Multi-Step Time Series Forecasting Using LSTM Networks: A Case Study on PM2.5 Concentration

Yixin Wang

*Ulster College*

*Shaanxi University of Science and Technology*

Xi’an, China

Wang-Y52@ulster.ac.uk

*Abstract*—This study examines how Long Short-Term Memory (LSTM) networks may be used for multi-step time series forecasting, with a particular emphasis on forecasting PM2.5 concentration levels. The study makes use of a dataset that includes information on Beijing's air quality along with other factors like temperature, pressure, and wind speed. Before dividing the data into training and testing sets, preprocessing and normalization are performed. An LSTM model with several hidden layers is created, trained, and used to a problem of sequence prediction. To evaluate the efficacy of the model, metrics including the coefficient of determination (R2 score), mean absolute error (MAE), and root mean squared error (RMSE) are utilized. Additionally, a time series plot of the PM2.5 readings and the data's correlation matrix are examined. The findings show that LSTM models can anticipate PM2.5 concentration levels with good accuracy and predictive capabilities.

Keywords—LSTM, PM2.5, time series forecasting, Adam optimizer, RMSE, MAE, R2 score, MFE, MAPE.

# Introduction

Air pollution is a pressing global problem that has substantial concerns for the environment and human health. Airborne particles with a diameter of 2.5 micrometers or smaller are referred to as PM2.5 (particulate matter 2.5), is a major component of air pollution and has been linked to various respiratory and cardiovascular diseases. In recent years, advances in machine learning and data analysis techniques have opened up new possibilities for accurate and efficient prediction of air pollution. In particular, using deep learning models like Long Short-Term Memory (LSTM) neural networks has shown promise in capturing the complex temporal patterns and dependencies in air quality data. LSTM models have proven to be highly effective in predicting time series and have been successfully applied in various fields, including finance, weather forecasting and natural language processing.

This paper aims to investigate the use of LSTM neural networks for PM2.5 forecasting, using historical air quality data. The dataset used in this study contains comprehensive records of PM2.5 concentrations over a period of time, offering a wealth of data for modelling training and evaluation. The data are preprocessed to address missing values and normalized to ensure consistent scaling. Then, the LSTM model is built considering the temporal nature of the data and the specific requirements of PM2.5 prediction. The main objective of this study is to evaluate the accuracy and performance of the LSTM model in predicting PM2.5 concentrations. Evaluation metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R2 score are used to quantify the prediction capabilities of the model. In addition, visualizations, including time series plots and graphs of predicted results, will provide a comprehensive understanding of the model's performance and patterns in the PM2.5 data

I selected the Beijing PM2.5 Data Set, a comprehensive dataset that provides valuable insights into the concentration of particulate matter 2.5 (PM2.5) in Beijing, China. The Beijing PM2.5 Data Set contains a comprehensive collection of hourly measurements of PM2.5 concentrations spanning from January 2010 to December 2014. The dataset includes a wide range of variables, including meteorological data such as temperature, air pressure, humidity, wind direction and wind speed, as well as additional contextual information such as date and time. This wealth of information allows me to investigate the complex relationships between PM2.5 levels and environmental factors.

The dataset comes from a network of air quality monitoring stations spread across different locations in Beijing. These monitoring stations use advanced sensors and instruments to measure the concentration of PM2.5 particles in real time, providing a comprehensive and accurate representation of air pollution in the city. Using this dataset, I aim to develop an LSTM model that is able to effectively predict PM2.5 concentration based on available meteorological and temporal information.

# METHODS

## Activation functions

The output of a neuron or layer in a neural network is subjected to a mathematical function known as an activation function. It gives the network non-linearity, enabling it to understand and express intricate correlations in the data. Based on the weighted total of a neuron's inputs, activation functions determine the output of the neuron. A neural network may learn and approximatively recognize non-linear patterns in the input data by using an activation function to apply non-linear modifications to the data. If activation functions weren't there, neural networks would only be able to represent linear connections between the input and output.

The three layers of my LSTM model employ three distinct activation functions, for a total of three activation functions [1]. By adding non-linearity to the network, activation functions provide the model the ability to recognize and learn complicated temporal correlations in the input. enabling it to recognize and learn intricate patterns in the data.

An activation function that is frequently employed in neural networks is the sigmoid activation function, sometimes called the logistic function. It's characterized as:

In many neural networks, the Rectified Linear Unit (ReLU) is a frequent activation function, known for its simplicity and effectiveness. The ReLU function is defined as:

The Hyperbolic Tangent (tanh) is an activation function commonly used in neural networks [2]. It is a non-linear function that squashes the input values to a range between -1 and 1. The tanh function is defined as:

My LSTM model can successfully detect and depict complicated patterns and dependencies in the input data by combining various activation functions. The sigmoid function controls gate behavior within the LSTM cell, the ReLU function creates non-linearity in the final output layer, and the tanh function allows the LSTM layers to simulate temporal relationships.

## Recurrent Neural networks

Recurrent neural networks are a particular type of neural network design (RNN) is made to efficiently process sequential data. RNNs, as opposed to feedforward neural networks, have feedback connections that let them store and use data from earlier time steps. Their suitability for problems involving sequences, such language modelling, speech recognition, and time series analysis, is as a result.

