

Air Quality Prediction using Machine Learning

ABSTRACT

In the last several years, air pollution has risen steadily in urban environments. Cities like Gurugram, Faisalabad, Delhi, Beijing are few of the world's most polluted cities and have seen a dangerous rise in air pollution levels. Forecasting is important because of the human, ecologic and economic toll of pollution, and is a useful investment at individual and community levels. Accurate forecasting will help us plan in advance, decreasing the effects on health and the costs associated. Local weather conditions strongly affect air pollution levels. Generating deterministic models to study air pollutant behavior in environmental science research is often not very accurate because they are complex and need simulation at the molecular interaction level. : Air pollution which is detrimental to people's health is a wide spread problem across many countries around the world. Developing better air quality prediction approaches is an important research issue. Here comes machine learning to the rescue with high computing facilities to predict air pollution.

So we proposed a system using machine learning techniques and algorithms like Linear Regresson and Random Forest Regressor to predict the air quality index and also classifies air quality is good, moderate, unhealthy ,unhealthy for strong people and hazardous based on the user entered new input data.

INTRODUCTION

With the development of the economy and society all over the world, most metropolitan cities are experiencing elevated concentrations of ground-level air pollutants, especially in fast developing countries like India and China. Air pollution is one of the main detriments to human health. According to World Health Organization, 7 million people are at health risk due to air pollution . It is a leading risk factor for majority of health problems like asthma, skin infections, heart issues, throat and eye diseases, bronchitis, lungs cancer and respiratory system's diseases. Besides the health problems related to air pollution, it also poses a serious threat to our planet. Pollution emissions from the sources like vehicles and industry is the underlying cause of greenhouse effect, CO₂ emissions are amongst the foremost contributors to the greenhouse phenomenon. Climate change has been widely discussed at the global forums and has remained a burning issue for the world since last two decades as a result of increased smog and ozone damage Exposure to air pollution can affect everyone, but it can be particularly harmful to people with a heart disease or a lung condition, elderly people and children. Studies show that long-term exposure to fine particulate air pollution or traffic-related air pollution is associated with environmental-cause mortality, even at concentration ranges well below the standard annual mean limit value. Damages produced by air pollution include its impact on the depletion of the ozone layer, which plays a significant role in preserving the planet from ultraviolet radiation. Air pollution also causes acid rain that is harmful to soil, trees, and wildlife. Urban smog and global climate change are primarily caused by the pollution of air. Also, air pollution problem can directly impacts humans and it can be the source for

long- and short-term health effects with the most affected people being children and the elderly. The reverse impacts of exposure to the air pollutants on health are tremendous such as short term effects to the eyes, throat, and nose as well as causing headaches, upper respiratory infections, and allergic reactions. Long-term effects may include brain damage, lung cancer, liver damage, and heart disease. Therefore, building an early warning system, which provides precise forecast and also alerts health alarm to local inhabitants will provide valuable information to protect humans from damage by air pollution. Air quality is an active topic at many social and political scales around the world. It is a significant concern for governments, environmentalists, and even data scientists who are raising awareness about this growing global problem. The availability of the massive amount of data in recent years enables better predictions of air quality using machine learning techniques.

So we proposed a system using machine learning techniques and algorithms like Linear Regression and Random Forest Regressor to predict the air quality index and also classifies air quality is good, moderate, unhealthy ,unhealthy for strong people and hazardous based on the user entered new input data.

MOTIVATION

Monitoring air quality is one of the best ways to prevent the harmful effects of air pollution. Having the information about the quality of air can lead to formulating suggestions and data driven recommendations to mitigate the possible harmful effects it can bring.

Objective

The main aim of our project is efficient to predict the air quality index and also classify the air quality based on predicted air quality index value using various machine learning techniques and algorithms like Linear Regression and Random Forest Regressor.

Problem Definition:

Seven of the world's most polluted cities are in India. And the severe problems in popular cities due to excessive air pollution are nothing new. In order to reduce this air pollution, predicting the air quality in early stage accurately helps to take the better decisions to maintain the air quality.

Existing System

Currently, two major approaches are used to forecast real-time air quality: simple empirical approaches, advanced physically-based approaches. Simple empirical approaches like persistence method and climatology method are based on assumptions or hypothesis that thresholds of forecasted meteorological variables can indicate future pollution level . They are computationally fast but have low accuracy and are primarily used as references by other methods. Advanced physically-based. approaches like chemical transport models (CTMs) simulate the formation and accumulation of air pollutants by a solution of the conservation equations and transformation relationships among the mass of various chemical species and physical states. They can provide valuable insights for understanding pollutant diffusion mechanisms. But they are computationally expensive, demanding reliable meteorological predictions and highly relevant to a high-level of expertise.

Proposed System:

We proposed a system using machine learning which predicts the air quality index efficiently and also classifies the air quality based on predicted air quality index value this helps to take the appropriate decisions in order to maintain the air quality. Air pollutant prediction is done using Machine learning techniques and algorithms such as Linear Regression and Random Forest Regressor and we will develop a front end web page and where user can enter different values of input required for air quality index prediction and the predicted output is displayed on the front end both air quality index value as well as quality of air that is good or moderate or unhealthy or hazardous.

Literature survey

- **T. M. Amado and J. C. Dela Cruz :** The aim of this paper is to find an alternative way of monitoring and characterizing air quality through the use of integrated gas sensors and building predictive models using machine learning algorithms that can be used to obtain datadriven solutions to mitigate the risk of air pollution. The proposed methodology is implemented by building a prototype for the integrated sensors using DHT 11 temperature and relative humidity sensor, MQ2, MQ5 and MQ135 gas sensors. Five predictive models are developed in the study, k-nearest neighbors (KNN), support vector machine (SVM), Naïve-Bayesian classifier, random forest and neural network. Results show that the researchers are able to obtain an accuracy of 78.67%, 76.78%, 81.67%, 74.22%, and 80.56% for all the five models respectively, having the neural network to be the best performing model.
- **Problem :** To find an alternative way of monitoring and characterizing air quality through the use of integrated gas sensors and building predictive models using machine learning algorithms that can be used to obtain data driven solutions to mitigate the risk of air pollution

- **Solution :** This paper proposes a methodology of characterizing the air quality by building predictive models that relates the sensor values to air quality index.

- **Drawback :** The solution given in this paper did not give good accuracy.

➤ **H. Zheng, Y. Cheng and H. Li :** . In this paper, near future fine-grained air quality level prediction task is explored with a series of machine learning ensemble methods. Included ensemble methods are majority voting, averaging, weighted averaging and 16 different stacking tactics. To investigate the performances of these ensemble methods, comprehensive comparative experiments are conducted. Included contrast models are classical Autoregressive Integrated Moving Average (ARIMA), popular deep learning model Long Short-Term Memory (LSTM) neural network, and nine of the base-level models such as Support Vector Machine (SVM), Random Forest (RF), Logistic Regression (LR) and several boosting models. Datasets acquired from a coastal city Hong Kong and an inland city Beijing are used to train and validate all the models. Experiments show that performances of the ensemble methods outperform most of the individual models, especially when stacking with probability distributions together with engineered original features, which demonstrates the best performance.

- **Problem :** Developing better air quality prediction because Existing methods often focus on the prediction of air pollution concentrations, which is not as intuitive to the public as the air quality levels.
- **Solution :** In this paper, near future fine-grained air quality level prediction task is explored with a series of machine learning ensemble

methods. Included ensemble methods are majority voting, averaging, weighted averaging and 16 different stacking tactics.

- **Drawback :** The solution given in this paper did not give good accuracy. All the models perform poorly over long term prediction tasks, models which can capture the long term trends should be further researched.

➤ **K. Nandini and G. Fathima :** In this work, air pollutant prediction is done using Machine learning techniques. K Means algorithm is used for clustering and different classifiers such as Multinomial Logistic Regression and Decision Tree algorithms are used to analyze the results based on available data in the R programming language. The results obtained using classifiers are compared based on error rate and accuracy. The multinomial logistic regression model has given high accuracy compared to decision tree model. In this work, the probabilistic model is used to reveal the values in cities in terms of different parameters. In most cases it's not possible to visit venues and obtaining the values easily. Hence, based upon the iteration values obtained from the multinomial logistic regression model the highly concentrated pollutant is been analyzed.

- **Problem :** Air pollution is one of the influential factors that can affect the quality of every living being in the environment. Monitoring the air pollution is a scathing issue.
- **Solution :** . In this work, air pollutant prediction is done using Machine learning techniques. K Means algorithm is used for clustering and different classifiers such as Multinomial Logistic Regression and

Decision Tree algorithms are used to analyze the results based on available data in the R programming language

- **Drawback :** The solution given in this paper did not give good accuracy. All the models perform poorly over long term prediction tasks.

- **R. Yang, H. Zhou and D. Ding :** This paper is a combination of machine learning classification algorithms in the computer field, urban housing in the real estate sector, and air quality assessment in the field of environmental protection. Using data mining ideas and technologies that are now widely considered by people, we use three machine learning algorithms with good classification performance (SVM, naive Bayes, KNN) to predict the air quality of urban residential communities. Finally, the experiment proves the feasibility and stability of this method, which can meet the actual needs of home buyers in real estate.

- **Problem :** Air pollution is one of the influential factors that can affect the quality of every living being in the environment. Predicting the air quality of urban residential communities.

- **Solution :** Air quality in urban residential district is predicted by feature variables. The method is very practical and stable for air quality in urban residential district prediction.

- **Drawback :** The solution given in this paper did not give good accuracy.

- **U. Mahalingam, K. Elangovan, H. Dobhal, C. Valliappa, S. Shrestha and G. Kedam :** This paper addresses the challenge of predicting the Air Quality Index (AQI), with the aim to minimize the pollution before it gets adverse, using two Machine Learning Algorithms: Neural Networks and Support Vector Machines. The air pollution databases were extracted from the Central Pollution Control Board (CPCB), Ministry of Environment, Forest and Climate change, Government of India. The proposed Machine Learning (ML) model is promising in prediction context for the Delhi AQI. The results show improvement of the prediction accuracy and suggest that the model can be used in other smart cities as well. In this research work, a Machine Learning model for Air Quality Index prediction for smart cities is proposed. The model is tested with the Delhi Air Quality data because of the notoriety of Delhi's air pollution. By using the Neural Networks and SVM, AQI is predicted successfully with 91.62% of accuracy for Neural Networks and 97.3% for Support Vector Machines. Six Support Vector Machine functions were used to predict accuracy, and it was learnt that the "Medium Gaussian SVM" gives the maximum accuracy of 97.3%.
- **Problem :** Air pollution has been a severe problem in the major smart cities like New Delhi. The values of certain parameters of air quality are higher than the average level so it causes damage to human health. The local and state governments have taken some actions for air pollution in rural and urban areas but it is not recovered completely. A widespread intrusion of moisture over the Northern Indian region including Delhi is always anticipated due to an active western disturbance. If the western disturbance causes a sufficient amount of rain then the National Air Quality Index is expected to move towards poor range otherwise little shower or pre-shower would disturbs the adverse meteorological conditions by increasing the AQI.

- **Solution :** In this paper, a Machine Learning model for Air Quality Index prediction for smart cities is proposed. The model is tested with the Delhi Air Quality data because of the notoriety of Delhi's air pollution. By using the Neural Networks and SVM, AQI is predicted successfully with 81.62% of accuracy for Neural Networks and 77.3% for Support Vector Machines
 - **Drawback :** The solution given in this paper did not give good accuracy.
- **S. Mahanta, T. Ramakrishnudu, R. R. Jha and N. Tailor :** This paper investigates how effective some available prediction models are in predicting the Air Quality Index(AQI) values given some input data, based on the pollution and meteorological information in New Delhi, India. We perform regression analysis on the dataset, and our results show which meteorological factors affect the AQI values more and how useful the predictive models are to help in air quality forecasting. In this paper, we focused on testing the usefulness of existing regression models in the sklearn library to predict the air quality index values given the past weather data. And we also tried to figure out which features are most useful for the prediction as depicted by some of those models. The results obtained show that most models achieve a decent accuracy of almost 85 %, with the highest being the Extra Trees regression model, which means that the models are quite useful as predictors.
- **Problem :.**Generating deterministic models to study air pollutant behavior in environmental science research is often not very accurate because they are complex and need simulation at the molecular interaction level.

- **Solution :** This paper investigates how effective some available prediction models are in predicting the Air Quality Index(AQI) values given some input data, based on the pollution and meteorological information in New Delhi, India.
- **Drawback :** The solution given in this paper did not give good accuracy because real-time and historic traffic data is not concatenated with the weather data for AQI prediction.
- **H. Ayyalasomayajula, E. Gabriel, P. Lindner and D. Price :** . This paper presents a comparison of the Hadoop MapReduce and Spark programming models for air quality simulations, guiding future code development for the research groups interested in these analyses. Two use cases have been used, namely (i) calculating the eight hour rolling average of pollutants in a restricted region; (ii) identifying clusters of sensors showing similar patterns in pollutant concentration over multiple years in the state of Texas. The data set used in this analysis is air pollution data collected over fifteen years at 179 monitor sites across the state of Texas for a variety of pollutants. Our results reveal 20–25% performance benefits for the Spark solutions over MapReduce. Furthermore, it documents performance benefits of the Spark MLlib machine learning library over the Mahout library which is based on the MapReduce programming model. This paper presented a comparison of the Hadoop MapReduce and Spark programming models and execution environment for air quality simulations, aiming to guide future code development for the research groups interested in these analyses. Our results reveal 20 – 25% performance benefits for the Spark solutions over MapReduce. Furthermore, it documents performance benefits of the Spark MLlib machine learning library over the Mahout library.

- **Problem** : There is a need to analyze larger timeframes across more locations to correlate long term developments for different pollutants with multiple serious health effects such as asthma.
 - **Solution** : This paper presents a comparison of the Hadoop MapReduce and Spark programming models for air quality simulations, guiding future code development for the research groups interested in these analyses. Two use cases have been used, namely (i) calculating the eight hour rolling average of pollutants in a restricted region; (ii) identifying clusters of sensors showing similar patterns in pollutant concentration over multiple years in the state of Texas.
 - **Drawback** : The solution given in this paper did not give good accuracy.
- **A. S. Alsaedi and L. Liyakathunisa** : In this study, they perform spatial and temporal analysis using Long-Short Term Memory (LSTM) neural networks to estimate the nitrogen dioxide concentration that is considered a dangerous air pollutant between Beijing and London. In their proposed approach, spatial and temporal data are collected, preprocessed, normalised, and classified with LSTM followed by a comparative analysis with alternate machine learning techniques. The results show that the performance from our adapted approach of LSTM is higher compared to other techniques for predicting pollution rates between London and Beijing
- **Problem** : Air quality is an active topic at many social and political scales around the world. It is a significant concern for governments, environmentalists, and even data scientists who are raising awareness about this growing global problem.

- **Solution :** In this study, they perform spatial and temporal analysis using Long-Short Term Memory (LSTM) neural networks to estimate the nitrogen dioxide concentration that is considered a dangerous air pollutant between Beijing and London. In our proposed approach, spatial and temporal data are collected, preprocessed, normalised, and classified with LSTM followed by a comparative analysis with alternate machine learning techniques.

- **Drawback :** The solution given in this paper did not give good accuracy.

