**Comparative Analysis of Air Quality Index using Machine Learning and Large Language Models**

A PROJECT REPORT

*Submitted by*

**V. Chandra Shekar Reddy [Reg No: RA2011003011180]**

### G Sai Harish [Reg No:RA2011003011185]

*Under the Guidance of*

### Dr. S. Shanmugam

**DEPARTMENT OF COMPUTING TECHNOLOGIES COLLEGE OF ENGINEERING AND TECHNOLOGY SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR– 603 203**

**January 2023**



# **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

# **KATTANKULATHUR–603 203**

## BONAFIDE CERTIFICATE

Certified that 18CSP107L project report titled “**Comparative Analysis of Air Quality Index using Machine Learning and Large Language Models** ” is the bonafide work of **Chandra Shekar Reddy [RegNo:RA2011003011180]** and **G. Sai Harish [RegNo:RA2011003011185]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

### Dr. S. Shanmugam

**SUPERVISOR**

Associate Professor

Department of Computing Technologies

###### 

###### **PANEL HEAD**

Associate Professor

Department of Computing Technologies

###### **Dr. M. PUSHPALATHA**

**HEAD OF THE DEPARTMENT**

Department of Computing Technologies



Department of Computing Technologies

### SRM Institute of Science and Technology

### Own Work Declaration Form

**Degree/Course :** B. Tech in Computer Science and Engineering

S**tudent Names :** Sai Harish, Chandra Shekar Reddy

**Registration Number :** RA2011003011185, RA2011003011180

**Title of Work :** Comparative Analysis of Air Quality Index using Machine Learning and Large Language Models

I/We here by certify that this assessment compiles with the University’s Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is our own except where indicated, and that we have met the following conditions:

* Clearly references / listed all sources as appropriate
* Referenced and put in inverted commas all quoted text (from books, web, etc.)
* Given the sources of all pictures, data etc that are not my own.
* Not made any use of the report(s) or essay(s) of any other student(s)either past or present
* Acknowledged in appropriate places any help that I have received from others (e.g fellow students, technicians, statisticians, external sources)
* Compiled with any other plagiarism criteria specified in the Course hand book / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

|  |
| --- |
| **DECLARATION:** |
| I am aware of and understand the University’s policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.  **Student 1 Signature:**  **Student 2 Signature:**  **Date:** |
| If you are working in a group, please write your registration numbers and sign with the date for every student in your group. |

## ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology,

**Dr. T. V. Gopal**, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, **Dr. M. Pushpalatha,** Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We register our immeasurable thanks to our Faculty Advisor, **Dr. M. Rajalakshmi** , Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr. S. Shanmugam,** Associate Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his mentorship. He provided us with the freedom and support to explore the research topics of our interest. His passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff and students of Computing Technologies Department, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement.

**Gokavarapu Sai Harish [RA2011003011185]**

**V. Chandra Shekar Reddy [RA2011003011180]**

## TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE** NO |
|  | **ABSTRACT** | 1 |
|  | **LIST OF TABLES** | 2 |
|  | **LIST OF FIGURES** | 3 |
|  | **LIST OF ABBREVIATIONS** | 4 |
| **1** | **INTRODUCTION** | 5-7 |
| **2** | **LITERATURE SURVEY** | 8-14 |
| **3** | **APPROACH FOR IMPLEMENTATION** | 14-16 |
| **4** | **METHODOLOGY** | 17-18 |
| **5** | **SYSTEM ARCHITECTURE** | 19-20 |
| **6** | **SYSTEM REQUIREMENT** | 21-22 |
| **7** | **DATASETS** | 23-24 |
| **8** | **CODING IMPLEMENTATION** | 25-29 |
| **9** | **RESULT** | 30-31 |
| **10** | **ANALYSIS** |  |
| **11** | **CONCLUSION AND FUTURE SCOPE** | 32-33 |
|  | **REFERENCES** | 34-35 |

**ABSTRACT**

Air pollution is a critical environmental challenge, characterized by the release of harmful substances into the atmosphere, posing a severe threat to human health and the overall well-being of the planet. The consequences of air pollution extend beyond affecting humans; its detrimental impact is felt on animals, crops, and ecosystems such as forests. The pollutants, including particulate matter, ozone, sulfur dioxide, and nitrogen oxides, originate from various sources like industrial activities, vehicular emissions, and natural processes.

Recognizing the urgency of addressing this global issue, researchers have turned to innovative approaches, with a particular focus on leveraging machine learning techniques for predicting air quality in the transport sector. These predictive models analyze historical and real-time data to assess the concentration of pollutants and anticipate future air quality levels. By harnessing the power of machine learning algorithms, scientists aim to enhance the accuracy of air quality predictions, enabling timely interventions and preventive measures.

The intersection of machine learning and air quality assessment has evolved into a significant research area, contributing to the development of smart systems capable of providing reliable forecasts and valuable insights. These advancements not only aid in understanding the patterns and dynamics of air pollution but also empower policymakers and environmentalists to implement targeted strategies for mitigating the adverse effects of air pollution on human health and the environment. As society continues to grapple with the complex challenges posed by air pollution, the integration of machine learning technologies offers a promising avenue for creating sustainable solutions and fostering a healthier planet.

**LIST OF TABLES**

###### Literature Survey 8

## LIST OF FIGURES

## 4.3.2 Training the dataset for 50 epoch and each epoch 417 images 13

* 1. Architecture diagram 17
  2. System Overview 19
  3. Using Text blob and Nltk modules 23
  4. Bigrams and sent phrase net 23
  5. Prediction of words 27
  6. After training completion 27
  7. Finding the occurrences of the pretrained words 28
  8. Polarity is being calculated for the words and incremented the value 29
  9. Code for the testing 33
  10. partial result code 38
  11. partial result 39
  12. result 1 39
  13. result 2 41

**LIST OF SYMBOLS AND ABBREVIATIONS**

|  |  |
| --- | --- |
| **CONV2D** | Convolutional 2D layer |
| **ReLU** | Rectified Linear Unit |
| **ANN** | Artificial Neural Network |
| **CNN** | Convolutional Neural Network |
| **ML** | Machine Learning |
| **DL** | Deep Learning |
| **Opencv** | opean source computer vision |
| **St** | Streamlit |
| **Np** | numpy |
| **DB** | Data Base |
| **GUI** | Graphical User Interface |
| **FER** | Face Emotion Recognition |

**CHAPTER- 1**

* 1. **Introduction**

Machine learning is to predict the future from past data. Computer studying (ML) with LLM Models is a style of artificial intelligence (AI) that delivers computers the capability to gain knowledge of without being explicitly programmed. Machine finding out makes a specialty of the progress of pc applications that can alternate when exposed to new information and the basics of laptop studying, implementation of a easy laptop finding out algorithm utilizing python. Process of coaching and prediction involves use of specialized algorithm. It feed the training data to an algorithm, and the algorithm uses this training knowledge to offer predictions on a brand new test information.