The ability of RNNs to keep a hidden state or memory, which is updated at each time step and affects how following inputs are processed, is their important attribute. As a form of short-term memory, this hidden state enables the network to identify dependencies and patterns in the sequential input. Traditional RNNs, however, have trouble learning from lengthy sequences. The vanishing gradient issue is one of the key issues. Gradients can either burst or disappear exponentially as they spread through several layers when they are back propagated through time. This problem results from the gradients' repeated multiplication during the backward pass, which might provide gradient values that are either very small or very large.

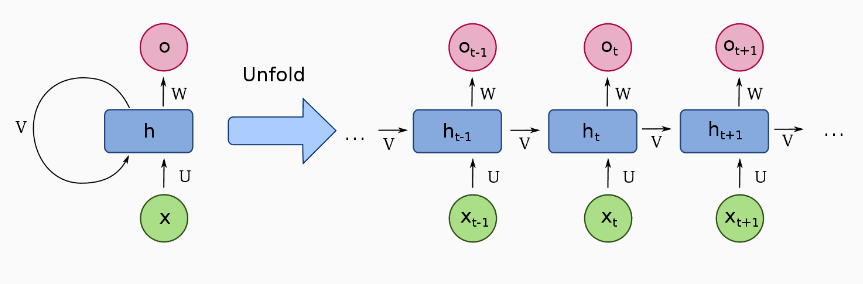


Fig. 1. Cell structure for RNN [3]

## Long Short-Term Memory

Long Short-Term Memory, often known as LSTM, is a sort of recurrent neural network (RNN) architecture made specifically to handle and model sequential input. It was developed to overcome typical RNNs' shortcomings in identifying long-term dependencies and avoiding the vanishing gradient issue. The network can store and access information for extended periods of time thanks to the LSTM design, which is based on the idea of a memory cell. The ability of LSTM to selectively keep or forget information using specialized gates is its primary characteristic. These gates control the information flow inside the network, enabling it to learn and retain pertinent information while eliminating redundant or irrelevant data.

Three primary gates typically make up the LSTM cell structure. The Forget Gate selects the data from the previous cell state that should be deleted or forgotten. It generates a forget vector that regulates the quantity of information to be forgotten by taking the previous hidden state and the current input as inputs. The input gate selects the fresh data that should be kept in the memory cell. It processes the current input and the prior hidden state using a sigmoid activation function to produce an input vector. A new candidate cell state is produced using this input vector and a tanh activation function. Output Gate: It regulates how much data is made available as the current cell's output or hidden state. It takes the previous hidden state and current input as inputs, processes them, and outputs an output vector.

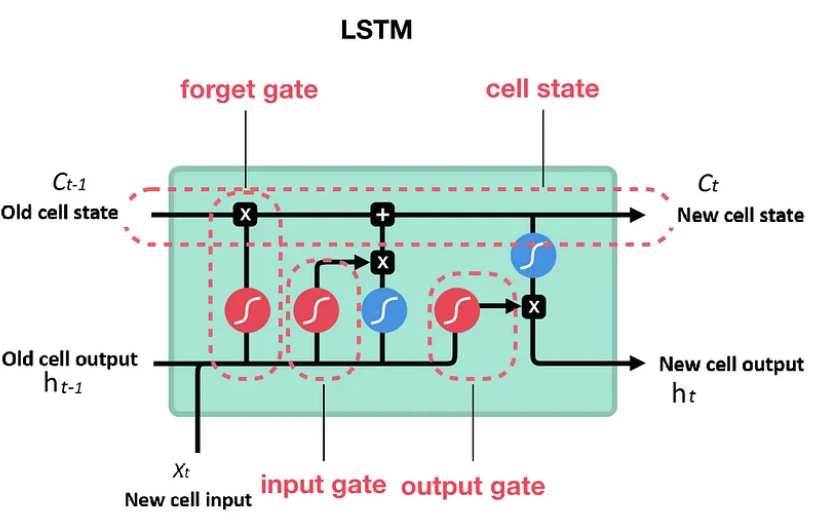


Fig. 2. Cell structure for LSTM [4]

These gates enable LSTM cells to capture long-term dependencies in the data, prevent the vanishing gradient problem, and selectively retain relevant information across lengthy sequences. By preserving an internal memory state that can store pertinent data from earlier time steps, the feedback connections in the LSTM design allow the network to interpret sequential data. This qualifies them for jobs requiring the processing and modelling of sequential data. LSTMs have been widely used in a variety of fields, including time series forecasting, speech recognition, and natural language processing, where a grasp of long-term dependencies is essential for making precise predictions.

## Evaluation of Time Series Models

For predicting future values based on historical data, time series models are frequently utilized. There are many evaluation metrics that are frequently used to evaluate the performance of these models. The most popular evaluation metrics for time series forecasting are covered in this essay, including coefficient of determination (R2 score), mean absolute percentage error (MAPE), mean squared error (MSE), the root mean squared error (RMSE), and the mean absolute error (MAE) [5].

