Assignment 2 Write-Up

Introduction

Feed Forward Neural Networks are a popular deep learning algorithm used for a variety of purposes such as regression and classification problems. They are an ideal model due to their ability to take into account and learn more complex relationships within data than a simple linear or logistic regression equation. There are some unique features that allow FFNN's to perform better than simpler models. The main one is backpropagation which is the process in which a Neural Networks learn the models features. It consists of a forward pass to calculate the loss, a backwards pass which computes the gradient of the loss function with respect to all the different parameters in a network, and finally uses gradient descent to update the weights and the biases. Activation functions are also used to make the data interpretable for the next layer and to shape the output depending on the type of problem. Optimizers are used to adjust the learning rate and aim for the global minimum of the model. In this case, three Feed Forward Neural Networks are used for regression, binary classification, and multiclass classification. Each dataset provides enough complexity (large dataset, many predictors) in order to warrant using a neural network as opposed to a simpler method.

Datasets

Dataset I - Bank Customer Churn

In order to complete the three models, two datasets were used, 'Bank Customer Churn" for regression and binary classification, and "Healthcare Analytics" for multiclass classification. In the customer churn dataset, there were originally 14 features and 10,000 observations. For both models the 11 features used were "CreditScore","Geography","Gender","Age", "Tenure", "Balance", "NumOfProducts", "HasCrCard", "IsActiveMember","EstimatedSalary", and "Exited", only leaving out variables that were unique to each observation and wouldn't be useful information to the model. The name of each is fairly straight forward, but to elaborate on a few, "Tenure" is the time in years that customer was with the bank, "NumberOfProducts" is the number of other services/products that the bank is providing that customer (e.g. loans, fixed deposits), "Excited" represents if a customer withdrew from the bank (0 for no, 1 for yes) and "EstimatedSalary" is a calculated variable based on a ratio between the customers balance and their corresponding salary (from discussion section in Kaggle). The first step in assessing the data was through visualization. Histograms were made to show the distribution of the features. Many seemed to have a fairly normal distribution, but some were skewed one way or another. A few notes are that Estimated Salary is very evenly distributed, almost uniform, which is unusual due to the fact that Balance is very skewed, having over 3000 customer balances equal to zero. This leads me to question what EstmiatedSalaray actually represents and how it is calculated. It is also important to note that the outcome variable for the binary classification model, Excited, is very skewed with 80% being observations that did not result in churn. Another thing to assess the data was with a correlation matrix which shows that none of the variables are highly correlated with one another. To make the data interpretable to the model, the categorical variables (Geograph, NumOfProduct, HasCrCard, IsActiveMember, Exited) were one-hot encoded using the get_dummies() method and merging the results with the continuous variables.

