Neural Network to Predict Patient Length of Stay Report

Neev Bhandari

1. **Problem and Data**

**Dataset Description**

The dataset used comes from the MIMIC-III project, which includes de-identified health-related data from over 40,000 ICU patients. For this project, we are provided with two datasets, mimic\_train\_X and mimic\_train\_y, for training, and mimic\_test\_X.csv and mimic\_test\_y.csv, for testing. The features describe the vitals and general characteristics of the patients upon ICU entry, and the target variable for the **regression**, is the Length of Stay (LOS), which indicates how long a patient stays in the ICU.

**Goal**

The objective is to predict the length of stay (LOS) for ICU patients based on the given features, crucial for resource allocation and hospital management.

**Data Preprocessing:**

* We could see that for these datasets we had no missing values for any of the variables, so there was no need of transformation in that
* But we could see that there were some outliers that, due to the quantity of the data in the dataset, we decided to handle these outliers by cutting them from the dataset and using the data that would give us the most to do a correct model.
* We also acknowledge that it was necessary to convert these categorical variables into numerical ones to fit the regression models and after that the neural networks.
* There was a merge of both train y and x to get a dataset that could have all the variables together.
* Scaling of the data was also needed for the models to be more accurate and have all the variables in the same parameters.

We are using the Mean Squared Error (MSE) which Indicates the variance of the errors, with a higher penalty for larger errors.

1. **Models**

***Model 1: Linear Regression Model***

**Learning Algorithm**

The linear model tested on both the training and testing data was constructed with numerical variables. For categorical variables, such as *Ethnicity* and *Insurance*, this involved transforming these categorical displays to numerical quantities using one-hot-encoding. Through the use of dummy variables, each unique category in one variable was assigned a numerical value, which allowed the linear model to be fitted to the data. All together, the model uses 15 parameters, one for each input variable and the intercept.

The learning algorithm for this linear model is based on the ordinary least squares (OLS) method, where it assumes a linear relationship between the target variable (LOS), and the inputs. This method involves using the loss function of the mean squared error (MSE). This OLS method aims to minimise the MSE between the actual values in the data, and the predicted values derived from the model.

**Evaluation**

Because the data was not standardised, and the numerical quantities from the categorical variables were also not standardised with the rest of the data, the values MSE were not relative to the data, and are given in absolute values instead.

In-sample error (MSE): 24.66

Out-of-sample error (MSE): 96.19

The in-sample error refers to the mean of the squared error of the difference between the training data LOS value, and the training data predicted LOS value based on the fitted linear model. The out-of-sample error was calculated using the same method, but using the testing data instead of the training data.

The out-of-sample error is considerably larger than the in-sample error, over 3 times as much, suggesting that the linear model fitted on the training data has potentially overfitted. This means that the model has not only learnt about the training data patterns, but also extra noise in the training data, which may not generalise well with the testing data. To reduce this overfitting problem, techniques such as regularisation and cross-validation may need to be used. Additionally, applying non-linear transformations and other fitting method types such as neural networks can help to reduce the out-of-sample error.

***Model 2: Non-Linear Transformations Model***

**Learning Algorithm**

This model applied non-linear transformations to some numerical variables in the data. Non-linear distributions in variables would allow more error in linear models, as the model assumes linearity to determine the final model. Non-linear transformations would give the model more flexibility and ability to fit more complex models to better represent the data and predictions.

The distributions of each numerical variable are plotted against the LOS to observe any patterns in their distributions. For the *TempC\_Mean* and *Sp02\_Mean* variables, the plot resembles a negatively skewed distribution, where there are a larger amount of data points towards the larger mean values. An exponential non-linear transformation would be suitable to address this non-linearity in the distribution. For the *Glucose\_Mean* variable, the plot resembles a positively skewed distribution, where there are a larger amount of data points towards the smaller mean values. A logarithmic non-linear transformation would be suitable to address this non-linearity.

Applying these non-linear transformations to these numerical variables, the new model is fitted with the same number of parameters as in the linear model, 15. This model also uses the same OLS method as in the linear model, with the same loss function of the MSE as well. This OLS method aims to minimise the MSE between the actual values in the data, and the predicted values derived from the model.

**Evaluation**

In-sample error (MSE): 24.51

Out-of-sample error (MSE): 23.82

Similar values of the in-sample and out-of-sample error indicates that the model has suitable generalisation. This means that the model is performing equally well on the training and testing data, implying that the model is not overfitting, and has captured underlying patterns in the data without incorporating the noise in the training set. This suggests that regularisation and cross-validation is not necessary to improve the model. This model therefore provides better flexibility to perform on new data sets, which is essential for real-world applications.

