

AIR POLLUTION PREDICTION USING DATA SCIENCE TECHNIQUE

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

**Divyadharshini S R (38110135)
Manju R (38110300)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

APRIL - 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600
119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **DIVYADHARSHINI S R (38110135) MANJU R (38110300)** who carried out the project entitled “**AIR POLLUTION PREDICTION USING DATA SCIENCE TECHNIQUE**” under my supervision from **June 2021 to March 2022**.

Internal Guide

MR.VIGNESH R,

M.E(Ph.D.,)

HEAD OF THE DEPARTMENT

DrS. VIGNESHWARI S, M.E., Ph. D.,

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

We, **DIVYADHARSHINI S R (38110135) MANJU R (38110300)** hereby declare that the Project Report entitled “**AIR POLLUTION PREDICTION USING DATA SCIENCE TECHNIQUE**” done by us under the guidance of Mr.Vignesh R (Internal) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering / Technology degree in **Computer Science and Engineering**.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T.Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr.S.Vigneshwari M.E., Ph.D.**, and **Dr.L.Lakshmanan M.E., Ph.D.**, Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mr.Vignesh M.E(Ph.D.)** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Generally, Air pollution refers to the release of pollutants into the air that are detrimental to human health and the planet. It can be described as one of the most dangerous threats that the humanity ever faced. It causes damage to animals, crops, forests etc. To prevent this problem in transport sectors, have to predict air quality from pollutants using machine learning techniques. Hence, air quality evaluation and prediction has become an important research area. The aim is to investigate machine learning based techniques for air quality forecasting by prediction results in best accuracy. The analysis of dataset by supervised machine learning technique (SMLT) to capture several information's like, variable identification, uni-variate analysis, bi-variate and multi-variate analysis, missing value treatments and analyse the data validation, data cleaning/preparing and data visualization will be done on the entire given dataset. Our analysis provides a comprehensive guide to sensitivity analysis of model parameters with regard to performance in prediction of air quality pollution by accuracy calculation. To propose a machine learning-based method to accurately predict the Air Quality Index value by prediction results in the form of best accuracy from comparing supervised classification machine learning algorithms. Additionally, to compare and discuss the performance of various machine learning algorithms from the given transport traffic department dataset with evaluation classification report, identify the confusion matrix and to categorizing data from priority and the result shows that the effectiveness of the proposed machine learning algorithm technique can be compared with best accuracy with precision, Recall and F1 Score.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
	Declaration	iii
	Acknowledgement	iv
	Abstract	v
	Table of Contents	vi
	List of Figures	vii
1	INTRODUCTION	08
	1.1 Data science	08
	1.2 Data scientists	08
	1.3 Artificial Intelligence	09
	1.4 Machine Learning	11
2	LITERATURE SURVEY	13
3	EXPERIMENTAL OR MATERIALS AND METHODS, ALGORITHMS USED	17
	3.1 Air Quality	17
	3.2 Dataset	18
	3.3 Proposed System	18
	3.4 System Architecture	19
	3.5 Module Implementation	21
	3.6 Using Supervised ML Algorithms	23
	3.7 Cross Validation	25
4	RESULTS AND DISCUSSION	28
	4.1 Results and Analysis	28
	4.2 System Requirements	32
5	SUMMARY AND FUTURE SCOPE	35
	5.1 Summary	35
	5.2 Future Scope	36
	References	37
	APPENDIX - Source Code	39
	Screenshots	58

LIST OF FIGURES

S. No	Figures	Page No.
1.4.1	Diagrammatic view machine learning process	12
3.1.1	Average pollutant range in India	17
3.4.1	Process of Dataflow diagram	20
3.5.2.1	Systematic representation of proposed system	22
4.1.1	Result of logistic regression algorithm	28
4.1.2	Result of random forest algorithm	29
4.1.3	Result of naives Bayes algorithm	30
4.1.4	Result of decision tree algorithm	31
B.1	Module 1 Dataset after preprocessing	58
B.2	Module 2 Value count of pollutant and states	59
B.3	Training and testing Dataset - Split	60
B.4	By air quality index	61
B.5	Bar plot representation of each pollutant and their range	62
B.6	Pie chart representation of each pollutant and their range	63
B.7	Module 3 Confusion matrix - predicted label and true label	64
B.8	Heat map representation of the pollutant	65
B.9	Box plot representation of the pollutant	66
B.10	Calculation of positive and negative predictive value	67
B.11	GUI - Results predicted by machine	68

CHAPTER – 1

INTRODUCTION

1. INTRODUCTION

1.1 DATA SCIENCE

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term "data science" was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.

Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

1.2 DATA SCIENTIST:

Data scientists examine which questions need answering and where to find the related data. They have business acumen and analytical skills as well as the ability to mine, clean, and present data. Businesses use data scientists to source, manage, and analyze large amounts of unstructured data.

Required Skills for a Data Scientist:

Programming: Python, SQL, Scala, Java, R, MATLAB.

Machine Learning: Natural Language Processing, Classification, Clustering.

Data Visualization: Tableau, SAS, D3.js, Python, Java, R libraries.

Big data platforms: MongoDB, Oracle, Microsoft Azure, Cloudera.

1.3. ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of "intelligent agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving", however this definition is rejected by major AI researchers.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

AI applications include advanced web search engines, recommendation systems (used by YouTube, Amazon and Netflix), Understanding human speech (such as Siri or Alexa), self-driving cars (e.g. Tesla), and competing at the highest level in strategic game systems (such as chess and Go), As machines become increasingly capable, tasks considered to require "intelligence" are often removed from the definition of AI, a phenomenon known as the AI effect. For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. AI research has tried and discarded many different approaches during its lifetime, including simulating the brain, modeling human problem solving, formal logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly successful, helping to solve many challenging problems throughout industry and academia.

The various sub-fields of AI research are centered around particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field's long-term goals. To solve these problems, AI researchers use versions of search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics.

The field was founded on the assumption that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence. These issues have been explored by myth, fiction and philosophy since antiquity. Science fiction and futurology have also suggested that, with its enormous potential and power, AI may become an existential risk to humanity. As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning. AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce life like exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.

AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

Learning processes. This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

Reasoning processes. This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

Self-correction processes. This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

AI is important because it can give enterprises insights into their operations that they may not have been aware of previously and because, in some cases, AI can perform tasks better than humans. Particularly when it comes to repetitive, detail-oriented tasks like analyzing large numbers of legal documents to ensure relevant fields are filled in properly, AI tools often complete jobs quickly and with relatively few errors.

Artificial neural networks and deep learning artificial intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.

Natural Language Processing (NLP):

Natural language processing (NLP) allows machines to read and understand human language. A

sufficiently powerful natural language processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval, text mining, question answering and machine translation. Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. "Keyword spotting" strategies for search are popular and scalable but dumb; a search query for "dog" might only match documents with the literal word "dog" and miss a document with the word "poodle". "Lexical affinity" strategies use the occurrence of words such as "accident" to assess the sentiment of a document. Modern statistical NLP approaches can combine all these strategies as well as others, and often achieve acceptable accuracy at the page or paragraph level. Beyond semantic NLP, the ultimate goal of "narrative" NLP is to embody a full understanding of commonsense reasoning. By 2019, transformer-based deep learning architectures could generate coherent text.

1.4. MACHINE LEARNING

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labeling to learn data has to be labeled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they "learn" about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program

learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.



Fig 1.4.1 Diagrammatic view of machine learning process

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is $y = f(X)$. The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression, multi-class classification, Decision Trees and support vector machines etc. Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

CHAPTER 2

LITERATURE SURVEY

General

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them

Review of Literature Survey:

Title : The Prediction of Indoor Air Quality in Office Room Using Artificial Neural Network

Author: Jouvan Chandra Pratama Putra^{1, a)}, Safrilah ^{1, b)} and Mohammad Ihsan

Year : 2018

The investigation of indoor air quality has been attracted a number of researchers in which previous findings show the higher polluted condition of indoor air quality in compared with that of outdoor air quality. However, the approaches of investigation were mostly completed through experiment and it required not only time but also cost. This paper aimed at predicting the condition of indoor air quality through artificial neural network. It was carried out by using the historical data from five days measurement consecutively, in which the measurement took eight hours per day by using Yes Plus LGA Meter to capture the existing CO₂. Once the measurement was completely done, then it was used to predict the further condition of indoor air quality through artificial neural network approaches. The results indicate a strong relation between input data and target data used to predict further condition of indoor air quality. Additionally, the use of Levenberg-Method to conduct artificial neural network also demonstrates good correlation between input data and target data. Finally, it can be concluded that beside the preparation of input data, the selection of training method has significant impact on the result of the data prediction

Title : Spatial-temporal Prediction of Air Quality based on Recurrent Neural Networks

Author: Xiaotong Sun, Wei Xu, Hongxun Jiang

Year : 2019

To predict air quality (PM2.5 concentrations, et al), many parametric regression models have been developed, while deep learning algorithms are used less often. And few of them takes the air pollution emission or spatial information into consideration or predict them in hour scale. In this paper, we proposed a spatialtemporal GRU-based prediction framework incorporating ground pollution monitoring (GPM), factory emissions (FE), surface meteorology monitoring (SMM) variables to predict hourly PM2.5 concentrations. The dataset for empirical experiments was built based on air quality monitoring in Shenyang, China. Experimental results indicate that our method enables more accurate predictions than all baseline models and by applying the convolutional processing to the GPM and FE variables notable improvement can be achieved in prediction accuracy In this paper, we propose a spatial-temporal GRU methods which is an extension of RNN in time and space to predict PM2.5. We use the characteristics of RNN to analyze the dependence of PM2.5 concentration time series. We design a convolution processing step to incorporate considerations of spatial information and pollution sources into the prediction process. The relevant experiments were carried out based on the data collected from the 2015-2017 Shenyang Air Pollution Project Monitoring Program. Empirical experiments validate the superiority of our proposed model and the significant improvements that convolutional variables can bring

Title : Ambient Air Pollution Forecasting System using Deep Neural Networks

Author: Geethika Jujjavarapu, Siddhartha Duggirala, Anulekha Kavutarapu, Ravikishan Surapaneni

Year : 2020

Air pollution is a major problem that has been recognized throughout the world. Harmful impacts of air contamination include hypersensitive reactions such as throat irritation, itchy eyes, nose, and some other serious problems. In recent years, the number of fatalities occurred due to air pollution has been increasing dramatically. In this paper, various air pollutants such as Carbon Monoxide, Methane or natural gas, LPG, and air quality at different places of city are measured using sensors. Further, the detected values are then used in the prediction of future values. The evolution of deep neural networks and Internet of Things made this possible to detect and forecast the concentration of pollutants underlying in the air. We use a special module called pyFirmata firmware which is used to connect the Arduino with python and upload the data into csv file on Jupyter Notebook. Here, the data collected is

univariate i.e. it varies with only time. Though there are many statistical models to predict time series datasets such as ARIMA, their efficiency is low. Deep Neural Networks works well for predicting univariate as well as time series datasets. Hence, the Keras sequential model is employed to predict the hourly future values of air pollutants based on previous readings. The final results of prediction are compared with the actual values and error is calculated. As a result, the level of air pollutants at a particular hour can be predicted. The concentration of air pollutants in coming years, month or week helps us to reduce its concentration to lesser than the harmful or toxic range. In this literature, we implemented deep neural network with keras sequential model to predict the air quality, CO, methane, and gas. This model works well for time series datasets. The validation loss of the model for air quality is 3.12, CO is 5.05, methane is 4.31, and gas is 4.80. The dataset used in this model is for a short duration; it limits the model capacity. Hence, use of dataset for longer durations with negligible time gaps is recommended for further improvement. In future, we can employ several climatic factors such as measure of temperature, humidity, dew etc to elevate the accuracy of model

Title : Forecasting And Prediction Of Air Pollution Levels To Protect Human Beings From Health Hazards

Author: S. Suganya, Professor Dr. T. Meyyappan

Year : 2020

Prevention and control of air pollution has become an essential activity in many cities. Air is polluted at unacceptable levels by industries and heavy vehicular traffic in cities which affects human health conditions to a great extent. Forecasting, Predicting and controlling air pollution is the need of the hour to protect human beings from health hazards. Air pollution poses threats not only to humans but also to entire flora and fauna. The prime objective of this paper is to propose a new method to predict air pollution using data collected on monthly basis and provide recommendations to prevent and control air pollution. This research work comprises of two phases. The first phase preprocesses the chosen dataset using python coding. The second phase analyzes the preprocessed data to predict air pollution levels. Kaggle dataset containing monthly air pollution data collected over the period 2000 to 2010 is subjected to the proposed method. Predictions for a future month are made by computing Air Quality Index(AQI) metric and computed threshold value for the previous two months. The proposed method shows acceptable accuracy in performance. Air pollution is dangerous for nature as well as for human beings. Prediction and remedial actions is the need of the hour. In this research work, the data set chosen from Kaggle website is preprocessed first to separate pollutant parameters NO₂, CO, SO₂, O₃. The prediction of air pollution is performed in two phases. The first phase computes AQI (Air Quality Index) values for all the days in a month. The second phase computes threshold value of AQI as an

average of previous months average AQI values. Air pollution for the days in chosen month is predicted by comparing the threshold value with the average of the previous two month values. Big data analytics are used to handle huge data volumes and Python coding is used to implement computational procedures. Prediction accuracy and error rate are computed. The results are found to be encouraging. Further research work is in progress to include other environmental parameters.

Title : Air pollutant severity prediction using Bi-directional LSTM Network

Author: Ishan Verma, Rahul Ahuja, Hardik Meisheri, Lipika Dey

Year : 2018

Air pollution has emerged as a universal concern across the globe affecting human health. This increasing danger motivates the study of systems for predicting air pollutant severities ahead of time. In this paper, we have proposed the use of a bi-directional LSTM model to predict air pollutant severity levels ahead of time. We have shown that the predictions can be significantly improved using an ensemble of three BiDirectional LSTMs (BiLSTM) that model the long-term, shortterm and immediate effects of PM_{2.5} (the key air pollutant) severity levels. Further, weather information data has been taken into account while modelling, since they are found to boost prediction accuracies. Experimental results for multiple locations in New Delhi, India are presented to demonstrate model superiority over earlier techniques. In this paper, we have presented an effective way of predicting the severity of pollutants by leveraging various sensor data in 6,12 and 24 hour in advance using Deep learning models. We have validated this claim over the pollution data from New Delhi, India by predicting the severity of PM_{2.5} pollutant. We present our experimentations with comparison to baseline system over different stations and over different time periods. In addition, we have presented a Ensemble system which performs better in most of the cases, and also proves to be more robust

CHAPTER – 3

EXPERIMENTAL OR MATERIALS AND METHODS, ALGORITHMS USED

3.1 AIR QUALITY INDEX

Globally, many cities continuously assess air quality using monitoring networks designed to measure and record air pollutant concentrations at several points deemed to represent exposure of the population to these pollutants. Current research indicates that guidelines of recommended pollution values cannot be regarded as threshold values below which a zero adverse response may be expected. Therefore, the simplistic comparison of observed values against guidelines may mislead unless suitably quantified. In recent years, air quality information are provided by governments to the public comes in a number of forms like annual reports, environment reviews, and site or subject specific analyses/report. These are generally having available or access to limited audiences and also require time, interest and necessary background to digest its contents. Presently, governments throughout the world have also started to use real-time access to sophisticated database management programs to provide their citizens with access to site-specific air quality index/air pollution index and its probable health consequences.

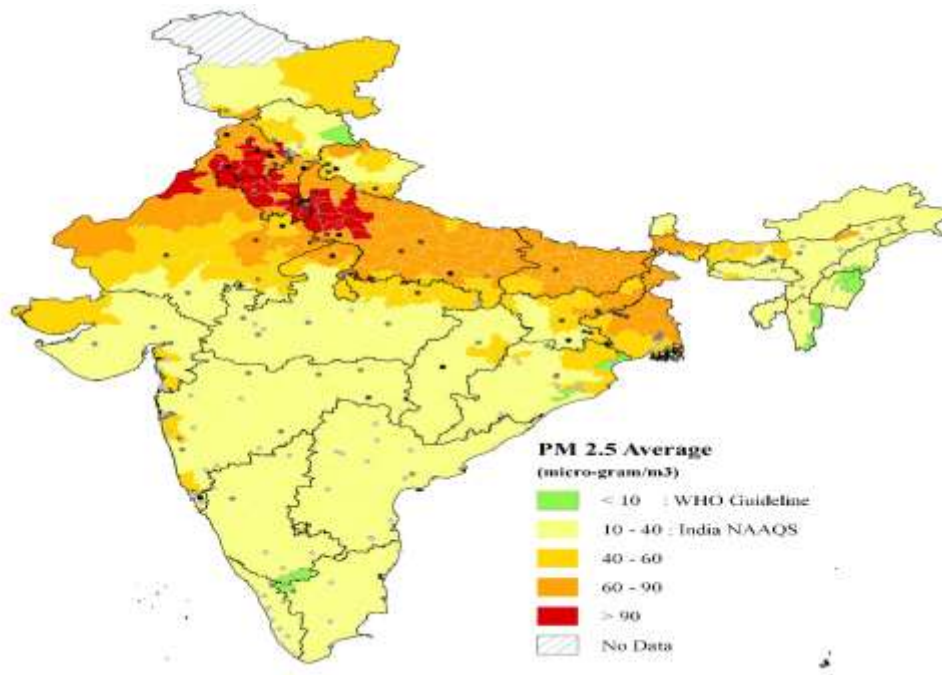


Fig 3.1.1 Average pollutant range in India

Thus, a more sophisticated tool has been developed to communicate the health risk of ambient concentrations using air pollution index (API) or air quality index (AQI). The World Health Organization (WHO) estimates that 25% of all deaths in the developing world can be directly attributed to environmental factors (WHO, 2006). The problem of air pollution and its corresponding adverse health impacts have been aggravated due to increasing industrial and other developmental activities. The monitoring concentrations of pre-determined air pollutants in the residential/commercial/industrial areas are used for the calculation of an air quality index (AQI) or air pollution index (API). The monitoring data are aggregated and converted into a single index with a variety of methods. This means that indexing systems and air pollution descriptors often differ from one country/region to another. The indicators of air quality give the public an opportunity to track the state of their local, regional and national air quality status without the need for an understanding of the details of the monitoring data upon which they are based.