These evaluation measures offer insightful information about the effectiveness of time series models. They make it possible to compare various models, pick the best model, and assess the precision and dependability of the forecasts. When selecting the best evaluation metric, it is crucial to take into account the unique traits of the time series and the nature of the issue. A time series model's accuracy and performance are evaluated using a variety of measures, including MSE, MAE, RMSE, R2 score, and MAPE. These indicators offer important details about the model's capacity for precise forecasting and can help with forecasting task decision-making. The selection of an evaluation metric should take into account the unique requirements and properties of the time series data.

# Experiment

Along with other pertinent variables that may have an impact on air pollution, the dataset offers a time series of PM2.5 concentration. It is frequently used for forecasting, time series analysis, and researching the connection between air pollution and weather patterns. The program's goal is to use an LSTM model to forecast time series data for the "pm2.5" variable. I may learn more about the variables influencing PM2.5 concentration, spot patterns and trends over time, and create prediction models for predicting future pollution levels by analyzing and modelling this dataset.

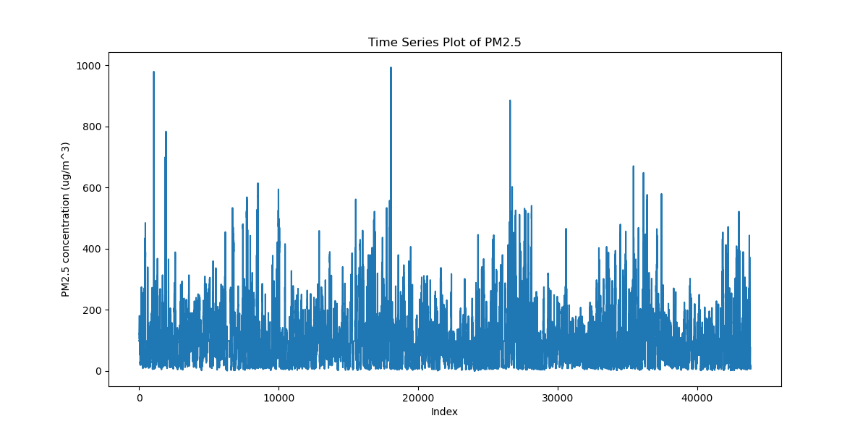


Fig. 3. PM2.5 Time Series Plot

To see the changes in PM2.5 concentration over time, I created a time series plot. The graphic gave a complete picture of the PM2.5 concentrations, enabling comprehension of trends, patterns, and oscillations. I was able to see how the PM2.5 concentration changed over the course of the dataset by plotting the PM2.5 values on the y-axis and the corresponding time points on the x-axis. For examining the temporal behavior of PM2.5 and learning more about its dynamics, this figure was a helpful tool.

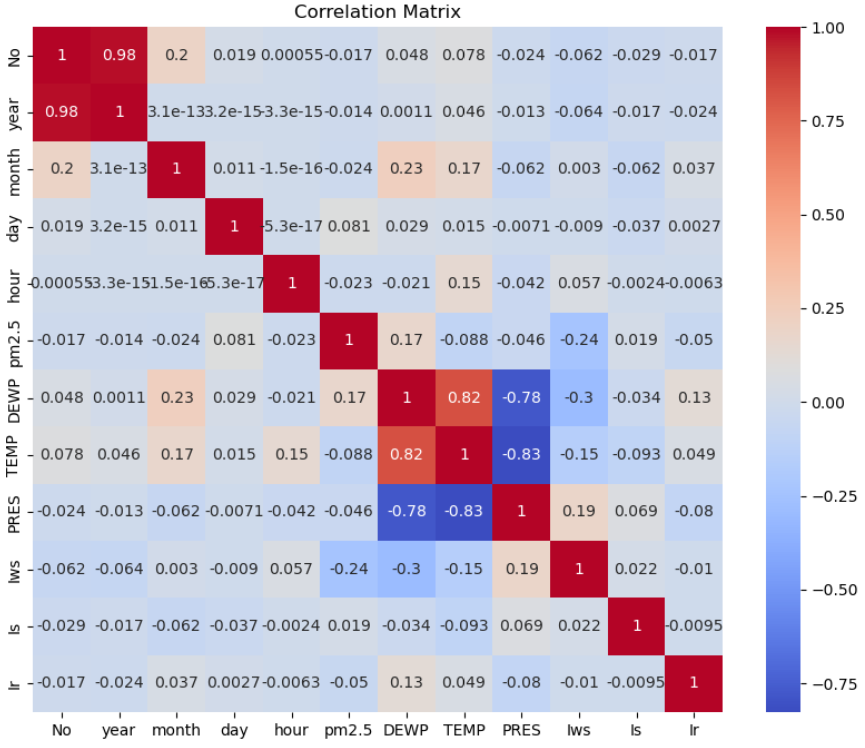


Fig. 4. Time Series Plot of PM2.5

I created a correlation matrix to look at how the various variables in the dataset related to one another. To better understand the interdependencies between the variables, the correlation matrix gave a visual depiction of the pairwise correlations between them. I added color gradients to the correlation values using a heatmap visualization, where warmer colors denoted stronger positive correlations and cooler colors denoted stronger negative correlations. This gave me vital information about the relationships between the variables in the dataset and their potential effects on the target variable by allowing me to discover potential correlations and dependencies between them.

