Pune Institute of Computer Technology
Dhankawadi, Pune

DMW PROJECT REPORT
ON

AIR QUALITY PREDICTION

SUBMITTED BY

Chetan Atole - 41307
Prem Bansod - 41310
Shailesh Borate - 41315
Class - BE 3

DEPARTMENT OF COMPUTER ENGINEERING
Academic Year 2020-21

Contents

1 TOPIC **1**
1.1 Title: Air Quality Prediction 1
1.2 Problem Statement 1
1.3 Objectives 1
1.4 Outcomes 1
1.5 Software and Hardware Requirements 1

2 DATASET DETAILS **2**
2.1 Data Analysis 2
2.2 Data Preparation 4

3 THEORY CONCEPTS **6**
3.1 Machine Learning Models Used 6

4 IMPLEMENTATION DETAILS **8**
4.1 GUI Details 8
4.1.1 Tkinter 8
4.1.2 GUI Implementation 8
4.2 Integration 8

5 TEST CASES **9**

6 RESULTS **11**
6.1 Support Vector Machine 11
6.2 eXtreme Gradient Boosting 12
6.3 Decision Tree 12
6.4 Random Forest 13
6.5 Output Screenshots 14

7 CONCLUSION **18**

List of Figures

1 Dataset Information 2
2 Dataset BoxPlot 3
3 Dataset Distribution 3
4 Training Dataset Distribution 4
5 Training Dataset Distribution After applying SMOTE 5
6 Confusion Matrix Support Vector Machine 11
7 Confusion Matrix Extreme Gradient Boost 12
8 Confusion Matrix Decision Tree 12
9 Confusion Matrix Random Forest 13
10 First Page 14
11 Training Page 14
12 SVM Model 15
13 XGB Model 15
14 RandomForest Model 16
15 DecisionTree Model 16
16 Single Prediction 17
17 Report Generation 17

List of Tables

1 Test Cases 9
2 Results 11

1 TOPIC

1.1 Title: Air Quality Prediction

1.2 Problem Statement

Consider a labeled dataset belonging to an application domain. Apply suitable data preprocessing steps such as handling of null values, data reduction, discretization. For prediction of class labels of given data instances, build classifier models using different techniques (minimum 3), analyze the confusion matrix and compare these models. Also apply cross validation while preparing the training and testing datasets.

1.3 Objectives

1. Learn how to apply preprocessing steps on a labelled dataset.
2. Learn to build various data classifier models.
3. Learn to split dataset into train and test set and apply cross validation.
4. Learn multiclass prediction and analysis of confusion matrix.

1.4 Outcomes

1. The dataset was analysed using bar charts and confusion matrices.
2. Different models were tested with variations in their parameters.
3. A Graphical User Interface was created over the selected model to make the testing more interactive.
4. Air Quality for different cities was predicted successfully depending upon the amount of various different elements