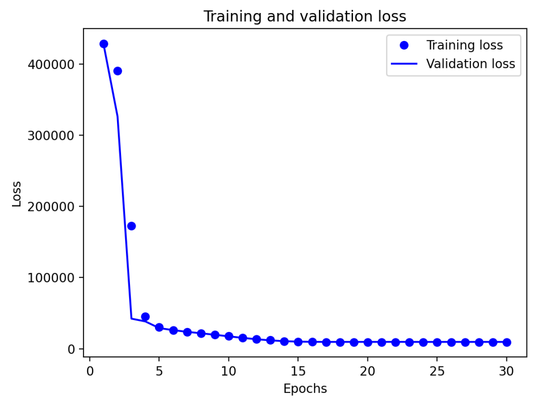
Dataset II - Healthcare Analytics

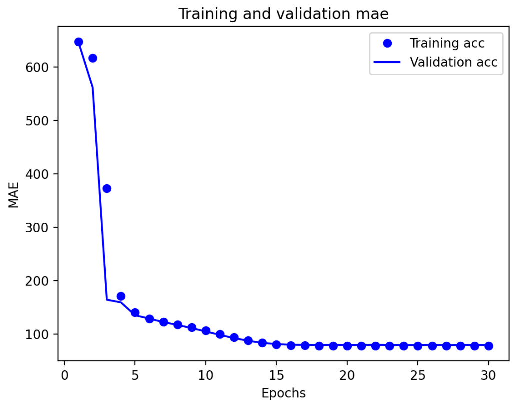
The “Healthcare Analytics" dataset consists of 13906 observations and 18 features. It is important to note that in the original Excel file holding the dataset, I randomly selected the 13,906 rows using a generator. The original dataset had over 300,000 observations and my computer continuously crashed when trying to manipulate that magnitude of data in Jupyter. I understand that this may lead to some of the poor results seen in the model discussed in later sections. The number of features was also narrowed down to 10. The variables removed were either unique to each observation or after trial it was found that they either reduced or didn’t not improve the performance of the model, likely due to a strong correlation with other variables. The final set of features are "Available Extra Rooms”,"Department","Ward_Type","Bed Grade","Type of Admission","Severity of Illness", "Visitors with Patient","Admission_Deposit", “Age”, and "Stay". To elaborate, “Bed Grade" is a continuous variable representing the condition of the bed in that particular hospital, the “Type of Admission” is categorical with the outcome being either Emergency, Trauma, or Urgent, and “Stay” is the length of time (11 discrete ranges) in days of how long that patient was at the hospital. Like the previous dataset, histograms were used in order to visualize the distribution of each variable. Some observations are that Visitors with Patient, Extra Rooms in Hospital, and Stay were all very left skewed. With Stay being the target variable, I remade the bins into 3 discrete ranges instead of 11 so that the observations were more evenly distributed. All of the categorical variables (Department, Ward_Type, Type of Admission, Severity of Illness, Age) were also one-hot encoded using get_dummies.

Methods

Regression

In order to build the regression model, the Bank Customer Churn dataset was used with the outcome variable CreditScore. As discussed in the dataset section, the data was first visualized to assess patterns and the categorical features were turned into dummy variables. The data was split into three parts: train, validation and test. The training data consists of 80% of the total observations, validation contains 10%, and so does the testing data. The train, validation, and test data were z-scored to scale/standardize the data and the training set was then fit. An initial model was then created to test how training data performs against the validation set. To do so, a model was created with only two hidden layers, like the textbook recommended, with 64 nodes and relu activation functions. The output layer used linear activation. To compile the model, “adam” was used as the optimizer, mean_squared_error to assess the loss, and mean_absolute_error for the performance metric. The model also went through 30 epochs (iterations) in batches of 512. Each time I compiled the model, I played around with the number of hidden layers, the nodes in each layer, and the optimizer. To assess, graphs were printed showing the training and validation loss and MAE for each epoch to visually represent how effective the model is at learning the patterns in the data (see below). The model was continually tuned by looking at factors such as mean absolute error, loss, stability of the model, and the distribution of the predicted values. One change was made at a time in order to come to an optimal solution. More discussion on why I chose the model that I did based on the result is discussed in the results section. After assessing the performance on the validation data and making changes to the model as I saw fit, I made a final model with 6 intermediate layers, the first having 128 nodes and the following with 64 nodes, all with a relu activation function minus the output layer which uses linear activation. More layers were added compared to the initial model due to the need for more complexity. It runs through 15 epochs which was found to be the optimal solution without overfitting. It is important to note that the “elbow” of the graph was at around 5, but that only gave a mean absolute error of around 130 and could be significantly improved to only 78 with the 15 epochs. The model uses “adam” as the optimizer, MSE for the loss, and MAE as a performance metric. Finally, to understand and visualize how the model is predicting data, a graph was created showing the model's predicted test values on the x axis and the true values on y axis (see in Results section).





