AUTOMATED AIR QUALITY DATA PREDICTION USING DENSITY BASED SPATIAL CLUSTERING OF APPLICATIONS WITH NOISE AND GRADIENT BOOSTING REGRESSOR ALGORITHM

Project report in partial fulfillment of the requirement for the award of the degree of

Master of Computer Applications

Submitted By:

Arkaprava Chakrabarty	**12021007015015**
Suman Manna	**12021007015033**
Sukanya Dey	**12021007015034**
Madhunita Mitra	**12021007015028**
Pritam Sarkar	**12021007015005**
Meghna Bose	**12021007015027**

Under the Guidance of
Prof. Poojarini Mitra
Department of Master of Computer Applications

UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA

University Area, Plot No. III – B/5, New Town, Action Area – III, Kolkata – 700160.

CERTIFICATE

This is to certify that the project titled "**Automated Air Quality Data Prediction using Density Based Spatial Clustering of Applications with Noise and Gradient Boosting Regressor Algorithm"** submitted by Arkaprava Chakrabarty **(Enrollment No.- 12021007015015),** Suman Manna **(Enrollment No.- 12021007015033),** Pritam Sarkar **(Enrollment No.- 12021007015005),** Sukanya Dey **(Enrollment No.- 12021007015034),** Madhunita Mitra **(Enrollment No.- 12021007015027),** Meghna Bose **(Enrollment No.- 12021007015027)** students of UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA, in partial fulfillment of requirement for the Degree of Master of Computer Applications, is a bonafide work carried out by them under the supervision and guidance of **Prof. Poojarini Mitra** during 4th Semester of academic session of 2022 - 2023. The content of this report has not been submitted to any other university or institute.

I am glad to inform that the work is entirely original and its performance is found to be quite satisfactory.

Prof. Poojarini Mitra	Prof. Kaustav Bhattacharjee
Assistant Professor	Head of the Department
Department of Master of Computer Applications	Department of Master of Computer Applications
UEM, Kolkata	UEM, Kolkata

ACKNOWLEDGEMENT

We would like to take this opportunity to thank everyone whose cooperation and encouragement throughout the ongoing course of this project remains invaluable to us.

We are sincerely grateful to our guide Prof. Poojarini Mitra of the Department of Master of Computer Applications, UEM, Kolkata, for her wisdom, guidance and inspiration that helped us to go through with this project and take it to where it stands now.

We would also like to express our sincere gratitude to Prof. Kaustav Bhattacharjee, HOD, Department of Master of Computer Applications, UEM, Kolkata and all other departmental faculties for their ever-present assistant and encouragement.

Last but not the least, we would like to extend our warm regards to our families and peers who have kept supporting us and always had faith in our work.

Arkaprava Chakrabarty

Suman Manna

Pritam Sarkar

Sukanya Dey

Madhunita Mitra

Meghna Bose

<u>**TABLE OF CONTENTS**</u>

ABSTRACT

In weather data prediction, the main motive is to correct the erroneous values from the collected values from the weather station nodes. The weather took a vital role in day-to-day life of a human being. The weather nodes collecting the values it may be an erroneous one. The main motive is to detect the erroneous value, that where the value is situated and complete the missing or erroneous value with the previous trend. The thorough analysis of the previous data will help to predict the values more accurately. Not only this it can be further extended to that level where the machine has trained itself for further future, where it can automatically detect the erroneous or the missing value, and try to correctly predict it accurately as produced in the thorough testing phase. The dataset we have currently the file is divided into two part one is the train dataset, by which we are training the machine that how the data is searched using Density Based Spatial Clustering Algorithm (DBSCAN). The DBSCAN Algorithm is making some cluster that are further helps to visualize and check the error or the missing value known as Noise. Further we are training the machine for predicting the noise value, by making a Boosting tree with the help of Gradient Boosting Algorithm. The algorithm makes a boosting tree from the nearest values to check and calculate a fruitful and correct data. The Gradient Boosting Algorithm further extends the Prediction to the extend where if the further collected data from the data nodes are get a noise value it can easily calculate and predict the value accurately. Thus, it will help the hardware nodes to collect data fast and more prominent way. By software processing we are correct the collected data purify it to produce perfect data in the future.

CHAPTER 1

INTRODUCTION

For clustering data, a well-liked unsupervised machine learning approach is density-based spatial clustering of applications with noise (DBSCAN).[6] The method assembles clusters of data points by locating dense areas of data points in a dataset. For datasets with complex forms and changing densities, it is very helpful. The algorithm is a useful tool for exploratory data analysis because it does not require the user to predetermine the number of clusters. Additionally, it can successfully handle datasets with non-uniform density and is robust to noise and outliers.

We analyze the algorithm's strategy [6] for locating core, boundary, and noise points and describe how it chooses the ideal neighborhood size and the bare minimum number of points needed for a core point. The algorithm's performance measures are also discussed, along with the algorithm's uses in a number of other contexts, such as picture segmentation, social network analysis, and anomaly identification. We emphasize the DBSCAN algorithm's advantages and disadvantages as well as its potential for further study and development before drawing to a close. DBSCAN is a strong tool for clustering spatial data and has a variety of real-world uses.

Identification of pollution sources, evaluation of the success of pollution management methods, and forecasting future pollution levels are some applications of air pollution noise detection using DBSCAN. The accuracy of air pollution analysis can be increased by identifying and deleting noisy data points, which will help with decision-making and policy creation for better air quality. Overall, air pollution datasets may be used to find patterns and detect noise using the DBSCAN air pollution noise detection method. It can promote initiatives to lessen the detrimental impacts of air pollution on human health and the environment and help us better understand issues related to air quality.

Gradient Boosting [7] is a well-liked machine learning approach for classification and regression tasks is gradient boosting. It is an ensemble method that builds a stronger model by combining several weak models, often decision trees. Iteratively adding new trees to the model while fixing the flaws in the prior trees is how the method operates. The capacity of gradient boosting to manage complicated datasets with high dimensional features and nonlinear interactions between variables is well established. It is a useful tool for real-world applications since it is efficient in handling missing data and outliers.

We give a summary of the Gradient Boosting algorithm's salient characteristics. We present the gradient descent method used by the algorithm to discover the best settings for each weak learner, as well as its strategy for minimizing the loss function. [7] We also go over the value of regularization approaches in avoiding overfitting and enhancing the model's generalization capabilities.

We discuss the algorithm's advantages and disadvantages as well as its uses in a number of industries, such as banking, healthcare, and natural language processing. We also cover current advancements in gradient boosting, such as the application of deep neural networks and the incorporation of methods for reinforcement learning. Gradient Boosting is an effective machine learning technique that has a variety of uses. It is a useful tool for real-world applications thanks to its capacity for dealing with complicated datasets and nonlinear interactions, and its potential for future development and innovation ensures its continuous importance in the machine learning community. Applications of missing air pollution data via gradient boosting include increasing the precision and completeness of air pollution datasets, assisting air quality monitoring, and forecasting, and enabling the creation of air quality improvement policies.

Overall, the air pollution missing data using gradient boosting is a useful tool for predicting missing data points in air pollution datasets, helping to close knowledge gaps and supporting initiatives to lessen the negative effects of air pollution on human health and the environment. Forecasting the weather is a crucial duty for meteorologists, authorities, and numerous sectors. The hazards posed by natural disasters can be decreased with the aid of accurate weather forecasting, which can also boost agricultural productivity and improve transportation infrastructure. The accuracy of weather predictions can be harmed by the fact that meteorological data is frequently incomplete due to missing values. By forecasting the values that are absent from meteorological databases, a project on missing weather data tries to solve this issue.

In this research, missing data values are predicted using machine learning techniques utilizing the data that is currently accessible. [8] Based on past meteorological data, a machine learning model is trained, and its performance is assessed using a variety of measures, including mean absolute error and mean squared error. The project is implemented using DBSCAN (Density-based spatial clustering of applications with noise) and Gradient Boosting Regressor Model. DBSCAN is used to detect the noise(outliers) present in the existing data-set and then, Gradient Boosting Algorithm is applied to the resultant data-set to predict the missing (NaN) values. The initiative may have important ramifications for meteorology as well as numerous industries that depend on weather information.

The project can fill in data gaps and provide more thorough and precise information about weather conditions by forecasting missing variables in meteorological databases. This could result in more accurate weather predictions and better choices across many industries. In this project, we want to create a model for predicting missing values in weather datasets that will be able to do just that. The model's performance will be assessed using past weather data and compared to other available techniques. Insights into patterns and trends in meteorological data will also be provided by the initiative, which will help us comprehend the natural world and inform our decisions.

