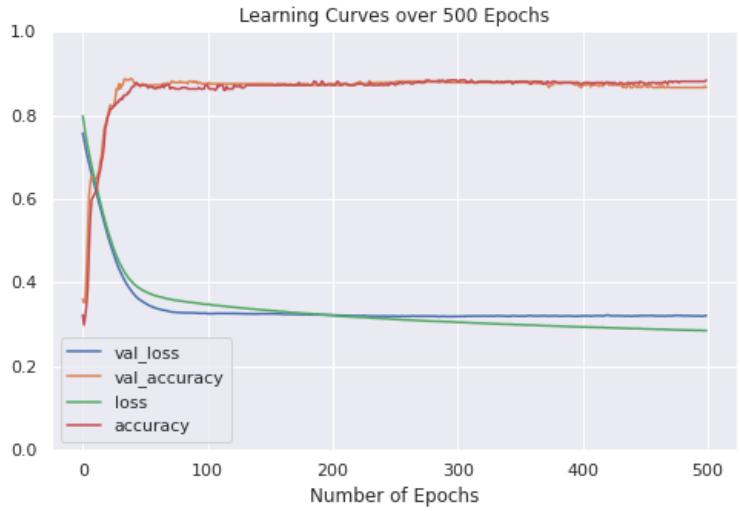


200 Epochs



Tarun Dhillon

500 Epochs



Observations

- With 50 epochs, the training and validation loss still seem to be decreasing indicating that the epochs should increase.
- With 75 epochs, the training and validation loss are starting to plateau indicating that indicating that the number of epochs may be approaching an appropriate number. Moreover, the training and validation loss curves are trending close to each other indicating that there is no overfitting with 75 epochs.
- With 100 epochs, the training and validation loss and accuracy seem to have completely plateaued and there is no noticeable change from the previous run with 75 epochs.
- With 150 and 200 epochs, there is still no decline in the training or validation loss or an increase in the accuracy.
- Similarly with 500 epochs, there does not seem to be much change in the metrics mentioned above.
- As such, 100 epochs were decided to be a reasonable training time and was used in the subsequent tests.

Comparison of SGD with ADAM

ADAM and SGD are both optimization algorithms used to train neural networks. Twenty experiments were carried with the same neural network specified above with the only change being the optimization algorithm used, SGD and ADAM. The training and test accuracies were calculated for each of the experiments with both SGD and ADAM. Note that the same train-test split was used to compare SGD with ADAM and for each iteration of the experiment the train-test split was changed. The results are given in the table below.

ADAM Accuracy on Train Set	ADAM Accuracy on Test Set	SGD Accuracy on Train Set	SGD Accuracy on Test Set
0.866	0.866	0.857	0.869
0.869	0.866	0.845	0.872
0.896	0.836	0.887	0.842
0.866	0.866	0.851	0.839
0.875	0.851	0.854	0.869
0.896	0.854	0.612	0.589
0.887	0.869	0.851	0.863
0.866	0.863	0.827	0.863
0.890	0.866	0.857	0.839
0.887	0.839	0.884	0.845
0.869	0.869	0.812	0.824
0.881	0.854	0.851	0.827
0.869	0.875	0.866	0.845
0.875	0.860	0.854	0.842
0.899	0.848	0.884	0.845
0.878	0.845	0.884	0.848
0.872	0.878	0.860	0.851
0.878	0.854	0.839	0.848
0.881	0.848	0.854	0.818
0.887	0.860	0.857	0.854

- The mean for the ADAM training accuracy is: 0.879
- The mean for the ADAM test accuracy is: 0.858
- The mean for the SGD training accuracy is: 0.844
- The mean for the SGD test accuracy is: 0.835

Observations:

It can be observed that the mean for the accuracy on the training set is higher than on the test set for both ADAM and SGD optimization. This is expected as a model has already been exposed to the training set and hence learns from it.

It can also be observed that the accuracy on both the training and test sets is higher for ADAM optimization as compared to SGD. This is expected as ADAM is a faster optimizer than SGD and hence can learn faster in the same number of epochs. However, the difference is not too significant, it is however, noteworthy.

Impact of Learning Rate on Test and Train accuracy using SGD

Three different learning rates were tested with the SGD optimization, across 10 different experiments. AS before, the train-test data was changed for each experiment. The three learning rates tested were:

- 0.001
- 0.01
- 0.1

The results are given in the table below.