**1.2 Problem Statement**

The challenge lies in the evolving complexity of calculating the Air Quality Index (AQI), with multiple algorithms in play. These algorithms vary in accuracy and other parameters, creating a problem of inconsistency. The lack of standardization hampers effective air quality management, leading to confusion among stakeholders. The problem statement emphasizes the need for a unified approach to AQI calculation, considering diverse stakeholder needs. Collaborative efforts are required to establish consensus on reliable algorithms and parameters, developing guidelines for consistent air quality assessments. This standardization is crucial for informed decision-making, ensuring a cohesive response to mitigate the impacts of air pollution on public health and the environment.

**1.3 Objective of the Project**

The current algorithms employed for air quality prediction exhibit certain limitations, particularly in their ability to accurately forecast extreme points, namely the maximum and minimum pollution levels. This deficiency results in a restricted capacity to predict peak pollution events or periods of exceptionally clean air. Consequently, the algorithms face challenges in determining accurate cut-offs, which are critical for establishing thresholds and guidelines for acceptable air quality levels.

The presence of complex mathematical calculations further complicates the algorithms' performance. These intricate computations can slow down the prediction process and potentially introduce errors, diminishing the overall reliability of the models. Simplifying and optimizing these calculations is essential for enhancing the efficiency and speed of air quality predictions.

In summary, the existing algorithms for air quality prediction face challenges in accurately predicting extreme pollution points, determining cut-offs, employing efficient prediction approaches, dealing with complex mathematical calculations, and appropriately handling the temporal aspect of data. Overcoming these limitations is crucial for advancing the precision and reliability of air quality predictions, thereby facilitating more effective environmental management and public health protection.

**1.4 Motivation**

Metropolitan cities like Delhi, Hyderabad, Mumbai and some big cities and places like Agra, Kanpur and places in the North India and foothills of the Himalayan terrain are reported to have very poor Air quality index.

With an average ***PM2.5***value of ***150+*** they are in red alert of air pollution.

In such places, pure air is a blessing for the people.

The primary motivation is to ensure the circulation of fresh air.

Reduction in the risk of impacts of a polluted air.

Requirement for an easy air purifier recommendation.

Fitting in budget and reducing health hazards from pollutions.

Increase in profit of e-commerce partners like Amazon and Flipkart from where purifiers are most likely to get ordered.

**1.5 Challenges in Existing System**

Urban air pollutant attention forecast is coping with a surge of large ecological monitoring data and intricate alterations in air pollution. This necessitates effective estimating methods to strengthen prediction accuracy and avoid grave contamination episodes, thereby improving ecological administration resolution-making capacity.

A brand-new contaminant concentration estimation process is established on sizeable amounts of ecological knowledge and deep learning approaches. This integrates colossal data using two forms of deep networks.They provide limited accuracy as they are unable to predict the extreme points i.e. the pollution maximum and minimum. Cut-offs cannot be determined using such approach. They use inefficient approach for better output prediction

