Data Mining and Machine Learning

COMP809

Assignment Two Part B

Submitted by: Samuel Meads, Leon Lee

Student ID: 20113456, 20125718

Group: 29

2024

Table of Contents

List of Figures

List of Tables

Introduction and Data Preprocessing

Air pollution is a critical environmental issue that poses serious threats to public health. Particulate matter (PM), especially particles smaller than $PM_{2.5}$, has been shown to have a strong correlation with adverse health effects, including cardiovascular disease. Accurately predicting the concentration of $PM_{2.5}$ is therefore essential for public health planning and intervention. This assignment aims to build robust prediction models for $PM_{2.5}$ and PM_{10} concentrations using advanced machine learning techniques, specifically multi-layer perceptron (MLP) and long short-term memory (LSTM) networks.

The dataset used for this assignment is sourced from the Environmental Auckland Council Data Portal, comprising hourly measurements from January 2019 to December 2023. The data includes $PM_{2.5}$/PM_{10} concentrations and various predictors such as air pollution indicators (SO_2, NO, NO_2), the Air Quality Index (AQI), and meteorological data (solar radiation, air temperature, relative humidity, wind direction, and wind speed) collected from Penrose Station (ID:7) and Takapuna Station (ID:23). To enhance the predictive accuracy, the dataset also includes lagged PM measurements (lag_1 and lag_2), representing the concentrations from the previous hour and two hours prior, respectively.

Ensuring the dataset's consistency and quality is crucial for building effective prediction models. The data pre-processing phase involves aligning the temporal resolution to hourly measurements, identifying and handling missing data, and detecting outliers.

Attribute-specific Information about Outliers and Missing Data

There are periodic gaps in data, most notably around the 10000th data entry, likely due to external factors with the station, such as power outages. Outliers might occur due to unique circumstances such as rare events or measurement/instrument errors. Both can result in skewed and biased data distributions and have a knock-on effect on the rest of the analysis.

Data Cleaning Approach

Based on the provided dataset, the data cleaning process involves several steps to ensure the quality and consistency of the data before proceeding with feature selection. All columns containing the substring 'value' (case insensitive) are renamed to 'Value'. This standardization ensures consistency across all dataframes, simplifying subsequent processing steps. The 'Start' and 'End' columns, if present, are converted to datetime format. This conversion is essential for accurate time-based analyses and ensures that any invalid date entries are handled appropriately. The 'Value' column is converted to a numeric data type. This step ensures that all measurements are in a numerical format, facilitating statistical and machine learning operations. Missing values in the 'Value' column are filled with the median of the column. Using the median provides robustness against outliers and prevents significant skewing of the data distribution.

- Temperature: Values above 40°C are removed, as they are considered unrealistic for the Auckland region.
- Humidity: Values above 100% are capped at 100%, reflecting the physical limitation of relative humidity.
- General Measurements: Negative values are removed, and extremely high values (above 500) are capped to prevent skewing the data

By implementing these steps, the cleaned dataset is now ready for feature selection.

Data Exploration and Feature Selection

Choose five attributes of your dataset that has the highest correlation with PM2.5 or PM10 concentration using Pearson Correlation or any other feature selection method of your choice with justification.
• Provide the correlation plot (or results of any other feature selection method of your choice) and elaborate on the rationale for your selection.
• Describe your chosen attributes and their influence on PM concentration.
• Provide graphical visualisation of variation of PM variation.
• Provide summary statistics of the PM concentration.
• Provide summary statistic of predictors of your choice that has the highest correlation in tabular format.

Top Five Attributes Based on Correlation:

1. **NO**: Correlation with PM2.5 = 0.276, Correlation with PM10 = 0.210

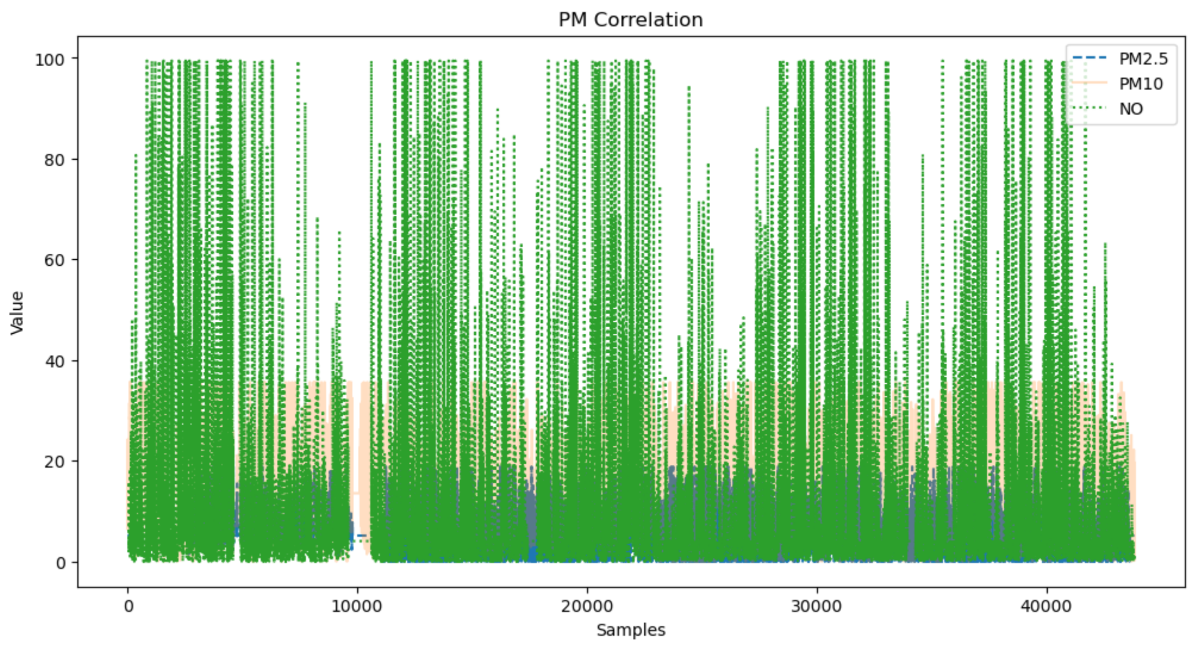


Figure 1: NO PM Correlation

2. **NO2**: Correlation with PM2.5 = 0.267, Correlation with PM10 = 0.105

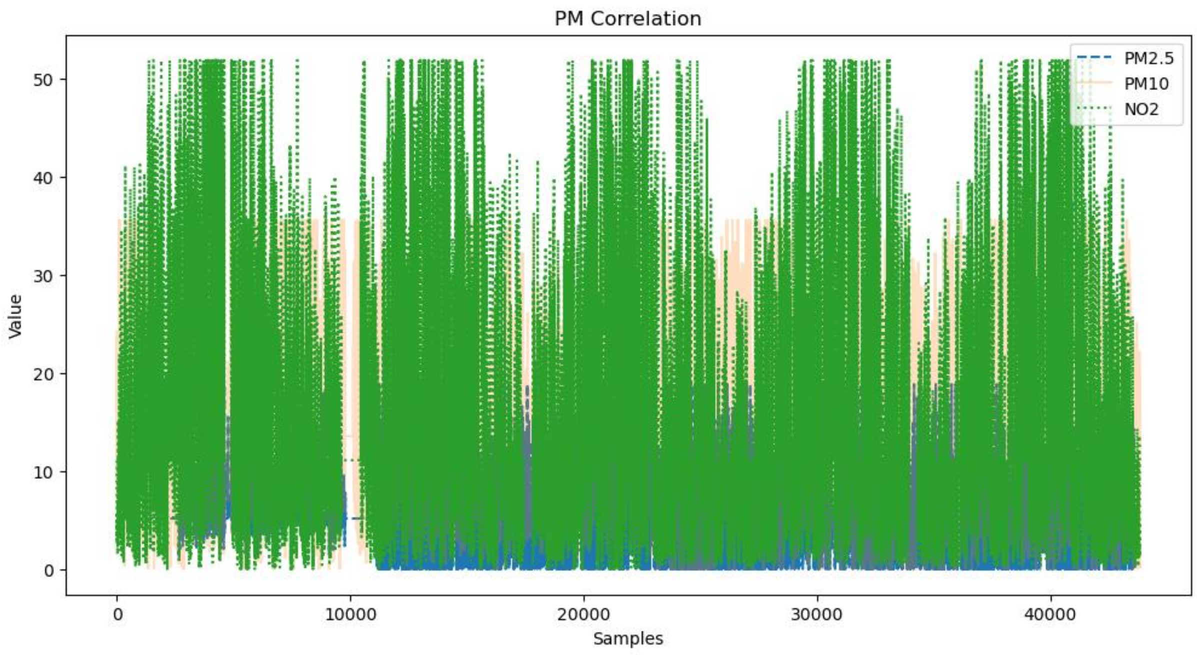


Figure 2: NO2 PM Correlation

3. **WindSpeed**: Correlation with PM2.5 = -0.06, Correlation with PM10 = 0.229

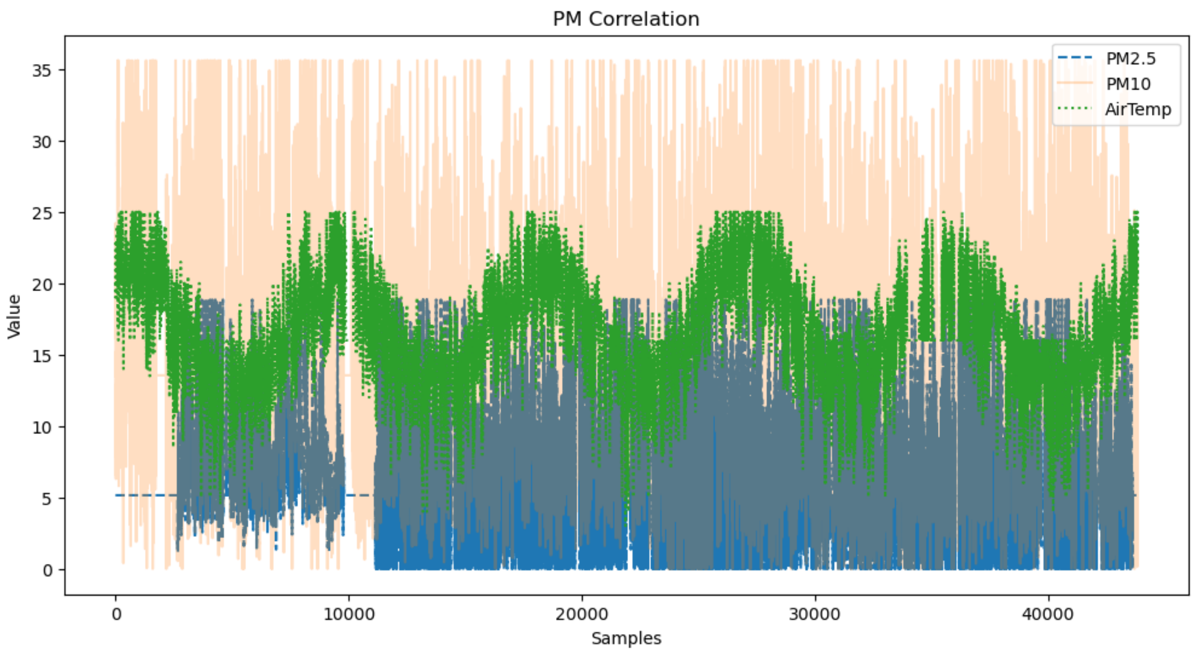


Figure 3: WindSpeed PM Correlation

4. **SO2**: Correlation with PM2.5 = 0.162, Correlation with PM10 = 0.129

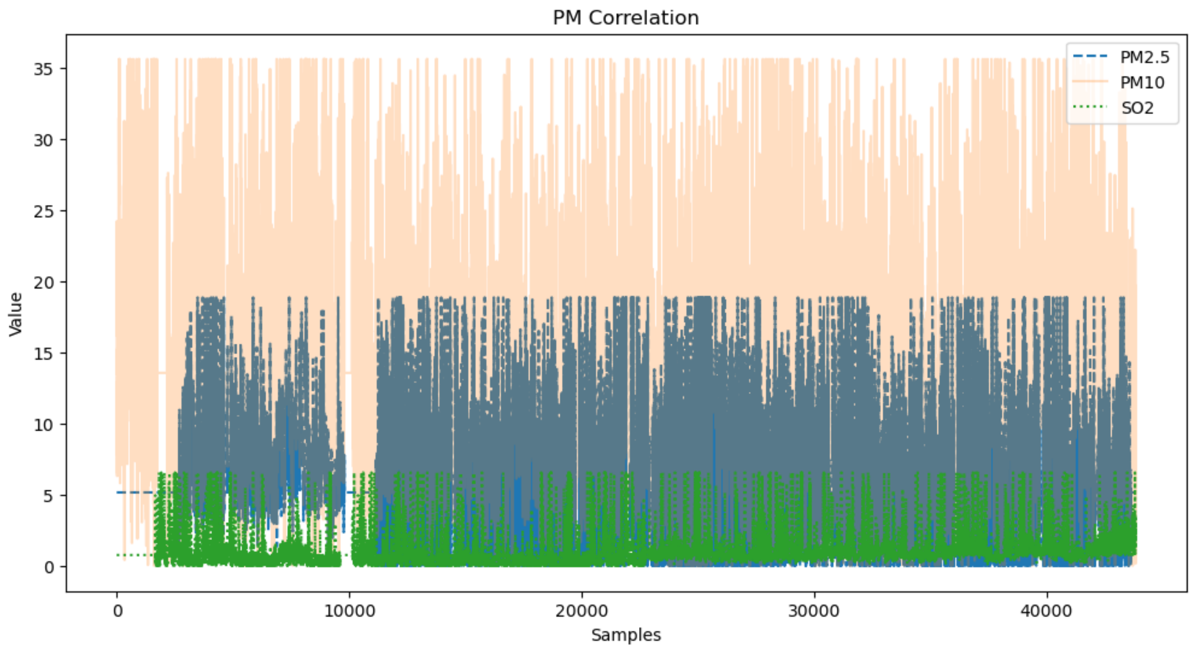


Figure 4: SO2 PM Correlation

5. **AirTemp**: Correlation with PM2.5 = -0.133, Correlation with PM10 = 0.102

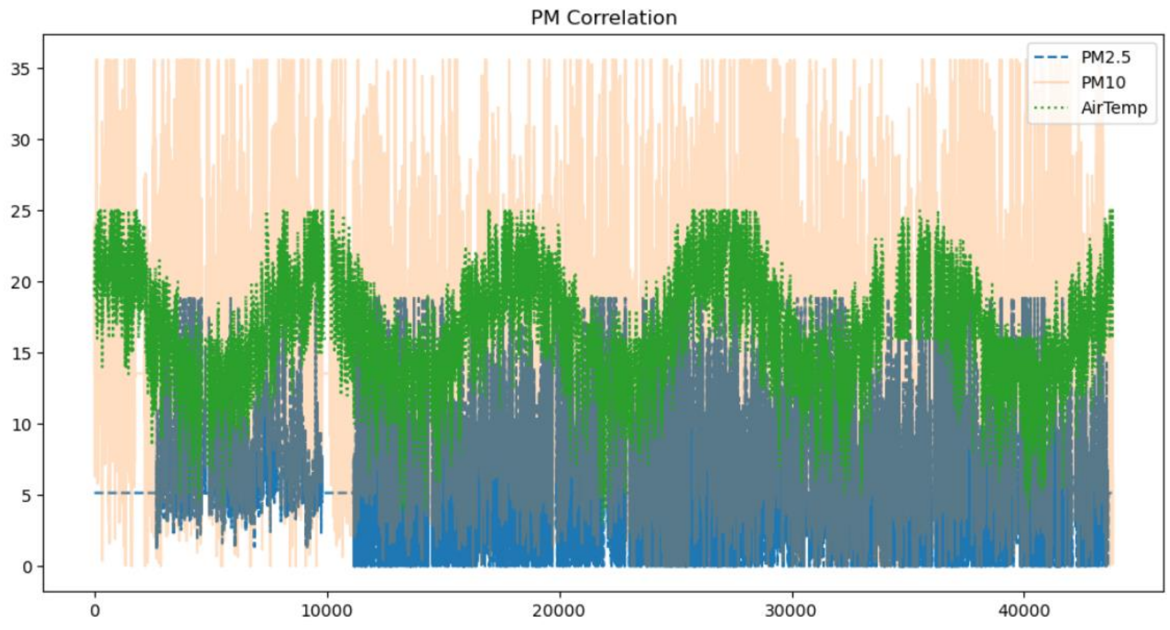


Figure 5: AirTemp PM Correlation

PM variation:

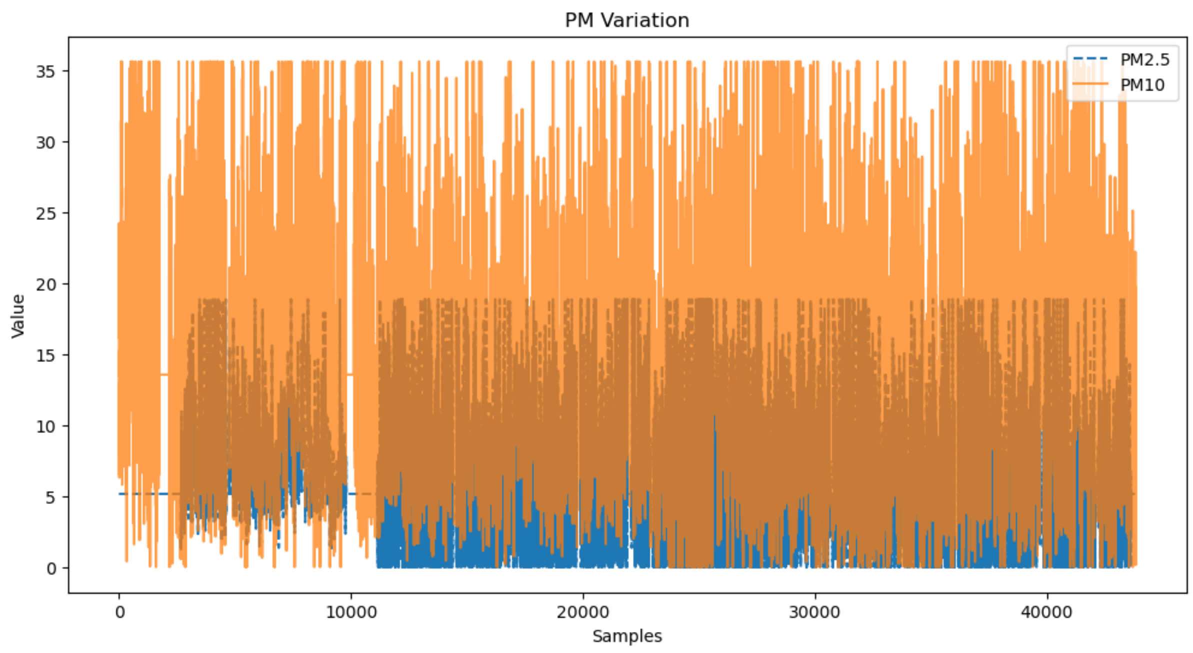


Figure 6: PM Variation

Justification and Rationale:

- **NO and NO2**: Both are precursors to particulate matter, especially in urban areas where traffic emissions contribute to secondary particulate formation via photochemical reactions in the atmosphere. Their positive correlation with PM levels indicates that higher nitrogen oxides concentrations can lead to higher concentrations of particulate matter.
