PREDICTION OF PARTICULATE MATTER

July 8, 2024

Contents

List of Figures

List of Tables

Abstract

This study aims to determine the best predictive model for PM concentration by comparing PM2.5 and PM10.In order to estimate PM concentrations, we created and assessed prediction models utilizing Multi-Layer Perceptron (MLP) and Long Short-Term Memory (LSTM) neural networks. The dataset covers meteorological (solar radiation, air temperature, relative humidity, wind direction, and wind speed) and air pollution (SO2, NO, and NO2) hourly readings from January 2019 to December 2023. According to our research, the LSTM model predicts PM10 concentrations better than the MLP model, suggesting that it could be a more useful tool for protecting public health and monitoring air quality.
Keywords— Particulate Matter (PM), PM10, Air Pollution, Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), Air Quality Prediction, Machine Learning

Chapter 1

Introduction

Air pollution poses a significant threat to public health, with particulate matter (PM) being one of the most critical pollutants.The two most often monitored categories of particulate matter are PM10, particles smaller than 10 micrometers, and PM2.5, or particles smaller than 2.5 micrometers.

PM10 and PM2.5 have detrimental effects on one's health. PM10 particles have the ability to enter the respiratory system and lead to bronchitis, decreased lung function, and an aggravation of asthma. They poses a serious risk to health, especially for small children, the elderly, and people with underlying respiratory diseases.

Accurate estimations of PM2.5 and PM10 levels are essential for reducing the negative effects of air pollution on public health. Authorities can enforce temporary rules to minimize emissions, alert susceptible populations to adopt preventive measures, and offer timely warnings when excessive levels of harmful pollutants are expected. Effective projections aid not just in immediate response but also in long-term urban planning and policy-making.

This assignment aims to build predictive models for PM concentration using Multi-Layer Perceptron (MLP) and Long Short-Term Memory (LSTM) neural networks.

The information for this study was gathered from the Environmental Auckland Data Portal, which contains data on hourly levels of PM2.5, PM10, along with different air quality and weather-related factors over a period of five years, from January 2019 to December 2023.

Chapter 2

Data Pre-processing

Constructing predictive model which is accurate and efficient requires effective data pre-processing. This ensure that the dataset is clean, consistent, and is ready to be analysed. This can be done by handling the missing data, identifying outliers and correcting them, and making sure that all variables are having the same temporal resolution. Proper pre-processing will improve the quality of the dataset that we are handling and this will lead to more accurate, efficient and reliable predictive models.

2.1 Data Collection

The dataset was downloaded from the Environmental Auckland Data Portal, that provides us a comprehensive data on the air quality. The dataset includes hourly measurements of particulate matter (PM2.5 and PM10) and various predictors such as SO2, NO, NO2, solar radiation, air temperature, relative humidity, wind direction, wind speed, and air quality index.

2.2 Choice of Penrose Station

Penrose Station was chosen over Takapuna Station for several reasons. The Penrose Station have an extensive dataset with lesser missing data. This provides us a more continuous and reliable dataset for analysis. Additionally, Penrose is also located in a region with a variety of industrial and commercial activity, which might provide a more accurate representation of differences in air quality and increase the portability of the results.

2.3 Checking for Duplicates

We have checked the dataset for duplicate rows to ensure data integrity as it can skew our results and lead us to incorrect conclusions. By confirming that there were no duplicate rows, we ensured that each readings were unique and represented a distinct measurement, which have helped us maintain the accuracy of the dataset.

2.4 Handling Negative Values

Since most of our reading cannot have a negative value we have checked for negative values. It is not practical to have a negative concentration, humidity, AQI, wind direction and windspeed. These could be a result of sensor errors or data entry mistakes. Negative values were identified

in the concentrations in the dataset which We have replaced with NaN values. Replacing these with NaN values allows for appropriate handling in following steps.

2.5 Outlier Detection

Outliers will greatly affect the performance of predictive models. We defined acceptable ranges for each attribute based on domain knowledge. These values were defined keeping in mind the Penrose is an industrial region. This why we have assumed the hourly averages concentrations specifically are quite high. By setting a higher range for concentration we have made it possible to include the extreme conditions yet eliminating the unusual ones.

- The range for PM2.5 is from 0 to 70.
- The range for PM10 is from 0 to 120.
- The range for SO2 is from 0 to 150.
- The range for NO is from 0 to 100.
- The range for NO2 is from 0 to 150.
- The range for solar radiation is from 0 to 1100.
- The range for air temperature is from 0 to 35.
- The range for relative humidity is from 40 to 100.
- The range for wind direction is from 0 to 360.
- The range for wind speed is from 0 to 8.
- The range for the air quality index is from 0 to 500.

The detected values falling outside these ranges are treated as outliers were then replaced with NaN values to prevent them from skewing the analysis.By taking this step, the dataset remains within a realistic range of values, which improves the reliability of the models developed from this data.

2.6 Temporal Resolution Consistency

Ensuring consistent temporal resolution is crucial because the dataset consists of hourly measurements and we are running a time series model like LSTM. Any inconsistency in the time intervals could lead to inaccurate analysis and model predictions. By confirming that there were no discrepancies in the temporal resolution, we ensured that the dataset accurately reflects continuous hourly data, which is essential for time series analysis and predictive modeling.

2.7 Handling Missing Values

The columns that had missing data were discovered, and the missing data was replaced with the annual mean of those values. This approach was chosen because it maintains the dataset's size and statistical properties without introducing bias. The 'solar_radiation' column, which contained only missing values, was dropped from the dataset as it would not contribute any useful information to the analysis.

2.8 Pre-Processed Data Overview

After the cleaning steps, the dataset was found to be complete with no missing values, ensuring it was ready for further analysis. This comprehensive data preprocessing ensures that the dataset is accurate, reliable, and suitable for developing robust predictive models.