Prediction of missing data [7] is frequently simply one step in a wider machine learning pipeline. Other models can be fed the anticipated missing data to create predictions or produce insights. The missing data must frequently be predicted in real-time for many applications. For the purpose of making predictions almost instantly, the models can be accelerated and made more precise. In the fields of machine learning and data science, numerous novel techniques are being developed. To increase the precision of the predictions, the missing data prediction project can investigate novel approaches like deep learning or reinforcement learning.

Weather prediction algorithms can be tailored to give industry-specific forecasts that assist businesses and governments in planning for and reducing weather-related risks using the entire dataset along with the predicted data. To give real-time information regarding weather conditions and their effects on traffic, public transit, and other city services, weather prediction models can be integrated with smart city efforts.

Storms with extreme weather, such hurricanes, tornadoes, and floods, can result in considerable property damage and fatalities. Accurate and early warning of these disasters can assist individuals and governments in taking the proper measures. To more accurately forecast extreme weather events and offer more precise information about their possible effects, weather prediction models can be enhanced.

CHAPTER 2

BACKGROUND

2.1. Density-Based Spatial Clustering of Applications with Noise (DBSCAN):

The primary idea behind clustering analysis is to partition the data points into a few distinct batches or groups, so that the data points within the same group have similar features and the data points within separate groups have, in some ways, different properties. It consists of numerous various differential evolution-based methodologies.

For instance, DBSCAN (for distance between nearest points) and K-Means (for distance between points), as well as affinity propagation (for graph distance) and mean-shift (for distance between points).

This common understanding of "clusters" and "noise" is the foundation of the DBSCAN algorithm. The main principle is that at least a certain number of points must be present in the vicinity of each point within a cluster within a particular radius.

eps: The DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm, which is used to cluster data points based on their density, uses the hyperparameter epsilon () to do this.

Based on a density criterion, DBSCAN clusters together data points that are closely spaced apart. If there are at least a certain amount of other points (MinPts) within a certain range of a given point, then that point is thought to be a component of a cluster. Noisy points are those that are not a part of any cluster.

The radius of the area surrounding a point is determined by the value of epsilon. When two points are relatively close to one another, they are regarded as neighbours and might be included in the same cluster. A point is regarded as a core point and a cluster can form around it if it has at least MinPts neighbours within distance. There could not be enough points to form a cluster if the value of is too small, leading to a large number of noise points. Clusters may merge if is too huge, leaving only one large cluster.

The DBSCAN algorithm depends on selecting the appropriate values of and MinPts, which is frequently done through trial and error.

MinPTS: Minimum neighbours (data points) inside the eps radius are referred to as MinPts. The higher the number of MinPts has to be selected the larger the dataset. In general, the minimal MinPts can be computed as MinPts >= D+1 from the dataset's D dimensions. MinPts must be set to a value of at least 3.

The DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm uses MinPts as additional hyperparameter.

The formula MinPts calculates the bare minimum of points needed to create a cluster at a distance of epsilon. A point is regarded as a core point and a cluster can form around it if it has at least MinPts neighbours within distance. Noise is defined as points where there are no MinPts neighbours within distance.

The complexity and density of the dataset are typically considered when determining the value of MinPts. A larger value of MinPts can be useful if the dataset contains a higher density or more intricate patterns. A lower value of MinPts can be useful if the dataset has a lower density or simpler patterns.

The DBSCAN algorithm performs best when and MinPts are set to the appropriate values. Selecting and MinPts based on existing knowledge of the data or by visualising the data to gain a sense of the suitable density and distance parameters is a good place to start. However, experimentation and validation are frequently used to determine the ideal values of these hyperparameters.

2.2. Gradient Boosting Regressor Algorithm:

It is a potent machine learning approach which is frequently applied to classification and regression issues. A few weak models, like decision trees, are iteratively trained, and their predictions are then added to create a stronger final model.

The algorithm pinpoints the areas where the previous model underperformed at each iteration and concentrates on enhancing the forecasts in those areas by including new models. To achieve this, one must compute the gradient of the loss function with respect to the predictions and use the results to train a new model that concentrates on erasing the flaws in the older models. The final model is created by combining each of the separate weak models and weighting them based on how much they contributed to the final forecast.

Parameter Tuning for Gradient Boosting Algorithm: Gradient Boosting parameter tuning entails determining the hyperparameters' ideal values in order to enhance the algorithm's performance on a particular dataset. Hyperparameters are parameters that are set by the user prior to training and are not learned from the data.

Gradient Boosting's primary hyperparameters include the following:

1. **Learning Rate**: The learning rate in gradient boosting is a hyperparameter that regulates how quickly the algorithm learns from errors. The contribution of each tree in the series of decision trees that are constructed during the boosting process is specifically determined by the learning rate.

   Each tree contributes more to the final prediction when the learning rate is higher; this is the opposite of the case when the learning rate is lower. A lower learning rate may require more trees to reach the same level of accuracy but may also lessen the danger of overfitting. A greater learning rate can speed convergence but may also lead to overfitting.

   Depending on the issue at hand and the size of the dataset, the learning rate is commonly set at a value between 0.01 and 0.1. In general, it is advised to start with a modest learning rate and progressively increase it while keeping an eye on the model's performance on a validation set.

In conclusion, the learning rate is a crucial hyperparameter in gradient boosting that regulates how much each tree contributes to the outcome of the prediction. To strike a balance between overfitting and convergence speed, it is crucial to select an optimal learning rate.

2. **Number of Trees:** The number of trees is a hyperparameter in gradient boosting that regulates how many decision trees are successively constructed throughout the boosting procedure. The ultimate forecast is the weighted sum of the predictions made by all trees, and each tree is constructed to address the flaws of the prior tree.

   In gradient boosting, the number of trees is a crucial hyperparameter that influences the model's performance, complexity, and training time. In general, more trees can improve performance, but they also increase the chance of overfitting and lengthen the training period.

   The most popular method for determining the ideal number of trees is to run a grid search or random search over a range of values for the number of trees and pick the value that yields the best results on a validation set. Another strategy is early stopping, when training is halted as soon as a validation set's performance begins to decline.

   The task, the size of the dataset, and the various hyperparameters employed in the model all influence the ideal number of trees. In general, it is advised to start with fewer trees and gradually grow them while keeping track of how the system performs on a validation set.

   In conclusion, the number of trees is a crucial hyperparameter in gradient boosting that affects the model's performance and complexity. Finding the ideal number of trees that strikes a balance between model complexity and performance is essential.

3. **Maximum Depth of each tree:** The complexity of each individual tree in the ensemble is controlled by this. Deeper trees can catch more intricate patterns in the data, but if they get too complicated, they may overfit.

   The maximum depth of trees in gradient boosting is a hyperparameter that regulates the maximum depth of each decision tree in the series of trees that are constructed throughout the boosting procedure. The number of node levels between the root and leaf nodes determines how deep a decision tree is.

   In gradient boosting, the maximum depth of trees is a significant hyperparameter that influences the model's complexity and generalizability. A model with a higher maximum depth may be more sophisticated and better able to fit the training data, but it may also be more prone to overfitting and have a lower capacity for generalisation. On the other hand, a less complicated model that can generalise can be produced with a lower maximum depth.

   It is typical to conduct a grid search or randomised search over a range of maximum depth values and select the one that yields the greatest performance on a validation set in order to determine the ideal maximum depth of trees. Another strategy is early stopping, when training is halted as soon as a validation set's performance begins to decline.

The problem, the amount of the dataset, and the various hyperparameters employed in the model all affect the best maximum depth of trees. Generally speaking, it is advised to start with a lower maximum depth and then progressively increase it while keeping an eye on the performance on a validation set.

4. **Minimum number of samples needed to split a node:** This determines the bare minimum of samples a node must contain in order to be divided. Setting this value incorrectly can result in either overfitting or underfitting.

   The minimum number of samples needed to split an internal node is controlled by the hyperparameter min_samples_split in gradient boosting and decision trees. A node won't be divided if it contains fewer samples than min_samples_split, and the tree-building process for that node will end.

   Lowering the complexity of the tree and preventing overfitting can be accomplished by increasing min_samples_split, although doing so can also result in underfitting and less precise predictions. On the other side, lowering the min_samples_split can lead to overfitting while simultaneously making the tree more complicated and more accurate.

   The size of the dataset, the difficulty of the issue, and the other hyperparameters used in the model all affect the ideal value for min_samples_split. A grid search or randomised search over a range of min_samples_split values is frequently used to find the value that gives the highest performance on a validation set.

   Min_samples_split in gradient boosting can be specified at either the tree-level or the boosting-level. If it is configured at the decision tree level, each decision tree is affected. If it is set at the boosting-level, the entire series of decision trees created throughout the boosting process will be affected.

   A strategy that is frequently used to adjust these hyperparameters is to perform a grid search or a randomized search over a range of values for each hyperparameter. The set of hyperparameters that generates the best performance is selected as the final model after the model's performance is assessed using a validation set or cross-validation.