1.5 Software and Hardware Requirements

1. Operating System : 64-bit Open source Linux or its derivative/Windows
2. Programming Language: Python

2 DATASET DETAILS

2.1 Data Analysis

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26219 entries, 0 to 26218
Data columns (total 16 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   City         26219 non-null  object
 1   Date         26219 non-null  datetime64[ns]
 2   PM2.5        21930 non-null  float64
 3   PM10         15453 non-null  float64
 4   NO           22986 non-null  float64
 5   NO2          23002 non-null  float64
 6   NOx          22176 non-null  float64
 7   NH3          16372 non-null  float64
 8   CO           24258 non-null  float64
 9   SO2          22675 non-null  float64
 10  O3           22559 non-null  float64
 11  Benzene      20932 non-null  float64
 12  Toluene      18664 non-null  float64
 13  Xylene       9412 non-null   float64
 14  AQI          21937 non-null  float64
 15  Air_quality  21937 non-null  object
dtypes: datetime64[ns](1), float64(13), object(2)
memory usage: 3.2+ MB
```

Figure 1: Dataset Information

The Dataset City_Day.csv is taken from Kaggle(https://www.kaggle.com/nareshbhat/air-quality-analysis-eda-and-classification). The dataset consists of 26219 entries indexed from 0 to 26218. There are total 16 columns, description of which is displayed in the figure above.

The dependent variable is the Airquality columns. There are 6 different classes of Airquality namely Good, Poor, Moderate, Satisfactory, severe and very poor

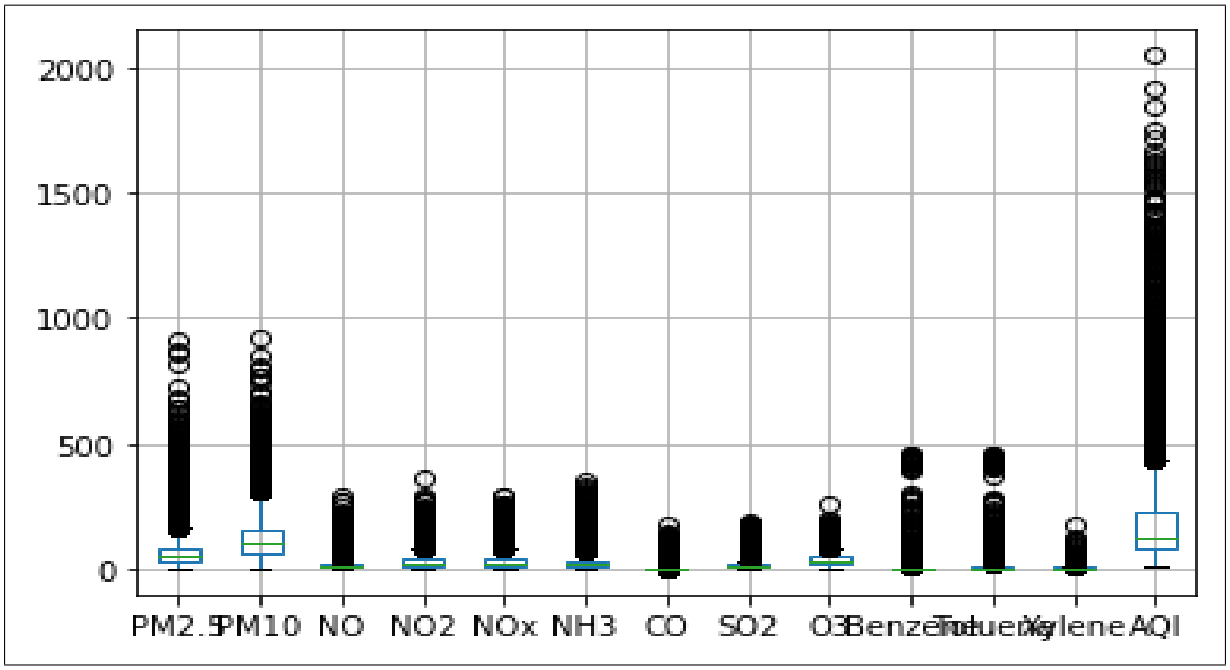

Figure 2: Dataset BoxPlot

The above boxplot gives a good indication of how the values in the data are spread out.

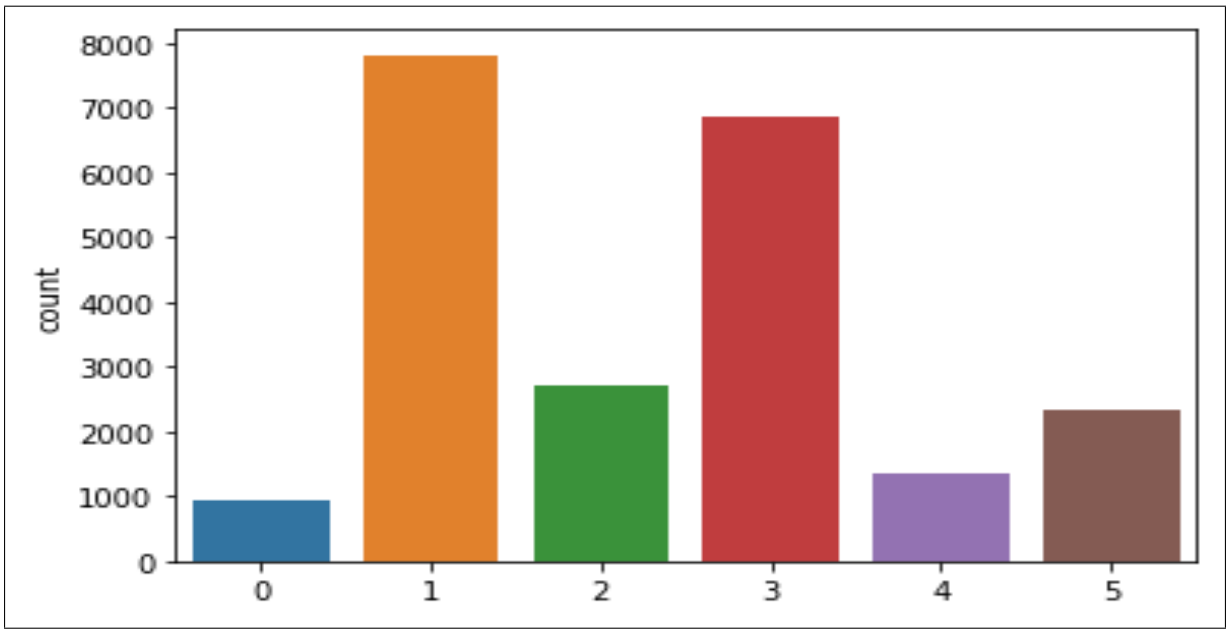

