# Prediction of $PM_{2.5}$ level at an AQMD station
## : using nearby stations' data

Nitish Venkatesh Septankulam Ramakrishnan, Meghana Vasanth Shettigar, Jooseok Lee
December 12, 2022

# 1. Problem Space

Predicting the level of particulate matter ($PM_{2.5}$) is becoming increasingly crucial in the field of public health. For example, the United States Environmental Protection Agency (US EPA) designates particulate matter as an important source of air pollution [1]. Accordingly, many air pollution observatory stations are set up to monitor the level of $PM_{2.5}$. However, not every station is able to measure the level of $PM_{2.5}$. For instance, only 13 of 30 stations in South Coast Air Quality Management District (AQMD) can monitor the level of $PM_{2.5}$. In this project, we aim to develop a machine learning model that can predict the level of $PM_{2.5}$ of stations that have no $PM_{2.5}$ level monitoring capability. To achieve that, we build a machine learning model that predicts the $PM_{2.5}$ level of a target station using $PM_{2.5}$ level data of nearby stations. We also utilize other commonly monitored air pollution data, such as level of $NO_2$ and CO, and basic meteorological data, such as wind direction and speed, to increase the performance of the model. The development direction is based on the assumption that the air quality of one station has a spatiotemporal correlation to the air quality of nearby stations [2].

# 2. Dataset

## *2.1. Data Source*

We utilize historical $PM_{2.5}$ and other air pollution data from AQMD (Link). AQMD offers historical data of various components including the level of $PM_{2.5}$, CO, wind speed, etc. for monitoring stations in California.

## *2.2. Train and Test Dataset*

In this project, we use data from Jan. 17$^{th}$, 2017 to Jan. 16$^{th}$ 2022. Each row represents an hourly measurement of various air pollutants and the total number of rows is 33,520. To avoid data leakage in the test dataset, we split data into train and test dataset according to the time sequence. The train dataset is from Jan. 17$^{th}$, 2017 to Jan. 10$^{th}$, 2021 and the test dataset is from Jan. 17$^{th}$, 2021 to Jan 16$^{th}$, 2022.

## *2.3. Target and Nearby Stations*

We select Mira Loma station as our target station and five other stations (i.e. Upland, LakeElsinore, Temecula, Banning, and Central San Bernardino stations) as nearby stations. The selection criteria is the number of missing values in $PM_{2.5}$ and proximity).

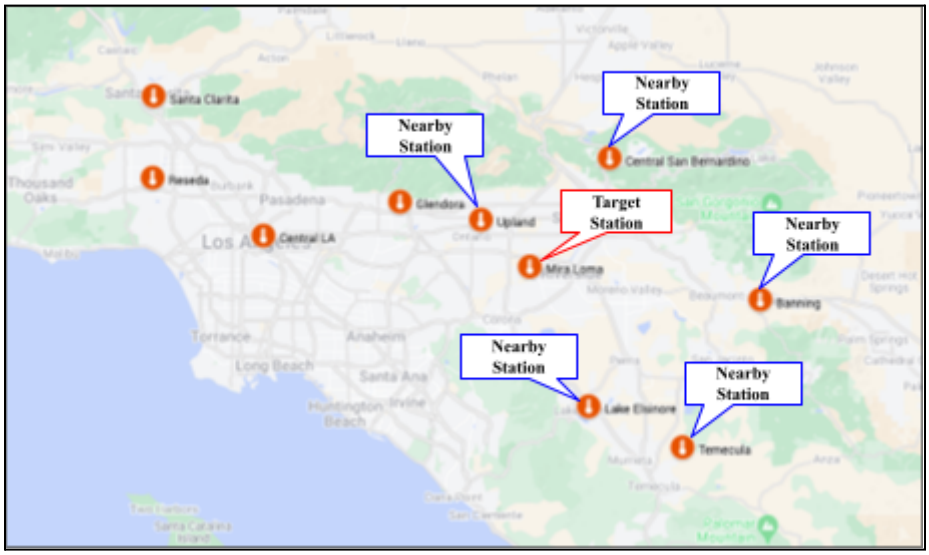


Figure 1: Target and Nearby Stations

# 3. Approach

## *3.1. Data Preprocessing*

We handle missing values with an interpolation technique supported by pandas library. It estimates missing values using adjacent data. In this project, we use spline interpolation to fill missing values. We also clip $PM_{2.5}$ data to handle outliers. Figure 2 shows the snapshot of data after preprocessing. After the preprocessing, we also standardize the data.