2.3. Grid Search CV

Grid Search CV (Cross Validation) is a machine learning technique for hyperparameter tuning that entails systematically assessing a model's performance with various combinations of hyperparameters. In Grid Search CV, a grid of hyperparameter values is established, and the algorithm is trained and assessed for each conceivable set of grids hyperparameter combinations. Using a scoring metric, such as accuracy, precision, recall, or F1-score, the model's performance is evaluated.

The dataset is often divided into training, validation, and testing sets to prevent overfitting. The validation set, testing set, and training set are used to train the model, validate the model's performance with various hyperparameter values, and test the model's final performance using the best hyperparameters. Finding the best hyperparameters for a model using this method can help the model perform better on a particular job.

<u>CHAPTER 3</u>

LITERATURE SURVEY

Zhu et. al. in [1] uses Machine Learning Algorithms such as Linear Regression, Decision Trees, Random Forest Method with the focus on predicting air relative humidity while taking into account a number of factors, including temperature, CO, tin oxide, non-metallic hydrocarbons, benzene, titanium, NO, tungsten, and indium oxide, temperature, etc. They have also used Root Mean Square Error to predict the accuracy. Doreswamy et. al. in [2] tackle air quality forecasting by using machine learning approaches to predict the hourly concentration of air pollutants like Ozone, PM2.5, and Sulfur dioxide. They use machine learning techniques because it can efficiently train a model on big data by using large-scale optimization algorithms. They use standard regression models (linear and nonlinear) to predict the hourly air pollution concentration. In this work, they propose refined models to predict the hourly air pollution concentration based on meteorological data of previous days by formulating the prediction over 24 h as a multi-task learning (MTL) problem. They propose a useful regularization by enforcing the prediction models of consecutive hours to be close to each other and compare it with several typical regularizations for MTL, including standard Frobenius norm regularization, nuclear norm regularization, and ℓ2,1-norm regularization. Their experiments have showed that the proposed parameter-reducing formulations and consecutive-hour-related regularizations achieve better performance than existing standard regression models and existing regularizations. Madhuri et. al. in [3] gathered the dataset using a variety of sensors and a microcontroller called an Arduino Uno. The air quality is then forecasted using the K-Nearest Neighbor (KNN) algorithm. This dataset has undergone many pre-processing features, such as attribute selection and normalization. The dataset is split into a training set and a test dataset once it is accessible. A machine learning method is then applied using the training dataset. The results are examined once the collected results are compared to the testing dataset. Bhalgat et. al. in [4] conduct experiments with the PRSA_Data_Aotizhongxin_20130301-20170228 dataset to assess the prediction error of five methods, namely the kernel ridge, linear regression, random forest, SVM imputation, and KNN imputation procedure, we propose a new method to handle missing values in weather data using machine learning algorithms. Each method's imputed values were then compared to the observed value. The proposed method's outcomes were contrasted with those of earlier methods. The process of estimating missing data is a crucial issue in data analysis, and many approaches have been put forth to address it, including statistics and data mining. The most common methods for handling missing values involve imputing missing values and eliminating all instances of missing values from a dataset. A line is fitted through a dataset made up of several features using the straightforward and traditional method of linear regression. The advantages of the linear regression algorithm are that it is a widely used technique and that it may be easily understood. It may be determined which parameters have the most influence and whether there is a positive or negative association based on the size of the weights. Deepu BP et. al. in [5] have forecasted the air quality of India by using machine learning algorithms to predict the air quality index (AQI) of a given area. Air quality Index is a standard measure to determine the quality of air. Concentration of Gases such as SO2, NO2, CO2, rspm, spm. etc. are recorded by the agencies. They have developed a model to predict the air quality index based on historical data of previous years and predicting over a particular upcoming year as a

Gradient decent boosted multivariable regression problem. They improved the efficiency of the model by applying cost Estimation for predictive Problem. They say that this model is capable of successfully predicting the air quality index of a total county or any state or any bounded region provided with the historical data of pollutant concentration. This paper presents an integrated model using Artificial Neural Networks and Kriging to predict the level of air pollutants at various locations in Mumbai and Navi Mumbai using past data available from meteorological department and Pollution Control Board. The proposed model is implemented and tested using MATLAB for ANN an R for Kriging and the results are presented. This system has used the Linear regression and Multilayer Perceptron (ANN) Protocol for prediction of the pollution of next day. The system helps to predict next date pollution details based on basic parameters and analyzing pollution details and forecast future pollution. Time Series Analysis was also used for recognition of future data points and air pollution prediction. This proposed system does two important tasks. Detects the levels of PM2.5 based on given atmospheric values. Predicts the level of PM2.5 for a particular date. Logistic regression is used to detect whether a data sample is either polluted or not polluted. Autoregression is employed to predict future values of PM2.5 based on the previous PM2.5 readings. The primary goal is to predict air pollution level in City with the ground data set.

The aforementioned articles lack an efficient method for removing noise from the data, and some of them use normalization, which just lowers the number of rows and hence, the total amount of data. In cases when only one or two columns are missing, the full row has been deleted; none of the articles have tried to forecast the missing data. This results in a lower accuracy of the employed methods than when the model is trained with more data and predicted values are added in place of null values.

CHAPTER 4

PROBLEM STATEMENT

There was no reliable way to remove noise from the dataset, noise values were included in the calculations, lowering the accuracy of the algorithms. Additionally, some data were missing from the entire datasets used, but these data were not predicted or used, reducing the accuracy of the trained models. These are the issues that the dataset being used was having. The project's specific goal is to forecast the concentration levels of the missing values of the air pollutants like PM2.5, PM10, CO, SO2, and NO2 in a specific area. The suggested system makes use of the Gradient Boosting Regression (missing values) algorithm to forecast the concentration levels of air pollutants and the DBSCAN algorithm(noise) to spatially cluster the data on air quality. This machine learning research aims to create an automated air quality forecast system using the Gradient Boosting Regressor Algorithm (GBR) and Density-based Spatial Clustering of Applications with Noise (DBSCAN). To forecast the concentrations of air pollutants, the proposed system will consider a number of meteorological variables as well as historical air quality data.

CHAPTER 5

PROPOSED SOLUTION

The air quality data are clustered and the locations with comparable air quality patterns are found using the DBSCAN algorithm and noise is filtered out as these values may potentially increase the mean values in turn decreasing the precision of the algorithms using this dataset. Based on the clustered data, a predictive model is created using the GBR method which predicts the missing values present in the dataset which gives an advantage by increasing accuracy of the predictions made using the dataset. Utilizing historical data on air quality as well as meteorological elements like temperature, humidity, wind speed, and wind direction, the system will be trained.