- **WindSpeed**: Higher wind speed can lead to increased dispersion of particulate matter, potentially explaining the negative correlation with PM2.5 and the positive correlation with PM10. This suggests different dispersion behaviours or sources for these particulate sizes.
- **SO2**: Another precursor to particulate matter, especially in regions with high fossil fuel combustion. SO2 can undergo various chemical reactions in the atmosphere to form secondary particulates, contributing to the PM load.
- **AirTemp**: Temperature can influence the rates of chemical reactions in the atmosphere, affecting secondary particulate formation. The negative correlation with PM2.5 suggests that higher temperatures might enhance the chemical breakdown of some particulates or their precursors.

Table 1: Summary Statistics of Highest Correlation Predictors

Predictor	Count	Mean	Std Deviation	Min	25%	50%	75%	Max
PM2.5	43,824	5.606	3.496	0.00	3.60	5.15	7.00	18.85
PM10	43,824	14.344	6.924	0.00	9.45	13.55	18.25	35.60
NO	43,824	9.201	15.559	0.00	1.25	4.05	10.00	99.45
NO2	43,824	14.182	11.710	0.00	5.30	11.10	19.70	51.90
SO2	43,824	0.976	1.075	0.00	0.35	0.75	1.20	6.55
AirTemp	43,824	16.361	3.732	3.00	14.00	16.00	19.00	25.00
WindSpeed	43,824	2.788	1.479	0.15	1.65	2.60	3.75	6.60

Experimental Methods

Use 70% of the data for training and the rest for testing the MLP and LSTM models. Use a Workflow diagram to illustrate the process of predicting PM concentrations using the MLP and LSTM models:

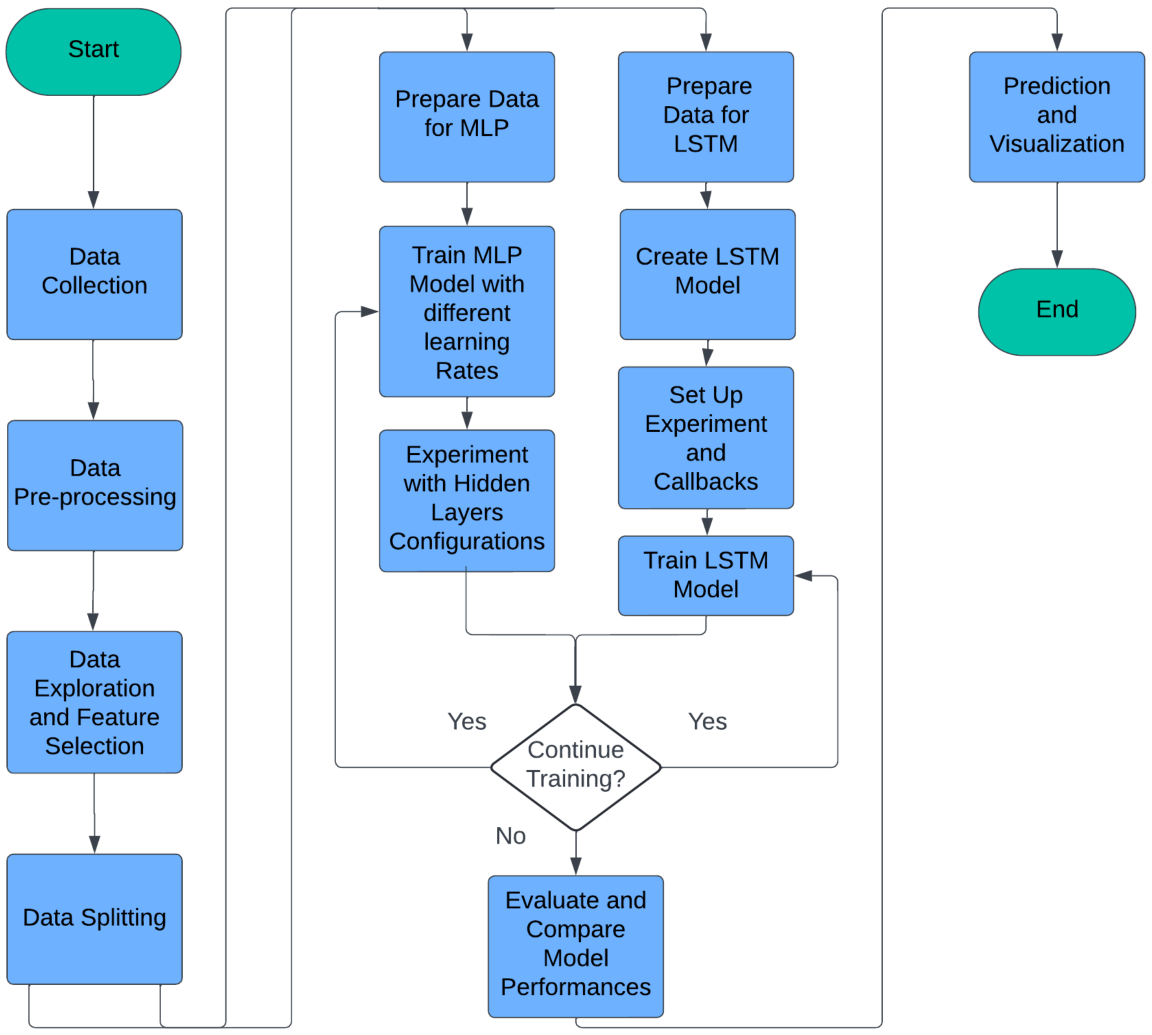

Start
Data Collection
Data Pre-processing
Data Exploration and Feature Selection
Data Splitting
Prepare Data for MLP
Train MLP Model with different learning Rates
Experiment with Hidden Layers Configurations
Prepare Data for LSTM
Create LSTM Model
Set Up Experiment and Callbacks
Train LSTM Model
Yes
Yes
Continue Training?
No
Evaluate and Compare Model Performances
Prediction and Visualization
End

Figure 7: MLP and LSTM Workflow Diagram

For both models, provide root mean square error (RMSE), Mean Absolute Error (MAE), and correlation coefficient (R^2) to quantify the prediction performance of each model.

Multilayer Perceptron

1) In your own words, describe multilayer perceptron (MLP). You may use one diagram in your explanation

A Multilayer Perceptron (MLP) is a type of artificial neural network (ANN) consisting of an input layer, one or more hidden layers, and an output layer. Each neuron in one layer connects to every neuron in the next layer through weighted connections. MLPs are powerful models used for tasks such as classification, regression, and time-series prediction.

Components of MLP:

- **Input Layer**: Neurons in this layer represent the features of the input data, with each neuron corresponding to one feature.
- **Hidden Layers**: These layers process inputs from the previous layer through weighted connections and apply activation functions to introduce non-linearity, enabling the network to learn complex patterns.
- **Output Layer**: Produces the final predictions of the network. For classification tasks, it typically uses a softmax activation function, for regression tasks, a linear activation function.

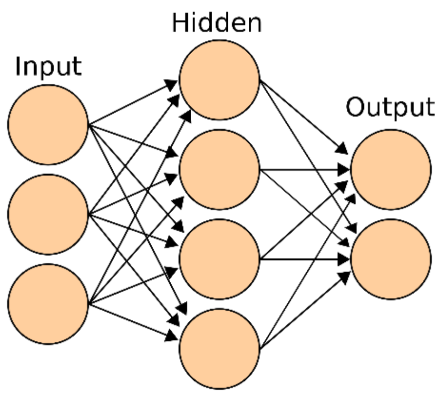
Working of MLP:

1. **Forward Propagation**: Input data is passed through the network layers, with each neuron computing a weighted sum of its inputs, adding a bias term, and applying an activation function to produce its output.
2. **Activation Functions:** Common functions include sigmoid, hyperbolic tangent (TanH), and Rectified Linear Unit (ReLU), introducing non-linearity to model complex relationships.
3. **Loss Function**: Measures the difference between the network's predictions and the actual target values. Cross-entropy loss is used for classification tasks, while mean squared error (MSE) is used for regression tasks.
4. **Backpropagation and Optimization**: Adjusts weights and biases to minimize the loss function using optimization algorithms like stochastic gradient descent (SGD) or Adam.

Major Parameters for MLPs:

- **Learning Rate**: Determines the size of the steps taken in the weight adjustment process. Larger steps mean faster learning but can reduce accuracy, while smaller steps allow for more precise adjustments but slow down the learning process.
- **Number of Epochs**: Refers to the number of times the training dataset is scanned during training. More epochs generally lead to a more accurate model.
- **Number of Hidden Neurons**: Influences the network's learning capability. A common heuristic is to set the number of hidden neurons as (attributes + classes) / 2.
- **Momentum**: Adds a term to current weight updates, helping to make the learning process smoother by reducing oscillations.

Below is a simple diagram of an MLP with one hidden layer:



Key Characteristics:
Fully Connected: Each neuron in a layer is connected to every neuron in the subsequent layer.
Feedforward: Information flows in one direction, from input to output, without looping back.
Non-linear Activation: Activation functions enable the network to capture complex patterns.

Figure 8: MLP Diagram

2) Use the *sklearn.neural_network.MLPRegressor* with default values for parameters and **a single hidden** layer with k= 25 neurons. Use default values for all parameters and experimentally determine the best learning rate that gives the highest performance on the testing dataset. Use this as a baseline for comparison in later parts of this question.

```
Learning Rate: 0.001 => RMSE: 0.17663235927479723, R^2: 0.1069685345964616
Learning Rate: 0.01 => RMSE: 0.17581657486708827, R^2: 0.11519849647205183
Learning Rate: 0.05 => RMSE: 0.17705535097675174, R^2: 0.10268622523077664
Learning Rate: 0.1 => RMSE: 0.17755694014413545, R^2: 0.09759493148470177
Learning Rate: 0.2 => RMSE: 0.18790991867737494, R^2: -0.010707831308497617
Best Learning Rate: 0.01 with RMSE: 0.17581657486708827
```

In this experiment, we tested multiple learning rates for an MLP with a single hidden layer of 25 neurons. The best performance on the testing dataset was achieved with a learning rate of 0.01, which resulted in an RMSE of 0.17 and an R^2 of 0.098. This learning rate will serve as the baseline for comparison in later parts of the question.

3) Experiment with **two hidden layers** and experimentally determine the split of the number of neurons across each of the two layers that gives the highest accuracy. In part 2, we had all k neurons in a single layer, in this part we will transfer neurons from the first hidden layer to the second iteratively in step size of 1. Thus, for example in the first iteration, the first hidden layer will have k-1 neurons whilst the second layer will have 1, in the second iteration k-2 neurons will be in the first layer with 2 in the second, and so on.

```
Best configuration: 16 neurons in the first layer and 9 in the second layer with MSE: 0.030636006490662
```

We experimented with different configurations of neurons split across two hidden layers. By iteratively transferring neurons from the first hidden layer to the second in steps of one, we found that the best configuration is 16 neurons in the first layer and 9 neurons in the second layer, resulting in an MSE of 0.031. This configuration achieved the highest accuracy in our experiments.

