# A MAJOR PROJECT REPORT

**ON**

**TO ANALYZE THE DATASET FOR AIR POLLUTION BY AIR QUALITY PREDICTION BASED ON SUPERVISED WITH CLASSIFICATION MACHINE LEARNING APPROACH**

*Submitted in partial fulfillment of the requirements for the award of degree of*

BACHELOR OF TECHNOLOGY

**IN**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

SUBMITTED

BY

-

**SAMALA ARUN KUMAR - 18P61A04I2**

**ANTHWAR NAVATHA RAO - 19P65A0423**

**GEESA MANIDEEP - 16P61A0477**

Under the Esteemed Guidance of **MD.MOHSIN ALI ASSISTANT PROFESSOR**

(Affiliated to JNTU Hyderabad, Approved by APSCHE &AICTE) Aushapur (v), Ghatkesar (m), MedchalDist, Hyderabad-501301

2021

-



# Department of Electronics & Communication Engineering

### CERTIFICATE

This is to certify that the Mini project report,**“ To Analyze The Dataset For Air Pollution By Air Quality Prediction Based On Supervised With Classification Machine Learning Approach”** being submitted by **SAMALA ARUN KUMAR (18P61A04I2),ANTHWAR NAVATHA RAO (19P65A0423),GEESA MANIDEEP (16P61A0477)** in partial fulfillment for the award of the Degree of Bachelor of Technology in **ELECTRONICS & COMMUNICATION ENGINEERING** to Jawaharlal Nehru Technological University is a record of a bonafide work carried out by them under my guidance and supervision.

The result(s) embodied in this project report have not been submitted to any other University/Institution for the award of any Degree/Diploma.

**Internal Guide**

**MD.MOHSIN ALI**

**Assistant Professor**

**Head of the Department**

**Dr.POORNA LAKSHMI**

**Associate Professor**



# Department of Electronics & Communication Engineering

### CANDIDATES DECLARATION

We hereby declare that this Mini Project report titled “**To Analyze The Dataset For Air Pollution By Air Quality Prediction Based On Supervised With Classification Machine Learning Approach”** submitted by us to the Department of **Electronics and Communication Engineering,** VBIT, Aushapur, Under JNTUH, is a bonafide work undertaken by and it is not submitted to any other University or Institution for the award of any degree or diploma.

By

**SAMALA ARUN KUMAR (18P61A04I2) ANTHWAR NAVATHA RAO (19P65A0423) GEESA MANIDEEP (16P61A0477)**



### ACKNOWLEDGEMENT

Firstly, we would thank our parents who have been a motivating factor throughout our lives.

Secondly, we sincerely thank our principal, **Dr. PVS SRINIVAS** and our Head of the department, **Dr POORNA LAKSHMI. U** for their kind cooperation and Encouragement for the successful completion of Seminar work and providing the necessary facilities.

We are most obliged and grateful to our project guide, **MD.MOHSIN ALI** ,for giving us guidance in completing this project successfully.

We express our sincere gratitude to our Project coordinators, Department of ECE and my other faculty for attending my project seminars and for their insightful comments and constructive suggestions to improve the quality of this project work.

BY

**SAMALA ARUN KUMAR (18P61A04I2) ANTHWAR NAVATHA RAO (19P65A0423) GEESA MANIDEEP (16P61A0477)**

**Abstract:**

Generally, Air pollution refers to the release of pollutants into the air that are detrimental to human health and the planet as a whole. 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 analyze 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 supervise 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.

**Keywords:** dataset, Machine learning-Classification method, python, Prediction of Accuracy result.

CERTIFICATES

CANDIDATE DECLARATION

ACKNOWLEDGMENT ABSTRACT

LIST OF FIGURES LIST OF TABLES

**TABLE OF CONTENTS**

I II III IV

VII VIII

[CHAPTER-1 INTRODUCTION](#_TOC_250000) 1

INTRODUCTION 1

AQI Table…………………………………………………………………………………………………. 3

LITERATURE SURVEY…………………………………………………………………………………..5

CHAPTER-2……………………………………………………………………………8-9

OUTLINE OF THE PROJECT…………………………………………………….………………………8

OVERVIEW OF THE PROJECT………………………………………………………………………….8

PROJECT GOALS…………………………………………………………………………………………8

OBJECTIVES………………………………………………………………………………………………9

SCOPES…………………………………………………………………………………………………….9

PROBLEM STATEMENT…………………………………………………………………………………9

CHAPTER-3 10-13

EXISTING SYSTEM……………………………………………………………………………………………10

PROPOSED SYSTEM…………………………………………………………………………………………..11

CONSTRUCTION OF A PREDICTIVE MODEL……………………………………………..………………12

TRAINING AND TESTING……………………………………………………………………………………12-13

CHAPTER-4 14

PROJECT REQUIREMENTS……………………….…………………………………………14

SOFTWARE REQUIREMENTS………………………………….……………………………15

CHAPTER-5…………………………………………………………………………….18

DESIGN ENGINEERING…………………………………………………………………………………18

WORKFLOW DIAGRAM…………………………………………………...……………………………19

CLASS DIAGRAM………………………………………………………… ..…………………………...20

ACTIVITY DIAGRAM……………………………………………………………………………………22

SEQUENCE DIAGRAM………………………………………………………………………………….23

ENTITY RELATION DIAGRAM………………………………………………………………………..24

CHAPTER-6……………………………………………………………………………25

MODULES DESCRIPTION……………………………………………………………….………………25

OUTLIER DETECTION PROCESS……………………………………………………………………….31

CROSS VALIDATION PROCESS…………………………………………………………………...……33

CHAPTER-7……………………………………………………………………………35

ALGORITMS AND TECHNOLOGY…………………………………….………………………………35

LOGISTIC REGRESSION………………………………………………………………………….……...35

DECISION TREE…………………………………………………………………………………………..36

K NEARST NEIGHBOURS……………………………………………………………………………….36

RANDOM FOREST ……………………………………………………………………………………….36

SUPPORT VECTOR MACHINE………………………………………………………………………….37

CHAPTER-8……………………………………………………………………….38

SAMPLE CODE…………………………………………………………………………………38

CHAPTER-9……………………………………………………………………52

RESULT AND DISCUSSION……………………………………………………………………52

CONCLUSION & FUTURE WORK--------------------------------------------------------55

REFERENCES----------------------------------------------------------------------------------56

V

LIST OF FIGURES



**S.NO**

**TITLE**

**PAGE.NO**

**1 WORKFLOW DIAGRAM 19**

**2 CLASS DIAGRAM 20**

**3 ACTIVITY DIAGRAM 22**

**4 SEQUENCE DIAGRAM 23**

**5 ENTITY RELATION DIAGRAM 24**

## LIST OF TABLES

**S.NO TITLE PAGE .NO**

1 AQI TABLE 3

2 PREPARING OF DATASET 4

CHAPTER-1

**INTRODUCTION**

**Domain overview**

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.

Analyses Predicts

Machine Learning

Past Dataset

Trains

Fig: Process of Machine learning

[Supervised Machine Learning](https://www.geeksforgeeks.org/supervised-unsupervised-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”.

Air pollution refers to the release of pollutants into the air that are detrimental to human health and the planet as a whole. It can be described as one of the most dangerous threats that the humanity ever faced. It causes damage to animals, crops and forests. It also contributes to the depletion of the ozone layer, which protects the Earth from the sun's UV rays. Some of the other environmental effects of air pollution are haze, eutrophication, and global climate changes. Most air pollution comes from energy use and production. Burningfossil fuels releases gases and chemicals into the air. And in an especially destructive feedback loop, air pollution not only contributes to climate change but is also exacerbated by it. Air pollution in the form of carbon dioxide and methane raises the earth’s temperature. Another type of air pollution is then worsened by that increased heat: Smog forms when the weather is warmer and there’s more ultraviolet radiation. Climate change also increases the production of allergenic air pollutants including mold (thanks to damp conditions caused by extreme weather and increased flooding) and pollen (due to a longer pollen season and more pollen production). Air pollution can be defined as an alteration of air quality that can be characterized by measurements of chemical, biological or physical pollutants in the air. Therefore, air pollution means the undesirable presence of impurities or the abnormal rise in the proportion of some constituents of the atmosphere. It can be classified in 2 sections: **visible** and **invisible** air pollution.

## Air pollution causes:

Air pollution is caused by the presence in the atmosphere of toxic substances, mainly produced by human activities, even though sometimes it can result from natural phenomena such as volcanic eruptions, dust storms and wildfires, also depleting the air quality.

Anthropogenic air pollution sources are:

* Combustion of fossil fuels, like coal and oil for electricity and road transport, producing air pollutants like nitrogen and sulfur dioxide.
* Emissions from industries and factories, releasing large amount of carbon monoxide, hydrocarbon, chemicals and organic compounds into the air.
* Agricultural activities, due to the use of pesticides, insecticides, and fertilizers that emit harmful chemicals.
* Waste production, mostly because of methane generation in landfills.

**Air Quality Index for India:**

|  |  |
| --- | --- |
| **AQI** | **Associated Health Impacts** |
| Good (0–50) | Minimal impact |
| Satisfactory (51–100) | May cause minor breathing discomfort to sensitive people. |
| Moderately polluted (101–200) | May cause breathing discomfort to people with lung disease such as asthma, and discomfort to people with heart disease, children and older adults. |
| Poor (201–300) | May cause breathing discomfort to people on prolonged exposure, and discomfort to people with heart disease. |
| Very poor (301–400) | May cause respiratory illness to the people on prolonged exposure. Effect may be more pronounced in people with lung and heart diseases. |
| Severe (401–500) | May cause respiratory impact even on healthy people, and serious health impacts on people with lung/heart disease. The health impacts may be experienced even during light physical activity. |

In India, as in many other countries, the Index is centred around five chief pollutants – Particulate Matter with a diameter less than 10 micrometres (PM10), Particulate Matter with a diameter of less than 2.5 micrometers (PM2.5), ozone (O3), Nitrogen Dioxide (NO2) ), and Carbon Monoxide (CO). A monitoring station should be able to give you the concentration of a particular pollutant at that moment in time, and its average over a period of time – for CO and O3, the average is taken over eight hours, while for the other three, it is a 24-hour average. The unit of measurement is microgram (or milligram in the case of CO) per cubic meter.