- **B. Baran :** In this study, it is aimed to predict the Air Quality Index (AQI) by the Extreme Learning Machines (ELM) algorithm. For this purpose, six parameters have been selected which can affect the AQI. These are temperature, humidity, pressure, wind speed, PM10 and SO2 respectively. First of all, “Forecast Sheet” application that presented in the Excel environment and the correlation analysis were used to determine the relationship status between these six parameters and AQI. Thus, the linear equations between each parameter and AQI were determined. Also, R values were determined for relation rate. Then, a training data set was created. which includes six parameters, was determined as training data. As a result of cross-validation, the activation code and the number of neurons that achieved the highest success were applied to the actual test data. Finally, the classification values predicted by ELM and the mathematical classification values were compared. Thus, the success of the ELM was measured.

- **Problem : .** Air pollution is one of the influential factors that can affect the quality of every living being in the environment. Predicting air quality is very important.
 - **Solution :** In this study, it is aimed to predict the Air Quality Index (AQI) by the Extreme Learning Machines (ELM) algorithm. For this purpose, six parameters have been selected which can affect the AQI. These are temperature, humidity, pressure, wind speed, PM10 and SO2 respectively.
 - **Drawback :** The solution given in this paper did not give good accuracy.
- **S. Ameer :** In this paper, they have performed the pollution prediction using four advanced regression techniques and have presented a comparative study to analyze the best model for accurately predicting the air quality with reference to data size and processing time. They have used Apache Spark for conducting experiments and performing pollution estimation using multiple available data sets. Mean Absolute error (MAE) and Root Mean Square Error (RMSE) have been used as evaluation criteria for the comparison of regression models. Furthermore, the processing time of each technique by standalone learning and by fitting the hyperparameter tuning on Apache Spark has also been calculated to find the best-fitted model in terms of processing time and least error rate.
- **Problem : .** Dealing with air pollution is one of the major environmental challenges in a smart city environment. Real-time monitoring of pollution data enables the metropolitans to analyze the current traffic situation of the

city and take their decisions accordingly. Existing research has used different machine learning tools for pollution prediction; however, comparative analysis of these techniques is often required to have a better understanding of their processing time for multiple datasets.

- **Solution :** In this paper, they have performed the pollution prediction using four advanced regression techniques and have presented a comparative study to analyze the best model for accurately predicting the air quality with reference to data size and processing time. They have used Apache Spark for conducting experiments and performing pollution estimation using multiple available data sets. Mean Absolute error (MAE) and Root Mean Square Error (RMSE) have been used as evaluation criteria for the comparison of regression models
- **Drawback :** The solution given in this paper did not give good accuracy. Gradient boosting regression has performed worst as it has achieved highest processing time in almost all data sets and has given a very high error rate in most cases.
- **C. Srivastava, S. Singh and A. P. Singh :** In this paper, they implemented different classification and regression techniques like Linear Regression, SDG Regression, Random Forest Regression, Decision Tree Regression, Support Vector Regression, Artificial Neural Networks, Gradient Boosting Regression and Adaptive Boosting Regression to forecast the Air Quality Index of major pollutants like PM_{2.5}, PM₁₀, CO, NO₂, SO₂ and O₃. The techniques are then evaluated using Mean square error, Mean absolute error and R² , which show that Support Vector Regression and Artificial Neural Networks are best suited for predicting the air quality .

- **Problem** : Urban air pollution prediction becomes an indispensable alternative to curb its detrimental consequences.
 - **Solution** : In this paper, they implemented different classification and regression techniques like Linear Regression, SDG Regression, Random Forest Regression, Decision Tree Regression, Support Vector Regression, Artificial Neural Networks, Gradient Boosting Regression and Adaptive Boosting Regression to forecast the Air Quality Index of major pollutants like PM2.5, PM10, CO, NO2, SO2 and O3.
 - **Drawback** : The solution given in this paper did not give good accuracy. The dataset used in this study is for a shorter duration which limits the capability of the model. Hence, use of data records spanning longer durations with negligible data gaps is recommended for further improvisations.
- **S. Jeya and L. Sankari** : This paper attempts to forecast PM2.5 pollutant which is one of the detrimental diseases triggering pollutants throughout the globe by using bidirectional long short term memory model. The proposed model accuracy is comparatively greater than the existing model by evaluating the following error estimation metrics Root mean square error = 9.86, mean absolute error = 7.53, and symmetric mean absolute percentage error = 0.1664. This proposed model by using BILSTM predicted the PM2.5 with improved performance comparing the existing model and produced exceptional MAE, RMSE, and SMAPE. Exposure to PM2.5 when its level exceeds the limit is hazardous to humans and predicting pollutants with deep

learning algorithms accurately helps the government body to signal its citizens as well as cut or regularize the sources of pollutants to a great extent

- **Problem :** The impact of harmful pollutants in the air on human health is a vast area of research, preventing or controlling, and also monitoring the pollutant is the huge responsibility of any governing body. Several computing models starting from statistical and machine learning to deep learning have compared and contrasted to prove the accuracy of forecasting air quality standards until date. The level of pollutants is still not in control in several parts of the world due to various sources and reasons.
 - **Solution :** This paper attempts to forecast PM_{2.5} pollutant which is one of the detrimental diseases triggering pollutants throughout the globe by using bidirectional long short term memory model.
 - **Drawback :** The solution given in this paper did not give good accuracy.
- **K. Maheshwari and S. Lamba :** This paper explores patterns in Beijing's Particulate Matter 2.5 concentration and forecasts future concentrations. Air quality has been an enormous health concern in recent decades as the place has become further industrialized and more and more of its citizens have begun driving automobiles. The occurrence of air pollution takes place in the following ways. 1. release and generation of pollutants from their source. 2. carry of pollutants in the atmosphere. 3. penetrating and negatively impacting human health and ecosystems. We tend to minimise the effects of these emissions as there is no practical, economical or technical method for zero emissions. PM 2.5 is especially dangerous because it can pass through the human body's natural filters and enter the lungs. Health concerns related to PM 2.5 include heart and lung disease, asthma, bronchitis, and other

respiratory problems. Machine learning, as one of the most accepted techniques, is capable to efficiently train a model using regression models to predict the hourly air pollution concentration . Following six regressors chosen for this problem were Linear Regression, K-Nearest Neighbor, Stochastic Gradient Descent, DecisionTree, Random Forest and Multi-layer Perceptron. Although performance of all models was comparable, Multi-layer Perceptron Algorithm model successfully bring about better accuracy and true positive rate with 95.4 accuracy.

- **Problem :** Air quality is major influencer of ecology and health .The relationship between air pollution and mortality rate goes on back in the days. Accurate prediction of air quality is very important.
- **Solution :** This paper explores patterns in Beijing's Particulate Matter 2.5[7] concentration and forecasts future concentrations. Air quality has been an enormous health concern in recent decades as the place has become further industrialized and more and more of its citizens have begun driving automobiles. Following six regressors chosen for this problem were Linear Regression, K-Nearest Neighbor, Stochastic Gradient Descent, DecisionTree, Random Forest and Multi-layer Perceptron.
- **Drawback :** The solution given in this paper did not give good accuracy.

Drawback of Literature Survey

Currently, two major approaches are used to forecast real-time air quality: simple empirical approaches, advanced physically-based approaches. They are computationally fast but have low accuracy and are primarily used as references by other methods.

Advanced physically-based approaches like chemical transport models (CTMs) simulate the formation and accumulation of air pollutants. But they are computationally expensive, demanding reliable meteorological predictions and highly relevant to a high-level of expertise.

System Requirements:

Hardware Requirement:-

- System :Pentium IV 2.4 GHz.
- Hard Disk : 500 GB.
- Ram : 4 GB.
- Any desktop / Laptop system with above configuration or higher level.

Software Requirements:-

- Operating system : Windows XP / 7
- Coding Language :Python, HTML
- Version :Python 3.6.8
- IDE : Python 3.6.8 IDLE
- ML Packages :Numpy, Pandas ,Sklearn , Flask , PymySql.
- ML Algorithms: Random Forest Regressor and Linear Regression.
- Other Requirements : Notepad, XAMPP Control Panel

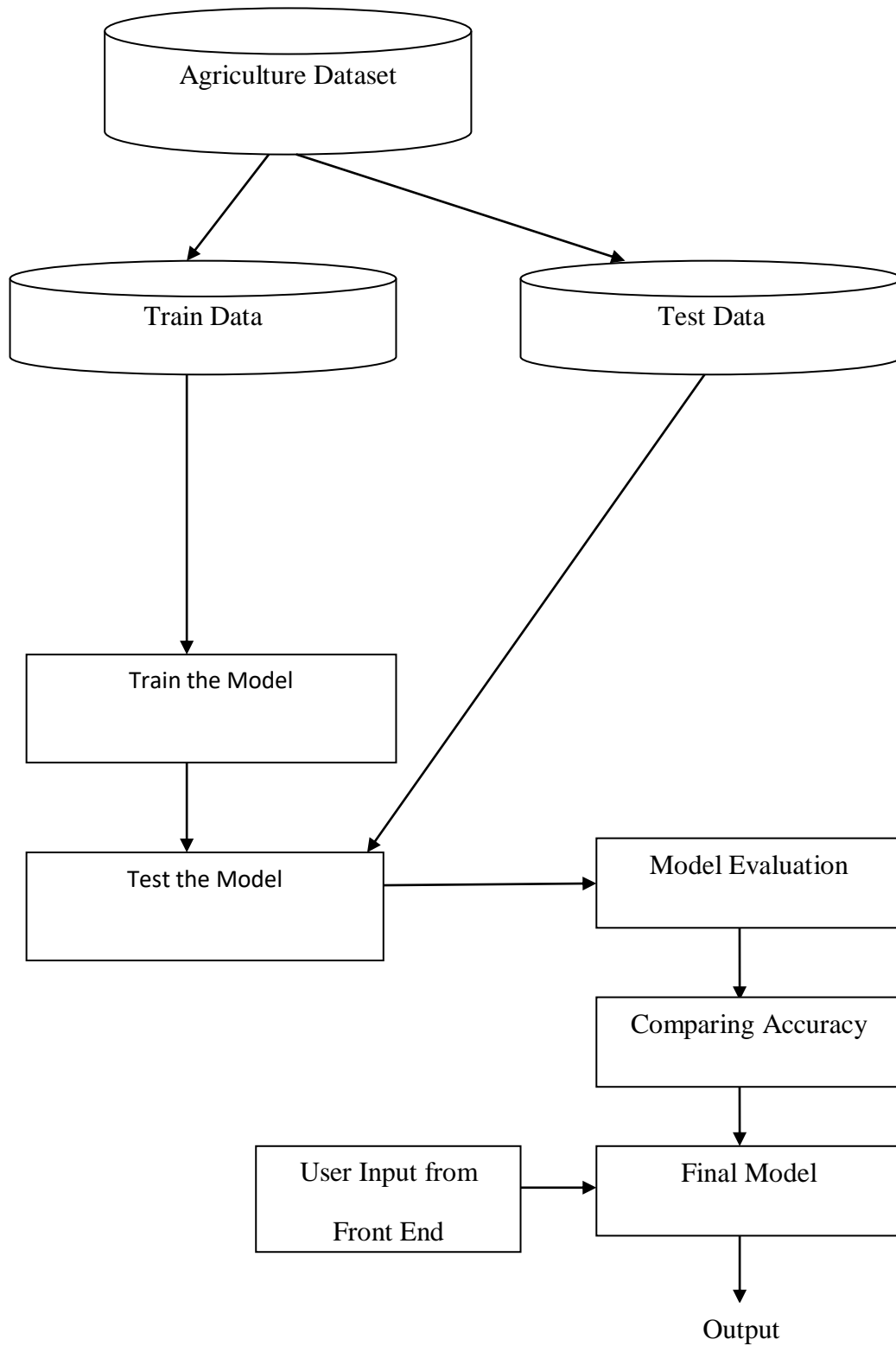
Methodology

- Air Quality dataset is taken.
- The dataset is loaded and preprocessed with various machine learning techniques.
- The preprocessed data is divided as training and testing data.
- The prediction model is built using machine learning algorithms Linear Regression and Random Forest Regressor.
- The model is trained using training dataset and once the model has been trained successfully it has to be tested.
- The trained model is tested using testing dataset and accuracy is calculated.
- The algorithm which gives the best accuracy is taken as our final prediction model.
- The finalized model is converted into pickle model (binary format data) and saved.
- A Front End is developed with the help of Flask and HTML.
- Now user will enter the 'Temperature', 'Humidity', 'WindSpeed', 'Visibility', 'Pressure', 'so2', 'no2', 'Rainfall', 'PM10', 'PM25' values in the front end.

- The collected data from the user is given as input to our finalized algorithm to predict the Air Quality Index.
- Finally the predicted Air Quality Index output is displayed on the front end and also displays the air quality is good, moderate, unhealthy ,unhealthy for strong people and hazardous based on predicted the air quality index value is the front end.

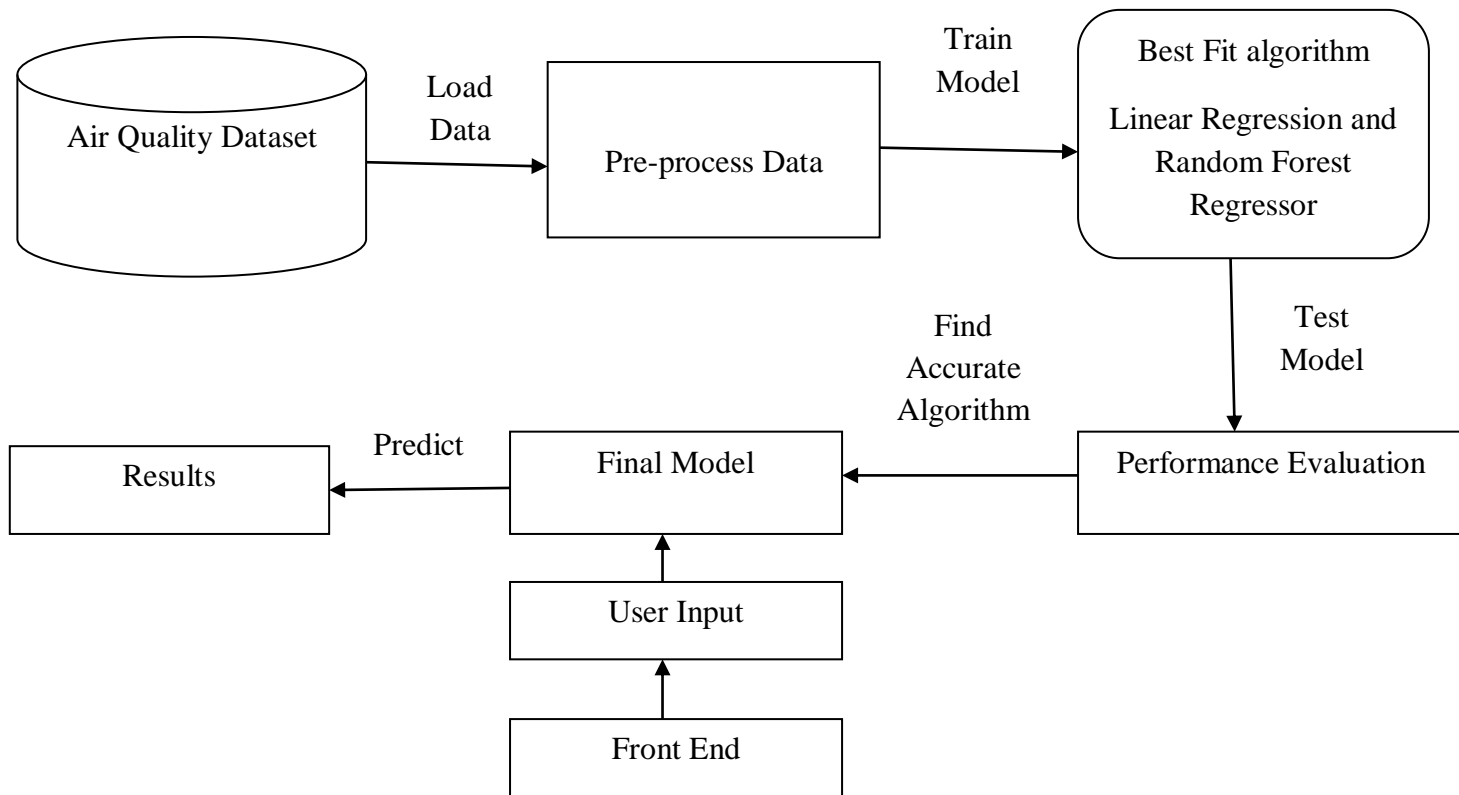
System Design:

System Architecture



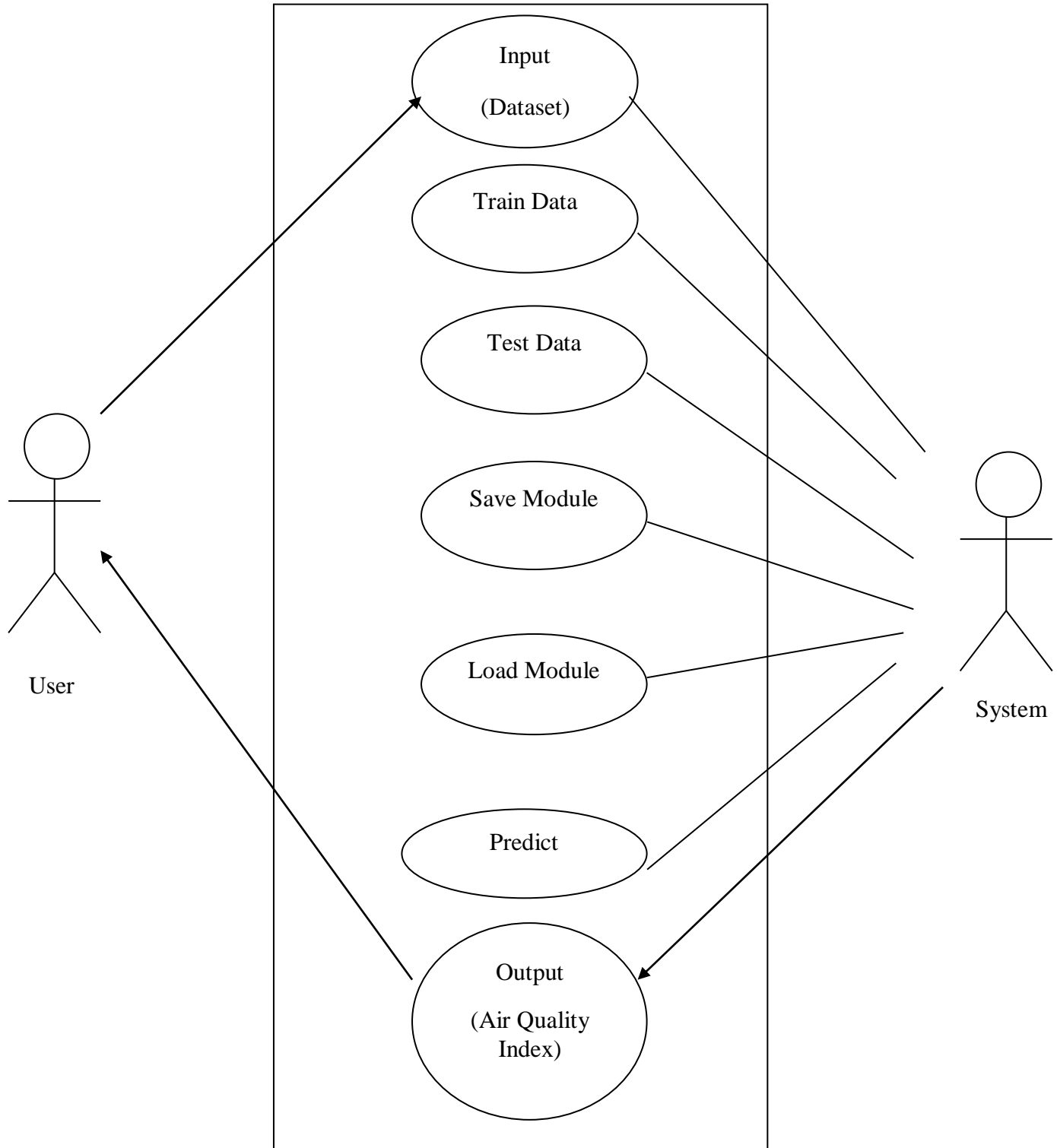
- Air Quality dataset is taken and loaded.
- The data is preprocessed to clean the data and understand the dataset.
- The data is split as training and testing data.
- The model is built using machine learning algorithms like Linear Regression and Random Forest Regressor.
- The model is trained with the preprocessed data.
- The model is tested and accuracy is calculated for different ML algorithms.
- The algorithm with best accuracy is finalized and that model will predict the air quality index and also based on predicted air quality index value we will display air quality is good , moderate, unhealthy, unhealthy for strong people and hazardous based on user given new data from front end.

Data Flow Diagram



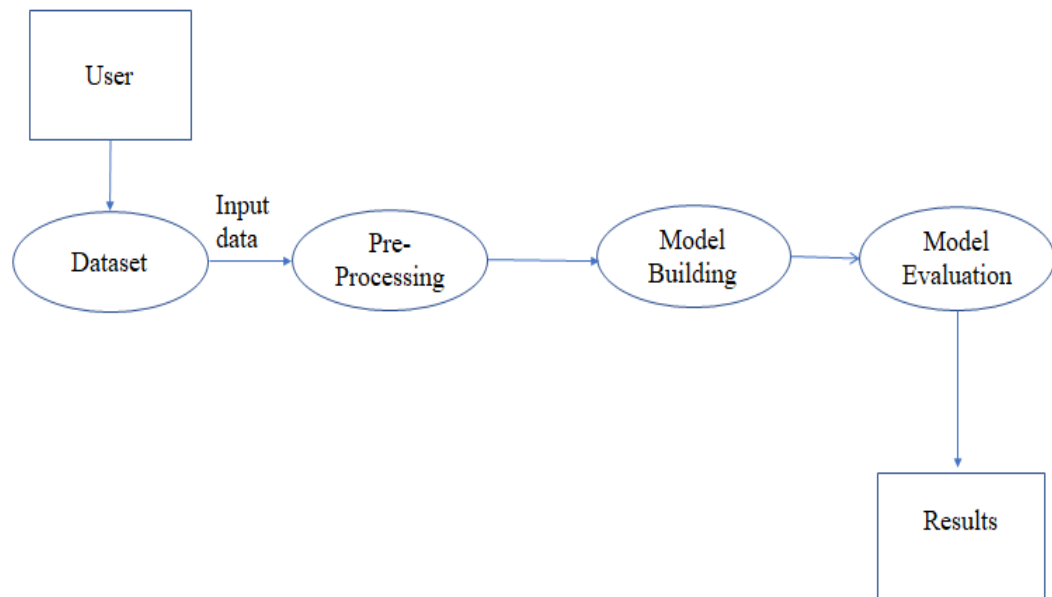
- Air Quality dataset is taken and loaded.
- The data is preprocessed in order to fill the missing values and drop duplicate values etc. in the dataset etc to increase the accuracy.
- The model is built using machine learning algorithms like Linear Regression and Random Forest Regressor.
- The model is trained with the preprocessed data.
- The model is tested and accuracy is calculated for different ML algorithms.
- The algorithm with best accuracy is finalized and that model will predict the air quality index and also based on predicted air quality index value we will display air quality is good , moderate, unhealthy, unhealthy for strong people and hazardous based on user given new data from front end.

Use Case Diagram



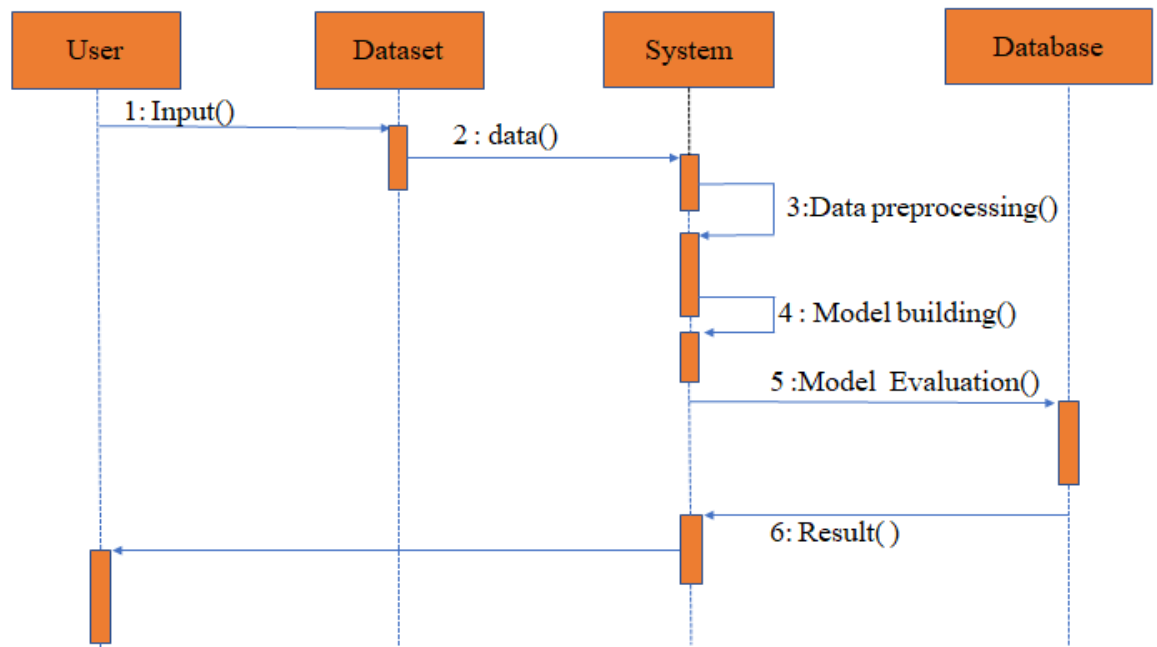
- The system will collect the data from the user.
- The system will analyze the collected data .
- The system will do preprocessing and model building.
- The built model is evaluated and tested for accuracy.
- The user will finalize the model with best accuracy.
- The user will give new data to the system and the model will predict results and display .

Activity Diagram



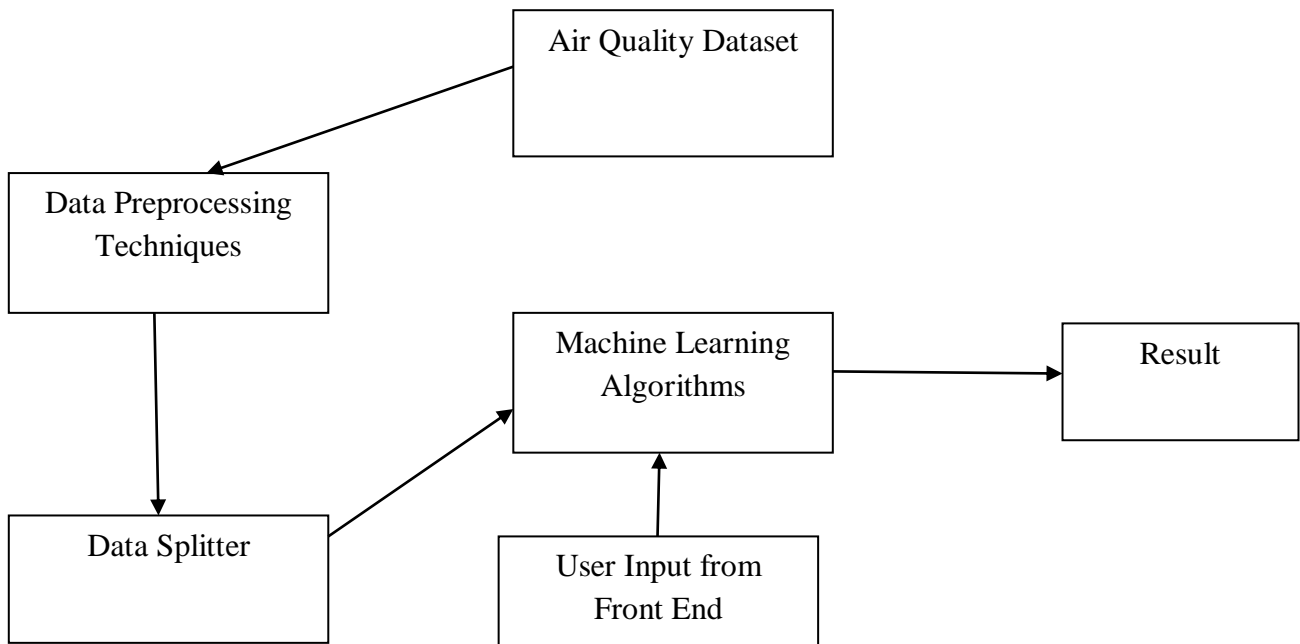
- The user will provide the dataset to the system.
- The dataset is preprocessed in order to increase the accuracy of the model.
- The model is built using different algorithms.
- The model is evaluated and model with best accuracy is finalized.
- The finalized model will predict the results.

Sequence Diagram

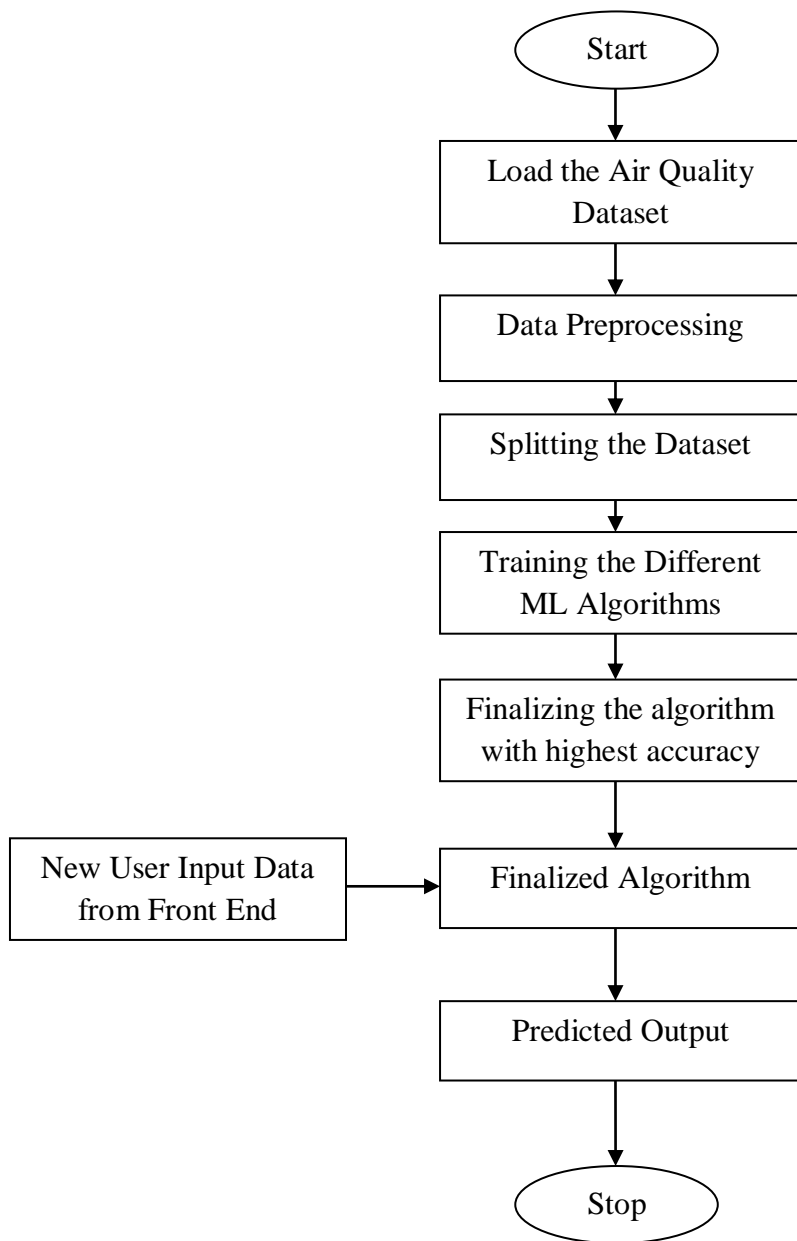


- The user will give dataset as input to the system.
- The system will store the dataset given by the user in its database.
- The system will do preprocessing of the data stored.
- The model is built using various ML algorithms and trained using preprocessed data.
- The model is evaluated and the algorithm with best accuracy is finalized.
- The finalized model will predict the results.