4) From the results in part 3 of this question, you will observe a variation in the obtained performance metrics with the split of neurons across the two layers. Give explanations for some possible reasons for this variation and which architecture gives the best performance

The variation in performance metrics observed in part 3 can be attributed to several factors:

1. **Model Capacity:** The number of neurons in each layer affects the model's capacity to learn and represent the underlying patterns in the data. Too few neurons may lead to underfitting, where the model cannot capture the complexity of the data. Conversely, too many neurons may cause overfitting, where the model learns noise in the training data and performs poorly on unseen data.
2. **Layer Depth:** Adding a second hidden layer allows the model to learn hierarchical representations of the data, potentially capturing more complex patterns. However, the distribution of neurons between the layers is crucial. A balanced distribution can help the model generalize better by leveraging the depth of the network.
3. **Training Dynamics:** Different configurations impact the training dynamics, including the convergence speed and stability. Properly balanced layers can facilitate more effective gradient flow during backpropagation, improving training efficiency and model performance.

The best architecture, identified as having 16 neurons in the first hidden layer and 9 in the second layer, strikes a balance between model capacity and complexity. This configuration provides sufficient representation power while maintaining the ability to generalize well on the testing data, resulting in the highest performance among the tested configurations.

Long Short-Term Memory

1) Describe LSTM architecture including the gates and state functions. How does LSTM differ from MLP? Discuss how does the number of neurons and batch size affect the performance of the network?

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) designed to model sequential data and capture long-term dependencies. LSTM networks are particularly effective at handling the vanishing gradient problem, which can hinder the performance of traditional RNNs. The key components of LSTM architecture are the memory cell, the forget gate, the input gate, and the output gate.

Components of LSTM Architecture

1. **Memory Cell**: The memory cell is the core of the LSTM network, maintaining information across time steps. It allows the network to store and retrieve information over long sequences.
2. **Forget Gate**: The forget gate determines which information should be discarded from the cell state. It takes the previous hidden state (h_{t-1}) and the current input (x_t) as inputs and outputs a number between 0 and 1 for each number in the cell state

   Forget Gate equation: $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
3. **Input Gate**: The input gate decides which new information should be added to the cell state.

   Input Gate equation: $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$

   Candidate values equation: $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
4. **Cell State Update**: The cell state is updated by combining the previous cell state (C_{t-1}) and the new candidate values $(\tilde{C}_t)$ modulated by the forget and input gates.

   Cell state equation: $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
5. **Output Gate**: The output gate determines the output of the current cell state. It controls which parts of the cell state should be output.

   Output gate equation: $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$

   Hidden state equation: $h_t = o_t * \tanh(C_t)$

Differences between LSTM and MLP:

LSTM networks are specialized RNNs designed to handle long-term dependencies in sequential data using a unique architecture involving memory cells and gating mechanisms. In contrast, MLPs are feedforward networks without recurrent connections, suitable for tasks with independent data points. The performance of both LSTM and MLP networks is influenced by the number of neurons and the batch size, requiring careful tuning to achieve optimal results.

Architecture:

- **LSTM**: Contains recurrent connections, allowing information to persist across time steps. It uses gates to control the flow of information.
- **MLP**: Feedforward neural network with no recurrent connections, processing each input independently.

Memory:

- **LSTM**: Capable of maintaining and updating a memory cell to capture long-term dependencies in sequential data.
- **MLP**: Lacks memory cells and recurrent connections, making it unsuitable for sequential data with long-term dependencies.

Applications:

- **LSTM**: Effective for tasks involving sequential data, such as time-series prediction, language modelling, and speech recognition.
- **MLP**: Suitable for tasks like classification, regression, and image recognition where data points are independent.

Impact of Number of Neurons and Batch Size on Performance

Number of Neurons:

- **Impact**: Increasing the number of neurons in LSTM or MLP can enhance the network's capacity to learn complex patterns, but it also increases the risk of overfitting and computational cost.
- **Optimal Selection**: It requires experimentation to balance between sufficient capacity to learn patterns and preventing overfitting. Cross-validation and hyperparameter tuning techniques are often used to find the optimal number of neurons.

Batch Size:

- **Impact**: The batch size determines the number of samples processed before updating the model's weights. Smaller batch sizes provide more frequent updates and can improve model convergence but might result in noisier gradient estimates. Larger batch sizes offer more stable gradient estimates but require more memory and might slow down the convergence process.
- **Optimal Selection**: Typically, batch sizes are chosen based on the available computational resources and the specific dataset characteristics. Common practices involve starting with a standard batch size (e.g., 32 or 64) and adjusting based on the model's performance and training stability.

Best Epoch

2) To create the LSTM Model and determine the optimal architecture, apply Adaptive Moment Estimation (ADAM) to train the networks. Identify an appropriate cost function to measure model performance based on training samples and the related prediction outputs. To find the best epoch, based on your cost function results, complete up to 30 runs keeping the learning rate and the number of batch sizes constant (e.g. at 0.01 and 4 respectively). Provide a line plot of the test and train cost function scores for each epoch. Report the summary statistics (Mean, Standard Deviation, Minimum and Maximum) of the cost function as well as the run time for each epoch. Choose the best epoch with justification.

Batch Size: 4, Neuron Count: 20
Mean Validation Loss: 0.031216138787567616
Std Dev of Loss: 0.0004973435069739052
Min/Max Loss: 0.030436329543590546/0.03260236978530884
Mean Time per Epoch: 15.887710944811504 seconds
Total Training Time: 476.6313283443451 seconds?
Best Epoch: 30 with Validation Loss: 0.030436329543590546

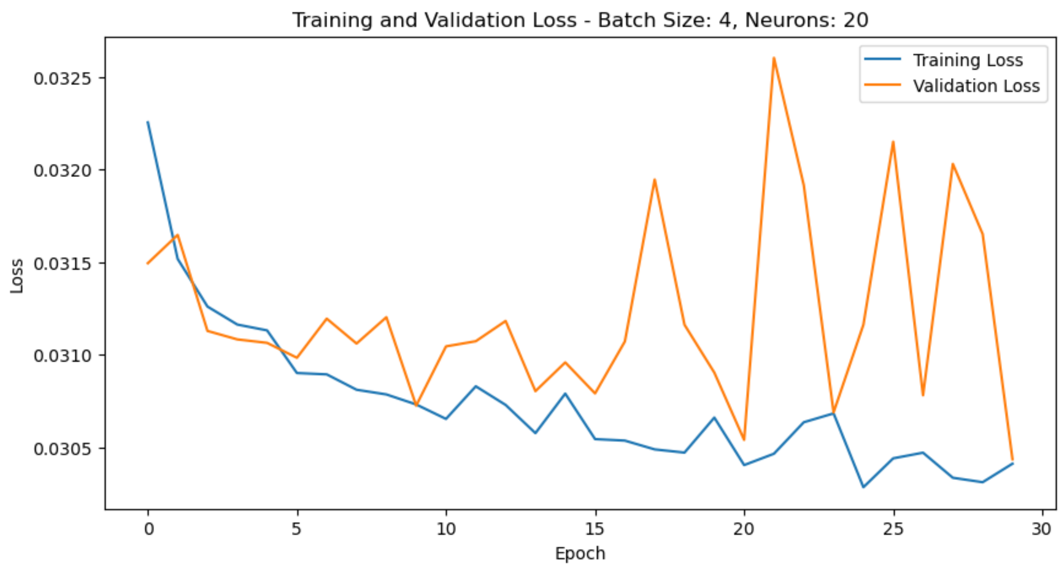


Figure 9: Loss per Epoch for LSTM Model with Batch Size 4 and 20 Neurons per Layer

We created an LSTM model with 20 neurons per layer and applied the ADAM optimizer to train the network with a learning rate of 0.01 and a batch size of 4. We used Mean Squared Error (MSE) as the cost function to measure model performance. The model was trained for 30 epochs, and we plotted the training and validation loss for each epoch. The summary statistics for the validation loss were as follows: mean validation loss of 0.0312, standard deviation of 0.000497, minimum loss of 0.0304, and maximum loss of 0.0326. The mean time per epoch was 15.89 seconds, with a total training time of 476.63 seconds. The best epoch, based on the lowest validation loss, was epoch 30 with a validation loss of 0.0304.