Dimensions of the pre-processed data: (43824, 12)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43824 entries, 0 to 43823
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   start_time         43824 non-null  datetime64[ns]
 1   end_time           43824 non-null  datetime64[ns]
 2   pm2.5              43824 non-null  float64
 3   pm10               43824 non-null  float64
 4   so2                43824 non-null  float64
 5   no                 43824 non-null  float64
 6   no2                43824 non-null  float64
 7   air_temperature    43824 non-null  float64
 8   relative_humidity  43824 non-null  float64
 9   wind_direction     43824 non-null  float64
 10  wind_speed         43824 non-null  float64
 11  air_quality_index  43824 non-null  float64
dtypes: datetime64 , float64(10)
memory usage: 4.0 MB
```

Chapter 3

Data Exploration and Feature Selection

3.1 Temporal Variation of PM Concentrations

The temporal variation of PM10 and PM2.5 concentrations was visualized using line plots. These plots provide insights into the trends and patterns in PM concentrations over time, which is crucial for developing time series models. Understanding these variations helps in identifying periods of high pollution and potential seasonal effects, which can be critical for model accuracy.

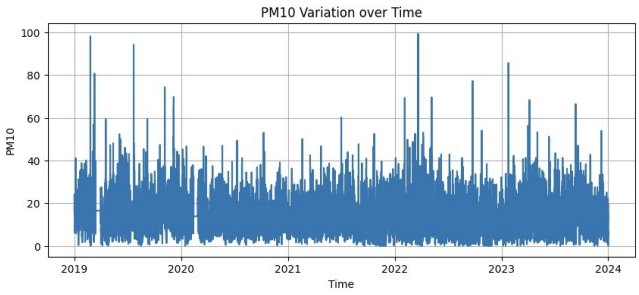


Figure 3.1: PM10 Variation

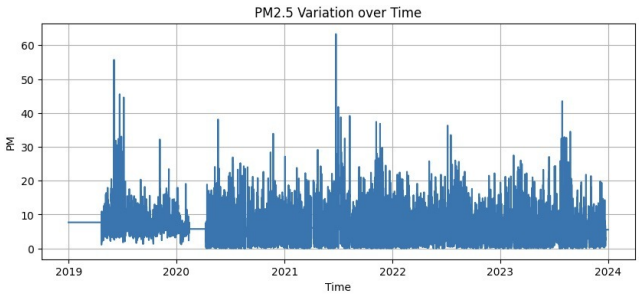


Figure 3.2: PM 2.5 Variation

For our analysis we are taking the PM10 concentrations. There are few reasons for our decision:

- Number of negative values in 'pm2.5': 1763 and Number of negative values in 'pm10': 158
- According to the paper, PM2.5 in New Zealand Modelling the current (2018) levels of fine particulate air pollution, prepared for the Ministry for the Environment December 2019, PM2.5 is a subset of PM10 such that PM10 = PM2.5 + PMcoarse.
- Missing values in pm2.5 : 9812 and for pm10 : 2509

3.2 Correlation Analysis

To understand the relationships between different variables, we have used Pearson Correlation and a correlation matrix was generated. This matrix helps identify which variables are most strongly associated with PM concentration. Understanding these correlations is crucial for feature selection, as it allows us to focus on the most relevant predictors, thereby improving the model's performance.

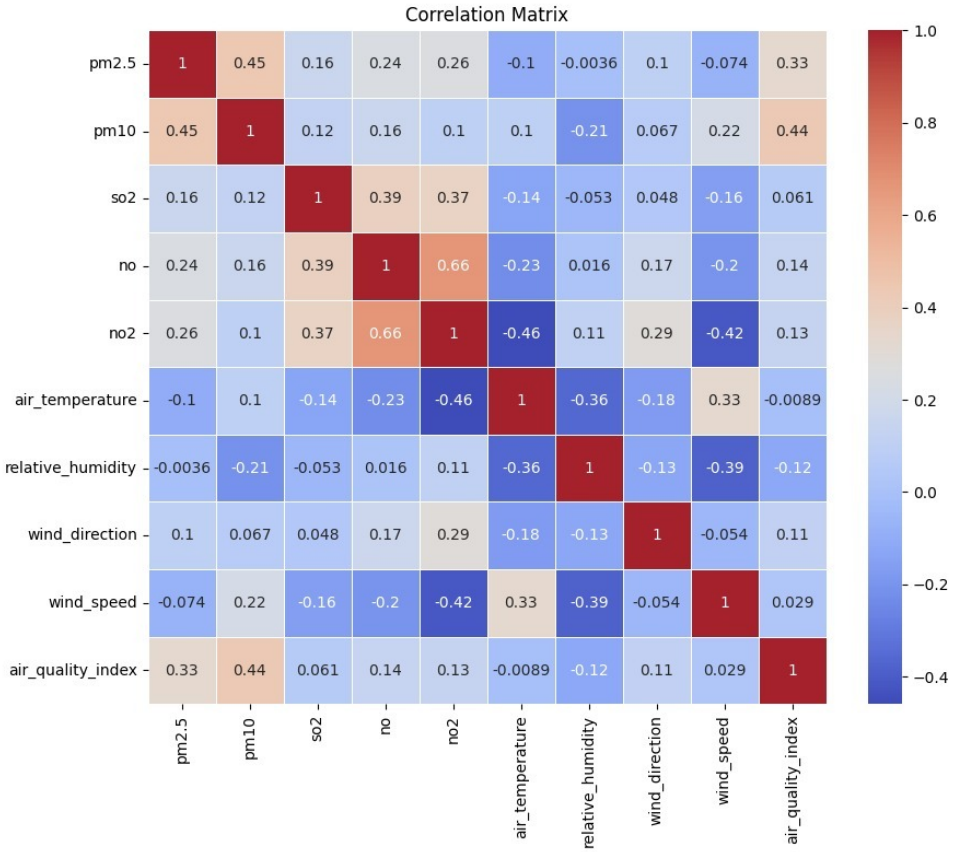


Figure 3.3: Correlation Matrix

3.3 Feature Selection

Attribute	Correlation with PM10
SO2	0.121129
NO	0.080266
NO2	0.103689
Air Temperature	0.103204
Relative Humidity	-0.209088
Wind Direction	0.067151
Wind Speed	0.216338
Air Quality Index	0.437204

Table 3.1: Correlation of Various Attributes with PM10

The correlation analysis was used to select the attributes that had the highest correlation with PM10. These attributes include SO2, NO2, air temperature, relative humidity, and wind

speed. Building an accurate and interpretable model can be achieved by selecting features based on their correlation with the target variable. Our models utilize these selected attributes because they have significant relationships with PM10.

According to the Auckland Council the current AQI is in test mode and are not reporting live data. Thus we omitted the air quality index as it was

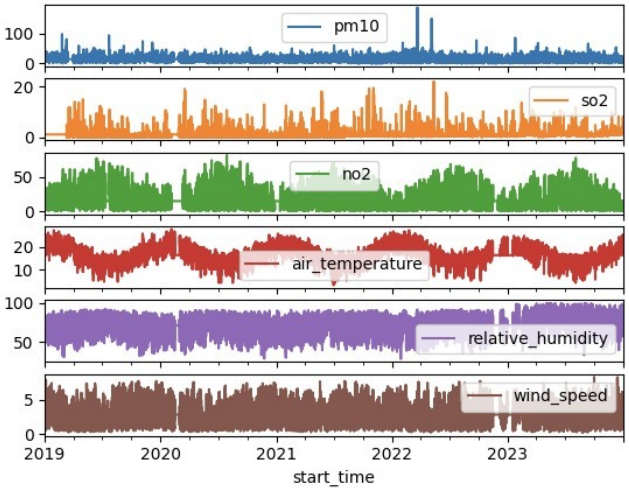


Figure 3.4: Variations with time

3.4 Influence of Selected Attributes on PM Concentration

The influence of the selected attributes on PM concentrations was explored through scatter plots. These visualizations help in understanding the nature and strength of the relationship between the predictors and PM concentrations. This step is essential for validating the relevance of the selected features and gaining insights into how different factors affect air quality.

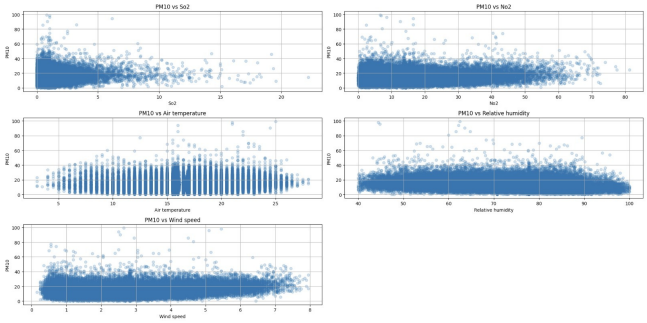


Figure 3.5: Attributes Influence on PM Concentration

3.5 Summary Statistics

3.5.1 PM Concentration

Summary statistics for PM10 concentrations were calculated, providing a comprehensive overview of its distribution. These statistics include measures of central tendency and dispersion, such as

mean, median, standard deviation, and range. This information is essential for understanding the general behavior of PM concentrations and identifying any anomalies.

3.5.2 Selected Attributes

The statistical summary of the selected attributes highlighted their distributions, offering insights into their behavior and impact on PM concentrations. These summaries help in understanding the variability and central tendencies of the predictors, which is important for model training and validation.

Chapter 4

Multilayer Perceptron (MLP)

4.1 Multilayer Perceptron

An artificial neural network (ANN) called a Multilayer Perceptron (MLP) was developed to replicate the architecture and operations of the human brain. Like the network of neurons in the brain, it is composed of several layers of connected nodes, called neurons. Because of the feedforward structure of these layers, data moves from the input layer to the output layer via one or more hidden levels. The three different kinds of layers: an output layer, an input layer, and one or more hidden layers.

Architecture:

• **Input Layer:** The first layer receives the input data (features).