**CHAPTER - 2**

**2.1 LITERATURE SURVEY**

Air pollution stands as a formidable threat to both the environment and public health, necessitating vigilant monitoring to understand and mitigate its adverse impacts. Traditional air quality monitoring stations, although invaluable, are constrained by their high costs and limited spatial coverage. These stations often offer a restricted and somewhat incomplete portrayal of air quality within a city, neglecting localized variations that may be crucial for a comprehensive understanding. The objective is to explore how these technologies contribute to overcoming the limitations of traditional monitoring approaches and provide a more nuanced understanding of air pollution patterns.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SNo** | **Title** **of** **the paper** | **Journal** | **Authors** | **merits** | **Demerits** | **Brief** **description** |
| 1 | Low Cost Sensor With IoT LoRaWAN Connectivity and Machine Learning-Based Calibration for Air Pollution Monitoring | IEEE xplorer 2020 | Sharafat Ali; Tyrel Glass; Baden Parr; Johan Potgieter |  |  | . Air pollution poses significant risk to environment and health. Air quality monitoring stations are often confined to a small number of locations due to the high cost of the monitoring equipment. They provide a low fidelity picture of the air quality in the city; local variations are overlooked. However, recent developments in low-cost sensor technology and wireless communication systems like Internet of Things (IoT) provide an opportunity to use arrayed sensor networks to measure air pollution, in real time, at a large number of locations |
| 2 | Selective Detection of VOCs With WO3 Nanoplates-Based Single Chemiresistive Sensor Device Using Machine Learning Algorithms | IEEE xplorer 2020 | Snehanjan Acharyya; Sudip Nag; Prasanta Kumar Guha |  |  | . The paper presents the integration of single metal-oxide based chemiresistive sensor device and machine learning tools for selective discrimination of different volatile organic compounds (VOCs) for indoor air quality monitoring applications. Tungsten oxide (WO 3 ) nanoplates has been employed as the gas sensing material which were obtained by acidification followed by low temperature hydrothermal process. |
| 3 | Graph Neural Network for Air Quality Prediction: A Case Study in Madrid | IEEE xplorer 2023 | Ditsuhi Iskandaryan, Francisco Ramos, Sergio Trilles |  |  | Air quality monitoring, modelling and forecasting are considered pressing and challenging topics for citizens and decision-makers, including the government. The tools used to achieve the above goals vary depending on the opportunities provided by technological development. Much attention is currently being paid to machine learning and deep learning methods, which, compared to domain knowledge methods, often perform better in terms of capturing, computing and processing multidimensional information and complex dependencies. |
| 4 | Air Quality Index Forecasting via Genetic Algorithm-Based Improved Extreme Learning Machine | IEEE xplorer 2023 | Chunhao Liu,   Guangyuan Pan, Dongming Song,  Hao Wei |  |  | Air quality has always been one of the most important environmental concerns for the general public and society. Using machine learning algorithms for Air Quality Index (AQI) prediction is helpful for the analysis of future air quality trends from a macro perspective. When conventionally using a single machine learning model to predict air quality, it is challenging to achieve a good prediction outcome under various AQI fluctuation trends. In order to effectively address this problem, a genetic algorithm-based improved extreme learning machine prediction method is enhanced. |
| 5 | A Sequence-to-Sequence Air Quality Predictor Based on the n-Step Recurrent Prediction | IEEE Xplorer 2019 | Bo Liu,  Shuo Yan,  Jianqiang Li,  Guangzhi Qu, Yong Li, Jianlei Lang, Rentao Gu |  |  | Increasingly, more people are suffering from the effects of air pollution. This study took Beijing as an example and proposed an attention-based air quality predictor (AAQP) that could better protect people from air pollution. The AAQP is a seq2seq model, and it exploits historical air quality data and weather data to predict future air quality indexes. Although existing research has promoted seq2seq for air quality prediction, there are still two problems |
| 6 | A Predictive Data Feature Exploration-Based Air Quality Prediction Approach | IEEE xplorer 2019 | Ying Zhang, Yanhao Wang, Minghe Gao, Qunfei Ma |  |  | People have been paying more and more attention to air quality because it directly affects people's health and daily life. Effective air quality prediction has become one of the hot research issues. This paper is suffering many challenges, such as the instability of data sources and the variation of pollutant concentration along time series. Aiming at this problem, we propose an improved air quality prediction method based on the LightGBM model to predict the PM2.5 concentration at the 35 air quality monitoring stations in Beijing over the next 24th |
| 7 | Revealing Influence of Meteorological Conditions on Air Quality Prediction Using Explainable Deep Learning | IEEE xplorer 2022 | Yuting Yang,  Gang Mei,  Stefano Izzo |  |  | Meteorological conditions have a strong influence on air quality and can play an important role in air quality prediction. However, due to the “black-box” nature of deep learning, it is difficult to obtain trustworthy deep learning models when considering meteorological conditions in air quality prediction. To address the above problem, in this paper, we reveal the influence of meteorological conditions on air quality prediction by utilizing explainable deep learning. |
| 8 | Innovative Spatial-Temporal Network Modeling and Analysis Method of Air Quality | IEEE Xplorer 2019 | Guyu Zhao,  Guoyan Huang,  Hongdou He,  Qian Wang |  |  | Air quality system is characterized by dynamism, dependency, and complexity. Scientifically representing the internal structure of air mass distribution and its relationship to reveal the dynamic evolution of air quality is the key to solve the air pollution problem. This paper abstracts the air quality system into the complex network innovatively by synthesizing spatial and temporal factors influencing air quality status. Based on quantifying the regional dynamic interconnection and interaction, our modeling approach is proposed to mine the relationship of different regions. |
| 9 | Air Quality Prediction Based on Integrated Dual LSTM Model | IEEE xplorer 2021 | Hongqian Chen, Mengxi Guan, Hui Li |  |  | Air quality prediction is an important reference for meteorological forecast and air controlling, but over fitting often occurs in prediction algorithms based on a single model. Aiming at the complexity of air quality prediction, a prediction method based on integrated dual LSTM (Long Short-Term Memory) model was proposed in this paper. Firstly, the Seq2Seq (Sequence to Sequence) technology is used to establish a single-factor prediction model which can obtain the predicted value of each component in air quality data, independently. |
| 10 | Real Time Localized Air Quality Monitoring and Prediction Through Mobile and Fixed IoT Sensing Network | IEEE xplorer 2020 | Dan Zhang,  Simon S. Woo |  |  | Air pollution and its harm to human health has become a serious problem in many cities around the world. In recent years, research interests in measuring and predicting the quality of air around people has spiked. Since the Internet of Things (IoT) has been widely used in different domains to improve the quality life for people by connecting multiple sensors in different places, it also makes the air pollution monitoring more easier than before. |

**Table: 2.1 Literature survey**

**CHAPTER-3**

**Approach for Implementation**

**3.1 Data collection and Data preprocessing**

Gather the premeasured data from diverse sources, including Kaggle and other social media websites, news articles to ensure a representative dataset.

Clean and preprocess the collected text, which includes null parameters removal, correlation, and specific processing for calculation of AQI.

**3.2 Calculation index parameter of each gas**

The index of each element will be calculated based upon the formula to categorize the element based upon the concentration and which effects the index value. The index value plays the role finding the accuracy and aqi for the dataset model and the prediction of output will effect the final result.

**3.3 Calculation of Air Quality Index (AQI)**

The described process involves categorizing elements based on concentration, using a calculated index value that significantly impacts the accuracy of a dataset model, the determination of Air Qualiy Index (AQI), and the overall prediction of the model's output. The accuracy of the index calculations is essential for obtaining reliable and meaningful results in the context of the dataset.

**3.4 Removing of Correlation**

Correlation is a statistical measure that expresses the extent to which two variables are linearly related (meaning they change together at a constant rate). It's a common tool for describing simple relationships without making a statement about cause and effect.

So after removal of correlation there are possible chances of increase in the accuracy of the results since the duplicates and accuracy affecting factors are removed.

**3.5 Machine learning Models**

After preparing the cleaned dataset, the subsequent step involves leveraging various Machine Learning regression algorithms to extract meaningful results. These algorithms, ranging from linear regression to more complex models, are applied to the dataset to identify patterns and relationships within the data. By systematically analyzing the cleaned dataset, the machine learning regression process aims to generate predictive models capable of estimating numerical outcomes. This enables the extraction of valuable insights, predictions, and trends from the data, contributing to informed decision-making and improved understanding of the underlying factors influencing the dataset.

.

**3.6 LLM Models**

Following the application of the Linear Regression model to the consistent dataset, the subsequent phase involves utilizing the model's predictions to calculate the Air Quality Index (AQI). This calculated AQI serves as a benchmark for assessing the model's efficacy in predicting air quality. The comparison of AQI results between the Linear Regression model and another model, presumably another regression algorithm, forms the next step in the process. This comparative analysis aims to evaluate the performance and accuracy of both models, providing valuable insights into their respective strengths and weaknesses. Ultimately, such comparisons contribute to refining and selecting the most effective model for predicting and understanding air quality based on the given dataset.