Figure 3: Dataset Distribution

The above figure shows the count of records of all the various classes

2.2 Data Preparation

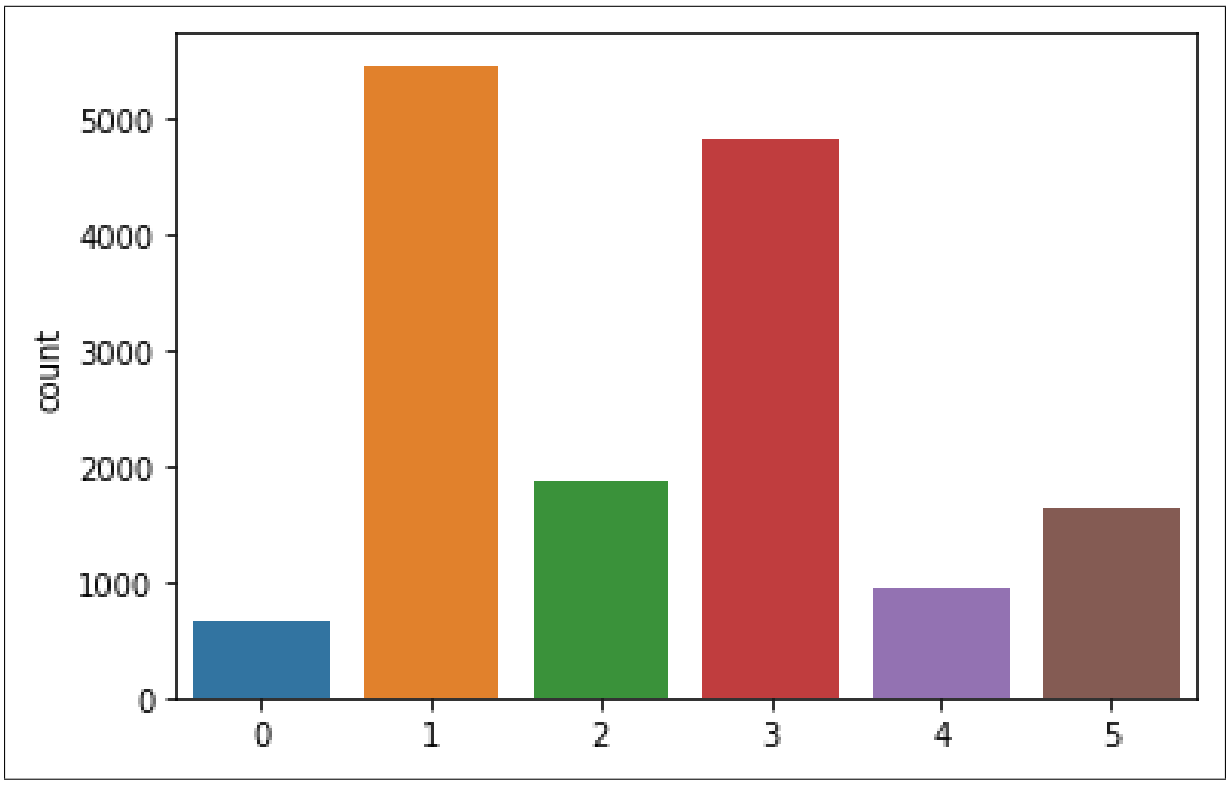

Figure 4: Training Dataset Distribution

Classes and number of values in trainset Counter(Moderate(1): 5450, Satisfactory(3): 4808, Poor(2): 1867, Very Poor(5): 1618, Severe(4): 951, Good(0): 661)

As seen in the above figure, the distribution of the classes is uneven. To make the dataset distribution even, SMOTE technique is used. The Synthetic Minority Over-sampling Technique (SMOTE) is an oversampling approach that creates synthetic minority class samples. SMOTE first selects a minority class instance a at random and finds its k nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b.

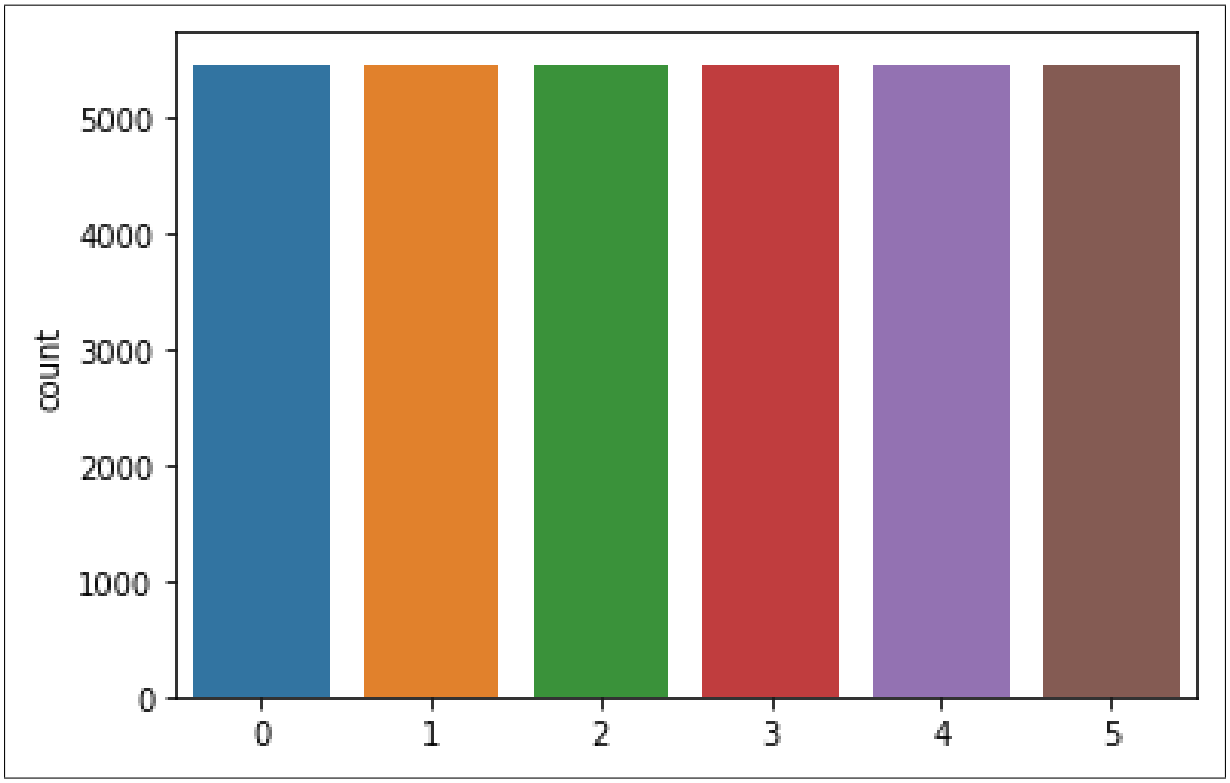

Figure 5: Training Dataset Distribution After applying SMOTE

Classes and number of values in trainset Counter(Moderate(1): 5450, Satisfactory(3): 5450, Poor(2): 5450, Very Poor(5): 5450, Severe(4): 5450, Good(0): 5450)

3 THEORY CONCEPTS

3.1 Machine Learning Models Used

1. **Support Vector Machine (SVM) Classifier**

   SVM is a supervised machine learning model. It is used for both regression as well as classification purpose, but mostly used in classification problems. The main objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N being the number of features) provided the hyperplane distinctly classify the data points. SVM has a good prediction efficiency and performance rate.

   We are using an SVM classifier to classify the given input. There are six types of output air quality and 16 input fields. Linear kernel, support vectors, and hyperplane to classify input values. Support vector Classifier takes Input values and provides support vector value compared with all the six classifiers support vectors values and classified given input based on nearest support vectors values.