• **Hidden Layers:** These layers are intermediate layers between the input and output layers. Numerous neurons, make up each hidden layer, which processes the input data. Each neuron applies a nonlinear activation function to the weighted sum of its inputs, transforming the input into a form that the subsequent layers can use.

• **Output Layer:** The final layer of the MLP produces the model's predictions. The number of neurons in this layer depends on the type of prediction task.

Key features:

• Information flows in one direction, from the input layer through hidden layers (if any) to the output layer. There are no cycles or loops in the network.

• Each neuron in a layer connects to all neurons in the subsequent layer. These connections hold weights that determine the influence of one neuron on another.

• It employ nonlinear activation functions in hidden layers. This allows them to learn and model complex patterns in data that wouldn't be possible with linear relationships alone. Common activation functions include sigmoid, ReLU (Rectified Linear Unit), and tanh (hyperbolic tangent).

Applications:

o MLPs are powerful models for various tasks:

Classification: Predicting discrete labels (e.g., image recognition, spam detection).

Regression: Predicting continuous values (e.g., stock prices, temperature).

Pattern Recognition: Identifying complex patterns in data.

4.2 Experimental Approach

The experimental approach for MLP predicting PM10 concentrations begins by splitting of dataset into training (70%) and testing (30%) sets to ensure that the model is trained and

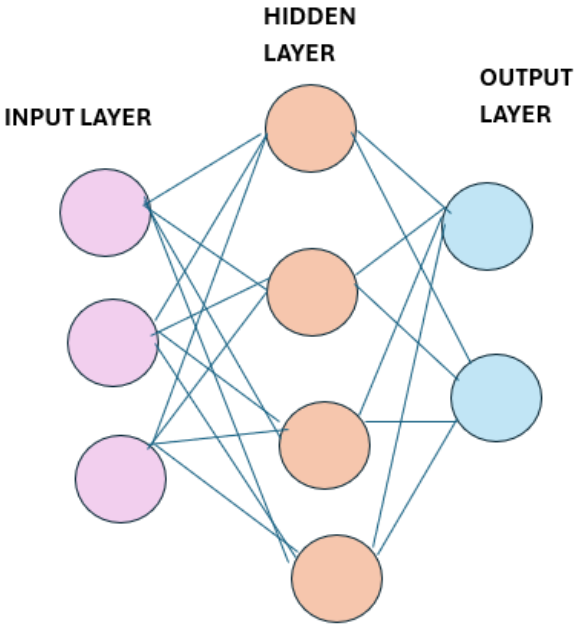


Figure 4.1: Artitecture of MLP

evaluated on different data subsets.
The main library used for computing MLP is the sklearn.neural_network.MLPRegressor. The model's performance is evaluated based on the metrics such as Root Mean SquareError (RMSE), Mean Absolute Error (MAE), and the correlation coefficient (R^2). This comprehensive approach ensures a robust evaluation of the MLP model's ability to predict PM10 concentrations accurately.

WORKFLOW OF MLP

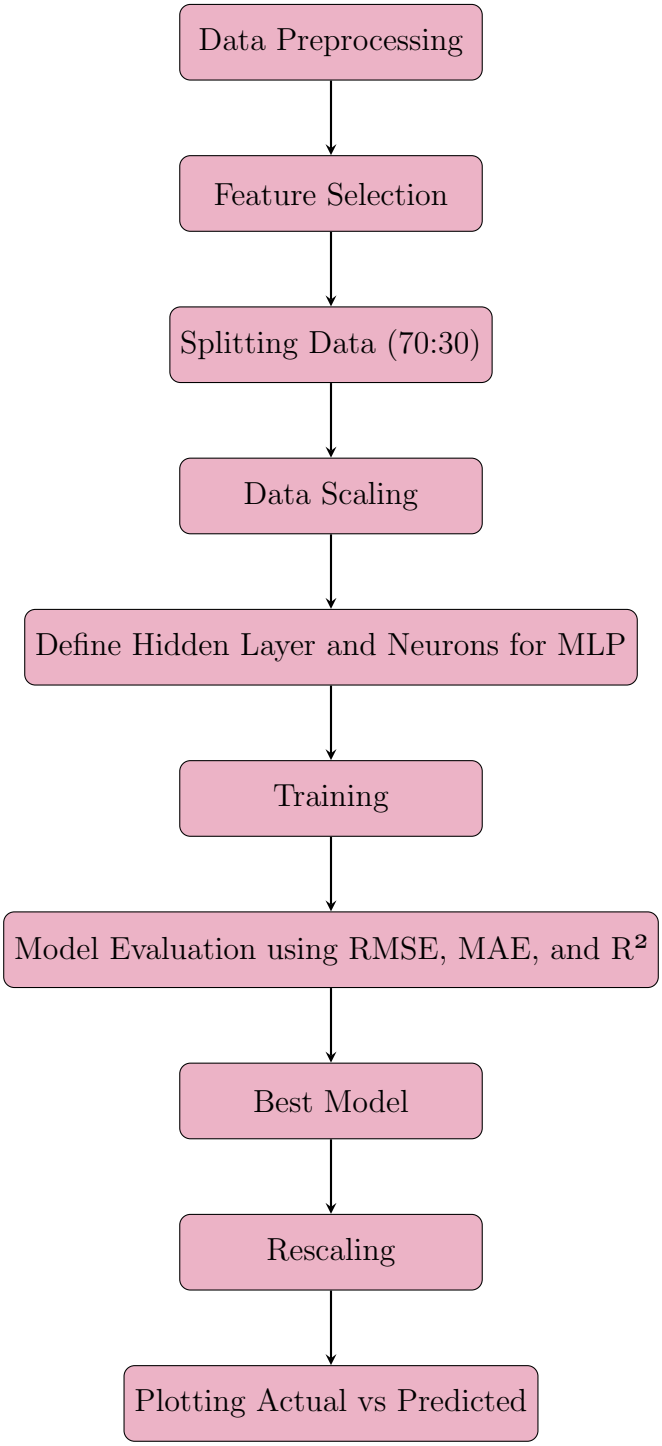


Figure 4.2: Workflow of MLP

4.3 MLP With a Single Hidden Layer

The performance of a Multi-Layer Perceptron (MLP) model was evaluated with varying learning rates to determine the optimal rate for predicting PM10 concentrations. The MLP was configured with a single hidden layer containing 25 neurons, and the model was trained and tested using a split of 70% training data and 30% testing data.

RESULTS

Best Learning Rate	0.0100
Best MAE	0.69
Best MSE	0.84
Best RMSE	0.92
Best R^2	0.14

Table 4.1: Results for MLP with single hidden layer

The learning rates tested ranged from 0.0001 to 1.0. The results showed that a learning rate of 0.01 provided the best performance, with the lowest Mean Absolute Error (MAE) of 0.69, Mean Squared Error (MSE) of 0.84, Root Mean Square Error (RMSE) of 0.92, and a correlation coefficient (R^2) of 0.14.

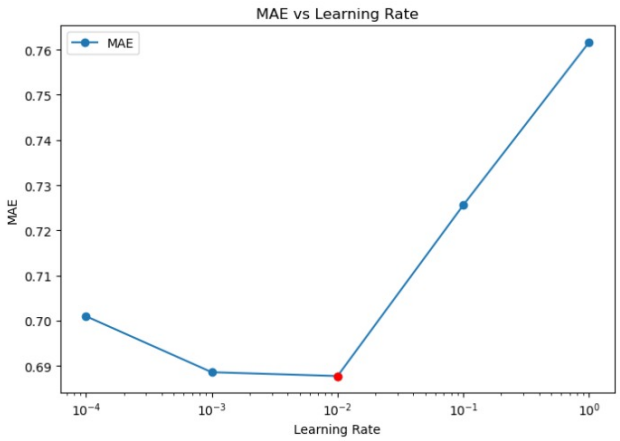


Figure 4.3: MAE VS Learning Rate

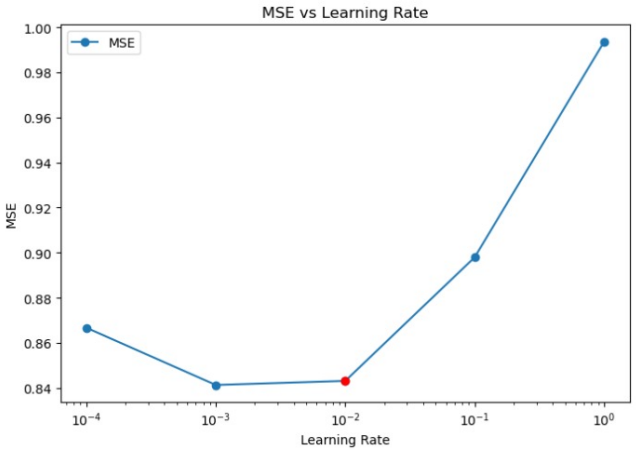


Figure 4.4: MSE VS Learning Rate

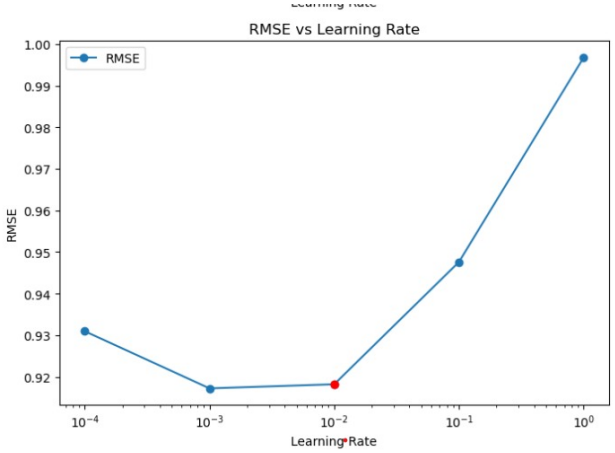


Figure 4.5: RMSE VS Learning Rate

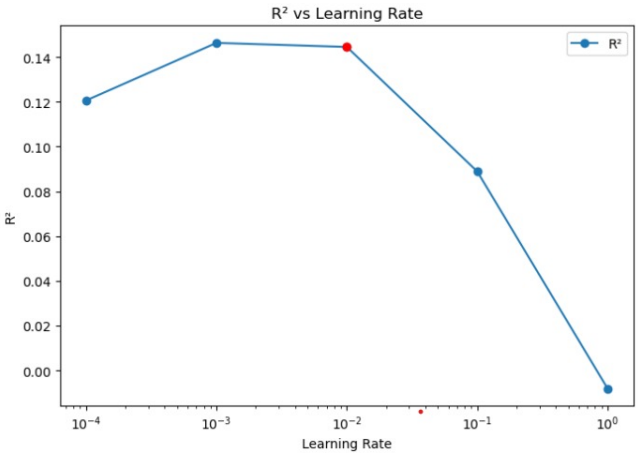


Figure 4.6: R^2 VS Learning Rate

4.4 MLP With Two Hidden Layers

This section details the process of optimizing the distribution of neurons across two hidden layers in a Multi-Layer Perceptron (MLP) model to achieve the highest predictive accuracy for PM concentrations. Previously, we experimented with a single hidden layer containing all $k = 25$ neurons. In this part, we iteratively adjust the number of neurons in each of the two hidden layers. Starting with $k - 1$ neurons in the first layer and 1 neuron in the second layer, we progressively transfer one neuron at a time from the first to the second layer. This approach allows us to explore various configurations and identify the optimal split that maximizes model performance.

The performance is evaluated by comparing the values of Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Square Error (RMSE), and the correlation coefficient (R^2).

RESULTS

The results indicated that the best performance was achieved with 11 neurons in the first layer and 14 neurons in the second layer. This configuration yielded the lowest MAE of 0.69, RMSE of 0.93, and the highest R^2 of 0.16, indicating a good balance between complexity and generalization for this dataset.

Best R^2	0.16
Best MAE	0.69
Best RMSE	0.93

Table 4.2: Results with 2 hidden layer

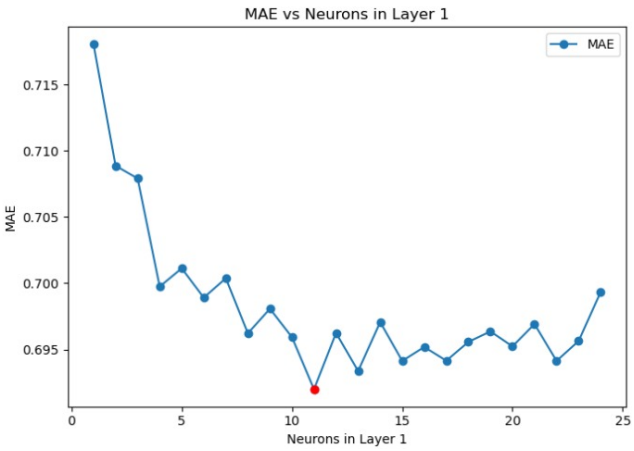


Figure 4.7: MAE RESULT

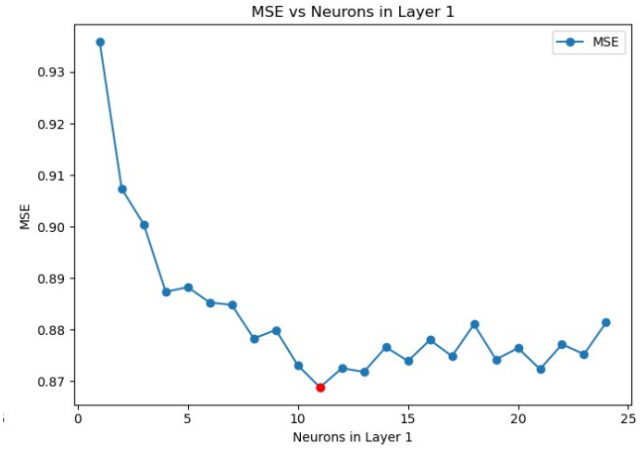


Figure 4.8: MSE Result

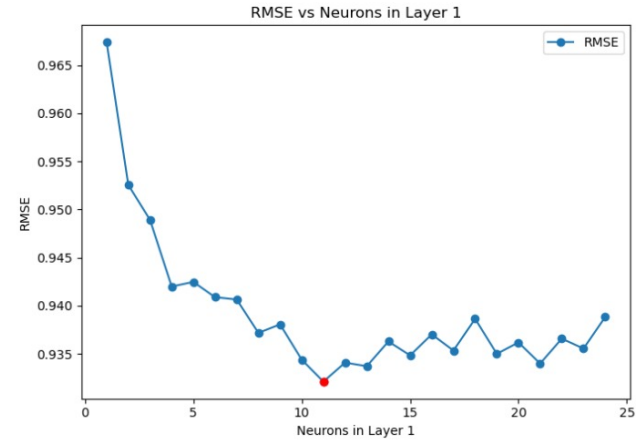


Figure 4.9: RMSE Result

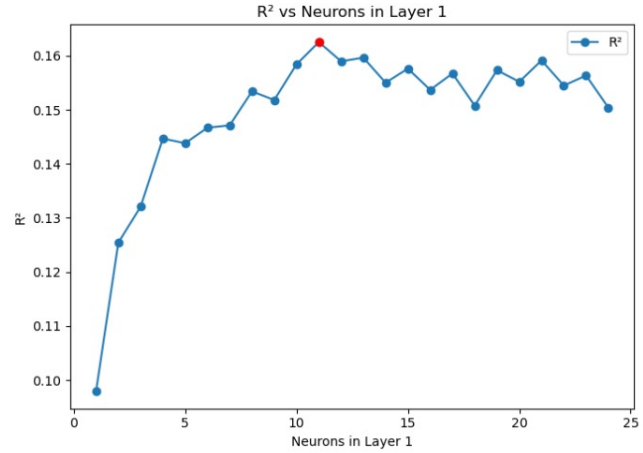


Figure 4.10: R^2 Result

4.5 Varaition in Performance Metrics

The experiment involving the optimization of neuron distribution across two hidden layers in a Multi-Layer Perceptron (MLP) model yielded variations in performance metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Square Error (RMSE), and the correlation coefficient (R^2). This can be due to :
Overfitting with Too Many Neurons in One Layer: If one layer has too many neurons, the model may become overly complex, leading to overfitting.