Class Diagram



Flow Chart:



Implementation:

Python:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting language. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available as open source. Python is widely considered as the preferred language for teaching and learning ML (Machine Learning).

Features :

- Easy to code
- Free and Open Source
- Object-Oriented Language
- High-Level Language
- Extensible feature
- Python is Portable language
- Python is Integrated language
- Interpreted Language
- Large Standard Library
- Dynamically Typed Language

Python IDLE :

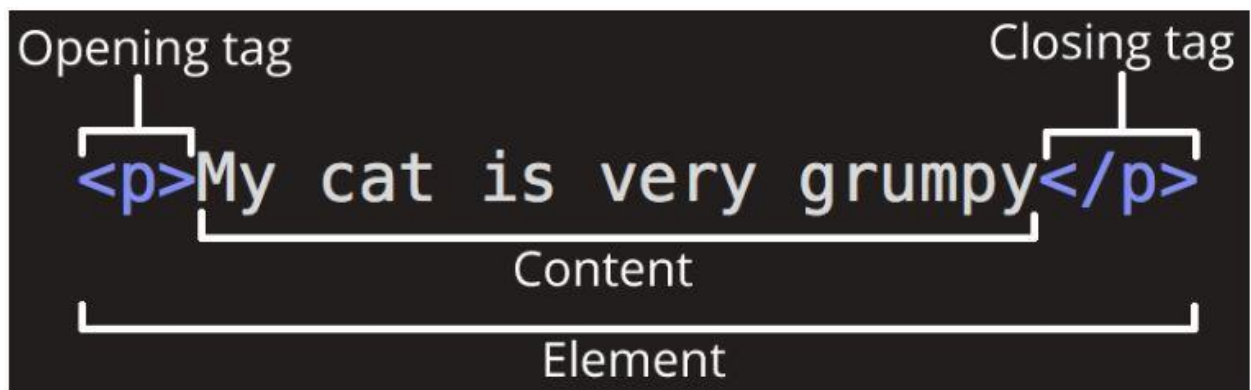
- An IDE (or Integrated Development Environment) is a program dedicated to software development.
- IDEs integrate several tools specifically designed for software development. These tools usually include:
 - An editor designed to handle code (with, for example, syntax highlighting and auto-completion)
 - Build, execution, and debugging tools
 - Some form of source control
- When you install Python, IDLE is also installed by default.
- Its major features include the Python shell window(interactive interpreter), auto-completion, syntax highlighting, smart indentation, and a basic integrated debugger.
- We used python version 3.6.8

HTML:

HTML (**H**ypertext **M**arkup **L**anguage) is the code that is used to structure a web page and its content. For example, content could be structured within a set of paragraphs, a list of bulleted points, or using images and data tables.

HTML is a *markup language* that defines the structure of your content. HTML consists of a series of [elements](#), which you use to enclose, or wrap, different parts of the content to make it appear a certain way, or act a certain way. The enclosing [tags](#) can make a word or image hyperlink to somewhere else, can italicize words, can make the font bigger or smaller, and so on.

Anatomy of HTML Element :



The main parts of our element are as follows:

1. **The opening tag:** This consists of the name of the element (in this case, `p`), wrapped in opening and closing **angle brackets**. This states where the element begins or starts to take effect — in this case where the paragraph begins.
2. **The closing tag:** This is the same as the opening tag, except that it includes a *forward slash* before the element name. This states where the element ends — in this case where the paragraph ends. Failing to add a closing tag is one of the standard beginner errors and can lead to strange results.
3. **The content:** This is the content of the element, which in this case, is just text.
4. **The element:** The opening tag, the closing tag, and the content together comprise the element.

Elements can also have attributes that look like the following:



```
<p class="editor-note">My cat is very grumpy</p>
```

Attributes contain extra information about the element that you don't want to appear in the actual content. Here, `class` is the attribute *name* and `editor-note` is the attribute *value*. The `class` attribute allows you to give the element a non-unique identifier that can be used to target it (and any other elements with the same `class` value) with style information and other things.

An attribute should always have the following:

1. A space between it and the element name (or the previous attribute, if the element already has one or more attributes).
2. The attribute name followed by an equal sign.
3. The attribute value wrapped by opening and closing quotation marks.

Some elements have no content and are called **empty elements**. Take the `` element that we already have in our HTML page:

Example:-

```

```

This contains two attributes, but there is no closing `` tag and no inner content. This is because an image element doesn't wrap content to affect it. Its purpose is to embed an image in the HTML page in the place it appears.

Anatomy of an HTML Document :

Example: `<!DOCTYPE html>`

```
<html>

<head>

  <meta charset="utf-8">

  <title>My test page</title>

</head>

<body>

</body>

</html>
```

Here, we have the following:

- `<!DOCTYPE html>` — doctype. It is a required preamble. In the mists of time, when HTML was young (around 1991/92), doctypes were meant to act as links to a set of rules that the HTML page had to follow to be considered good HTML, which could mean automatic error checking and

other useful things. However these days, they don't do much and are basically just needed to make sure your document behaves correctly. That's all you need to know for now.

- `<html></html>` — the [<html>](#) element. This element wraps all the content on the entire page and is sometimes known as the root element.
- `<head></head>` — the [<head>](#) element. This element acts as a container for all the stuff you want to include on the HTML page that *isn't* the content you are showing to your page's viewers. This includes things like [keywords](#) and a page description that you want to appear in search results, CSS to style our content, character set declarations, and more.
- `<meta charset="utf-8">` — This element sets the character set your document should use to UTF-8 which includes most characters from the vast majority of written languages. Essentially, it can now handle any textual content you might put on it. There is no reason not to set this and it can help avoid some problems later on.
- `<title></title>` — the [<title>](#) element. This sets the title of your page, which is the title that appears in the browser tab the page is loaded in. It is also used to describe the page when you bookmark/favorite it.
- `<body></body>` — the [<body>](#) element. This contains *all* the content that you want to show to web users when they visit your page, whether that's text, images, videos, games, playable audio tracks, or whatever else.

Images

Let's turn our attention to the [](#) element again:

```

```

As we said before, it embeds an image into our page in the position it appears. It does this via the `src` (source) attribute, which contains the path to our image file.

We have also included an alt (alternative) attribute. In this attribute, you specify descriptive text for users who cannot see the image, possibly because of the following reasons:

1. They are visually impaired. Users with significant visual impairments often use tools called screen readers to read out the alt text to them.
2. Something has gone wrong causing the image not to display. For example, try deliberately changing the path inside your src attribute to make it incorrect.

Headings :

Heading elements allow you to specify that certain parts of your content are headings — or subheadings. In the same way that a book has the main title, chapter titles, and subtitles, an HTML document can too. HTML contains 6 heading levels, `<h1>`–`<h6>`, although you'll commonly only use 3 to 4 at most:

```
<h1>My main title</h1>
```

```
<h2>My top level heading</h2>
```

```
<h3>My subheading</h3>
```

```
<h4>My sub-subheading</h4>
```

Links

Links are very important — they are what makes the web a web! To add a link, we need to use a simple element — `<a>` — "a" being the short form for "anchor". To make text within your paragraph into a link, follow these steps:

1. Choose some text. We chose the text "Mozilla Manifesto".
2. Wrap the text in an `<a>` element, as shown below:

```
<a>Mozilla Manifesto</a>
```

3. Give the `<a>` element an href attribute, as shown below:

```
<a href="">Mozilla Manifesto</a>
```

4. Fill in the value of this attribute with the web address that you want the link to link to:

```
<a href="https://www.mozilla.org/en-US/about/manifesto/">Mozilla  
Manifesto</a>
```

You might get unexpected results if you omit the `https://` or `http://` part, called the *protocol*, at the beginning of the web address. After making a link, click it to make sure it is sending you where you wanted it to.

Xampp Control Panel :

XAMPP is the title used for a **compilation of free software**. The name is an acronym, with each letter representing one of the five key components. The software packet contains the web server **A**ppache, the relational database management system **M**ySQL (or **M**ariaDB), and the scripting languages **P**erl and **P**HP. The initial **X** stands for the operating systems that it works with: Linux, Windows, and Mac OS X.

- **Apache:** the open source web server Apache is the most widely used server worldwide for delivery of web content. The server application is made available as a free software by the Apache Software Foundation.
- **MySQL/MariaDB:** in MySQL, XAMPP contains one of the most popular relational database management systems in the world. In combination with the web server Apache and the scripting language PHP, MySQL offers data storage for web services. Current XAMPP versions have replaced MySQL with MariaDB (a community-developed fork of the MySQL project, made by the original developers).
- **PHP:** the server-side programming language [PHP](#) enables users to create dynamic websites or applications. PHP can be installed on all platforms and supports a number of diverse database systems.
- **Perl:** the scripting language Perl is used in system administration, web development, and network programming. Like PHP, Perl also enables users to program dynamic web applications.

Alongside these core components, this free-to-use Apache distribution contains some other useful tools, which vary depending on your operating system. These

tools include the mail server **Mercury**, the database administration tool **phpMyAdmin**, the web analytics software solutions **Webalizer**, **OpenSSL**, and **Apache Tomcat**, and the FTP servers **FileZilla** or **ProFTPD**.

Application areas

An XAMPP server can be installed and used with a single executable file quickly and easily, functioning as a local test system for Linux, Windows, and Mac OS X. The software packet contains the same components that are found on common web servers. Developers have the chance to test out their projects locally and to transfer them easily to productive systems. But XAMPP isn't suitable to use as a public server, because **many safety features have been deliberately left out** to simplify and speed up the system for testing.

Flask:

Flask (source code) is a Python web framework built with a small core and easy-to-extend philosophy. Flask is considered more Pythonic than the Django web framework because in common situations the equivalent Flask web application is more explicit. Flask is also easy to get started with as a beginner because there is little boilerplate code for getting a simple app up and running. Flask is a popular Python web framework, meaning it is a third-party Python library used for developing web applications.

What is Web Framework?

Web Application Framework or simply Web Framework represents a collection of libraries and modules that enables a web application developer to

write applications without having to bother about low-level details such as protocols, thread management etc.

What is Flask?

Flask is a web application framework written in Python. It is developed by **Armin Ronacher**, who leads an international group of Python enthusiasts named Pocco. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

WSGI

Web Server Gateway Interface (WSGI) has been adopted as a standard for Python web application development. WSGI is a specification for a universal interface between the web server and the web applications.

Werkzeug

It is a WSGI toolkit, which implements requests, response objects, and other utility functions. This enables building a web framework on top of it. The Flask framework uses Werkzeug as one of its bases.

Jinja2

Jinja2 is a popular templating engine for Python. A web templating system combines a template with a certain data source to render dynamic web pages.

Flask is often referred to as a micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have built-in abstraction layer

for database handling, nor does it have form a validation support. Instead, Flask supports the extensions to add such functionality to the application. Some of the popular Flask extensions are discussed later in the tutorial.

Flask – Application

Example :

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World'

if __name__ == '__main__':
    app.run()
```

Importing flask module in the project is mandatory. An object of Flask class is our **WSGI** application.

Flask constructor takes the name of **current module** (**__name__**) as argument.

The **route()** function of the Flask class is a decorator, which tells the application which URL should call the associated function.

app.route(rule, options)

- The **rule** parameter represents URL binding with the function.
- The **options** is a list of parameters to be forwarded to the underlying Rule object.

In the above example, ‘/’ URL is bound with **hello_world()** function. Hence, when the home page of web server is opened in browser, the output of this function will be rendered.

Finally the **run()** method of Flask class runs the application on the local development server.

app.run(host, port, debug, options)

All parameters are optional

Sr.No.	Parameters & Description
1	host Hostname to listen on. Defaults to 127.0.0.1 (localhost). Set to ‘0.0.0.0’ to have server available externally
2	port Defaults to 5000
3	debug Defaults to false. If set to true, provides a debug information
4	options

To be forwarded to underlying Werkzeug server.
--

The above given **Python** script is executed from Python shell.

Python Hello.py

A message in Python shell informs you that

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

Open the above URL (**localhost:5000**) in the browser. '**Hello World**' message will be displayed on it.

Debug mode

A **Flask** application is started by calling the **run()** method. However, while the application is under development, it should be restarted manually for each change in the code. To avoid this inconvenience, enable **debug support**. The server will then reload itself if the code changes. It will also provide a useful debugger to track the errors if any, in the application.

The **Debug** mode is enabled by setting the **debug** property of the **application** object to **True** before running or passing the debug parameter to the **run()** method.

<pre>app.debug = True app.run() app.run(debug = True)</pre>

Flask – Routing

Modern web frameworks use the routing technique to help a user remember application URLs. It is useful to access the desired page directly without having to navigate from the home page.

The **route()** decorator in Flask is used to bind URL to a function. For example –

```
@app.route('/hello')  
def hello_world():  
    return 'hello world'
```

Here, URL **‘/hello’** rule is bound to the **hello_world()** function. As a result, if a user visits **http://localhost:5000/hello** URL, the output of the **hello_world()** function will be rendered in the browser.

The **add_url_rule()** function of an application object is also available to bind a URL with a function as in the above example, **route()** is used.

Flask – HTTP methods

Http protocol is the foundation of data communication in world wide web. Different methods of data retrieval from specified URL are defined in this protocol.

The following table summarizes different http methods –

Sr.No.	Methods & Description
1	GET

	Sends data in unencrypted form to the server. Most common method.
2	HEAD Same as GET, but without response body
3	POST Used to send HTML form data to server. Data received by POST method is not cached by server.
4	PUT Replaces all current representations of the target resource with the uploaded content.
5	DELETE Removes all current representations of the target resource given by a URL

By default, the Flask route responds to the **GET** requests. However, this preference can be altered by providing methods argument to **route()** decorator.

In order to demonstrate the use of **POST** method in URL routing, first let us create an HTML form and use the **POST** method to send form data to a URL.

Flask – Templates

It is possible to return the output of a function bound to a certain URL in the form of HTML. For instance, in the following script, **hello()** function will render '**Hello World**' with **<h1>** tag attached to it.