2. **Decision Tree Classifier**

   There are two main types of Decision Tree - Classification Tree and Regression Tree. It is a supervised machine learning algorithm wherein data is frequently split according to a certain variable. The decision variable in the Classification tree is discrete/categorical. The decision tree asks a series of questions about the attributes of the record. Each time it receives an answer, it further asks up a question till it reaches about a conclusion on the label of the class record.

   We are using a Decision Tree classifier to classify the given input. There are six types of output air quality and 16 input fields. Given Dataset split in different splits based on the group of air quality. Input is processed using different conditions and leads to the leaf node using certain conditions. The leaf node of the decision tree consists of the output classified air quality.

3. **XGBoost Classifier**

   XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now.

   XGB classifier is excessively used in Kaggle competitions due to its speed and performance. Gradient boosting involves sequential formation of Decision trees such that, the trees formed later try to reduce the mistakes of weak learners formed before it. Here, in order to classify the Air Quality the number of decision tree formed are 100. XGB was able to classify all the test instances correctly to acheive an accuracy of 100%.

4. **Random Forest Classifier**

Random forest classifier creates a set of decision trees from randomly selected subset of training set. It then aggregates the votes from different decision trees to decide the final class of the test object. Random decision forests correct for decision trees' habit of overfitting to their training set.Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

We are also using a Random Forest Classifier to classify the given input. There are six types of output air quality and 16 input fields. The hyperparameters n_estimators, max_features, min_sample_leaf are used for increasing the predictive power. The hyperparameters n_jobs, random_state, oob_score are used for increasing the model's speed. For our model we have set n_estimator value to 100 i.e 100 decision trees are created and their average value is taken. Similarly criterion is set to "gini"(calculates the amount of probability of a specific feature that is classified incorrectly when selected randomly), min_samples_split is set to 2, max_features is set to "auto", random_state is set to 0(to produce the same result every time)

4 IMPLEMENTATION DETAILS

4.1 GUI Details

4.1.1 Tkinter

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

To create a Tkinter app:

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

4.1.2 GUI Implementation

Initially the user has to select the CSV file of the dataset. Once the CSV file is selected, the dataset will be loaded in backend and a Train Button will be rendered on the screen. When user presses the Train Button all the 4 Classifiers (SVM, Random Forest, Decision Tree and XGB) will be trained and the training progress will be shown to the user through the progress bar. Once the classifiers are trained the user will be directed to a page where the user will be provided with two options (Test/Predict). If the user selects the Test option then he/she needs to select the classifier from drop down menu and click Test. Once the Test Button is clicked, the Accuracy, Precision, Recall and F1 Score will be rendered along with the Confusion Matrix. If the user presses the Predict Button, then the user will be directed to a page where he/she can enter the values of the attributes required for Air Quality prediction. If the user doesn't know the values of certain attributes, then the corresponding fields can be left blanked as those fields will be replaced by the median of training data. Once the user clicks on Predict Button, the prediction output of all the 4 classifiers will be displayed on the screen.

4.2 Integration

2 Different files are being maintained, one for backend and other for frontend. The backend file consists of a class named as Air Quailty.The Air Quality class consists of different functions such as readCSV(), train(), test(), predict(), etc. which are being called based on user actions. The frontend file contains the interface design. In frontend file the Air Quality class is imported and an object is created. The Air Quality class functions are mappped to different eventts of the user interface.

5 TEST CASES

Sr. No.	Algorithm Name	Input	Expected Output	Actual Output	Result
1	Decision Tree Classifier	City = Delhi, Date = 01/11/2020, PM 2.5 = 391, PM 10 = 388, NO2 = 116, NH3 = 9, CO = 122, SO2 = 19, O3 = 19, AQI = 391	Very Poor	Very Poor	Pass
2	Random Forest Classifier	City = Delhi, Date = 01/11/2020, PM 2.5 = 391, PM 10 = 388, NO2 = 116, NH3 = 9, CO = 122, SO2 = 19, O3 = 19, AQI = 391	Very Poor	Very Poor	Pass