| Date Time | CO | O3 | WD | NO2 | T | WS | Upland_PM25 | Lake_Elsinore_PM25 | Temecula_PM25 | Banning_PM25 | Central_San_Bernardino_PM25 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2021-01-09 23:00:00 | 1.0 | 43.0 | 344.0 | 2.0 | 64.0 | 18.0 | 11.833333 | 7.0 | 8.0 | 11.0 | 14.0 |
| 2021-01-10 00:00:00 | 1.0 | 35.0 | 23.0 | 7.0 | 60.0 | 7.0 | 11.916667 | 7.0 | 11.0 | 14.0 | 14.0 |
| 2021-01-10 01:00:00 | 1.0 | 36.0 | 14.0 | 6.0 | 60.0 | 8.0 | 12.000000 | 6.0 | 11.0 | 12.0 | 16.0 |
| 2021-01-10 02:00:00 | 1.0 | 29.0 | 57.0 | 10.0 | 57.0 | 6.0 | 12.000000 | 5.0 | 9.0 | 9.0 | 17.0 |
| 2021-01-10 03:00:00 | 1.0 | 33.0 | 47.0 | 6.0 | 57.0 | 5.0 | 12.000000 | 5.0 | 8.0 | 8.0 | 16.0 |

Figure 2: Snapshot of data

We aim to build machine learning models that predict the PM2.5 level after six steps (i.e. after six hours) using the previous seven days' data (i.e. 168 hours of past data). To achieve this, we add an additional preprocessing step described in Figure 3 to change the shape of data into the desired one.

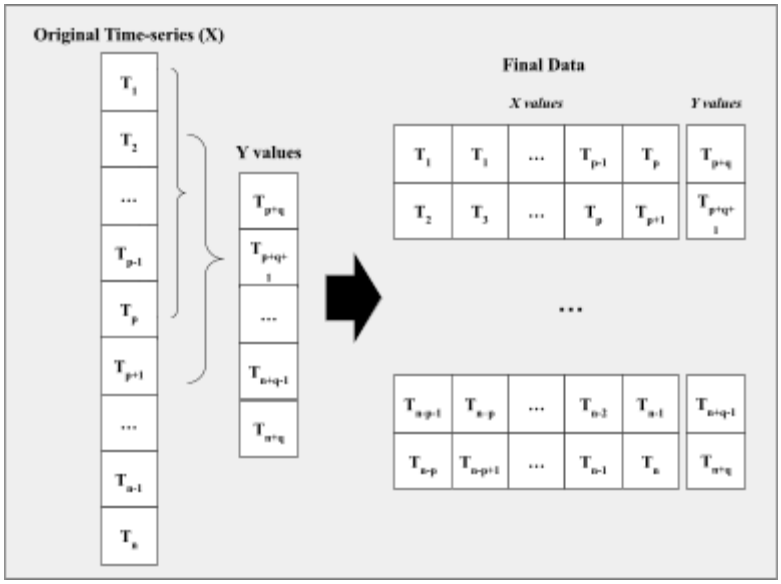


Figure 3: Additional preprocessing step

## *3.2. $PM_{2.5}$ Prediction*

In this project, we aim to develop a machine learning model that can predict the level of $PM_{2.5}$ of stations that have no $PM_{2.5}$ level monitoring capability. In order to do this, we build three types of machine learning models (i.e. CNN, LSTM, and XGBoost) that take nearby stations' data as input and predict the target station's $PM_{2.5}$ level. Figure 4 shows the illustrative representation of our approach.