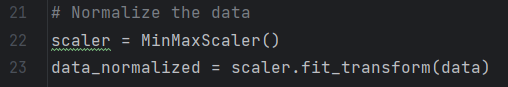


Fig. 5. Normalize the data

I performed the normalization in this research using the MinMaxScaler from the scikit-learn module. The features in the dataset must all be on a comparable scale during this preprocessing stage, which commonly ranges between 0 and 1. I converted the results to this standard range by using the Min-Max scaler on the dataset.

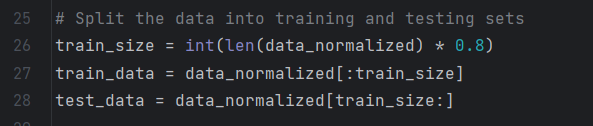


Fig. 6. Split of Dataset

The training set and the testing set were two separate subsets of the dataset. This division was a critical stage in determining the LSTM neural network's effectiveness and generalizability. Using a sequential splitting method, the temporal order of the data was preserved. The time variable prior to the divide was used to chronologically order the dataset. By testing the model with upcoming, unobserved data and modelling real-world circumstances, this made sure that it learned from prior observations. For this experiment, a splitting ratio of 80:20 was used, wherein 20% of the data were provided to the testing set, and the remaining 80% to the training set. This ratio achieves a balance between having enough data to train the model and having just enough to judge how well it performs in the presence of unknown data.

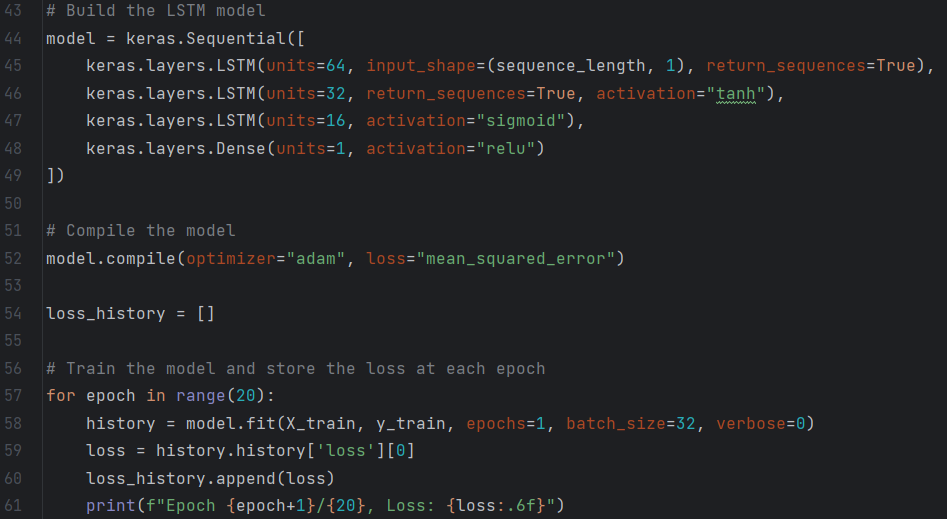


Fig. 7. Build and Train model

The Keras library is used to build and compile the LSTM model during the training phase. Three LSTM layers of 64, 32, and 16 units each make up the model architecture, which is followed by a Dense layer with one unit. Three different activation functions are used in the LSTM model architecture to add non-linearity and improve the model's ability to learn complex patterns. The Adam optimizer, a well-liked option for gradient-based optimization techniques, is used to optimize the model [6]. Using the mean squared error loss function, the discrepancy between predicted and actual data is calculated.

A complete pass through the training data is represented by each of the 20 epochs that make up the training process. The fit function is used to train the model at the beginning of each epoch. The model adjusts its weights based on the calculated loss after batching the training data into 32-piece chunks. The loss\_history list, which enables tracking of the model's training progress, stores the loss value for each epoch. The model gradually learns and modifies its internal parameters as it cycles through the epochs in order to minimize the loss function and enhance its capacity for precise prediction. A model that is more closely fitted to the training set of data has a lower loss value.

The predicted values generated from the model were used to calculate several performance metrics in order to evaluate the precision and dependability of the LSTM model in forecasting PM2.5 concentrations.