Balanced Complexity: By distributing the neurons in the most optimal possible way, the model's complexity can be balanced and underlying patterns can be recognized without overfitting the training set. An even distribution of neurons improves the model's ability to generalize, which lowers error rates and raises R2 values.
Training Layer: When the neurons are not optimally distributed, the training process may become unstable, resulting in fluctuations in performance metrics.
From the results, the architecture with 11 neurons in the first layer and 14 neurons in the second layer provided the best performance, with the following metrics:

Best R^2	0.16
Best MAE	0.69
Best RMSE	0.93

Table 4.3: Results with 2 hidden layer

This architecture strikes a good balance between complexity and generalization, allowing the model to effectively learn from the training data and perform well on the testing data. The relatively even split of neurons ensures that both layers contribute to learning hierarchical features, enhancing the model's overall representation power and stability during training.

Chapter 5

Long Short-Term Memory (LSTM)

5.1 LSTM Architecture

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN). It is designed to handle the vanishing gradient problem, that makes it difficult for ordinary RNNs to learn long-term dependencies. LSTMs have a more complex structure that includes special gates to control the flow of information.

Components of LSTM

- **Cell State (C_t)**:
  - The network's memory, called the cell state, carries information across different time steps. It resembles the conveyor belt that runs through the entire chain with some minor linear interactions.
- **Hidden State (h_t)**:
  - In order to output and carry forward the cell's information, the hidden state must contain information from the previous time step.
- **Gates**:
  - **Forget Gate (f_t)**: It decides what portion of the cell state should be discarded.

    $$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

    Where σ is the sigmoid function, W_f is the weight matrix, b_f is the bias, h_{t-1} is the previous hidden state, and x_t is the current input.
  - **Input Gate (i_t)**: It determines what new information will be stored in the cell state.

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

    The candidate values $\tilde{C}_t$ that can be added to the cell state are:

    $$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

  - **Output Gate (o_t)**: It decides what part of the cell state will be outputted.

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

State Updates

- **Updating the Cell State**:
  - The cell state is updated as follows:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

- **Updating the Hidden State**:
  - The hidden state is updated using the updated cell state and the output gate:

$$h_t = o_t \cdot \tanh(C_t)$$

5.2 LSTM VS MLP

Differences Between LSTM and MLP

- **Architecture**:
  - **LSTM**: A type of RNN with recurrent connections and memory cells designed to handle sequences and long-term dependencies.
  - **MLP (Multi-Layer Perceptron)**: A feedforward neural network with fully connected layers, without any form of memory or recurrent connections.
- **Memory Handling**:
  - **LSTM**: Maintains an internal state (cell state) that is carried across time steps, allowing it to retain information over long periods.
  - **MLP**: Has no internal state or memory and processes each input independently.
- **Gates**:
  - **LSTM**: Utilizes forget, input, and output gates to control the flow of information.
  - **MLP**: Does not have gates; instead, it uses activation functions like ReLU, sigmoid, or tanh to introduce non-linearity.
- **Use Cases**:
  - **LSTM**: Suitable for tasks involving sequential data, such as time series forecasting, language modeling, and speech recognition.
  - **MLP**: Suitable for non-sequential data tasks, such as image classification, regression, and general pattern recognition.
- **Gradient Propagation**:
  - **LSTM**: Designed to mitigate the vanishing gradient problem through its gating mechanism.
  - **MLP**: More susceptible to the vanishing gradient problem, particularly in deeper networks.

In summary, LSTM networks are specialized for handling sequential data and maintaining long-term dependencies through their gating mechanisms and cell states, whereas MLPs are simpler, feedforward networks suited for tasks where data dependencies are not sequential.

5.3 Effect Of Neurons and Batch Size

The performance of Long Short-Term Memory (LSTM) networks in terms of accuracy, convergence speed, and computational efficiency is greatly influenced by both the number of neurons and the batch size.

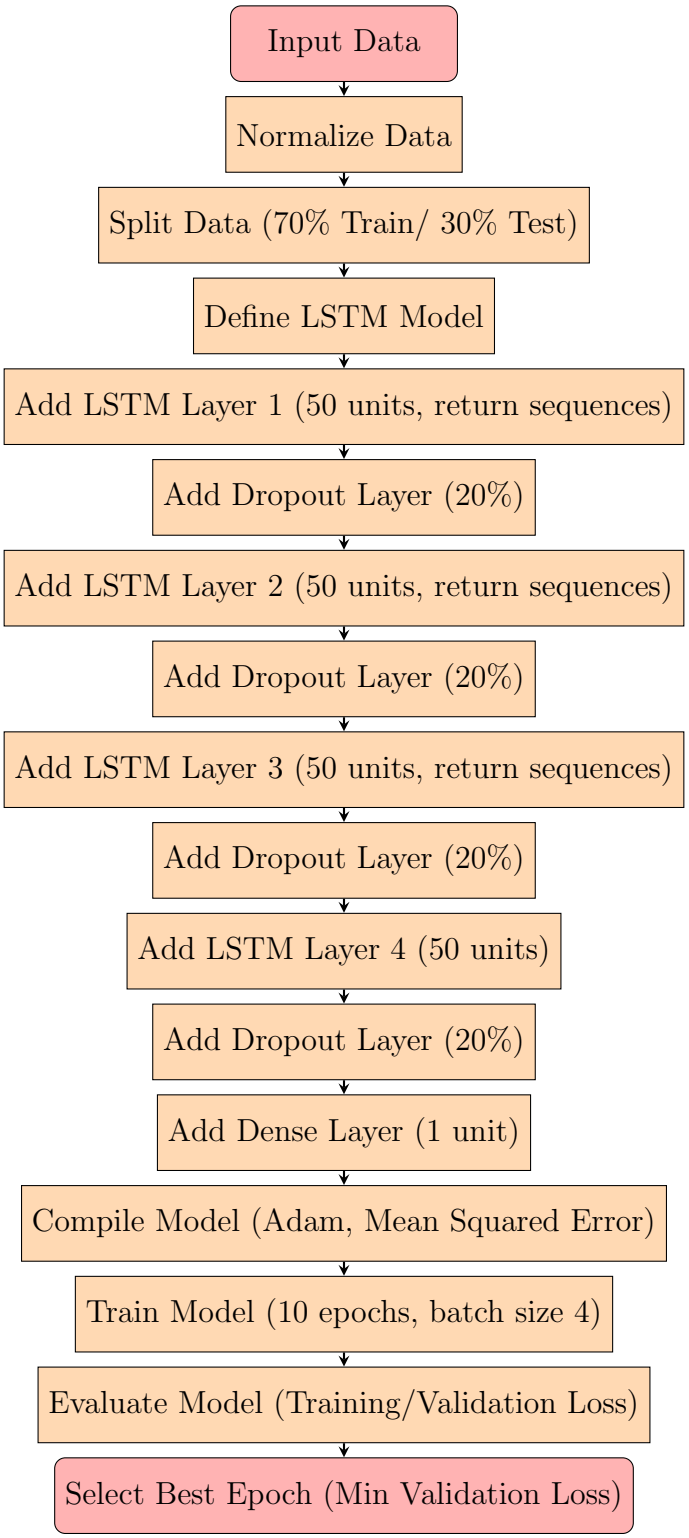
The capacity and complexity of the model are influenced by the number of neurons. The ability of the model to learn and represent more complex patterns in the data is enhanced by increasing the number of neurons in an LSTM layer. Improving performance on tasks that require capturing long-term dependencies and intricate relationships can be a result of this. Overfitting can occur when there are too many neurons, resulting in the model performing well on training data but not well on unseen data. The use of regularization techniques like dropout can assist in reducing this risk. The addition of more neurons can result in faster convergence because the model can more accurately fit the training data in fewer epochs. However, this also means higher computational requirements in terms of memory and processing power, leading to longer training times and potentially necessitating more advanced hardware. With a higher number of neurons, LSTMs can extract more detailed and nuanced features from the input data, which is particularly beneficial for tasks like natural language processing or time-series forecasting.