**CHAPTER-4**

**Methodology**

* 1. **Algorithms Used**

The following Algorithms are used to perform the machine learning analysis of the air quality Index.

**Polynomial Regression, Logistic Regression, Linear Regression, Step-Wise Regression, Ridge Regression**

**4.1.1 Linear Regression:**

Linear Regression is a fundamental algorithm used for predicting a continuous outcome based on linear relationships between independent and dependent variables. In the context of air quality monitoring, it establishes a linear model to predict numerical values, making it suitable for regression tasks.

**4.1.2 Polynomial Regression:**

Polynomial Regression extends Linear Regression by allowing the relationship between variables to be polynomial, capturing more complex patterns. In air quality analysis, it accommodates non-linear variations, providing a more flexible model that can better represent the intricacies of pollution factors.

**4.1.3 Logistic Regression:**

Despite its name, Logistic Regression is primarily employed for binary classification tasks. In air quality applications, it could be used to predict categorical outcomes, such as classifying air quality as 'acceptable' or 'unhealthy,' based on given features.

**4.1.4 Step-Wise Regression:**

Step-Wise Regression is a method that iteratively adds or removes variables from a model to identify the most influential predictors. In air quality modeling, this approach can help refine the model by selecting the most relevant features, enhancing predictive accuracy.

**4.1.5 Ridge Regression:**

Ridge Regression is a regularization technique that addresses multicollinearity by adding a penalty term to the linear regression cost function. In air quality prediction, Ridge Regression can prevent overfitting and improve generalization by mitigating the impact of highly correlated variables.

* 1. **Modules Description**

4.2.1 Gather a Regression Dataset

Here data will be extracted and filtered using numpy and pandas , sklearn libraries

The special charcters and emoji’s will be and Phrases will be formed.

4.2.2Remove the correlation

The cleaned dataframe will be identified and null values from the dataframe and made available for the further process to calculate the index values of gases and dataframes.

* + 1. Calculation of AQI

Here the testing will be done using naïve bayes algorithm. It is the final step to identify these and classify the pharse Whether it is Postive or Negative or Neutral

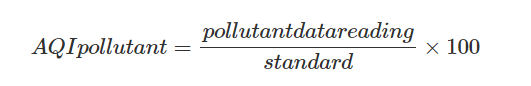




Fig4.2.3 AQI Pollutant will be calculated

**CHAPTER -5**

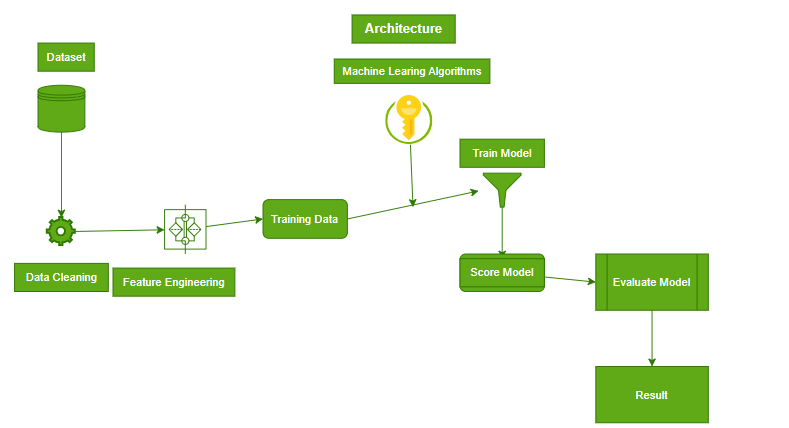
**Architecture Diagram**

Fig: 5.1 Architecture diagram of Machine Learning

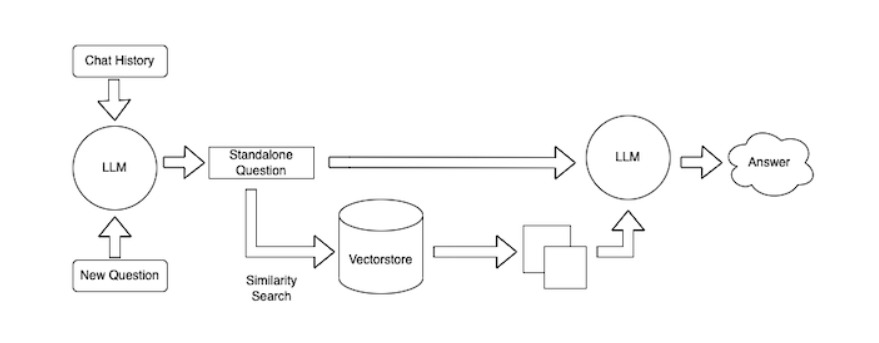


Fig: 5.2 Architecture diagram of LLM

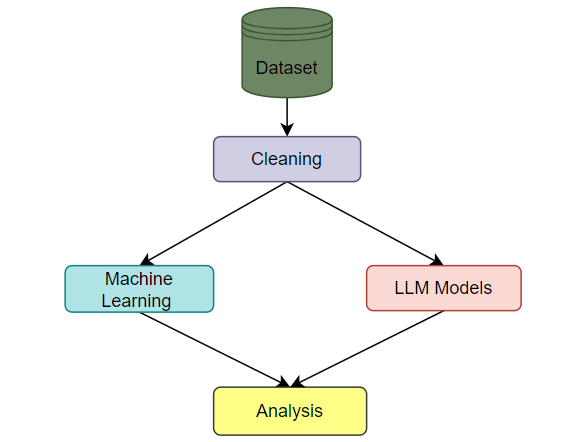
**SYSTEM OVERVIEW**

Fig: 5.3 system overview

**CHAPTER-6**

**SYSTEM REQUIRMENTS**

System requirements refer to the description of capabilities, specifications and conditions that a system software applications or product needs to meet to fulfill certain functions and achieve desired outcomes. These requirements serve as the foundation for the design, development, and evolution of a system.

