# **Neural Network Section Summary**

Our goal for the neural network section was to build a model that could effectively model the relationship between the supply of electricity in South Africa (by Eskom) and the demand for electricity by South African citizens, firms and export customers. The aim was to establish a foundational model that could serve as a foundation for a loadshedding estimator and ultimately develop a model capable of optimizing the allocation of resources for electricity production in order to mitigate loadshedding.

We collected data on electricity supply and demand from the Eskom data portal (Eskom Data Portal, 21 October 23) and additional data regarding load shedding levels from Eskom se Push (Eskom Se Push, 21 October 23). We had access to only 6 years of Eskom data (from 2019/04/01 to 2023/01/31). Each observation of this dataset is broken up into hourly measurements of the electricity landscape.

## **Feature engineering**

On inspection of the Eskom data, we found that many of the variables were highly correlated with each other or were not important to our modelling. We partitioned the dataset into three primary categories: Demand, Supply, and Grid Health (a breakdown of these subsets is shown in Figure 1 - [link to Eskom glossary for variable explanations]) In Grid Health we used variables that indicated how well the Electricity Grid was able to meet demand with their supply. These variables included the amount of demand reduced by loadshedding (MLR), and other demand reducing variables.

Within the Supply dataset, we considered the quantity of electricity generated by the primary resources utilized for electricity production in South Africa, including Renewable Energy, Gas, Water, and Diesel. Figure 2 and 3 shows the high correlation between variables in the supply dataset as well as variables in the demand dataset being correlated with each other. These correlations could negatively affect our model, as it might attribute excessive weight to the influence of these variables on the final Grid Health parameters. In order to reduce this effect, we aggregated these correlated variables: thus creating variables that acted as signals for their specific resources and demand type. No feature engineering was done on the Grid health variables as we believed each standalone variable was an important indication for how demand was being met.

## **Data Preparation for the modelling**

The first step was to split the data up into training, validation and test sets (using a 80%/10%/10% split). Then we normalised each Set of data using a standard scalar. Each set was scaled independently, preventing any information leakage between the validation, training and test datasets.

Next we grouped the observations into weekly windows, and batching these into a batch of 16 weeks (ie 4 months). Thus our model will learn the short term relationship within consecutive weeks as well as learn long term relationships within 4 month time periods. We used Keras TimeSeriesGenerator to do this. Where the model will take in a week’s worth of data (168 hours worth) and look to predict the next hour’s Grid health variables.

## **Modelling**

To capture both short-term and long-term patterns in our dataset, we used Long Short-Term Memory (LSTM) networks. The distinctive feature of LSTMs is their ability to maintain both short-term and long-term memory through the use of forget, input, and output gates. This ensures that the model captures immediate data dependencies while also updating its long-term memory during each training iteration—something that Vanilla Recurrent Neural Networks (RNNs) lack.

The initial model used the Mean Absolute Error as the loss function. We quickly discovered that the Huber Loss function was better suited to our project, shown in figure 5 - validation test error is significantly better than the training error of the original loss function. The validation error is also more stable (indicating less overfitting).

The Huber Loss function is particularly well-suited to our application due to its robustness to outliers. Huber loss combines the best characteristics of both Mean Squared Error (MSE) and MAE. It behaves like MSE with small errors and like MAE for large errors. This makes it ideal for scenarios where our dataset might contain outliers. The hyperparameter delta is used to determine the point at which the Huber loss stops acting like MAE and moves to MSE. During hyperparameter training we found that a value of 0.3 was significantly better than the default 1.0 and 0.5 (Figure 6) - validation error is lower than other delta values (1 and 0.5) and is more stable.

## **Architecture**

On analysis we found that the optimal architecture of the model included a 3 layer deep neural network, that included 2 Bidirectional LSTM layers, 1 standard LSTM layer and a dense output layer which generates the estimated vector for the four Grid Health variables.