3.2 DATASET:

Pre-processing the given dataset from kaggle - Indian metrological department, past collected dataset and Air quality index. This dataset contains 824 records of features, of following seven pollutants around India,

1. PM2.5
2. PM10
3. NO2
4. NH3
5. SO2
6. CO
7. OZONE

3.3 PROPOSED SYSTEM:

Exploratory Data Analysis of Air Quality Prediction

Multiple datasets from different sources would be combined to form a generalized dataset, and then different machine learning algorithms would be applied to extract patterns and to obtain results with maximum accuracy.

Data Wrangling

In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

Data collection

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using Random Forest, logistic, Decision tree algorithms, K-Nearest Neighbor (KNN) and Support vector classifier (SVC) are applied on the Training set and based on the test result accuracy, Test set prediction is done.

Building the classification model

The predicting the air quality problem, decision tree algorithm prediction model is effective because of the following reasons: It provides better results in classification problem.

It is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical and discrete variables.

It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.

3.4 SYSTEM ARCHITECTURE

This dataset contains 824 records of features, which were then classified into 2 classes: PM2.5, PM10, NO2, NH3, SO2, CO, OZONE. Data wrangling: In this section of the report can load within the information, check for cleanliness, and so trim and clean the given dataset for analysis. Check that that the document steps fastidiously and justify for improvement choices. Data collection: The information set collected for predicting given data is split into coaching set and check set. Generally, 7:3 ratios are applied to separate the coaching set and check set. The information Model that was created victimization Random Forest, logistic, call tree algorithms and Support vector classifier (SVC) is applied on the coaching set and supported the check result accuracy, check set prediction is finished.

Preprocessing:

The data that was collected may contain missing values that will cause inconsistency. To realize higher results information ought to be preprocessed thus on improve the potency of the rule. The outliers have to be compelled to be removed and conjointly variable conversion ought to be done. Here, we have a tendency to be victimization a number of its modules like train_test_split, Decision Tree Classifier or Linear Regression, and accuracy_score. NumPy: It could be a numeric python module that provides quick math functions for calculations. It is employed to browse information in NumPy arrays and for manipulation purposes. Pandas: Used to browse and write completely different files. Data manipulation will be done simply with information frames. Matplotlib: Data visualization could be helpful thanks to facilitating with establishing of the patterns from a given dataset.

In this project, we have chosen to obtain the air pollution data for three specific species, namely Nitrogen Dioxide (NO₂), Ozone (O₃) and PM_{2.5} Particulate Matter, from the Tower Hamlets monitoring station in Central London. The sampling rate was chosen to be hourly which resulted in a total for each pollutant during the chosen time period. The dataset was available to download as a CSV file. The weather data was obtained from the Integrated Surface Global Hourly Dataset hosted by the National Oceanic and Atmospheric Administration (NOAA). The Integrated Surface Dataset is composed of worldwide surface weather observations from over 35,000 stations. Parameters included are atmospheric pressure, temperature/dew point, atmospheric winds, clouds, precipitation and more. ISD provides hourly weather data that is appropriate for the application in this project. The Integrated Surface Dataset offers a data search tool that allows the user to choose the observed species, location and time period. In this project, we have chosen to collect temperature, humidity, wind speed and direction since these are expected to demonstrate a large correlation to air pollution. The dataset was available to download as a CSV file.

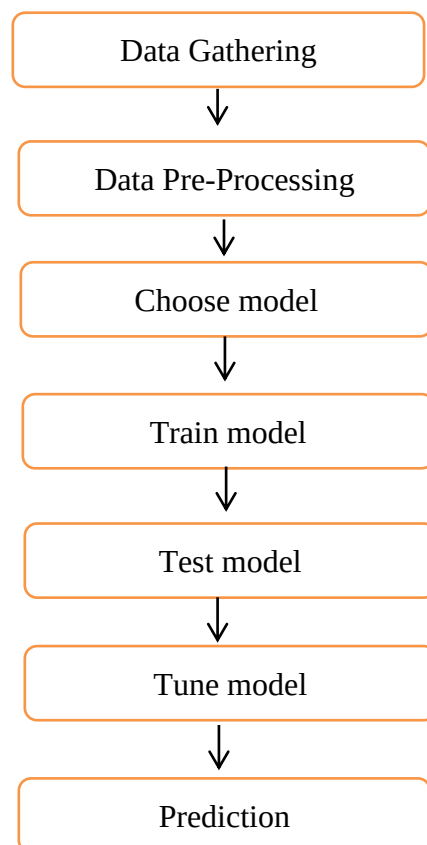


Fig 3.4.1 Process of dataflow diagram

3.5 MODULE IMPLEMENTATION:

3.5.1 DATA PRE-PROCESSING

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

A number of different data cleaning tasks using Python's Pandas library and specifically, it focus on probably the biggest data cleaning task, missing values and it able to more quickly clean data. It wants to spend less time cleaning data, and more time exploring and modeling. Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It's important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to deal with filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical approach for dealing with missing data. Before, joint into code, it's important to understand the sources of missing data.

3.5.2 DATA VALIDATION

Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning model's and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process. The steps and techniques

for data cleaning will vary from dataset to dataset. The primary goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making.

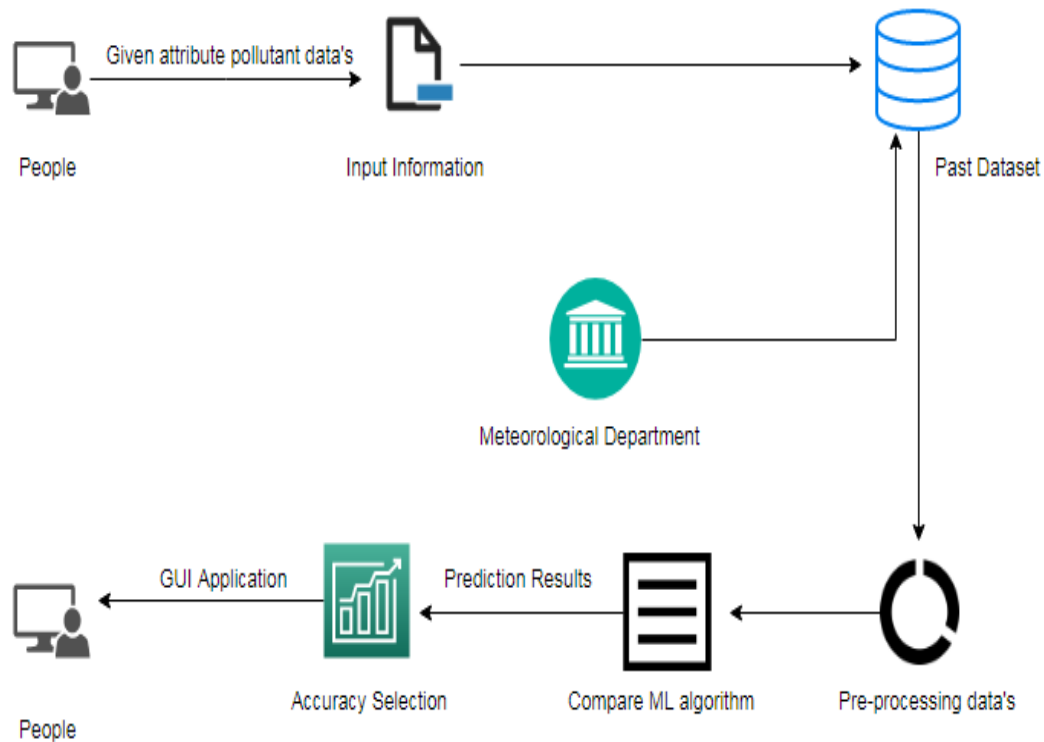


Fig 3.5.2.1 Systematic representation of proposed method

3.5.3 EXPLORATION DATA ANALYSIS OF VISUALIZATION

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end. Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize

of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

3.6 USING SUPERVISED MACHINE LEARNING ALGORITHM:

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc. In Supervised Learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new data based on pattern and associating the patterns to the unlabeled new data.

1. Logistic Regression
2. Random Forest
3. Decision Tree Classifier
4. Naive Bayes

3.6.1 LOGISTIC REGRESSION

It is a statistical method for analysing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).

In other words, the logistic regression model predicts $P(Y=1)$ as a function of X . Logistic regression Assumptions: Binary logistic regression requires the dependent variable to be binary.

For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome. Only the meaningful variables should be included. The independent variables should be

independent of each other. That is, the model should have little. The independent variables are linearly related to the log odds. Logistic regression requires quite large sample sizes.

3.6.2 RANDOM FOREST CLASSIFIER

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set. Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision *trees*, resulting in a *forest of trees*, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks. The following are the basic steps involved in performing the random forest algorithm: Pick N random records from the dataset. Build a decision tree based on these N records. Choose the number of trees you want in your algorithm and repeat steps 1 and 2. In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

3.6.3 DECISION TREE CLASSIFIER

It is one of the most powerful and popular algorithm. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables. Assumptions of Decision tree: At the beginning, we consider the whole training set as the root. Attributes are assumed to be categorical for information gain, attributes are assumed to be continuous. On the basis of attribute values records are distributed recursively. We use statistical methods for ordering attributes as root or internal node. Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. A decision node has two or more branches and a leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data. Decision tree builds classification or regression models in the form of a

tree structure. It utilizes an if-then rule set which is mutually exclusive and exhaustive for classification. The rules are learned sequentially using the training data one at a time. Each time a rule is learned, the tuples covered by the rules are removed. This process is continued on the training set until meeting a termination condition. It is constructed in a top-down recursive divide-and-conquer manner. All the attributes should be categorical. Otherwise, they should be discretized in advance. Attributes in the top of the tree have more impact towards in the classification and they are identified using the information gain concept. A decision tree can be easily over-fitted generating too many branches and may reflect anomalies due to noise or outliers.

3.6.4 NAIVE BAYES ALGORITHM:

The Naive Bayes algorithm is an intuitive method that uses the probabilities of each attribute belonging to each class to make a prediction. It is the supervised learning approach you would come up with if you wanted to model a predictive modeling problem probabilistically. Naive bayes simplifies the calculation of probabilities by assuming that the probability of each attribute belonging to a given class value is independent of all other attributes. This is a strong assumption but results in a fast and effective method. The probability of a class value given a value of an attribute is called the conditional probability. By multiplying the conditional probabilities together for each attribute for a given class value, we have a probability of a data instance belonging to that class. To make a prediction we can calculate probabilities of the instance belonging to each class and select the class value with the highest probability.

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets. Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location. Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence.

3.7 CROSS VALIDATION

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning

algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. Each model will have different performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize. A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

In the next section you will discover exactly how you can do that in Python with scikit-learn. The key to a fair comparison of machine learning algorithms is ensuring that each algorithm is evaluated in the same way on the same data and it can achieve this by forcing each algorithm to be evaluated on a consistent test harness.

The K-fold cross validation procedure is used to evaluate each algorithm, importantly configured with the same random seed to ensure that the same splits to the training data are performed and that each algorithm is evaluated in precisely the same way. Before that comparing algorithm, Building a Machine Learning Model using install Scikit-Learn libraries. In this library package have to done preprocessing, linear model with logistic regression method, cross validating by KFold method, ensemble with random forest method and tree with decision tree classifier. Additionally, splitting the train set and test set. To predicting the result by comparing accuracy.

Prediction result by accuracy:

Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity. It need the output of the algorithm to be classified variable data. Higher accuracy predicting result is logistic regression model by comparing the best accuracy.

True Positive Rate(TPR) = $TP / (TP + FN)$

False Positive rate(FPR) = $FP / (FP + TN)$

Accuracy: The Proportion of the total number of predictions that is correct otherwise overall how often the model predicts correctly defaulters and non-defaulters.

Accuracy calculation:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same.

Precision: The proportion of positive predictions that are actually correct.

$$\text{Precision} = TP / (TP + FP)$$

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

Recall: The proportion of positive observed values correctly predicted. (The proportion of actual defaulters that the model will correctly predict)

$$\text{Recall} = TP / (TP + FN)$$

Recall(Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

General Formula:

$$F\text{- Measure} = 2TP / (2TP + FP + FN)$$

F1-Score Formula:

$$F1 \text{ Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

CHAPTER – 4

RESULTS AND DISCUSSION

4.1RESULTS:

```
Confusion Matrix result of Logistic Regression is:  
[[ 72   1]  
 [  1 174]]  
  
Sensitivity : 0.9863013698630136  
  
Specificity : 0.9942857142857143  
  
Cross validation test results of accuracy:  
[0.99393939 0.99393939 0.98181818 0.97575758 0.99390244]  
  
Accuracy result of Logistic Regression is: 98.78713968957872
```

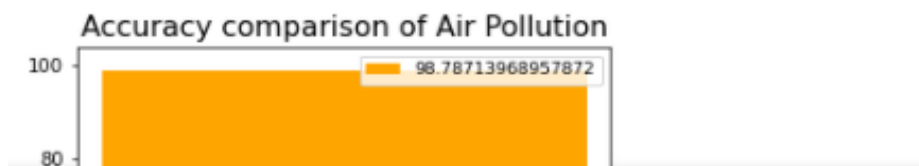


FIG 4.1.1 Result of logistic regression algorithm

Here is the result (fig 4.1.1) using logistic regression algorithm the accuracy we have got here is 98.7% Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable. Logistic regression can suffer from complete separation. This is because the weight for that feature would not converge, because the optimal weight would be infinite. This is really a bit unfortunate, because such a feature is really useful. But you do not need machine learning if you have a simple rule that separates both classes. The problem of complete separation can be solved by introducing penalization of the weights or defining a prior probability distribution of weights. On the good side, the logistic regression model is not only a classification model, but also gives you probabilities. This is a big advantage over models that can only provide the final classification. Knowing that an instance has a 99% probability for a class compared to 51% makes a big difference. Logistic regression can also be extended from binary classification to multi-class classification. Then it is called Multinomial Regression.

```

Confusion Matrix result of Random Forest is:
[[ 72   1]
 [  1 174]]

Sensitivity : 0.9863013698630136

Specificity : 0.9942857142857143

Cross validation test results of accuracy:
[0.99393939 0.98787879 0.99393939 0.98787879 1.         ]

Accuracy result of Random Forest is: 99.27272727272725

```

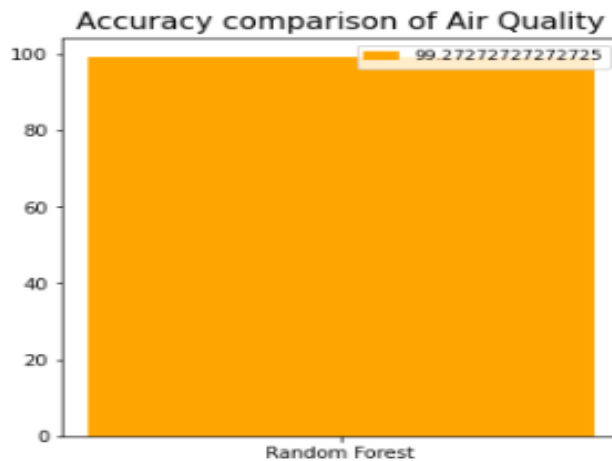


FIG 4.1.2 Result of Random forest algorithm

Here is the result (fig 4.1.2) using Random forest algorithm the accuracy we have got here is 99.2%. But still we are going use naives bayes and decision algorithm to get the best Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems.

```

Classification report of Naive Bayes Results:

```

	precision	recall	f1-score	support
0	0.94	0.99	0.96	73
1	0.99	0.97	0.98	175
accuracy			0.98	248
macro avg	0.96	0.98	0.97	248
weighted avg	0.98	0.98	0.98	248

```

Accuracy result of Naive Bayes is: 97.58064516129032
Confusion Matrix result of Naive Bayes is:
[[ 72  1]
 [ 5 170]]
Sensitivity : 0.9863013698630136
Specificity : 0.9714285714285714
Cross validation test results of accuracy:
[0.99393939 0.96969697 0.96363636 0.96969697 1.
]
Accuracy result of Random Forest is: 97.93939393939392