### Functional Requirements:

Functional Requirements define specific functions of the software system and its behavior under specific inputs or conditions. For predicting facial emotion results, the following functional requirements are essential:

###### **Software system Configuration**

* + - * Operating System: Windows 11
      * Coding Language: Python
      * IDE: Python IDE
      * Platform: VS CODE

###### **Database**

* + - * Data

### Non-Functional Requirements:

Non-functional requirements are criteria used to judge the system's operation. These

include:

1. **Performance Requirements:** The system should respond promptly and efficiently, providing accurate predictions.
2. **Design Requirements:** The user interface should be intuitive and easy to use, eliminating the need for complex user interactions.
3. **Security Constraints:** Data security and user privacy should be maintained, adhering to industry standards.
4. **Operational Requirements:** The system should be operational across various platforms, Guaranteeing inclusivity for a broad spectrum of users.

The following are minimum requirements to develop this application

###### **Hardware requirements**

* + System : 2.4 GHz
  + RAM : 3GB
  + Processor : 2 GHz
  + Hard Disk : 40 GB
  + Floppy Drive : 1.44 Mb
  + Monitor : 15 VGA Colour
  + Mouse : Logitech
  + Ram : 512 Mb

**CHAPTER – 7**

17

**DATASET**

Dataset is the Heart of any project. For this project too the dataset is the main crucial step which will be learned from the above dataset and gives the prescribed output for the requirements. The dataset was created for the purpose of training and evaluating machine learning models to detect sentiment of the Telugu words.

The dataset have the following Parameters

**7.1 State and location**

The raw data will be passed to this package and having the modules to know the locations and having different values based upon location, categorization. Developers use this dataset to develop and evaluate machine learning model and algorithm for accurately recognizing these Analysis.

**7.2 Sampling date and type**

The data and type of the dataset will play a crucial role in some analysis if it is a recent dataset and even favors to make so many changes in the society based on the type of location it is and have a look at the alternatives to avoid the concentration of the pollutants.

**7.3 So2, No2, RSPM, SPM, PM2.5:**

These are the main parameters which takes place a lead role in the analysis. The AQI will be calculated using these values only. These concentrations based upon different location will play a different role in the analysis. These are pollutants Sulfur Dioxide(So2), Nitrogen Dioxide(No2), Respirable Suspended Particulate Matter, Suspended Particulate matter, Particulate Matter of 2.5 Microns. The pollutant values favors in calculating AQI and this AQI decides the quality of air in the environment based upon different locations and types of places.