Experiment	Training Accuracy LR: 0.001	Test Accuracy LR: 0.001	Training Accuracy LR: 0.01	Test Accuracy LR: 0.01	Training Accuracy LR: 0.1	Test Accuracy LR: 0.1
1	0.731	0.690	0.839	0.863	0.881	0.872
2	0.576	0.563	0.875	0.857	0.869	0.851
3	0.603	0.607	0.845	0.866	0.854	0.878
4	0.499	0.500	0.887	0.833	0.907	0.833
5	0.681	0.699	0.875	0.833	0.893	0.848
6	0.758	0.702	0.857	0.830	0.890	0.872
7	0.567	0.557	0.878	0.848	0.881	0.842
8	0.493	0.557	0.845	0.798	0.884	0.869
9	0.499	0.539	0.887	0.860	0.904	0.848
10	0.522	0.545	0.851	0.866	0.851	0.872

- The mean for the training accuracy for learning rate 0.001 is: 0.593
- The mean for the test accuracy for learning rate 0.001 is: 0.596
- The mean for the training accuracy for learning rate 0.01 is: 0.864
- The mean for the test accuracy for learning rate 0.01 is: 0.846
- The mean for the training accuracy for learning rate 0.1 is: 0.881
- The mean for the test accuracy for learning rate 0.1 is: 0.859

Observations

It can be seen from the mean accuracies that the accuracy on both train and test sets is much lower when the learning rate is 0.001 as compared to other learning rates. This indicates that the learning rate of 0.001 is too small and the SGD algorithm has not found the minimum of the loss function in 100 epochs.

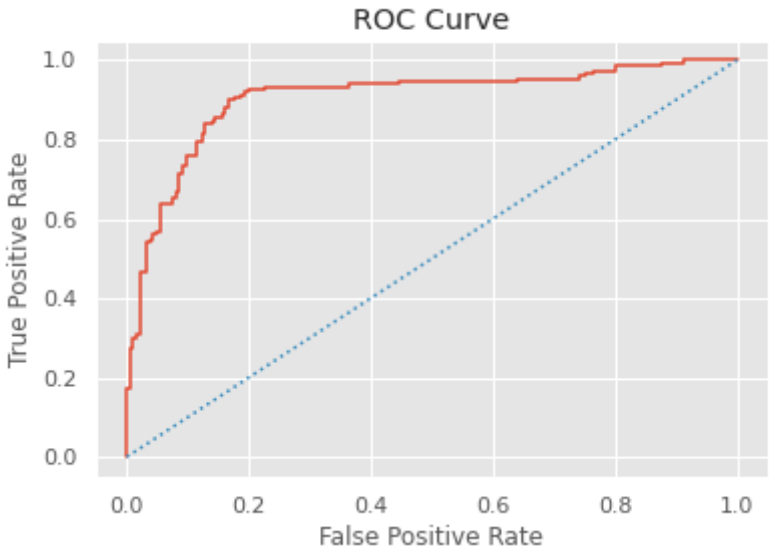
The mean accuracies are higher when the learning rate is 0.01 however they are still higher when the learning rate is 0.1 reflected in the highest train and test accuracies. The SGD algorithm is learning faster (that is, each step is larger) and this is closer to the minimum in 100 epochs than with other learning rates.

Confusion Matrix with SGD when Learning Rate: 0.1

As mentioned above, the highest mean training and test accuracy is from the neural network model with learning rate = 0.1. The confusion matrix for the classification resulting from this model is given below.

Confusion matrix	Actual Not Approved	Actual Approved	Total
Predicted Not Approved	150	36	186
Predicted Approved	12	138	150
Total	162	174	336

Receiver Operator Curve with SGD when Learning Rate: 0.1



The area under the curve 0.905.

Future work

We were only able to get accuracy up in the high eighty percent mark, as such further improvements could be made by increasing the complexity of the neural network by increasing the number of neurons or increasing the number of hidden layers. Further, no hyperparameter tuning was carried out even though some hyperparameter combinations were tested. This can also be conducted in future with a view to further improve the results. Furthermore, other modelling techniques such as SVM, random forests can also be incorporated in a future expanded scope for this data

Conclusion

This report presented the results of applying a simple neural network to predict whether credit will be approved or not. We presented standard visualisations of the features and some observations pertaining to these visualisations. The report then presented the neural network architecture that was used and how it was determined. Specifically, the number of epochs to use to train the model was discussed and results of the experimental runs to determine this were presented. Eventually, 100 epochs were decided to be used in model training. Further, tests were conducted to compare the performance of different optimization algorithms, specifically SGD and ADAM and the effect of learning rate on the model accuracy. Accuracy scores in the high 80% were achieved. Further work as highlighted above can be undertaken to improve the model's performance.