```
from flask import Flask
```

```
app = Flask(__name__)

@app.route('/')
def index():
    return '<html><body><h1>Hello World</h1></body></html>'

if __name__ == '__main__':
    app.run(debug = True)
```

However, generating HTML content from Python code is cumbersome, especially when variable data and Python language elements like conditionals or loops need to be put. This would require frequent escaping from HTML.

This is where one can take advantage of **Jinja2** template engine, on which Flask is based. Instead of returning hardcoded HTML from the function, a HTML file can be rendered by the **render_template()** function.

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('hello.html')

if __name__ == '__main__':
    app.run(debug = True)
```

Flask will try to find the HTML file in the templates folder, in the same folder in which this script is present.

- Application folder
 - Hello.py

- templates
 - hello.html

The term ‘**web templating system**’ refers to designing an HTML script in which the variable data can be inserted dynamically. A web template system comprises of a template engine, some kind of data source and a template processor.

Flask uses **Jinja2** template engine. A web template contains HTML syntax interspersed placeholders for variables and expressions (in these case Python expressions) which are replaced values when the template is rendered.

The following code is saved as **hello.html** in the templates folder.

```
<!doctype html>
<html>
  <body>

    <h1>Hello {{ name }}!</h1>

  </body>
</html>
```

Next, run the following script from Python shell.

```
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/hello/<user>')
def hello_name(user):
    return render_template('hello.html', name = user)

if __name__ == '__main__':
```



```
app.run(debug = True)
```

As the development server starts running, open the browser and enter URL as
– **http://localhost:5000/hello/mvl**

The **variable** part of URL is inserted at **{{ name }}** place holder.

Flask – Sending Form Data to Template

We have already seen that the http method can be specified in URL rule. The **Form** data received by the triggered function can collect it in the form of a dictionary object and forward it to a template to render it on a corresponding web page.

In the following example, **‘/’** URL renders a web page (student.html) which has a form. The data filled in it is posted to the **‘/result’** URL which triggers the **result()** function.

The **results()** function collects form data present in **request.form** in a dictionary object and sends it for rendering to **result.html**.

The template dynamically renders an HTML table of **form** data.

Flask Framework: Mysql Connection Using PyMysql

PyMysql is python library,used to connect with mysql database.

```
import pymysql
```

We import the `pymysql` module.

```
con = pymysql.connect('localhost', 'user7',  
    '$cret', 'testdb')
```

We connect to the database with `connect`. We pass four parameters: the hostname, the MySQL user name, the password, and the database name.

```
with con.cursor() as cur:
```

Using the `with` keyword, the Python interpreter automatically releases the resources. It also provides error handling. We get a cursor object, which is used to traverse records from the result set.

```
cur.execute()
```

We call the `execute` function of the cursor and execute the SQL statement.

```
version = cur.fetchone()
```

The `fetchone` function fetches the next row of a query result set, returning a single sequence, or `None` when no more data is available.

The `fetchAll` method retrieves all (remaining) rows of a query result, returning them as a sequence of sequences.

A new row is inserted with the `INSERT INTO SQL` statement.

In `pymysql`, the `autocommit` is off by default. We need to call `commit` to execute the changes.

Machine Learning:

1. What is Machine Learning ?

- Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.
- Machine Learning is the science of getting computers to learn and act like humans do,
- Machine learning is like farming or gardening. Seeds is the algorithms, nutrients is the data, the gardner is you and plants is the programs.

2. Applications of Machine learning

- Web search: ranking page based on what you are most likely to click on.
- Computational biology: rational design drugs in the computer based on past experiments.
- Finance: decide who to send what credit card offers to. Evaluation of risk on credit offers. How to decide where to invest money.
- Robotics: how to handle uncertainty in new environments. Autonomous. Self-driving car.
- E-commerce: Predicting customer churn. Whether or not a transaction is fraudulent.
- Space exploration: space probes and radio astronomy.
- Information extraction: Ask questions over databases across the web.
- Social networks: Data on relationships and preferences. Machine learning to extract value from data.

- Debugging: Use in computer science problems like debugging. Labor intensive process. Could suggest where the bug could be.
- Machine learning algorithms also help to improve user experience and customization for online platforms.
- Facebook, Netflix, Google, and Amazon all use recommendation systems to prevent content glut and provide unique content to individual users based on their likes and dislikes.
- Facebook utilizes recommendation engines for its news feed on both Facebook and Instagram, as well as for its advertising services to find relevant leads.
- Netflix collects user data and recommends various movies and series based on the preferences of the user. Google utilizes machine learning to structure its results and for YouTube's recommendation system, among many other applications.
- Amazon uses ML to place relevant products in the user's field of view, maximizing conversion rates by recommending products that the user actually wants to buy.

3. Key elements of machine learning

1. **Representation** : how to represent knowledge.

Examples include decision trees, sets of rules, instances, graphical models, neural networks, support vector machines, model ensembles and others.

2. **Evaluation**: the way to evaluate candidate programs (hypotheses).

Examples include accuracy, prediction and recall, squared error, likelihood, posterior probability, cost, margin, entropy k-L divergence and others.

4. Types of Machine learning

There are four types of machine learning:

1.Supervised learning: (also called inductive learning) Training data includes desired outputs. This is spam this is not, learning is supervised.

2.Unsupervised learning: Training data does not include desired outputs. Example is clustering. It is hard to tell what is good learning and what is not.

3. Semi-supervised learning: Training data includes a few desired outputs.

4.Reinforcement learning: Rewards from a sequence of actions. It is the most ambitious type of learning.

Machine Learning Packages:

Numpy:

1. What is NumPy ?

- NumPy is the fundamental package for scientific computing in Python.

- NumPy is a Python package that stands for ‘Numerical Python’ which contains a powerful n-dimensional array object.
- It is used in the industry for array computing.
- NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

2. Features of Numpy:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

3. Applications of Numpy

- Numpy provides a high-performance multidimensional array and basic tools to compute with and manipulate these arrays.
- It is an alternative for lists and arrays in Python.
- It maintains minimal memory.
- We can perform different types of mathematical operations.
- Shape Manipulations.
- It is used with other libraries like Pandas, SciPy, Matplotlib and Tkinter etc.

Pandas :

1. What is Pandas ?

- Pandas is a Python package that provides fast, flexible, and expressive data structures.
- Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
- It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

2. Features of Pandas

- Dataframe object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and subsetting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging.

- Provides data filtration.

3. Pandas deals with the following three data structures:-

1. Series

2. DataFrame

3. Panel

Data Structure	Dimensions	Description
Series	1	1D labeled homogeneous array, size immutable.
Data Frames	2	General 2D labeled, size-mutable tabular structure with potentially heterogeneously typed columns.
Panel	3	General 3D labeled, size-mutable array.

1. Series

- Series is a one-dimensional array like structure with homogeneous data.
- For example, the following series is a collection of integers 10, 23, 56, ...

10	23	56	17	52	61	73	90	26	72
----	----	----	----	----	----	----	----	----	----

2. DataFrame

- DataFrame is a two-dimensional array with heterogeneous data.
- For example, the following table show the data is represented in rows and columns. Each column represents an attribute and each row represents a person.

Name	Age	Gender	Rating
Steve	32	Male	3.45
Lia	28	Female	4.6
Vin	45	Male	3.9
Katie	38	Female	2.78

3. Panel

- Panel is a three-dimensional data structure with heterogeneous data.
- It is hard to represent the panel in graphical representation. But a panel can be illustrated as a container of DataFrame.

4. Applications

- Economics
- Recommendation Systems
- Stock Prediction.
- Neuroscience.

- Statistics.
- Advertising.
- Analytics.
- Natural Language Processing.
- Big Data
- Data Sceince

SkLearn:

1. What is SciKit-Learn?

- Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python.
- It provides a selection of efficient tools for machine learning and statistical modelling.
- It includes classification, regression, clustering and dimensionality reduction via a consistence interface in Python.
- This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

2. Features of Sklearn

- Datasets
- Feature extraction
- Feature selection
- Parameter Tuning
- Clustering
- Cross-Validation

- Supervised Models
- Unsupervised Models
- Dimensionality Reduction
- Ensemble methods

3. **Benifits of SciKit-Learn**

- **BSD license:** Scikit-learn has a BSD license; hence, there is minimal restriction on the use and distribution of the software, making it free to use for everyone.
- **Easy to use:** The popularity of Scikit-learn is because of the ease of use it offers.
- **Document detailing:** It also offers document detailing of the API that users can access at any time on the website, helping them integrate Machine Learning into their own platforms.
- **Extensive use in the industry:** Scikit-learn is used extensively by various organizations to predict consumer behavior, identify suspicious activities, and much more.
- **Machine Learning algorithms:** Scikit-learn covers most of the Machine Learning algorithms Huge community support.
- **Algorithms flowchart:** Unlike other programming languages where users usually face a problem of having to choose from multiple competing implementations of same algorithms, Scikit-learn has an algorithm cheat sheet or flowchart to assist the users.

4. Applications

- Scikit-learn is a library that contains several implementations of machine learning algorithms.
- financial cyber security analytics, product development, neuro imaging, barcode scanner development, medical modeling.
- Regression modeling
- Unsupervised classification and clustering
- Decision tree pruning and induction
- Comprehensive and neural network training with regression and classification algorithms
- Decision boundary learning with SVMs
- Advanced probability modeling
- Feature analysis and selection
- Reduction of dimensionality
- Outlier detection and rejection

Machine Learning Algorithms:

1. What is an Algorithm?

- An algorithm is a finite sequence of well-defined, computer-implementable instructions.
- Machine learning algorithms are the engines of machine learning, meaning it is the algorithms that turn a data set into a model.
- Algorithms are always unambiguous and are used as specifications for performing calculations, data processing, automated reasoning, and other tasks.

2. What is Regression ?

- Regression analysis is a statistical method that helps us to analyse and understand the relationship between two or more variables of interest.
- Regression models are used to predict a continuous value.
- It is used in Financial Industry, marketing, manufacturing and medicine etc...

3. What is Linear Regression ?

- Linear Regression is a supervised machine learning algorithm which performs a regression task.
- The predicted output is continuous and has a constant slope in this algorithm.
- It is a very widely used statistical tool to establish a relationship model between two variables.
- It is a linear model that establishes the relationship between a dependent variable and one or more independent variables.
- It works on the principle of Mean square error (MSE).
- To decide whether our line is best fitted or not we will define a cost function.

Linear regression is probably one of the most important and widely used regression techniques. It's among the simplest regression methods. One of its main advantages is the ease of interpreting results.

When implementing linear regression of some dependent variable y on the set of independent variables $\mathbf{x} = (x_1, \dots, x_r)$, where r is the number of predictors, you assume a linear relationship between y and \mathbf{x} : $y = \beta_0 + \beta_1 x_1 + \dots + \beta_r x_r + \varepsilon$. This equation is the **regression equation**. $\beta_0, \beta_1, \dots, \beta_r$ are the **regression coefficients**, and ε is the **random error**. Linear regression calculates the **estimators** of the regression coefficients or simply the **predicted weights**, denoted with b_0, b_1, \dots, b_r . They define the **estimated regression function** $(\mathbf{x}) = b_0 + b_1 x_1 + \dots + b_r x_r$. This function should capture the dependencies between the inputs and output sufficiently well. The **estimated or predicted response**, (\mathbf{x}_i) , for each observation $i = 1, \dots, n$, should be as close as possible to the corresponding **actual response** y_i . The differences $y_i - (\mathbf{x}_i)$ for all observations $i = 1, \dots, n$, are called the **residuals**. Regression is about determining the **best predicted weights**, that is the weights corresponding to the smallest residuals. To get the best weights, you usually **minimize the sum of squared residuals** (SSR) for all observations $i = 1, \dots, n$: $SSR = \sum_i (y_i - f(\mathbf{x}_i))^2$. This approach is called the **method of ordinary least squares**.

The variation of actual responses $y_i, i = 1, \dots, n$, occurs partly due to the dependence on the predictors \mathbf{x}_i . However, there is also an additional inherent variance of the output. The **coefficient of determination**, denoted as R^2 , tells you which amount of variation in y can be explained by the dependence on \mathbf{x} using the particular regression model. Larger R^2 indicates a better fit and means that the model can better explain the variation of the output with different inputs. The value $R^2 = 1$ corresponds to $SSR = 0$, that is to the **perfect fit** since the values of predicted and actual responses fit completely to each other.

There are five basic steps when you're implementing linear regression:

1. Import the packages and classes you need.
2. Provide data to work with and eventually do appropriate transformations.
3. Create a regression model and fit it with existing data.
4. Check the results of model fitting to know whether the model is satisfactory.
5. Apply the model for predictions.

Applications of Linear Regression

- It used to generate insights on consumer behavior, understanding business and factors influencing profitability.
- It is used in business to evaluate trends and make estimates or forecasts.
- It is used to analyze the marketing effectiveness, pricing and promotions on sales of a product.

Random Forest Regressor:

Random forest is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

It technically is an ensemble method (based on the divide-and-conquer approach) of decision trees generated on a randomly split dataset. This collection of decision tree classifiers is also known as the forest. The individual decision trees are generated using an attribute selection indicator such as information gain, gain ratio, and Gini index for each attribute. Each tree depends on an independent random sample. These predictions are then averaged to produce a single result. The averaging makes a Random Forest better than a single Decision Tree hence improves its accuracy and reduces overfitting. A prediction from the Random Forest Regressor is an average of the predictions produced by the trees in the forest.

It works in four steps:

1. Select random samples from a given dataset.
2. Construct a decision tree for each sample and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.
4. Select the prediction result with the most votes as the final prediction.

Advantages:

- Random forests is considered as a highly accurate and robust method because of the number of decision trees participating in the process.
- It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.
- The algorithm can be used in both classification and regression problems.

- Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing values

Programming:

Dataset:

This is the dataset used in this project.