Binary Classification

The binary classification model uses the same Bank Customer Churn dataset as the regression model so all of the data preprocessing was the same. However, the outcome variable for binary classification is the “Exited” so “CreditScore” has now become a feature instead of the output variable. To start, the dataset was broken down into train, test, and validation sets. The training data made up 80%, test 10%, and validation 10% of the total observations. Each of the sets were then z-scored to standardize the data and the training data was fit. An initial model was then built that resembled the example in the textbook with two hidden layers, each with 16 nodes, adam as the optimizer, binary_crossentropy for the loss, accuracy for the metric value. This model was then compiled in batches of 512 over 20 epochs and compared against the validation set. To visualize the performance of the model, two graphs were made to show the training vs validation loss and the training vs validation accuracy. I continued to turn the model by looking at factors such as accuracy, loss, and stability of the model. I made one change at a time in order to come to an optimal solution. After reviewing the model and making adjustments, a final model was created. This model has four hidden layers, each with 16 nodes, rmsprop for the optimizer, and the loss and accuracy metrics remained the same. The number of hidden layers was increased to account for more complexity in the model and the optimizer was changed because convergence seemed to be a lot more stable which could be seen by the two graphs. The new model was compiled with the training data and assessed against the test data, this time with only 7 epochs and batches of 512 once again. 7 epochs were chosen because this is where the model stopped exponentially improving and less epochs prevent overfitting and save compute (see Results for graph).

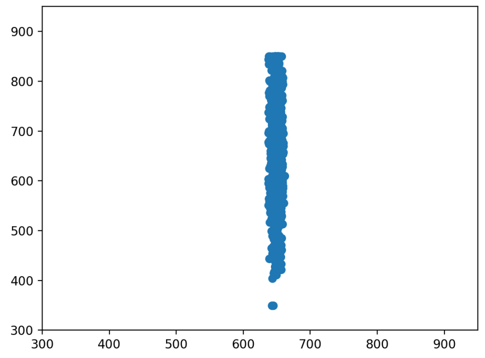
Multiclass Classification

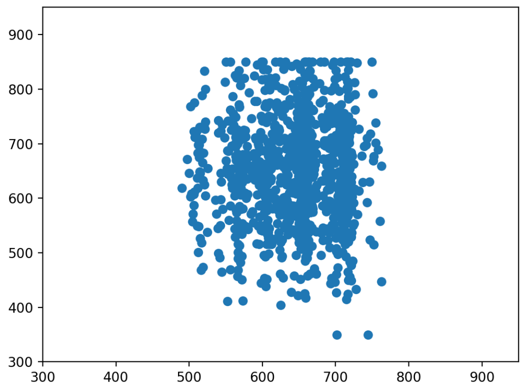
The last model was the multiclass classification model using the Healthcare Analytics dataset. The data was first preprocessed; the categorical variables were one-hot encoded and the outcome variable “Stay” was altered so the observations fell in three, evenly dispersed categories ('Stay_0-20', 'Stay_21-40','Stay_>60') instead of the previous 11. Next, the data was broken down into train, test, and validation sets which were then z-scored, making the ranges more interpretable to the model. Then an initial model was made with two hidden layers, each with 64 nodes using relu activation, and an output layer with three potential outputs using softmax. The model was compiled using adam as the optimizer, categorical_crossentropy for the loss, and accuracy as the performance metric. It was tested against the validation data with 20 epochs and batches of 512. In order to view the performance graphically, the training loss was plotted along with the validation loss for each epoch. Another graph was made showing the progression of the training and validation accuracy. I continued to tune the model by looking at factors such as accuracy, loss, and stability of the model. I made one change at a time in order to come to an optimal solution. After some tuning, a final model was made. This model contains five hidden layers, each with 64 nodes using relu for the activation, and an output layer with nodes equal to the number of classes (3) using softmax. The addition of hidden layers helped enhance the complexity of the model. The final model was compiled using rmsprop as the optimizer, categorical_crossentropy for the loss, and accuracy as the metric. The training data was fit to the model and assessed against the test data with 4 epochs and batches of 10. The number of epochs was chosen using the “elbow method”. This was the inflection point where the error and the loss plateaued and stopped decreasing. After about 4 epochs, the loss increased and the validation scores jumped all over the place, possible due to overfitting (see Results section).