**Preparing the Dataset**

The dataset is now supplied to machine learning model on the basis of this data set the model is trained. Every new data details filled at the time of application form acts as a test data set.

Table shows details of the datasets:

|  |  |
| --- | --- |
| **Variable** | **Description** |
| Country | Home country (India) |
| State | Indian States name lists |
| city | City names for each state |
| place | Place names for each city |
| lastupdate | Date and time (DD/MM/YYYY HH:MM) |
| Avg | Average range of pollutants |
| Max | Maximum range of pollutants |
| Min | Minimum range of pollutants |
| Pollutants | Pollutants name |

**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. Loan default trends have been long studied from a socio-economic stand point. Most economics surveys believe in empirical modeling of these complex systems in order to be able to predict the loan default rate for a particular individual. The use of machine learning for such tasks is a trend which it is observing now. Some of the survey’s to understand the past and present perspective of loan approval or not.

**Review of Literature Survey**

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

**Author:** Ishan Verma, Rahul Ahuja and Hardik Meisheri

**Year :** 2018

Recurrent Neural Networks (RNN) has proved to be very efficient in processing temporal data. However, future input information coming up later than the current time instance is also useful for prediction. RNNs can partially achieve this by delaying the output by a certain number of time frames to include future information. Theoretically, a large delay could be used but in practice, it is found that prediction results drop if the delay is too large. While delaying the output by some frames has been used successfully to improve results for sequential data, the optimal delay is task dependent and need to be obtained by the trial and error method. Also, two separate networks, one for each direction could be trained on all input information and then the results could be merged using arithmetic or geometric averaging for final prediction. However, it is difficult to obtain optimal merging since different networks trained on the same data can no longer be regarded as independent. To overcome these limitations, it proposed bidirectional recurrent neural network (BRNN) that can be trained using all available input information in the past and future of a specific time frame. Pollutant data like any other sensor data is not free from missing data and abnormal values. The irregularities may occur due to instrumental error or some other external factors like power-shutdown or severance of connectivity etc. There were instances where pollutant data was not reported by a source monitoring station. These missing values were interpolated using rolling average of available data values of past three time instances. A value lying outside the permissible range for a parameter is treated as an abnormal value. Abnormal values are also replaced by rolling average of past three instances. It 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 PM2.5 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.

**Title :** Air pollution monitoring and prediction using IoT

**Author:** Temesegan Walelign Ayele, Rutvik Mehta

**Year :** 2018

It is obvious that those who work in a factory or plant will be far more at risk of inhalation of harmful chemicals and gases due to their prolonged exposure to emissions. Air pollution adds to the harmful condition that makes unfavorable impact on living things. It is one of the real concerns for the entire world. Air pollution is a worldwide issue including international organizations, governments, and the mass media. Any utilization of natural assets at a higher rate than the nature's ability to reestablish itself can bring about contamination of plants, air, and water. Other than human exercises, there are a couple of intermittent characteristic cycles that additionally result in release of risky stuff. Beside human made activities natural disaster such as volcanic eruption may result in the contamination of air. Technology is passing around its wing in almost every walk of human life activities. Now a day it is better if every action is done using new technology in order to satisfy the demand of human being, Organization, Enterprise etc. Internet of Things (IoT) is one of the main communication developments in the last decade. Through this concept, it is possible to connect countless low-powered smart embedded objects to each other and to the Internet. The pervasive presence around us of various wireless technologies such as radio Frequency Identification (RFID) tags, sensors, actuators and mobile phones constitutes the cornerstone of the IoT concept. These objects can send and receive data autonomously, thus opening new horizons for home, health, and industrial applications. In fact, technology advances along with increasing demand will foster a widespread deployment of IoT services, which would radically transform our corporations, communities and personal lives. Now a day the air pollution in urban areas is a major issue in developed cities due to significant impacts of air pollution on public health, global environment and the whole worldwide economy. The proposed work on an air pollution monitoring and prediction system is enables us to monitor air quality with the help IoT devices. The system utilizes air sensors to detect and transmit this data to microcontroller. Then the microcontroller stores the data into the web server. For predicting the LSTM is implemented. It has a quick convergence and reduces the training cycles with a good accuracy.

**Title :** Adverse health effects of outdoor air pollutants

**Author:** Luke Curtis, William Rea, Patricia Smith-Willis

**Year :** 2006

Outdoor air quality plays an important role in human health. Air pollution causes large increases in medical expenses, morbidity and is estimated to cause about 800,000 annual premature deaths worldwide (Cohen et al., 2005). The outdoor air often contains biologically significantly levels of many pollutants including particulates (PM10 or PM2.5), ozone, carbon monoxide, oxides of nitrogen and sulfur, bioaerosols, metals, volatile organics and pesticides. A large percentage of these pollutants are produced by anthropogenic activities. While most people spend the majority of their time indoors, outdoor air quality can affect indoor air quality to a large degree. In addition many patients such as asthmatics, patients with allergies and chemical sensitivities, COPD patients, heart and stroke patients, diabetics, pregnant women, the elderly and children are especially susceptible to poor outdoor and indoor air quality. Much research on the health effects of outdoor air pollution has been published in the last decade. The goal of this review is to concisely summarize a wide range of the recent research on health effects of many types of outdoor air pollution. A review of the health effects of major outdoor air pollutants including particulates, carbon monoxide, sulfur and nitrogen oxides, acid gases, metals, volatile organics, solvents, pesticides, radiation and bio aerosols is presented. Numerous studies have linked atmospheric pollutants to many types of health problems of many body systems including the respiratory, cardiovascular, immunological, hematological, neurological and reproductive/ developmental systems.

**Title :** Urban Air Pollution Monitoring System with Forecasting Models

**Author:** Khaled Bashir Shaban, Abdullah Kadri and Eman Rezk

**Year :** 2016

It is widely believed that urban air pollution has a direct impact on human health especially in developing and industrial countries, where air quality measures are not available or minimally implemented or enforced. Recent studies have shown substantial evidences that exposure to atmospheric pollutants has strong links to adverse diseases including asthma and lung inflammation. The modules are responsible for receiving and storing the data, preprocessing and converting the data into useful information, forecasting the pollutants based on historical information, and finally presenting the acquired information through different channels, such as mobile application, Web portal, and short message service. The focus of this paper is on the monitoring system and its forecasting module. Three machine learning (ML) algorithms are investigated to build accurate forecasting models for one-step and multi-step ahead of concentrations of ground-level ozone (O3), nitrogen dioxide (NO2), and sulfur dioxide (SO2). These ML algorithms are support vector machines, M5P model trees, and artificial neural networks (ANN). Two types of modeling are pursued: 1) uni-variate and 2) multivariate. The performance evaluation measures used are prediction trend accuracy and root mean square error (RMSE). The results show that using different features in multivariate modeling with M5P algorithm yields the best forecasting performances. For example, using M5P, RMSE is at its lowest, reaching 31.4, when hydrogen sulfide (H2S) is used to predict SO2. Contrarily, the worst performance, i.e., RMSE of 62.4, for SO2 is when using ANN in uni-variate modeling. The outcome of this paper can be significantly useful for alarming applications in areas with high air pollution levels. Air quality is an important problem that directly affects human health. Air quality data are collected wirelessly from monitoring motes that are equipped with an array of gaseous and meteorological sensors. These data are analyzed and used in forecasting concentration values of pollutants using intelligent machine to machine platform. The platform uses ML-based algorithms to build the forecasting models by learning from the collected data.

**Title :** A Comprehensive Evaluation of Air Pollution Prediction Improvement by a

Machine Learning Method

**Author:** Xia Xi, Zhao Wei and Rui Xiaoguang

**Year :** 2015

Urban air pollution prediction is one of the most important tasks in the treatment of urban air pollution. Due to the disadvantage that source list updated not in time for WRFChem which is a numeric model, the prediction result may be not good enough and it take full advantages of forecast on pollution, weather, chemical component from WRF-Chem model as input features, design a comprehensive evaluation framework to improve the prediction performance. Experiments are implemented with different features groups and classification algorithms in machine learning method for 74 cities in China, to find the best model for each city. From experiments, for different city, the best result can be obtained by different group of feature selection and model selection. Experimental results indicate that the more feature it used, the more possibility to enhance the accuracy. For method aspect, the result from combined model is better than the unique model. In recent years, with the rapid development of China’s economy and society, the concentration of pollutants in the atmosphere has increased significantly. Issues in environmental air quality assessment and pollution control attached public attentions. An air quality index (AQI) is a number used by government agencies to communicate to the public how polluted the air currently is or how polluted it is forecast to become. As the AQI increases, an increasingly large percentage of the population is likely to experience increasingly severe adverse health effects. Air quality prediction can timely provide to the government the changing trend of the atmospheric environment quality which could be used to effectively conduct air pollution control and management. The Air Quality Index (AQI) is a quantitative method to profile air pollution level. The daily AQI is an index for reporting daily air quality. It is determined by the maximum value of the Individual Air Quality Index (IAQI), which is calculated from mass concentrations of PM10 and PM2.5, SO2, NO2, CO and O3 in ambient air respectively; furthermore, the pollutant with maximum IAQI is called the primary pollutant. AQI is measured at monitoring stations throughout 120 cities and reported daily by MEP-China.