A bidirectional LSTM allows us to retrieve dependencies of a current time step with all the previous time steps as well as future time steps. Thus the model has a clear understanding of the full make up of the environment and is able to learn it. The training process is very similar to that of a standard LSTM however a single unit is connected to all the other units in the layer. This increases the complexity of the model - increasing the likelihood of overfitting. This overfitting is seen in Figure 7, where the 2 Bidirectional Layers clearly have a reducing validation error compared to the base bidirectional model (3 LSTM layers) but the validation loss is still oscillating between 0.33 and 0.28. Another telltale sign of overfitting is the substantial decrease in training loss compared to validation error, with the former being significantly smaller than the latter.

Regularisation techniques were used to reduce overfitting. We introduced dropout layers after each LSTM layer - 20% of the nodes in each layer are randomly dropped (Figure 8 shows that 20% overfits less than 40% drop out - Model 3 vs Model 2). This means that the model’s variance increases but reduces the chance of the model memorising the training data.

In order to reduce overfitting more, we altered the weight decay of the Huber Loss function. This reduced the influence of each particular weight in the forward pass, we found that the optimal level for this was 0.001. However this is not seen as lowering overfitting, Figure 7: Model 3 vs Mode 1.

We thus decided that our final model is shown in Figure 9, with no altered weight decay nor an altered learning rate.

When running our model on the test dataset, we get a Huber loss of 0.32. Which in the case of regression estimate is relatively good and a Mean Absolute error of 1.22.

# **Conclusion**

Our model was able to predict Grid Health values based on a week's worth of supply and demand dynamics. However, we were constrained by the absence of Load Shedding stage data that matched with the time period of our Eskom data. Thus, we couldn't map these predictions to specific load shedding stages. To address this limitation and create a more accurate model, one potential approach is to explore data augmentation techniques to generate additional data points.

Furthermore, there is an exciting avenue for future work using Reinforcement Learning. This approach could be used to optimize the allocation of resources within the electricity grid, potentially leading to more efficient and effective management of the grid and mitigating loadshedding.