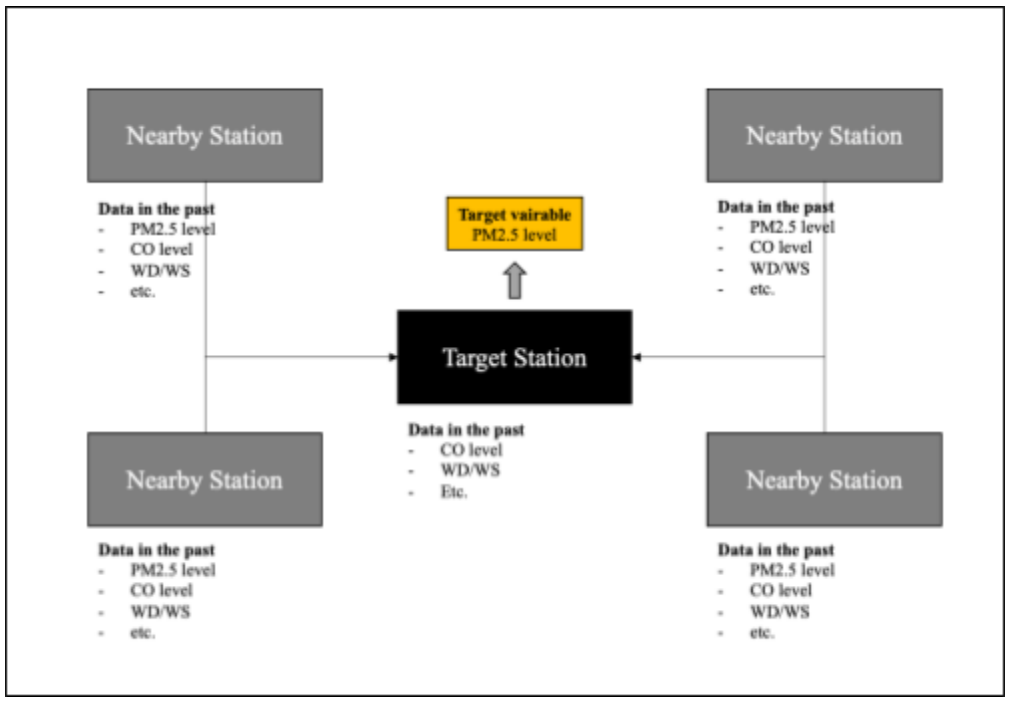


Figure 4: Illustration of approach

### *3.2.1. Model 1: Convolutional Neural Network (CNN)*

This model has one 1D convolutional layer. It has a kernel size of (24,) and 12 channels. The batch normalization is done after the convolutional layer followed by RELU activation. Then, the AveragePooling1D layer and Flatten layer are applied to the model. Finally, two Dense layers, Dense(16) and Dense(8) respectively, are applied to produce the final output. Adam optimizer with learning rate 0.001 and MSE loss function are used to train the model. The CNN structure we used in this project is initially inspired by the work in [3] and is selected after trying several different options, such as increasing the number of 1D convolutional layers, changing the size of kernel and channels, or changing the learning rate.