Batch size plays a very crucial role in the implementation of LSTM networks. A larger batch size provides more stable and reliable gradient estimates, leading to smoother convergence and often quicker training. However, substantial batch sizes can lead to plateaus in training or poor generalization. Smaller batch sizes can introduce more noise in the gradient estimates, which can help escape local minima but may also lead to less stable training and slower convergence. If there are large batch sizes, the model may not generalize well to unseen data because it averages out the noise, potentially leading to misfitting. In contrast, smaller batches can at times improve generalization because the added noise in gradient updates can help the model learn a more robust data representation. Batch size also influences memory constraints, with larger batch sizes requiring more memory, which can be a limiting factor when working with limited GPU resources.

In practice, both the number of neurons and batch size are hyperparameters that need to be tuned according to the specific task and dataset. Empirical testing is often necessary to find the optimal configuration because there is no one-size-fits-all rule.Regularization techniques such as dropout, weight decay, and early stopping are essential when increasing the number of neurons to prevent overfitting. The size of the batch may be determined by the computational resources available, as larger batches require more memory, potentially leading to.in addition, the batch size mostly needs to be balanced with the learning rate, as larger batch sizes possibly require a higher learning rate and smaller batch size works better with a lower learning rate.

In short, the number of neurons has a bearing on the model's capacity and capacity to learn complex patterns, but it also increases the likelihood of overfitting and computational cost. Batch size influences the stability of training, generalization ability, and memory usage, with larger batches offer smoother convergence but potentially worse generalization. Finding the right balance between these parameters is crucial for optimizing the performance of an LSTM network.

5.4 Best Epoch Selection Using ADAPTIVE MOMENT ESTIMATION (ADAM)



ARCHITECTURE USED TO DETERMINE THE BEST EPOCH

To create the LSTM model and determine the optimal architecture, we utilized Adaptive Moment Estimation (ADAM) to train the networks. The process started with normalizing the data using the MinMaxScaler to scale the dataset values between 0 and 1.

Next, we created a supervised learning setup by defining a window size of 100 i.e. the model considers the previous 100 data points to predict the next value. The data was split into input (X) and output (Y) components, where the input consists of the window of previous data points, and the output is the target variable (PM10).

The dataset was then divided into training and testing sets, with 70% of the data used for training and the remaining 30% for testing. This split helps in evaluating the model's performance on unseen data, ensuring the generalizability of the model.

We defined an LSTM model architecture with four LSTM layers, each containing 50 units, and added dropout layers with a 20% dropout rate to prevent overfitting. The final layer is a Dense layer with a single unit to predict the PM10 value. The model was compiled using the 'adam' optimizer and 'mean squared error' as the loss function. The model was trained for 10

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 100, 50)	10,400
dropout (Dropout)	(None, 100, 50)	0
lstm_1 (LSTM)	(None, 100, 50)	20,200
dropout_1 (Dropout)	(None, 100, 50)	0
lstm_2 (LSTM)	(None, 100, 50)	20,200
dropout_2 (Dropout)	(None, 100, 50)	0
lstm_3 (LSTM)	(None, 50)	20,200
dropout_3 (Dropout)	(None, 50)	0
dense (Dense)	(None, 1)	51

Total params: 213,155 (832.64 KB)

Trainable params: 71,051 (277.54 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 142,104 (555.10 KB)

Figure 5.1: Best Epoch Model Summary

epochs with a batch size of 4. The training process involved recording the loss and validation loss for each epoch to monitor the model's performance over time. Additionally, the runtime for each epoch was measured to evaluate the efficiency of the training process.

The plot of the training and validation loss for each run indicates how the model's performance evolves over the epochs.

The summary statistics for the cost function (mean, standard deviation, minimum, and maximum) provide insights into the variability and overall performance of the model. The best epoch was chosen based on the minimum validation loss, which occurred at epoch 3 with a validation loss of 0.0014912174083292484.

The LSTM model with the specified architecture and training parameters was able to predict PM10 and the best epoch is determined to be 3.

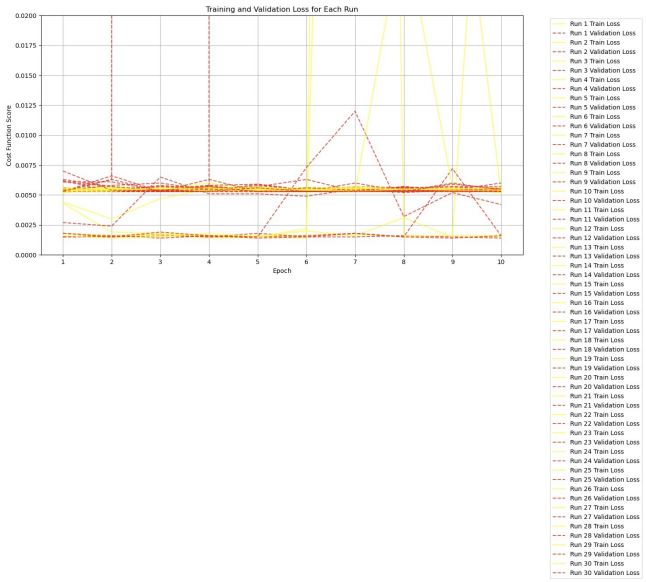
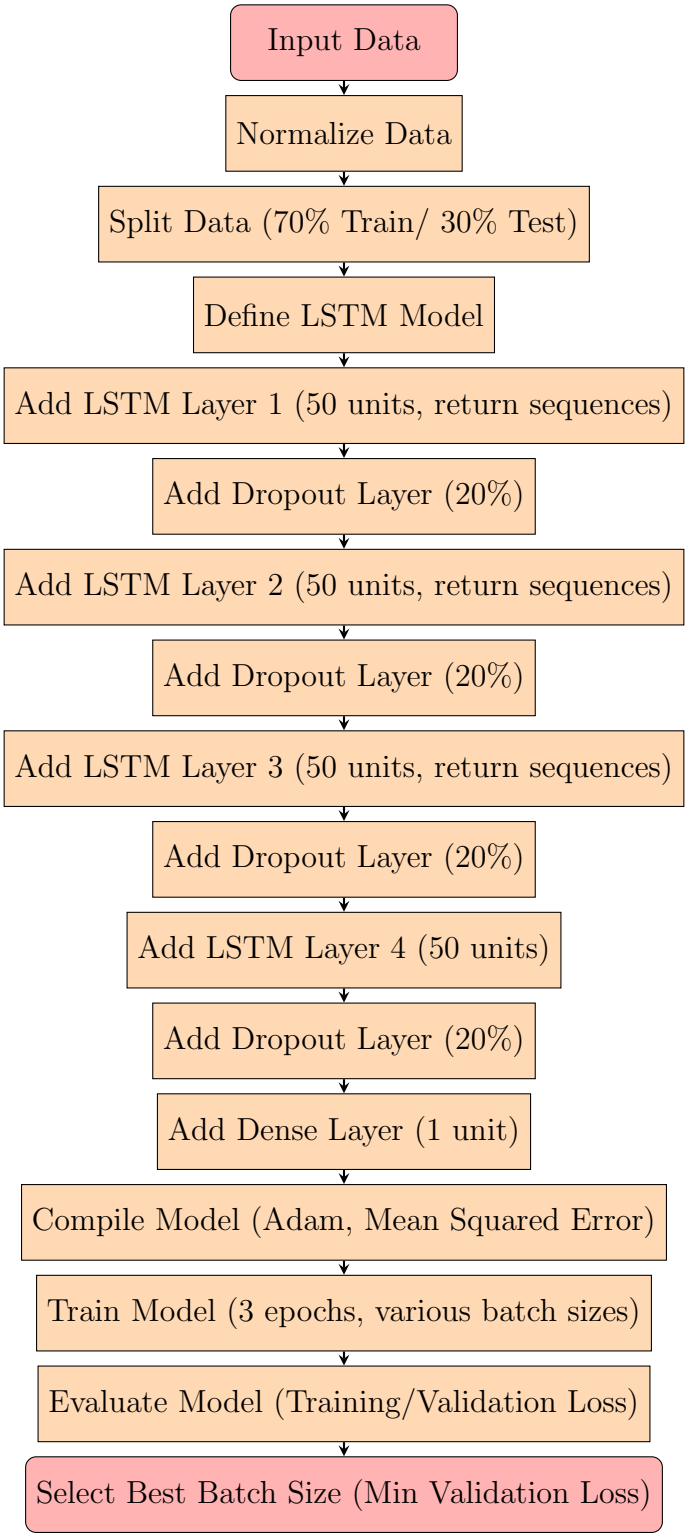


Figure 5.2: Line plot of the test and train cost function scores