Figure 1

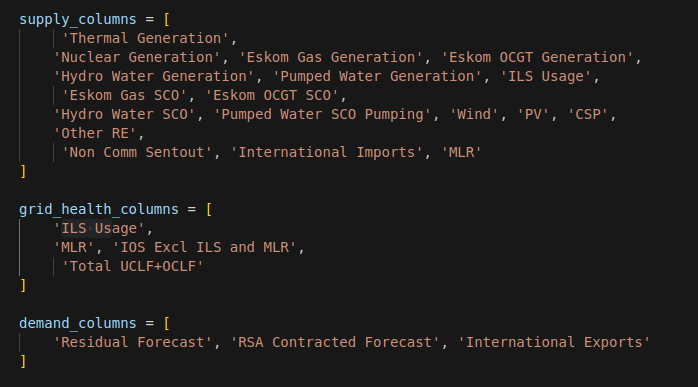


Figure 2

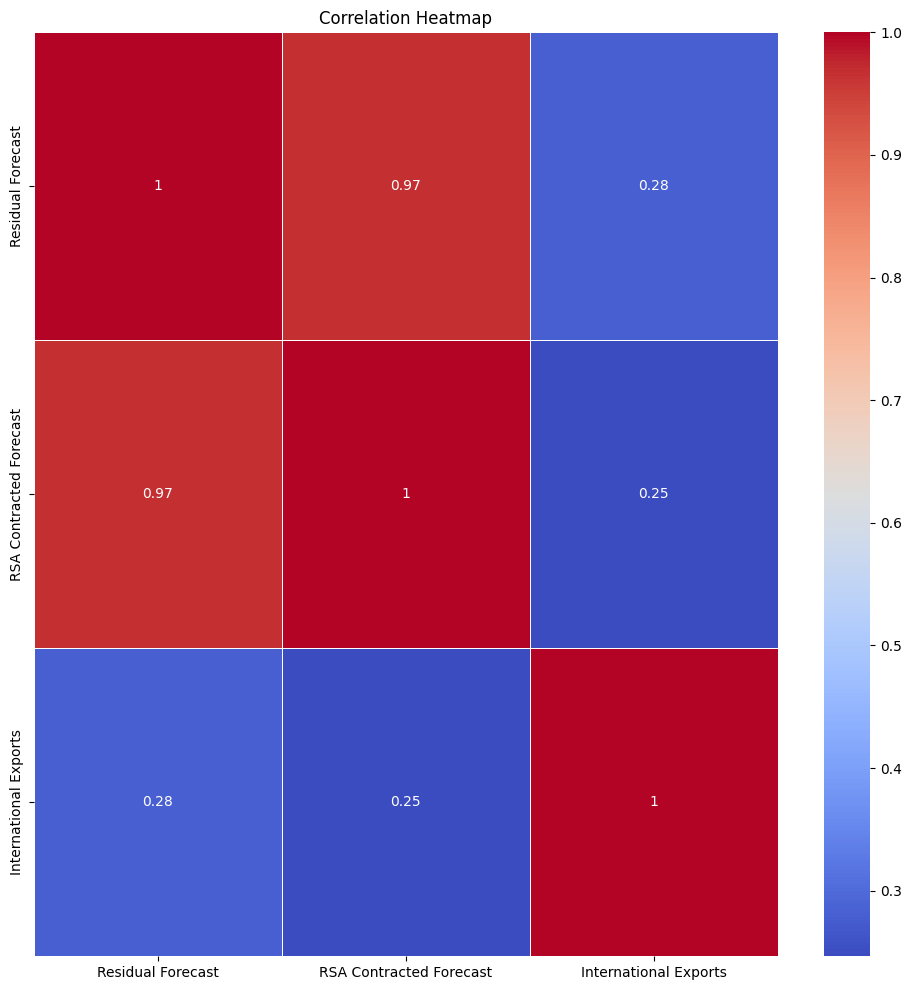


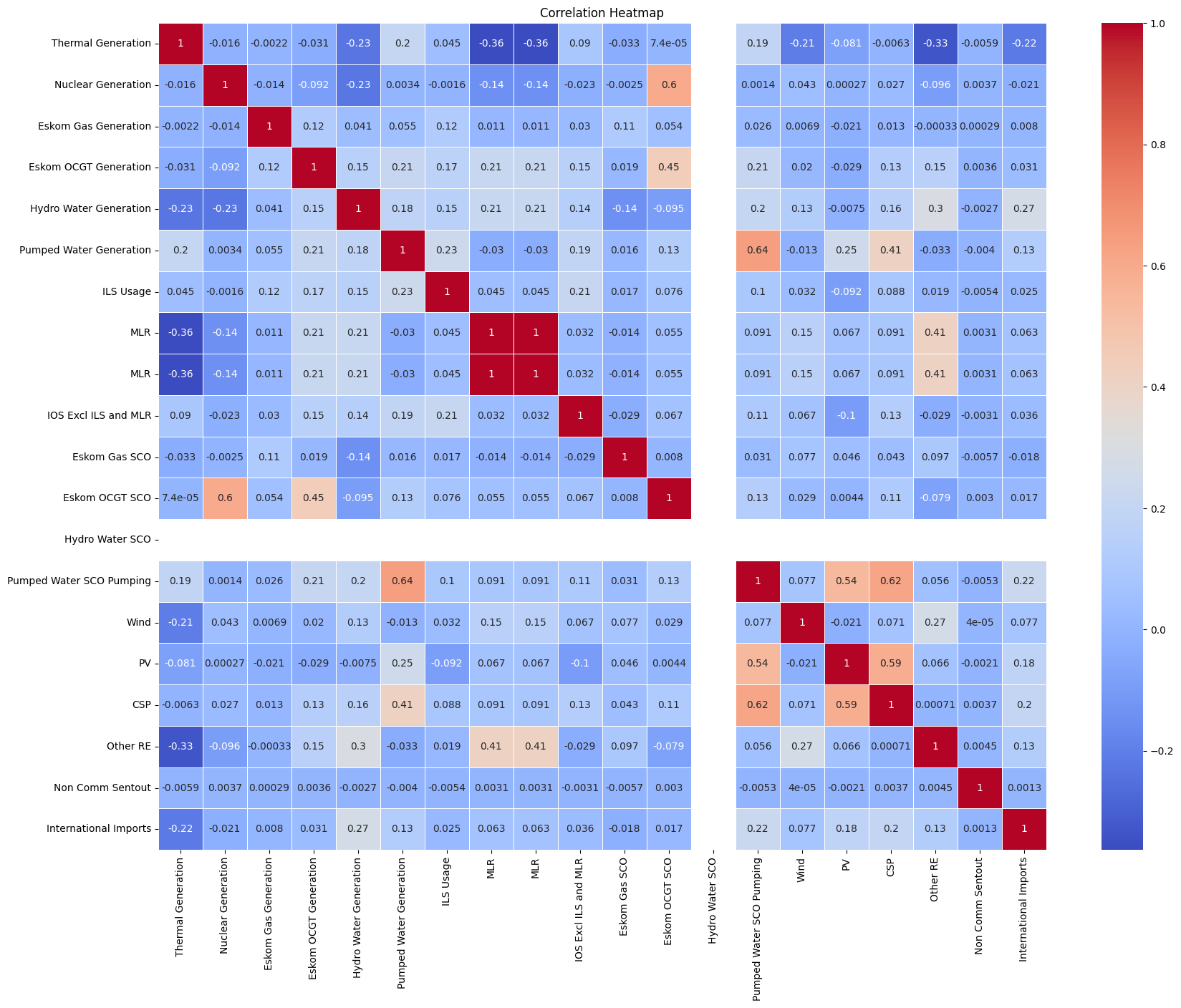
Figure 3

