Air Pollution Monitoring and Prediction using IOT and Machine Learning

Ssneha Balasubramanian
Sri Sivasubramaniya Nadar College of Engineering
2 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Talapala Sneha
Sri Sivasubramaniya Nadar College of Engineering
2 PUBLICATIONS **2** CITATIONS
SEE PROFILE

ISSN:0975-9646

Air Pollution Monitoring and Prediction using IOT and Machine Learning

Ssneha Balasubramanian[1], Talapala Sneha[2], Vinushiya B[3], Saraswathi S[4]
Computer Science and Engineering, SSN College of Engineering
Kalavakkam, Chennai, India
ssneha17167@cse.ssn.edu.in

Abstract— **Air pollution is the presence of substances in the atmosphere that are harmful to the health of humans and other living beings or that can cause damage to the climate or to materials. Soot, smoke, mold, pollen, methane, and carbon dioxide are few examples of common pollutants. There is a critical need for systems that not only monitor air pollution levels but can also predict future levels of pollution. Hence, in this paper, a system which monitors the air quality using MiCS6814 sensor, MQ135 sensor, MQ131 sensor and PM2.5 sensor and forecasts the Air Quality Index for next five hours using Linear Regression, Support Vector Regression, SARIMAX, GBDT and Stacked ensemble model is proposed. The proposed Machine Learning models are compared using RMSE as a metric and the model with lower RMSE value is chosen. This project can be used in major cities to monitor the air quality remotely and can in turn help to reduce the air pollution level.**

Keywords— **IOT, Machine Learning, Air Quality, AQI, Forecasting, Stacking model, GBDT model, SARIMAX model, Linear regression model, SVR model, RMSE**

I. INTRODUCTION

Air pollution is the presence of substances in the atmosphere that are harmful to the health of humans and other living beings or cause damage to the climate or to materials. There are different types of air pollutants, such as gases, particulates, and biological molecules. The reasons for this recent steep increase in air pollution include a warming climate, increased human consumption patterns driven by population growth and the increasing level of factories and mining operations. Air pollution causes diseases including stroke, heart disease, lung cancer, chronic obstructive pulmonary diseases and respiratory infections. While these effects emerge from long-term exposure, air pollution can also cause short-term problems such as sneezing and coughing, eye irritation, headaches, and dizziness. Air pollutants cause less-direct health effects when they contribute to climate change, heat waves, extreme weather, food supply disruptions, and increased greenhouse gases. Industry, transportation, coal power plants and household solid fuel usage are major contributors to air pollution. Air pollution continues to rise at an alarming rate, and affects economies and people’s quality of life. The most popular measure of outdoor air pollution is the Air Quality Index or AQI which rates air conditions based on concentrations of five major pollutants: ground-level ozone, particle pollution (or particulate matter), carbon monoxide, sulfur dioxide, and nitrogen dioxide. Worldwide, bad outdoor air causes an estimated 4.2 million premature deaths every year, according to the World Health Organization. Hence, we are at a higher risk of air pollution now than ever. This calls for immediate action and preventive measures to not only control the increasing air pollution levels but also save millions if not thousands of people. Therefore, there is a critical need for systems that not only monitor air pollution levels but can also predict future levels of pollution.

II. RELATED WORK

A. A Smart Air Pollution Monitoring System [1]

This paper proposes an air pollution monitoring system developed using the Arduino microcontroller. The main objective of this paper is to design a smart air pollution monitoring system that can monitor, analyse and log data about air quality to a remote server and keep the data up to date over the internet. Air quality measurements are taken based on the Parts per Million (PPM) metrics and analyzed using Microsoft Excel. The level of pollutants in the air are monitored using a gas sensor, Arduino microcontroller and a Wi-Fi module. Air quality data is collected using the MQ135 sensor. The data is first displayed on the LCD screen and then sent to the Wi-Fi module. The Wi-Fi module transfers the measured data valve to the server via the internet. The Wi-Fi module is configured to transfer measured data to an application on a remote server called “Thing speak”. The online application provides global access to measured data via any device that has internet connection capabilities. The results are displayed on the designed hardware's display interface and are accessed via the cloud on any smart mobile device.