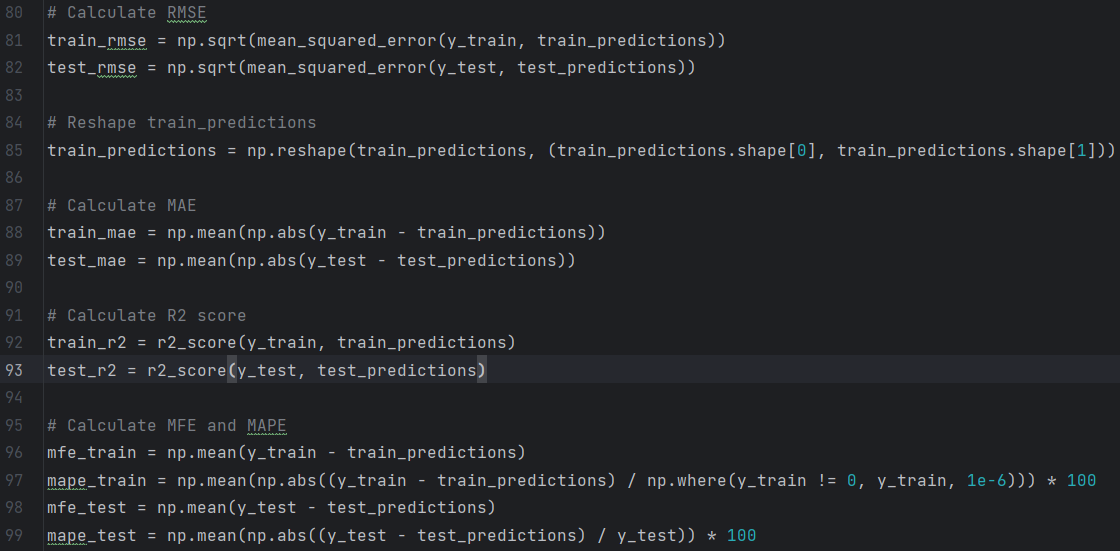


Fig. 8. Evaluation model prediction results

First, the Root Mean Squared Errors (RMSEs) were calculated. The average size of variations between expected and actual values is measured by the RMSE. to assess the Mean Absolute Error (MAE), which is the typical absolute difference between the expected and actual values. The proportion of the variance in the dependent variable (PM2.5) that can be explained by the independent variables (predicted values) was also determined using the R-squared (R2) score. Also computed were the Mean Forecast Error (MFE) and Mean Absolute Percentage Error (MAPE). The MAPE gauges the average amount of separation between the expected and actual values relative to the actual values, whereas the MFE quantifies the typical discrepancy between expected and actual numbers.

These performance indicators shed important light on the precision and dependability of the LSTM model's PM2.5 concentration predictions. These metrics provide for a thorough assessment of the model's performance by quantifying the errors and differences between the anticipated and actual values.

# RESULTS

The LSTM model is trained utilizing the training data for a total of 20 epochs during the training phase. A full iteration across the entire training dataset is represented by each epoch. To reduce the loss function, the model learns from the input sequences (X\_train) and the corresponding target sequences (Y\_train).The model modifies its internal weights in each epoch by utilizing the Adam optimizer to improve the mean squared error loss function. The loss value is calculated and recorded in the loss\_history list after each epoch. A line plot of the loss history is created in order to see how the training is progressing.

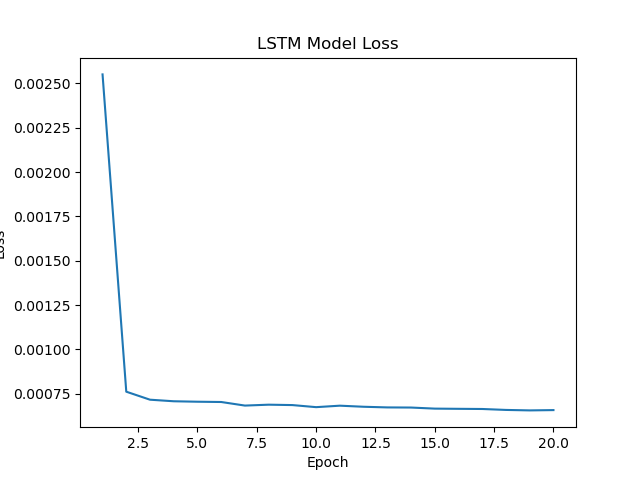


Fig. 9. LSTM Model Loss

The model's loss begins at a value of 0.0025 during the initial training phases. The model optimizes its internal weights and gains knowledge from the training data as the training goes on. In about 2.5 epochs, the loss value is reduced to 0.00075.The model is effectively modifying its parameters to reduce the difference between the anticipated outputs and the actual target values, as seen by the diminishing loss. The fact that the loss is declining suggests that the model is performing better predictions and identifying the underlying patterns in the data.

I used the Python Matplotlib module to display the outcomes of the LSTM model's predictions for the PM2.5 levels. The resulting graphic offers a precise depiction of the predicted values and enables a thorough evaluation of the model's effectiveness. Each line in the plot is carefully labelled to aid comprehension. For both the training and test datasets, the legend helps to distinguish between the actual and predicted values.