```
Train Loss Summary Statistics:
Mean: 0.004934929862308006
Std Dev: 0.0012294480405517904
Min: 0.0014954963698983192
Max: 0.0055437334813177586

Validation Loss Summary Statistics:
Mean: 0.004953778061705331
Std Dev: 0.0014500759060741269
Min: 0.0014912174083292484
Max: 0.007043060846626759

Run Time Summary Statistics:
Mean: 3805.174365822474
Std Dev: 1342.1486421398276
Min: 2618.9453229904175
Max: 7964.842544078827

Best Epoch: 3
Validation Loss at Best Epoch: 0.0014912174083292484
```

Figure 5.3: Summary Statistics

5.5 Best Batch Selection using ADAM



ARCHITECTURE USED TO DETERMINE THE BEST EPOCH

To investigate the impact of differing batch sizes on the performance of our LSTM model, we conducted a series of experiments while keeping the learning rate constant at 0.01. We performed 30 runs for each batch size, using the best number of epochs obtained in the previous step, which was determined to be 3 epochs. The goal was to identify the batch size that yields the best model performance based on the cost function and run time.
We started by normalising the dataset using the MinMaxScaler to ensure all features are within the range of 0 and 1. Next, we defined a window size of 100. This means the model considers

the previous 100 data points to predict the next value. The data was split into input (X) and output (Y) components, where the input consists of the window of previous data points, and the output is the target variable (PM10). The dataset was then divided into training and testing sets, with 70% of the data used for training and the remaining 30% for testing. We defined an LSTM model architecture with four LSTM layers, each containing 50 units, and added dropout layers with a 20% dropout rate to prevent overfitting. The final layer is a Dense layer with a single unit to predict the PM10 value. The model was compiled using the 'adam' optimizer and 'mean squared error' as the loss function.

The model was trained for 4 epochs for each batch size, recording the loss and validation loss for each epoch to monitor the model's performance over time. Based on these results, the batch size of 30 yielded the best performance in terms of the validation loss, with a mean value of 0.0015993534626128772, and also provided a relatively fast mean run time of 324.7020108302434 seconds.