The selection of epoch 30 as the best epoch is justified by its performance in minimizing the validation loss, which is a critical metric for assessing how well the model generalizes to new data. Lower validation loss suggests better performance and reduced overfitting.

Cost Summary Statistics

3) Investigate the impact of differing the number of the batch size, complete 30 runs keeping the learning rate constant at 0.01 and use the best number of epochs obtained in previous step 2. Report the summary statistics (Mean, Standard Deviation, Minimum and Maximum) of the cost.

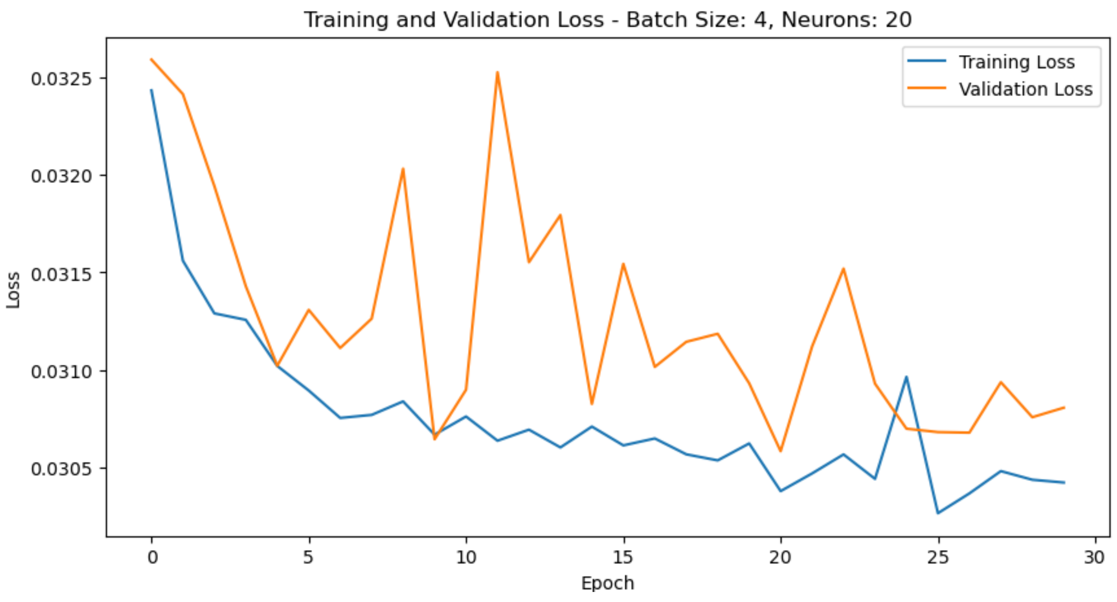


Figure 10: 2. Loss per Epoch for LSTM Model with Batch Size 4 and 20 Neurons per Layer

Batch Size: 4, Neuron Count: 20
Mean Validation Loss: 0.0313
Standard Deviation of Loss: 0.0006
Min/Max Validation Loss: 0.0306/0.0326
Mean Time per Epoch: 16.44 seconds
Total Training Time: 493.19 seconds

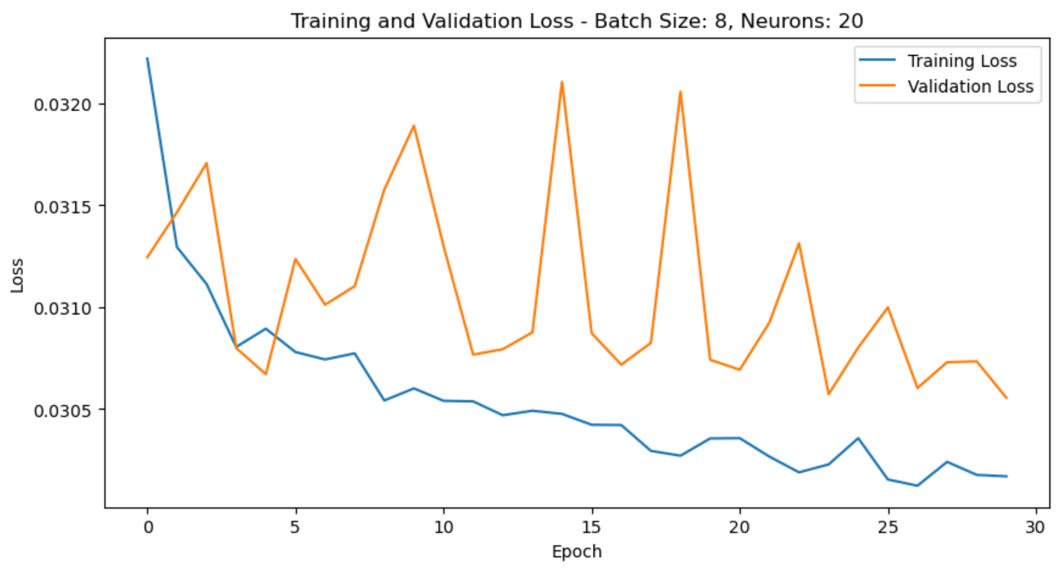


Figure 11: Loss per Epoch for LSTM Model with Batch Size 8 and 20 Neurons per Layer

Batch Size: 8, Neuron Count: 20
Mean Validation Loss: 0.0311
Standard Deviation of Loss: 0.0004
Min/Max Validation Loss: 0.0306/0.0321
Mean Time per Epoch: 10.74 seconds
Total Training Time: 322.16 seconds

The best batch size is 16 with the lowest mean validation loss of 0.0310.

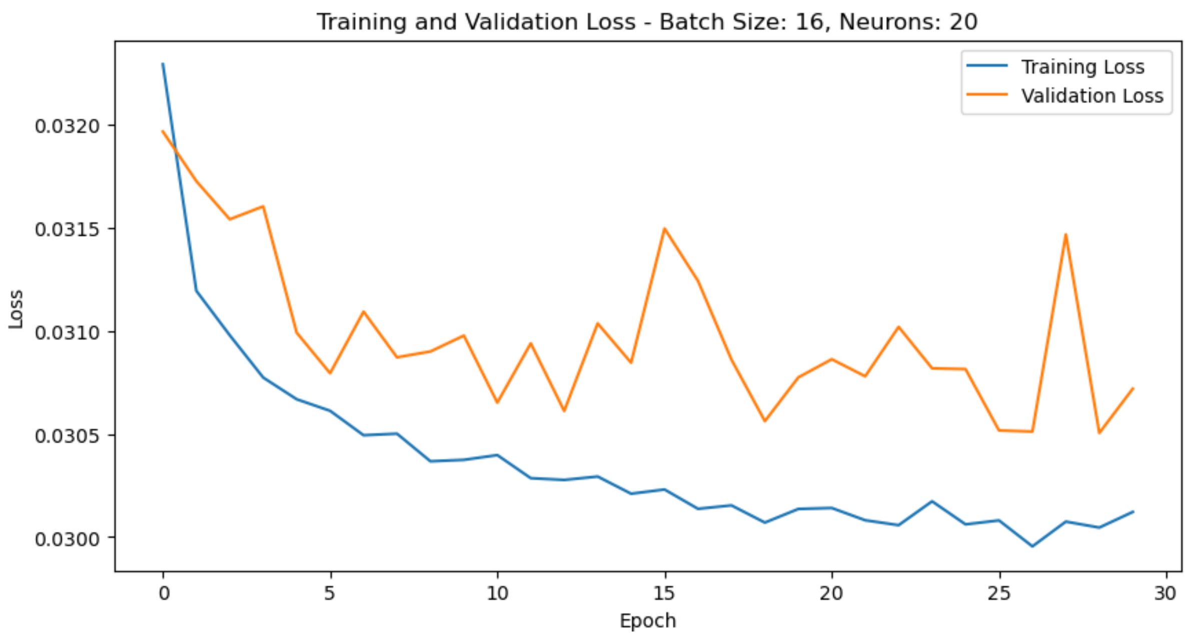


Figure 12: Loss per Epoch for LSTM Model with Batch Size 16 and 20 Neurons per Layer