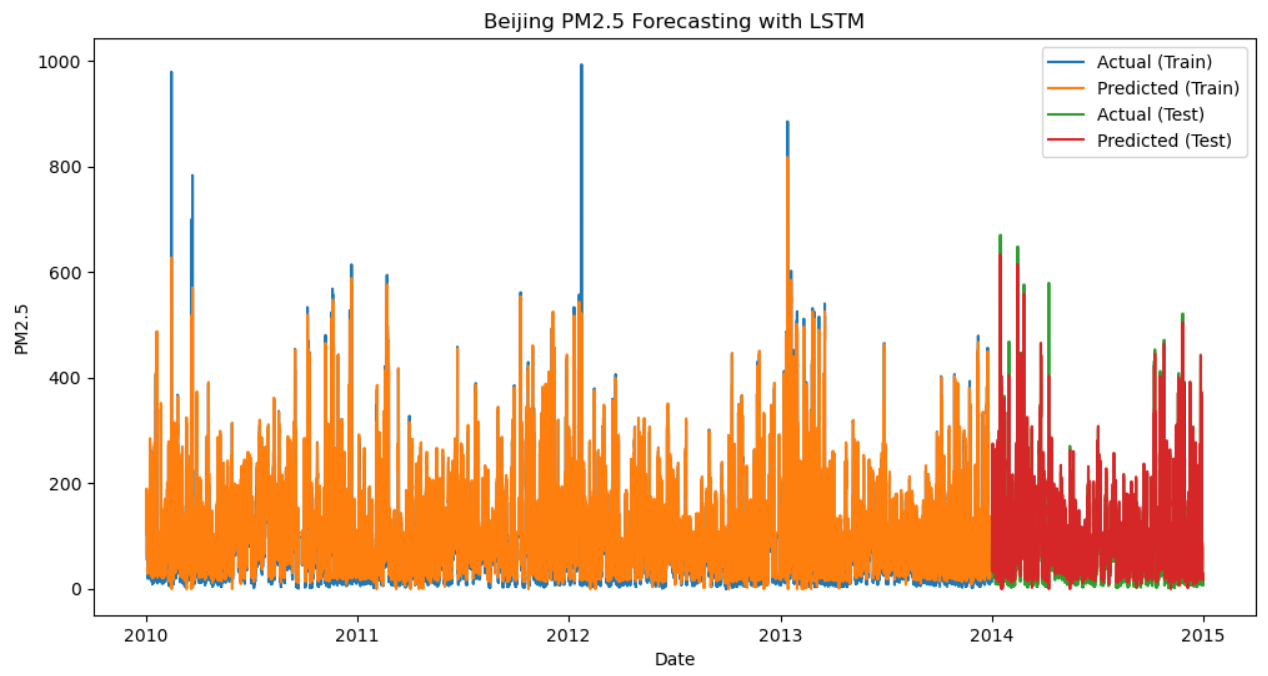
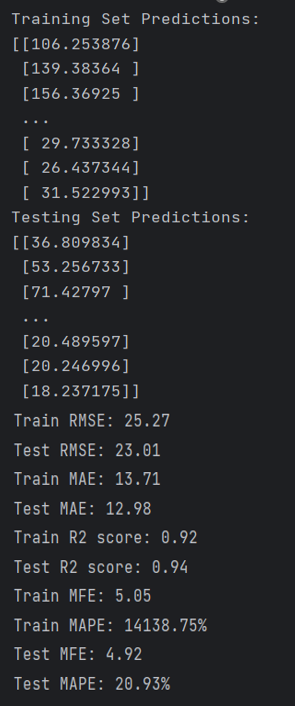


Fig. 10. PM2.5 prediction results of LSTM

The graphic, which has a figure size of 12 by 6, shows the changes in PM2.5 readings over time. Dates are shown on the x-axis, while the PM2.5 concentration is shown on the y-axis. Four unique lines that each reflect a different facet of the data make up the graphic. The training dataset's actual PM2.5 values are shown in the first line as a basis for comparison. The projected PM2.5 values for the training set, as produced by the LSTM model, are shown in the second line. These two lines make it possible to visually assess how well the predictions of the model match the actual values in the training dataset. The graphic then contains two more lines to reflect the test dataset's actual and anticipated PM2.5 levels. By comparing these predictions to the actual values in the test dataset, it is possible to assess the LSTM model's generalization capacity.

I ran a residuals visualization to evaluate how well the LSTM model performed in forecasting PM2.5 concentrations. The discrepancies between the actual values of PM2.5 and the values that the model anticipated are known as residuals. This study aids in our comprehension of how effectively the model captures the data's leftover information that was left out of the predictions. I made a visualization in Matplotlib to show the residuals for the training and test datasets. Dates are represented on the x-axis, and residuals are represented on the y-axis. The "Residuals Plot" plot has two lines that indicate the residuals for both the training and test sets of data, respectively: "Residuals (Train)" and "Residuals (Test)". The legend aids in separating the two lines.