Results

Regression

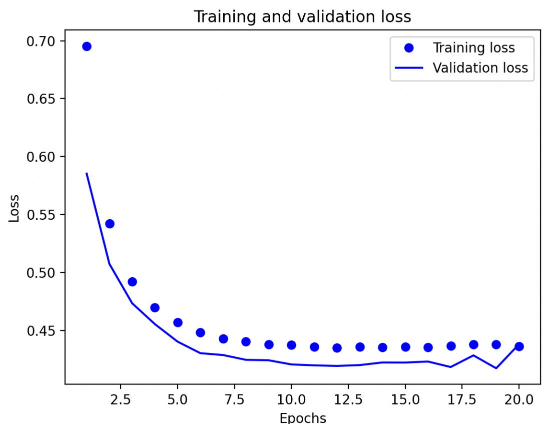
In the final model, it was able to obtain a 80.3502 mean absolute error on the testing data with a 13368.2266 mean squared error. Overall, this is not a terrible performance. Considering that the range of CreditScore values is between 350 and 850, a 500 point difference, an error of 80 means that the model is performing decently. This first time the initial model was run using the textbook example with only two layers, the model did not perform as well and the minimum mean absolute error that it achieved was just above 131. To adjust for this, more layers were added and, but ultimately stopped at 6. More than 6 layers tended to either not improve the model or the validation loss would greatly increase, a result of potential overfitting. Additionally, the model was tested using both rmsprop and adam. While the two optimizers had similar mean absolute error, the loss was significantly higher when using rmsprop. The graphs below show the true (y axis) vs predicted (x axis) values of CreditScore. You can see that the graph on the left using rmsprop is essentially only predicting the mean value of the data which is 650. The model using adam on the right predicts a wider range of credit scores. This could be due to the nature of the optimizers in that rmsprop uses the second moment with a decay rate to speed up from AdaGrad. Adam uses both first and second moments. Additionally, because some of the predictors are very skewed, this could also lead to the model being unable to interpret the relationship of the features to the output. It makes sense that the model is predicting credit scores close to the mean and on the higher end of the range due to the fact that the original data is skewed in this same way. Another important note is that this dataset wasn't originally intended for predicting credit score so there may not be strong relationships between the features and the output.