Batch Size: 16, Neuron Count: 20
Mean Validation Loss: 0.0310
Standard Deviation of Loss: 0.0004
Min/Max Validation Loss: 0.0305/0.0320
Mean Time per Epoch: 5.52 seconds
Total Training Time: 165.50 seconds

Cost Function Summary Statistics

4) Investigate the impact of differing the number of neurons in the hidden layer while keeping the epoch (step 2) and Batch size (step 3) constant for 30 runs. Report the summary statistics (Mean, Standard Deviation, Minimum and Maximum) of the cost function as well as the run time. Discuss how does the number of neurons affect performance and what is the optimal number of neurons in your experiment?

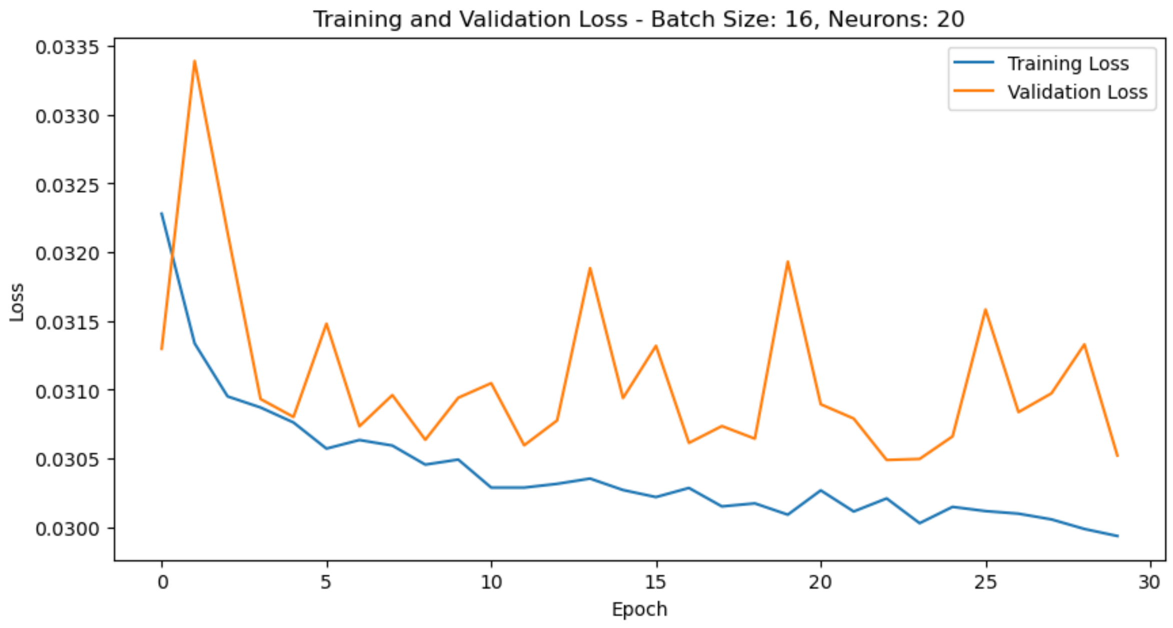


Figure 13: Loss per Epoch for LSTM Model with Batch Size 16 and 20 Neurons per Layer

Batch Size: 16, Neuron Count: 20
Mean Validation Loss: 0.031079659114281337
Std Dev of Loss: 0.0006085978581887188
Min/Max Loss: 0.030489180237054825/0.033391065895557404
Mean Time per Epoch: 5.463182783126831 seconds
Total Training Time: 163.89548349380493 seconds?

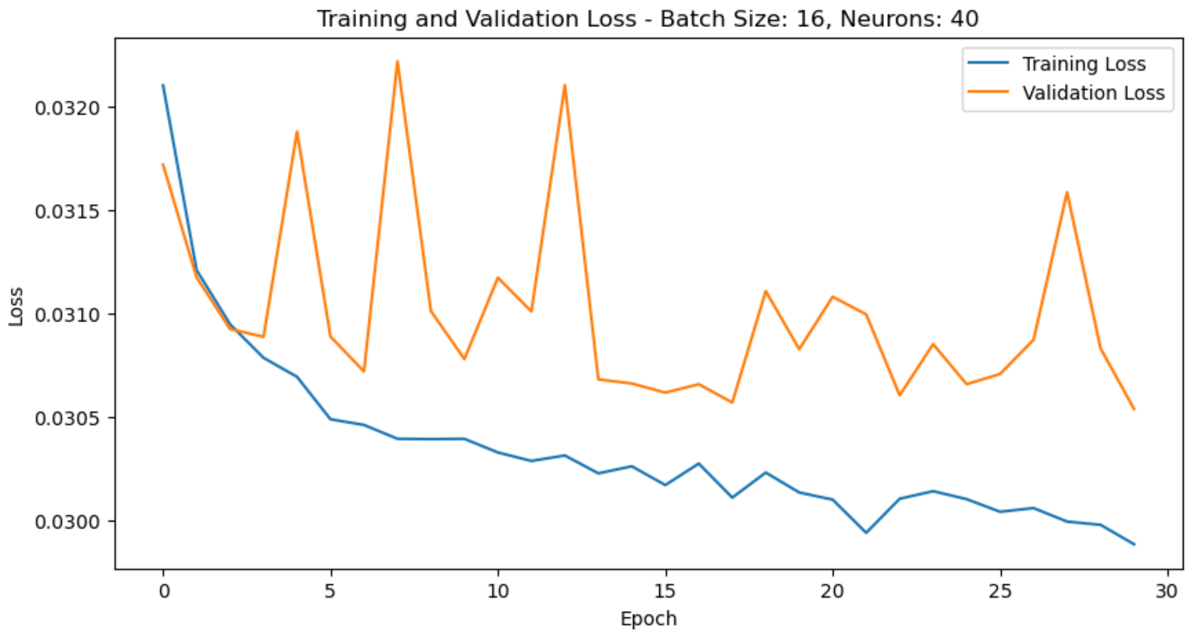


Figure 14: Loss per Epoch for LSTM Model with Batch Size 16 and 40 Neurons per Layer

Batch Size: 16, Neuron Count: 40
Mean Validation Loss: 0.031010025863846144
Std Dev of Loss: 0.0004436743806456964
Min/Max Loss: 0.030536998063325882/0.03221756964921951
Mean Time per Epoch: 6.021631940205892 seconds
Total Training Time: 180.64895820617676 seconds?