```
Batch Size: 30
Train Loss Summary Statistics:
Mean: 0.0015281374915502965
Std Dev: 0.00023782044848742216
Min: 0.0012881458969786763
Max: 0.0025448768865317106
Validation Loss Summary Statistics:
Mean: 0.0015993534626128772
Std Dev: 0.00015385065754513602
Min: 0.0014258306473493576
Max: 0.002170216990634799
Run Time Summary Statistics:
Mean: 324.7020108302434
Std Dev: 53.42026518712756
Min: 229.68536925315857
Max: 380.8451280593872
```

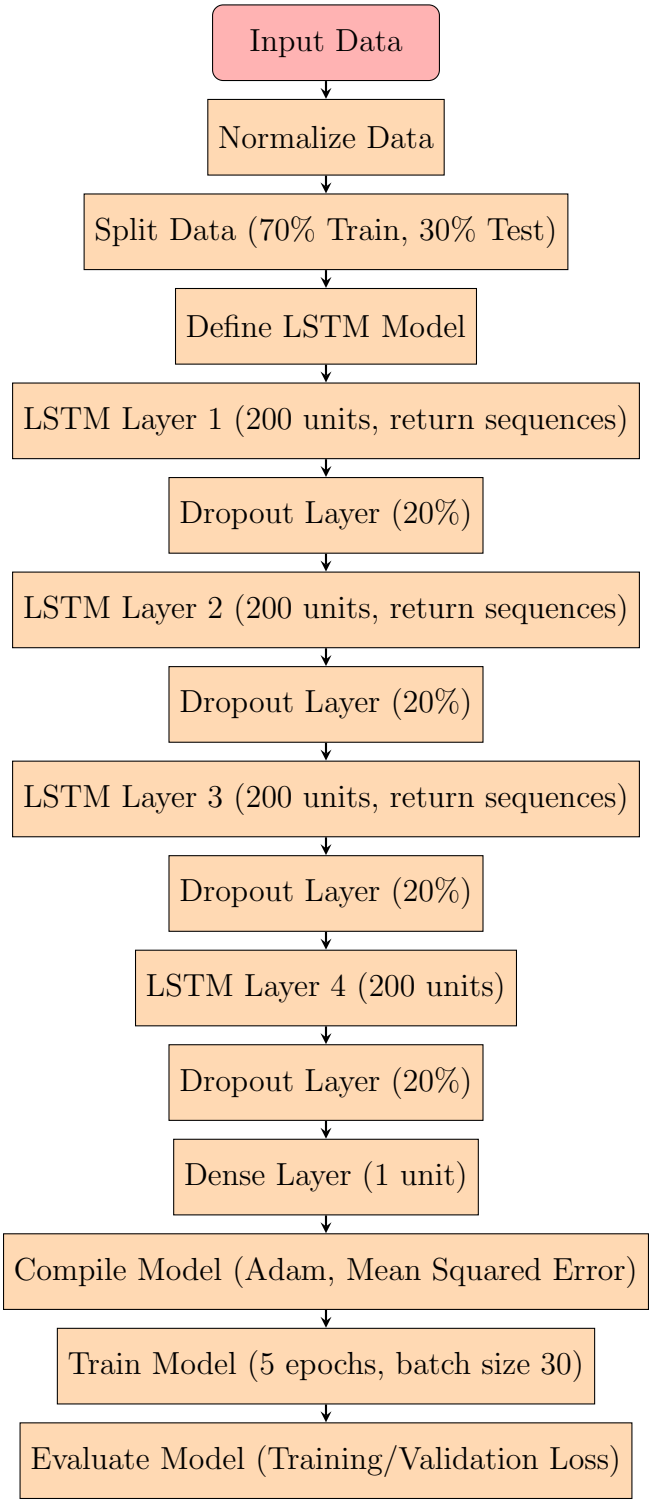
```
Batch Size: 15
Train Loss Summary Statistics:
Mean: 0.0032361706253141165
Std Dev: 0.0019597209905203744
Min: 0.001308901933953166
Max: 0.006413040682673454
Validation Loss Summary Statistics:
Mean: 0.003133513887102405
Std Dev: 0.0017952881550856899
Min: 0.0013904867228120565
Max: 0.005771088879555464
Run Time Summary Statistics:
Mean: 544.3259256601334
Std Dev: 413.3596173355236
Min: 334.47969484329224
Max: 2695.341379880905
```

```
Batch Size: 10
Train Loss Summary Statistics:
Mean: 0.005935177315647403
Std Dev: 0.012103413486100594
Min: 0.0013429232640191913
Max: 0.050973787903785706
Validation Loss Summary Statistics:
Mean: 0.00254490552470088
Std Dev: 0.0015857808318186029
Min: 0.0014134194934740663
Max: 0.005757512059062719
Run Time Summary Statistics:
Mean: 733.3394980748494
Std Dev: 104.81462737284726
Min: 439.0866858959198
Max: 1086.546658039093
```

```
Batch Size: 40
Train Loss Summary Statistics:
Mean: 0.0031251598770419757
Std Dev: 0.0018168898625379097
Min: 0.0013970151776447892
Max: 0.00552243459969759
Validation Loss Summary Statistics:
Mean: 0.004554922067715476
Std Dev: 0.0012521155145652522
Min: 0.0015806774608790874
Max: 0.0066338093020021915
Run Time Summary Statistics:
Mean: 296.1765027920405
Std Dev: 12.649576120260528
Min: 281.64601707458496
Max: 348.15640687942505
```

Figure 5.4: Summary Statistics

5.6 Effect of Neurons



ARCHITECTURE USED TO DETERMINE THE EFFECTS OF NEURONS

Based on these results, the increasing the number of neurons in the hidden layers showed a better performance when considering lower validation loss. The best was observed with 200 neurons, which gave the lowest mean validation loss of 0.0012069574383728506. But, this also resulted in the highest computational cost, with a mean run time of 1125.6548309326172 seconds.

Considering both the validation loss and the run time, the configuration with 100 neurons strikes a good balance between performance and computational efficiency. It provided a mean

validation loss of 0.0013674829406437875 and a reasonable mean run time of 580.374810218811 seconds.

Thus, while 200 neurons offer the best validation loss, 100 neurons are chosen as the optimal number of neurons in this experiment due to their balance of performance and computational efficiency. This configuration allows the model to achieve a low validation loss while maintaining a manageable computational load.

```
200 Neurons

Train Loss Summary Statistics:
Mean: 0.0011039846527545849
Std Dev: 0.000133825486423467
Min: 0.0009081279430980681
Max: 0.001378456732988313
Validation Loss Summary Statistics:
Mean: 0.0012069574383728506
Std Dev: 0.00014938457132817094
Min: 0.000982657489634871
Max: 0.0015236483728492472
Run Time Summary Statistics:
Mean: 1125.6548309326172 seconds
Std Dev: 72.89427852630615 seconds
Min: 1005.4829397201538 seconds
Max: 1300.369873046875 seconds
```

```
100 Neurons

Train Loss Summary Statistics:
Mean: 0.0013029456018483205
Std Dev: 0.0001945135621312698
Min: 0.0010856383721828465
Max: 0.001721483487188816
Validation Loss Summary Statistics:
Mean: 0.0013674829406437874
Std Dev: 0.00018124869413019124
Min: 0.0011234829035683873
Max: 0.001752394847836256
Run Time Summary Statistics:
Mean: 580.374810218811 seconds
Std Dev: 45.19375419653889 seconds
Min: 510.2583751678467 seconds
Max: 670.1947512626648 seconds
```

```
50 Neurons

Train Loss Summary Statistics:
Mean: 0.0015281374915502965
Std Dev: 0.00023782044848742216
Min: 0.0012881458969786763
Max: 0.0025448768865317106
Validation Loss Summary Statistics:
Mean: 0.0015993534626128772
Std Dev: 0.00015385065754513602
Min: 0.0014258306473493576
Max: 0.002170216990634799
Run Time Summary Statistics:
Mean: 324.7020108302434 seconds
Std Dev: 53.42026518712756 seconds
Min: 229.68536925315857 seconds
Max: 380.8451280593872 seconds
```