B. Detection and Prediction of Air Pollution using Machine Learning Models [2]

In this paper, Logistic regression is employed to detect whether a data sample is either polluted or not polluted and Autoregression is employed to predict future values of PM2.5 based on the previous PM2.5 readings. Knowledge of level of PM2.5 in nearing years, month or week, enables us to reduce its level to lesser than the harmful range. This system attempts to predict PM2.5 level and detect air quality based on a data set consisting of daily atmospheric conditions in a specific city. The dataset used in this system has the following attributes - temperature, wind speed, dewpoint, pressure, PM2.5 Concentration(ug/m^3) and the classification result – data sample is classified as either polluted or not polluted. Based on the logit function, the Logistic Regression model classifies the training data to be either 0 (not polluted) or 1 (polluted) and accuracy is verified using the test data. The Autoregressive model modifies the dataset into time series dataset by taking the date and previous PM2.5 values from the main data set and makes the future predictions.

C. Air Quality Index and Air Pollutant Concentration Prediction Based on Machine Learning Algorithms [3]

In this paper, support vector regression (SVR) and random forest regression (RFR) are used to build regression models for predicting the Air Quality Index (AQI) in Beijing and the nitrogen oxides (NOX) concentration in an Italian city, based on two publicly available datasets. In this experiment, the AQI of Beijing is taken as the regression target. For the SVR-based model training, radial basis function (RBF) is chosen as the kernel function.The kernel parameter gamma (γ) and the penalty parameter (C) are selected by a grid search method. For the RF-based model, 100 regression trees were used to build the regression model. The root-mean-square error (RMSE), correlation coefficient (r), and coefficient of determination (R2) were used to evaluate the performance of the regression models. This work also illustrates that combining machine learning with air quality prediction is an efficient and convenient way to solve some related environment problems.

D. Prediction of Air Quality Index in Metro Cities using Time Series Forecasting Models [4]

In this paper, SARIMAX and Holt-Winter's models are used to predict the air quality index. This work discusses how these time series forecasting models can be utilized to predict the values of the Air Quality Index(AQI) based on past data. It also compares the various models used for prediction. These models have their strengths and weaknesses which can be measured and based upon them, the best model out of these is used for the prediction of AQI. The Mean Absolute Percentage Error (MAPE) is used as the score function to analyze the performance of models. The prediction accuracy of both models is calculated and compared by comparing their respective MAPE values. Though, the Holt- Winter's algorithm has an advantage over the ARIMA model that it can handle seasonality, but, the results produced by the Holt Winter's model are not much accurate. The SARIMAX model, on the other hand, handles seasonality and delivers results much better than the Holt-Winter's model.

E. A Bagging-GBDT ensemble learning model for city air pollutant concentration prediction [5]

In this paper, the Gradient Boosting Decision Tree (GBDT) method is introduced into the base learner training process of Bagging framework. A prediction model that predicts the level of the pollutant PM2.5 based on the Bagging ensemble learning framework is proposed. The city Beijing of China is considered as an example and an PM2.5 concentration prediction model to forecast the PM2.5 concentration for the next 48 hours at a given time point is established. The first Bagging-GBDT model corresponds to the number of training rounds as 5, the number of GBDT basic decision trees per round as 20, and the maximum height as 6 while the second Bagging-GBDT model corresponds to the number of training rounds as 10, the number of GBDT basic decision trees per round as 50 and the maximum height as 6. To measure the validity of the model, support vector machine regression models and random forest models are also used to calculate three statistical indicators (RMSE, MAE and R²) for the proposed models on the test set to compare models performance.

III. DATASET

The air quality data is taken from Kaggle titled "Air Quality Data in India (2015 - 2020)" that has been created by user Rohan Rao[6]. The dataset used "city_hour.csv" is used as the training dataset for the models.

The data collected locally using the specified sensors over a period of 780 hours is used as the testing dataset for the models.

IV. MODELS

A. Linear regression model

Linear regression models are the most basic types of statistical techniques and widely used predictive analysis. They show a relationship between two variables with a linear algorithm and equation. Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear relationship between explanatory (independent) variables and response (dependent) variables.

The multiple linear regression equation is as follows:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \epsilon$$

where, for i=n observations, y_i is the dependent variable, x_i are the explanatory variables, β_0 is the y-intercept (constant term), β_p are the slope coefficients for each explanatory variable and

ϵ is the model's error term (also known as the residuals).