```

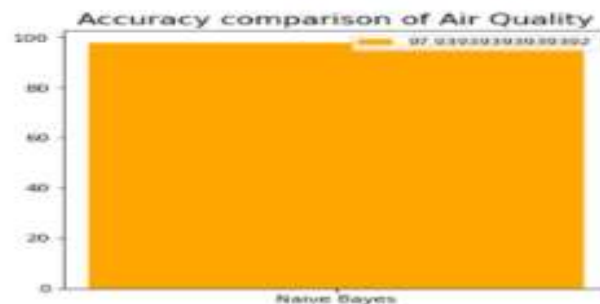


FIG 4.1.3 Result of Naives bayes algorithm

Here is the result (fig 4.1.3) using Naives bayes algorithm the accuracy we have got here is 97.9%. Which is relatively less, when we compared to the before results. This algorithm has more cons than rest of the algorithms. But still it too has many advantages. This the reason we are doing cross validation so that we can cross the result multiple time by using different supervised algorithm to decide a best accuracy as a final result. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

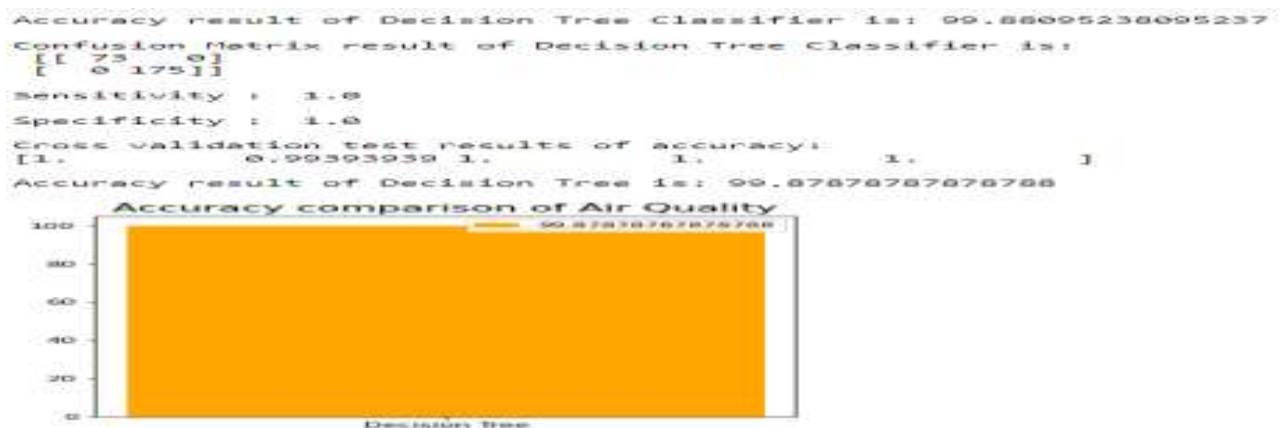


FIG 4.1.4 Result of Decision tree algorithm

Here is the result (fig 4.1.4) using Decision tree algorithm the accuracy we have got here is 99.8%. This is the highest result that we got compared to most of the three algorithms. Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. One of the advantages of decision trees is that their outputs are easy to read and interpret without requiring statistical knowledge. For example, when using decision trees to present demographic information on customers, the marketing department staff can read and interpret the graphical representation of the data without requiring statistical knowledge. The data can also generate important insights on the probabilities, costs, and alternatives to various strategies formulated by the marketing department. Compared to other decision techniques, decision trees take less effort for data preparation. However, users need to have ready information to create new variables with the power to predict the target variable. They can also create classifications of data without having to compute complex calculations. For complex situations, users can combine decision trees with other methods.

4.2 SYSTEM REQUIREMENTS:

PACKAGES:

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.

The two primary data structures of pandas, Series (1- dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, DataFrame provides everything that R's data.frame provides and much more. pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

NumPy is a library for Python, adding support for large, multidimensional arrays and matrices, along with a large collection of highlevel mathematical functions to operate on these arrays. NumPy is an open-source software and has many contributors.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops using NumPy.

In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality.

Scikit-learn (formerly **scikits.learn**) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines (svm), random forest, gradient boosting, k means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. It has been built on top of NumPy, SciPy and matplotlib.

Matplotlib is a Python 2D plotting library which produces publication quality figures in a

variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt or GTK+.

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data. Its plotting functions operate on data frames and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative.

Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. In this, to represent more common values or higher activities brighter colors basically reddish colors are used and to represent less common or activity values, darker colors are preferred. Heatmap is also defined by the name of the shading matrix. Heatmaps in Seaborn can be plotted by using the `seaborn.heatmap()` function.

GUI

Tkinter tutorial provides basic and advanced concepts of Python Tkinter. Our Tkinter tutorial is designed for beginners and professionals. Python provides the standard library Tkinter for creating the graphical user interface for desktop based applications.

Developing desktop based applications with python Tkinter is not a complex task. An empty Tkinter top-level window can be created by using the following steps. Tkinter is a python library for developing GUI (Graphical User Interfaces).

We use the tkinter interface and Tkinter will come with Python as a standard package, it can be used for security purpose of each users or accountants. There will be two kinds of pages like registration user purpose and login entry purpose of library for creating an application of UI (User Interface), to create windows and all other graphical user users.

1. import the Tkinter module.
2. Create the main application window.
3. Add the widgets like labels, buttons, frames, etc. to the window.

4. Call the main event loop so that the actions can take place on the user's computer screen.

Tkinter widgets

There are various widgets like button, canvas, checkbutton, entry, etc. that are used to build the python GUI applications.

CHAPTER – 5

SUMMARY AND FUTURE SCOPE

5.1 SUMMARY

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set is higher accuracy score will be find out. This application can help to find the air quality prediction.

This study combines the pollutant components, meteorological data, and adjacents stations in different time periods into the input variables. After preprocessing data by filling the missing values, encoding, normalizing data and analyzing the correlation between features and PM2.5 concentration as a features selection. Spatial and temporal correlations are complex and comprehensive. In our study, historical data from the target station and adjacent stations are integrated with other features and entered into the model. From the results, the proposed combination is found more effective in extracting spatiotemporal features and performs PM2.5 prediction accuracy more than others.

Through the proposed model, the Spatio-temporal characteristics of the data are extracted. By combining the advantages of the Convolutional Neural Network CNN model, which is effective at filtering out the spatial characteristics include the characteristics of the data between pollutant components and Weather and between different adjacent stations. At the same time, an LSTM network is used for the extraction of temporal features.

Comparing the performances of Four popular deep learning methods in the air pollution prediction problem, we validated the practicality and feasibility of the proposed model in PM2.5 concentration prediction by comparing the Metrics in different batch sizes, and lags. So by using four supervised machine learning algorithms we have got highest result in Decision tree algorithm which 99.8%. Moreover, the results achieved in this work are least comparable to other state-of-the-art deep learning approaches reported in the literature.

5.2 FUTURE SCOPE

The architecture proposed in the system will help in analysing the air quality for the particular surrounding in the real time and it will help the authorities to get the accurate data for the action.

Combining the above model with a real-time API would help in predicting the pollution bands almost in real-time. Further research can be done on Geographic Information System (GIS) on Air Pollution which can help make decisions to geography based on human thinking patterns. It can also organize geographic data, to select and use specific task or project like checking Air quality at location. We can also improve our project by working with Live API and keep predicting the future values automatically.

Realtime data ingestion for air quality:As mentioned in the challenges, the suitable form of collecting data is to obtain from the source itself. When it is obtained from the IoT devices, it is quite common to the data is delivered in the form of API. The IoT devices send data to cloud which in turn provides the API capability. This readily available API service also allows the predictions of AQI in real time.

GIS for air quality:A geographic information system (GIS) is a technological tool for comprehending geography and making intelligent decisions. By understanding geography and people's relationship to location, we can make informed decisions about the way we live on our planet. [9]A good GIS program can process geographic data from a variety of sources and integrate it into a map project. Many countries have an abundance of geographic data for analysis, and governments often make GIS datasets publicly available.

References

- [1] Xi, X., Wei, Z., Xiaoguang, R., Yijie, W., Xinxin, B., Wenjun, Y. and Jin, D., 2015, November. A comprehensive evaluation of air pollution prediction improvement by a machine learning method. In *2015 IEEE international conference on service operations and logistics, and informatics (SOLI)* (pp. 176-181). IEEE.
- [2] Martínez-España, R., Bueno-Crespo, A., Timon-Perez, I.M., Soto, J.A., Ortega, A.M. and Cecilia, J.M., 2018. Air-Pollution Prediction in Smart Cities through Machine Learning Methods: A Case of Study in Murcia, Spain. *J. Univers. Comput. Sci.*, 24(3), pp.261-276. T. Kalaiselvi and K. Somasundaram,
- [3] Delavar, M.R., Gholami, A., Shiran, G.R., Rashidi, Y., Nakhaeizadeh, G.R., Fedra, K. and Hatefi Afshar, S., 2019. A novel method for improving air pollution prediction based on machine learning approaches: a case study applied to the capital city of Tehran. *ISPRS International Journal of Geo-Information*, 8(2), p.99.
- [4] Bellinger, C., Jabbar, M.S.M., Zaïane, O. and Osornio-Vargas, A., 2017. A systematic review of data mining and machine learning for air pollution epidemiology. *BMC public health*, 17(1), pp.1-19.
- [5] Aditya, C. R., Chandana R. Deshmukh, D. K. Nayana, and Praveen Gandhi Vidyavastu. "Detection and prediction of air pollution using machine learning models." In *International Journal of Engineering Trends and Technology (IJETT)*, vol. 59, no. 4, pp. 204-207. 2018
- [6] Sinnott, Richard O., and Ziyue Guan. "Prediction of air pollution through machine learning approaches on the cloud." In *2018 IEEE/ACM 5th International Conference on Big Data Computing Applications and Technologies (BDCAT)*, pp. 51-60. IEEE, 2018.
- [7] Wang, A., Xu, J., Tu, R., Saleh, M. and Hatzopoulou, M., 2020. Potential of machine learning for prediction of traffic related air pollution. *Transportation Research Part D: Transport and Environment*, 88, p.102599.
- [8] Xiao, D., Fang, F., Zheng, J., Pain, C.C. and Navon, I.M., 2019. Machine learning-based rapid response tools for regional air pollution modelling. *Atmospheric environment*, 199, pp.463-473.
- [9] Vidnerová, P. and Neruda, R., 2016, May. Sensor data air pollution prediction by kernel models. In *2016 16th IEEE/ACM International symposium on Cluster, Cloud and Grid Computing (ccGrid)* (pp.666-673). IEEE

- [10] Han, Yang, Jacqueline CK Lam, Victor OK Li, and Qi Zhang. "A Domain-Specific Bayesian Deep-learning Approach for Air Pollution Forecast." *IEEE Transactions on Big Data* (2020).
- [11] Zhao, Bu, Long Yu, Chunyan Wang, Chenyang Shuai, Ji Zhu, Shen Qu, Morteza Taiebat, and Ming Xu. "Urban Air Pollution Mapping Using Fleet Vehicles as Mobile Monitors and Machine Learning." *Environmental Science & Technology* 55, no. 8 (2021): 5579-5588.
- [12] Li, Zhiyuan, Steve Hung-Lam Yim, and Kin-Fai Ho. "High temporal resolution prediction of street-level PM_{2.5} and NO_x concentrations using machine learning approach." *Journal of Cleaner Production* 268 (2020): 121975.
- [13] ul Samee, I., Jilani, M.T. and Wahab, H.G.A., 2019, December. An Application of IoT and Machine Learning to Air Pollution Monitoring in Smart Cities. In *2019 4th International Conference on Emerging Trends in Engineering, Sciences and Technology (ICEEST)* (pp. 1-6). IEEE.
- [14] Mishra, Ayaskanta. "Air pollution monitoring system based on IoT: Forecasting and predictive modeling using machine learning." In *International Conference on Applied Electromagnetics, Signal Processing and Communication (AESPC)*. 2018.
- [15] Shaban, K.B., Kadri, A. and Rezk, E., 2016. Urban air pollution monitoring system with forecasting models. *IEEE Sensors Journal*, 16(8), pp.2598-2606.

APPENDIX

A. SOURCE CODE

Module – 1

Pre-Processing

```
#import library packages  
import pandas as p  
import numpy as n
```

```
import warnings  
warnings.filterwarnings("ignore")
```

```
#Load given dataset  
data = p.read_excel("AirQuality.xlsx")
```

Before drop the given dataset

```
data.head(10)
```

```
#shape  
data.shape
```

After drop the given dataset

```
df=data.dropna()
```

```
df.head(10)
```

```
#shape  
df.shape
```

```
#show columns  
df.columns
```

```
#To describe the dataframe  
df.describe()
```

```
#Checking datatype and information about dataset  
df.info()
```

Checking duplicate values of dataframe

```
#Checking for duplicate data  
df.duplicated()
```

```
#find sum of duplicate data  
sum(df.duplicated())
```

```
#Checking sum of missing values
```

In []:

In []:

In []:

In []:

In []:

In []:

In []:

In []:

In []:

In []:

In []:

In []:

In []:

```
df.isnull().sum()
```

In []:

```
df.Pollutants.unique()
```

To validate avg pollution values

In []:

```
#Checking minimum or maximum of average pollution values
print("Minimum value of average pollution is:", df.Avg.min())
print("Maximum value of average pollution is:", df.Avg.max())
```

In []:

```
#check the average pollution range
print("Avg pollution range :", sorted(df['Avg'].unique()))
```

In []:

```
df.Pollutants.unique()
```

In []:

```
PMkitchen = df.loc[df['Pollutants'] == 'PM2.5']
PMclean = df.loc[df['Pollutants'] == 'PM10']
```

Module – 2

Visualization

```
#import library packages
import pandas as p
import matplotlib.pyplot as plt
import seaborn as s
import numpy as n
```

In []:

```
import warnings
warnings.filterwarnings("ignore")
```

In []:

```
data = p.read_excel("AirQuality.xlsx")
```

In []:

```
df=data.dropna()
```

In []:

```
df.columns
```

In []:

```
p.crosstab(df.city,df.Pollutants)
```

In []:

```
df["Pollutants"].unique()
```

In []:

```
fig, ax = plt.subplots(figsize = (16,10))
ax = s.barplot(x='Pollutants', y='Avg', data=df)
ax.set(ylabel = 'Pollutants details', title = 'Pollutants by average pollution values')
plt.show()
```

In []:

```
#Propagation by variable
```

```
def PropByVar(df, variable):
    dataframe_pie = df[variable].value_counts()
    ax = dataframe_pie.plot.pie(figsize=(10,10), autopct='%1.2f%%', fontsize = 12)
    ax.set_title(variable + ' \n', fontsize = 15)
    return n.round(dataframe_pie/df.shape[0]*100,2)
PropByVar(df, 'Pollutants')
```

In []:

```
#Propagation by variable
def PropByVar(df, variable):
    dataframe_pie = df[variable].value_counts()
    ax = dataframe_pie.plot.pie(figsize=(10,10), autopct='%1.2f%%', fontsize = 12)
    ax.set_title(variable + ' \n', fontsize = 15)
    return n.round(dataframe_pie/df.shape[0]*100,2)
```

```
PropByVar(df, 'city')
```

In []:

```
# Heatmap plot diagram
fig, ax = plt.subplots(figsize=(15,10))
s.heatmap(df.corr(), ax=ax, annot=True)
```

In []:

```
df.boxplot(column="Avg", by="Pollutants")
```

In []:

```
from sklearn.preprocessing import LabelEncoder
var_mod = ['Country', 'State', 'city', 'place', 'lastupdate', 'Avg', 'Max',
'Min','Pollutants']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(str)
```

Module – 3

Logistic Regression :

In []:

```
#import library packages
import pandas as p
import matplotlib.pyplot as plt
import seaborn as s
import numpy as n
```

In []:

```
import warnings
warnings.filterwarnings("ignore")
```

In []:

```
df=p.read_excel("AirQuality.xlsx")
```

In []:

```
df.columns
```

In []:

```
from sklearn.preprocessing import LabelEncoder
var_mod = ['Country','State','city','place','lastupdate','Avg','Max','Min','Pollutants']
le = LabelEncoder()
for i in var_mod:
```

```
df[i] = le.fit_transform(df[i]).astype(str)
```