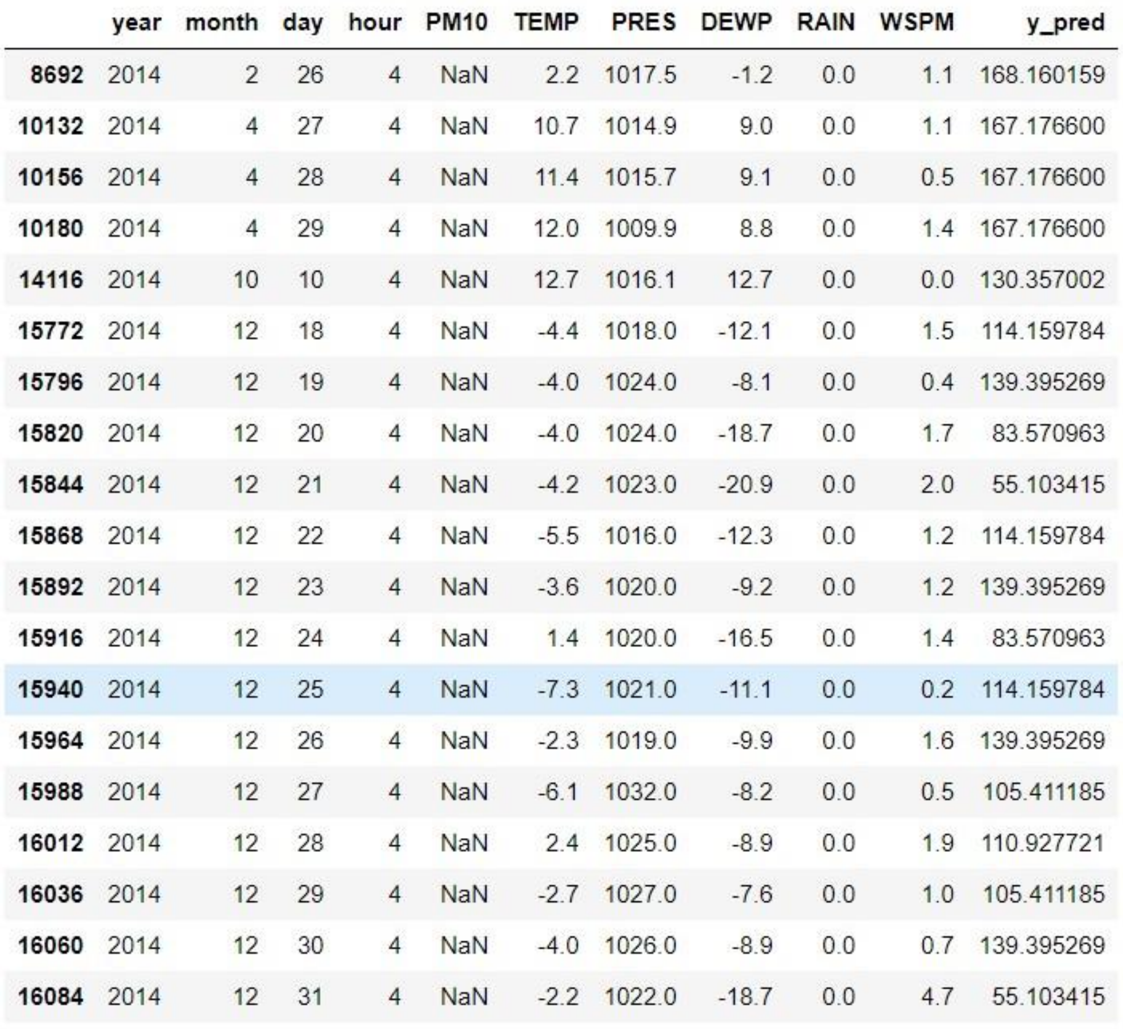
The importance of this initiative rests in its potential to increase the precision and effectiveness of air quality forecasting techniques. The suggested method can assist in locating places with high levels of air pollution and provide early warning of air pollution incidents. Policymakers and environmental organizations can utilize the system to create plans that effectively manage and control air pollution. Generally speaking, the project seeks to enhance air quality and safeguard public health.

5.1) DBSCAN algorithm is implemented in this project with following steps:

1. Identify the core points or points that were visited by more than MinPts neighbours by finding all the neighbour points inside eps.
2. Create a new cluster for each core point if it is not already associated with one.
3. Recursively locate all the points that are associated to it by density and add them to the same cluster as the core point.
4. Walk through the dataset's remaining unknown points iteratively. Noise consists of every point that do not belong to any cluster.

5.2) Gradient Boosting Regression's implementation in this project:

1. We first check the dataset for the presence of any null values, then we choose one column whose value needs to be predicted and remove any other null columns from the dataset.
2. The dataset is then divided into train and test, with the dataset containing NaN values being placed under test data and the remaining dataset being utilised for training.
3. The tuning of hyperparameters for the implementation of the gradient boosting algorithm is carried out using the GridSearchCV approach. The best parameters are chosen by GridSearchCV from a list of potential values.
4. Finally, we use the Gradient Boosting Regressor along with the parameters given by GridSearchCV and the fit method is implemented and prediction is done using the. predict () method.



	year	month	day	hour	PM10	TEMP	PRES	DEWP	RAIN	WSPM	y_pred
8692	2014	2	26	4	NaN	2.2	1017.5	-1.2	0.0	1.1	168.160159
10132	2014	4	27	4	NaN	10.7	1014.9	9.0	0.0	1.1	167.176600
10156	2014	4	28	4	NaN	11.4	1015.7	9.1	0.0	0.5	167.176600
10180	2014	4	29	4	NaN	12.0	1009.9	8.8	0.0	1.4	167.176600
14116	2014	10	10	4	NaN	12.7	1016.1	12.7	0.0	0.0	130.357002
15772	2014	12	18	4	NaN	-4.4	1018.0	-12.1	0.0	1.5	114.159784
15796	2014	12	19	4	NaN	-4.0	1024.0	-8.1	0.0	0.4	139.395269
15820	2014	12	20	4	NaN	-4.0	1024.0	-18.7	0.0	1.7	83.570963
15844	2014	12	21	4	NaN	-4.2	1023.0	-20.9	0.0	2.0	55.103415
15868	2014	12	22	4	NaN	-5.5	1016.0	-12.3	0.0	1.2	114.159784
15892	2014	12	23	4	NaN	-3.6	1020.0	-9.2	0.0	1.2	139.395269
15916	2014	12	24	4	NaN	1.4	1020.0	-16.5	0.0	1.4	83.570963
15940	2014	12	25	4	NaN	-7.3	1021.0	-11.1	0.0	0.2	114.159784
15964	2014	12	26	4	NaN	-2.3	1019.0	-9.9	0.0	1.6	139.395269
15988	2014	12	27	4	NaN	-6.1	1032.0	-8.2	0.0	0.5	105.411185
16012	2014	12	28	4	NaN	2.4	1025.0	-8.9	0.0	1.9	110.927721
16036	2014	12	29	4	NaN	-2.7	1027.0	-7.6	0.0	1.0	105.411185
16060	2014	12	30	4	NaN	-4.0	1026.0	-8.9	0.0	0.7	139.395269
16084	2014	12	31	4	NaN	-2.2	1022.0	-18.7	0.0	4.7	55.103415

Fig.1: Prediction of missing values for SO2 in the year 2014 in the 4th hour or 4AM

CHAPTER 6

EXPERIMENTAL SETUP AND RESULT ANALYSIS

6.1. Experimental Setup:

Hardware Requirements:

1. For which minimum RAM required is 8GB.
2. Any 64-bit intel or AMD processor.
3. 3 GB of Hard disk space is required

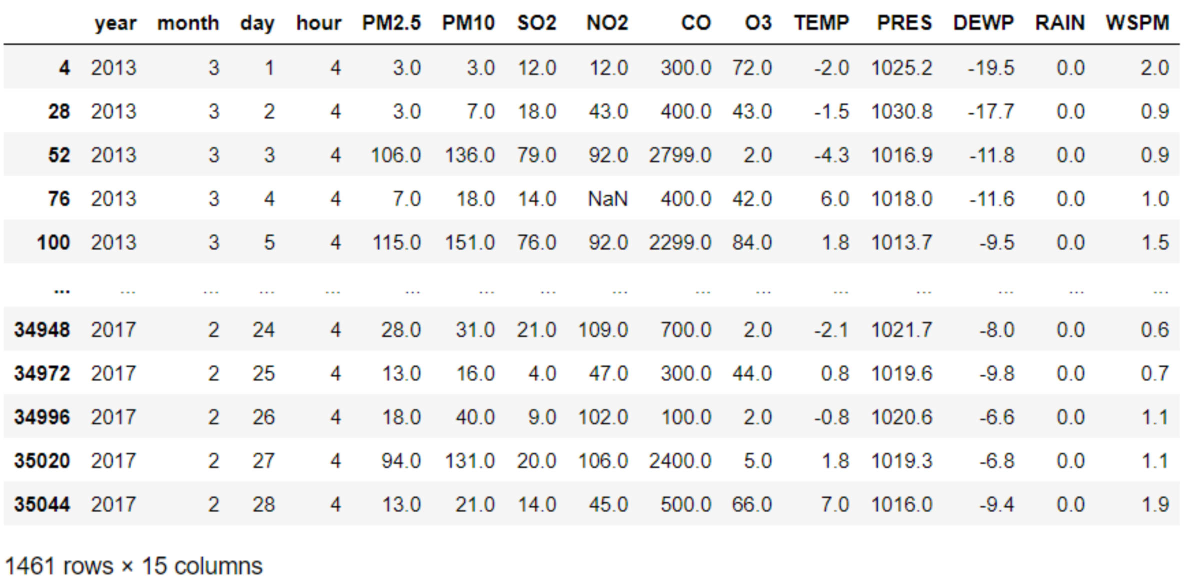
Software Requirements:

1. We have run the whole project in Anaconda Navigator, an open source distribution for python and R programming languages for data science. From which we are using Jupyter Notebook.