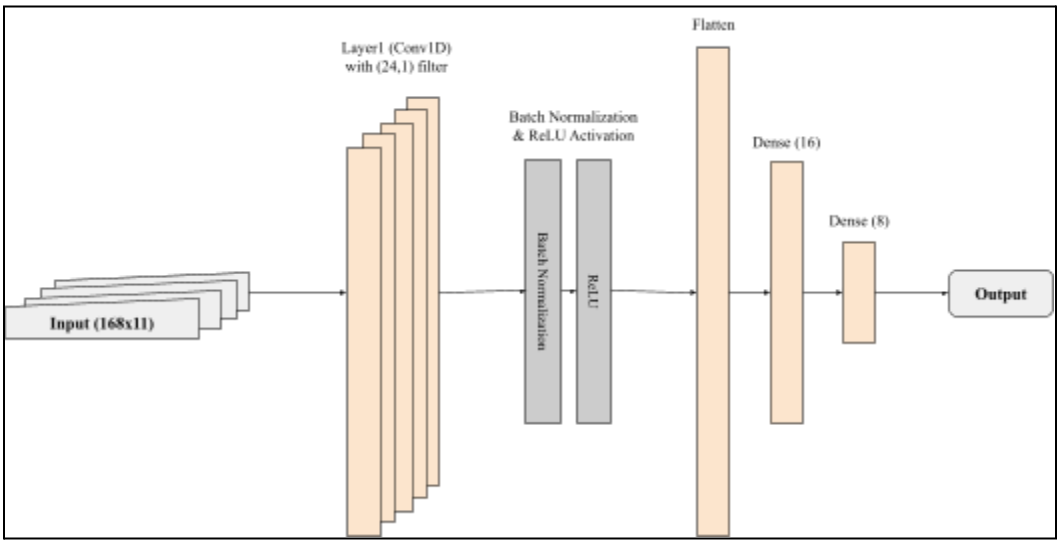


Figure 5: CNN Architecture

### *3.2.2. Model 2: Long Short-Term Memory Networks (LSTM)*

This model has one LSTM layer that has 24 nodes. The LSTM layer is then connected to two Dense layers, Dense(16) and Dense(8) respectively, to produce the final output. Adam optimizer with learning rate 0.001 and MSE loss function are used to train the model. The LSTM structure we used in this project is initially inspired by the work in [4] and is selected after trying several different options, such as increasing the number of LSTM layers, changing the number of Dense layers, or changing the learning rate.

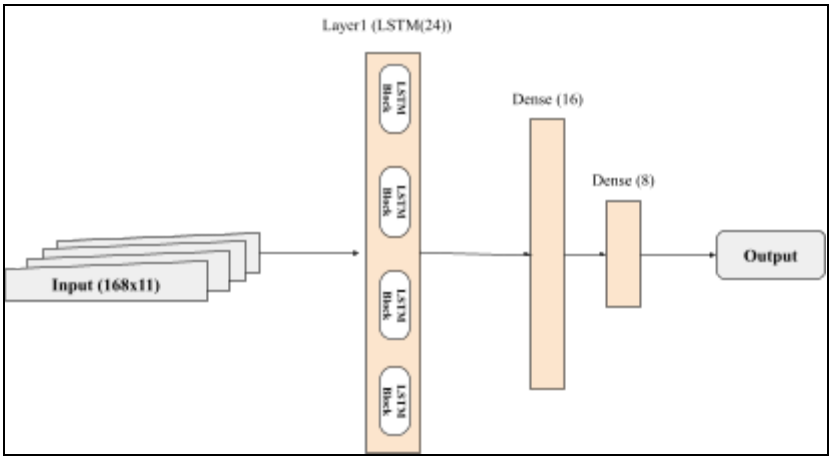


Figure 6: LSTM Architecture

### *3.2.3. Model 3: XGBoost*

Unlike the previous two models, this model unrolled the input data to change the shape of it suitable for XGBoost. Then PCA is carried out to reduce the number of input features from 1,848 to 150. Default parameter tuning is used to train the model. We tried several hyper-parameters such as the number of reduced input features, max depth of trees, and sampling rate of columns to select the final model.

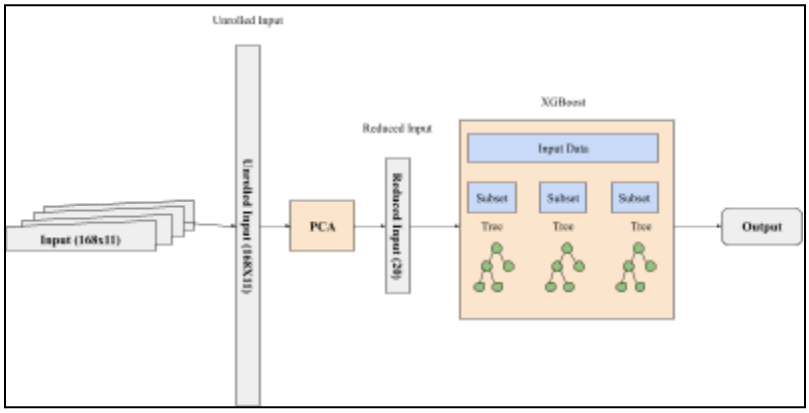


Figure 7: XGBoost Architecture

### *3.2.4. Evaluation Metrics and Base Model*

We use three kinds of evaluation metrics, namely mean absolute error (MAE), mean squared error (MSE), and total accuracy index (P) used in [2], to compare the performance of the models. The formal definition of each metric is given below.