CHAPTER-2

**OUTLINE OF THE PROJECT**

**Overview of the system**

Adverse health impacts from exposure to outdoor air pollutants are complicated functions of pollutant compositions and concentrations. Major outdoor air pollutants in cities include ozone (O3), particle matter (PM), sulfur dioxide (SO2), carbon monoxide (CO), nitrogen oxides (NOx), volatile organic compounds (VOCs), pesticides, and metals, among others. Increased mortality and morbidity rates have been found in association with increased air pollutants (such as O3, PM and SO2) concentrations. Although O3 precursor (such as VOCs, NOx, and CO) emissions have significantly decreased since, O3 levels in have not been in compliance with standards set by the Environmental Protection Agency (EPA) to protect public health. Particle size is critical in determining the particle deposition location in the human respiratory system. PM2.5, referring to particles with a diameter less than or equal to 2.5, has been an increasing concern, as these particles can be deposited into the lung gas-exchange region. This helps all others department to carried out other formalities. It have to find Accuracy of the training dataset, Accuracy of the testing dataset, Specification, False Positive rate, precision and recall by comparing algorithm using python code. The following Involvement steps are,

* Define a problem
* Preparing data
* Evaluating algorithms
* Improving results
* Predicting results

The steps involved in Building the data model is depicted below.

**Data collection** (Splitting Training set & Test set)

**Building classification Model**

**Pre Processing**

**Prediction (**Air quality/ no quality**)**

Fig: data flow diagram for Machine learning model

**Project Goals**

# Exploration data analysis of variable identification

* Loading the given dataset
* Import required libraries packages
* Analyze the general properties
* Find duplicate and missing values
* Checking unique and count values

# Uni-variate data analysis

* Rename, add data and drop the data
* To specify data type

# Exploration data analysis of bi-variate and multi-variate

* Plot diagram of pairplot, heatmap, bar chart and Histogram

# Method of Outlier detection with feature engineering

* Pre-processing the given dataset
* Splitting the test and training dataset
* Comparing the Decision tree and Logistic regression model and random forest etc

# Comparing algorithm to predict the result

* Based on the best accuracy

#### *Objectives*

The goal is to develop a machine learning model for real-time air quality forecasting, to potentially replace the updatable supervised machine learning classification models by predicting results in the form of best accuracy by comparing supervised algorithm.

**Problem Description/ Problem Statements**

Monitoring and preserving air quality has become one of the most essential activities in many industrial and urban areas today. The quality of air is adversely affected due to various forms of pollution caused by transportation, electricity, fuel uses etc. The deposition of harmful gases is creating a serious threat for the quality of life in smart cities. With increasing air pollution, we need to implement efficient air quality monitoring models which collect information about the concentration of air pollutants and provide assessment of air pollution in each area.

**Scope**

The scope of this project is to investigate a dataset of air pollutants records for India meteorological sector using machine learning technique. To identifying air quality is more difficult. we try to reduce this risk factor behind predicting from Air Quality Index (AQI) of India to safe human so as to save lots of meteorological efforts and assets and to predict whether assigning the air quality is bad or good.

CHAPTER-3

**EXISTING SYSTEM**

Urban air pollutant concentration prediction is dealing with a surge of massive environmental monitoring data and complex changes in air pollutants. This requires effective prediction methods to improve prediction accuracy and prevent serious pollution incidents, thereby enhancing environmental management decision-making capacity. A new pollutant concentration prediction method is based on vast amounts of environmental data and deep learning techniques. This integrates big data using two kinds of deep networks. This method is based on a design that uses a Convolutional Neural Network as the base layer, automatically extracting features of input data. A Long Short Term Memory network is used for the output layer to consider the time dependence of pollutants. It consists of these two deep networks. With performance optimization, the model can predict future particulate matter (PM2:5) concentrations as time series. Finally, the prediction results are compared with the results of numerical models. The applicability and advantages of the model are also analyzed. Experimental results show that it improves prediction performance compared with classic models. Air pollution has attracted substantial attention regarding the daily life of people. It has negative impacts on human health and daily life during episodes of severe air pollution with the increase of sources and types of air pollutants, the complexity of pollutant concentration prediction has increased. Therefore, it is necessary to use environmental monitoring data to more accurately predict urban air pollutant concentrations. Conventional prediction methods, such as numerical analysis and machine learning, are widely used in this type of prediction. However, several drawbacks of these methods have been recently identified as follows. First, numerical prediction methods are based on experience as summarized by historical data or the nature of pollutant change.

It is a machine learning process that can carry out a series of training for sample data through unsupervised training methods and obtain a deep network structure. It exploited CNN characteristics that compress and extract important features of input data, along with the unique structure of LSTM designed for the time-series problem. The prediction model uses the CNN as the base layer, compressing and extracting features using its convolutional and pooling layers. The output of the CNN layer is the input of a higher layer, LSTM, for the time series prediction. Meteorological factors and pollutant concentration from the past are included in the prediction model as input. They are converted to several two-dimensional matrices with time series. Then, these matrices are input to the CNN network to extract the features. The output is used as input to the LSTM. The fully connected layer is used to decode the LSTM output and obtain the final prediction result. Monitoring and preserving air quality has become one of the most essential activities in many industrial and urban areas today. The quality of air is adversely affected due to various forms of pollution caused by transportation, electricity, fuel uses etc. The deposition of harmful gases is creating a serious threat for the quality of life in smart cities. With increasing air pollution, we need to implement efficient air quality monitoring models which collect information about the concentration of air pollutants and provide assessment of air pollution in each area. Hence, air quality evaluation and prediction has become an important research area. The quality of air is affected by multi-dimensional factors including location, time, and uncertain variables. Recently, many researchers began to use the big data analytics approach due to advancements in big data applications and availability of environmental sensing networks and sensor data. The aim of this research paper is to investigate various big-data and machine learning based techniques for air quality forecasting.

Air is one of the most essential natural resources for the existence and survival of the entire life on this planet. All forms of life including plants and animals depend on air for their basic survival. Thus, all living organisms need good quality of air which is free of harmful gases to continue their life.

**Drawbacks**

* The training data of our model is used from multiple sites.
* the work is given based on only one city (Shanghai) and we want to collect more monitoring data from other cities to verify the generalization of our work and more factors, e.g., geomorphic conditions, need to be taken into account in our future work.
* We can thereby better determine the regularity of air pollutant data and achieve more accurate prediction results.

**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.

**Preprocessing**

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve the efficiency of the algorithm. The outliers have to be removed and also variable conversion need to be 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.

**Construction of a Predictive Model**

## Machine learning needs data gathering have lot of past data’s. Data gathering have sufficient historical data and raw data. Before data pre-processing, raw data can’t be used directly. It’s used to preprocess then, what kind of algorithm with model. Training and testing this model working and predicting correctly with minimum errors. Tuned model involved by tuned time to time with improving the accuracy.

Data Gathering

Data Pre-Processing

Choose model

Train model

Prediction

Tune model

Test model

Fig: Process of dataflow diagram

**Training the Dataset**

* The first line imports iris data set which is already predefined in sklearn module and raw data set is basically a table which contains information about various varieties.
* For example, to import any algorithm and train\_test\_split class from sklearn and numpy module for use in this program.
* To encapsulate load\_data() method in data\_dataset variable. Further divide the dataset into training data and test data using train\_test\_split method. The X prefix in variable denotes the feature values and y prefix denotes target values.
* This method divides dataset into training and test data randomly in ratio of 67:33 / 70:30. Then we encapsulate any algorithm.
* In the next line, we fit our training data into this algorithm so that computer can get trained using this data. Now the training part is complete.

**Testing the Dataset**

* Now, the dimensions of new features in a numpy array called ‘n’ and it want to predict the species of this features and to do using the predict method which takes this array as input and spits out predicted target value as output.
* So, the predicted target value comes out to be 0. Finally to find the test score which is the ratio of no. of predictions found correct and total predictions made and finding accuracy score method which basically compares the actual values of the test set with the predicted values.

Data Processing

Test dataset

Classification ML Algorithm

Model

Training dataset

Air Pollutants

Fig: Architecture of Proposed model

**General Properties**

Create cells freely to explore the given data and it should not perform too many operations in each cell. One option that can take with this is to do a lot of explorations in an initial notebook. These don't have to be organized, but make sure you use enough comments to understand the purpose of each code cell. Then, after done with your analysis, create a duplicate notebook where it will trim the excess and organize steps so that have a flowing, cohesive report and make sure that informed on the steps that are taking in your investigation. Follow every code cell, or every set of related code cells, with a markdown cell to describe to the reader what was found in the preceding cell. Try to make it so that the reader can then understand what they will be seeing in the following cell. The AQI is an index for reporting daily air quality. It tells you how clean or unhealthy your air is, and what associated health effects might be a concern. The AQI focuses on health affects you may experience within a few hours or days after breathing unhealthy air. The AQI is calculated for four major air pollutants regulated by the Clean Air Act: [ground level ozone](https://airnow.gov/index.cfm?action=aqi_brochure.index#ozone), [particle pollution](https://airnow.gov/index.cfm?action=aqi_brochure.index#particle), [carbon monoxide](https://airnow.gov/index.cfm?action=aqi_brochure.index#carbon), and [sulfur dioxide](https://airnow.gov/index.cfm?action=aqi_brochure.index#sulfur). For each of these pollutants, EPA has established national air quality standards to protect public health. EPA is currently reviewing the national air quality standard for nitrogen dioxide. If the standard is revised, the AQI will be revised as well. Think of the AQI as a yardstick that runs from 0 to 500. The higher the AQI value, the greater the level of air pollution and the greater the health concern. For example, an AQI value of 50 represents good air quality with little or no potential to affect public health, while an AQI value over 300 represents air quality so hazardous that everyone may experience serious effects. An AQI value of 100 generally corresponds to the national air quality standard for the pollutant, which is the level EPA has set to protect public health. AQI values at or below 100 are generally thought of as satisfactory. When AQI values are above 100, air quality is considered to be unhealthy—at first for certain sensitive groups of people, then for everyone as AQI values increase.

**Advantages:**

* These reports are to the investigation of applicability of machine learning techniques for air quality forecasting in operational conditions.
* Finally, it highlights some observations on future research issues, challenges, and needs.