B. Support Vector Regression model

Support Vector Regression is a supervised learning technique based on the concept of Vapnik's support vectors. It aims at reducing the error by determining the hyperplane and minimising the range between the predicted and the observed values. Minimising the value of w in the equation given below is similar to the value defined to maximise the margin, where the summation part represents an empirical error.

$$\min ||w||^2 + C\sum_{i}^{n}(\xi_i^+ + \xi_i^-)$$

Hence, to minimise this error, the following equation is used.

$$f(x) = \sum_{i}^{n}(\alpha_i^* + a_i)K(x, x_i) + B$$

Here the alpha term represents the Lagrange multiplier and its value is greater than equal to 1. K represents the kernel function and B represents the bias term. The kernel SVMs have more flexibility for non-linear data because more features to fit a hyperplane instead of a two-dimensional space can be added. The equation for the Gaussian Radial Basis Function (RBF) is given below.

f(X1, X2) = exp(-gamma * ||X1 - X2||^2)

In this equation, gamma specifies how much a single training point has on the other data points around it. ||X1 - X2|| is the dot product between the features.

C. SARIMAX time series model

Time series is a sequence of observations recorded at regular time intervals. Time series analysis involves understanding various aspects about the inherent nature of the series so that more meaningful and accurate forecasts can be created. There are 11 different classical time series forecasting methods - AR, MA, ARMA, ARIMA, SARIMA, SARIMAX, VAR, VARMA, VARMAX, SES and HWES. The SARIMAX model -the Seasonal Autoregressive Integrated Moving Average Exogenous model is used here. There are seven parameters in an SARIMAX model - p,d,q where values of p and q are determined based on the autocorrelation and partial autocorrelation plots and the value of d depends on the level of stationarity in the data. We have a seasonal autoregressive order denoted by upper-case P, an order of seasonal integration denoted by upper-case D, and a seasonal moving average order signified by upper-case Q while the The fourth and last order is the length of the cycle.

For n exogenous variables defined at each time step t, denoted by xit for i≤n, with coefficients βi, the SARIMAX(p,d,q) (P,D,Q,s) model is denoted by

$$\Theta(L)^p\theta(L^s)^P\Delta^d\Delta_s^D y_t=\Phi(L)^q\phi(L^s)^Q\Delta^d\Delta_s^D\epsilon_t+\sum_{i=1}^n\beta_i x_{it}$$

D. Gradient Boosted Decision Tree ensemble model

Gradient boosted decision trees (GBDT) is an ensemble learning method that combines many learners to build a more robust and accurate model.

In many supervised learning problems one has an output variable y and a vector of input variables x described via a joint probability distribution P(x,y). Using a training set $\{(x_1,y_1), \dots (x_n,y_n)\}$of known values of x and corresponding values of y, the goal is to find an approximation F^(x) to a function F(x) that minimizes the expected value of some specified loss function L(y, F(x)):

$$\hat{F} = \arg\min_F \mathbb{E}_{x,y}[L(y, F(x))].$$

The gradient boosting method assumes a real-valued y and seeks an approximation F(x) in the form of a weighted sum of functions h_i (x) from some class H, called base (or weak) learners:

$$\hat{F}(x) = \sum_{i=1}^{M} \gamma_i h_i(x) + \text{const}.$$

In accordance with the empirical risk minimization principle, the method tries to find an approximation F(x) that minimizes the average value of the loss function on the training set, i.e., minimizes the empirical risk. It does so by starting with a model, consisting of a constant function F_0(x), and incrementally expands it in a greedy fashion:

$$F_0(x) = \arg\min_\gamma \sum_{i=1}^n L(y_i, \gamma),$$

$$F_m(x) = F_{m-1}(x) + \arg\min_{h_m\in\mathcal{H}} \left[\sum_{i=1}^n L(y_i, F_{m-1}(x_i) + h_m(x_i))\right],$$

where h_m belonging to H is a base learner function.

The idea is to apply a steepest descent step to this minimization problem (functional gradient descent). If we considered the continuous case, i.e. where H is the set of arbitrary differentiable functions on R.

$$F_m(x) = F_{m-1}(x) - \gamma_m \sum_{i=1}^n \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i)),$$

$$\gamma_m = \arg\min_\gamma \sum_{i=1}^n L\left(y_i, F_{m-1}(x_i) - \gamma\nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i))\right),$$

where the derivatives are taken with respect to the functions F_i belonging to {1,...,m}.

E. Stacking Ensemble model

Stacking is an ensemble learning technique that combines multiple classification or regression models using a meta-classifier or a meta-regressor. The architecture of a stacking model involves two or more base models, often referred to as level-0 models, and a meta-model that combines the predictions of the base models, referred to as a level-1 model. Level - 0 models fit on the training data and whose predictions are compiled while level - 1 models learn how to best combine the predictions of the base models.