3	Support Vector Machine Classifier	City = Delhi, Date = 01/11/2020, PM 2.5 = 391, PM 10 = 388, NO2 = 116, NH3 = 9, CO = 122, SO2 = 19, O3 = 19, AQI = 391	Very Poor	Severe	Fail
4	XGBoost Classifier	City = Delhi, Date = 01/11/2020, PM 2.5 = 391, PM 10 = 388, NO2 = 116, NH3 = 9, CO = 122, SO2 = 19, O3 = 19, AQI = 391	Very Poor	Very Poor	Pass

Table 1: Test Cases

1. The input for predicting the air quality for a specific day is taken from the website https://app.cpcbccr.com/AQI$_I ndia/NationalAirQualityIndex$
2. Air Quality was properly predicted using Decision Tree, Random Forest and eXtreme Gradient Boosting machine learning models

6 RESULTS

Classifier	Accuracy	Precision	Recall	F1 Score
Decision Tree	100.0	100.0	100.0	100.0
Random Forest	99.91	99.91	99.91	99.91
Support Vector Machine	90.09	91.92	90.97	90.54
eXtreme Gradient Boosting	100.0	100.0	100.0	100.0

Table 2: Results

6.1 Support Vector Machine

Support Vector Machine classified 5929 test instances correctly out of 6582 test instances and achieved an accuracy of 90.09%. SVM performed well in classifying Severe class as only single test instance was miss classified where as it performed poor for Satisfactory class as 323 test instances were miss classified.

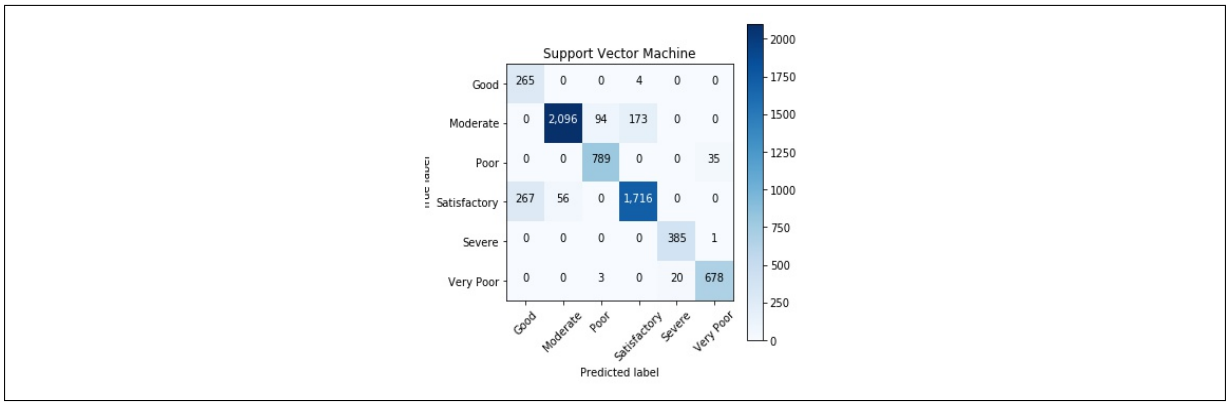

Figure 6: Confusion Matrix Support Vector Machine

6.2 eXtreme Gradient Boosting

eXtreme Gradient Boosting classified all the test instances correctly to achieve an accuracy of 100%. eXtreme Gradient Boosting gave the best results.

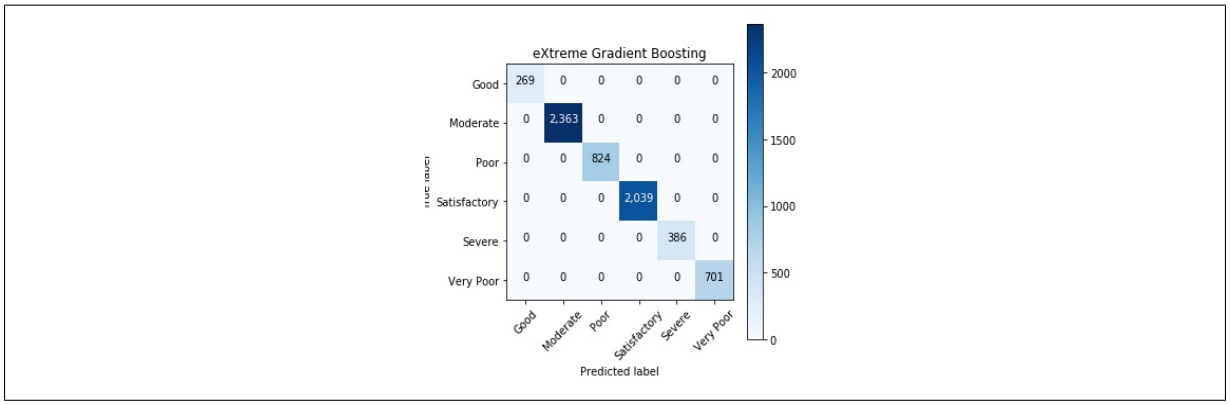

Figure 7: Confusion Matrix Extreme Gradient Boost

6.3 Decision Tree

Decision Tree also classified all the test instances correctly to achieve an accuracy of 100%.

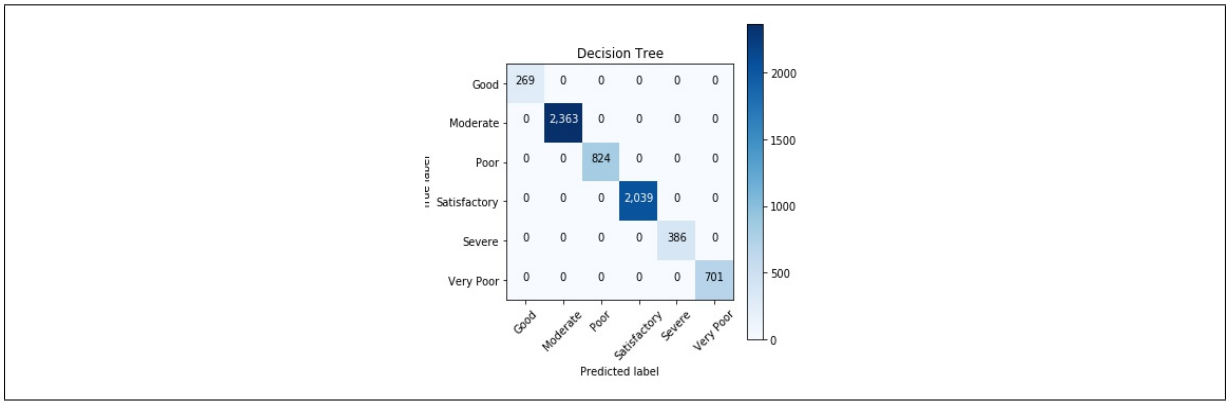

Figure 8: Confusion Matrix Decision Tree