6.2. Result Analysis:

DBSCAN Algorithm implementation:

1. First we check that we have correctly split the dataset for sake of our convenience.

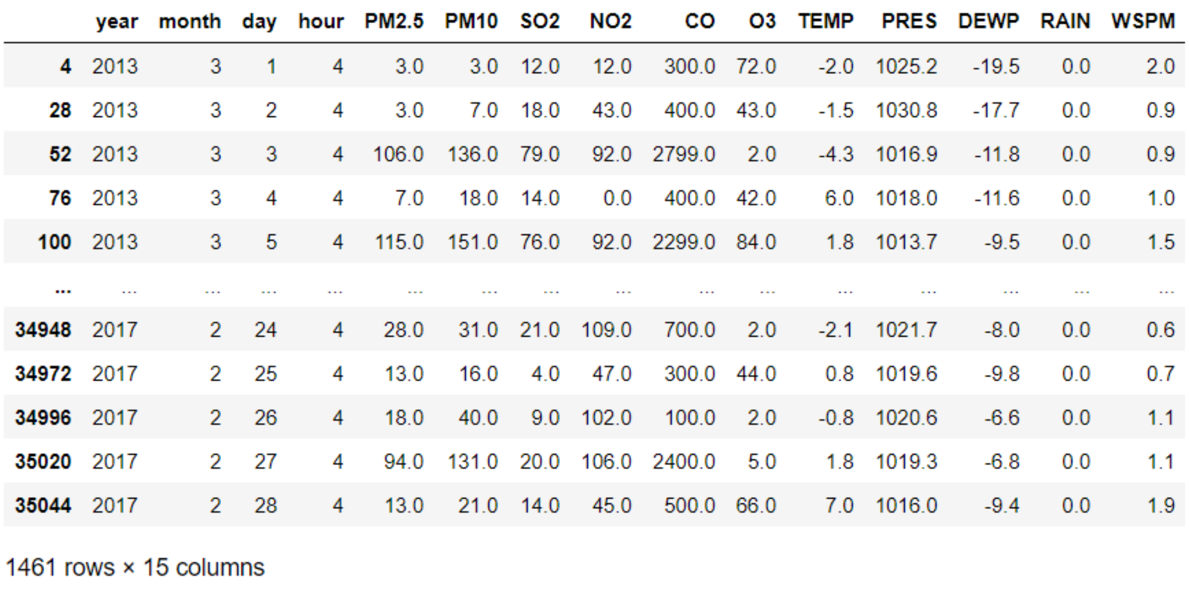


	year	month	day	hour	PM2.5	PM10	SO2	NO2	CO	O3	TEMP	PRES	DEWP	RAIN	WSPM
4	2013	3	1	4	3.0	3.0	12.0	12.0	300.0	72.0	-2.0	1025.2	-19.5	0.0	2.0
28	2013	3	2	4	3.0	7.0	18.0	43.0	400.0	43.0	-1.5	1030.8	-17.7	0.0	0.9
52	2013	3	3	4	106.0	136.0	79.0	92.0	2799.0	2.0	-4.3	1016.9	-11.8	0.0	0.9
76	2013	3	4	4	7.0	18.0	14.0	NaN	400.0	42.0	6.0	1018.0	-11.6	0.0	1.0
100	2013	3	5	4	115.0	151.0	76.0	92.0	2299.0	84.0	1.8	1013.7	-9.5	0.0	1.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
34948	2017	2	24	4	28.0	31.0	21.0	109.0	700.0	2.0	-2.1	1021.7	-8.0	0.0	0.6
34972	2017	2	25	4	13.0	16.0	4.0	47.0	300.0	44.0	0.8	1019.6	-9.8	0.0	0.7
34996	2017	2	26	4	18.0	40.0	9.0	102.0	100.0	2.0	-0.8	1020.6	-6.6	0.0	1.1
35020	2017	2	27	4	94.0	131.0	20.0	106.0	2400.0	5.0	1.8	1019.3	-6.8	0.0	1.1
35044	2017	2	28	4	13.0	21.0	14.0	45.0	500.0	66.0	7.0	1016.0	-9.4	0.0	1.9

1461 rows × 15 columns

Fig.2: The main Dataset of 4th Hour or 4 A.M

2. Then we checked the values if there any null value, if Null (NaN) value occurred we replaced it with 0.



	year	month	day	hour	PM2.5	PM10	SO2	NO2	CO	O3	TEMP	PRES	DEWP	RAIN	WSPM
4	2013	3	1	4	3.0	3.0	12.0	12.0	300.0	72.0	-2.0	1025.2	-19.5	0.0	2.0
28	2013	3	2	4	3.0	7.0	18.0	43.0	400.0	43.0	-1.5	1030.8	-17.7	0.0	0.9
52	2013	3	3	4	106.0	136.0	79.0	92.0	2799.0	2.0	-4.3	1016.9	-11.8	0.0	0.9
76	2013	3	4	4	7.0	18.0	14.0	0.0	400.0	42.0	6.0	1018.0	-11.6	0.0	1.0
100	2013	3	5	4	115.0	151.0	76.0	92.0	2299.0	84.0	1.8	1013.7	-9.5	0.0	1.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
34948	2017	2	24	4	28.0	31.0	21.0	109.0	700.0	2.0	-2.1	1021.7	-8.0	0.0	0.6
34972	2017	2	25	4	13.0	16.0	4.0	47.0	300.0	44.0	0.8	1019.6	-9.8	0.0	0.7
34996	2017	2	26	4	18.0	40.0	9.0	102.0	100.0	2.0	-0.8	1020.6	-6.6	0.0	1.1
35020	2017	2	27	4	94.0	131.0	20.0	106.0	2400.0	5.0	1.8	1019.3	-6.8	0.0	1.1
35044	2017	2	28	4	13.0	21.0	14.0	45.0	500.0	66.0	7.0	1016.0	-9.4	0.0	1.9

1461 rows × 15 columns

Fig.3: The main Dataset of 4th Hour or 4 A.M After replacing the NaN to 0

3. Then again check for the NaN (if any) present.

4. Then we make a elbow graph to check the epsilon value using K- nearest neighbor algorithm

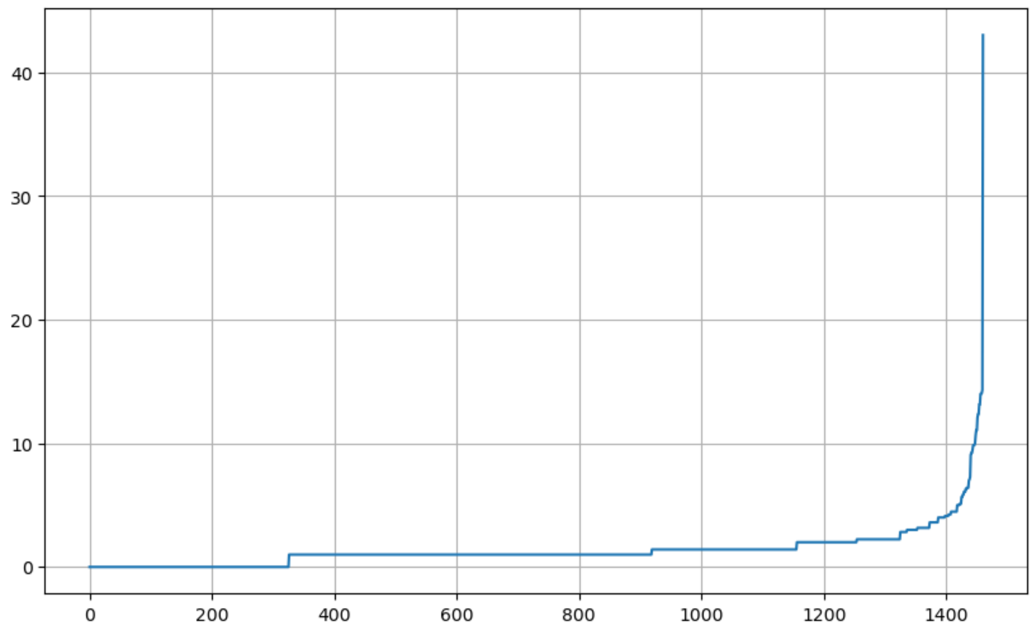
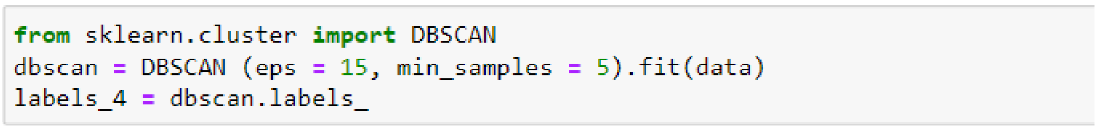


Fig.4 : Elbow graph to detect pesilon value

5. The first elbow like curve is where the epsilon value lies



```
from sklearn.cluster import DBSCAN
dbscan = DBSCAN (eps = 15, min_samples = 5).fit(data)
labels_4 = dbscan.labels_
```