The base models used are:

1) Xgboost model

XGBoost is a scalable end-to-end machine learning system for gradient tree boosting. The tree ensemble model used in XGBoost is trained in an additive manner until stopping criteria (e.g., the number of boosting iterations, early stopping rounds and so on) are satisfied. The optimization objective of iteration t can be approximately described to minimize the following formula:

$$\mathscr{L}^{(t)} \simeq \sum_{i=1}^n \left[l\left(y_i, \hat{y}^{(t-1)}\right) + \partial_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right) f_t(x_i) + \frac{1}{2}\partial^2_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right) {f_t}^2(x_i)\right] + \Omega(f_t)$$

where $\mathscr{L}$(t) is the objective function to be solved at the t-th iteration. l is a differentiable convex loss function that measures the difference between the prediction of the i-th instance at the t-th iteration and the target yi. ft(x) is the increment. The left-hand side of the equation refers to a two order Taylor approximation of a loss function that controls the bias of the model fitting the training data while the right-hand side of Ω(ft) is the regularization term, which penalizes the complexity of the model and helps smooth final learned weights to prevent overfitting.

2) Support Vector Regression (SVR) model

Support Vector Regression is a supervised learning technique based on the concept of Vapnik's support vectors. It aims at reducing the error by determining the hyperplane and minimising the range between the predicted and the observed values. Minimising the value of w in the equation given below is similar to the value defined to maximise the margin,

where the summation part represents an empirical error.

$$\min ||w||^2 + C\sum_i^n (\xi_i^+ + \xi_i^-)$$

Hence, to minimise this error, the following equation is used.

$$f(x) = \sum_i^n (\alpha_i^* + a_i)K(x, x_i) + B$$

Here the alpha term represents the Lagrange multiplier and its value is greater than equal to 1. K represents the kernel function and B represents the bias term. The kernel SVMs have more flexibility for non-linear data because more features to fit a hyperplane instead of a two-dimensional space can be added. The equation for the Polynomial kernel is given below.

$$K(x, x_i) = \gamma(x * x_i + 1)^d$$

Where d is the polynomial degree and γ is the polynomial constant.

3) Random forest regression model

Random forests (RFs are an ensemble learning method for classification, regression, and other tasks. An RF operates by constructing multiple decision trees at different training times, and outputting the class representing the mode of classes (classification) or the mean prediction (regression) of individual trees.

The RF algorithm incorporates growing classification and regression trees (CARTs). Each CART is built using random vectors. For the RF-based classifier model, the main parameters were the number of decision trees, as well as the number of features (NF) in the random subset at each node in the growing trees. During model training, the number of decision trees was determined first.

For the number of trees, a larger number is better, but takes longer to compute. A lower NFleads to a greater reduction in variance, but a larger increase in bias. NFcan be defined using the empirical formula: NF=√M, where M denotes the total number of features.

RF can be applied to classification and regression problems, depending on whether the trees are classification or regression trees. The regression model is shown in Figure 2. Assuming that the model includes T Regression trees (learners) for regression prediction, the final output of the regression model is:

$$H(x) = \frac{1}{T}\sum_{i=1}^{T} h_i(x),$$

where T is the number of regression trees, and hi(x) is the output of the i-th regression tree (hi) on sample x. Therefore, the prediction of the RF is the average of the predicted values of all the trees.

V. METHODOLOGY

The overall architecture of the system is shown in the figure below:

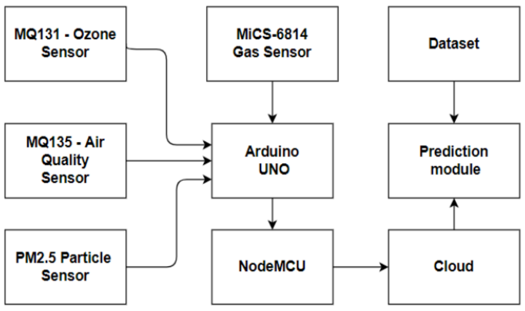

Fig 1 Air Pollution Monitoring and Prediction system

A. Collection of Air Quality Data

An IOT device is built using Arduino, MQ-131 Ozone sensor, MQ-135 Air Quality sensor, PM2.5 Particle sensor, MiCS 6814 Gas sensor and NodeMCU. The MQ-131 sensor measures the concentration of ozone (O3) in air, the MQ-135 Air Quality sensor measures the Air Quality Index(AQI) and the PM 2.5 Particle sensor measures the concentration of particulate matter that are less than two and one half microns in diameter. The MiCS 6814 Gas sensor is used to measure the concentrations of Carbon Monoxide (CO), Nitrogen Dioxide (NO2) and Ammonia (NH3) gases. The sensors collect the required data and the Arduino connected to the sensors is used to read the data. Once the data is collected, it is sent to the cloud using NodeMCU. This collected data can be used as a testing dataset in order to forecast the Air Quality Index(AQI) of upcoming hours.