6.4 Random Forest

Random Forest classified 6579 test instances correctly out of 6582 test instances and achieved an accuracy of 99.91%. Only 3 instances were miss classified. One test instance whose actual class was Good was miss classified as Moderate, one test instance whose actual class was Moderate was miss classified as Very Poor and one instance whose actual class was Poor was miss classified as Moderate.

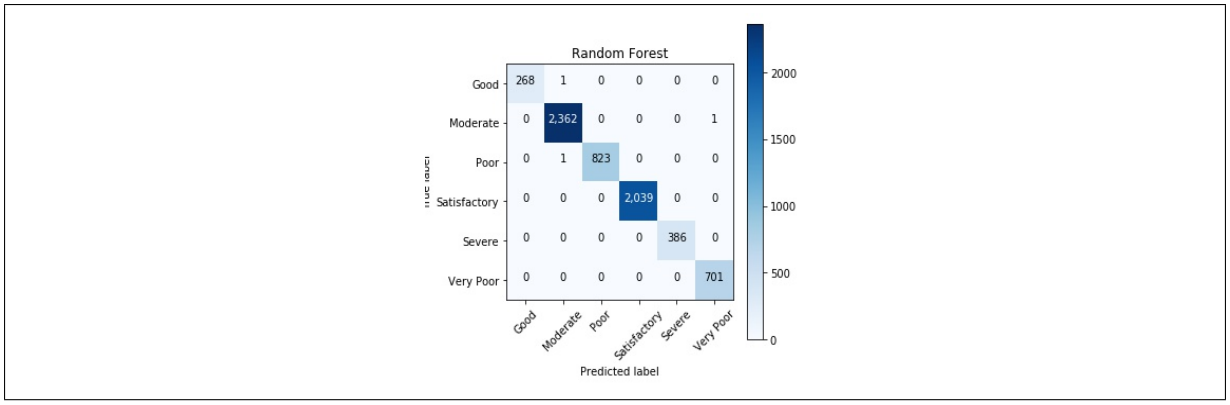

Figure 9: Confusion Matrix Random Forest

6.5 Output Screenshots

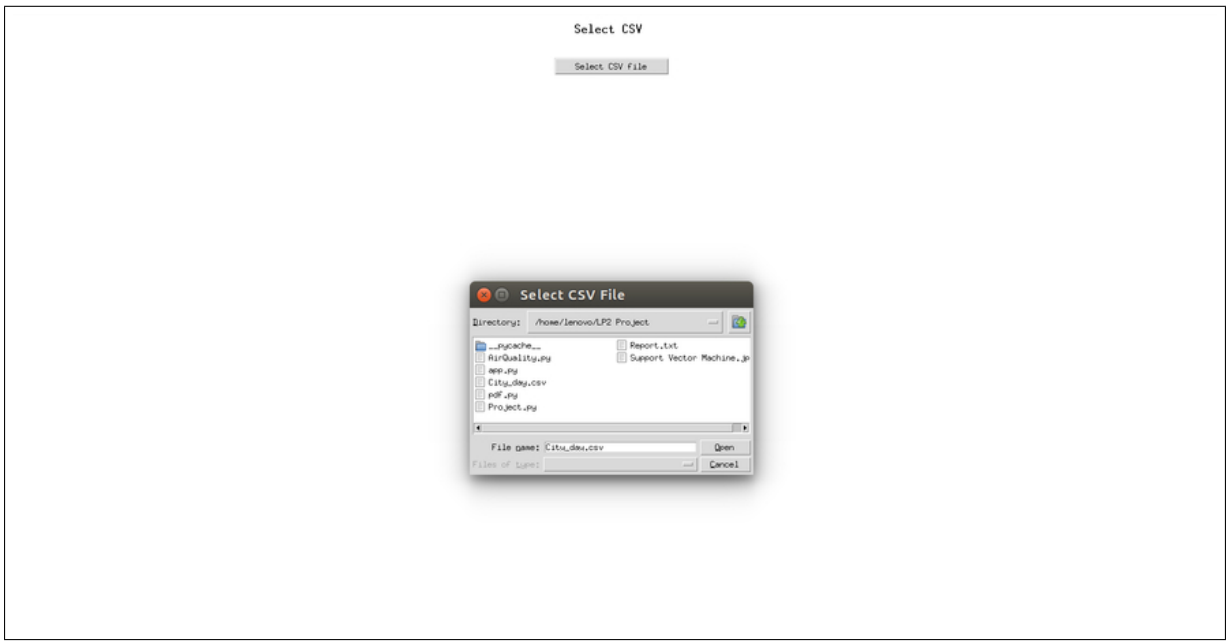

Figure 10: First Page

Figure 11: Training Page

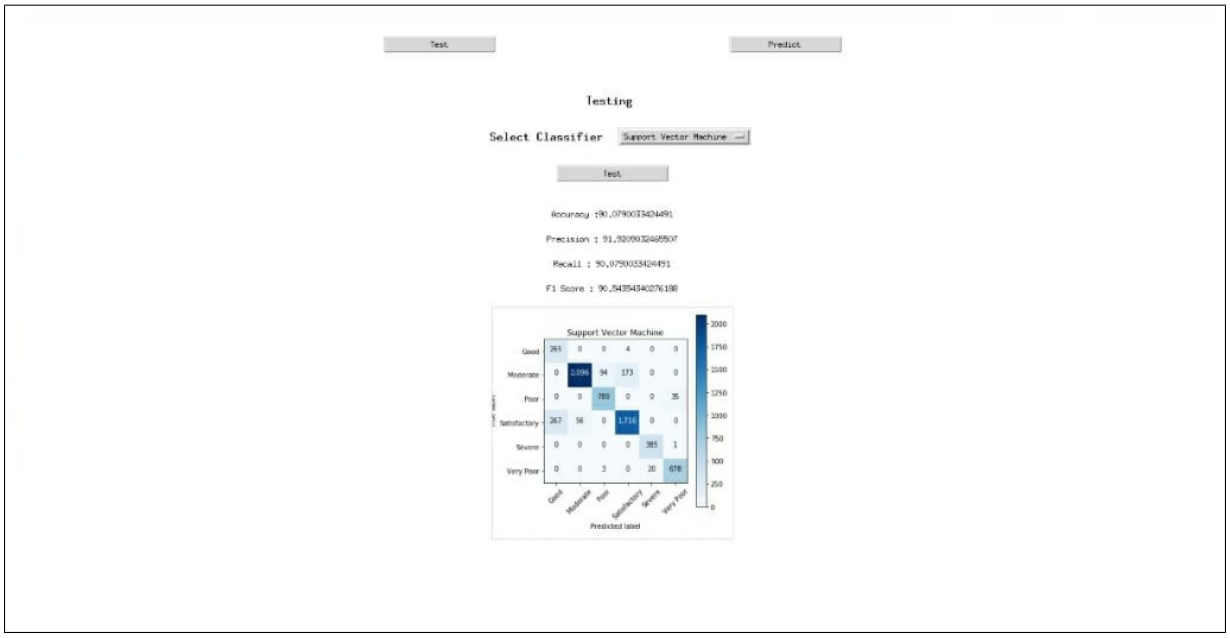

Figure 12: SVM Model

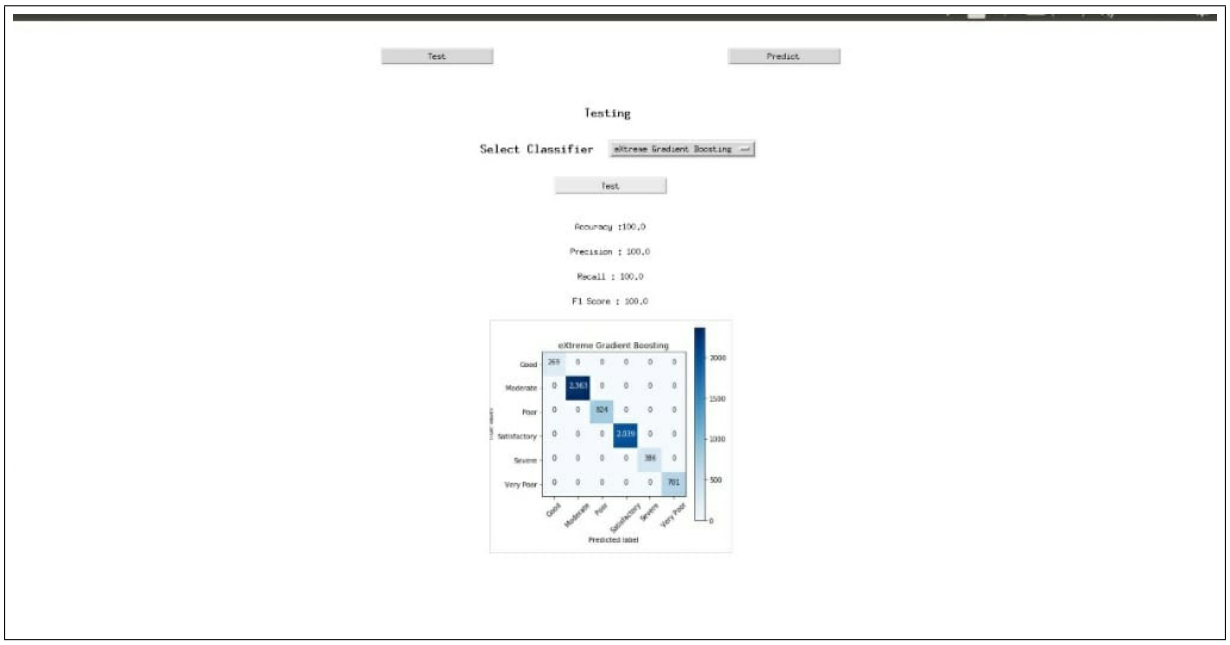

Figure 13: XGB Model

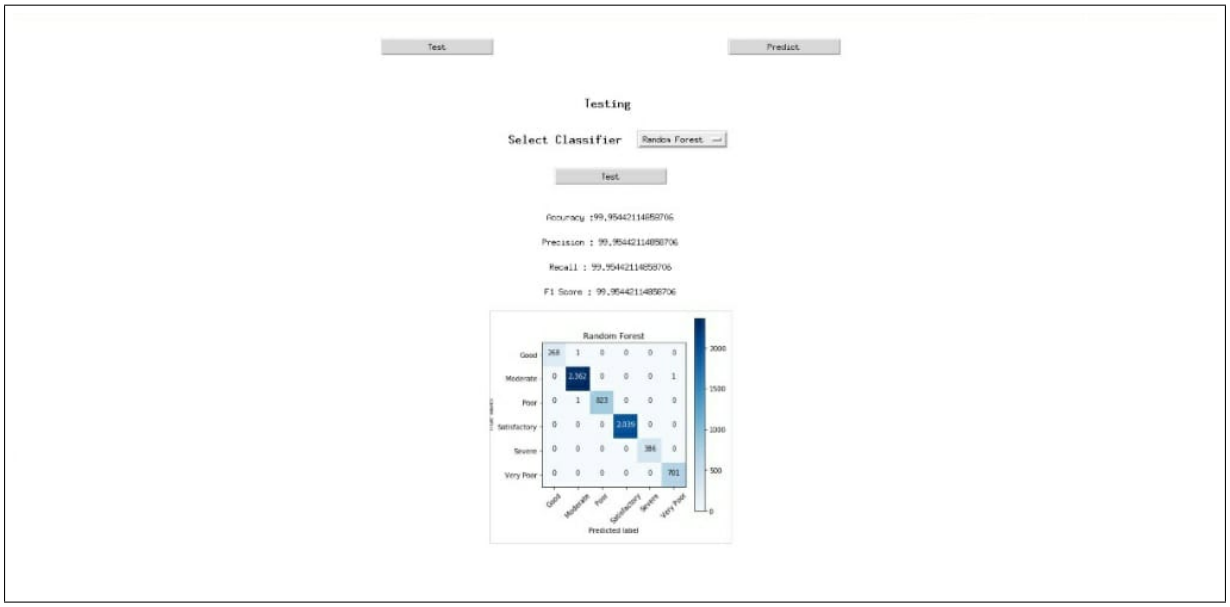

Figure 14: RandomForest Model

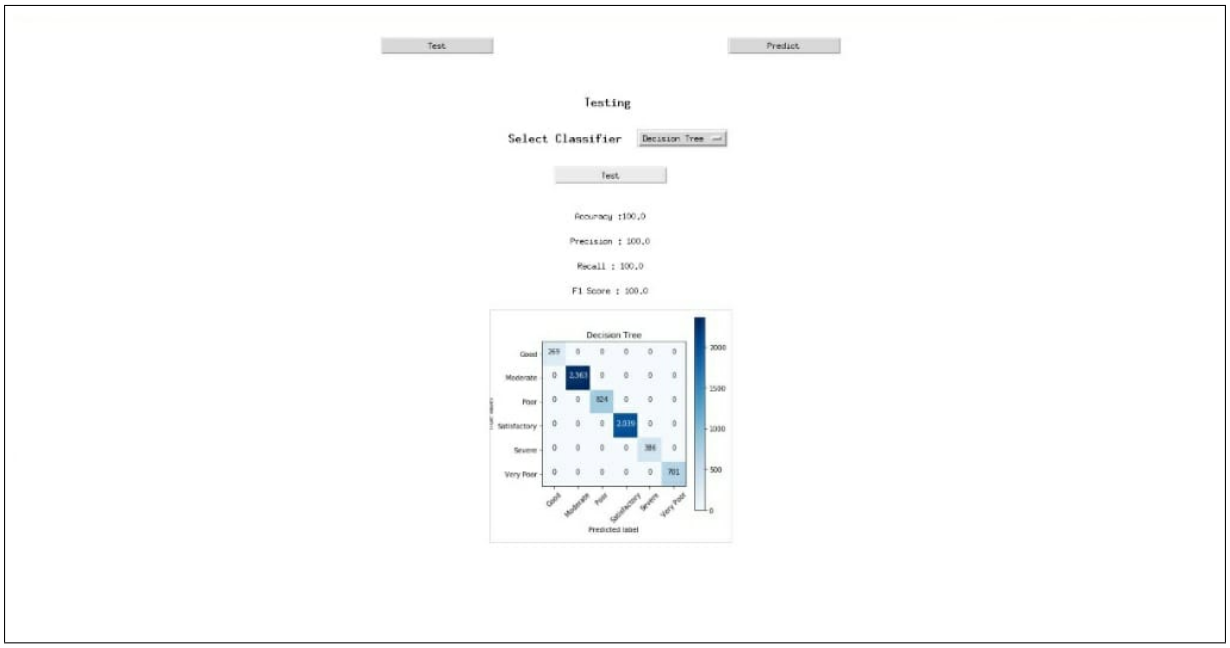

Figure 15: DecisionTree Model

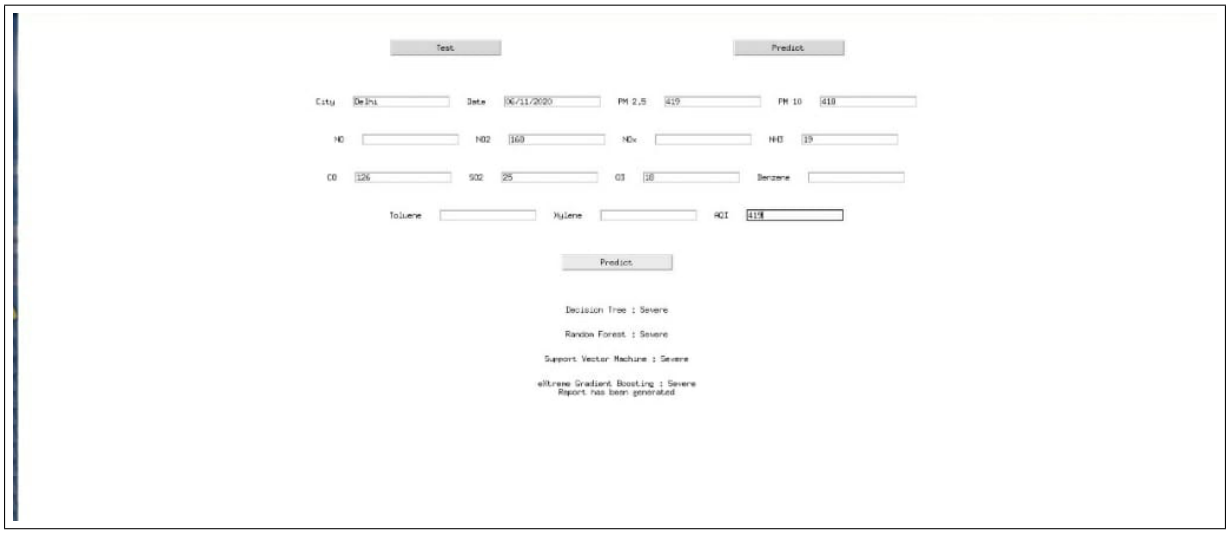

Figure 16: Single Prediction

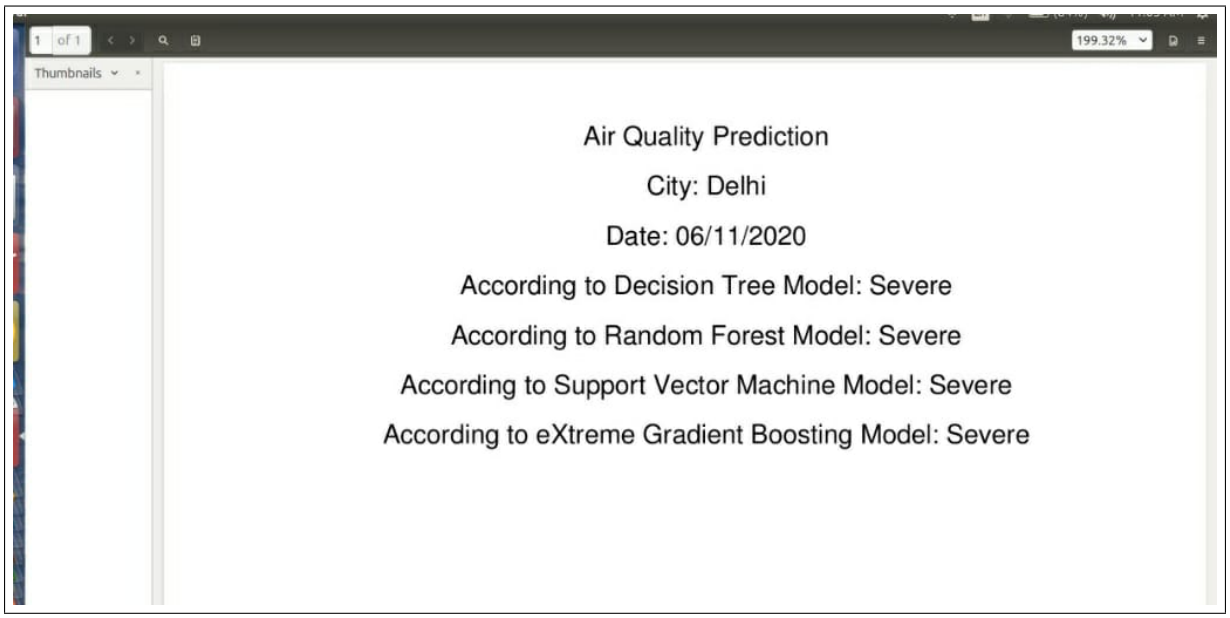

Figure 17: Report Generation

7 CONCLUSION

Predicting the air quality is a complex task due to the dynamic nature, volatility, and high variability in space and time of pollutants and particulates. At the same time, being able to model, predict, and monitor air quality is becoming more and more important, especially in urban areas, due to the observed critical impacts of air pollution for populations and the environment.

In this work, the task of predicting the Air Quality based on the pollutants content in an area was done. The Classifiers used for the task were Decision Tree, Random Forest, Support Vector Machine and eXtreme Gradient Boosting. Out of which eXtreme Gradient Boosting provided the best accuracy of 100%.