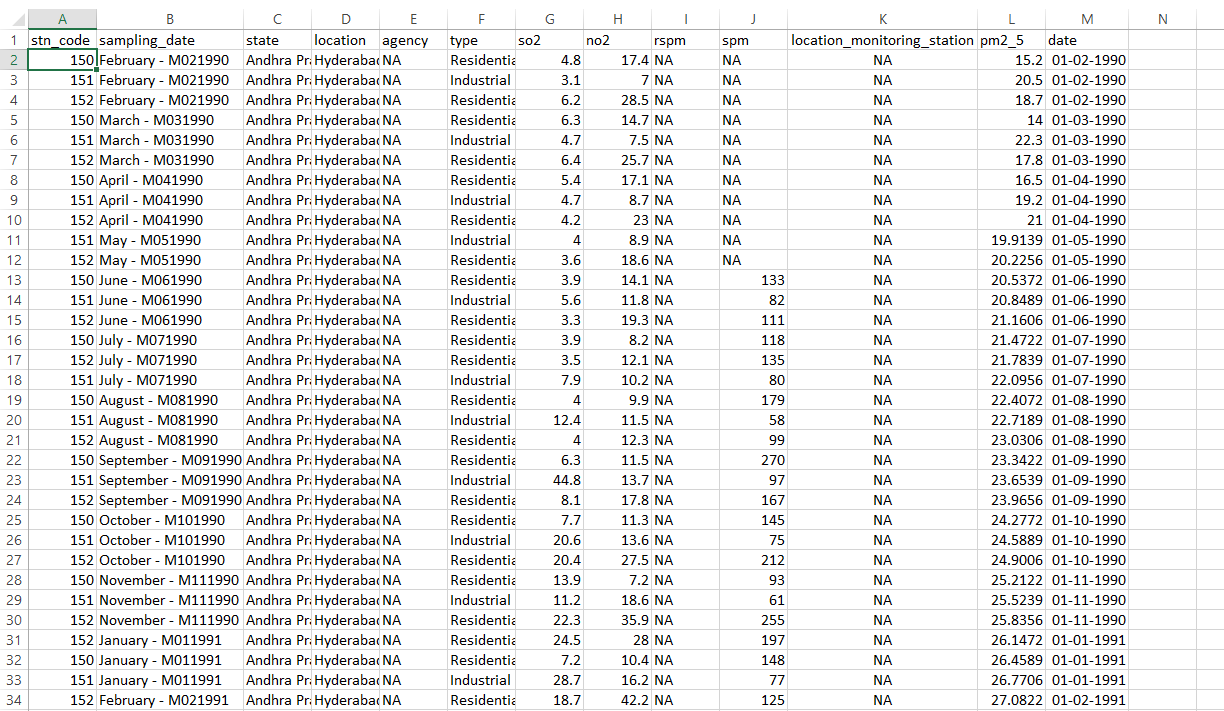


Fig: 7.1 The Raw Dataset

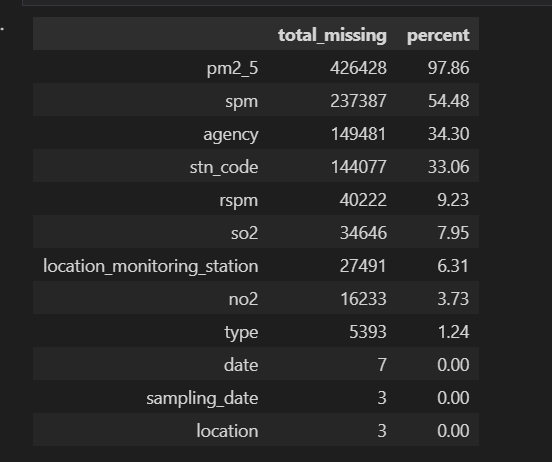
****

Fig: 7.2. Summary Before cleaning Dataset

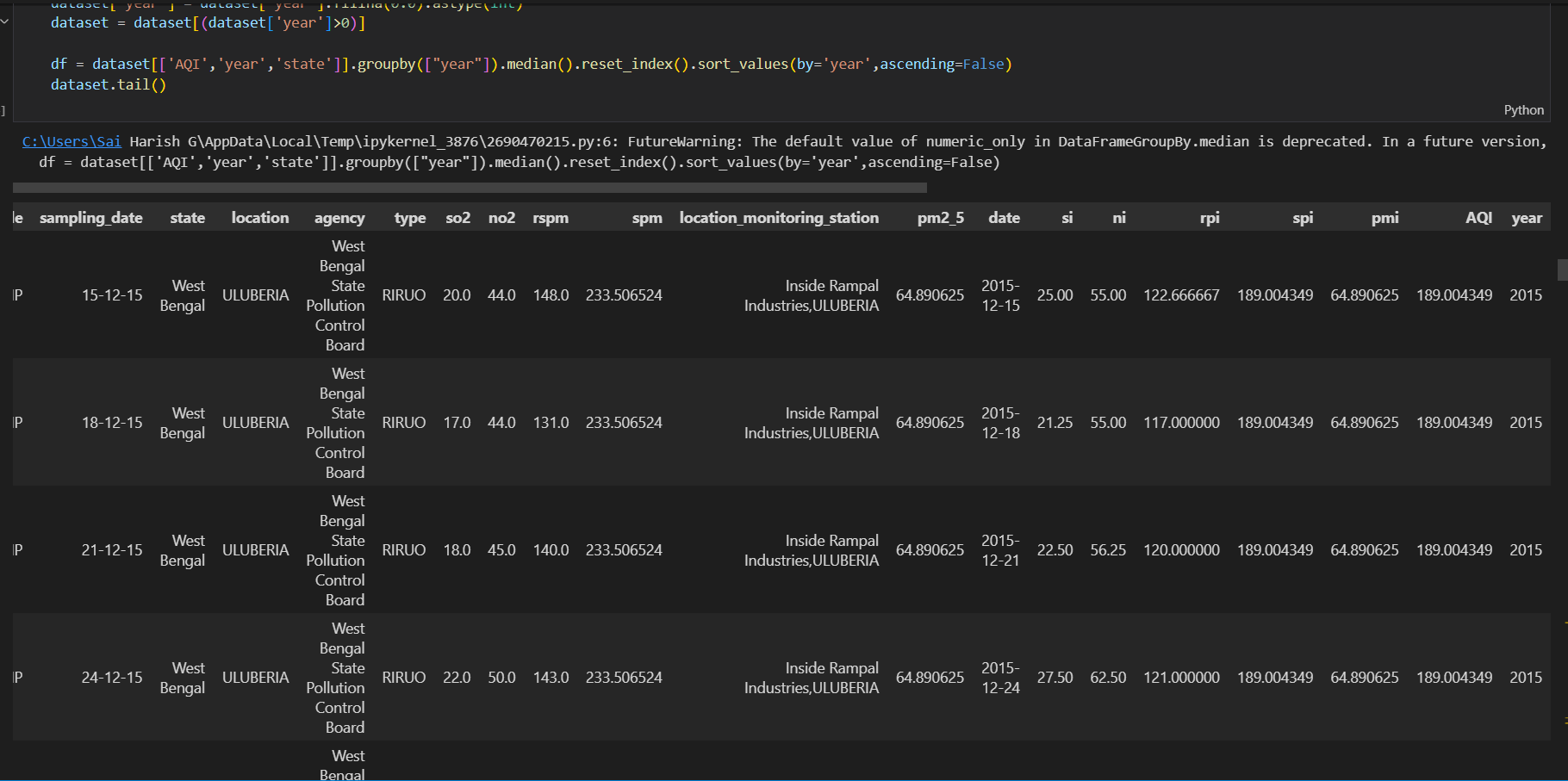
****

Fig: 7.3 Cleaned Dataset

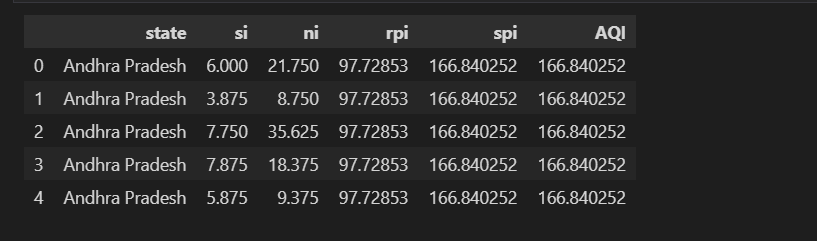


Fig 7.4 The dataset for Machine Learning Model

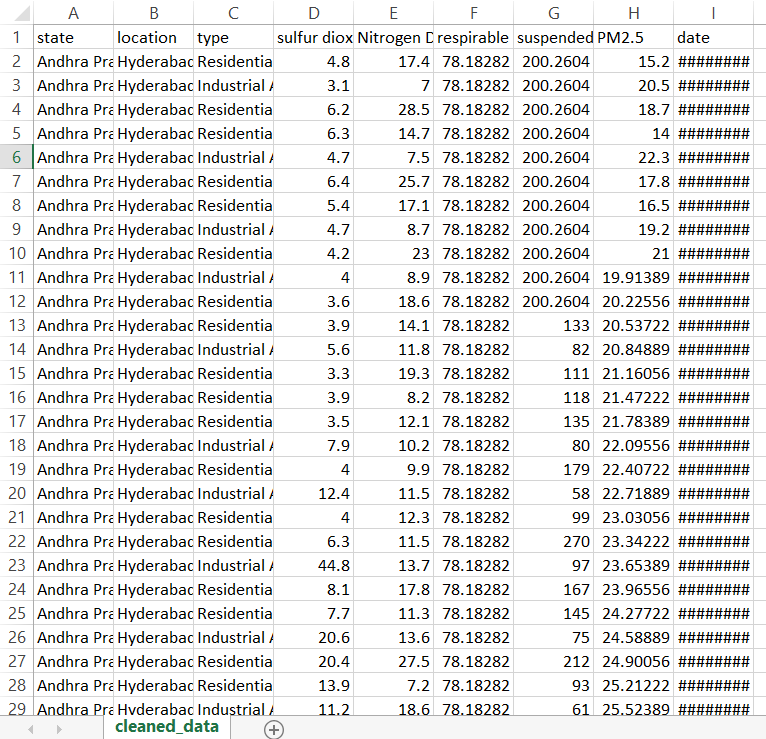
****

Fig 7.5 The dataset for Large Language Model

**CHAPTER-8**

**CODE IMPLEMENTATION**

**8.1 Briefing and cleaning of dataset:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, KFold

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

from sklearn.metrics import r2\_score, mean\_squared\_error

import statsmodels.api as sm

dataset=pd.read\_csv('data.csv',encoding="ISO-8859-1")