***Model 3: Single Layer Neural Network***

**Learning Algorithm**

This model has 340,351 parameters and is flexible as it can recognise non-linear features but due to its simple nature it is not able to decipher more complex patterns. The single layer neural network model fitted to the training data iteratively calculates the non-linear function of all the signals, and evaluates the hypothesis for each individual observation to predict the LOS.

The learning algorithm for this single layer neural network utilises backpropagation with a loss function and activation function to evaluate the prediction for each observation. In neural networks, inputs are passed layer by layer, using weights and biases associated with each connection to compute activations for each neuron. For each neuron in the fully connected layer with 64 units, the weighted sum of the inputs are computed. An activation function, ReLu, is introduced to allow faster computation and optimisation. The second layer is the output layer, which outputs a single unit, as the model is predicting a single value of LOS for each observation.

Compiling this single layer neural network incorporates the MSE loss function. This loss function is used to compute the numerical error between the actual and predicted values. Using this MSE, the network computes the gradient of the loss function to each weight and bias in the network, through the process of backpropagation. With these gradients, the weights and biases in the network are updated using an optimiser. In this model, the ADAM optimiser is used, which utilises stochastic gradient descent (SGD) to update the weights through a learning rate, thus allowing the model to be more efficient in updating weights to converge faster and predict more accurate observations. By changing the number of epochs and batch size, the neural network can be better enhanced to achieve faster convergence and more accurate values.

**Optimisations**

**Standardisation**: The data was standardised to have a mean of 0 and a standard deviation of 1, which ensures that all features are on the same scale, this helps avoid the model being biased towards features with larger magnitudes by bringing all features to the same scale.

**Leaky ReLU Activation Function**: The Leaky ReLU activation function was used instead of the ReLU to avoid the "dying ReLU" problem, which can occur when neurons output zero and stop updating. Leaky ReLU allows a small gradient to pass even when the input is negative, promoting more effective learning.

**He Initialization:** We want to ensure the initialised weights are not too large or too small to avoid both the vanishing and exploding gradient problems. The He initialiser is designed for the ReLU and leaky ReLU activation functions so using it will ensure our model converges and finds good weights.

**Parameter Regularisation**: The complexity of the model is limited by penalising large weights by adding a term to the loss function that discourages complex models. Less complexity means the model will generalise better, which helps reduce the loss of the testing data. To get the regularisation parameter that worked best for the model we used cross validation and found that 0.01 works best.

**Early Stopping**: As we train or our training data, there will come a point where the model has learned all it can about the target function and will learn the noise in the training data. To avoid this overfitting, we use 20% of the training data as a validation set and treat the fitting on this validation set as Eout. When the Eout has increased for 5 consecutive batches, the training is stopped and the model that minimised Eout is recovered and saved.

**Evaluation**

In-sample error (MSE): 16

Out-of-sample error (MSE): 25

***Model 4: Deep Neural Network Model***

The deep neural network model has 887,809 parameters and works in a similar way to the single layer neural network, but instead has multiple hidden layers allowing for a lot more flexibility. The model was designed as a Deep Neural Network (DNN) with three hidden layers, each incorporating L2 regularisation, Leaky ReLU activations, and Dropout layers to prevent overfitting. This allows the model to predict more complex non-linear relationships between inputs, and produce more high-level features and computational abilities. As it is a DNN, we have a high flexibility with this model with high complex non-linear relationships. The Adam optimizer was chosen for its ability to dynamically adapt the learning rate during training, leading to faster convergence. This DNN has a large number of parameters due to the increased number of layers and neurons. For instance: 17281, with 17 for the input layer

First Hidden Layer

* 128 neurons with L2 regularisation (regulariser\_l2(0.1)) to reduce overfitting by penalising large weights.
* He normal initialization to initialise weights effectively for layers with ReLU activations.
* Leaky ReLU activation with an alpha of 0.1 to prevent "dying ReLU" problems, ensuring that negative input values do not get ignored.
* Dropout rate of 0.2 to randomly drop 20% of the neurons during training, preventing overfitting.

Second Hidden Layer:

* 128 units with L2 regularisation and He normal initialization.
* Leaky ReLU activation with alpha = 0.1.
* Dropout rate of 0.5 to randomly drop 50% of neurons during training, further helping with regularisation.

**Optimisation**

In addition to all the optimisation done on the simple neural network, the deep neural network had some additional optimisation.