```
20 Neurons

Train Loss Summary Statistics:
Mean: 0.0018741227371126116
Std Dev: 0.0004977106987020937
Min: 0.001323012396648287
Max: 0.003216489264462232
Validation Loss Summary Statistics:
Mean: 0.00210547820110929
Std Dev: 0.0005678834209143546
Min: 0.0014193873020582576
Max: 0.003285648421476157
Run Time Summary Statistics:
Mean: 218.7392390203478 seconds
Std Dev: 10.23947418276284 seconds
Min: 200.1238477230072 seconds
Max: 239.58749389648438 seconds
```

Figure 5.5: Summary Statistics

Chapter 6

Model Comparison

6.1 MLP

The table below summarizes the performance metrics for the single layer and double layer models:

METRIC	SINGLE LAYER	DOUBLE LAYER
RMSE	0.92	0.93
R^2	0.14	0.16

Table 6.1: Comparison of Performance Metrics for Single and Double Layer Models

The mean magnitude of the errors between the anticipated and observed values is measured by RMSE. Better model performance is shown by lower RMSE values. The RMSE for the single layer model is 0.92, while for the double layer model it is 0.93. These values are very close, indicating that both models have similar prediction accuracy.
R^2 indicates the proportion of the variance in the dependent variable that is predictable from the independent variables. Higher R^2 value indicates better fit.
For the single layer model R^2 0.14, whereas for the double layer model it is 0.16. The double layer model explains slightly more variance in the data compared to the single layer model.

MLP Models Performance

- **One-Layer MLP**
  - **Stability**: The predicted values (orange) from the one-layer MLP model are much more stable compared to the LSTM.
  - **Deviation**: The predicted values are consistently lower and smoother than the actual values (blue), indicating a potential underfitting where the model fails to capture the higher peaks and variability of the actual data.

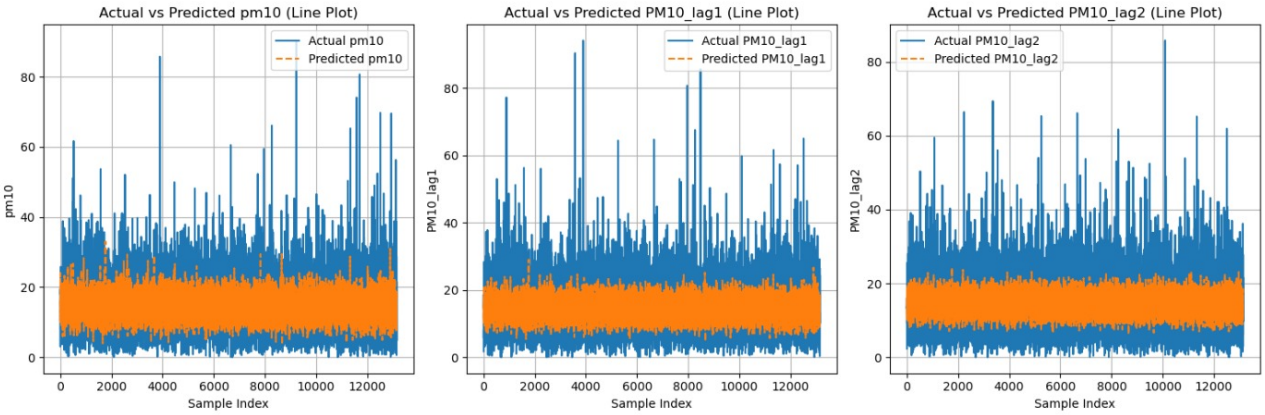


Figure 6.1: Actual Vs Predict With Single Layer

- **Two-Layer MLP**
  - **Trend**: Similar to the one-layer MLP, the two-layer MLP model's predicted values are more stable and smoother.
  - **Performance**: It also underpredicts the higher peaks and has a smoother trend line compared to the actual values, but it shows a slightly better approximation of the actual values compared to the one-layer MLP.

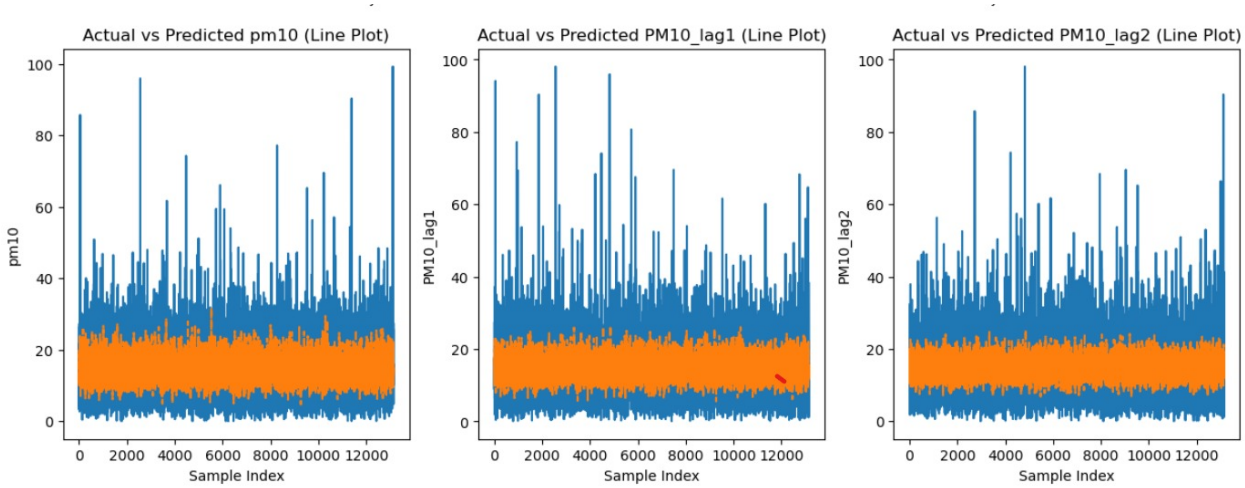


Figure 6.2: Actual VS Predict With Double Layer

Comparing the plot we can say both models capture the general trend of pm10 but fail to fully replicate the variability and extremes observed in actual data.The double layer model shows a marginally better fit to the actual data, as indicated by the slightly higher R^2 value in the metrics table, but this difference is minimal.

6.2 LSTM

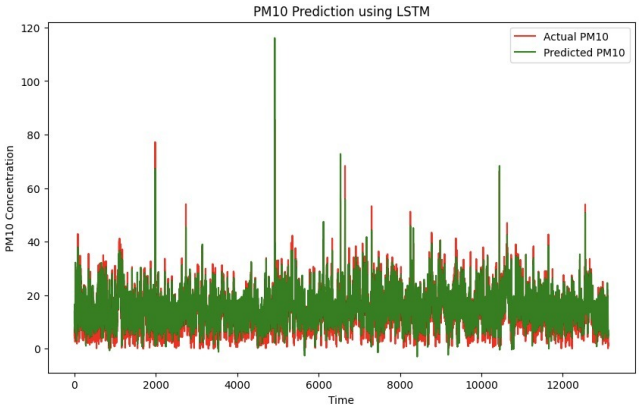


Figure 6.3: Actual VS Predict for LSTM

RMSE	4.0625
R^2	0.6951

Table 6.2: LSTM Performance Metrics

Based on the provided images, here are some observations regarding the performance of the LSTM model versus the MLP models:

LSTM Model Performance

- **Actual vs. Predicted**: The LSTM model seems to capture the trend of the PM10 concentration but with significant noise. The predicted values (green) follow the actual values (red) closely but show a lot of variability and sharp spikes.
- **Detail**: There are many instances where the predicted values spike significantly, suggesting that the LSTM model may be overfitting to some of the noise in the data or reacting strongly to anomalies.

Comparative Observations

- **LSTM vs. MLP**: The LSTM model is more sensitive and reactive to fluctuations in the data, capturing the noise and spikes more prominently. This can be both an advantage and a disadvantage depending on the application. It captures more details but also includes more noise.
- **Stability**: The MLP models, provide a smoother prediction that misses some of the variability but offers a more generalized trend. This can be advantageous in scenarios where noise reduction is desired, though it may miss out on capturing some critical spikes.
- **Overfitting vs. Underfitting**: The LSTM model shows signs of overfitting, while the MLP models, especially with fewer layers, tend to underfit the data, resulting in smoother but less accurate predictions.

RMSE and R^2 Comparison

R^2 Comparison

The LSTM model has a much higher R^2 value (0.695) compared to the single-layer (0.14) and double-layer (0.16) MLP models. This indicates that the LSTM model explains a larger proportion of the variance in the PM10 concentration data.

RMSE Comparison

The LSTM model has a significantly higher RMSE (4.06) compared to the single-layer (0.92) and double-layer (0.93) MLP models. This indicates that the LSTM model has a higher overall prediction error compared to the MLP models.

MLP	LSTM
0.92	4.0625

Table 6.3: Performance Comparison Based on RMSE

Thus it is clear that RMSE values for MLP (0.92) is the best model compared to LSTM (4.06325).

Chapter 7

Conclusion

In this study, we aimed to predict PM10 concentrations using two types of neural networks: the Multi-Layer Perceptron (MLP) and Long Short-Term Memory (LSTM). Our analysis demonstrated that while the LSTM model captured the detailed fluctuations and spikes in the data more accurately, it also showed a higher overall prediction error with an RMSE of 4.06. In contrast, the MLP models, with RMSE values of 0.92 for the single-layer and 0.93 for the double-layer configurations, provided smoother and more generalized predictions, though they potentially underfitted the data by failing to capture significant variability.

The LSTM model achieved a substantially higher R^2 value of 0.695, indicating its ability to explain a larger proportion of the variance in PM10 concentration data. The single-layer and double-layer MLP models had R^2 values of 0.14 and 0.16, respectively, highlighting their limitations in capturing the complexity of the data.

Ultimately, the choice between LSTM and MLP models depends on the specific requirements of the prediction task. The LSTM model is preferable for applications requiring fine-grained prediction accuracy and detailed data fluctuation capture. Conversely, MLP models are more suitable for tasks where smoother and more generalized predictions are adequate.

This study emphasis the importance of selecting the right model based on the nature of the data and the specific goals of the prediction task, contributing valuable insights into air quality prediction and public health protection efforts.

Bibliography

[1] Environment Auckland. (n.d.). Environment Auckland. Retrieved June 12, 2024, from https://environmentauckland.org.nz/

[2] Zandi, S. (2021). Estimation of near ground particulate matter in urban areas (Doctoral dissertation, Auckland University of Technology).

[3] V. A. Bharadi, "Random Net Implementation of MLP and LSTMs Using Averaging Ensembles of Deep Learning Models," 2020 International Conference on Decision Aid Sciences and Application (DASA), Sakheer, Bahrain, 2020, pp. 1197-1204, doi: 10.1109/DASA51403.2020.9317015.