In []:

```
df['AQI'] = df.Avg.map({ '68':'satisfactor', '74':'satisfactor', '71':'satisfactor',
'4':'good', '39':'good', '42':'good', '27':'good', '263':'poor', '250':'poor',
'249':'poor', '258':'poor', '284':'poor', '28':'good', '106':'satisfactor', '44':'good',
'62':'satisfactor', '59':'satisfactor', '1':'good', '9':'good', '15':'good',
'34':'good', '63':'satisfactor', '14':'good', '13':'good', '69':'satisfactor', '37':'good',
'2':'good', '5':'good', '66':'satisfactor', '202':'poor', '11':'good',
'10':'good', '52':'satisfactor', '54':'satisfactor', '246':'poor', '248':'poor',
'32':'good', '93':'satisfactor', '226':'poor', '189':'moderate', '124':'moderate',
'272':'poor', '277':'poor', '278':'poor', '279':'poor', '280':'poor', '87':'satisfactor',
'19':'good', '240':'poor', '197':'moderate', '38':'good',
'84':'satisfactor', '61':'satisfactor', '196':'moderate', '171':'moderate', '23' : 'good',
'111':'moderate', '43' : 'good', '188':'moderate', '203' : 'poor' , '16' : 'good', '20'
: 'good', '225' : 'poor' , '190':'moderate', '26' : 'good', '104':'moderate', '25' : 'good',
'224' : 'poor' , '210' : 'poor' , '83':'satisfactor', '82':'satisfactor', '233' : 'poor'
, '205' : 'poor' , '78':'satisfactor', '70':'satisfactor', '211' : 'poor' , '179':'moderate',
'91':'satisfactor', '49':'good', '134':'moderate', '161':'moderate', '223' : 'poor'
, '57':'satisfactor', '73':'satisfactor', '207' : 'poor' , '184':'moderate', '48' : 'good',
'6' : 'good', '7' : 'good', '194':'moderate', '21' : 'good', '50' : 'good', '231' : 'poor'
, '175':'moderate', '45' : 'good', '86':'satisfactor', '243' : 'poor' , '232' : 'poor' , '18'
: 'good', '31' : 'good', '95':'satisfactor', '237' : 'poor' , '217' : 'poor'
, '99':'satisfactor', '33' : 'good', '176':'moderate', '22' : 'good', '29' : 'good',
'107':'moderate', '17' : 'good', '75':'satisfactor', '220' : 'poor' ,
'165':'moderate', '149':'moderate', '3' : 'good', '245' : 'poor' , '239' : 'poor' ,
'72':'satisfactor', '77':'satisfactor', '208' : 'poor' , '24' : 'good', '198':'moderate',
'153':'moderate', '53':'satisfactor', '228' : 'poor' , '186':'moderate', '51':'satisfactor',
'247' : 'poor' , '219' : 'poor' , '100':'satisfactor', '36' : 'good', '204' : 'poor' , '30'
: 'good', '221' : 'poor' , '229' : 'poor' , '187':'moderate', '56':'satisfactor', '230' : 'poor'
, '192':'moderate', '114':'moderate', '274' : 'poor' , '236' : 'poor' , '222' : 'poor' , '206'
: 'poor' , '174':'moderate', '58':'satisfactor', '235' : 'poor' , '212' : 'poor' ,
'120':'moderate', '148':'moderate', '242' : 'poor' , '215' : 'poor' , '80':'satisfactor',
'79':'satisfactor', '67':'satisfactor', '40' : 'good', '35' : 'good', '185':'moderate',
'101':'moderate', '46' : 'good', '209' : 'poor' , '169':'moderate', '12' : 'good', '47'
: 'good', '227' : 'poor' , '200' : 'moderate' , '244' : 'poor' , '238' : 'poor' ,
'160':'moderate', '282' : 'poor' , '218' : 'poor' , '261' : 'poor' , '262' : 'poor' , '260'
: 'poor' , '256' : 'poor' , '252' : 'poor' , '162':'moderate', '172':'moderate',
'156':'moderate', '133':'moderate', '116':'moderate', '193':'moderate', '123':'moderate',
'65':'satisfactor', '96':'satisfactor', '173':'moderate', '103':'moderate', '8' : 'good',
'273' : 'poor' , '275' : 'poor' , '283' : 'poor' , '276' : 'poor' , '254' : 'poor' ,
'152':'moderate', '105':'moderate', '64':'satisfactor', '0' : 'good', '76':'satisfactor',
'127':'moderate', '97':'satisfactor', '115':'moderate', '150':'moderate', '142':'moderate',
'41' : 'good', '102':'moderate', '90':'satisfactor', '132':'moderate', '98':'satisfactor',
'199':'moderate', '191':'moderate', '157':'moderate', '139':'moderate', '94':'satisfactor',
'110':'moderate', '270' : 'poor' , '268' : 'poor' , '281' : 'poor' , '265' : 'poor' , '267'
: 'poor' , '269' : 'poor' , '81':'satisfactor', '128':'moderate', '146':'moderate',
'145':'moderate', '129':'moderate', '85':'satisfactor', '117':'moderate', '108':'moderate',
'154':'moderate', '131':'moderate', '255' : 'poor' , '257' : 'poor' , '88':'satisfactor',
'113':'moderate', '118':'moderate', '167':'moderate', '164':'moderate',
'137':'moderate', '135':'moderate', '138':'moderate', '109':'moderate', '60':'satisfactor',
'177':'moderate', '271' : 'poor' , '170':'moderate', '144':'moderate',
'166':'moderate', '155':'moderate', '136':'moderate', '151':'moderate', '89':'satisfactor',
'119':'moderate', '266' : 'poor' , '264' : 'poor' , '251' : 'poor' , '253' : 'poor' , '259'
: 'poor' , '163':'moderate', '159':'moderate', '241' : 'poor' , '214' : 'poor' ,
'122':'moderate', '234' : 'poor' , '195':'moderate', '130':'moderate', '216' : 'poor' , '213'
: 'poor' , '121':'moderate', '112':'moderate', '168':'moderate', '201' : 'poor' ,
'141':'moderate', '181':'moderate', '143':'moderate', '126':'moderate', '182':'moderate',
'178':'moderate', '158':'moderate', '140':'moderate', '183':'moderate', '92':'satisfactor',
```

```

'180':'moderate', '147':'moderate', '55':'satisfactor','125':'moderate'})
In [ ]:
df['class'] = df.AQI.map({'good':1,'moderate':0, 'servere':0, 'verypoor':0,
'satisfactor':1,'poor':0})
In [ ]:
del df['AQI']
In [ ]:
df.head()
In [ ]:
df.columns
In [ ]:
#According to the cross-validated MCC scores, the random forest is the best-performing
model, so now let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef,
cohen_kappa_score, accuracy_score, average_precision_score, roc_auc_score
In [ ]:
X = df.drop(labels='class', axis=1)
#Response variable
y = df.loc[:, 'class']
In [ ]:
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,
stratify=y)
Logistic Regression :
In [ ]:
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score

logR= LogisticRegression()

logR.fit(X_train,y_train)

predictLR = logR.predict(X_test)

print("")
print('Classification report of Logistic Regression Results:')
print("")
print(classification_report(y_test,predictLR))

print("")
cm2=confusion_matrix(y_test,predictLR)
print('Confusion Matrix result of Logistic Regression is:\n',cm2)
print("")
sensitivity2 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity2 )
print("")
specificity2 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity2)
print("")

accuracy = cross_val_score(logR, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)

```

```

#get the mean of each fold
print("")
print("Accuracy result of Logistic Regression is:",accuracy.mean() * 100)
LR=accuracy.mean() * 100

```

Module – 4

```

#import library packages
import pandas as p
import matplotlib.pyplot as plt
import seaborn as s
import numpy as n

```

```

import warnings
warnings.filterwarnings("ignore")

```

```

df=p.read_excel("AirQuality.xlsx")

```

```

from sklearn.preprocessing import LabelEncoder
var_mod = ['Country','State','city','place','lastupdate','Avg','Max','Min','Pollutants']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(str)

```

```

df['AQI'] = df.Avg.map({ '68':'satisfactor', '74':'satisfactor', '71':'satisfactor',
'4':'good', '39':'good', '42':'good', '27':'good', '263':'poor', '250':'poor',
'249':'poor','258':'poor', '284':'poor', '28':'good', '106':'satisfactor', '44':'good',
'62':'satisfactor', '59':'satisfactor', '1':'good', '9':'good', '15':'good',
'34':'good','63':'satisfactor', '14':'good', '13':'good', '69':'satisfactor', '37':'good',
'2':'good', '5':'good', '66':'satisfactor', '202':'poor', '11':'good',
'10':'good','52':'satisfactor', '54':'satisfactor', '246':'poor', '248':'poor',
'32':'good', '93':'satisfactor', '226':'poor', '189':'moderate', '124':'moderate',
'272':'poor','277':'poor', '278':'poor', '279':'poor', '280':'poor', '87':'satisfactor',
'19':'good', '240':'poor', '197':'moderate', '38':'good',
'84':'satisfactor','61':'satisfactor', '196':'moderate', '171':'moderate', '23' : 'good',
'111':'moderate', '43' : 'good', '188':'moderate', '203' : 'poor' , '16' : 'good', '20'
: 'good', '225' : 'poor' , '190':'moderate', '26' : 'good', '104':'moderate', '25' : 'good',
'224' : 'poor' , '210' : 'poor' , '83':'satisfactor', '82':'satisfactor', '233' : 'poor'
, '205' : 'poor' , '78':'satisfactor', '70':'satisfactor', '211' : 'poor' , '179':'moderate',
'91':'satisfactor', '49':'good', '134':'moderate', '161':'moderate', '223' : 'poor'
, '57':'satisfactor', '73':'satisfactor', '207' : 'poor' , '184':'moderate', '48' : 'good',
'6' : 'good', '7' : 'good', '194':'moderate', '21' : 'good', '50' : 'good', '231' : 'poor'
, '175':'moderate', '45' : 'good', '86':'satisfactor', '243' : 'poor' , '232' : 'poor' , '18'
: 'good', '31' : 'good', '95':'satisfactor', '237' : 'poor' , '217' : 'poor'
, '99':'satisfactor', '33' : 'good', '176':'moderate', '22' : 'good', '29' : 'good',
'107':'moderate', '17' : 'good', '75':'satisfactor', '220' : 'poor' ,
'165':'moderate','149':'moderate', '3' : 'good', '245' : 'poor' , '239' : 'poor' ,
'72':'satisfactor', '77':'satisfactor', '208' : 'poor' , '24' : 'good', '198':'moderate',
'153':'moderate','53':'satisfactor', '228' : 'poor' , '186':'moderate', '51':'satisfactor',
'247' : 'poor' , '219' : 'poor' , '100':'satisfactor', '36' : 'good', '204' : 'poor' , '30'

```

```
: 'good', '221' : 'poor' , '229' : 'poor' , '187' : 'moderate', '56' : 'satisfactor', '230' : 'poor'
, '192' : 'moderate', '114' : 'moderate', '274' : 'poor' , '236' : 'poor' , '222' : 'poor' , '206'
: 'poor' , '174' : 'moderate', '58' : 'satisfactor', '235' : 'poor' , '212' : 'poor' ,
'120' : 'moderate', '148' : 'moderate', '242' : 'poor' , '215' : 'poor' , '80' : 'satisfactor',
'79' : 'satisfactor', '67' : 'satisfactor', '40' : 'good', '35' : 'good', '185' : 'moderate',
'101' : 'moderate', '46' : 'good', '209' : 'poor' , '169' : 'moderate', '12' : 'good', '47'
: 'good', '227' : 'poor' , '200' : 'moderate' , '244' : 'poor' , '238' : 'poor' ,
'160' : 'moderate', '282' : 'poor' , '218' : 'poor' , '261' : 'poor' , '262' : 'poor' , '260'
: 'poor' , '256' : 'poor' , '252' : 'poor' , '162' : 'moderate', '172' : 'moderate',
'156' : 'moderate', '133' : 'moderate', '116' : 'moderate', '193' : 'moderate', '123' : 'moderate',
'65' : 'satisfactor', '96' : 'satisfactor', '173' : 'moderate', '103' : 'moderate', '8' : 'good',
'273' : 'poor' , '275' : 'poor' , '283' : 'poor' , '276' : 'poor' , '254' : 'poor' ,
'152' : 'moderate', '105' : 'moderate', '64' : 'satisfactor', '0' : 'good', '76' : 'satisfactor',
'127' : 'moderate', '97' : 'satisfactor', '115' : 'moderate', '150' : 'moderate', '142' : 'moderate',
'41' : 'good', '102' : 'moderate', '90' : 'satisfactor', '132' : 'moderate', '98' : 'satisfactor',
'199' : 'moderate', '191' : 'moderate', '157' : 'moderate', '139' : 'moderate', '94' : 'satisfactor',
'110' : 'moderate', '270' : 'poor' , '268' : 'poor' , '281' : 'poor' , '265' : 'poor' , '267'
: 'poor' , '269' : 'poor' , '81' : 'satisfactor', '128' : 'moderate', '146' : 'moderate',
'145' : 'moderate', '129' : 'moderate', '85' : 'satisfactor', '117' : 'moderate', '108' : 'moderate',
'154' : 'moderate', '131' : 'moderate', '255' : 'poor' , '257' : 'poor' , '88' : 'satisfactor',
'113' : 'moderate', '118' : 'moderate', '167' : 'moderate', '164' : 'moderate',
'137' : 'moderate', '135' : 'moderate', '138' : 'moderate', '109' : 'moderate', '60' : 'satisfactor',
'177' : 'moderate', '271' : 'poor' , '170' : 'moderate', '144' : 'moderate',
'166' : 'moderate', '155' : 'moderate', '136' : 'moderate', '151' : 'moderate', '89' : 'satisfactor',
'119' : 'moderate', '266' : 'poor' , '264' : 'poor' , '251' : 'poor' , '253' : 'poor' , '259'
: 'poor' , '163' : 'moderate', '159' : 'moderate', '241' : 'poor' , '214' : 'poor' ,
'122' : 'moderate', '234' : 'poor' , '195' : 'moderate', '130' : 'moderate', '216' : 'poor' , '213'
: 'poor' , '121' : 'moderate', '112' : 'moderate', '168' : 'moderate', '201' : 'poor' ,
'141' : 'moderate', '181' : 'moderate', '143' : 'moderate', '126' : 'moderate', '182' : 'moderate',
'178' : 'moderate', '158' : 'moderate', '140' : 'moderate', '183' : 'moderate', '92' : 'satisfactor',
'180' : 'moderate', '147' : 'moderate', '55' : 'satisfactor', '125' : 'moderate'})
```

In []:

```
df['class'] = df.AQI.map({'good':1, 'moderate':0, 'servere':0, 'verypoor':0,
'satisfactor':1, 'poor':0})
```

In []:

```
del df['AQI']
```

In []:

```
#According to the cross-validated MCC scores, the random forest is the best-performing
model, so now let's evaluate its performance on the test set.
```

```
from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef,
cohen_kappa_score, accuracy_score, average_precision_score, roc_auc_score
```

In []:

```
X = df.drop(labels='class', axis=1)
#Response variable
y = df.loc[:, 'class']
```

In []:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,
stratify=y)
```

Random Forest:

In []:

```
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
```

```

rfc.fit(X_train,y_train)

predictR = rfc.predict(X_test)

print("")
print('Classification report of Random Forest Results:')
print("")

print(classification_report(y_test,predictR))
accuracy = cross_val_score(rfc, X, y, cv=70)
print('Cross validation test results of accuracy:')
print(accuracy)
print("")

#get the mean of each fold
print("")
print("Accuracy result of Random Forest is:",accuracy.mean() * 100)
print("")

cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Random Forest is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(rfc, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Random Forest is:",accuracy.mean() * 100)
rfc=accuracy.mean() * 100

def graph():
    import matplotlib.pyplot as plt
    data=[rfc]
    alg="Random Forest"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("orange"))
    plt.title("Accuracy comparison o

```

In []:

Module – 5

```

#import library packages
import pandas as p
import matplotlib.pyplot as plt
import seaborn as s
import numpy as n

```

In []:

```
import warnings
warnings.filterwarnings("ignore")
```

In []:

```
df=p.read_excel("AirQuality.xlsx")
```

In []:

```
from sklearn.preprocessing import LabelEncoder
var_mod = ['Country','State','city','place','lastupdate','Avg','Max','Min','Pollutants']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(str)
```

In []:

```
df['AQI'] = df.Avg.map({ '68':'satisfactor', '74':'satisfactor', '71':'satisfactor',
'4':'good', '39':'good', '42':'good', '27':'good', '263':'poor', '250':'poor',
'249':'poor','258':'poor', '284':'poor', '28':'good', '106':'satisfactor', '44':'good',
'62':'satisfactor', '59':'satisfactor', '1':'good', '9':'good', '15':'good',
'34':'good','63':'satisfactor', '14':'good', '13':'good', '69':'satisfactor', '37':'good',
'2':'good', '5':'good', '66':'satisfactor', '202':'poor', '11':'good',
'10':'good','52':'satisfactor', '54':'satisfactor', '246':'poor', '248':'poor',
'32':'good', '93':'satisfactor', '226':'poor', '189':'moderate', '124':'moderate',
'272':'poor','277':'poor', '278':'poor', '279':'poor', '280':'poor', '87':'satisfactor',
'19':'good', '240':'poor', '197':'moderate', '38':'good',
'84':'satisfactor','61':'satisfactor', '196':'moderate', '171':'moderate', '23' : 'good',
'111':'moderate', '43' : 'good', '188':'moderate', '203' : 'poor' , '16' : 'good', '20'
: 'good','225' : 'poor' , '190':'moderate', '26' : 'good', '104':'moderate', '25' : 'good',
'224' : 'poor' , '210' : 'poor' , '83':'satisfactor', '82':'satisfactor', '233' : 'poor'
, '205' : 'poor' , '78':'satisfactor', '70':'satisfactor', '211' : 'poor' , '179':'moderate',
'91':'satisfactor', '49':'good', '134':'moderate', '161':'moderate', '223' : 'poor'
, '57':'satisfactor', '73':'satisfactor', '207' : 'poor' , '184':'moderate', '48' : 'good',
'6' : 'good', '7' : 'good', '194':'moderate', '21' : 'good', '50' : 'good', '231' : 'poor'
, '175':'moderate', '45' : 'good', '86':'satisfactor', '243' : 'poor' , '232' : 'poor' , '18'
: 'good', '31' : 'good', '95':'satisfactor', '237' : 'poor' , '217' : 'poor'
, '99':'satisfactor', '33' : 'good', '176':'moderate', '22' : 'good', '29' : 'good',
'107':'moderate', '17' : 'good', '75':'satisfactor', '220' : 'poor' ,
'165':'moderate','149':'moderate', '3' : 'good', '245' : 'poor' , '239' : 'poor' ,
'72':'satisfactor', '77':'satisfactor', '208' : 'poor' , '24' : 'good', '198':'moderate',
'153':'moderate','53':'satisfactor', '228' : 'poor' , '186':'moderate', '51':'satisfactor',
'247' : 'poor' , '219' : 'poor' , '100':'satisfactor', '36' : 'good', '204' : 'poor' , '30'
: 'good','221' : 'poor' , '229' : 'poor' , '187':'moderate', '56':'satisfactor', '230' : 'poor'
, '192':'moderate', '114':'moderate', '274' : 'poor' , '236' : 'poor' , '222' : 'poor' , '206'
: 'poor' , '174':'moderate', '58':'satisfactor', '235' : 'poor' , '212' : 'poor' ,
'120':'moderate', '148':'moderate', '242' : 'poor' , '215' : 'poor' , '80':'satisfactor',
'79':'satisfactor', '67':'satisfactor', '40' : 'good', '35' : 'good', '185':'moderate',
'101':'moderate', '46' : 'good', '209' : 'poor' , '169':'moderate', '12' : 'good', '47'
: 'good', '227' : 'poor' , '200' : 'moderate' , '244' : 'poor' , '238' : 'poor' ,
'160':'moderate', '282' : 'poor' , '218' : 'poor' , '261' : 'poor' , '262' : 'poor' , '260'
: 'poor' , '256' : 'poor' , '252' : 'poor' , '162':'moderate', '172':'moderate',
'156':'moderate', '133':'moderate','116':'moderate', '193':'moderate', '123':'moderate',
'65':'satisfactor', '96':'satisfactor', '173':'moderate', '103':'moderate', '8' : 'good',
'273' : 'poor' , '275' : 'poor' , '283' : 'poor' , '276' : 'poor' , '254' : 'poor' ,
'152':'moderate', '105':'moderate', '64':'satisfactor', '0' : 'good', '76':'satisfactor',
'127':'moderate', '97':'satisfactor','115':'moderate', '150':'moderate', '142':'moderate',
'41' : 'good', '102':'moderate', '90':'satisfactor', '132':'moderate', '98':'satisfactor',
'199':'moderate', '191':'moderate','157':'moderate', '139':'moderate', '94':'satisfactor',
'110':'moderate', '270' : 'poor' , '268' : 'poor' , '281' : 'poor' , '265' : 'poor' , '267'
: 'poor' , '269' : 'poor' , '81':'satisfactor', '128':'moderate', '146':'moderate',
'145':'moderate', '129':'moderate', '85':'satisfactor', '117':'moderate', '108':'moderate',
'154':'moderate','131':'moderate', '255' : 'poor' , '257' : 'poor' , '88':'satisfactor',
```

```
'113':'moderate', '118':'moderate', '167':'moderate', '164':'moderate',
'137':'moderate', '135':'moderate', '138':'moderate', '109':'moderate', '60':'satisfactor',
'177':'moderate', '271': 'poor' , '170':'moderate', '144':'moderate',
'166':'moderate', '155':'moderate', '136':'moderate', '151':'moderate', '89':'satisfactor',
'119':'moderate', '266': 'poor' , '264': 'poor' , '251': 'poor' , '253': 'poor' , '259':
'poor' , '163':'moderate', '159':'moderate', '241': 'poor' , '214': 'poor' ,
'122':'moderate', '234': 'poor' , '195':'moderate', '130':'moderate', '216': 'poor' , '213':
'poor' , '121':'moderate', '112':'moderate', '168':'moderate', '201': 'poor' ,
'141':'moderate', '181':'moderate', '143':'moderate', '126':'moderate', '182':'moderate',
'178':'moderate', '158':'moderate', '140':'moderate', '183':'moderate', '92':'satisfactor',
'180':'moderate', '147':'moderate', '55':'satisfactor', '125':'moderate'})
```

In []:

```
df['class'] = df.AQI.map({'good':1, 'moderate':0, 'servere':0, 'verypoor':0,
'satisfactor':1, 'poor':0})
```

In []:

```
del df['AQI']
```

In []:

```
#According to the cross-validated MCC scores, the random forest is the best-performing
model, so now let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef,
cohen_kappa_score, accuracy_score, average_precision_score, roc_auc_score
```

In []:

```
X = df.drop(labels='class', axis=1)
#Response variable
y = df.loc[:, 'class']
```

In []:

```
#We'll use a test size of 30%. We also stratify the split on the response variable, which
is very important to do because there are so few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,
stratify=y)
```

Decision Tree Classifier :

In []:

```
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()

dtree.fit(X_train, y_train)

predictDT = dtree.predict(X_test)

print("")
print('Classification report of Decision Tree Classifier Results:')
print("")

print(classification_report(y_test, predictDT))
accuracy = cross_val_score(dtree, X, y, cv=70)

print('Cross validation test results of accuracy:')
print(accuracy)
print("")

#get the mean of each fold
print("")
```

```

print("Accuracy result of Decision Tree Classifier is:",accuracy.mean() * 100)
print("")

cml=confusion_matrix(y_test,predictDT)
print('Confusion Matrix result of Decision Tree Classifier is:\n',cml)
print("")
sensitivity1 = cml[0,0]/(cml[0,0]+cml[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cml[1,1]/(cml[1,0]+cml[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(dtree, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Decision Tree is:",accuracy.mean() * 100)
dtree=accuracy.mean() * 100

```

In []:

Module – 6

```

#import library packages
import pandas as p
import matplotlib.pyplot as plt
import seaborn as s
import numpy as n

```

In []:

```

import warnings
warnings.filterwarnings("ignore")

```

In []:

```
df=p.read_excel("AirQuality.xlsx")
```

In []:

```

from sklearn.preprocessing import LabelEncoder
var_mod = ['Country','State','city','place','lastupdate','Avg','Max','Min','Pollutants']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(str)

```

In []:

```

df['AQI'] = df.Avg.map({ '68':'satisfactor', '74':'satisfactor', '71':'satisfactor',
'4':'good', '39':'good', '42':'good', '27':'good', '263':'poor', '250':'poor',
'249':'poor','258':'poor', '284':'poor', '28':'good', '106':'satisfactor', '44':'good',
'62':'satisfactor', '59':'satisfactor', '1':'good', '9':'good', '15':'good',
'34':'good','63':'satisfactor', '14':'good', '13':'good', '69':'satisfactor', '37':'good',
'2':'good', '5':'good', '66':'satisfactor', '202':'poor', '11':'good',
'10':'good','52':'satisfactor', '54':'satisfactor', '246':'poor', '248':'poor',
'32':'good', '93':'satisfactor', '226':'poor', '189':'moderate', '124':'moderate',
'272':'poor','277':'poor', '278':'poor', '279':'poor', '280':'poor', '87':'satisfactor',
'19':'good', '240':'poor', '197':'moderate', '38':'good',
'84':'satisfactor','61':'satisfactor', '196':'moderate', '171':'moderate', '23' : 'good',
'111':'moderate', '43' : 'good', '188':'moderate', '203' : 'poor' , '16' : 'good', '20'
: 'good','225' : 'poor' , '190':'moderate', '26' : 'good', '104':'moderate', '25' : 'good',
'224' : 'poor' , '210' : 'poor' , '83':'satisfactor', '82':'satisfactor', '233' : 'poor'

```

```
, '205': 'poor', '78': 'satisfactor', '70': 'satisfactor', '211': 'poor', '179': 'moderate',
'91': 'satisfactor', '49': 'good', '134': 'moderate', '161': 'moderate', '223': 'poor',
'57': 'satisfactor', '73': 'satisfactor', '207': 'poor', '184': 'moderate', '48': 'good',
'6': 'good', '7': 'good', '194': 'moderate', '21': 'good', '50': 'good', '231': 'poor',
'175': 'moderate', '45': 'good', '86': 'satisfactor', '243': 'poor', '232': 'poor', '18':
'good', '31': 'good', '95': 'satisfactor', '237': 'poor', '217': 'poor',
'99': 'satisfactor', '33': 'good', '176': 'moderate', '22': 'good', '29': 'good',
'107': 'moderate', '17': 'good', '75': 'satisfactor', '220': 'poor',
'165': 'moderate', '149': 'moderate', '3': 'good', '245': 'poor', '239': 'poor',
'72': 'satisfactor', '77': 'satisfactor', '208': 'poor', '24': 'good', '198': 'moderate',
'153': 'moderate', '53': 'satisfactor', '228': 'poor', '186': 'moderate', '51': 'satisfactor',
'247': 'poor', '219': 'poor', '100': 'satisfactor', '36': 'good', '204': 'poor', '30':
'good', '221': 'poor', '229': 'poor', '187': 'moderate', '56': 'satisfactor', '230': 'poor',
'192': 'moderate', '114': 'moderate', '274': 'poor', '236': 'poor', '222': 'poor', '206':
'poor', '174': 'moderate', '58': 'satisfactor', '235': 'poor', '212': 'poor',
'120': 'moderate', '148': 'moderate', '242': 'poor', '215': 'poor', '80': 'satisfactor',
'79': 'satisfactor', '67': 'satisfactor', '40': 'good', '35': 'good', '185': 'moderate',
'101': 'moderate', '46': 'good', '209': 'poor', '169': 'moderate', '12': 'good', '47':
'good', '227': 'poor', '200': 'moderate', '244': 'poor', '238': 'poor',
'160': 'moderate', '282': 'poor', '218': 'poor', '261': 'poor', '262': 'poor', '260':
'poor', '256': 'poor', '252': 'poor', '162': 'moderate', '172': 'moderate',
'156': 'moderate', '133': 'moderate', '116': 'moderate', '193': 'moderate', '123': 'moderate',
'65': 'satisfactor', '96': 'satisfactor', '173': 'moderate', '103': 'moderate', '8': 'good',
'273': 'poor', '275': 'poor', '283': 'poor', '276': 'poor', '254': 'poor',
'152': 'moderate', '105': 'moderate', '64': 'satisfactor', '0': 'good', '76': 'satisfactor',
'127': 'moderate', '97': 'satisfactor', '115': 'moderate', '150': 'moderate', '142': 'moderate',
'41': 'good', '102': 'moderate', '90': 'satisfactor', '132': 'moderate', '98': 'satisfactor',
'199': 'moderate', '191': 'moderate', '157': 'moderate', '139': 'moderate', '94': 'satisfactor',
'110': 'moderate', '270': 'poor', '268': 'poor', '281': 'poor', '265': 'poor', '267':
'poor', '269': 'poor', '81': 'satisfactor', '128': 'moderate', '146': 'moderate',
'145': 'moderate', '129': 'moderate', '85': 'satisfactor', '117': 'moderate', '108': 'moderate',
'154': 'moderate', '131': 'moderate', '255': 'poor', '257': 'poor', '88': 'satisfactor',
'113': 'moderate', '118': 'moderate', '167': 'moderate', '164': 'moderate',
'137': 'moderate', '135': 'moderate', '138': 'moderate', '109': 'moderate', '60': 'satisfactor',
'177': 'moderate', '271': 'poor', '170': 'moderate', '144': 'moderate',
'166': 'moderate', '155': 'moderate', '136': 'moderate', '151': 'moderate', '89': 'satisfactor',
'119': 'moderate', '266': 'poor', '264': 'poor', '251': 'poor', '253': 'poor', '259':
'poor', '163': 'moderate', '159': 'moderate', '241': 'poor', '214': 'poor',
'122': 'moderate', '234': 'poor', '195': 'moderate', '130': 'moderate', '216': 'poor', '213':
'poor', '121': 'moderate', '112': 'moderate', '168': 'moderate', '201': 'poor',
'141': 'moderate', '181': 'moderate', '143': 'moderate', '126': 'moderate', '182': 'moderate',
'178': 'moderate', '158': 'moderate', '140': 'moderate', '183': 'moderate', '92': 'satisfactor',
'180': 'moderate', '147': 'moderate', '55': 'satisfactor', '125': 'moderate'})
```

In []:

```
df['class'] = df.AQI.map({'good':1, 'moderate':0, 'servere':0, 'verypoor':0,
'satisfactor':1, 'poor':0})
```

In []:

```
del df['AQI']
```

In []:

```
#According to the cross-validated MCC scores, the random forest is the best-performing
model, so now let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef,
cohen_kappa_score, accuracy_score, average_precision_score, roc_auc_score
```

In []:

```
X = df.drop(labels='class', axis=1)
#Response variable
y = df.loc[:, 'class']
```

In []:

```
#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,
stratify=y)
```

Gaussian Naive Bayes :

In []:

```
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()

gnb.fit(X_train,y_train)

predictR = gnb.predict(X_test)

print("")
print('Classification report of Naive Bayes Results:')
print("")

print(classification_report(y_test,predictR))
x = (accuracy_score(y_test,predictR)*100)

print('Accuracy result of Naive Bayes is:', x)
print("")
cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Naive Bayes is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(gnb, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Random Forest is:",accuracy.mean() * 100)
gnb=accuracy.mean() * 100

def graph():
    import matplotlib.pyplot as plt
    data=[gnb]
    alg="Naive Bayes"
    plt.figure(figsize=(5,
```

Module – 7

GUI

```
from tkinter import *
import numpy as np
import pandas as pd
import os
```

```
import warnings
warnings.filterwarnings("ignore")
```

```
data = pd.read_excel("AirQuality.xlsx")
```

```
data.head()
```

```
data.shape
```

```
df=data.dropna()
```

```
df.head()
```

```
df['place'].unique()
```

```
df.shape
```

```
del df["Country"]
del df["place"]
del df["lastupdate"]
```

```
df.columns
```

```
df['Pollutants'].unique()
```

```
df['Source_of_Pollutants'] = df.Pollutants.map({'PM2.5':'Combustion, Fireplaces, Car engines and Coal or Natural gas fired power plants.',
'PM10':'Burning of fuels, Wood stoves, Fires, Wind generated dust, Brick works, Refineries, Cement Works, Iron and Steel making.',
'NO2':'Internal combustion engines burning fossil fuels, Cigarette smoke, Butane, Kerosene heaters and Stoves.',
'NH3':'Emissions is agriculture, Including animal husbandry and NH-based fertilizer applications.',
'SO2':'Burning of fossil fuels by power plants and Other industrial facilities.',
'CO':'Clothes dryers, Water heaters, Furnaces or Boilers, Fireplaces, Both gas and wood burning, Gas stoves and ovens, motor vehicles.',
'OZONE':'Pollutants emitted by cars, Power plants, Industrial boilers, Refineries, Chemical plants, Other sources renet chemically in th presence of sunlight.'})
```

```
list(df['Source_of_Pollutants'])
```

```
df['State'].unique()
```

```
df['Andhra_Pradesh'] = df.State.map({'Andhra_Pradesh':1, 'Bihar':0, 'Delhi':0, 'Gujarat':0,
```

```

'Haryana':0,
    'Jharkhand':0, 'Karnataka':0, 'Kerala':0, 'Madhya Pradesh':0,
    'Maharashtra':0, 'Odisha':0, 'Punjab':0, 'Rajasthan':0, 'TamilNadu':0,
    'Telangana':0, 'Uttar_Pradesh':0, 'West_Bengal':0})
df['Bihar'] = df.State.map({'Andhra_Pradesh':0, 'Bihar':1, 'Delhi':0, 'Gujarat':0,
'Haryana':0,
    'Jharkhand':0, 'Karnataka':0, 'Kerala':0, 'Madhya Pradesh':0,
    'Maharashtra':0, 'Odisha':0, 'Punjab':0, 'Rajasthan':0, 'TamilNadu':0,
    'Telangana':0, 'Uttar_Pradesh':0, 'West_Bengal':0})
df['Delhi'] = df.State.map({'Andhra_Pradesh':0, 'Bihar':0, 'Delhi':1, 'Gujarat':0,
'Haryana':0,
    'Jharkhand':0, 'Karnataka':0, 'Kerala':0, 'Madhya Pradesh':0,
    'Maharashtra':0, 'Odisha':0, 'Punjab':0, 'Rajasthan':0, 'TamilNadu':0,
    'Telangana':0, 'Uttar_Pradesh':0, 'West_Bengal':0})

```

In []:

```

df['Gujarat'] = df.State.map({'Andhra_Pradesh':0, 'Bihar':0, 'Delhi':0, 'Gujarat':1,
'Haryana':0,
    'Jharkhand':0, 'Karnataka':0, 'Kerala':0, 'Madhya Pradesh':0,
    'Maharashtra':0, 'Odisha':0, 'Punjab':0, 'Rajasthan':0, 'TamilNadu':0,
    'Telangana':0, 'Uttar_Pradesh':0, 'West_Bengal':0})
df['Haryana'] = df.State.map({'Andhra_Pradesh':0, 'Bihar':0, 'Delhi':0, 'Gujarat':0,
'Haryana':1,
    'Jharkhand':0, 'Karnataka':0, 'Kerala':0, 'Madhya Pradesh':0,
    'Maharashtra':0, 'Odisha':0, 'Punjab':0, 'Rajasthan':0, 'TamilNadu':0,
    'Telangana':0, 'Uttar_Pradesh':0, 'West_Bengal':0})
df['Jharkhand'] = df.State.map({'Andhra_Pradesh':0, 'Bihar':0, 'Delhi':0, 'Gujarat':0,
'Haryana':0,
    'Jharkhand':1, 'Karnataka':0, 'Kerala':0, 'Madhya Pradesh':0,
    'Maharashtra':0, 'Odisha':0, 'Punjab':0, 'Rajasthan':0, 'TamilNadu':0,
    'Telangana':0, 'Uttar_Pradesh':0, 'West_Bengal':0})

```

In []:

```

df['Karnataka'] = df.State.map({'Andhra_Pradesh':0, 'Bihar':0, 'Delhi':0, 'Gujarat':0,
'Haryana':0,
    'Jharkhand':0, 'Karnataka':1, 'Kerala':0, 'Madhya Pradesh':0,
    'Maharashtra':0, 'Odisha':0, 'Punjab':0, 'Rajasthan':0, 'TamilNadu':0,
    'Telangana':0, 'Uttar_Pradesh':0, 'West_Bengal':0})
df['Kerala'] = df.State.map({'Andhra_Pradesh':0, 'Bihar':0, 'Delhi':0, 'Gujarat':0,
'Haryana':0,
    'Jharkhand':0, 'Karnataka':0, 'Kerala':1, 'Madhya Pradesh':0,
    'Maharashtra':0, 'Odisha':0, 'Punjab':0, 'Rajasthan':0, 'TamilNadu':0,
    'Telangana':0, 'Uttar_Pradesh':0, 'West_Bengal':0})
df['Madhya_Pradesh'] = df.State.map({'Andhra_Pradesh':0, 'Bihar':0, 'Delhi':0, 'Gujarat':0,
'Haryana':0,
    'Jharkhand':0, 'Karnataka':0, 'Kerala':0, 'Madhya Pradesh':1,
    'Maharashtra':0, 'Odisha':0, 'Punjab':0, 'Rajasthan':0, 'TamilNadu':0,
    'Telangana':0, 'Uttar_Pradesh':0, 'West_Bengal':0})

```

In []:

```

df['Maharashtra'] = df.State.map({'Andhra_Pradesh':0, 'Bihar':0, 'Delhi':0, 'Gujarat':0,
'Haryana':0,
    'Jharkhand':0, 'Karnataka':0, 'Kerala':0, 'Madhya Pradesh':0,
    'Maharashtra':1, 'Odisha':0, 'Punjab':0, 'Rajasthan':0, 'TamilNadu':0,
    'Telangana':0, 'Uttar_Pradesh':0, 'West_Bengal':0})
df['Odisha'] = df.State.map({'Andhra_Pradesh':0, 'Bihar':0, 'Delhi':0, 'Gujarat':0,
'Haryana':0,
    'Jharkhand':0, 'Karnataka':0, 'Kerala':0, 'Madhya Pradesh':0,
    'Maharashtra':0, 'Odisha':1, 'Punjab':0, 'Rajasthan':0, 'TamilNadu':0,
    'Telangana':0, 'Uttar_Pradesh':0, 'West_Bengal':0})

```

```

df['Punjab'] = df.State.map({'Andhra_Pradesh':0, 'Bihar':0, 'Delhi':0, 'Gujarat':0,
'Haryana':0,
'Jharkhand':0, 'Karnataka':0, 'Kerala':0, 'Madhya Pradesh':0,
'Maharashtra':0, 'Odisha':0, 'Punjab':0, 'Rajasthan':1, 'TamilNadu':0,
'Telangana':0, 'Uttar_Pradesh':0, 'West_Bengal':0})

In [ ]:
df['Rajasthan'] = df.State.map({'Andhra_Pradesh':0, 'Bihar':0, 'Delhi':0, 'Gujarat':0,
'Haryana':0,
'Jharkhand':0, 'Karnataka':0, 'Kerala':0, 'Madhya Pradesh':0,
'Maharashtra':0, 'Odisha':0, 'Punjab':0, 'Rajasthan':0, 'TamilNadu':0,
'Telangana':0, 'Uttar_Pradesh':0, 'West_Bengal':0})
df['TamilNadu'] = df.State.map({'Andhra_Pradesh':0, 'Bihar':0, 'Delhi':0, 'Gujarat':0,
'Haryana':0,
'Jharkhand':0, 'Karnataka':0, 'Kerala':0, 'Madhya Pradesh':0,
'Maharashtra':0, 'Odisha':0, 'Punjab':0, 'Rajasthan':0, 'TamilNadu':1,
'Telangana':0, 'Uttar_Pradesh':0, 'West_Bengal':0})
df['Telangana'] = df.State.map({'Andhra_Pradesh':0, 'Bihar':0, 'Delhi':0, 'Gujarat':0,
'Haryana':0,
'Jharkhand':0, 'Karnataka':0, 'Kerala':0, 'Madhya Pradesh':0,
'Maharashtra':0, 'Odisha':0, 'Punjab':0, 'Rajasthan':0, 'TamilNadu':0,
'Telangana':1, 'Uttar_Pradesh':0, 'West_Bengal':0})

In [ ]:
df['Uttar_Pradesh'] = df.State.map({'Andhra_Pradesh':0, 'Bihar':0, 'Delhi':0, 'Gujarat':0,
'Haryana':0,
'Jharkhand':0, 'Karnataka':0, 'Kerala':0, 'Madhya Pradesh':0,
'Maharashtra':0, 'Odisha':0, 'Punjab':0, 'Rajasthan':0, 'TamilNadu':0,
'Telangana':0, 'Uttar_Pradesh':1, 'West_Bengal':0})
df['West_Bengal'] = df.State.map({'Andhra_Pradesh':0, 'Bihar':0, 'Delhi':0, 'Gujarat':0,
'Haryana':0,
'Jharkhand':0, 'Karnataka':0, 'Kerala':0, 'Madhya Pradesh':0,
'Maharashtra':0, 'Odisha':0, 'Punjab':0, 'Rajasthan':0, 'TamilNadu':0,
'Telangana':0, 'Uttar_Pradesh':0, 'West_Bengal':1})

In [ ]:
df['city'].unique()

In [ ]:
df['Amaravati'] = df.city.map({'Amaravati':1, 'Rajamahendravaram':0, 'Tirupati':0,
'Vijayawada':0,
'Visakhapatnam':0, 'Gaya':0, 'Muzaffarpur':0, 'Patna':0, 'Delhi':0,
'Ahmedabad':0, 'Faridabad':0, 'Gurugram':0, 'Manesar':0, 'Panchkula':0,
'Rohtak':0, 'Jorapokhar':0, 'Bengaluru':0, 'Chikkaballapur':0, 'Hubballi':0,
'Thiruvananthapuram':0, 'Dewas':0, 'Mandideep':0, 'Pithampur':0, 'Satna':0,
'Singrauli':0, 'Ujjain':0, 'Aurangabad':0, 'Chandrapur':0, 'Mumbai':0,
'Nagpur':0, 'Nashik':0, 'Pune':0, 'Solapur':0, 'Thane':0, 'Brajrajnagar':0,
'Talcher':0, 'Amritsar':0, 'Bathinda':0, 'Jalandhar':0, 'Khanna':0,
'Ludhiana':0, 'Mandi Gobindgarh':0, 'Patiala':0, 'Rupnagar':0, 'Ajmer':0,
'Alwar':0, 'Bhiwadi':0, 'Jaipur':0, 'Jodhpur':0, 'Kota':0, 'Pali':0, 'Udaipur':0,
'Chennai':0, 'Hyderabad':0, 'Agra':0, 'Baghpat':0, 'Bulandshahr':0,
'Ghaziabad':0, 'Greater Noida':0, 'Kanpur':0, 'Lucknow':0, 'Moradabad':0,
'Muzaffarnagar':0, 'Noida':0, 'Varanasi':0, 'Asanol':0, 'Durgapur':0,
'Haldia':0, 'Howrah':0, 'Kolkata':0, 'Siliguri':0})

In [ ]:
df['Rajamahendravaram'] = df.city.map({'Amaravati':0, 'Rajamahendravaram':1, 'Tirupati':0,
'Vijayawada':0,
'Visakhapatnam':0, 'Gaya':0, 'Muzaffarpur':0, 'Patna':0, 'Delhi':0,
'Ahmedabad':0, 'Faridabad':0, 'Gurugram':0, 'Manesar':0, 'Panchkula':0,
'Rohtak':0, 'Jorapokhar':0, 'Bengaluru':0, 'Chikkaballapur':0, 'Hubballi':0,

```

```
'Thiruvananthapuram':0, 'Dewas':0, 'Mandideep':0, 'Pithampur':0, 'Satna':0,
'Singrauli':0, 'Ujjain':0, 'Aurangabad':0, 'Chandrapur':0, 'Mumbai':0,
'Nagpur':0, 'Nashik':0, 'Pune':0, 'Solapur':0, 'Thane':0, 'Brajrajnagar':0,
'Talcher':0, 'Amritsar':0, 'Bathinda':0, 'Jalandhar':0, 'Khanna':0,
'Ludhiana':0, 'Mandi Gobindgarh':0, 'Patiala':0, 'Rupnagar':0, 'Ajmer':0,
'Alwar':0, 'Bhiwadi':0, 'Jaipur':0, 'Jodhpur':0, 'Kota':0, 'Pali':0, 'Udaipur':0,
'Chennai':0, 'Hyderabad':0, 'Agra':0, 'Baghpat':0, 'Bulandshahr':0,
'Ghaziabad':0, 'Greater_Noida':0, 'Kanpur':0, 'Lucknow':0, 'Moradabad':0,
'Muzaffarnagar':0, 'Noida':0, 'Varanasi':0, 'Asanol':0, 'Durgapur':0,
'Haldia':0, 'Howrah':0, 'Kolkata':0, 'Siliguri':0}}
```

In []:

```
df['Tirupati'] = df.city.map({'Amaravati':0, 'Rajamahendravaram':0, 'Tirupati':1,
'Vijayawada':0,
'Visakhapatnam':0, 'Gaya':0, 'Muzaffarpur':0, 'Patna':0, 'Delhi':0,
'A Ahmedabad':0, 'Faridabad':0, 'Gurugram':0, 'Manesar':0, 'Panchkula':0,
'Rohtak':0, 'Jorapokhar':0, 'Bengaluru':0, 'Chikkaballapur':0, 'Hubballi':0,
'Thiruvananthapuram':0, 'Dewas':0, 'Mandideep':0, 'Pithampur':0, 'Satna':0,
'Singrauli':0, 'Ujjain':0, 'Aurangabad':0, 'Chandrapur':0, 'Mumbai':0,
'Nagpur':0, 'Nashik':0, 'Pune':0, 'Solapur':0, 'Thane':0, 'Brajrajnagar':0,
'Talcher':0, 'Amritsar':0, 'Bathinda':0, 'Jalandhar':0, 'Khanna':0,
'Ludhiana':0, 'Mandi Gobindgarh':0, 'Patiala':0, 'Rupnagar':0, 'Ajmer':0,
'Alwar':0, 'Bhiwadi':0, 'Jaipur':0, 'Jodhpur':0, 'Kota':0, 'Pali':0, 'Udaipur':0,
'Chennai':0, 'Hyderabad':0, 'Agra':0, 'Baghpat':0, 'Bulandshahr':0,
'Ghaziabad':0, 'Greater_Noida':0, 'Kanpur':0, 'Lucknow':0, 'Moradabad':0,
'Muzaffarnagar':0, 'Noida':0, 'Varanasi':0, 'Asanol':0, 'Durgapur':0,
'Haldia':0, 'Howrah':0, 'Kolkata':0, 'Siliguri':0})
```

In []:

```
df['Vijayawada'] = df.city.map({'Amaravati':0, 'Rajamahendravaram':0, 'Tirupati':0,
'Vijayawada':1,
'Visakhapatnam':0, 'Gaya':0, 'Muzaffarpur':0, 'Patna':0, 'Delhi':0,
'A Ahmedabad':0, 'Faridabad':0, 'Gurugram':0, 'Manesar':0, 'Panchkula':0,
'Rohtak':0, 'Jorapokhar':0, 'Bengaluru':0, 'Chikkaballapur':0, 'Hubballi':0,
'Thiruvananthapuram':0, 'Dewas':0, 'Mandideep':0, 'Pithampur':0, 'Satna':0,
'Singrauli':0, 'Ujjain':0, 'Aurangabad':0, 'Chandrapur':0, 'Mumbai':0,
'Nagpur':0, 'Nashik':0, 'Pune':0, 'Solapur':0, 'Thane':0, 'Brajrajnagar':0,
'Talcher':0, 'Amritsar':0, 'Bathinda':0, 'Jalandhar':0, 'Khanna':0,
'Ludhiana':0, 'Mandi Gobindgarh':0, 'Patiala':0, 'Rupnagar':0, 'Ajmer':0,
'Alwar':0, 'Bhiwadi':0, 'Jaipur':0, 'Jodhpur':0, 'Kota':0, 'Pali':0, 'Udaipur':0,
'Chennai':0, 'Hyderabad':0, 'Agra':0, 'Baghpat':0, 'Bulandshahr':0,
'Ghaziabad':0, 'Greater_Noida':0, 'Kanpur':0, 'Lucknow':0, 'Moradabad':0,
'Muzaffarnagar':0, 'Noida':0, 'Varanasi':0, 'Asanol':0, 'Durgapur':0,
'Haldia':0, 'Howrah':0, 'Kolkata':0, 'Siliguri':0})
```

In []:

```
df['Visakhapatnam'] = df.city.map({'Amaravati':0, 'Rajamahendravaram':0, 'Tirupati':0,
'Vijayawada':0,
'Visakhapatnam':1, 'Gaya':0, 'Muzaffarpur':0, 'Patna':0, 'Delhi':0,
'A Ahmedabad':0, 'Faridabad':0, 'Gurugram':0, 'Manesar':0, 'Panchkula':0,
'Rohtak':0, 'Jorapokhar':0, 'Bengaluru':0, 'Chikkaballapur':0, 'Hubballi':0,
'Thiruvananthapuram':0, 'Dewas':0, 'Mandideep':0, 'Pithampur':0, 'Satna':0,
'Singrauli':0, 'Ujjain':0, 'Aurangabad':0, 'Chandrapur':0, 'Mumbai':0,
'Nagpur':0, 'Nashik':0, 'Pune':0, 'Solapur':0, 'Thane':0, 'Brajrajnagar':0,
'Talcher':0, 'Amritsar':0, 'Bathinda':0, 'Jalandhar':0, 'Khanna':0,
'Ludhiana':0, 'Mandi Gobindgarh':0, 'Patiala':0, 'Rupnagar':0, 'Ajmer':0,
'Alwar':0, 'Bhiwadi':0, 'Jaipur':0, 'Jodhpur':0, 'Kota':0, 'Pali':0, 'Udaipur':0,
'Chennai':0, 'Hyderabad':0, 'Agra':0, 'Baghpat':0, 'Bulandshahr':0,
'Ghaziabad':0, 'Greater_Noida':0, 'Kanpur':0, 'Lucknow':0, 'Moradabad':0,
'Muzaffarnagar':0, 'Noida':0, 'Varanasi':0, 'Asanol':0, 'Durgapur':0,
```

```
'Haldia':0, 'Howrah':0, 'Kolkata':0, 'Siliguri':0})
```

In []:

```
df['Gaya'] = df.city.map({'Amaravati':0, 'Rajamahendravaram':0, 'Tirupati':0,
'Vijayawada':0,
'Visakhapatnam':0, 'Gaya':1, 'Muzaffarpur':0, 'Patna':0, 'Delhi':0,
'Ahmedabad':0, 'Faridabad':0, 'Gurugram':0, 'Manesar':0, 'Panchkula':0,
'Rohtak':0, 'Jorapokhar':0, 'Bengaluru':0, 'Chikkaballapur':0, 'Hubballi':0,
'Thiruvananthapuram':0, 'Dewas':0, 'Mandideep':0, 'Pithampur':0, 'Satna':0,
'Singrauli':0, 'Ujjain':0, 'Aurangabad':0, 'Chandrapur':0, 'Mumbai':0,
'Nagpur':0, 'Nashik':0, 'Pune':0, 'Solapur':0, 'Thane':0, 'Brajrajnagar':0,
'Talcher':0, 'Amritsar':0, 'Bathinda':0, 'Jalandhar':0, 'Khanna':0,
'Ludhiana':0, 'Mandi Gobindgarh':0, 'Patiala':0, 'Rupnagar':0, 'Ajmer':0,
'Alwar':0, 'Bhiwadi':0, 'Jaipur':0, 'Jodhpur':0, 'Kota':0, 'Pali':0, 'Udaipur':0,
'Chennai':0, 'Hyderabad':0, 'Agra':0, 'Baghpat':0, 'Bulandshahr':0,
'Ghaziabad':0, 'Greater_Noida':0, 'Kanpur':0, 'Lucknow':0, 'Moradabad':0,
'Muzaffarnagar':0, 'Noida':0, 'Varanasi':0, 'Asanol':0, 'Durgapur':0,
'Haldia':0, 'Howrah':0, 'Kolkata':0, 'Siliguri':0})
```

In []:

```
df['Muzaffarpur'] = df.city.map({'Amaravati':0, 'Rajamahendravaram':0, 'Tirupati':0,
'Vijayawada':0,
'Visakhapatnam':0, 'Gaya':0, 'Muzaffarpur':1, 'Patna':0, 'Delhi':0,
'Ahmedabad':0, 'Faridabad':0, 'Gurugram':0, 'Manesar':0, 'Panchkula':0,
'Rohtak':0, 'Jorapokhar':0, 'Bengaluru':0, 'Chikkaballapur':0, 'Hubballi':0,
'Thiruvananthapuram':0, 'Dewas':0, 'Mandideep':0, 'Pithampur':0, 'Satna':0,
'Singrauli':0, 'Ujjain':0, 'Aurangabad':0, 'Chandrapur':0, 'Mumbai':0,
'Nagpur':0, 'Nashik':0, 'Pune':0, 'Solapur':0, 'Thane':0, 'Brajrajnagar':0,
'Talcher':0, 'Amritsar':0, 'Bathinda':0, 'Jalandhar':0, 'Khanna':0,
'Ludhiana':0, 'Mandi Gobindgarh':0, 'Patiala':0, 'Rupnagar':0, 'Ajmer':0,
'Alwar':0, 'Bhiwadi':0, 'Jaipur':0, 'Jodhpur':0, 'Kota':0, 'Pali':0, 'Udaipur':0,
'Chennai':0, 'Hyderabad':0, 'Agra':0, 'Baghpat':0, 'Bulandshahr':0,
'Ghaziabad':0, 'Greater_Noida':0, 'Kanpur':0, 'Lucknow':0, 'Moradabad':0,
'Muzaffarnagar':0, 'Noida':0, 'Varanasi':0, 'Asanol':0, 'Durgapur':0,
'Haldia':0, 'Howrah':0, 'Kolkata':0, 'Siliguri':0})
```

In []:

```
df['Patna'] = df.city.map({'Amaravati':0, 'Rajamahendravaram':0, 'Tirupati':0,
'Vijayawada':0,
'Visakhapatnam':0, 'Gaya':0, 'Muzaffarpur':0, 'Patna':1, 'Delhi':0,
'Ahmedabad':0, 'Faridabad':0, 'Gurugram':0, 'Manesar':0, 'Panchkula':0,
'Rohtak':0, 'Jorapokhar':0, 'Bengaluru':0, 'Chikkaballapur':0, 'Hubballi':0,
'Thiruvananthapuram':0, 'Dewas':0, 'Mandideep':0, 'Pithampur':0, 'Satna':0,
'Singrauli':0, 'Ujjain':0, 'Aurangabad':0, 'Chandrapur':0, 'Mumbai':0,
'Nagpur':0, 'Nashik':0, 'Pune':0, 'Solapur':0, 'Thane':0, 'Brajrajnagar':0,
'Talcher':0, 'Amritsar':0, 'Bathinda':0, 'Jalandhar':0, 'Khanna':0,
'Ludhiana':0, 'Mandi Gobindgarh':0, 'Patiala':0, 'Rupnagar':0, 'Ajmer':0,
'Alwar':0, 'Bhiwadi':0, 'Jaipur':0, 'Jodhpur':0, 'Kota':0, 'Pali':0, 'Udaipur':0,
'Chennai':0, 'Hyderabad':0, 'Agra':0, 'Baghpat':0, 'Bulandshahr':0,
'Ghaziabad':0, 'Greater_Noida':0, 'Kanpur':0, 'Lucknow':0, 'Moradabad':0,
'Muzaffarnagar':0, 'Noida':0, 'Varanasi':0, 'Asanol':0, 'Durgapur':0,
'Haldia':0, 'Howrah':0, 'Kolkata':0, 'Siliguri':0})
```

In []:

```
df['Delhi'] = df.city.map({'Amaravati':0, 'Rajamahendravaram':0, 'Tirupati':0,
'Vijayawada':0,
'Visakhapatnam':0, 'Gaya':0, 'Muzaffarpur':0, 'Patna':0, 'Delhi':1,
'Ahmedabad':0, 'Faridabad':0, 'Gurugram':0, 'Manesar':0, 'Panchkula':0,
'Rohtak':0, 'Jorapokhar':0, 'Bengaluru':0, 'Chikkaballapur':0, 'Hubballi':0,
'Thiruvananthapuram':0, 'Dewas':0, 'Mandideep':0, 'Pithampur':0, 'Satna':0,
'Singrauli':0, 'Ujjain':0, 'Aurangabad':0, 'Chandrapur':0, 'Mumbai':0,
```

```
'Nagpur':0, 'Nashik':0, 'Pune':0, 'Solapur':0, 'Thane':0, 'Brajrajnagar':0,
'Talcher':0, 'Amritsar':0, 'Bathinda':0, 'Jalandhar':0, 'Khanna':0,
'Ludhiana':0, 'Mandi Gobindgarh':0, 'Patiala':0, 'Rupnagar':0, 'Ajmer':0,
'Alwar':0, 'Bhiwadi':0, 'Jaipur':0, 'Jodhpur':0, 'Kota':0, 'Pali':0, 'Udaipur':0,
'Chennai':0, 'Hyderabad':0, 'Agra':0, 'Baghpat':0, 'Bulandshahr':0,
'Ghaziabad':0, 'Greater_Noida':0, 'Kanpur':0, 'Lucknow':0, 'Moradabad':0,
'Muzaffarnagar':0, 'Noida':0, 'Varanasi':0, 'Asanol':0, 'Durgapur':0,
'Haldia':0, 'Howrah':0, 'Kolkata':0, 'Siliguri':0}}
```

In []:

```
df['Ahmedabad'] = df.city.map({'Amaravati':0, 'Rajamahendravaram':0, 'Tirupati':0,
'Vijayawada':0,
'Visakhapatnam':0, 'Gaya':0, 'Muzaffarpur':0, 'Patna':0, 'Delhi':0,
'Ahmedabad':1, 'Faridabad':0, 'Gurugram':0, 'Manesar':0, 'Panchkula':0,
'Rohtak':0, 'Jorapokhar':0, 'Bengaluru':0, 'Chikkaballapur':0, 'Hubballi':0,
'Thiruvananthapuram':0, 'Dewas':0, 'Mandideep':0, 'Pithampur':0, 'Satna':0,
'Singrauli':0, 'Ujjain':0, 'Aurangabad':0, 'Chandrapur':0, 'Mumbai':0,
'Nagpur':0, 'Nashik':0, 'Pune':0, 'Solapur':0, 'Thane':0, 'Brajrajnagar':0,
'Talcher':0, 'Amritsar':0, 'Bathinda':0, 'Jalandhar':0, 'Khanna':0,
'Ludhiana':0, 'Mandi Gobindgarh':0, 'Patiala':0, 'Rupnagar':0, 'Ajmer':0,
'Alwar':0, 'Bhiwadi':0, 'Jaipur':0, 'Jodhpur':0, 'Kota':0, 'Pali':0, 'Udaipur':0,
'Chennai':0, 'Hyderabad':0, 'Agra':0, 'Baghpat':0, 'Bulandshahr':0,
'Ghaziabad':0, 'Greater_Noida':0, 'Kanpur':0, 'Lucknow':0, 'Moradabad':0,
'Muzaffarnagar':0, 'Noida':0, 'Varanasi':0, 'Asanol':0, 'Durgapur':0,
'Haldia':0, 'Howrah':0, 'Kolkata':0, 'Siliguri':0})
```

In []:

```
df['Faridabad'] = df.city.map({'Amaravati':0, 'Rajamahendravaram':0, 'Tirupati':0,
'Vijayawada':0,
'Visakhapatnam':0, 'Gaya':0, 'Muzaffarpur':0, 'Patna':0, 'Delhi':0,
'Ahmedabad':0, 'Faridabad':1, 'Gurugram':0, 'Manesar':0, 'Panchkula':0,
'Rohtak':0, 'Jorapokhar':0, 'Bengaluru':0, 'Chikkaballapur':0, 'Hubballi':0,
'Thiruvananthapuram':0, 'Dewas':0, 'Mandideep':0, 'Pithampur':0, 'Satna':0,
'Singrauli':0, 'Ujjain':0, 'Aurangabad':0, 'Chandrapur':0, 'Mumbai':0,
'Nagpur':0, 'Nashik':0, 'Pune':0, 'Solapur':0, 'Thane':0, 'Brajrajnagar':0,
'Talcher':0, 'Amritsar':0, 'Bathinda':0, 'Jalandhar':0, 'Khanna':0,
'Ludhiana':0, 'Mandi Gobindgarh':0, 'Patiala':0, 'Rupnagar':0, 'Ajmer':0,
'Alwar':0, 'Bhiwadi':0, 'Jaipur':0, 'Jodhpur':0, 'Kota':0, 'Pali':0, 'Udaipur':0,
'Chennai':0, 'Hyderabad':0, 'Agra':0, 'Baghpat':0, 'Bulandshahr':0,
'Ghaziabad':0, 'Greater_Noida':0, 'Kanpur':0, 'Lucknow':0, 'Moradabad':0,
'Muzaffarnagar':0, 'Noida':0, 'Varanasi':0, 'Asanol':0, 'Durgapur':0,
'Haldia':0, 'Howrah':0, 'Kolkata':0, 'Siliguri':0})
```

In []:

```
df['Gurugram'] = df.city.map({'Amaravati':0, 'Rajamahendravaram':0, 'Tirupati':0,
'Vijayawada':0,
'Visakhapatnam':0, 'Gaya':0, 'Muzaffarpur':0, 'Patna':0, 'Delhi':0,
'Ahmedabad':0, 'Faridabad':0, 'Gurugram':1, 'Manesar':0, 'Panchkula':0,
'Rohtak':0, 'Jorapokhar':0, 'Bengaluru':0, 'Chikkaballapur':0, 'Hubballi':0,
'Thiruvananthapuram':0, 'Dewas':0, 'Mandideep':0, 'Pithampur':0, 'Satna':0,
'Singrauli':0, 'Ujjain':0, 'Aurangabad':0, 'Chandrapur':0, 'Mumbai':0,
'Nagpur':0, 'Nashik':0, 'Pune':0, 'Solapur':0, 'Thane':0, 'Brajrajnagar':0,
'Talcher':0, 'Amritsar':0, 'Bathinda':0, 'Jalandhar':0, 'Khanna':0,
'Ludhiana':0, 'Mandi Gobindgarh':0, 'Patiala':0, 'Rupnagar':0, 'Ajmer':0,
'Alwar':0, 'Bhiwadi':0, 'Jaipur':0, 'Jodhpur':0, 'Kota':0, 'Pali':0, 'Udaipur':0,
'Chennai':0, 'Hyderabad':0, 'Agra':0, 'Baghpat':0, 'Bulandshahr':0,
'Ghaziabad':0, 'Greater_Noida':0, 'Kanpur':0, 'Lucknow':0, 'Moradabad':0,
'Muzaffarnagar':0, 'Noida':0, 'Varanasi':0, 'Asanol':0, 'Durgapur':0,
'Haldia':0, 'Howrah':0, 'Kolkata':0, 'Siliguri':0})
```

B.SCREENSHOTS

jupyter M1 Last Checkpoint: 11 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

In [18]: `df.head(10)`

Out[18]:

	Country	State	city	place	lastupdate	Avg	Max	Min	Pollutants
0	India	Andhra Pradesh	Amaravati	Secretariat, Amaravati - APPCB	21-12-2018 03:00:00	70.0	108.0	42.0	PM2.5
1	India	Andhra Pradesh	Amaravati	Secretariat, Amaravati - APPCB	21-12-2018 03:00:00	76.0	102.0	43.0	PM10
2	India	Andhra Pradesh	Amaravati	Secretariat, Amaravati - APPCB	21-12-2018 03:00:00	73.0	118.0	46.0	NO2
3	India	Andhra Pradesh	Amaravati	Secretariat, Amaravati - APPCB	21-12-2018 03:00:00	5.0	6.0	4.0	NH3
4	India	Andhra Pradesh	Amaravati	Secretariat, Amaravati - APPCB	21-12-2018 03:00:00	41.0	109.0	2.0	SO2
5	India	Andhra Pradesh	Amaravati	Secretariat, Amaravati - APPCB	21-12-2018 03:00:00	44.0	102.0	18.0	CO
6	India	Andhra Pradesh	Amaravati	Secretariat, Amaravati - APPCB	21-12-2018 03:00:00	29.0	35.0	12.0	OZONE
12	India	Andhra Pradesh	Rajahmendravaram	Anand Kala Kshetram, Rajamahendravaram - APPCB	21-12-2018 03:00:00	30.0	103.0	2.0	CO
13	India	Andhra Pradesh	Rajahmendravaram	Anand Kala Kshetram, Rajamahendravaram - APPCB	21-12-2018 03:00:00	108.0	130.0	51.0	OZONE
14	India	Andhra Pradesh	Tirupati	Tirumala, Tirupati - APPCB	21-12-2018 03:00:00	46.0	72.0	28.0	PM2.5

In [19]: `#shape
df.shape`

Out[19]: (788, 9)

In [20]: `#show columns
df.columns`

Out[20]: Index(['Country', 'State', 'city', 'place', 'lastupdate', 'Avg', 'Max', 'Min', 'Pollutants'], dtype='object')

Fig B.1Module 1- Dataset after preprocessing

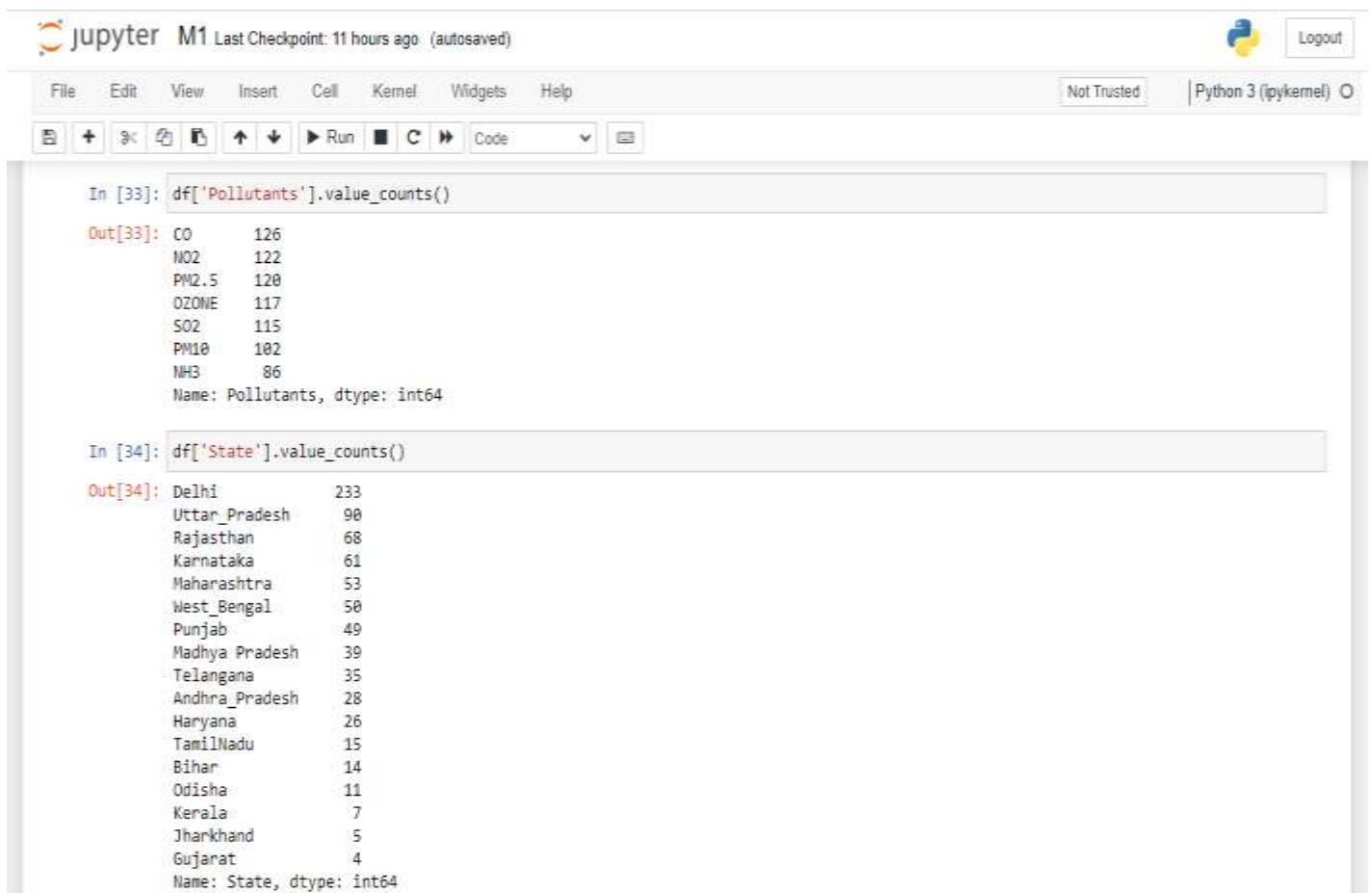


Fig b.2 Module 2 - value count of pollutants and states in the dataset

Jupyter M2 Last Checkpoint: 12/21/2021 (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Splitting Train/Test:

```
In [18]: #preprocessing, split test and dataset, split response variable
X = df.drop(labels='class', axis=1)
#Response variable
y = df.loc[:, 'class']
```