Fig.5: DBSCAN Algorithm Implementation

6. Then we implemented DBSCAN algorithm and fit the data to plot it in a 2D graph.

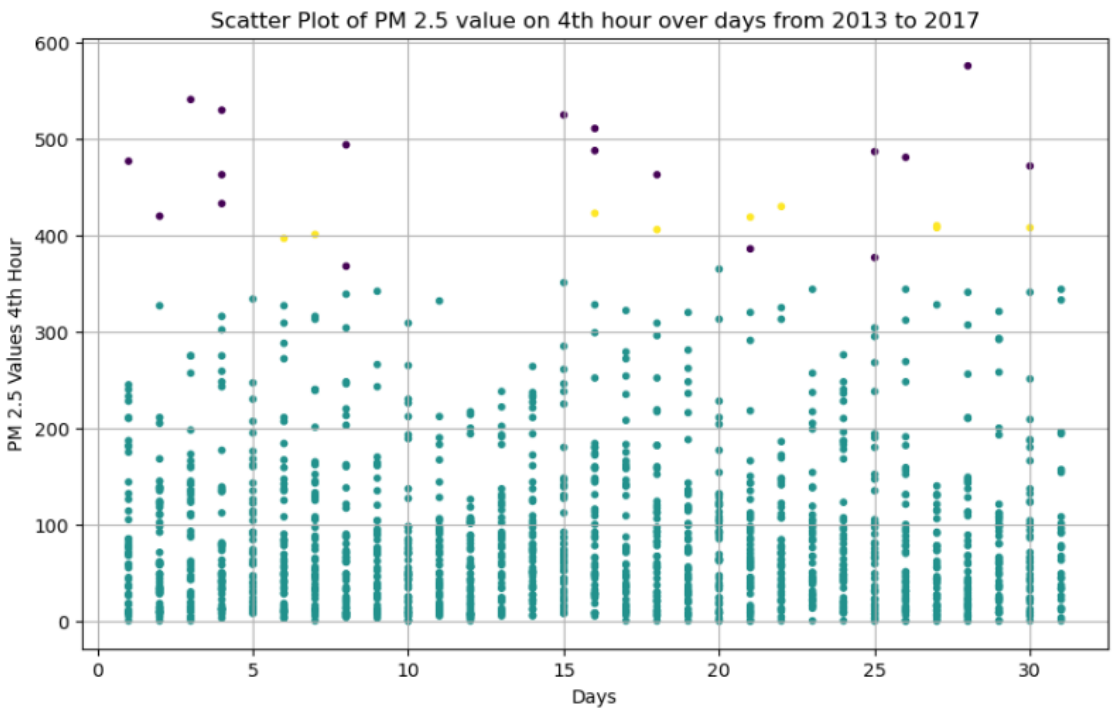


Fig.6: Scatter Graph to show the different clusters and noise

7. Then we check for the unique labels that returns an array. Here -1 is the index where all the noise value is present

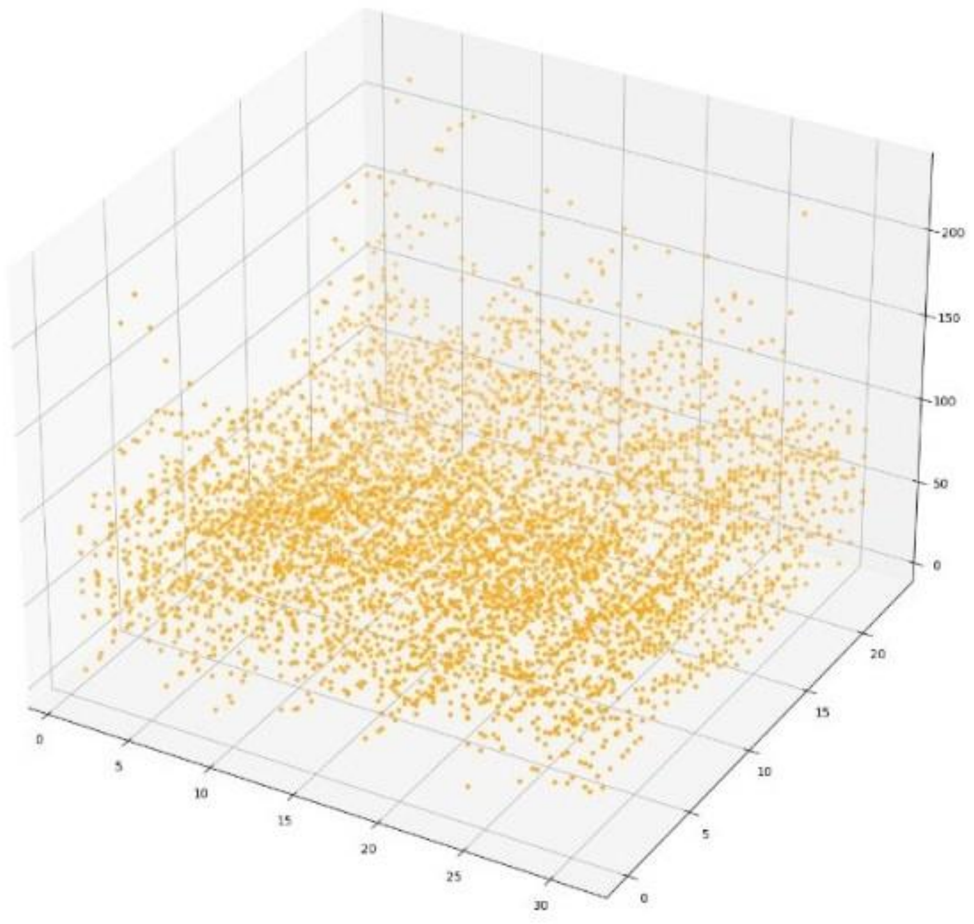
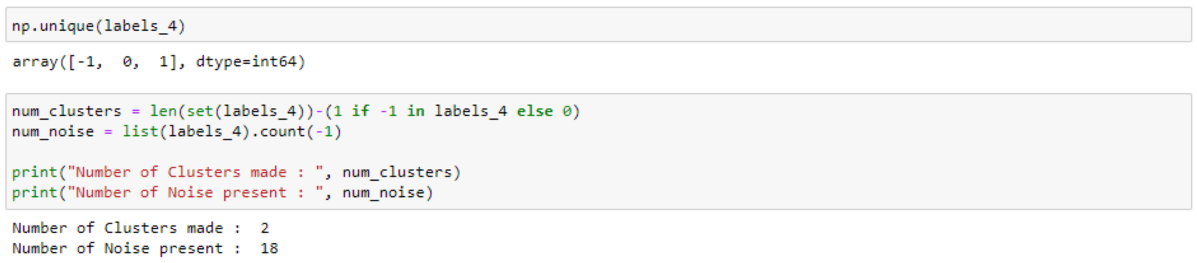


Fig.7: 3D plot of Day, Particular Hour and Working column value

8. Then we check the number of cluster and number of Noise in the algorithm implemented



```
np.unique(labels_4)
```

```
array([-1,  0,  1], dtype=int64)
```

```
num_clusters = len(set(labels_4))-(1 if -1 in labels_4 else 0)
num_noise = list(labels_4).count(-1)

print("Number of Clusters made : ", num_clusters)
print("Number of Noise present : ", num_noise)
```

```
Number of Clusters made :  2
Number of Noise present :  18
```

Fig.8: Checking the noise and the Cluster values

9. Then we plotted the noise value for sake of visualization

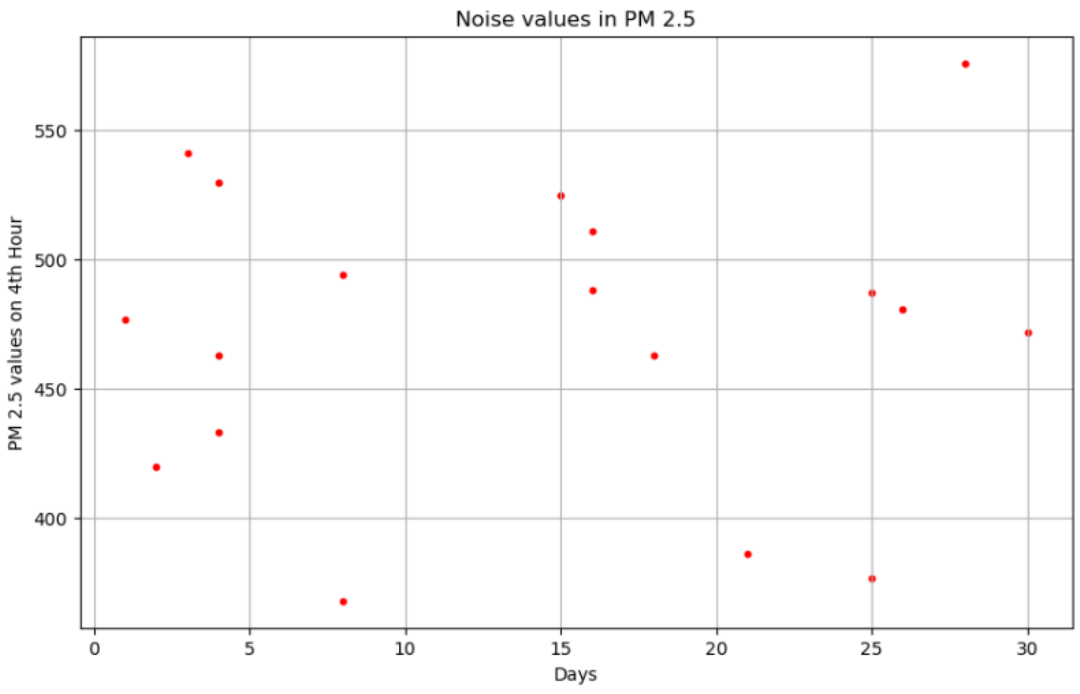
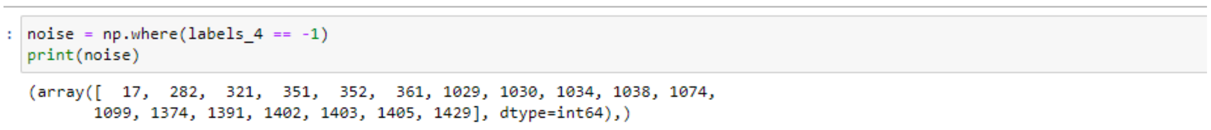