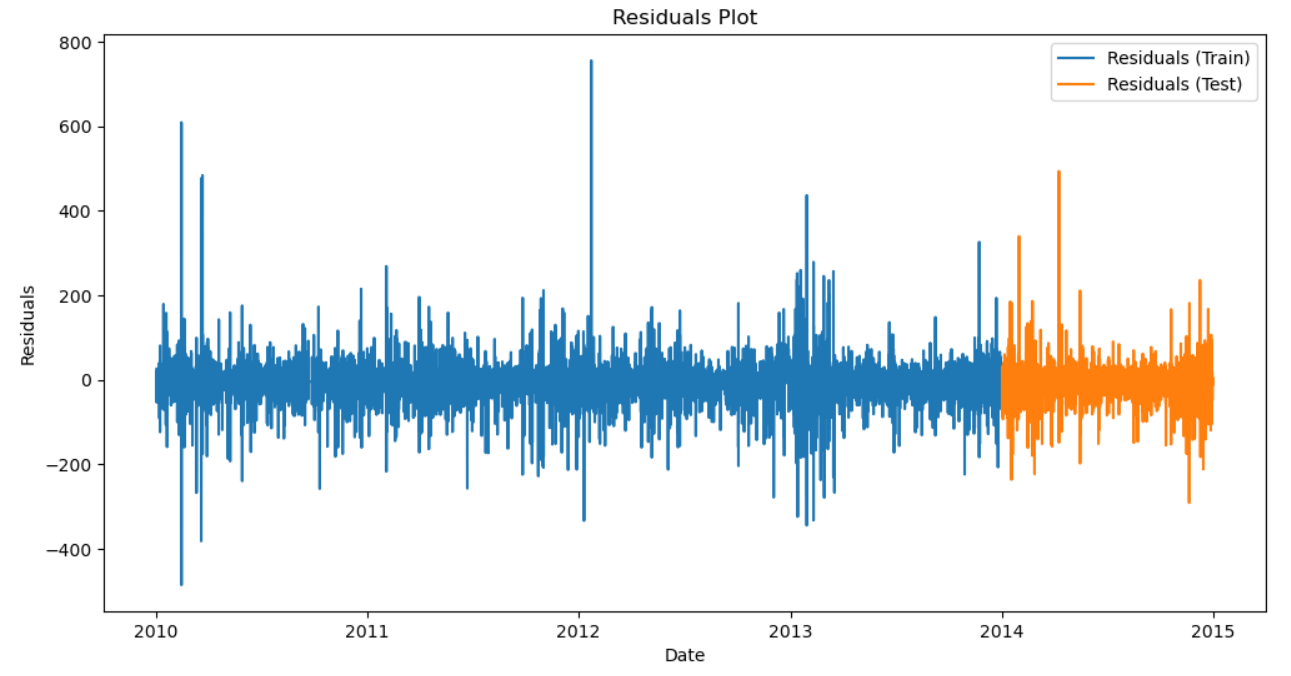


Fig. 11. Residuals Plot

I can spot any patterns or trends in the variations between the expected and real values by looking at the residuals plot. If the model has successfully captured the variability in the data, the residuals should ideally display a random pattern near zero. However, it indicates that the model might be missing certain parts of the data if there are obvious patterns or trends in the residuals. I may assess the overall effectiveness and dependability of the LSTM model in forecasting PM2.5 concentrations by analyzing the residuals. I can decide whether the model needs any additional tweaks or changes by analyzing the patterns and trends in the residuals.

The effectiveness of the LSTM model on the training and testing sets is output by this code block. Lower values signify greater performance. The RMSE and MAE metrics show the average magnitude of deviations between anticipated and actual values. Better fit is indicated by higher values. The R2 score measures the proportion of the dependent variable's variation that can be accounted for by the independent variable. The average difference and percentage difference between the expected and actual values are also measured by the MFE and MAPE metrics.

In-depth evaluation of the LSTM model's performance on the Beijing PM2.5 dataset is provided in this block. These findings provide a thorough assessment of the LSTM model's PM2.5 concentration prediction abilities, offering important details on its precision and dependability. The LSTM model looks to perform well on both the training and testing sets, with relatively low errors and high R2 scores, according to these evaluation metrics.

Fig. 12. Forecast and evaluation results

# CONCLUSION

In-depth time series data analysis utilizing a Python Long Short-Term Memory (LSTM) network was provided in this essay. I used an organized approach for the entire investigation, which included data preprocessing, model building, training, evaluation, and performance analysis. The dataset, which concentrated on air pollution levels (more specifically, the "pm2.5" variable), was treated with care by filling in any missing values and formatting the data appropriately.

Three hidden layers and various activation functions were incorporated into the LSTM model's construction to increase its capacity for prediction. The model was then trained using the mean squared error loss function and the Adam optimizer. The loss at each epoch, which continuously decreased over time, was used to track the training process. The model's examination demonstrated how well it predicted air pollution levels. Techniques for exploratory data visualization were also used in the analysis. The correlations between the variables were discovered using a correlation matrix, and the time series plot of the "pm2.5" values revealed information about the historical trend of air pollution. Line graphs were also used to visually contrast the model's predictions with the actual values, showing how well the model performed on both the training and testing sets.