B. Cleaning and Preprocessing of Dataset

The dataset contains 707876 records with the attributes-City, Date time, PM2.5, PM10, NO, NO2, NOx, NH3, CO, SO2, O3, Benzene, Toluene, Xylene, AQI and AQI_Bucket. The attribute ‘City’ contains 26 unique values of which the records with value ‘Chennai’ are considered. The total number of records considered for cleaning is 48192. The dataset is cleaned by handling the missing data values and noisy data. The missing values are handled by filling it with the most probable value. This is done by calling the interpolate() function in which the related known values are used to estimate the unknown value. Finally, the data is reduced by attribute subset selection method, in which the highly relevant attributes are used and other attributes are discarded. Date time, PM2.5, NO2, NH3, CO, O3 and AQI are chosen as the highly relevant attributes.

C. Forecasting the Air Quality Index (AQI) using non-time series algorithms

The required libraries to read and preprocess the dataset are imported. The dataset is read and stored in a dataframe. The preprocessing of data is carried out as explained earlier. Since most of the machine learning algorithms proposed in this system are supervised , it is necessary to convert this time series problem to supervised problem.

1) Conversion of Time series to Supervised Problem

The time series problem is converted to a supervised problem by storing the AQI values of the past 4 hours (AQI(t-1),AQI(t-2),...,AQI(t-4)) and AQI values for the next 5 hours (AQI(t+1),AQI(t+2),...,AQI(t+5)) for the AQI of a particular hour. The above AQI values are stored in a new dataframe and are concatenated with the older dataframe .The above procedure is done for both training and testing datasets. In the concatenated test data frame, the AQI values in the last 5 rows are considered as actual values to be forecasted. Hence, these values are stored in a list named actual values and the corresponding rows are dropped.

2) Building the Machine Learning models

Once the required data frame is created, the input features(X) are chosen as ‘PM2.5’, ’NO2’, ’NH3’, ’CO’, ’O3’, ’AQI’, ’AQI(t-

4)', 'AQI(t-3)', 'AQI(t-2)', 'AQI(t-1)' and the target variables(Y) are chosen as 'AQI(t+1)', 'AQI(t+2)', 'AQI(t+3)', 'AQI(t+4)', 'AQI(t +5)'. The numpy arrays named X_train and Y_train contain the input and target values of the training dataset whereas the numpy arrays named X_test and Y_test contain the input and target values of the testing dataset respectively.

The Machine Learning (ML) models which employ the above procedure are as follows:

Linear Regression model

A model is created using LinearRegression() function and is fitted with the training dataset (X_train and Y_train).

Support Vector Regression model

A regressor model is created using the SVR() function which predicts one target variable at a time. In order to predict 5 target variables, another model is created using the MultiOutputRegressor() function by passing the regressor model as a parameter. This model is fitted with the training dataset (X_train and Y_train).

Gradient Boosted Decision Tree Ensemble model

A regressor model is created using the GradientBoostingRegressor() function with parameters max_depth as 18 and n_estimators as 44 which predicts one target variable at a time. In order to predict 5 target variables, another model is created using the MultiOutputRegressor() function by passing the regressor model as a parameter . This model is fitted with the training dataset (X_train and Y_train).

Stacking Ensemble model

A model is created using the StackingRegressor() function which receives a list of base estimators and a final estimator as parameters. The base estimators are chosen as Random Forest Regressor , XGBoost Regressor and Support Vector Machine (SVM). By default, the final estimator is RidgeCV. Since this model can predict only one target variable at a time, using a loop it is trained 5 times with different input features and a different target variable . Each time when the model is fitted with the training data and the predictions are made for the testing data , the predicted values are added as the new input feature for the next target variable to be predicted and the predicted value of X_test[rows-1] is stored in a list named predicted values.

3) Prediction of AQI values

Once the model is trained, predictions for the test dataset are made using the predict() method when X_test is passed as a parameter. The predicted values of X_test[rows-1] are the AQI values for the next 5 hours. Hence, they are stored in a list named predicted values. Incase of the Stacking Ensemble model , the AQI values for the next 5 hours are obtained as and when the loop ends.