CHAPTER-4

**Project Requirements**

**General:**

Requirements are the basic constrains that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements

2. Non-Functional requirements

3. Environment requirements

A. Hardware requirements

B. software requirements

**Functional requirements:**

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

**Non-Functional Requirements:**

Process of functional steps,

1. Problem define
2. Preparing data
3. Evaluating algorithms
4. Improving results
5. Prediction the result

**Environmental Requirements:**

1. Software Requirements:

Operating System : Windows

Tool : Anaconda with Jupyter Notebook

2. Hardware requirements:

Processor : Pentium IV/III

Hard disk : minimum 80 GB

RAM : minimum 2 GB

**SOFTWARE DESCRIPTION**

Anaconda is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source) distribution of the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) and [R](https://en.wikipedia.org/wiki/R_(programming_language)) programming languages for [scientific computing](https://en.wikipedia.org/wiki/Scientific_computing) ([data science](https://en.wikipedia.org/wiki/Data_science), [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications, large-scale data processing, [predictive analytics](https://en.wikipedia.org/wiki/Predictive_analytics), etc.), that aims to simplify [package management](https://en.wikipedia.org/wiki/Package_management) and deployment. Package versions are managed by the [package management system](https://en.wikipedia.org/wiki/Package_manager) “Conda”. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the [Conda](https://en.wikipedia.org/wiki/Conda_(package_manager)" \o "Conda (package manager)) package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently. The open source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. [Pip packages](https://en.wikipedia.org/wiki/Pip_(package_manager)) provide many of the features of conda packages and in most cases they can work together. Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud,[[10]](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_note-AnacondaCloud-10) [PyPI](https://en.wikipedia.org/wiki/Python_Package_Index" \o "Python Package Index) or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

Anaconda Navigator

Anaconda Navigator is a desktop [graphical user interface (GUI)](https://en.wikipedia.org/wiki/Graphical_user_interface) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using [command-line commands](https://en.wikipedia.org/wiki/Command-line_interface). Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux).

The following applications are available by default in Navigator:

* [JupyterLab](https://en.wikipedia.org/wiki/Project_Jupyter#Jupyter_Lab)
* [Jupyter Notebook](https://en.wikipedia.org/wiki/Project_Jupyter#Jupyter_Notebook)
* [QtConsole](https://qtconsole.readthedocs.io/en/latest/)
* [Spyder](https://en.wikipedia.org/wiki/Spyder_(software))
* [Glueviz](http://glueviz.org/)
* [Orange](https://en.wikipedia.org/wiki/Orange_(software))
* [Rstudio](https://en.wikipedia.org/wiki/Rstudio)
* [Visual Studio Code](https://en.wikipedia.org/wiki/Visual_Studio_Code)

### Conda:

Conda is an [open source](https://en.wikipedia.org/wiki/Open-source_software), [cross-platform](https://en.wikipedia.org/wiki/Cross-platform), language-agnostic [package manager](https://en.wikipedia.org/wiki/Package_manager) and environment management system that installs, runs and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., [R](https://en.wikipedia.org/wiki/R_(programming_language))), including multi-languages. The Conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

### The Jupyter Notebook

#### *The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.*

## [**Notebook document**](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#id5)**:**

Notebook documents (or “notebooks”, all lower case) are documents produced by the [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html" \l "notebook-app), which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc…). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

## [**Jupyter Notebook App**](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#id6)**:**

The Jupyter Notebook App is a server-client application that allows editing and running [notebook documents](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-document) via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet. In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a “Dashboard” ([Notebook Dashboard](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#dashboard)), a “control panel” showing local files and allowing to open notebook documents or shutting down their [kernels](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#kernel).

## [**kernel**](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#id7)**:**

A notebook kernel is a “computational engine” that executes the code contained in a [Notebook document](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-document). The ipython kernel, referenced in this guide, executes python code. Kernels for many other languages exist ([official kernels](http://jupyter.readthedocs.org/en/latest/#kernels)). When you open a [Notebook document](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-document), the associated kernel is automatically launched. When the notebook is executed (either cell-by-cell or with menu Cell -> Run All), the kernel performs the computation and produces the results. Depending on the type of computations, the kernel may consume significant CPU and RAM. Note that the RAM is not released until the kernel is shut-down.

## [**Notebook Dashboard**](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#id8)**:**

The Notebook Dashboard is the component which is shown first when you launch [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html" \l "notebook-app). The Notebook Dashboard is mainly used to open [notebook documents](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-document), and to manage the running [kernels](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#kernel) (visualize and shutdown). The Notebook Dashboard has other features similar to a file manager, namely navigating folders and renaming/deleting files.

**Working Process:**

* Download and install anaconda and get the most useful package for machine learning in Python.
* Load a dataset and understand its structure using statistical summaries and data visualization.
* machine learning models, pick the best and build confidence that the accuracy is reliable.

Python is a popular and powerful interpreted language. Unlike R, Python is a complete language and platform that you can use for both research and development and developing production systems. There are also a lot of modules and libraries to choose from, providing multiple ways to do each task. It can feel overwhelming.

The best way to get started using Python for machine learning is to complete a project.

* It will force you to install and start the Python interpreter (at the very least).
* It will give you a bird’s eye view of how to step through a small project.
* It will give you confidence, maybe to go on to your own small projects.

When you are applying machine learning to your own datasets, you are working on a project. A machine learning project may not be linear, but it has a number of well-known steps:

* Define Problem.
* Prepare Data.
* Evaluate Algorithms.
* Improve Results.
* Present Results.

The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

Here is an overview of what we are going to cover:

1. Installing the Python anaconda platform.
2. Loading the dataset.
3. Summarizing the dataset.
4. Visualizing the dataset.
5. Evaluating some algorithms.
6. Making some predictions.

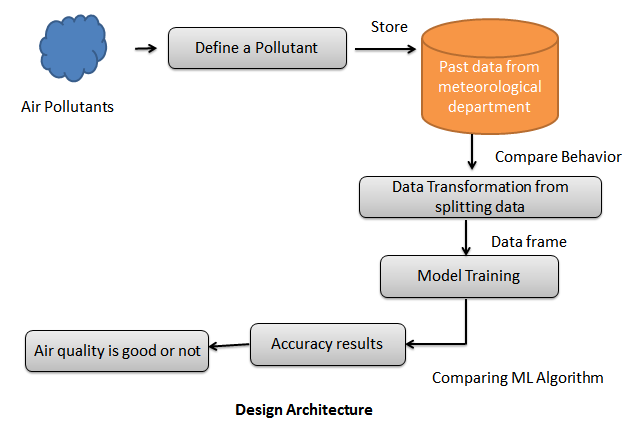
**CHAPTER-5**

**DESIGN ENGINEERING**

**General**

Design is meaningful engineering representation of something that is to be built. Software design is a process design is the perfect way to accurately translate requirements in to a finished software product. Design creates a representation or model, provides detail about software data structure, architecture, interfaces and components that are necessary to implement a system.

**System Architecture**



**Work flow diagram**

Source Data

Data Processing and Cleaning

Testing Dataset

Training Dataset

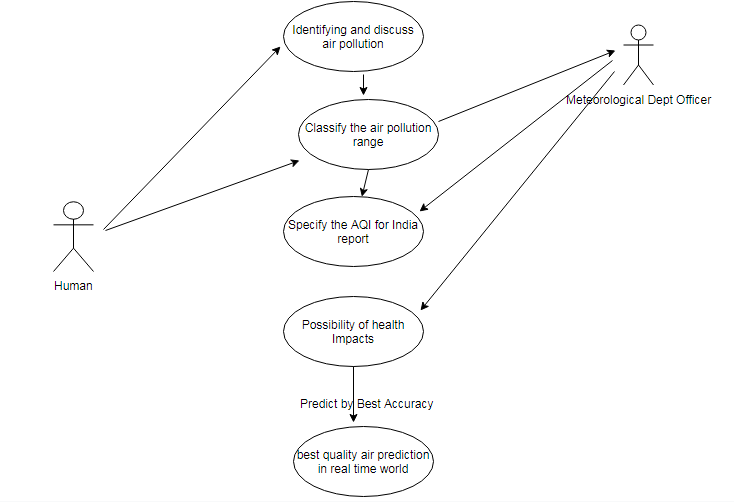
Best Model by Accuracy

Classification ML Algorithms

Finding air quality by AQI

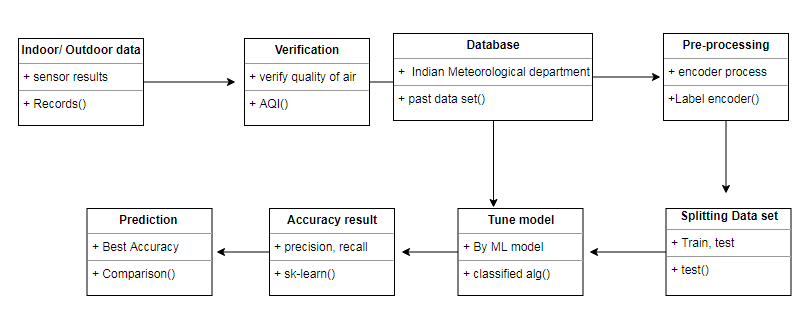
Fig: Workflow Diagram

**Use Case Diagram**



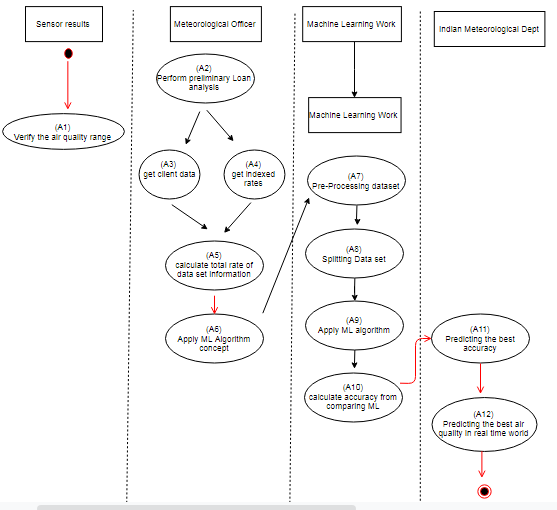
Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