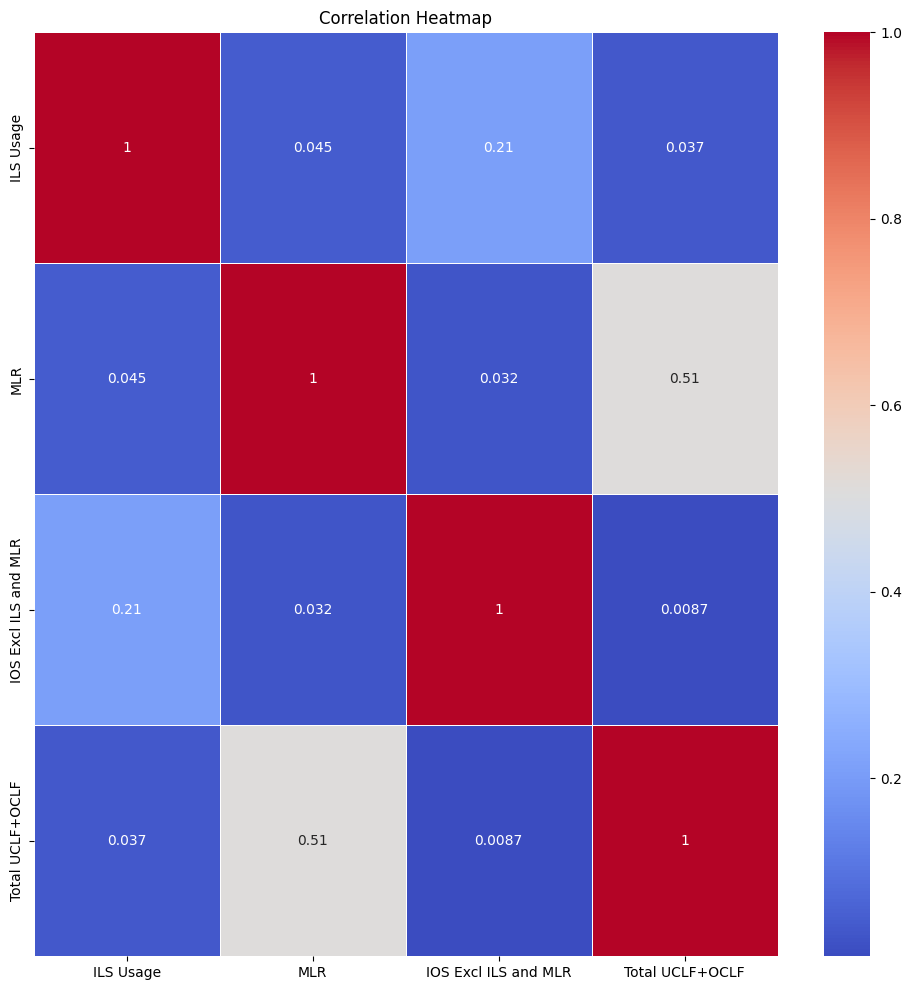
Figure 4

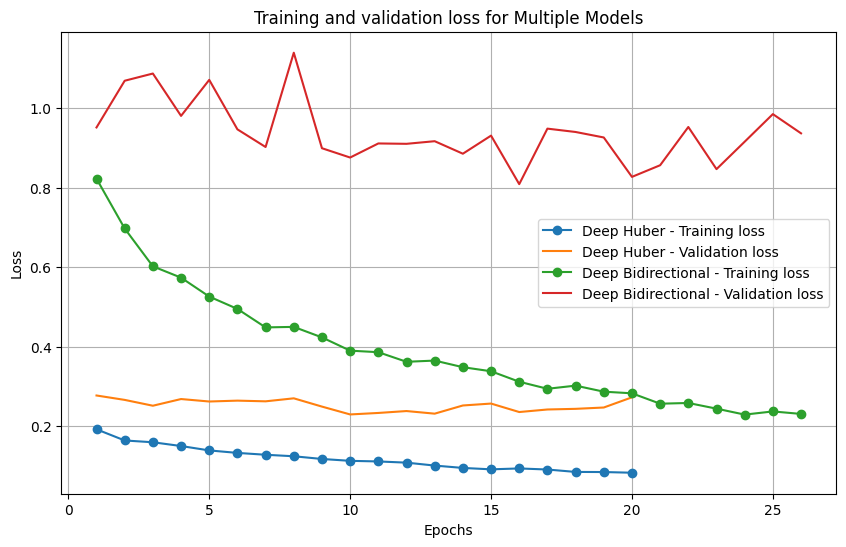
Figure 5

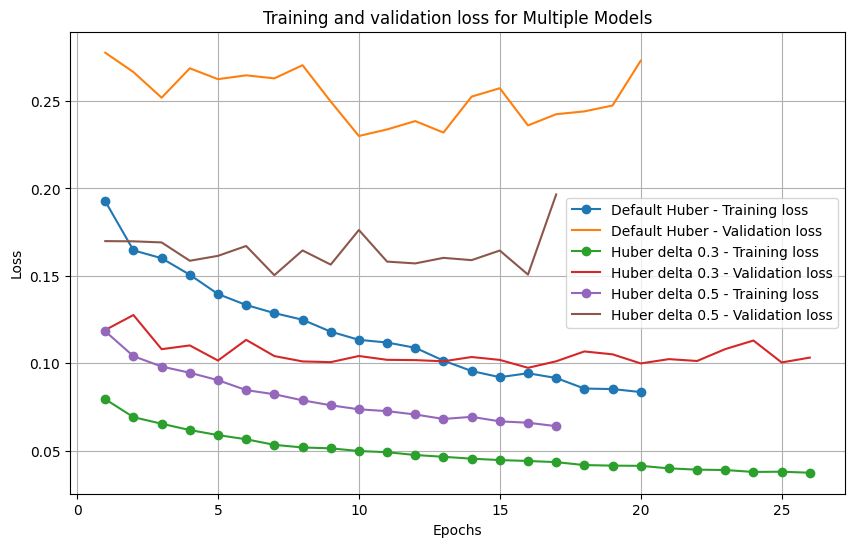
Figure 6

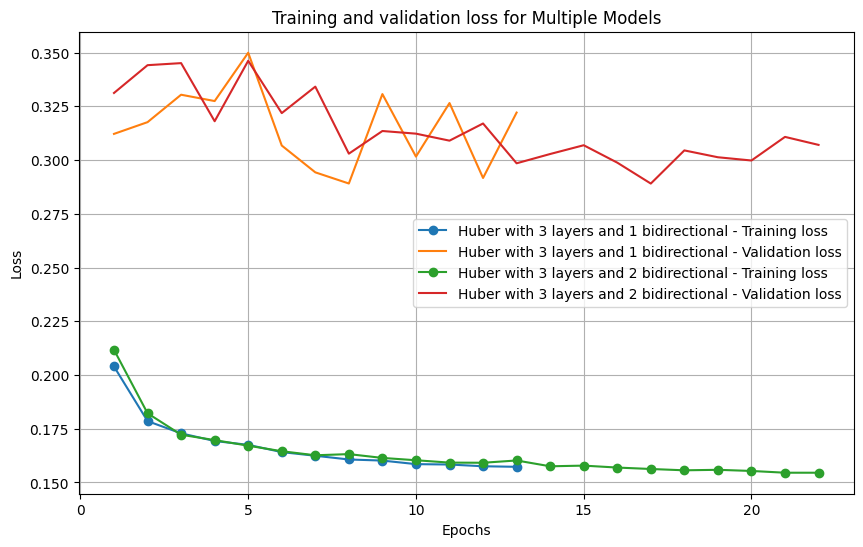