Microsoft Excel

HomeInsertPage LayoutFormulasDataReviewView

T2

<

Back End Code:

This code is to train our system with machine learning techniques and algorithms and calculating the R2 score and finalizing the algorithm with good accuracy as final model and converting it into pickle (binary format) file.

```
Air Quality Prediction.py - E:\MIFRATECH\Machine Learning\Machine Learning Finalized Projects\Air Quality Prediction\Air Quality Prediction.py (3.6.8)
File Edit Format Run Options Window Help

# for numerical computing
import numpy as np

# for dataframes
import pandas as pd

# Ignore Warnings
import warnings
warnings.filterwarnings("ignore")

# to split train and test set
from sklearn.model_selection import train_test_split

import sklearn.metrics as sm

df=pd.read_csv('Dataset1.csv')
print(df.shape)
print(df.columns)
print(df.head())
print(df.describe())
print(df.corr())
df = df.drop_duplicates()
print(df.shape)
print(df.isnull().sum())
df=df.dropna()
print(df.isnull().sum())

y = df.AQI
X = df.drop('AQI', axis=1)
```

```
Air Quality Prediction.py - E:\MIFRATECH\Machine Learning\Machine Learning Finalized Projects\Air Quality Prediction\Air Quality Prediction.py (3.6.8)
File Edit Format Run Options Window Help

# Split X and y into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Print number of observations in X_train, X_test, y_train, and y_test
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

from sklearn.ensemble import RandomForestRegressor
model1 = RandomForestRegressor()

from sklearn.linear_model import LinearRegression
model2 = LinearRegression()

model1.fit(X_train, y_train)
model2.fit(X_train, y_train)

## Predict Test set results
y_pred1 = model1.predict(X_test)
y_pred2 = model2.predict(X_test)

df1 = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred1})
print(df1)

y_pred1=np.array(y_pred1).reshape(-1,1)
```

Ln: 1 Col: 0

```
Air Quality Prediction.py - E:\MIFRATECH\Machine Learning\Machine Learning Finalized Projects\Air Quality Prediction\Air Quality Prediction.py (3.6.8)
File Edit Format Run Options Window Help

y_pred1=np.array(y_pred1).reshape(-1,1)

y_pred2=np.array(y_pred2).reshape(-1,1)

accuracy=sm.r2_score(y_test,y_pred1)
print("Accuracy of RandomForestRegressor is {:.2f} %".format(accuracy*100))

accuracy=sm.r2_score(y_test,y_pred2)
print("Accuracy of Linear Regression is {:.2f} %".format(accuracy*100))

#from import joblib
import joblib
# Save the model as a pickle in a file
joblib.dump(model1, 'final_pickle_model.pkl')

# Load the model from the file
final_model= joblib.load('final_pickle_model.pkl')

pred=final_model.predict(X_test)
accuracy=sm.r2_score(y_test,pred)
print("Accuracy of Final Model is {:.2f} %".format(accuracy*100))
```

Ln: 1 Col: 0

Output of back end code:

```
Python 3.6.8 Shell
File Edit Shell Debug Options Window Help
(7288, 11)
Index(['Temperature', 'Humidity', 'WindSpeed', 'Visibility', 'Pressure', 'so2',
      'no2', 'Rainfall', 'PM10', 'PM25', 'AQI'],
      dtype='object')
Temperature Humidity WindSpeed Visibility ... Rainfall PM10 PM25 AQI
0 14.033333 93 14.1197 15.8263 ... 50.7 87 89.1 168
1 15.095556 93 14.2646 15.8263 ... 52.1 122 109.5 177
2 15.916662 89 3.9284 14.9569 ... 53.8 95 100.2 174
3 16.094444 93 14.1036 15.8263 ... 53.7 79 89.6 169
4 16.094444 94 11.0446 15.8263 ... 54.5 63 76.3 162

[5 rows x 11 columns]
Temperature Humidity ... PM25 AQI
count 7288.000000 7288.000000 ... 7288.000000 7288.000000
mean 12.034334 75.892426 ... 30.336389 84.537870
std 9.345925 18.406588 ... 24.861132 42.370068
min -10.133333 23.000000 ... 1.000000 1.000000
25% 4.050000 64.000000 ... 14.700000 57.000000
50% 11.650000 81.000000 ... 22.600000 73.000000
75% 19.136111 91.000000 ... 36.700000 104.000000
max 34.811111 100.000000 ... 311.900000 362.000000

[8 rows x 11 columns]
Temperature Humidity WindSpeed ... PM10 PM25 AQI
Temperature 1.000000 -0.640853 -0.007853 ... 0.018393 0.122030 0.167722
Humidity -0.640853 1.000000 -0.041377 ... -0.047126 -0.119834 -0.143279
WindSpeed -0.007853 -0.041377 1.000000 ... -0.003909 -0.015770 -0.031410
Visibility 0.108522 0.057076 0.107416 ... 0.036917 0.117586 0.130885
Pressure -0.008497 0.004141 -0.058970 ... 0.008507 0.008915 -0.000402
so2 0.093858 -0.091301 -0.079710 ... 0.090637 0.226010 0.247686
no2 0.072414 -0.021266 -0.072865 ... 0.004923 0.007830 0.024833
Rainfall -0.026200 -0.038498 0.041548 ... 0.007034 -0.015103 -0.026996
PM10 0.018393 -0.047126 -0.003909 ... 1.000000 0.592303 0.503657
PM25 0.122030 -0.119834 -0.015770 ... 0.592303 1.000000 0.548092
AQI 0.167722 -0.143279 -0.031410 ... 0.503657 0.548092 1.000000

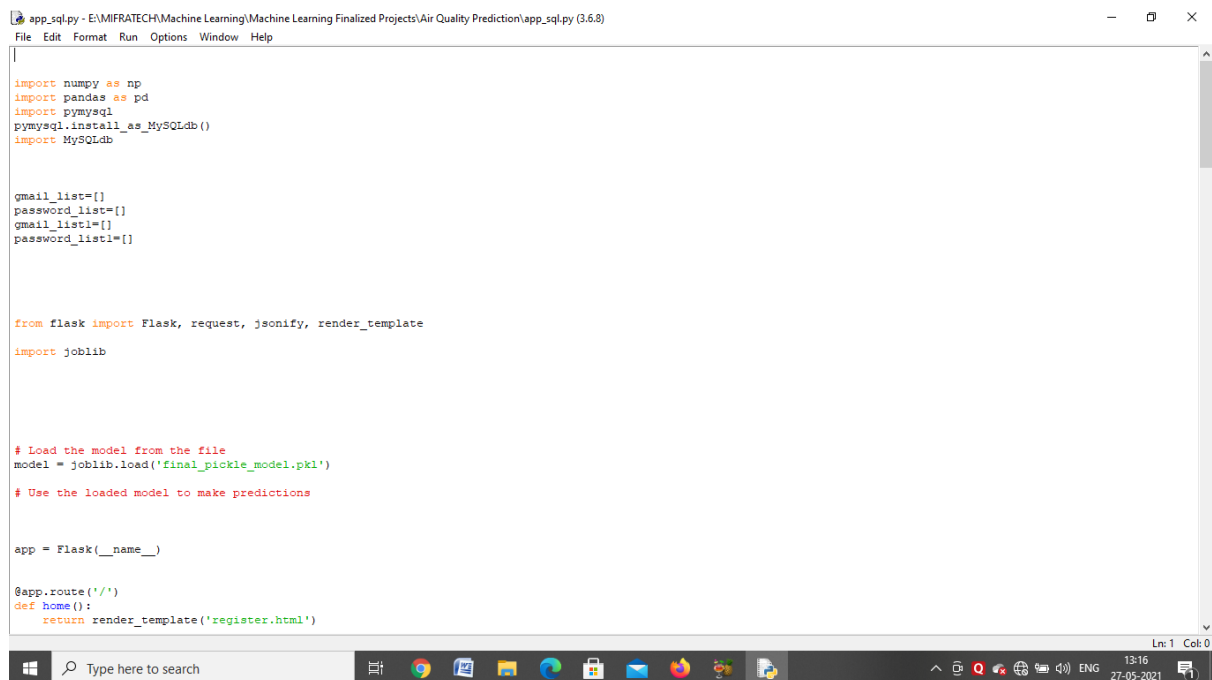
[11 rows x 11 columns]
(7288, 11)
Temperature 0
Humidity 0
WindSpeed 0
Ln: 5 Col: 0
```

```
Python 3.6.8 Shell
File Edit Shell Debug Options Window Help
(5830, 10)
WindSpeed 0
Visibility 0
Pressure 0
so2 0
no2 0
Rainfall 0
PM10 0
PM25 0
AQI 0
dtype: int64
Temperature 0
Humidity 0
WindSpeed 0
Visibility 0
Pressure 0
so2 0
no2 0
Rainfall 0
PM10 0
PM25 0
AQI 0
dtype: int64
(5830, 10) (1458, 10) (5830,) (1458,)
Actual Predicted
5733 55 55.00
3433 21 20.95
982 148 148.41
6104 72 72.00
2521 24 24.10
... ...
401 102 102.16
5034 83 83.00
3069 37 37.03
1979 74 74.00
1262 57 57.00

[1458 rows x 2 columns]
Accuracy of RandomForestRegressor is 99.84 %
Accuracy of Linear Regression is 90.11 %
Accuracy of Final Model is 99.84 %
>>>
```


Front End Code:

This code is to get create front end with the help of flask, pymysql and HTML. Here we will collect user data like username and password for registration and will be stored in localhost database only and next login credentials will collect and compare with the database and once credentials were correct the user will go to the prediction page. In the prediction page user will enter the different input data. The collected data from front end is given to our finalized machine learning trained algorithm to predict the output and the predicted air quality index and air quality is good or bad is displayed on the prediction front end webpage.



```
app_sql.py - E:\MIFRATECH\Machine Learning\Machine Learning Finalized Projects\Air Quality Prediction\app_sql.py (3.6.8)
File Edit Format Run Options Window Help

import numpy as np
import pandas as pd
import pymysql
pymysql.install_as_MySQLdb()
import MySQLdb

gmail_list=[]
password_list=[]
gmail_list1=[]
password_list1=[]

from flask import Flask, request, jsonify, render_template
import joblib

# Load the model from the file
model = joblib.load('final_pickle_model.pkl')
# Use the loaded model to make predictions

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('register.html')
```

```
app_sql.py - E:\MIFRATECH\Machine Learning\Machine Learning Finalized Projects\Air Quality Prediction\app_sql.py (3.6.8)
File Edit Format Run Options Window Help

@app.route('/')
def home():
    return render_template('register.html')

@app.route('/register',methods=['POST'])
def register():

    int_features2 = [str(x) for x in request.form.values()]

    r1=int_features2[0]
    print(r1)

    r2=int_features2[1]
    print(r2)
    logu=int_features2[0]
    passw1=int_features2[1]

    # if int_features2[0]==12345 and int_features2[1]==12345:

    import MySQLdb

    # Open database connection
    db = MySQLdb.connect("localhost","root","","dadb" )

    # prepare a cursor object using cursor() method
    cursor = db.cursor()
    cursor.execute("SELECT user FROM user_register")
    result1=cursor.fetchall()

    for row1 in result1:
        print(row1)
        print(row1[0])
        gmail_list1.append(str(row1[0]))

Ln: 1 Col: 0
```

```
app_sql.py - E:\MIFRATECH\Machine Learning\Machine Learning Finalized Projects\Air Quality Prediction\app_sql.py (3.6.8)
File Edit Format Run Options Window Help

print(gmail_list1)
if logu in gmail_list1:
    return render_template('register.html',text="This Username is Already in Use ")
else:

# Prepare SQL query to INSERT a record into the database.
sql = "INSERT INTO user_register(user,password) VALUES (%s,%s)"
val = (r1, r2)

try:
    # Execute the SQL oommand
    cursor.execute(sql,val)
    # Commit your changes in the database
    db.commit()
except:
    # Rollback in case there is any error
    db.rollback()

# disconnect from server
db.close()
return render_template('register.html',text="Successfully Registered")

@app.route('/login')
def login():
    return render_template('login.html')

@app.route('/loggedin',methods=['POST'])
def loggedin():

    int_features3 = [str(x) for x in request.form.values()]
    print(int_features3)
    logu=int_features3[0]
    passw=int_features3[1]

Ln: 1 Col: 0
```

```
app_sql.py - E:\MIFRATECH\Machine Learning\Machine Learning Finalized Projects\Air Quality Prediction\app_sql.py (3.6.8)
File Edit Format Run Options Window Help

import MySQLdb

# Open database connection
db = MySQLdb.connect("localhost","root","","dadb" )

# prepare a cursor object using cursor() method
cursor = db.cursor()
cursor.execute("SELECT user FROM user_register")
result1=cursor.fetchall()

for row1 in result1:
    print(row1)
    print(row1[0])
    gmail_list.append(str(row1[0]))

print(gmail_list)

cursor1= db.cursor()
cursor1.execute("SELECT password FROM user_register")
result2=cursor1.fetchall()

for row2 in result2:
    print(row2)
    print(row2[0])
    password_list.append(str(row2[0]))

print(password_list)
print(gmail_list.index(logu))
print(password_list.index(passw))

if gmail_list.index(logu)==password_list.index(passw):
    return render_template('index.html')
else:
    return render_template('login.html',text='Use Proper Username and Password')
```

Ln: 1 Col: 0

```
app_sql.py - E:\MIFRATECH\Machine Learning\Machine Learning Finalized Projects\Air Quality Prediction\app_sql.py (3.6.8)
File Edit Format Run Options Window Help

@app.route('/production')
def production():
    return render_template('index.html')

@app.route('/production/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    int_features = [str(x) for x in request.form.values()]
    print(int_features)
    a=int_features

    temp = float(a[0])
    humidity=float(a[1])
    WindSpeed = float(a[2])
    Visibility =float(a[3])
    Pressure = float(a[4])
    so2=float(a[6])
    no2=float(a[7])
    Rainfall=float(a[5])
    pm10=float(a[8])
    pm2_5=float(a[9])

    data= {'Temperature':[temp], 'Humidity':[humidity], 'Wind Speed':[WindSpeed], 'Visibility':[Visibility], 'Pressure':[Pressure], 'so2':[so2], 'no2':[no2], 'Rainfall':[Rainfall]}
    df = pd.DataFrame(data)

    # Print the output.
    print(df)

    prediction=model.predict(df)
```

Ln: 1 Col: 0

app_sql.py (dirty)

```
# Print the output.
print(df )

prediction=model.predict(df)
prediction=int(prediction)

if((prediction>=0) and (prediction<=50)):
    return render_template('index.html',prediction_text='Air Quality Index is {}'.format(prediction),prediction_text1='Air Quality is Good')

if((prediction>=51) and (prediction<=100)):
    return render_template('index.html',prediction_text='Air Quality Index is {}'.format(prediction),prediction_text1='Air Quality is Moderate')

if((prediction>=101) and (prediction<=150)):
    return render_template('index.html',prediction_text='Air Quality Index is {}'.format(prediction),prediction_text1='Air Quality is Unhealthy')

if((prediction>=151) and (prediction<=200)):
    return render_template('index.html',prediction_text='Air Quality Index is {}'.format(prediction),prediction_text1='Air Quality is Unhealthy for Strong People')

if(prediction>201):
    return render_template('index.html',prediction_text='Air Quality Index is {}'.format(prediction),prediction_text1='Air Quality is Hazardous')

if __name__ == "__main__":
    app.run(debug=False)
```

Ln: 1 Col: 0

HTML Code:

Register Page:

```
register - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html >

<head>

<style>
body {
    background-image: url({{ url_for('static', filename='templates/air quality image.JPG') }})

    background-repeat: no-repeat;
    background-size: 100% 100%;
    background-position: center;
    background-size: cover;
}

a:link, a:visited {
    background-color: #f44336;
    padding: 15px 25px;
    color: white;
    text-align: center;
    display: inline-block;
}

a:hover, a:active {
    background-color: red;
}

</style>
</head>
<
```

```
register - Notepad
File Edit Format View Help

</style>
</head>

<body background="{{ url_for('static', filename='air quality image.JPG') }}">
<div class="login" align="center">
    <b><font style="color:white" "font-family:verdana" size="5"><h1>REGISTER</h1></font></b>

    <br>
    <form action="{{ url_for('register') }}" method="post">
        <input type="text" name="user_name" placeholder="user_name" required="required" /><br><br><br>
        <input type="password" value="FakePSW" id="myInput" name="password" placeholder="password" required="required" /><br><br><br>

        <button type="submit" class="btn btn-primary btn-block btn-large" >Register</button><br><br>

    </form>
    <br>
    <br>
    <b><font style="color:white" "font-family:verdana" size="5">{{text}}</font></b><br><br>

    <a href="/login">Login</a><br><br>

</div>

</body>
</html>
```