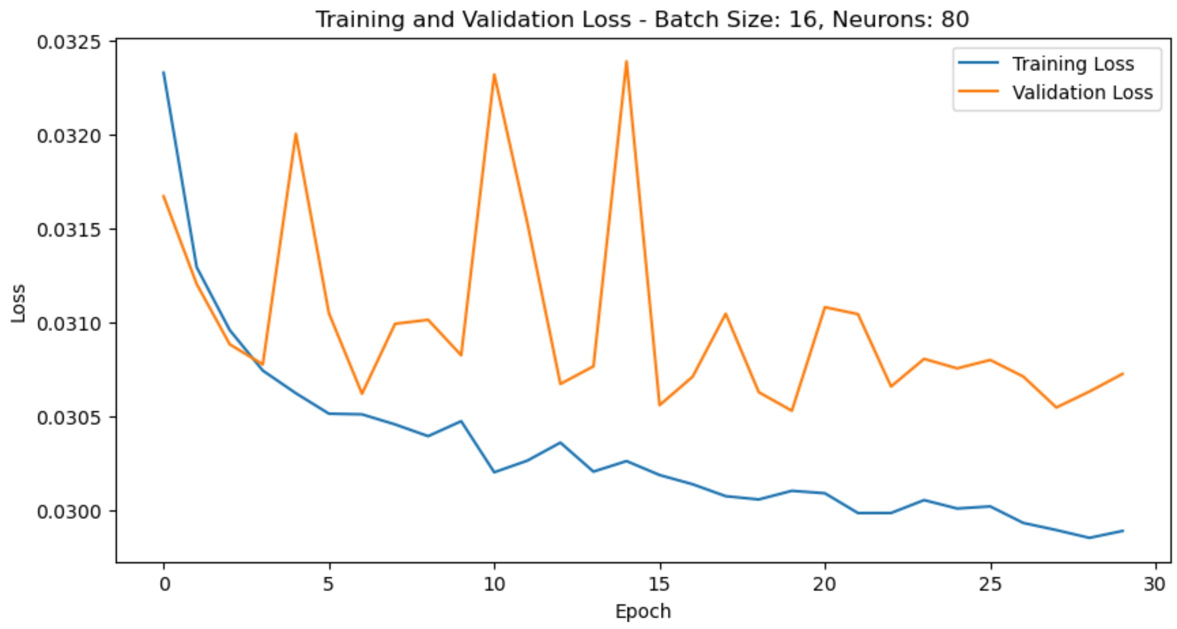


Figure 15: Loss per Epoch for LSTM Model with Batch Size16 and 80 Neurons per Layer

Batch Size: 16, Neuron Count: 80
Mean Validation Loss: 0.030999732079605262
Std Dev of Loss: 0.0004918212138777411
Min/Max Loss: 0.03053099475800991/0.0323910191655159
Mean Time per Epoch: 8.691556406021117 seconds
Total Training Time: 260.74669218063354 seconds?

Increasing the number of neurons from 20 to 40, and then to 80 results in a slight decrease in the Mean Validation Loss, making slight improvements to model performance.

The reduction in loss is minimal, suggesting diminishing returns with increasing complexity (more neurons).

As the number of neurons increases, the Standard Deviation of the Loss decreases slightly from 20 to 40 neurons but remains stable between 40 and 80 neurons. Suggesting that higher neuron counts may offer slightly more consistent model performance.

There is a notable increase in both the mean time per epoch and total training time as the number of neurons increases. This increase is particularly pronounced when moving from 40 to 80 neurons.

The increase in computational time with more neurons is due to the larger number of parameters that need to be updated during training, which also increases the computational burden.

The optimal number of neurons appears to be 40 based on the balance between improved loss (albeit minimal improvement) and the increase in computational time. With 40 neurons, you achieve nearly the lowest mean validation loss with less increase in computational time compared to 80 neurons.

Although 80 neurons provide the lowest mean validation loss, the slight improvement in our opinion does not justify the significantly higher computational cost and time.

While increasing the number of neurons can slightly improve the model's accuracy, it also significantly increases the computational cost. The optimal configuration should balance performance with computational efficiency, which in this case appears to be achieved with 40 neurons.

Model Comparison

1) Plot model-specific actual and predicted PM to visually compare the model performance. What is your observation?

The actual PM2.5 values (in blue) show significant variability which both models strive to predict.
The MLP model predictions (red dashed line) fail to capture many of the peaks and fluctuations observed in the actual data.
The LSTM model predictions (green area) seem to follow the trends in the actual data more closely, though they too miss capturing some of the higher peaks.

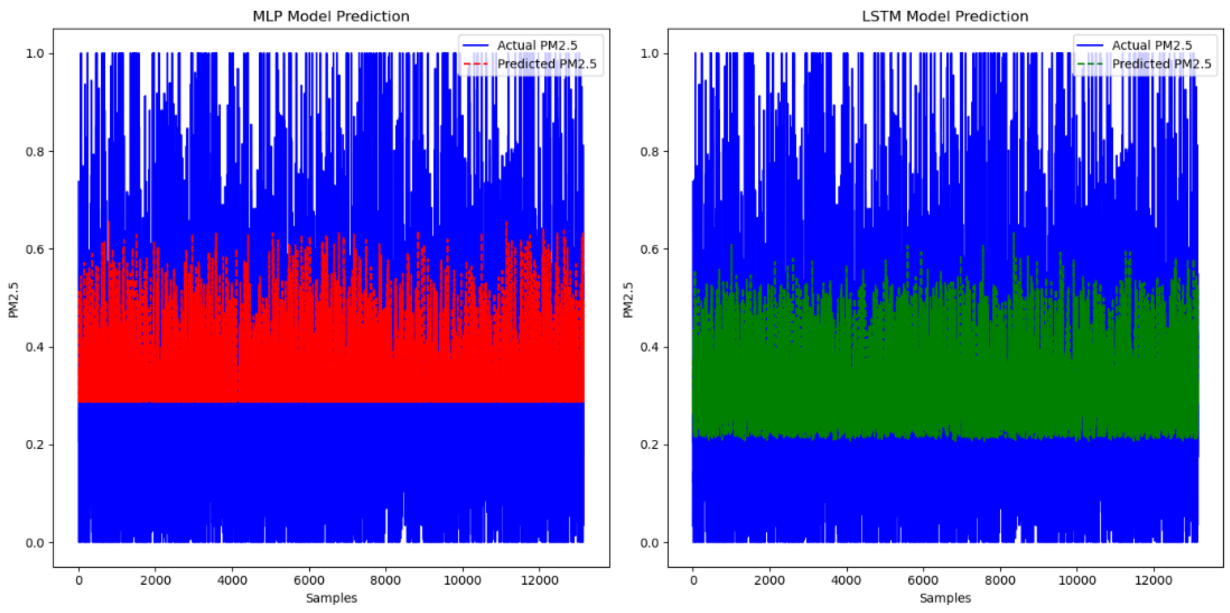


Figure 16: Comparison of model predictions

Performance Comparison:
MLP RMSE: 0.178, MLP MAE: 0.129, MLP R^2: 0.0935
LSTM RMSE: 0.175, LSTM MAE: 0.127, LSTM R^2: 0.123
LSTM model performed better than MLP based on RMSE.

2) Compare the performance of both MLP and LSTM using RMSE. Which model performed better? Justify your finding.

The RMSE for the LSTM model is lower than that of the MLP, indicating that the LSTM model's predictions were closer to the actual values on average. RMSE is a crucial metric for regression models as it measures the average magnitude of the errors in prediction, with lower values indicating better performance.

Similarly, the LSTM model shows a lower MAE, which suggests that the average absolute errors from the LSTM model are smaller than those from the MLP model.

The R^2 score for the MLP model is negative, indicating that the model is arbitrarily worse than a constant baseline. In contrast, the LSTM model's R^2 score, while still low, is positive, suggesting it provides a better fit to the data than the MLP model.

The LSTM model's superior performance is likely due to its ability to capture temporal dependencies and patterns in time-series data, which are crucial in PM2.5 predictions where past values significantly influence future values. Given that the data involves time-series sequences, LSTM's architecture naturally lends itself better for this task than the traditional MLP model.