**Class Diagram**:



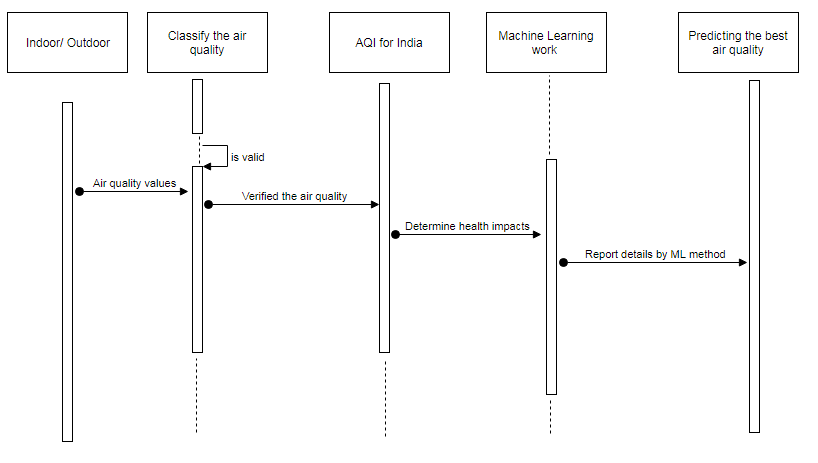
Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

**Activity Diagram**:



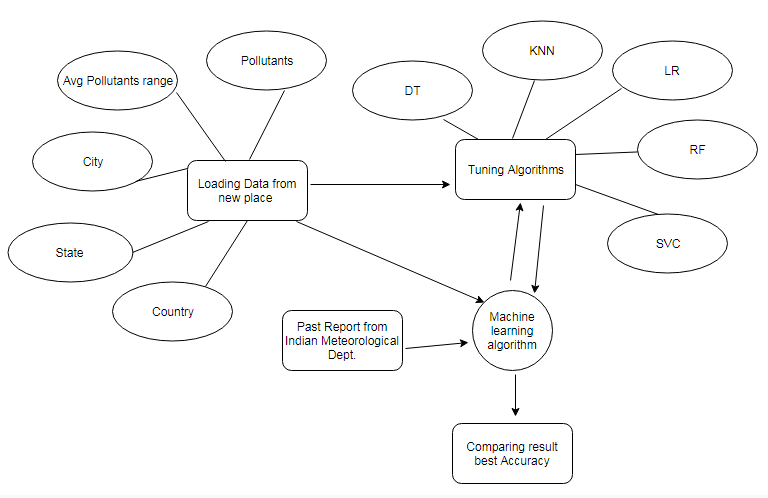
Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

**Sequence Diagram**:



Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system. Other dynamic modeling techniques include [activity diagramming](http://agilemodeling.com/artifacts/activityDiagram.htm), [communication diagramming](http://agilemodeling.com/artifacts/communicationDiagram.htm), [timing diagramming](http://agilemodeling.com/artifacts/timingDiagram.htm), and [interaction overview diagramming](http://agilemodeling.com/artifacts/interactionOverviewDiagram.htm). Sequence diagrams, along with [class diagrams](http://agilemodeling.com/artifacts/classDiagram.htm) and [physical data models](http://agiledata.org/essays/dataModeling101.html) are in my opinion the most important design-level models for modern business application development.

**Entity Relationship Diagram (ERD)**

****

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a [data modeling](https://searchdatamanagement.techtarget.com/definition/data-modeling) technique that can help define business processes and be used as the foundation for a [relational database](https://searchdatamanagement.techtarget.com/definition/relational-database). Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

**CHAPTER-6**

**MODULES DESCRIPTION**

**List of Modules:**

* Variable Identification Process
* Exploration data analysis of visualization
* Probability of Loan Analysis
* Outlier detection process
* Comparing Algorithm with prediction in the form of best accuracy result

**Variable Identification Process**

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 following diagram is given dataset.



Fig: data frame of demo dataset

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. For example, time series data can be analyzed by regression algorithms; classification algorithms can be used to analyze discrete data. For example to show the data type format of given dataset.

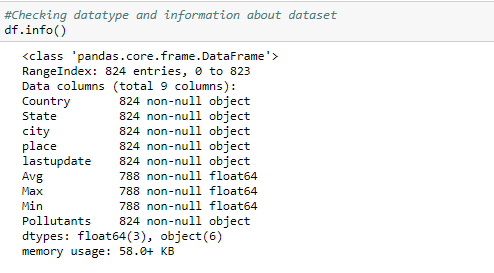


Fig: data type of demo dataset

A number of different **data cleaning** tasks using Python’s [Pandas library](https://pandas.pydata.org/) and specifically, it focus on probably the biggest data cleaning task, **missing values** and it able to **more**[**quickly clean data**](https://www.dataoptimal.com/data-cleaning-with-python-2018/). It wants to **spend less time cleaning data**, and more time exploring and modeling.

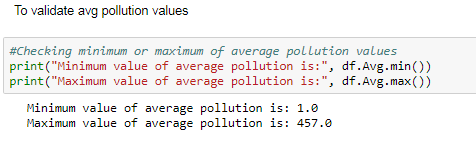


Fig: Validation of average pollution values

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](https://en.wikipedia.org/wiki/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](https://github.com/matthewbrems/ODSC-missing-data-may-18/blob/master/Analysis%20with%20Missing%20Data.pdf). Before, joint into code, it’s important to understand the sources of missing data. Here are some typical reasons why data is missing:

* User forgot to fill in a field.
* Data was lost while transferring manually from a legacy database.
* There was a programming error.
* Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

Variable identification with Uni-variate, Bi-variate and Multi-variate analysis:

* import libraries for access and functional purpose and read the given dataset
* General Properties of Analyzing the given dataset
* Display the given dataset in the form of data frame
* show columns
* shape of the data frame
* To describe the data frame
* Checking data type and information about dataset
* Checking for duplicate data
* Checking Missing values of data frame
* Checking unique values of data frame
* Checking count values of data frame
* Rename and drop the given data frame
* To specify the type of values
* To create extra columns

Data Validation/ Cleaning/Preparing Process

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.

**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.

* How to chart time series data with line plots and categorical quantities with bar charts.
* How to summarize data distributions with histograms and box plots.
* How to summarize the relationship between variables with scatter plots.

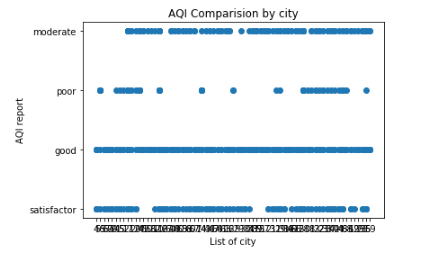
Data Visualization types:

* + Introduction to Matplotlib
  + Line Plot
  + Bar Chart
  + Histogram Plot
  + Box and Whisker Plot
  + Scatter Plot

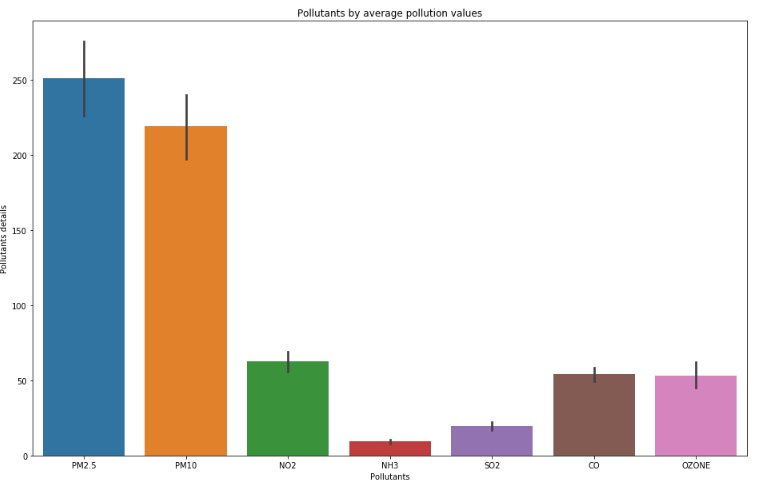
There are many excellent plotting libraries in Python and it recommend exploring them in order to create presentable graphics. For quick and dirty plots intended for your own use, it recommends using the matplotlib library. It is the foundation for many other plotting libraries and plotting support in higher-level libraries such as Pandas. The matplotlib provides a context, one in which one or more plots can be drawn before the image is shown or saved to file and context can be accessed via functions on pyplot.

Bar Chart:

A bar chart is generally used to present relative quantities for multiple categories. The x-axis represents the categories and are spaced evenly. The y-axis represents the quantity for each category and is drawn as a bar from the baseline to the appropriate level on the y-axis.



**Fig: AQI by each city**



**Fig: Pollutants by average pollution values**

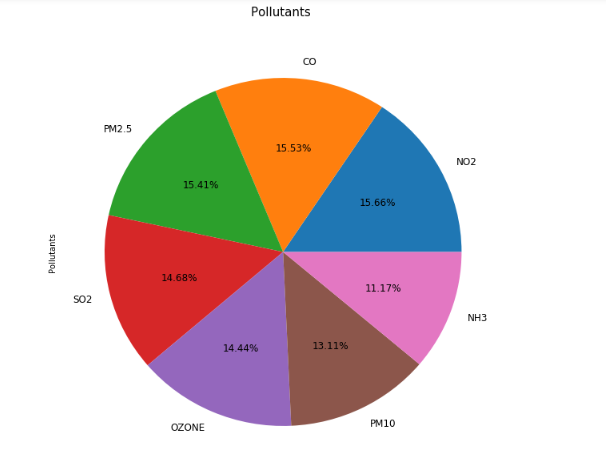


Fig: Percentage of each pollutants

Box and Plot:

A box and whisker plot, or boxplot for short, is generally used to summarize the distribution of a data sample. The x-axis is used to represent the data sample, where multiple boxplots can be drawn side by side on the x-axis if desired. The boxplot is a graphical technique that displays the distribution of variables. It helps us see the location, skewness, spread, tile length and outlying points. The boxplot is a graphical representation of the Five Number Summary.