**Dropout:** Dropout randomly deactivates a subset of neurons during training to prevent overfitting, especially for more complex neural networks. This is because no neuron has seen all the data which means it is a lot more difficult to overfit to the training data. By fitting a larger network, we can ensure that we get complex patterns while avoiding overfitting.

**Parameter Regularisation:** Using cross validation, a higher regularisation parameter of 0.1 was more suitable for this more complex model. This makes sense due to the increased complexity of this neural network architecture.

**Evaluation**

After training, the model was evaluated on the test set using the Mean Squared Error (MSE) as the performance metric. The model made predictions on the test data and calculated the MSE between the actual and predicted lengths of stay.

**Evaluation**

In-sample error (MSE): 18

Out-of-sample error (MSE): 23

1. **Results**

**Regularisation Comparison Across Models**

**Linear and Polynomial Models:** No regularisation was used

**Neural Networks (Simple and Deep):** Regularisation was essential for the neural networks. Both L2 regularisation and dropout were applied to reduce overfitting.

**Training Speed Comparison**

**Fastest**: Linear and polynomial regression

**Moderate**: Simple neural network (~15s)

**Slowest**: Deep neural network, which requires more time due to the large number of parameters and multiple layers.(~50s)

**MSE Comparison between Models**

|  |  |  |  |
| --- | --- | --- | --- |
| **MODEL** | **IN-SAMPLE MSE** | **OUT-S MSE** | **OVERFITTING** |
| **Linear Regression** | 24.66 | 96.19 | YES |
| **Non-Linear Model** | 24.51 | 23.82 | NO |
| **Single Layer NN** | 16 | 24-26 | NO |
| **Deep Neural Network** | 18 | 22-23 | NO |

The Linear Regression model sufferers form significant overfitting, with a much larger out of sample error compared to the in sample error. It has inability to capture complex and non linear relationships present in the data

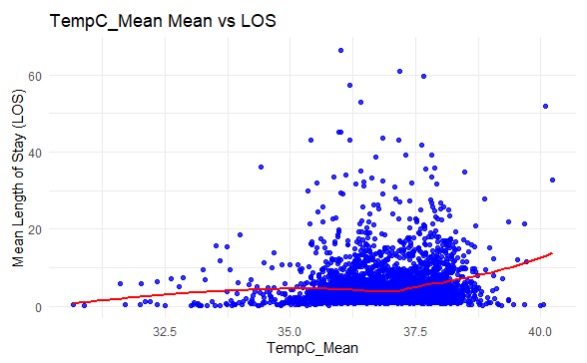
The Non-Linear Transformation model was much better than the linear regression one, by reducing the out of sample errors and with this one we can have a flexible approach to model non linear relationships between features and LOS.

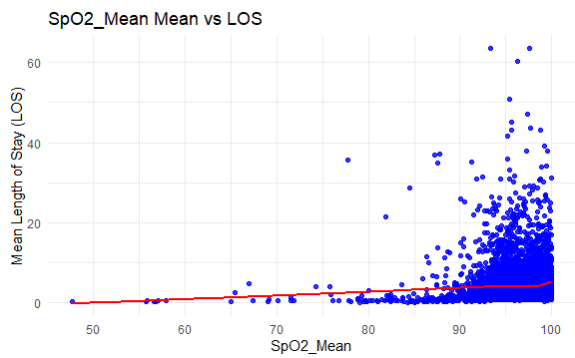
The Single Layer Neural Network performed similarly to the non linear model, with regularisation and early stopping preventing overfitting. It is a flexible model which captures non linearities, but doesn't add much benefit over simpler models and just makes a more complex model, with less interpretability. *Appendix B*

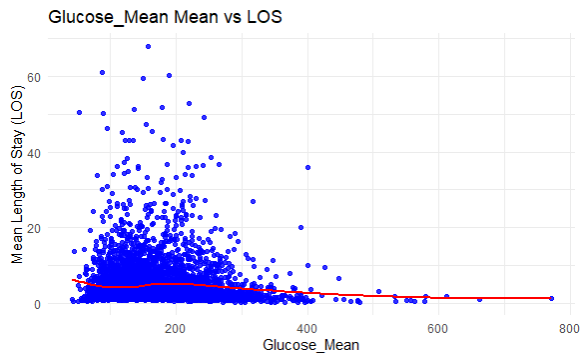
The Deep Neural Network was the best performing model achieving the lowest out of sample MSE. The combination of multiple hidden layers, regularisation, and dropout helped the model generalise well to the test set. We would just have to take into consideration the complexity and training time. *Appendix C*