D. Forecasting the Air Quality Index (AQI) using SARIMAX time series algorithm

The dataset is read from the .csv file and the preprocessing is carried out as explained earlier. The dataset's stationality is checked using Augmented Dickey-Fuller test. The Augmented Dickey–Fuller test (ADF) test is an augmented version of the Dickey–Fuller test for a larger and more complicated set of time series models. It tests the null hypothesis that a unit root is present in a time series sample. The alternative hypothesis is different depending on which version of the test is used, but is usually stationarity or trend-stationarity. The model is created by calling the SARIMAX() function with parameters - endogenous values as the data to be predicted, exogenous values the values that affect the values to be predicted, the order of the model as (0,1,0) and the seasonal order as (0,0,0,24). The model is fitted and the predictions are made for the next 5 hours.

E. Evaluation of models

Root Mean Squared Error (RMSE) is used as a metric to measure the deviation between actual and predicted values.The RMSE value is calculated as the square root of mean squared error between actual and predicted values. The actual values and predicted values are passed as parameters to sqrt(mean squared error()) function and the RMSE value is obtained.

VI. EXPERIMENTAL RESULTS

The specified models are tested using the dataset collected using IOT to forecast the Air Quality Index of the upcoming hours. The predicted values are shown in the table below:

TABLE - I
PREDICTED AQI VALUES OF MODELS

Hour	Linear model	SVR model	SARIMAX model	GBDT model	Stacking model
1	116.808	114.584	109.473	112.937	115.870
2	120.582	114.851	105.988	116.055	116.679
3	124.184	114.845	103.775	116.409	110.490
4	127.909	114.820	103.974	108.979	105.939
5	130.590	114.977	121.011	106.464	106.659

The metric used in this paper to compare the various models' performance is Root Mean Squared Error (RMSE). Root Mean Square Error (RMSE) is a standard way to measure the error of a model in predicting quantitative data. RMSE is a kind of distance between the vector of predicted values and the vector of observed values. The RMSE values of the models are shown in the table below:

TABLE - II
RMSE VALUES OF MODELS

Model	RMSE value
Linear model	17.762892474541093
SVR model	7.490859762783875
SARIMAX model	9.64983160065426
GBDT ensemble model	4.767973792073978
Stacking ensemble model	3.3863095891044606

VII. CONCLUSIONS

In this paper, various Machine Learning models such as Linear Regression model, Support Vector Regression model, SARIMAX time series model, GBDT ensemble model and Stacking ensemble model are used to forecast the AQI values for the next five hours. By comparing the Root Mean Squared Error values of all the models, it can be seen that the Stacking Ensemble model has the lowest RMSE value. Hence, this model is chosen to forecast the Air Quality Index of the next five hours.

REFERENCES

[1] Okokpujie, Kennedy & Noma-Osaghae, Etinosa & Odusami, Modupe & John, Samuel & Oluwatosin, Oluga. (2018). A Smart Air Pollution Monitoring System. International Journal of Civil Engineering and Technology. 9. 799-809.

[2] C R, Aditya & Deshmukh, Chandana & K, Nayana & Gandhi, Praveen & astu, Vidyav. (2018). Detection and Prediction of Air Pollution using Machine Learning Models. International Journal of Engineering Trends and Technology. 59. 204-207. 10.14445/22315381/IJETT-V59P238.

[3] Liu, Huixiang & Li, Qing & Yu, Dongbing & Gu, Yu. (2019). Air Quality Index and Air Pollutant Concentration Prediction Based on Machine Learning Algorithms. Applied Sciences. 9. 4069. 10.3390/app9194069.

[4] Arora, Himanshu & Solanki, Arun. (2020). Prediction of Air Quality Index in Metro Cities using Time Series Forecasting Models. Xi'an Jianzhu Keji Daxue Xuebao/Journal of Xi'an University of Architecture & Technology. 12. 3052-3067. 10.37896/JXAT12.05/1721.

[5] Liu, Xinle & Tan, Wenan & Tang, Shan. (2019). A Bagging-GBDT ensemble learning model for city air pollutant concentration prediction. IOP Conference Series: Earth and Environmental Science. 237. 022027. 10.1088/1755-1315/237/2/022027.

[6] “Air Quality Data in India (2015 - 2020)”, Kaggle. 28-July-2020. [Online].Available:www.kaggle.com/rohan rao/air-quality-data-in-india [Accessed: 23-Mar.-2021].