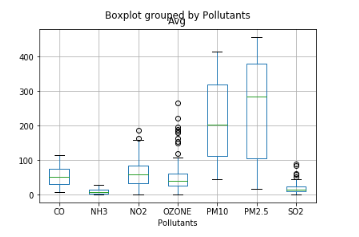


Fig: Average pollution range of each pollutant

Heat map

A [heat map](https://www.data-to-viz.com/graph/heatmap.html) is a graphical representation of data where the individual values contained in a matrix are represented as colors. It is a bit like looking a data table from above. It is really useful to display a general view of numerical data, not to extract specific data point. It is quite straight forward to make a heat map, as shown on the examples below.

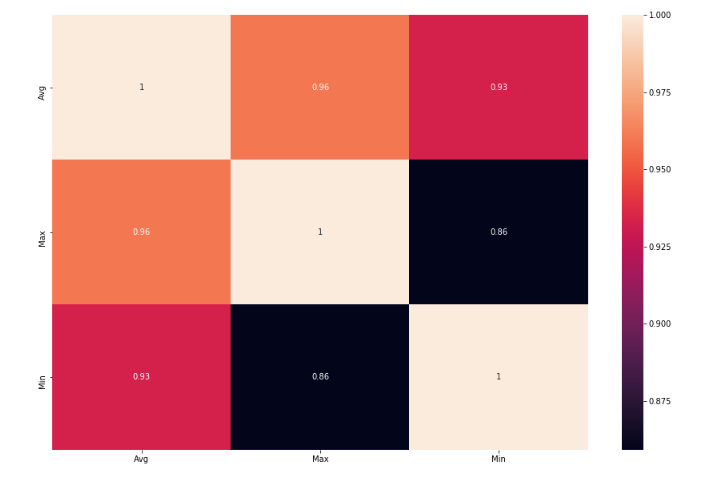


Fig: Heat map diagram of given dataset

**Outlier detection process**

Many machine learning algorithms are sensitive to the range and distribution of attribute values in the input data. Outliers in input data can skew and mislead the training process of machine learning algorithms resulting in longer training times, less accurate models and ultimately poorer results.

Even before predictive models are prepared on training data, outliers can result in misleading representations and in turn misleading interpretations of collected data. Outliers can skew the summary distribution of attribute values in descriptive statistics like mean and standard deviation and in plots such as histograms and scatterplots, compressing the body of the data. Finally, outliers can represent examples of data instances that are relevant to the problem such as anomalies in the case of fraud detection and computer security.

It couldn’t fit the model on the training data and can’t say that the model will work accurately for the real data. For this, we must assure that our model got the correct patterns from the data, and it is not getting up too much noise. Cross-validation is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary subset of the data-set.

The three steps involved in cross-validation are as follows:

1. Reserve some portion of sample data-set.
2. Using the rest data-set train the model.
3. Test the model using the reserve portion of the data-set.

Advantages of train/test split:

1. This runs K times faster than Leave One Out cross-validation because K-fold cross-validation repeats the train/test split K-times.
2. Simpler to examine the detailed results of the testing process.

Advantages of cross-validation:

1. More accurate estimate of out-of-sample accuracy.
2. More “efficient” use of data as every observation is used for both training and testing.

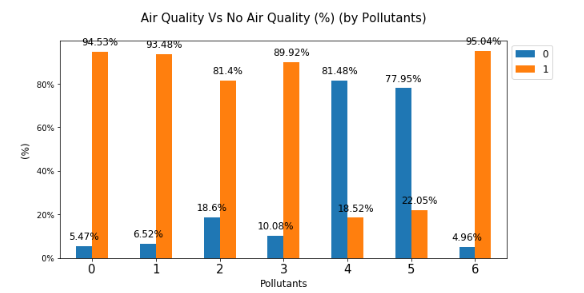
****

Fig: Air Quality good Vs Air quality bad

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. To achieving better results from the applied model in Machine Learning method of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values. Therefore, to execute random forest algorithm null values have to be managed from the original raw data set. And another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in given dataset.

**False Positives (FP):** A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

**False Negatives (FN):** A person who default predicted as payer. When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

**True Positives (TP):** A person who will not pay predicted as defaulter. These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

**True Negatives (TN):** A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

**Comparing Algorithm with prediction in the form of best accuracy result**

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.

In the example below 5 different algorithms are compared:

* Logistic Regression
* Random Forest
* K-Nearest Neighbors
* Decision tree
* Support Vector Machines

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.

**Cross validation process:**

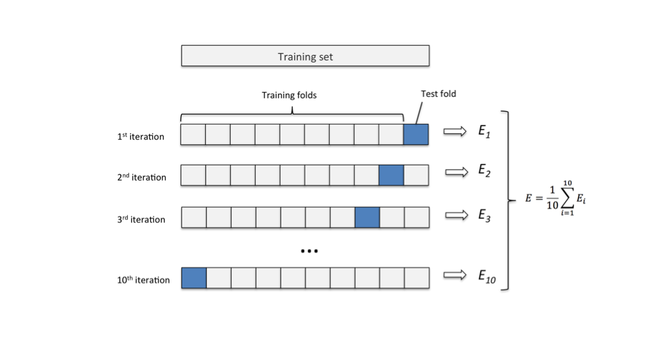
****

Fig: Cross validation process

Over-fitting is a common problem in machine learning which can occur in most models. K-fold cross-validation can be conducted to verify that the model is not over-fitted. In this method, the data-set is randomly partitioned into kmutually exclusive subsets, each approximately equal size and one is kept for testing while others are used for training. This process is iterated throughout the whole k folds.

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:**

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. (When the model predicts default: how often is correct?)

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)

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- Measure = 2TP / (2TP + FP + FN)

**F1-Score Formula:**

F1 Score = 2\*(Recall \* Precision) / (Recall + Precision)

**CHAPTER-7**

**ALGORITHM AND TECHNIQUES**

**Algorithm Explanation**

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.

Used Python Packages:

**sklearn:**

* + In python, sklearn is a machine learning package which include a lot of ML algorithms.
  + Here, we are using some of its modules like train\_test\_split, DecisionTreeClassifier or Logistic Regression and accuracy\_score.

**NumPy:**

* + It is a numeric python module which provides fast maths functions for calculations.
  + It is used to read data in numpy arrays and for manipulation purpose.

**Pandas:**

* + Used to read and write different files.
  + Data manipulation can be done easily with data frames.

**Matplotlib:**

* + Data visualization is a useful way to help with identify the patterns from given dataset.
  + Data manipulation can be done easily with data frames.

[**Logistic Regression**](https://en.wikipedia.org/wiki/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.

[**Decision Tree**](https://www.geeksforgeeks.org/decision-tree/)

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.

**K*-*Nearest Neighbor (KNN)**

K-Nearest Neighbor is a supervised machine learning algorithm which stores all instances correspond to training data points in n-dimensional space. When an unknown discrete data is received, it analyzes the closest k number of instances saved (nearest neighbors) and returns the most common class as the prediction and for real-valued data it returns the mean of k nearest neighbors. In the distance-weighted nearest neighbor algorithm, it weights the contribution of each of the k neighbors according to their distance using the following query giving greater weight to the closest neighbors.

Usually KNN is robust to noisy data since it is averaging the k-nearest neighbors. The k-nearest-neighbors algorithm is a classification algorithm, and it is supervised: it takes a bunch of labeled points and uses them to learn how to label other points. To label a new point, it looks at the labeled points closest to that new point (those are its nearest neighbors), and has those neighbors vote, so whichever label the most of the neighbors have is the label for the new point (the “k” is the number of neighbors it checks). Makes predictions about the validation set using the entire training set. KNN makes a prediction about a new instance by searching through the entire set to find the k “closest” instances. “Closeness” is determined using a proximity measurement (Euclidean) across all features.

**Random Forest**

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](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/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.

**Support Vector Machines**

A classifier that categorizes the data set by setting an optimal hyper plane between data. I chose this classifier as it is incredibly versatile in the number of different kernelling functions that can be applied and this model can yield a high predictability rate. Support Vector Machines are perhaps one of the most popular and talked about machine learning algorithms. They were extremely popular around the time they were developed in the 1990s and continue to be the go-to method for a high-performing algorithm with little tuning.

* How to disentangle the many names used to refer to support vector machines.
* The representation used by SVM when the model is actually stored on disk.
* How a learned SVM model representation can be used to make predictions for new data.
* How to learn an SVM model from training data.
* How to best prepare your data for the SVM algorithm.
* Where you might look to get more information on SVM.