Login Page:

```
*login - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">

<style>
body {
  background-image: url({{ url_for('static', filename='templates/air quality image.JPG') }});

  background-repeat: no-repeat;
  background-size: 100% 100%;
  background-position: center;
  background-size: cover;
}
</style>
</head>

<body background="{{ url_for('static', filename='air quality image.JPG') }}">
  <div class="login" align="center">
    <b><font style="color:white" font-family:verdana size="5"><h1>LOGIN</h1></font></b>

    <form action="{{ url_for('logged_in') }}" method="post">
      <input type="text" name="user_name" placeholder="user_name" required="required" /><br><br><br>
      <input type="password" value="FakePSW" id="myInput" name="password" placeholder="password" required="required" /><br><br><br>

      <button type="submit" class="btn btn-primary btn-block btn-large">Login</button>

    </form>

    <b><font style="color:white" font-family:verdana size="5">{{text}}</font></b><br><br>
  </div>
</body>
</html>
Ln 33, Col 8 100% Windows (CRLF) UTF-8
Type here to search 13:20 27-05-2021
```

Prediction Page Code:

```
index - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html >

<head>

<style>

body {
  background-image: url({{ url_for('static', filename='templates/air quality image.JPG') }})

  background-repeat: no-repeat;
  background-size: 100% 100%;
  background-position: center;
  background-size: cover;
}

label:after {
  content: " ";
}

}

label {
  color: white;
  font-weight: bold;
  display: inline-block;
  width: 100px;
  text-align:center;
}

form button[type='submit'] {
  display: inline-block;
  width: 70px;
}
```

```
index - Notepad
File Edit Format View Help
</style>

</head>

<body background="{{ url_for('static', filename='air quality image.JPG') }}">
<div class="login" align="center" >
  <br><font style="color:white" "font-family:verdana" size="5"><h1 >Air Quality Prediction</h1></font></b>

  <form action="{{ url_for('predict') }}"method="post">
    <label for="Temperature">Temperature</label>
    <input type="text" name="Temperature" id="Temperature" placeholder="" required="required"/><br><br>
    <label for="Humidity">Humidity</label>
    <input type="text" name="Humidity" id="Humidity" placeholder="" required="required"/><br><br>
    <label for="Wind Speed">Wind Speed</label>
    <input type="text" name="Wind Speed" id="Wind Speed" placeholder="" required="required"/><br><br>
    <label for="Visibility">Visibility</label>
    <input type="text" name="Visibility" id="Visibility" placeholder="" required="required"/><br><br>
    <label for="Pressure">Pressure</label>
    <input type="text" name="Pressure" id="Pressure" placeholder="" required="required"/><br><br>
    <label for="so2">so2</label>
    <input type="text" name="so2" id="so2" placeholder="" required="required"/><br><br>
    <label for="no2">no2</label>
    <input type="text" name="no2" id="no2" placeholder="" required="required"/><br><br>
    <label for="Rainfall">Rainfall</label>
    <input type="text" name="Rainfall" id="Rainfall" placeholder="" required="required"/><br><br>
    <label for="PM10">PM10</label>
    <input type="text" name="PM10" id="PM10" placeholder="" required="required"/><br><br>
    <label for="PM25">PM25</label>
    <input type="text" name="PM25" id="PM25" placeholder="" required="required"/><br><br>

    <center> <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button></center>

  </form>

<br>
<br>
```

```
index - Notepad
File Edit Format View Help

<input type="text" name="Temperature" id="Temperature" placeholder="" required="required"/><br><br><br>
<label for="Humidity">Humidity</label>
<input type="text" name="Humidity" id="Humidity" placeholder="" required="required"/><br><br><br>
<label for="Wind Speed">Wind Speed</label>
<input type="text" name="Wind Speed" id="Wind Speed" placeholder="" required="required"/><br><br><br>
<label for="Visibility">Visibility</label>
<input type="text" name="Visibility" id="Visibility" placeholder="" required="required"/><br><br><br>
<label for="Pressure">Pressure</label>
<input type="text" name="Pressure" id="Pressure" placeholder="" required="required"/><br><br><br>
<label for="so2">so2</label>
<input type="text" name="so2" id="so2" placeholder="" required="required"/><br><br><br>
<label for="no2">no2</label>
<input type="text" name="no2" id="no2" placeholder="" required="required"/><br><br><br>
<label for="Rainfall">Rainfall</label>
<input type="text" name="Rainfall" id="Rainfall" placeholder="" required="required"/><br><br><br>
<label for="PM10">PM10</label>
<input type="text" name="PM10" id="PM10" placeholder="" required="required"/><br><br><br>
<label for="PM25">PM25</label>
<input type="text" name="PM25" id="PM25" placeholder="" required="required"/><br><br><br>

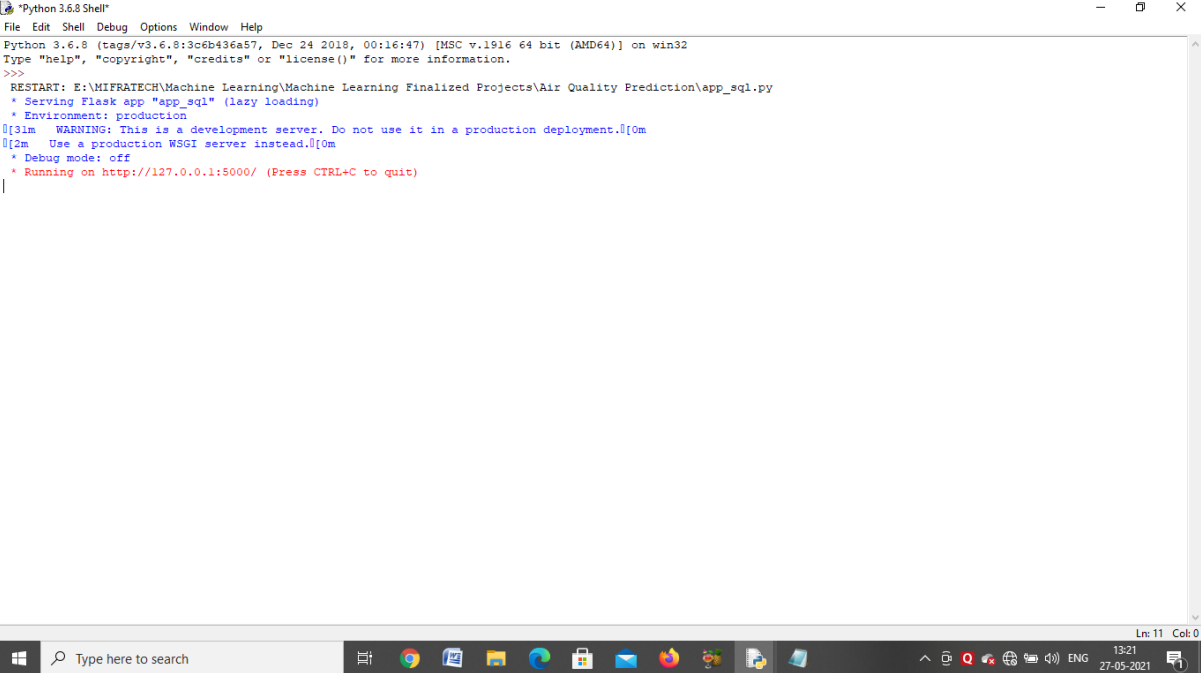
<center> <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button></center>

</form>

<br>
<br>
<b><font style="color:white" "font-family:verdana" size="5">{{ prediction_text}}</font></b>
<br>
<b><font style="color:white" "font-family:verdana" size="5">{{ prediction_text1}}</font></b>
</div>

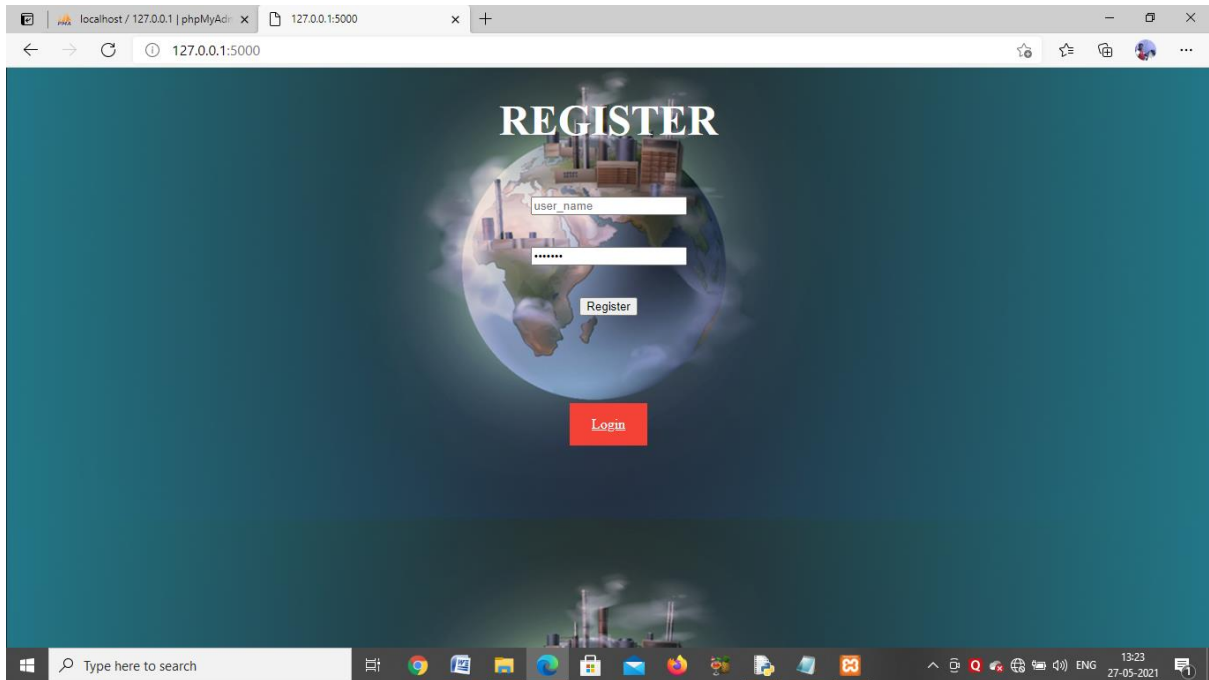
</body>
</html>
```

Front End Output:

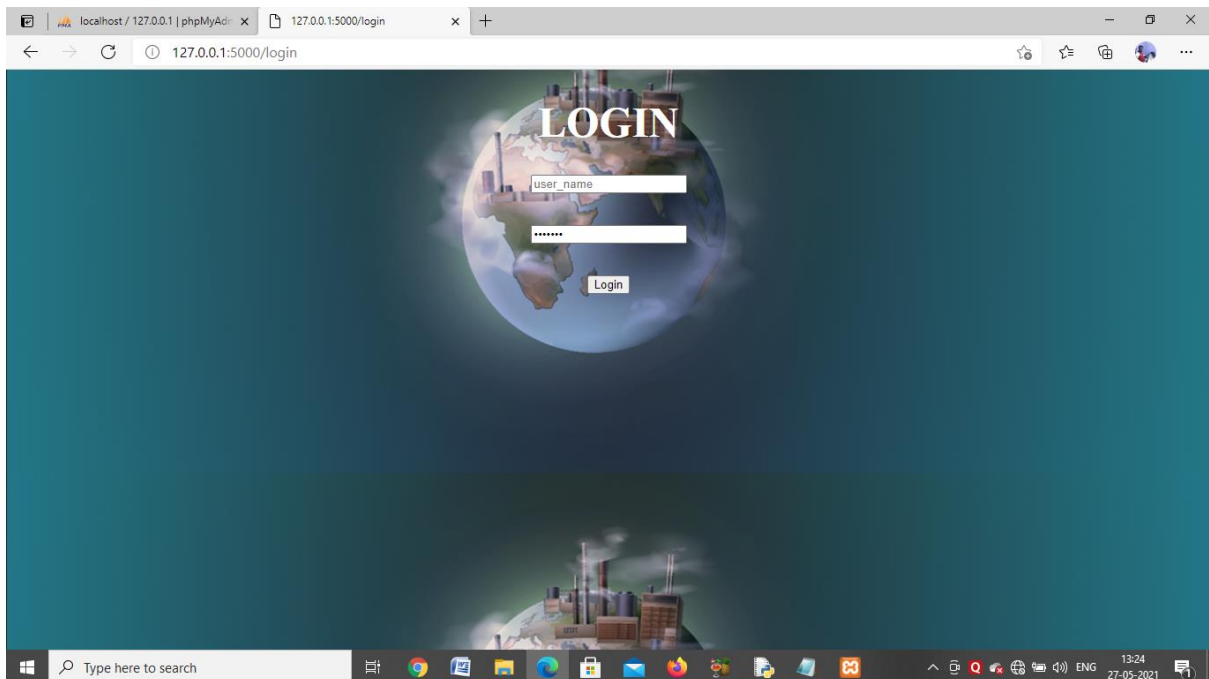


```
Python 3.6.8 Shell
File Edit Shell Debug Options Window Help
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: E:\MIFRATECH\Machine Learning\Machine Learning Finalized Projects\Air Quality Prediction\app_sql.py
* Serving Flask app "app_sql" (lazy loading)
* Environment: production
[[31m WARNING: This is a development server. Do not use it in a production deployment.
[[2m Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Register Page:



Login Page:



Prediction Page:

A screenshot of a web browser displaying the 'Air Quality Prediction' page. The page has a dark teal background with a central graphic of a globe showing Africa and Europe, with industrial smokestacks emitting smoke. Overlaid on the right side of the globe are ten input fields, each with a label and a white text box. The labels are: Temperature, Humidity, Wind Speed, Visibility, Pressure, so2, no2, Rainfall, PM10, and PM25. The browser's address bar shows '127.0.0.1:5000/logedin'. The Windows taskbar at the bottom shows the search bar and various application icons.

Air Quality Prediction

Temperature:

Humidity:

Wind Speed:

Visibility:

Pressure:

so2:

no2:

Rainfall:

PM10:

PM25:

A second screenshot of the same 'Air Quality Prediction' web application. This view includes a 'Predict' button located at the bottom center of the input area, below the PM25 field. The rest of the interface, including the globe graphic, labels, and input fields, is identical to the first screenshot. The browser and taskbar details are also consistent.

Temperature:

Humidity:

Wind Speed:

Visibility:

Pressure:

so2:

no2:

Rainfall:

PM10:

PM25:

Final Output of the Project:

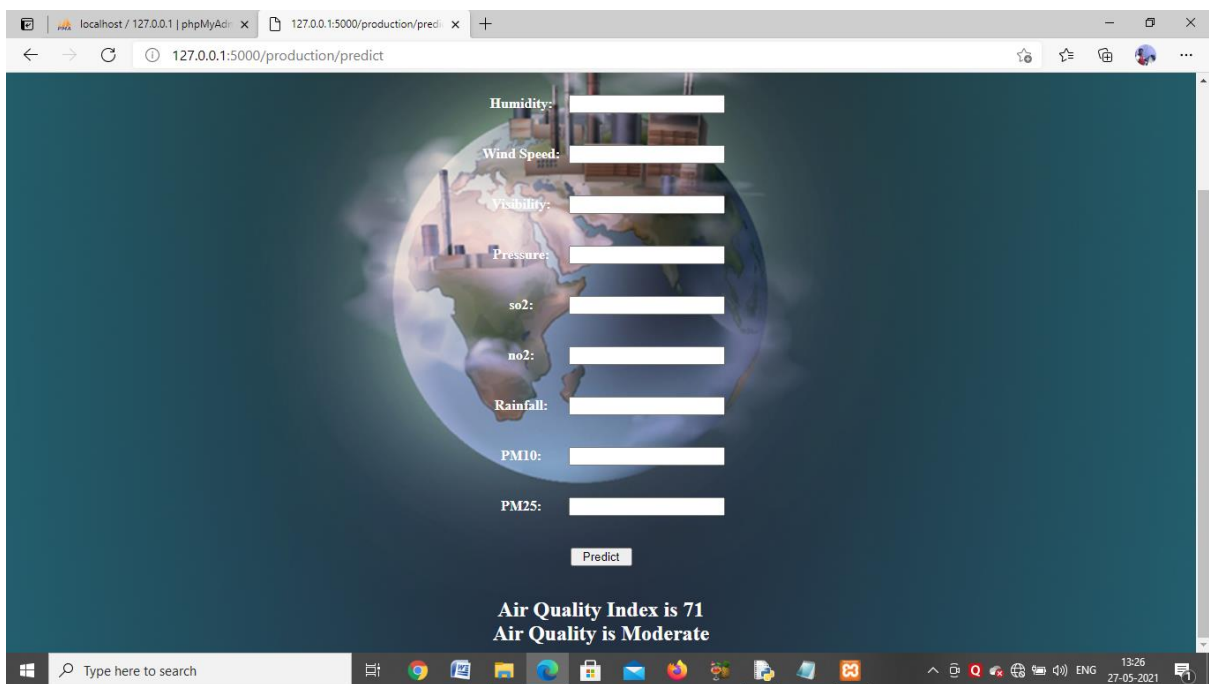
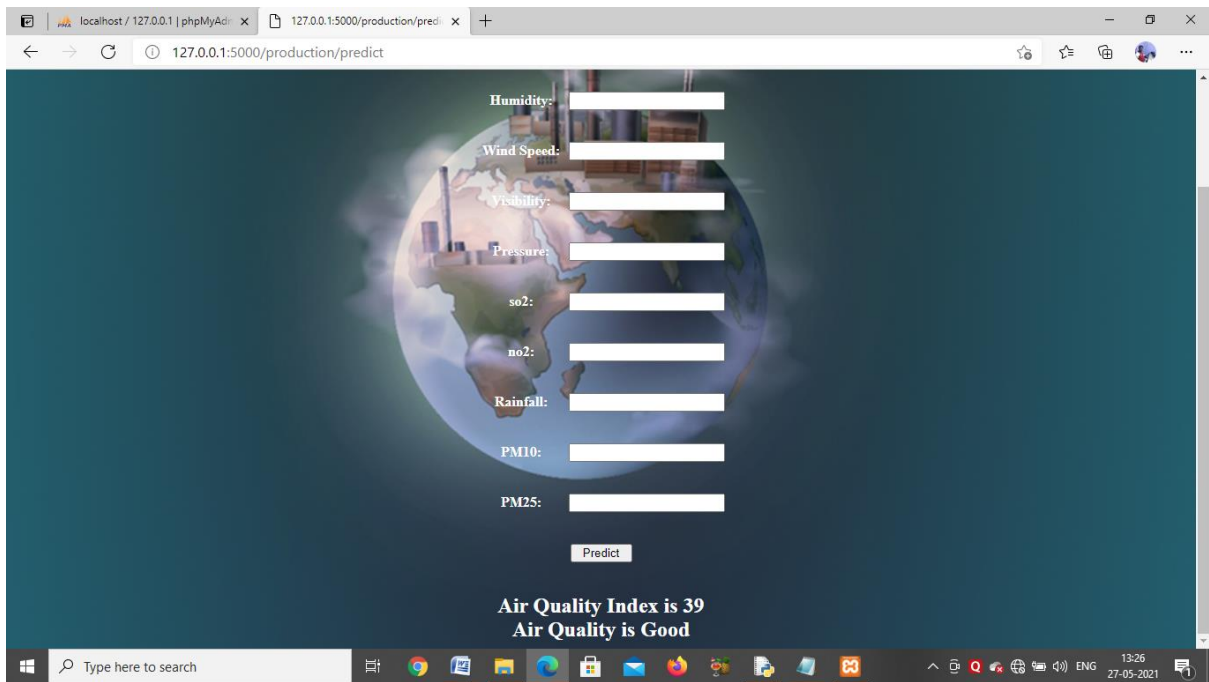
A screenshot of a web browser displaying the 'Air Quality Prediction' application. The interface features a dark teal background with a central graphic of a globe showing industrial smokestacks. Overlaid on the right side of the globe are ten input fields, each with a label and a numerical value. The labels are: Temperature, Humidity, Wind Speed, Visibility, Pressure, so2, no2, Rainfall, PM10, and PM25. The values entered are: 34, 67, 11, 13, 33, 11.3, 22, 22, 22, and 9 respectively. The browser's address bar shows '127.0.0.1:5000/logedin'. The Windows taskbar at the bottom includes a search bar and various application icons.

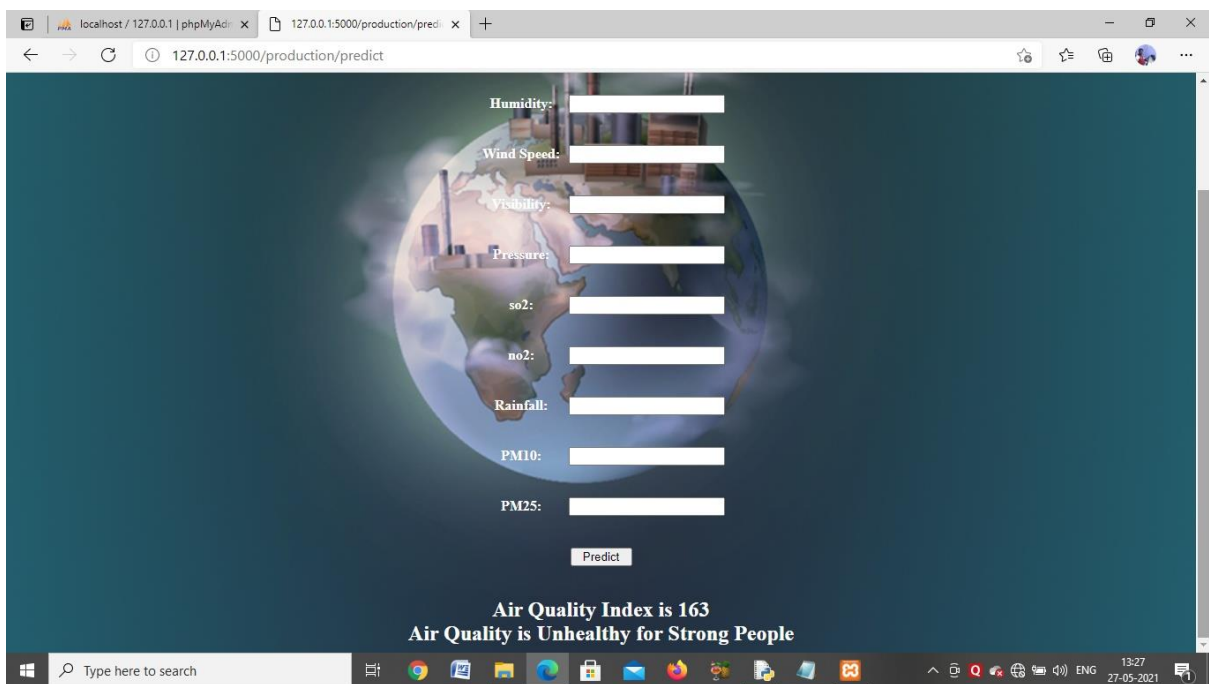
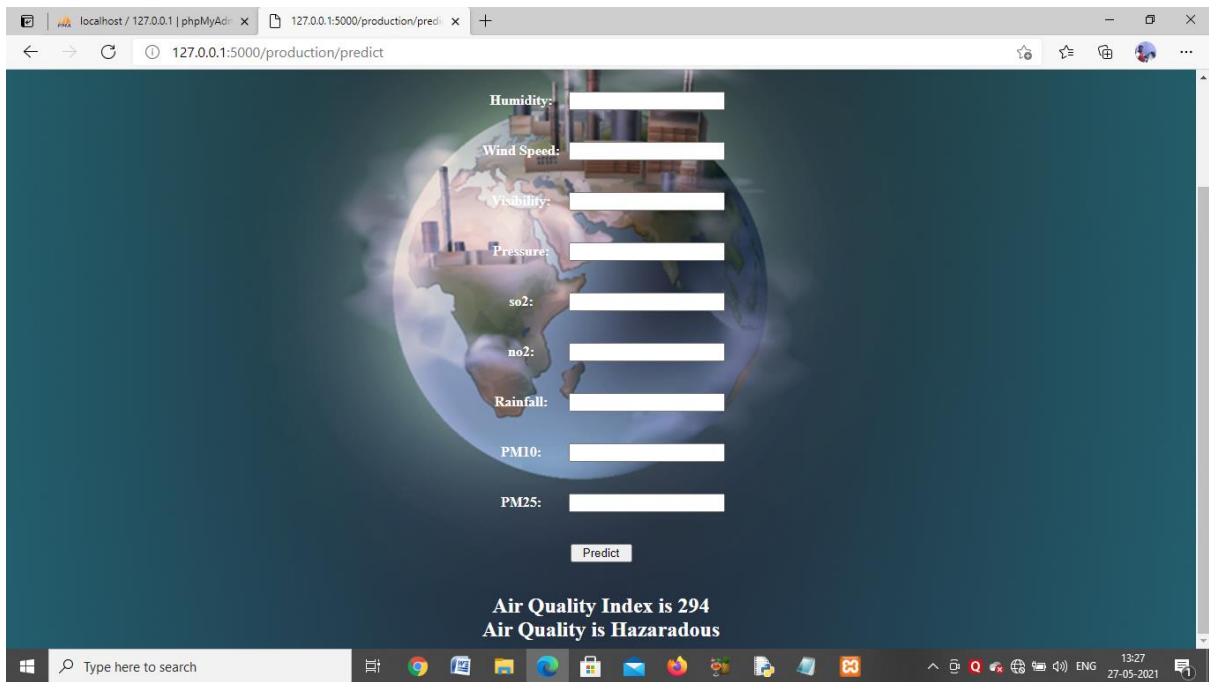
Parameter	Value
Temperature:	34
Humidity:	67
Wind Speed:	11
Visibility:	13
Pressure:	33
so2:	11.3
no2:	22
Rainfall:	22
PM10:	22
PM25:	9

A second screenshot of the same 'Air Quality Prediction' web application. This view includes a 'Predict' button located at the bottom center of the input area, below the PM25 field. The input values are identical to the first screenshot: Temperature (34), Humidity (67), Wind Speed (11), Visibility (13), Pressure (33), so2 (11.3), no2 (22), Rainfall (22), PM10 (22), and PM25 (9). The browser interface and Windows taskbar remain the same.

Parameter	Value
Temperature:	34
Humidity:	67
Wind Speed:	11
Visibility:	13
Pressure:	33
so2:	11.3
no2:	22
Rainfall:	22
PM10:	22
PM25:	9

Predict





localhost / 127.0.0.1 | phpMyAdmin x 127.0.0.1:5000/production/predict x

127.0.0.1:5000/production/predict

Humidity:

Wind Speed:

Visibility:

Pressure:

so2:

no2:

Rainfall:

PM10:

PM25:

Predict

Air Quality Index is 148
Air Quality is Unhealthy

Type here to search

13:31 27-05-2021

Conclusion

According to World Health Organization, 7 million people are at health risk due to air pollution. It is a leading risk factor for majority of health problems like asthma, skin infections, heart issues, throat and eye diseases, bronchitis, lungs cancer and respiratory system's diseases. Besides the health problems related to air pollution, it also poses a serious threat to our planet. In order reduce this air pollution predicting the air quality in early stage accurately helps to take the better decisions to maintain the air quality.

So our proposed system using machine learning techniques and algorithms like Linear Regresson and Random Forest Regressor to predict the air quality index and also classifies air quality is good, moderate, unhealthy ,unhealthy for strong people and hazardous based on the user entered new input data successfully with the accuracy of 99.84% given by random forest regressor algorithm.

References

1. T. M. Amado and J. C. Dela Cruz, "Development of Machine Learning-based Predictive Models for Air Quality Monitoring and Characterization," TENCON 2018 - 2018 IEEE Region 10 Conference, Jeju, Korea (South), 2018, pp. 0668-0672, doi: 10.1109/TENCON.2018.8650518.
2. H. Zheng, Y. Cheng and H. Li, "Investigation of model ensemble for fine-grained air quality prediction," in China Communications, vol. 17, no. 7, pp. 207-223, July 2020, doi: 10.23919/J.CC.2020.07.015.
3. K. Nandini and G. Fathima, "Urban Air Quality Analysis and Prediction Using Machine Learning," 2019 1st International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE), Bangalore, India, 2019, pp. 98-102, doi: 10.1109/ICATIECE45860.2019.9063845.
4. R. Yang, H. Zhou and D. Ding, "Air Quality Prediction Method in Urban Residential Area," 2018 11th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 2018, pp. 16-20, doi: 10.1109/ISCID.2018.00010.
5. U. Mahalingam, K. Elangovan, H. Dobhal, C. Valliappa, S. Shrestha and G. Kedam, "A Machine Learning Model for Air Quality Prediction for Smart Cities," 2019 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET), Chennai, India, 2019, pp. 452-457, doi: 10.1109/WiSPNET45539.2019.9032734.
6. S. Mahanta, T. Ramakrishnudu, R. R. Jha and N. Tailor, "Urban Air Quality Prediction Using Regression Analysis," TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), Kochi, India, 2019, pp. 1118-1123, doi: 10.1109/TENCON.2019.8929517.
7. H. Ayyalasomayajula, E. Gabriel, P. Lindner and D. Price, "Air Quality Simulations Using Big Data Programming Models," 2016 IEEE Second International Conference on Big Data Computing Service and

Applications (BigDataService), Oxford, 2016, pp. 182-184, doi: 10.1109/BigDataService.2016.26.

8. A. S. Alsaedi and L. Liyakathunisa, "Spatial and Temporal Data Analysis with Deep Learning for Air Quality Prediction," 2019 12th International Conference on Developments in eSystems Engineering (DeSE), Kazan, Russia, 2019, pp. 581-587, doi: 10.1109/DeSE.2019.00111.
9. B. Baran, "Prediction of Air Quality Index by Extreme Learning Machines," 2019 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, Turkey, 2019, pp. 1-8, doi: 10.1109/IDAP.2019.8875910.
10. S. Ameer et al., "Comparative Analysis of Machine Learning Techniques for Predicting Air Quality in Smart Cities," in IEEE Access, vol. 7, pp. 128325-128338, 2019, doi: 10.1109/ACCESS.2019.2925082.
11. C. Srivastava, S. Singh and A. P. Singh, "Estimation of Air Pollution in Delhi Using Machine Learning Techniques," 2018 International Conference on Computing, Power and Communication Technologies (GUCON), Greater Noida, Uttar Pradesh, India, 2018, pp. 304-309, doi: 10.1109/GUCON.2018.8675022.
12. S. Jeya and L. Sankari, "Air Pollution Prediction by Deep Learning Model," 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2020, pp. 736-741, doi: 10.1109/ICICCS48265.2020.9120932.
13. K. Maheshwari and S. Lamba, "Air Quality Prediction using Supervised Regression Model," 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), GHAZIABAD, India, 2019, pp. 1-7, doi: 10.1109/ICICT46931.2019.8977694.