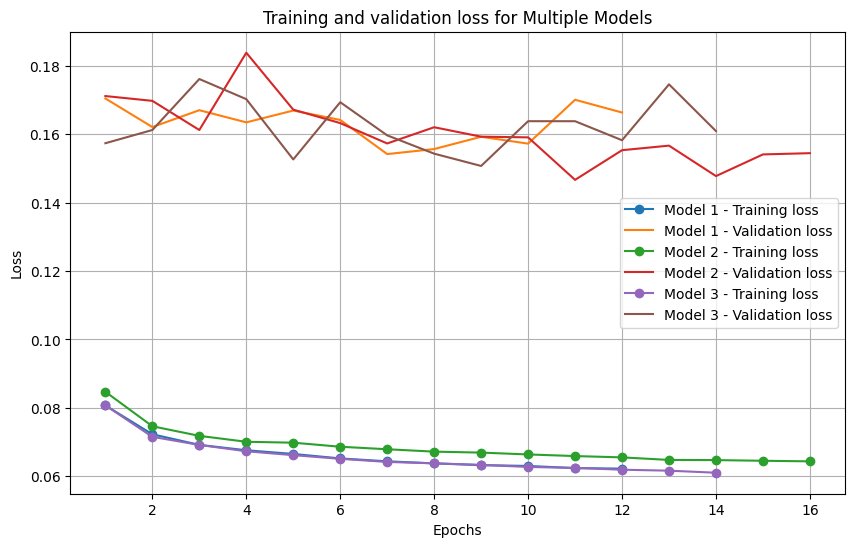
Figure 7

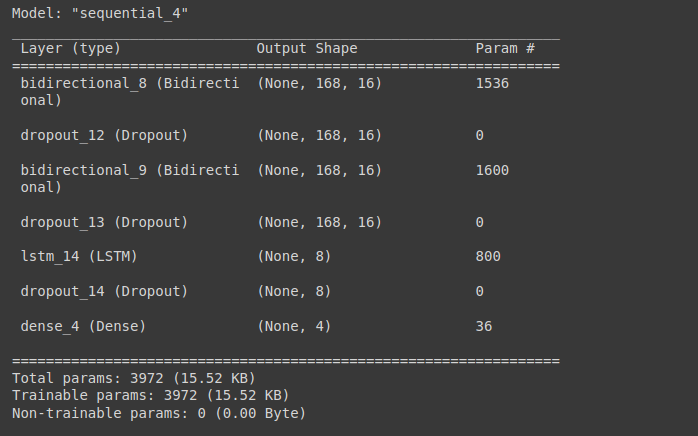
Figure 8

Model 1: Dropout = 0.2 and weight decay = 0.001

Model 2: Dropout = 0.4

Model 3: Dropout = 0.2 and no weight decay

Figure 9



## Bibliography of Data

Eskom Data Portal, <https://www.eskom.co.za/dataportal/>

Eskom se Push, <https://esp.info/>