**CHAPTER-8**

**SAMPLE CODE**

#import library packages

import pandas as p

import matplotlib.pyplot as plt

import seaborn as s

import numpy as n

df = p.read\_csv("aqi.csv")

#show columns

df.columns

#shape of the dataframe

df.shape

#To describe the dataframe

df.describe()

#Checking datatype and information about dataset

df.info()

#Checking for duplicate data

df.duplicated()

#find sum of duplicate data

sum(df.duplicated())

#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())

#check the average pollution range

print("Avg pollution range :", sorted(df['Avg'].unique()))

#check the over pollution value from AQI Remark

good=df.query('Avg<=50') # Between 0 - 50

satisfactor=df.query('Avg>50 & Avg<=100') # Between 51 - 100

moderate=df.query('Avg>100 & Avg<=200') # Between 101 - 200

poor=df.query('Avg>200 & Avg<=300') # Between 201 - 300

verypoor=df.query('Avg>300 & Avg<=400') # Between 301 - 400

servers=df.query('Avg>400') # Between 400 - 500 (above 400)

satisfactor.head()

moderate.head()

df.Pollutants.unique()

PMkitchen = df.loc[df['Pollutants'] == 'PM2.5']

PMclean = df.loc[df['Pollutants'] == 'PM10']

N02 = df.loc[df['Pollutants'] == 'NO2']

NH3 = df.loc[df['Pollutants'] == 'NH3']

S02 = df.loc[df['Pollutants'] == 'SO2']

CO = df.loc[df['Pollutants'] == 'CO']

O3 = df.loc[df['Pollutants'] == 'OZONE']

n\_pollutants = df.Pollutants.nunique()

pmk = PMkitchen.shape[0]

pmc = PMclean.shape[0]

n = N02.shape[0]

nh = NH3.shape[0]

s = S02.shape[0]

c = CO.shape[0]

o = O3.shape[0]

print ("total number of Pollutants:", n\_pollutants)

print("")

print ("Number of Kitchen pollution - Particulate Matter(PM2.5) affected in each state:",pmk )

print ("Number of Dust pollution - Particulate Matter(PM10) affected in each state:",pmc )

print ("Number of Chemical pollution - Nitrogen dioxide(N02) affected in each state:",n)

print ("Number of Chemical pollution - Ammonia(NH3) affected in each state:",nh )

print ("Number of Chemical pollution - Sulfur dioxide(SO2) affected in each state:",s)

print ("Number of Chemical pollution - Carbon monoxide(CO) affected in each state:",c)

print ("Number of Chemical pollution - Ozone(O3) affected in each state:",o)

# percent of missing "Avg Pollution"

print('Percent of missing "Average pollution values" records is %.2f%%' %((df['Avg'].isnull().sum()/df.shape[0])\*100))

listTN = df[df["State"]=="TamilNadu"]

listTN.shape

p.Categorical(df['Pollutants']).describe()

df.corr()

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

df["Avg"].unique()

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.dtypes

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

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'})

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

tn = p.crosstab(df['class'],df['Pollutants'])

#outliers

% matplotlib inline

import matplotlib.pyplot as plt

fig=plt.figure()

ax = fig.add\_subplot(1,1,1)

#variable

ax.scatter(df['city'],df['AQI'])

#labels and lit

plt.title('AQI Comparision by city')

plt.xlabel('List of city')

plt.ylabel('AQI report')

plt.show()

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()

#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')

#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, 'State')

#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, 'place')

# Heatmap plot diagram

fig, ax = plt.subplots(figsize=(15,10))

s.heatmap(df.corr(), ax=ax, annot=True)

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

avg = df["Avg"]

pol = df["Pollutants"]

city = df["city"]

state = df["State"]

pl = df["place"]

plt.plot(pl, avg, color='g')

plt.xlabel('Average value of pollution')

plt.ylabel('Number of places')

plt.title('Pollutants Details by each places')

plt.show()

def qul\_No\_qul\_bar\_plot(df, bygroup):

dataframe\_by\_Group = p.crosstab(df[bygroup], columns=df["class"], normalize = 'index')

dataframe\_by\_Group = n.round((dataframe\_by\_Group \* 100), decimals=2)

ax = dataframe\_by\_Group.plot.bar(figsize=(10,5));

vals = ax.get\_yticks()

ax.set\_yticklabels(['{:3.0f}%'.format(x) for x in vals]);

ax.set\_xticklabels(dataframe\_by\_Group.index,rotation = 0, fontsize = 15);

ax.set\_title('Air Quality Vs No Air Quality (%) (by ' + dataframe\_by\_Group.index.name + ')\n', fontsize = 15)

ax.set\_xlabel(dataframe\_by\_Group.index.name, fontsize = 12)

ax.set\_ylabel('(%)', fontsize = 12)

ax.legend(loc = 'upper left',bbox\_to\_anchor=(1.0,1.0), fontsize= 12)

rects = ax.patches

# Add Data Labels

for rect in rects:

height = rect.get\_height()

ax.text(rect.get\_x() + rect.get\_width()/2,

height + 2,

str(height)+'%',

ha='center',

va='bottom',

fontsize = 12)

return dataframe\_by\_Group

#preprocessing, split test and dataset, split response variable

X = df.drop(labels='class', axis=1)

#Response variable

y = df.loc[:,'class']

#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)

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))

def accuracy(prediction,actual):

correct = 0

not\_correct = 0

for i in range(len(prediction)):

if prediction[i] == actual[i]:

correct+=1

else:

not\_correct+=1

return (correct\*100)/(correct+not\_correct)

def metrics(prediction,actual):

tp = 0

tn = 0

fp = 0

fn = 0

for i in range(len(prediction)):

if prediction[i] == actual[i] and actual[i]==1:

tp+=1

if prediction[i] == actual[i] and actual[i]==0:

tn+=1

if prediction[i] != actual[i] and actual[i]==0:

fp+=1

if prediction[i] != actual[i] and actual[i]==1:

fn+=1

metrics = {'Precision':(tp/(tp+fp+tn+fn)),'Recall':(tp/(tp+fn)),'F1':(2\*(tp/(tp+fp+tn+fn))\*(tp/(tp+fn)))/((tp/(tp+fp+tn+fn))+(tp/(tp+fn)))}

return (metrics)

#Generic function for making a classification model and accessing performance:

def classification\_model(model, df, predictors, outcome):

model.fit(df[predictors],df[outcome])

predictions = model.predict(df[predictors])

accuracy = metrics.accuracy\_score(predictions,df[outcome])

print ("Accuracy : %s" % "{0:.3%}".format(accuracy))

kf = KFold(df.shape[0], n\_folds=5)

error = []

for train, test in kf:

train\_predictors = (df[predictors].iloc[train,:])

train\_target = df[outcome].iloc[train]

model.fit(train\_predictors, train\_target)

error.append(model.score(df[predictors].iloc[test,:], df[outcome].iloc[test]))

print ("Cross-Validation Score : %s" % "{0:.3%}".format(n.mean(error)))

#Fit the model again so that it can be refered outside the function:

model.fit(df[predictors],df[outcome])

#using Logistic reasoning

outcome\_var = ['class']

predictor\_var = ['Country', 'State', 'city', 'place', 'lastupdate', 'Avg', 'Max', 'Min','Pollutants']

from sklearn.linear\_model import LogisticRegression

from sklearn.cross\_validation import KFold

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn import metrics

from sklearn.model\_selection import train\_test\_split

# LogisticRegression

model = LogisticRegression()

classification\_model(model, df,predictor\_var,outcome\_var)

# RandomForestClassifier

model = RandomForestClassifier()

classification\_model(model, df,predictor\_var,outcome\_var)

# DecisionTreeClassifier

model = DecisionTreeClassifier()

classification\_model(model, df,predictor\_var,outcome\_var)

# SVC

model = SVC()

classification\_model(model, df,predictor\_var,outcome\_var)

# KNN

model = KNeighborsClassifier()

classification\_model(model, df,predictor\_var,outcome\_var)

#drop the data

df.drop(['AQI'], axis=1, inplace=True)

#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

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import SGDClassifier

#We'll conduct a grid search over several hyperparameter choices. The search uses 5-fold cross-validation with stratified folds. The type of linear classifier is chosen with the loss hyperparameter. For a linear SVC we set loss = 'hinge', and for logistic regression we set loss = 'log'.

pipeline\_sgd = Pipeline([

('scaler', StandardScaler(copy=False)),

('model', SGDClassifier(max\_iter=1000, tol=1e-3, random\_state=1, warm\_start=True))

])

#Set the hyperparameter grids to search over, one grid for the linear SVC and one for logistic regression:

param\_grid\_sgd = [{

'model\_\_loss': ['log'],

'model\_\_penalty': ['l1', 'l2'],

'model\_\_alpha': n.logspace(start=-3, stop=3, num=20)

}, {

'model\_\_loss': ['hinge'],

'model\_\_alpha': n.logspace(start=-3, stop=3, num=20),

'model\_\_class\_weight': [None, 'balanced']

}]

#The grid search, implemented by GridSearchCV, uses StratifiedKFold with 5 folds for the train/validation splits. We'll use matthews\_corrcoef (the Matthews correlation coefficient, MCC) as our scoring metric.

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import make\_scorer, matthews\_corrcoef

MCC\_scorer = make\_scorer(matthews\_corrcoef)

grid\_sgd = GridSearchCV(estimator=pipeline\_sgd, param\_grid=param\_grid\_sgd, scoring=MCC\_scorer, n\_jobs=-1, pre\_dispatch='2\*n\_jobs', cv=5, verbose=1, return\_train\_score=False)

X = df.drop(labels='class', axis=1)

#Response variable

y = df.loc[:,'class']

del df

#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)

#for our convienient we delete X,y variable for differentiate confusion

del X, y

# Prevent view warnings

X\_train.is\_copy = False

X\_test.is\_copy = False

#Perform the grid search:

import warnings

with warnings.catch\_warnings(): # Suppress warnings from the matthews\_corrcoef function

warnings.simplefilter("ignore")

grid\_sgd.fit(X\_train, y\_train)

#Next we'll try a random forest model, implemented in RandomForestClassifier

from sklearn.ensemble import RandomForestClassifier

#We do not need to rescale the data for tree-based models, so our pipeline will simply consist of the random forest model. We'll leave the pipeline implementation in place in case we want to add preprocessing steps in the future.

pipeline\_rf = Pipeline([

('model', RandomForestClassifier(n\_jobs=-1, random\_state=1))

])

#The random forest takes much longer to train on this fairly large dataset, so we don't actually do a hyperparameter grid search, only specifiying the number of estimators. We'll leave the grid search implemented in case we decide to try different hyperparameter values in the future.

param\_grid\_rf = {'model\_\_n\_estimators': [75]}

grid\_rf = GridSearchCV(estimator=pipeline\_rf, param\_grid=param\_grid\_rf, scoring=MCC\_scorer, n\_jobs=-1, pre\_dispatch='2\*n\_jobs', cv=5, verbose=1, return\_train\_score=False)

#Perform the grid search:

grid\_rf.fit(X\_train, y\_train)

from sklearn.linear\_model import LogisticRegression

logR= LogisticRegression()

logR.fit(X\_train,y\_train)

predictR = logR.predict(X\_test)

print(classification\_report(y\_test,predictR))

print("")

print(confusion\_matrix(y\_test,predictR))

from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier()

dtree.fit(X\_train, y\_train)

predictDT = dtree.predict(X\_test)

print(confusion\_matrix(y\_test,predictDT))

print("")

print(classification\_report(y\_test,predictDT))

from sklearn.svm import SVC

s = SVC()

s.fit(X\_train, y\_train)

predictSV = s.predict(X\_test)

print(confusion\_matrix(y\_test,predictSV))

print("")

print(classification\_report(y\_test,predictSV))

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()

rf.fit(X\_train, y\_train)

predictrf = rf.predict(X\_test)

print(confusion\_matrix(y\_test,predictrf))

print("")

print(classification\_report(y\_test,predictrf))

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()

knn.fit(X\_train, y\_train)

predictknn = knn.predict(X\_test)

print(confusion\_matrix(y\_test,predictknn))

print("")

print(classification\_report(y\_test,predictknn))

**RESULT AND DISCUSSION**

Precision is the fraction of relevant instances among the retrieved instances, while recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. Precision and Recall are used as a measurement of the relevance.