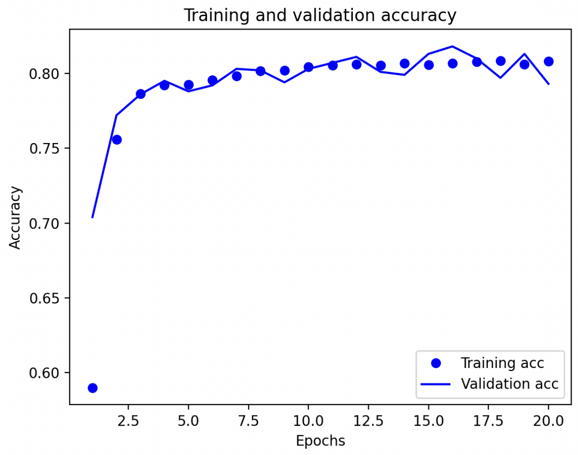




Binary Classification

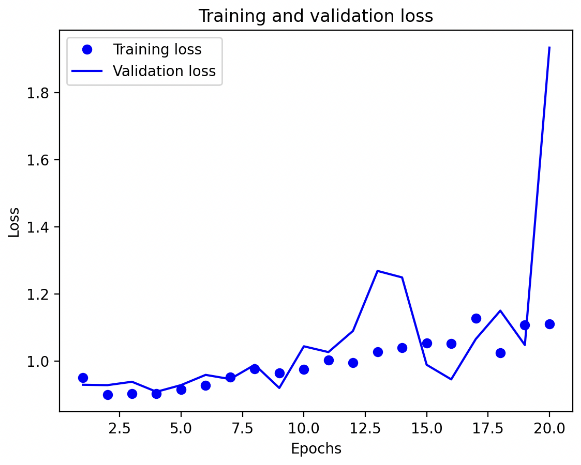
For the final binary classification model there is a loss of 0.426 and an accuracy of 82%. I would say that this is a fairly successful model. Considering that after one epoch the model is only 70% accurate compared to a final 82% accuracy after only 7 epochs, it is performing well. However, a likely reason that this model isn't able to perform much above this number is due to the class imbalance. In the dataset, there are 8,000 observations where a customer has exited and only 2,000 where they have not. This means that the model has a lot more 1's to train on and it is likely that it is more successful in predicting positives (has exited) than negatives. In the future, upsampling for negatives or downsampling of positives could be helpful in making the predictions more accurate. Another thing to take into consideration is that as discussed in the regression section above, many of the predictor variables are also highly skewed and the model probably has a difficult time finding patterns amongst data that has some variation. However, overall this model is doing pretty well. We can see by the two graphs below that the validation and training data are improving together and neither is oscillating. There is a very steady decline in loss and an improvement in overall accuracy, meaning that this data pretty accurately predicts whether or not a customer will churn.

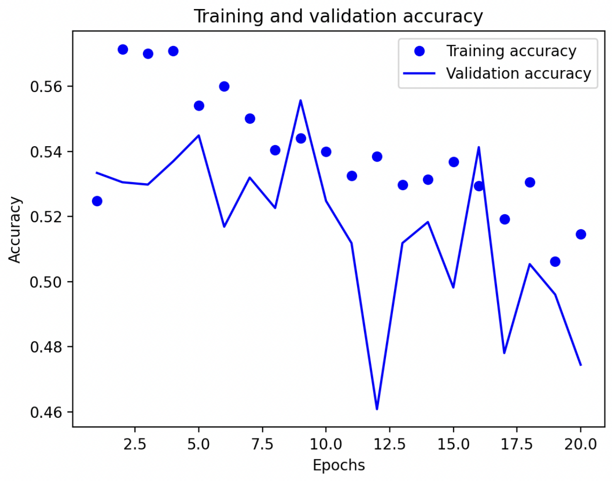




Multiclass Classification

Finally, the results of the multiclass classification shows a loss of 0.867 and an accuracy of 58.5%. While this is a pretty poor accuracy, it is important to take into account that the model is now predicting three classes instead of two so if it were to randomly assign classes, it would have just a 33.33% chance of success. So, by that reasoning the model is doing quite a bit better than just randomly guessing. However, it is very far from a perfect model. There are a few reasons for this. The first is that many of the variables are very highly skewed. Most of the variables are categorical and due to the fact that a majority of them were skewed, the model likely had a very hard time interpreting the data. Additionally, I manipulated the output variable so that it was in three fairly equal sized bins instead of 11 highly skewed ones. Still, this could lead to some issues and more buckets could have been better. Other things I did to improve the model were I tried to add more layers, increase and decrease the batch size, and take out features that were skewed, and my final model is the highest accuracy and lowest loss that was able to be achieved. A note about the optimizer is that rmsprop and adam actually had similar accuracy results after the 5 epochs. However, overall the model with rmsprop was significantly more stable and the loss was a lot lower, so I ultimately decided to choose rmsprop over adam. Another thing to take into consideration is that I did only select a portion of the observations from the original dataset. While they were a random selection, they could have been skewed. That being said, I believe that the biggest reason this model doesn't perform well is because it would have benefited from the other 300,000 observations.





Conclusion

Overall, it can be concluded that Feed Forward Neural networks are a highly effective algorithm used for predicting a wide range of outputs including regression, binary classification, and multiclass classification. However, due to the complexity of the algorithms, the relationships in the datasets are more and there are a lot more hyperparameters that can be tuned to enhance the performance of the model. Moreover, for nearly all of the models, it was difficult to find the optimal solutions due to data that was skewed and unevenly balanced classes. To improve upon this, in the future upsampling and downsampling of classes and features could enhance the performance of the models. Bigger datasets with more samples could also be beneficial so that the complexity of the datasets match the complexity of the model.