dataset.describe()

dataset.tail()

print(dataset.info())

dataset.head()

total = dataset.isnull().sum()[dataset.isnull().sum() != 0].sort\_values(ascending = False)

percent = pd.Series(round(total/len(dataset)\*100,2))

pd.concat([total, percent], axis=1, keys=['total\_missing', 'percent'])

def remove\_outlier(df\_in, col\_name):

    q1 = df\_in[col\_name].quantile(0.25)

    q3 = df\_in[col\_name].quantile(0.75)

    iqr = q3-q1 #Interquartile range

    fence\_low  = q1-1.5\*iqr

    fence\_high = q3+1.5\*iqr

    df\_out = df\_in.loc[(df\_in[col\_name] > fence\_low) & (df\_in[col\_name] < fence\_high)]

    #return df\_out

remove\_outlier(dataset,'so2')

remove\_outlier(dataset,'no2')

remove\_outlier(dataset,'rspm')

remove\_outlier(dataset,'spm')

dataset.groupby('state')[['spm', 'pm2\_5', 'rspm', 'so2', 'no2']].mean()

by\_State=dataset.groupby('state')

def impute\_mean(series):

    return series.fillna(series.mean())

dataset['rspm']=by\_State['rspm'].transform(impute\_mean)

dataset['so2']=by\_State['so2'].transform(impute\_mean)

dataset['no2']=by\_State['no2'].transform(impute\_mean)

dataset['spm']=by\_State['spm'].transform(impute\_mean)

dataset['pm2\_5']=by\_State['pm2\_5'].transform(impute\_mean)

for col in dataset.columns.values:

    if dataset[col].isnull().sum() == 0:

        continue

    if col == 'date':

        guess\_values = dataset.groupby('state')['date'].apply(lambda x: x.mode().max())

    elif col=='type':

        guess\_values = dataset.groupby('state')['type'].apply(lambda x: x.mode().max())

    else:

        guess\_values = dataset.groupby('state')['location'].apply(lambda x: x.mode().max())

dataset.head()

dataset.to\_csv('cleaned\_data.csv', index=False)

**8.2 Calculating Index Values of Pollutants:**

from IPython.display import Image, display

# Specify the path to your image file with double backslashes

image\_path = 'Screenshot 2024-01-21 003647.png'

# Display the image

display(Image(filename=image\_path))

def calculate\_si(so2):

    si=0

    if (so2<=40):

     si= so2\*(50/40)

    elif (so2>40 and so2<=80):

     si= 50+(so2-40)\*(50/40)

    elif (so2>80 and so2<=380):

     si= 100+(so2-80)\*(100/300)

    elif (so2>380 and so2<=800):

     si= 200+(so2-380)\*(100/420)

    elif (so2>800 and so2<=1600):

     si= 300+(so2-800)\*(100/800)

    elif (so2>1600):

     si= 400+(so2-1600)\*(100/800)

    return si

dataset['si']=dataset['so2'].apply(calculate\_si)

df= dataset[['so2','si']]

df.head()

def calculate\_ni(no2):

    ni=0

    if(no2<=40):

     ni= no2\*50/40

    elif(no2>40 and no2<=80):

     ni= 50+(no2-40)\*(50/40)

    elif(no2>80 and no2<=180):

     ni= 100+(no2-80)\*(100/100)

    elif(no2>180 and no2<=280):

     ni= 200+(no2-180)\*(100/100)

    elif(no2>280 and no2<=400):

     ni= 300+(no2-280)\*(100/120)

    else:

     ni= 400+(no2-400)\*(100/120)

    return ni

dataset['ni']=dataset['no2'].apply(calculate\_ni)

df= dataset[['no2','ni']]

df.head()

def calculate\_(rspm):

    rpi=0

    if(rpi<=30):

     rpi=rpi\*50/30

    elif(rpi>30 and rpi<=60):

     rpi=50+(rpi-30)\*50/30

    elif(rpi>60 and rpi<=90):

     rpi=100+(rpi-60)\*100/30

    elif(rpi>90 and rpi<=120):

     rpi=200+(rpi-90)\*100/30

    elif(rpi>120 and rpi<=250):

     rpi=300+(rpi-120)\*(100/130)

    else:

     rpi=400+(rpi-250)\*(100/130)

    return rpi

dataset['rpi']=dataset['rspm'].apply(calculate\_si)

df= dataset[['rspm','rpi']]

df.head()

def calculate\_spi(spm):

    spi=0

    if(spm<=50):

     spi=spm\*50/50

    elif(spm>50 and spm<=100):

     spi=50+(spm-50)\*(50/50)

    elif(spm>100 and spm<=250):

     spi= 100+(spm-100)\*(100/150)

    elif(spm>250 and spm<=350):

     spi=200+(spm-250)\*(100/100)

    elif(spm>350 and spm<=430):

     spi=300+(spm-350)\*(100/80)

    else:

     spi=400+(spm-430)\*(100/430)

    return spi

dataset['spi']=dataset['spm'].apply(calculate\_spi)

df= dataset[['spm','spi']]

df.head()