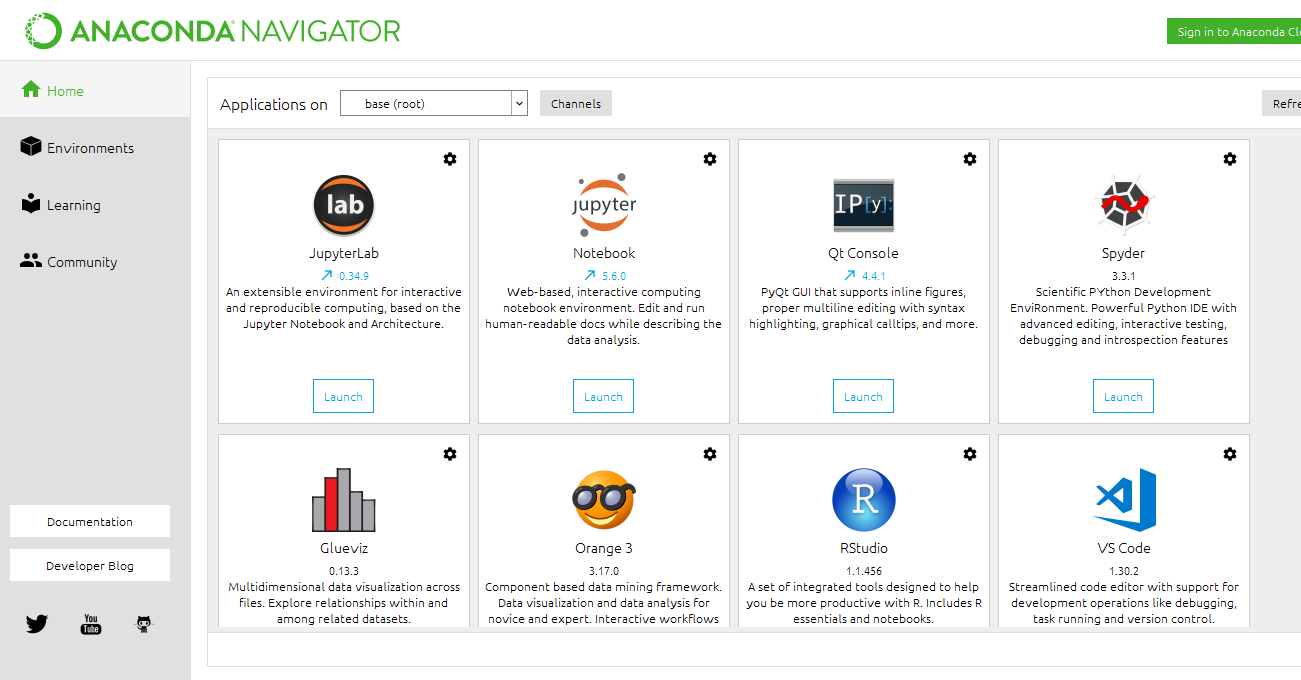
****

Fig: Open the anaconda navigator

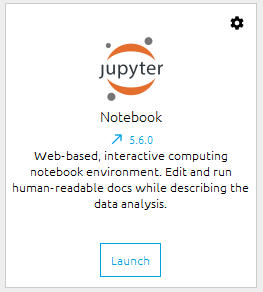
****

Fig: Launch the jupyter notebook platform

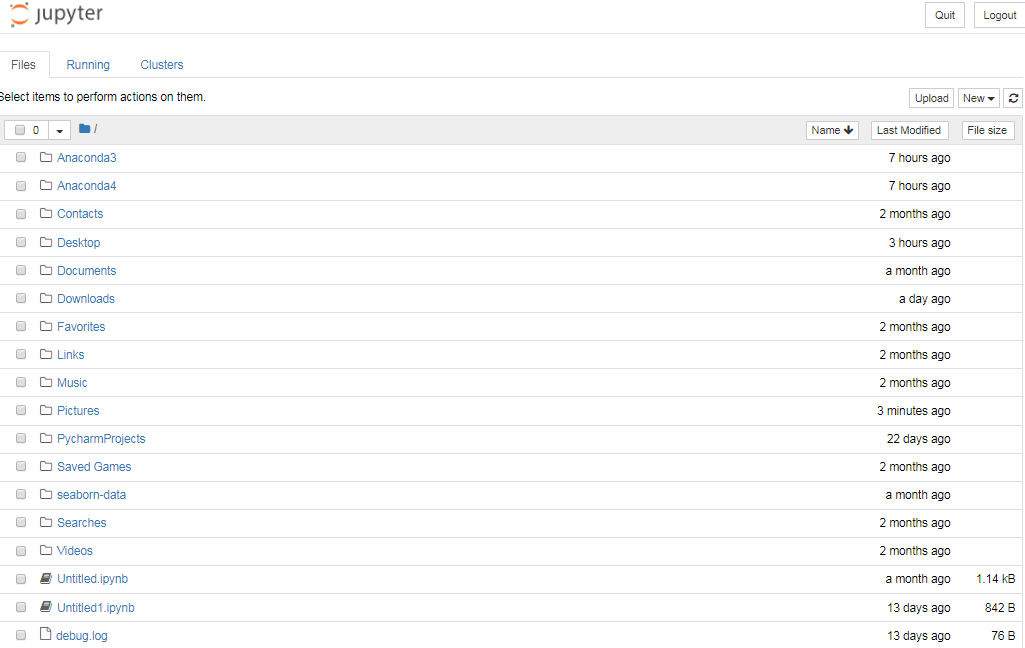
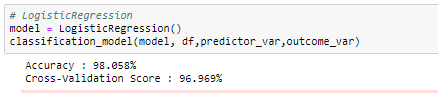
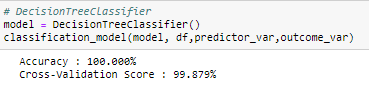
****

Fig: Open the correspondent result folder

**Logistic regression Accuracy Result:**

****

**Decision tree Accuracy Result:**

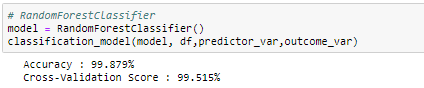


**Support vector classifier Accuracy Result:**

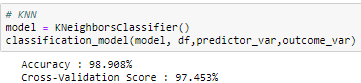




**Random forest Result:**



**K Nearest Neighbor Accuracy Result:**



**Comparison result for best Accuracy**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Precision** | | **Recall** | | **F1-Score** | | **Cross Validation** | **Accuracy (100%)** |
| Class 0 | Class 1 | Class 0 | Class 1 | Class 0 | Class 1 |
| **DT** | 1 | 1 | 1 | 1 | 1 | 1 | 99.87 | 100 |
| **SVC** | 0 | 0.71 | 0 | 1 | 0 | 0.83 | 70.62 | 100 |
| **LR** | 0.97 | 0.99 | 0.97 | 0.99 | 0.97 | 0.99 | 96.96 | 98.05 |
| **KNN** | 0.96 | 0.99 | 0.97 | 0.98 | 0.97 | 0.99 | 97.45 | 98.90 |
| **RF** | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 99.51 | 99.87 |

**CONCLUSION AND FUTURE WORK**

**Conclusion**

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 of decision tree algorithm method by accuracy with classification report. This application can help India meteorological department in predicting the future of air quality and its status and depends on that they can take action.

**Future Work**

* India meteorological department wants to automate the detecting the air quality is good or not from eligibility process (real time).
* To automate this process by show the prediction result in web application or desktop application.
* To optimize the work to implement in Artificial Intelligence environment.

# REFERENCES

[1] Srividya K. Bansal, Sebastian Kagemann, “Integrating Big Data: A

Semantic Extract- Transform-Load Framework,” Computer, Vol. 48, No.

3, pp. 42-50, Mar. 2015

[2] Gautam Pant, Filippo Menczer, “MySpiders: Evolve Your Own

Intelligent Web Crawlers,” Autonomous Agents and Multi-Agent

Systems, Vol. 5, No. 2, pp 221–229, June 2002.

[3] Yung-Sheng Lin, Yu-Hsiang Chang, Yue-Shan Chang, “Constructing

PM2.5 Map Based on Mobile PM2.5 Sensor and Cloud Platform,” 2016

IEEE International Conference on Computer and Information

Technology (CIT), 8-10 Dec. 2016, pp. 702-707.

[4] Feng-Wen Chen, Chen-Wuing Liu, “Estimation of the spatial rainfall

distribution using inverse distance weighting (IDW) in the middle of

Taiwan,” Paddy and Water Environment, September 2012, Vol. 10, No.

3, pp 209–222.

[5] Hakkani-Tür, D., Tur, G., Celikyilmaz, A., Chen, Y., Gao, J., Deng, L.,

Wang, Y. (2016) Multi-Domain Joint Semantic Frame Parsing Using Bi-

Directional RNN-LSTM. Proc. Interspeech 2016, 715-719.