Fig.9: Visualisation of the noise points

10. Finally we are checking that which row lies under '-1' where the noise is present



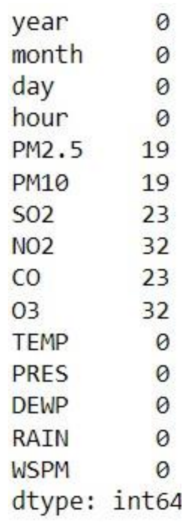
```
: noise = np.where(labels_4 == -1)
  print(noise)

  (array([  17,  282,  321,  351,  352,  361, 1029, 1030, 1034, 1038, 1074,
         1099, 1374, 1391, 1402, 1403, 1405, 1429], dtype=int64),)
```

Fig.10: Rows where noise values are occurs

Gradient Boosting Regressor implementation:

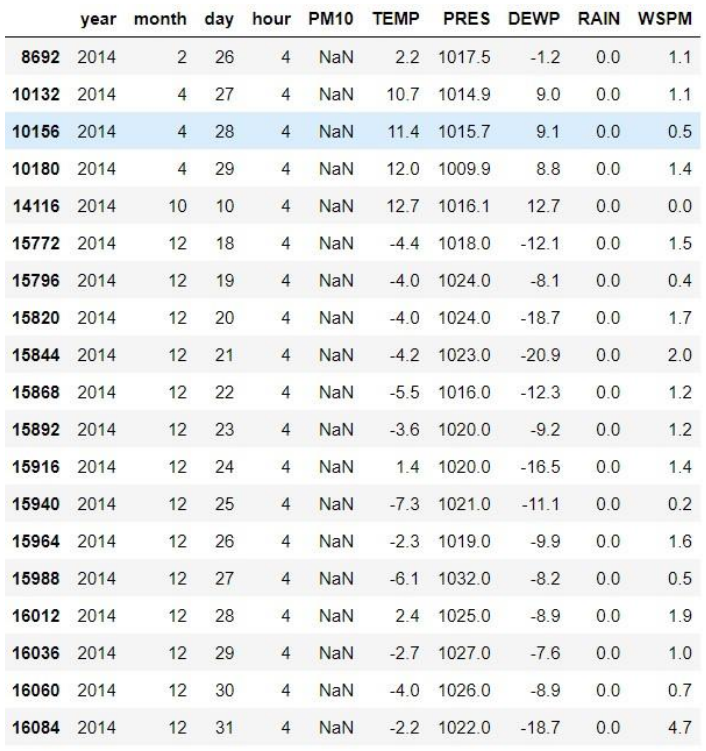
1. We first check the dataset for the presence of any null values, then we choose one column whose value needs to be predicted and remove any other null columns from the dataset.



```
year      0
month     0
day       0
hour      0
PM2.5    19
PM10     19
SO2      23
NO2      32
CO       23
O3       32
TEMP      0
PRES      0
DEWP      0
RAIN      0
WSPM      0
dtype: int64
```

Fig.11: Checking where the erronous value is present

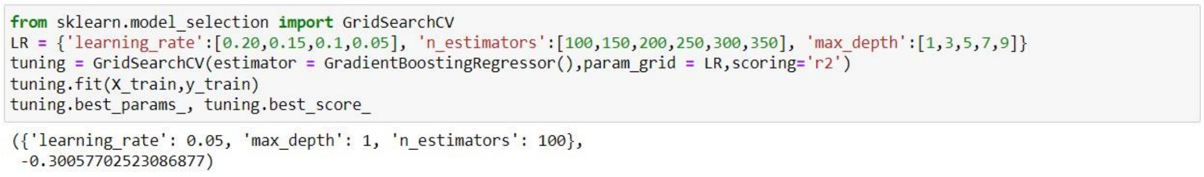
2. The dataset is then divided into train and test, with the dataset containing NaN values being placed under test data and the remaining dataset being utilised for training.



	year	month	day	hour	PM10	TEMP	PRES	DEWP	RAIN	WSPM
8692	2014	2	26	4	NaN	2.2	1017.5	-1.2	0.0	1.1
10132	2014	4	27	4	NaN	10.7	1014.9	9.0	0.0	1.1
10156	2014	4	28	4	NaN	11.4	1015.7	9.1	0.0	0.5
10180	2014	4	29	4	NaN	12.0	1009.9	8.8	0.0	1.4
14116	2014	10	10	4	NaN	12.7	1016.1	12.7	0.0	0.0
15772	2014	12	18	4	NaN	-4.4	1018.0	-12.1	0.0	1.5
15796	2014	12	19	4	NaN	-4.0	1024.0	-8.1	0.0	0.4
15820	2014	12	20	4	NaN	-4.0	1024.0	-18.7	0.0	1.7
15844	2014	12	21	4	NaN	-4.2	1023.0	-20.9	0.0	2.0
15868	2014	12	22	4	NaN	-5.5	1016.0	-12.3	0.0	1.2
15892	2014	12	23	4	NaN	-3.6	1020.0	-9.2	0.0	1.2
15916	2014	12	24	4	NaN	1.4	1020.0	-16.5	0.0	1.4
15940	2014	12	25	4	NaN	-7.3	1021.0	-11.1	0.0	0.2
15964	2014	12	26	4	NaN	-2.3	1019.0	-9.9	0.0	1.6
15988	2014	12	27	4	NaN	-6.1	1032.0	-8.2	0.0	0.5
16012	2014	12	28	4	NaN	2.4	1025.0	-8.9	0.0	1.9
16036	2014	12	29	4	NaN	-2.7	1027.0	-7.6	0.0	1.0
16060	2014	12	30	4	NaN	-4.0	1026.0	-8.9	0.0	0.7
16084	2014	12	31	4	NaN	-2.2	1022.0	-18.7	0.0	4.7

Fig.12: Implementation of GBA to PM 10

3. The tuning of hyperparameters for the implementation of the gradient boosting algorithm is carried out using the GridSearchCV approach. The best parameters are chosen by GridSearchCV from a list of potential values.