```
In [19]: #We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there c
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)
print("Number of training dataset: ", len(X_train))
print("Number of test dataset: ", len(X_test))
print("Total number of dataset: ", len(X_train)+len(X_test))
```

Number of training dataset: 551
Number of test dataset: 237
Total number of dataset: 788

Fig b.3 Training and testing dataset - Split

jupyter M2 Last Checkpoint: 12/21/2021 (autosaved) Python 3 (ipykernel)  Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted

Run    Code 

Pollutants

```
In [13]: from sklearn.preprocessing import LabelEncoder
var_mod = ['Country', 'State', 'city', 'place', 'lastupdate', 'Avg', 'Max', 'Min', 'Pollutants']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(str)
```

```
In [14]: df['AQI'] = df.Avg.map({'00': 'satisfactor', '74': 'satisfactor', '71': 'satisfactor', '4': 'good', '39': 'good', '42': 'good', '27': 'good'})
df.head()
```

```
In [16]: df['class'] = df.AQI.map({'good':1, 'moderate':0, 'servere':0, 'verypoor':0, 'satisfactor':1, 'poor':0})
```

```
In [17]: df.head()
```

Out[17]:

	Country	State	city	place	lastupdate	Avg	Max	Min	Pollutants	AQI	class
0	0	0	4	99	0	88	106	41	5	satisfactor	1
1	0	0	4	99	0	74	99	42	4	satisfactor	1
2	0	0	4	99	0	71	115	45	2	satisfactor	1
3	0	0	4	99	0	4	5	3	1	good	1
4	0	0	4	99	0	39	106	1	6	good	1

Fig B.4 By Air quality index assigning values as binary

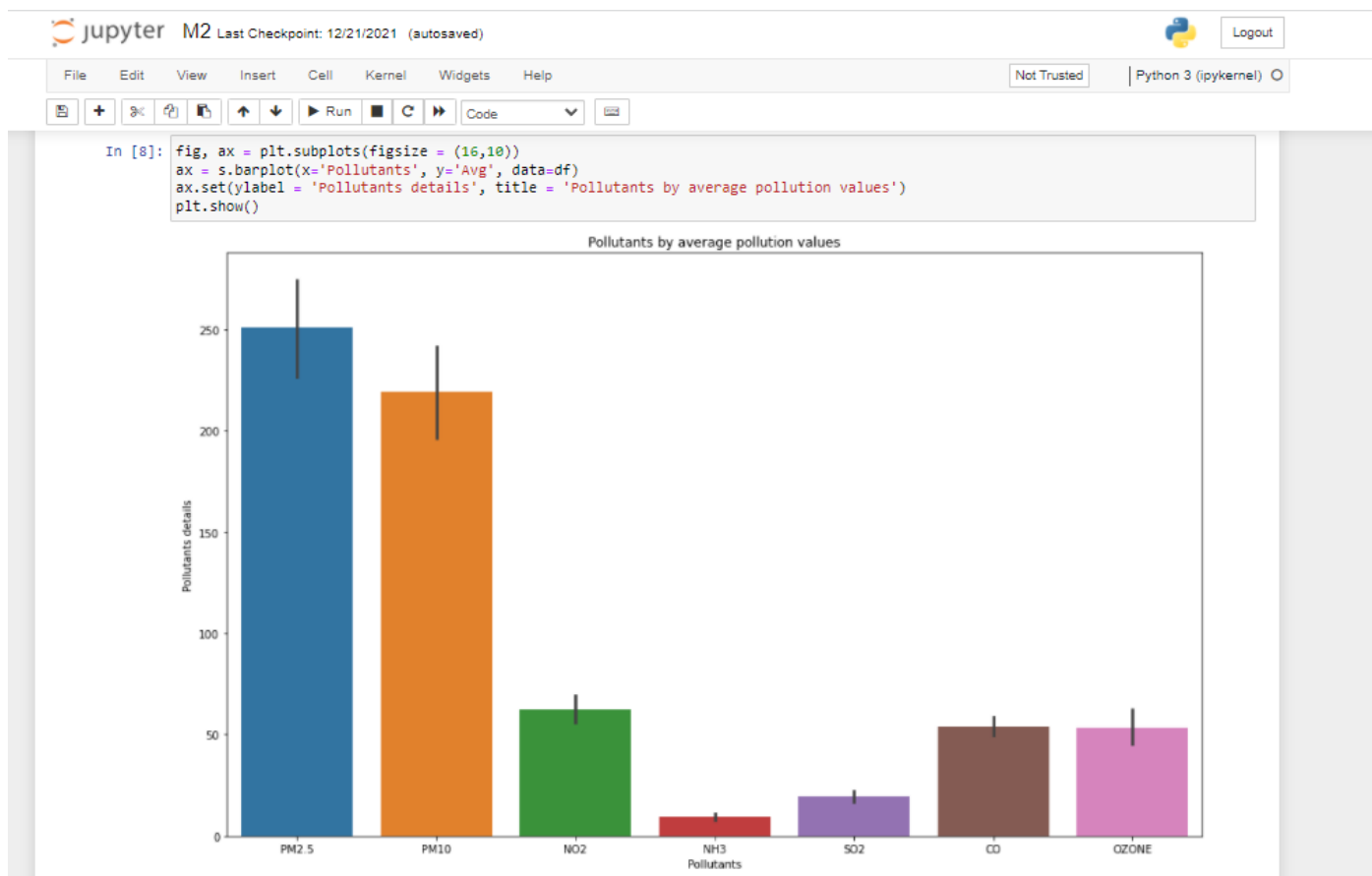


Fig B 5 Bar plot representation of each pollutant and their range

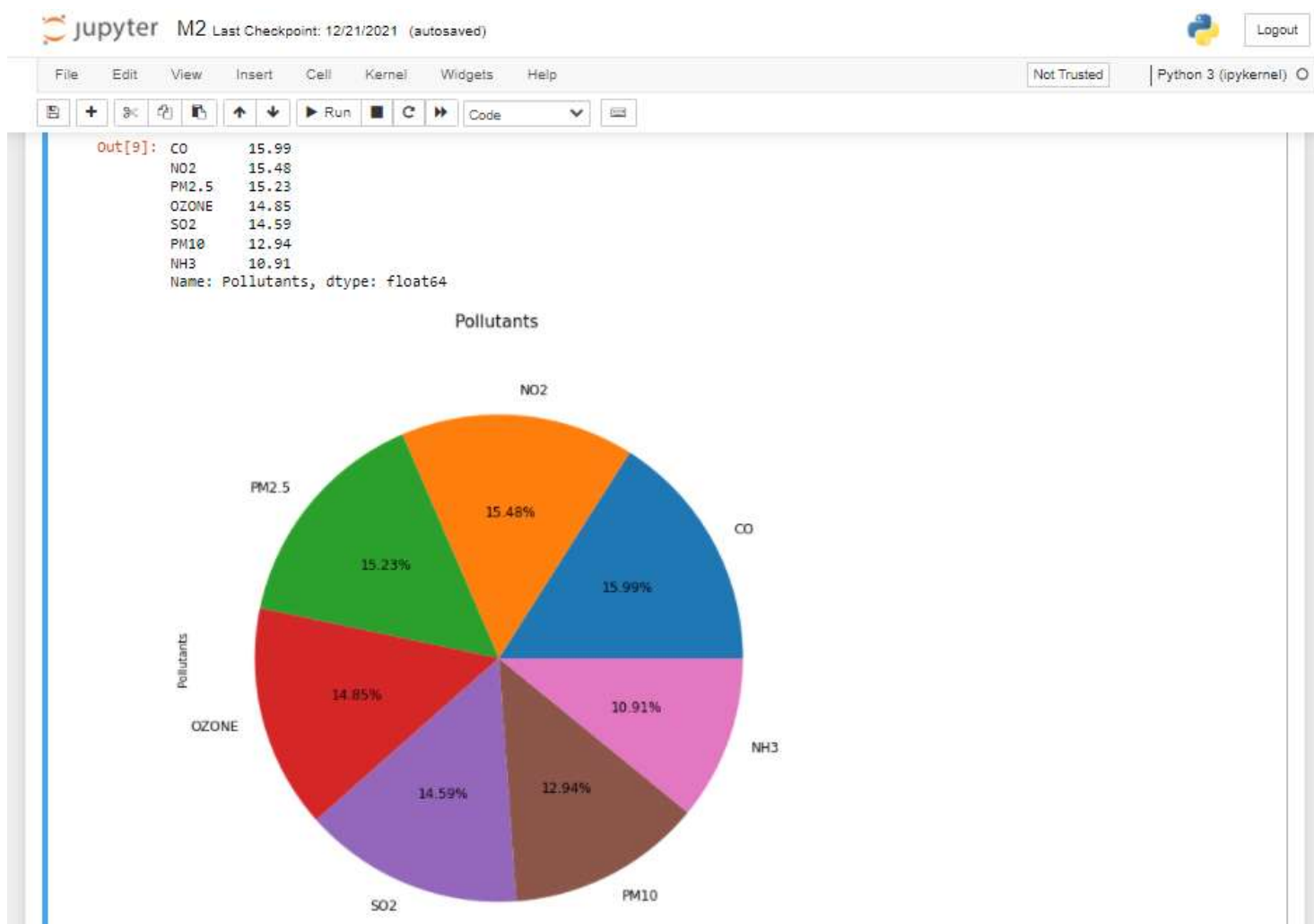


Fig B.6 Pie chart representation of each pollutant and their percentage

```
In [23]: def plot_confusion_matrix(cm2, title='Confusion matrix-logR', cmap=plt.cm.Blues):
    target_names=[]
    plt.imshow(cm2, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = n.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

    cm2=confusion_matrix(y_test, predictLR)
    print('Confusion matrix-logR:')
    print(cm2)
    plot_confusion_matrix(cm2)
```

```
Confusion matrix-logR:
[[ 72  1]
 [ 1 174]]
```

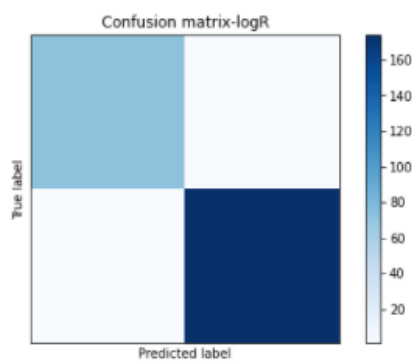


Fig B.7 Module 3 - Confusion matrix for predicted label and true label

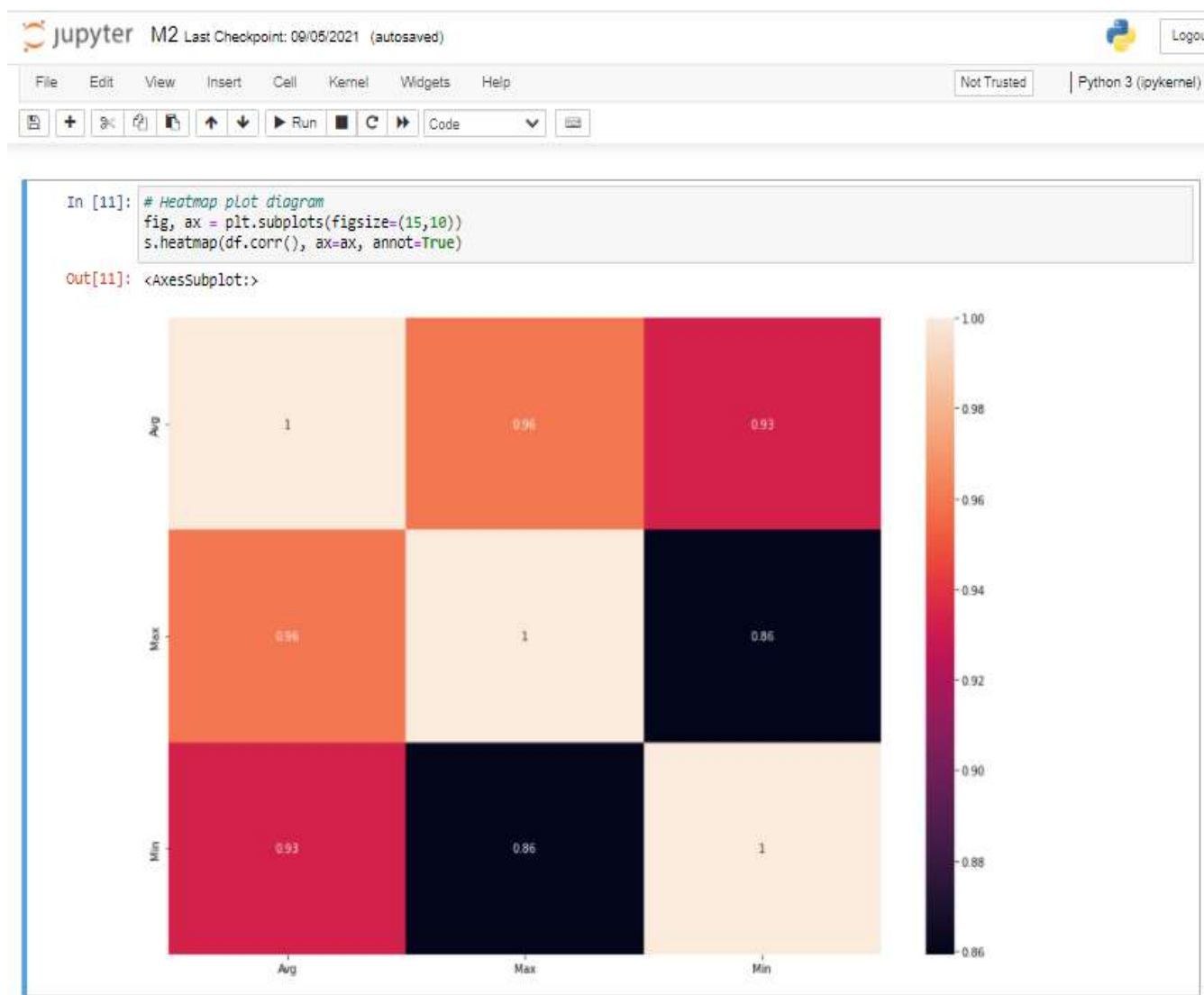


Fig B.8 Heat map representation of the pollutants

```
In [12]: df.boxplot(column="Avg", by="Pollutants")
```

```
Out[12]: <AxesSubplot:title={'center':'Avg'}, xlabel='Pollutants'>
```

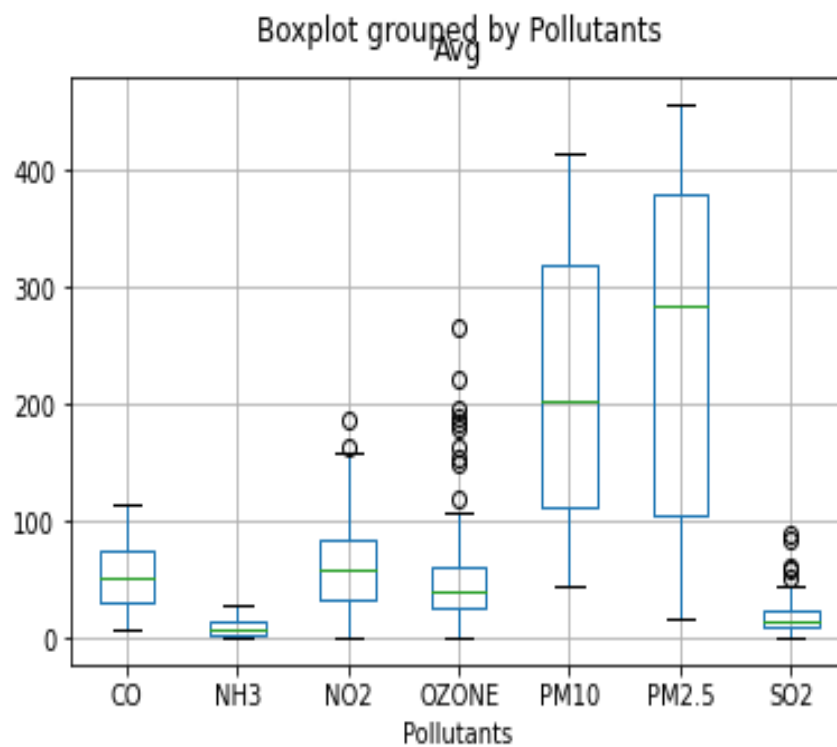


Fig B.9 Box plot representation of the pollutants

```

In [16]: TN = cm2[1][0]
FN = cm2[0][0]
TP = cm2[1][1]
FP = cm2[0][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV = TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)

True Positive : 174
True Negative : 1
False Positive : 1
False Negative : 72

True Positive Rate : 0.7073170731707317
True Negative Rate : 0.5
False Positive Rate : 0.5
False Negative Rate : 0.2926829268292683

Positive Predictive Value : 0.9942857142857143

```

Fig B.10 Calculation of positive predictive and negative predictive value

AIR QUALITY PREDICTION - INDIA

NAME:

Check: Prediction of Pollutants

Pollution Prediction:

STATE NAME:

Check: AQI-Stages

AQI-Cause:
(Satisfactory)-May cause minor breathing discomfort to sensitive people.

CITY NAME:

Check: Sources

Sources of Pollutants:
Internal combustion engines burning fossil fuels, Cigarette smoke, Butane, Kerosene heaters and Stoves.

AQI VALUE:

Reset

Fig 11 GUI – Results predicted by the machine