$$MAE = \frac{1}{n}\sum^{n}\left|y_i - \hat{y}_i\right|$$

$$MSE = \frac{1}{n}\sum^{n}(y_i - \hat{y}_i)^2$$

$$p = 1 - \frac{\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|}{\sum_{i=1}^{n} y_i}$$

Figure 8: Evaluation Metrics

In addition, we build a base model for each model we developed to estimate the effectiveness of them. Each base model literally has the same architecture with the developed model but has an additional input, $PM_{2.5}$ level of the target station. By comparing the performance of the developed model with its corresponding base model, we can gauge the relative usefulness of the model we developed.

| No | Main model | Base model |
|---|---|---|
| 1 | CNN model *without* target station's $PM_{2.5}$ data | CNN model *with* target station's $PM_{2.5}$ data |
| 2 | LSTM model *without* target station's $PM_{2.5}$ data | LSTM model *with* target station's $PM_{2.5}$ data |
| 3 | XGBoost *without* target station's $PM_{2.5}$ data | XGBoost *with* target station's $PM_{2.5}$ data |

Table 1: Models developed and corresponding base models

# 4. Results

## *4.1. Comparison between Models*

Overall, the performance of base models, which use the target station's $PM_{2.5}$ data, show better performance in terms of three metrics, namely MSE, MAE, and Total Accuracy Index. However, the performance of main models, which do not use the target station's $PM_{2.5}$ data, show comparable performance indicating the possibility of the approach we proposed in this project, which is using nearby stations' data to predict the target station's $PM_{2.5}$ level. One limitation is that the overall performance of models is pretty low, which will be discussed in the discussion section below. Among three algorithms we used, CNN with 1D convolutional layer showed the best performance.

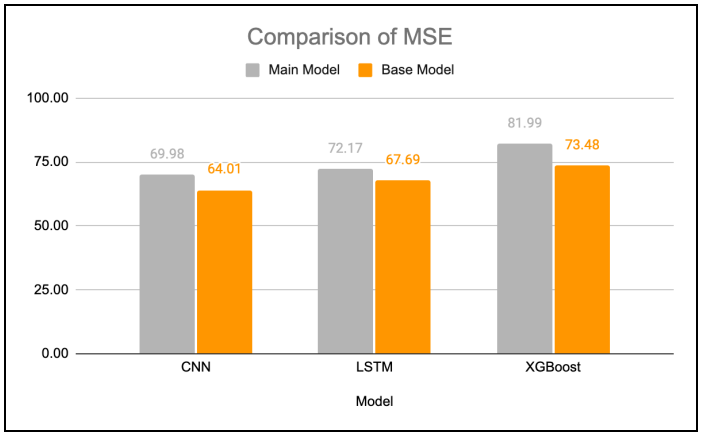


Figure 9: Comparison between Models (MSE)

| | MSE | | MAE | | Total Accuracy Index | |
|---|---|---|---|---|---|---|
| | Main Model | Base Model | Main Model | Base Model | Main Model | Base Model |
| **CNN** | 69.98 | 64.01 | 5.57 | 5.38 | 0.7085 | 0.7284 |
| **LSTM** | 72.17 | 67.69 | 5.50 | 5.42 | 0.7111 | 0.7187 |
| **XGBoost** | 81.99 | 73.48 | 6.07 | 5.86 | 0.6847 | 0.6320 |

Table 2: Performance comparison between models

## *4.2. Error Analysis*

We conducted additional error analysis to gain more insight into the problem. As shown in figure 10, the error goes up as the real $PM_{2.5}$ level goes up. It seems that it is mainly due to the lack of data in higher levels of $PM_{2.5}$. As a result, the model we developed failed to accurately predict high levels of $PM_{2.5}$.