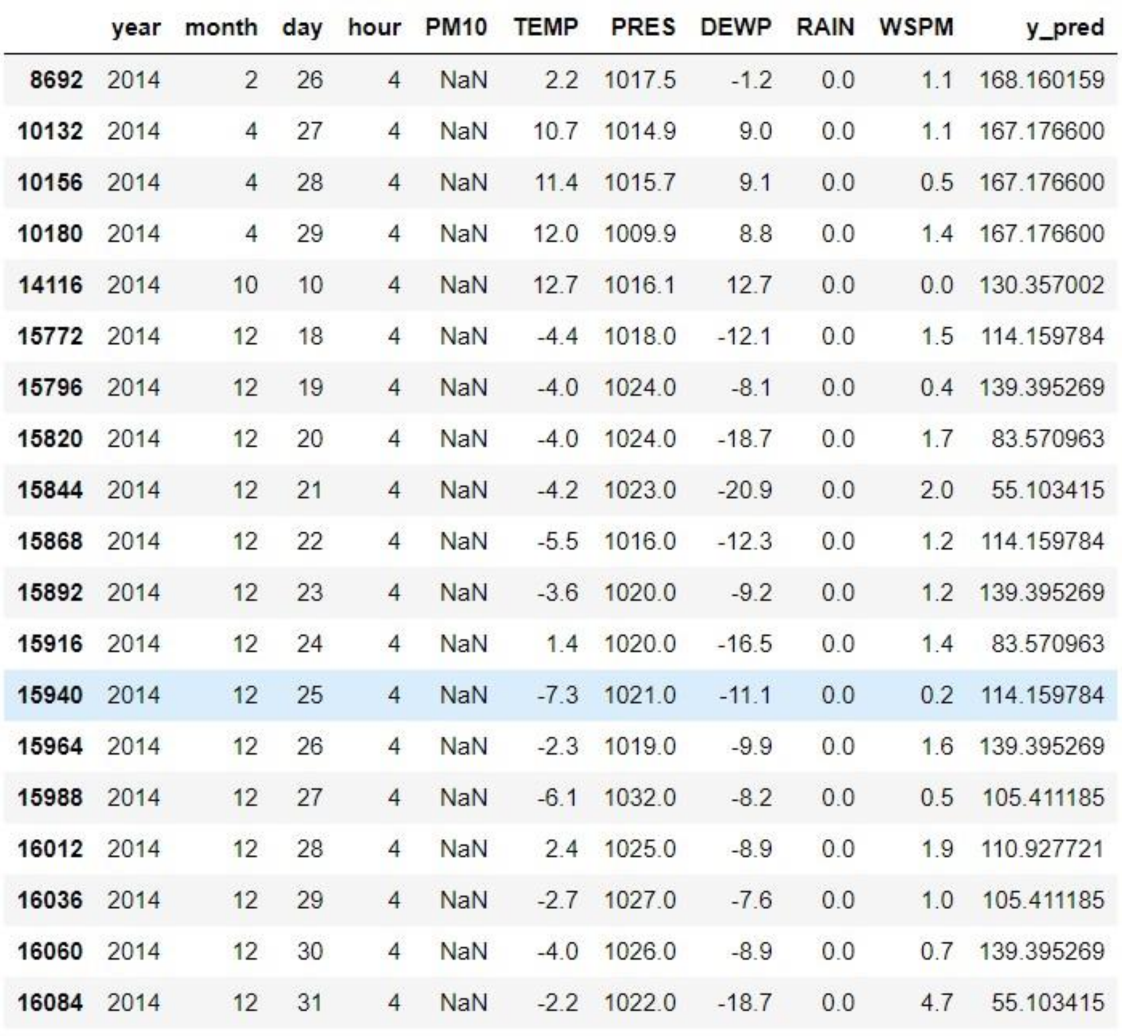


```python
from sklearn.model_selection import GridSearchCV
LR = {'learning_rate':[0.20,0.15,0.1,0.05], 'n_estimators':[100,150,200,250,300,350], 'max_depth':[1,3,5,7,9]}
tuning = GridSearchCV(estimator = GradientBoostingRegressor(),param_grid = LR,scoring='r2')
tuning.fit(X_train,y_train)
tuning.best_params_, tuning.best_score_
```

```
({'learning_rate': 0.05, 'max_depth': 1, 'n_estimators': 100},
 -0.30057702523086877)
```

Fig.13: Extracting the learning rate to train dataset

4. Finally, we use the Gradient Boosting Regressor along with the parameters given by GridSearchCV and the fit method is implemented and prediction is done using the .predict() method.



	year	month	day	hour	PM10	TEMP	PRES	DEWP	RAIN	WSPM	y_pred
8692	2014	2	26	4	NaN	2.2	1017.5	-1.2	0.0	1.1	168.160159
10132	2014	4	27	4	NaN	10.7	1014.9	9.0	0.0	1.1	167.176600
10156	2014	4	28	4	NaN	11.4	1015.7	9.1	0.0	0.5	167.176600
10180	2014	4	29	4	NaN	12.0	1009.9	8.8	0.0	1.4	167.176600
14116	2014	10	10	4	NaN	12.7	1016.1	12.7	0.0	0.0	130.357002
15772	2014	12	18	4	NaN	-4.4	1018.0	-12.1	0.0	1.5	114.159784
15796	2014	12	19	4	NaN	-4.0	1024.0	-8.1	0.0	0.4	139.395269
15820	2014	12	20	4	NaN	-4.0	1024.0	-18.7	0.0	1.7	83.570963
15844	2014	12	21	4	NaN	-4.2	1023.0	-20.9	0.0	2.0	55.103415
15868	2014	12	22	4	NaN	-5.5	1016.0	-12.3	0.0	1.2	114.159784
15892	2014	12	23	4	NaN	-3.6	1020.0	-9.2	0.0	1.2	139.395269
15916	2014	12	24	4	NaN	1.4	1020.0	-16.5	0.0	1.4	83.570963
15940	2014	12	25	4	NaN	-7.3	1021.0	-11.1	0.0	0.2	114.159784
15964	2014	12	26	4	NaN	-2.3	1019.0	-9.9	0.0	1.6	139.395269
15988	2014	12	27	4	NaN	-6.1	1032.0	-8.2	0.0	0.5	105.411185
16012	2014	12	28	4	NaN	2.4	1025.0	-8.9	0.0	1.9	110.927721
16036	2014	12	29	4	NaN	-2.7	1027.0	-7.6	0.0	1.0	105.411185
16060	2014	12	30	4	NaN	-4.0	1026.0	-8.9	0.0	0.7	139.395269
16084	2014	12	31	4	NaN	-2.2	1022.0	-18.7	0.0	4.7	55.103415

Fig.14: Predicting the data in y_pred column

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1. CONCLUSION

In order to increase the precision of weather forecasts and lessen the effects of weather-related disasters, a weather missing data prediction project might be an extremely useful tool. The initiative can assist close data gaps and provide more thorough and accurate information about weather conditions by forecasting missing data values in meteorological databases.

The project not only forecasts missing data values but also offers insights into the patterns and trends in meteorological data, which can help us better understand the natural world and inform our decision-making.

The importance of this kind of undertaking will only increase if weather patterns become more erratic and intense. The project's ability to forecast missing data values is complemented by its capacity to offer insights into the patterns and trends in meteorological data, which can enhance our comprehension of the natural world and inform decision-making.

7.2. FUTURE SCOPE:

Prediction of missing data is frequently simply one step in a wider machine learning pipeline. Other models can be fed the anticipated missing data to create predictions or produce insights. The missing data must frequently be predicted in real-time for many applications. For the purpose of making predictions almost instantly, the models can be accelerated and made more precise.

In the fields of machine learning and data science, numerous novel techniques are being developed. To increase the precision of the predictions, the missing data prediction project can investigate novel approaches like deep learning or reinforcement learning.

Weather prediction algorithms can be tailored to give industry-specific forecasts that assist businesses and governments in planning for and reducing weather-related risks using the entire dataset along with the predicted data.

To give real-time information regarding weather conditions and their effects on traffic, public transit, and other city services, weather prediction models can be integrated with smart city efforts.

Storms with extreme weather, such hurricanes, tornadoes, and floods, can result in considerable property damage and fatalities. Accurate and early warning of these disasters can assist individuals and governments in taking the proper measures. To more accurately forecast extreme weather events and offer more precise information about their possible effects, weather prediction models can be enhanced.

BIBLIOGRAPHY

[1] Zhu, D., Cai, C., Yang, T., & Zhou, X. (2018). A Machine Learning Approach for Air Quality Prediction: Model Regularization and Optimization. *Big Data and Cognitive Computing*, *2*(1), 5. https://doi.org/10.3390/bdcc2010005

[2] Doreswamy, I. Gad and B. R. Manjunatha, "Performance evaluation of predictive models for missing data imputation in weather data," 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 2017, pp. 1327-1334, doi: 10.1109/ICACCI.2017.8126025.

[3] Madhuri, V. M., Gunjal GH Samyama, and Savitha Kamalapurkar. "Air pollution prediction using machine learning supervised learning approach." Int J Sci Technol Res 9.4 (2020): 118-123.

[4] Bhalgat, Pooja, Sachin Bhoite, and Sejal Pitare. "Air quality prediction using machine learning algorithms." International Journal of Computer Applications Technology and Research 8.9 (2019): 367-370.

[5] scikit-learn, machine learning in python. URL https://scikit-learn.org/stable/modules/model_evaluation.html#clustering-metrics

[6] Anghel, A., Papandreou, N., Parnell, T., De Palma, A., & Pozidis, H. (2018). Benchmarking and Optimization of Gradient Boosting Decision Tree Algorithms. ArXiv. /abs/1809.04559

[7] Singh, Upma, Mohammad Rizwan, Muhannad Alaraj, and Ibrahim Alsaidan. 2021. "A Machine Learning-Based Gradient Boosting Regression Approach for Wind Power Production Forecasting: A Step towards Smart Grid Environments" Energies 14, no. 16: 5196. https://doi.org/10.3390/en14165196

[8] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. ACM Trans. Database Syst. 42, 3, Article 19 (September 2017), 21 pages. https://doi.org/10.1145/3068335