1. **Appendix**

Appendix A



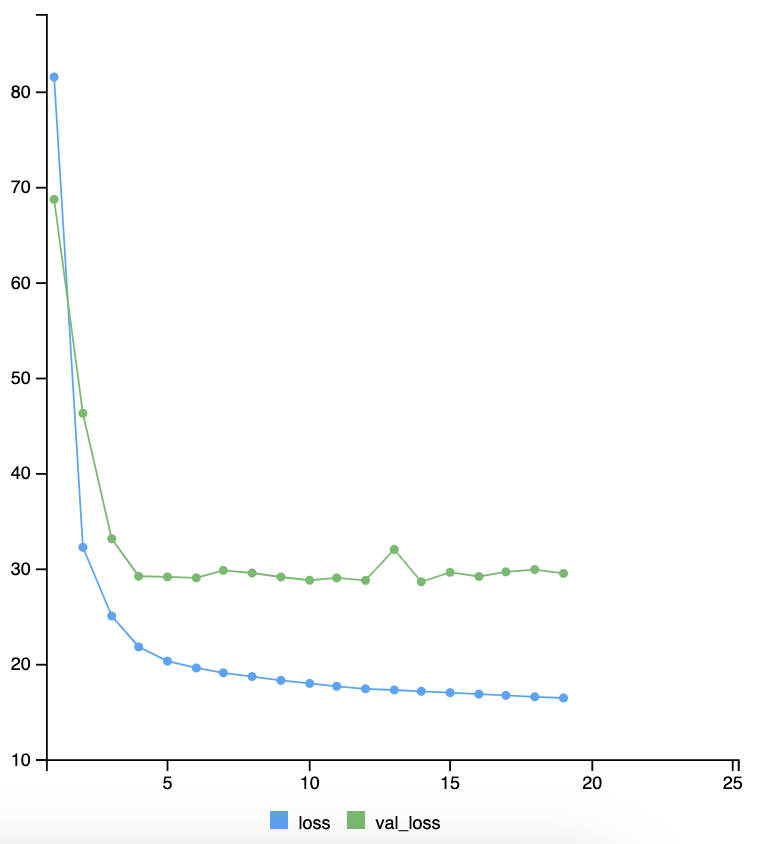
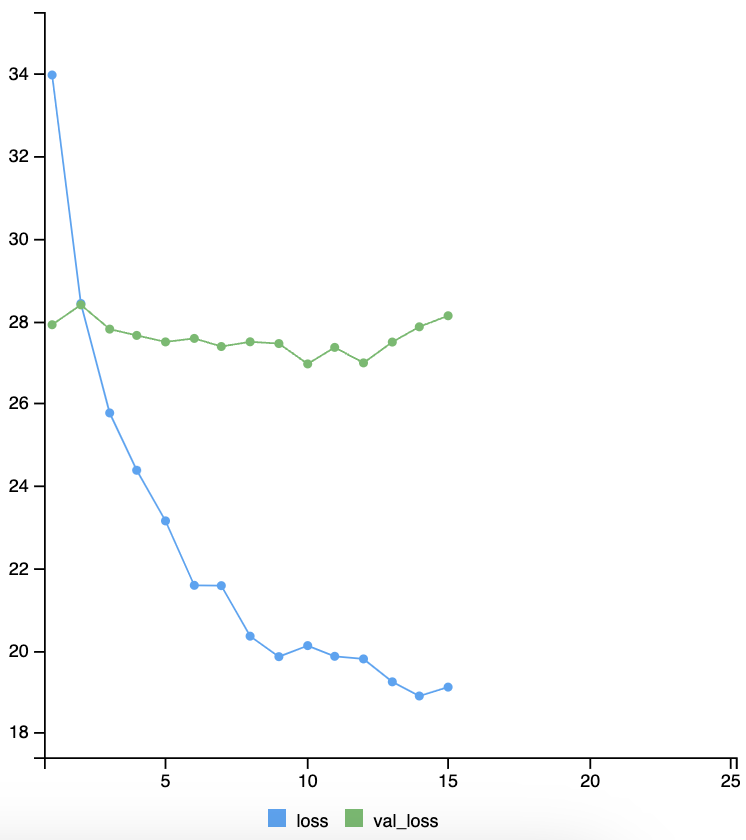




Appendix B

Graph of loss over epochs for simple neural network

Appendix C



Graph of loss over epochs for deep neural network