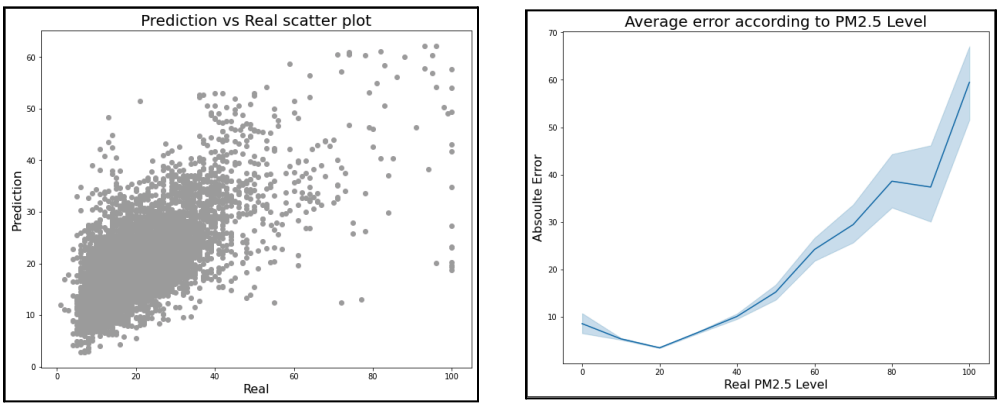


Figure 10: Scatter plot of error & average error according to $PM_{2.5}$ Level

# 5. Discussion

In this project, we aim to build a machine learning model that predicts the level of $PM_{2.5}$ of a target station using nearby stations' $PM_{2.5}$ data. The result showed that machine learning models that do not use the target station's $PM_{2.5}$ data show comparable performance to the models that use the target station's $PM_{2.5}$ data. It showed the possibility of utilizing our approach to estimate the $PM_{2.5}$ level of stations that do not have the capability to measure it. However, there are some limitations that should be overcome. First of all, the overall performance of models we developed showed pretty low performance. The $R^2$ of the best model we developed was 0.4552, which is quite low. According to the result of [2], which showed higher MSE in predicting the level of $PM_{2.5}$ in China, it seems that predicting the future level of $PM_{2.5}$ is quite a complex and difficult problem. To make the proposed approach practical, it is necessary to increase the $R^2$ of prediction models. According to the error analysis, the prediction models seem to show lower performance on high levels of $PM_{2.5}$. We might need to build an additional model specialized in handling higher levels of $PM_{2.5}$. Combining domain knowledge into the model similar to previous works such as [5], [6], [7] is another possible future direction. Finally, generalizing the model that utilizes geographical data will be the ultimate goal of the problem.

# 6. Reference


[1] https://www.epa.gov/pm-pollution

[2] Xiao, F., Yang, M., Fan, H. et al. An improved deep learning model for predicting daily PM2.5 concentration. Sci Rep 10, 20988 (2020). https://doi.org/10.1038/s41598-020-77757-w

[3] https://keras.io/examples/timeseries/timeseries_classification_from_scratch/

[4] https://keras.io/examples/timeseries/timeseries_weather_forecasting/

[5] Minh, V.T.T., Tin, T.T., Hien, T.T. (2021). PM2.5 Forecast System by Using Machine Learning and WRF Model, A Case Study: Ho Chi Minh City, Vietnam. Aerosol Air Qual. Res. 21, 210108. https://doi.org/10.4209/aaqr.210108

[6] Doreswamy, Harishkumar K S, Yogesh KM, Ibrahim Gad, Forecasting Air Pollution Particulate Matter (PM2.5) Using Machine Learning Regression Models, Procedia Computer Science, Volume 171, 2020, Pages 2057-2066, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2020.04.221.

[7] Jiang,X.;Luo,Y.;Zhang,B. Prediction of PM2.5 Concentration Based on the LSTM-TSLightGBM Variable Weight Combination Model. Atmosphere2021,12,1211. https://doi.org/10.3390/atmos12091211