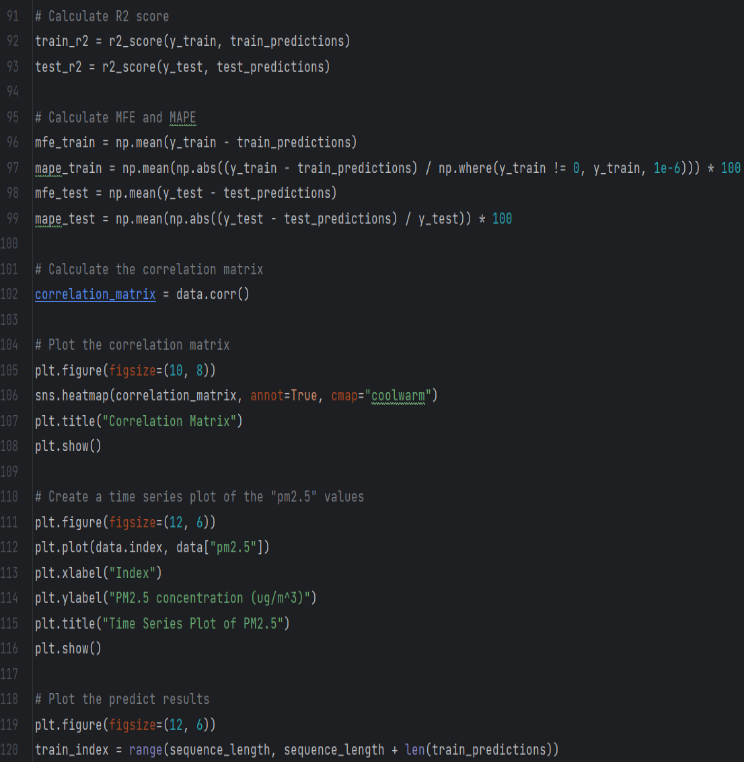
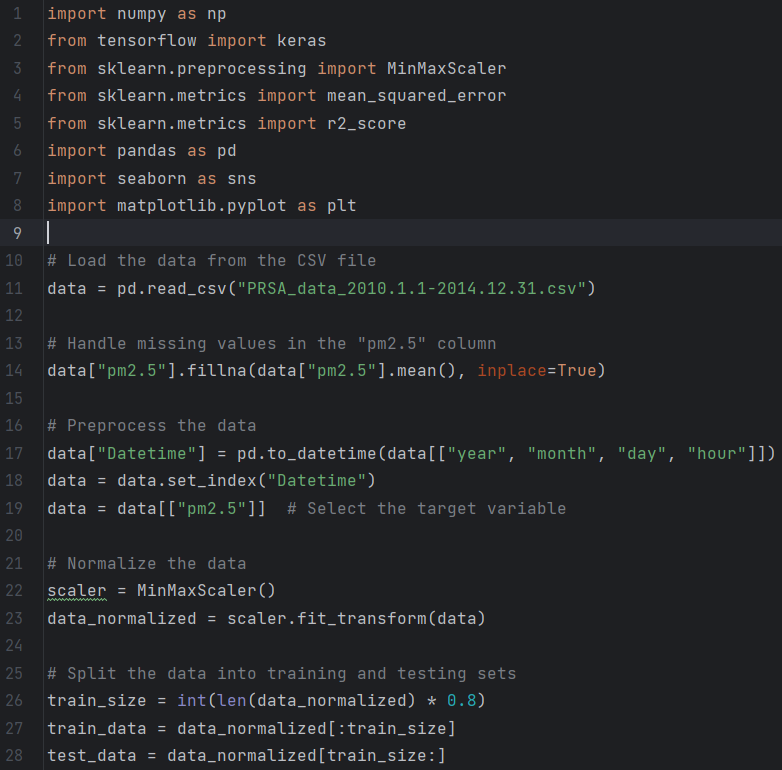
I have noted a number of factors for potential future development of the LSTM model for time series forecasting. I will keep up with the most recent developments in deep learning methods. Investigating cutting-edge methods may result in further advancements in forecasting precision and uncertainty estimation. I can make sure that my model stays at the top of the field by keeping up with these developments.

Another crucial factor to think about is the model's interpretability. To learn more about the most important features or time intervals, I will investigate techniques like attention maps and gradient-based attribution methods. This will help me comprehend and explain the logic behind the model's predictions better, building trust and easing the model's adoption in practical applications. I will modify the model to take streaming data and generate predictions in an online manner if real-time forecasting is necessary. To achieve precise and timely predictions, methods such as online learning or window-based update of the model's parameters will be investigated.

I want to improve the LSTM model for time series forecasting by taking into account these potential improvement strategies. I anticipate enhancing the model's interpretability, accuracy, and applicability in real-world circumstances through these efforts. The LSTM model for time series forecasting can be enhanced to obtain higher accuracy, improved interpretability, and increased application in real-world scenarios by taking these factors into account for future improvement.

##### References

1. Sharma, S., Sharma, S. and Athaiya, A., 2017. Activation functions in neural networks. *Towards Data Sci*, *6*(12), pp.310-316.
2. Baldwin, D., Göktaş, Ü. and Hereman, W., 2004. Symbolic computation of hyperbolic tangent solutions for nonlinear differential–difference equations. *Computer Physics Communications*, *162*(3), pp.203-217.
3. Coding Ninjas. (n.d.). Understanding an RNN Cell. [online] Available at: https://www.codingninjas.com/codestudio/library/understanding-an-rnn-cell [Accessed 15 May 2023].
4. Humble Bee. (n.d.). RNN - Recurrent Neural Networks (LSTM). [online] Medium. Available at: https://medium.com/@humble\_bee/rnn-recurrent-neural-networks-lstm-842ba7205bbf [Accessed 15 May 2023].
5. Lakshminarayanan, S.K. and McCrae, J.P., 2019, December. A Comparative Study of SVM and LSTM Deep Learning Algorithms for Stock Market Prediction. In *AICS* (pp. 446-457).
6. Chandriah, K.K. and Naraganahalli, R.V., 2021. RNN/LSTM with modified Adam optimizer in deep learning approach for automobile spare parts demand forecasting. *Multimedia Tools and Applications*, *80*(17), pp.26145-2615

APPENDIX