#Function to calculate pm2\_5 individual pollutant index(pmi)

def calculate\_pmi(pm2\_5):

    pmi=0

    if(pm2\_5<=50):

     pmi=pm2\_5\*(50/50)

    elif(pm2\_5>50 and pm2\_5<=100):

     pmi=50+(pm2\_5-50)\*(50/50)

    elif(pm2\_5>100 and pm2\_5<=250):

     pmi= 100+(pm2\_5-100)\*(100/150)

    elif(pm2\_5>250 and pm2\_5<=350):

     pmi=200+(pm2\_5-250)\*(100/100)

    elif(pm2\_5>350 and pm2\_5<=450):

     pmi=300+(pm2\_5-350)\*(100/100)

    else:

     pmi=400+(pm2\_5-430)\*(100/80)

    return pmi

dataset['pmi']=dataset['pm2\_5'].apply(calculate\_pmi)

df= dataset[['pm2\_5','pmi']]

def calculate\_aqi(si,ni,spi,rpi):

    aqi=0

    if(si>ni and si>spi and si>rpi):

     aqi=si

    if(spi>si and spi>ni and spi>rpi):

     aqi=spi

    if(ni>si and ni>spi and ni>rpi):

     aqi=ni

    if(rpi>si and rpi>ni and rpi>spi):

     aqi=rpi

    return aqi

dataset['AQI']=dataset.apply(lambda x:calculate\_aqi(x['si'],x['ni'],x['spi'],x['rpi']),axis=1)

df= dataset[['state','si','ni','rpi','spi','AQI']]

df.head()

dataset['date'] = pd.to\_datetime(dataset['date'],format='%Y-%m-%d') # date parse

dataset['year'] = dataset['date'].dt.year # year

dataset['year'] = dataset['year'].fillna(0.0).astype(int)

dataset = dataset[(dataset['year']>0)]

df = dataset[['AQI','year','state']].groupby(["year"]).median().reset\_index().sort\_values(by='year',ascending=False)

dataset.tail()

**8.3 Removing Correlation:**

dataset.fillna(0.0,inplace=True)

states=dataset.groupby(['state','location'],as\_index=False).mean()

state=states.groupby(['state',],as\_index=False).mean()

state

selected\_columns = ['state','date','si', 'ni', 'rpi', 'spi', 'pmi','AQI']

new\_dataframe = dataset[selected\_columns]

# Save the new DataFrame to a CSV file

new\_dataframe.to\_csv('new\_dataset.csv', index=False)

dataset['date'] = pd.to\_datetime(dataset['date'],format='%Y-%m-%d') # date parse

dataset['year'] = dataset['date'].dt.year # year

dataset['year'] = dataset['year'].fillna(0.0).astype(int)

dataset = dataset[(dataset['year']>0)]

data\_p = dataset.drop(['state', 'location', 'type','date','AQI','year'],axis=1)

corr = data\_p.corr()

columns = np.full((corr.shape[0],), True, dtype=int)

selected\_column=data\_p.columns[columns]

data\_p=data\_p[selected\_column]

**8.4 Implementing Regression Algorithms:**

**Linear Regression:**

def linear\_regression(X, y):

    # SPLIT TEST AND TRAIN

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

    # One Hot Encoding

    X\_train = pd.get\_dummies(X\_train)

    X\_test = pd.get\_dummies(X\_test)

    # Linear Regression

    LR = LinearRegression()

    LR.fit(X\_train, y\_train)

    predictions = LR.predict(X\_test)

    print(X\_test.shape, X\_train.shape, y\_test.shape, y\_train.shape)

    print('r2\_Square: %.2f ' % r2\_score(y\_test, predictions))

    print('MSE: %.2f ' % np.sqrt(mean\_squared\_error(y\_test, predictions)))

    print("KfoldCrossVal mean score using Linear regression is %s" % cross\_val\_score(LR, X\_train, y\_train, cv=10).mean())

    regressor\_OLS = sm.OLS(y\_train, X\_train).fit()

    plt.figure(figsize=(18, 10))

    plt.scatter(predictions, y\_test, alpha=0.3)

    plt.xlabel('Predictions')

    plt.ylabel('AQI')

    plt.title("Linear Prediction ")

    plt.show()

    # Cross-validation

    Kfold = KFold(len(X), shuffle=True)

    # X\_train = sc.fit\_transform(X\_train)

    # X\_test = sc.transform(X\_test)

    z = print(regressor\_OLS.summary())

    return z

def cross\_val(X,y):

    #SPLIT TEST AND TRAIN

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size=0.2)

#One Hot Encoding

    X\_train = pd.get\_dummies(X\_train)

    X\_test = pd.get\_dummies(X\_test)

#Linear Regression

    LR = LinearRegression()

    LR.fit(X\_train, y\_train)

    predictions = LR.predict(X\_test)

    Kfold = KFold(len(X), shuffle=True)

    print("KfoldCrossVal mean score using Linear regression is %s" %cross\_val\_score(LR,X\_train,y\_train,cv=10).mean())

# print(dataset.columns)

X\_1=dataset[['si','ni','rpi','spi']]

y\_1=dataset['AQI']

linear\_regression(X\_1,y\_1)

**Polynomial Regression:**

def polynomial\_regression(X, y, degree=2):

    # SPLIT TEST AND TRAIN

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

    # One Hot Encoding

    X\_train = pd.get\_dummies(X\_train)

    X\_test = pd.get\_dummies(X\_test)

    # Polynomial Regression

    poly\_features = PolynomialFeatures(degree=degree, include\_bias=False)

    X\_train\_poly = poly\_features.fit\_transform(X\_train)

    X\_test\_poly = poly\_features.transform(X\_test)

    LR\_poly = LinearRegression()

    LR\_poly.fit(X\_train\_poly, y\_train)

    predictions\_poly = LR\_poly.predict(X\_test\_poly)

    print(X\_test\_poly.shape, X\_train\_poly.shape, y\_test.shape, y\_train.shape)

    print('r2\_Square (Polynomial): %.2f ' % r2\_score(y\_test, predictions\_poly))

    print('MSE (Polynomial): %.2f ' % np.sqrt(mean\_squared\_error(y\_test, predictions\_poly)))

    print("KfoldCrossVal mean score using Polynomial regression is %s" % cross\_val\_score(LR\_poly, X\_train\_poly, y\_train, cv=10).mean())

    plt.figure(figsize=(18, 10))

    plt.scatter(predictions\_poly, y\_test, alpha=0.3)

    plt.xlabel('Predictions (Polynomial)')

    plt.ylabel('AQI')

    plt.title(f"Polynomial Regression (Degree {degree})")

    plt.show()

    return LR\_poly

from sklearn.preprocessing import PolynomialFeatures

from sklearn.pipeline import make\_pipeline

def cross\_val(X, y, degree=1):

    # SPLIT TEST AND TRAIN

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

    # One Hot Encoding

    X\_train = pd.get\_dummies(X\_train)

    X\_test = pd.get\_dummies(X\_test)

    LR = LinearRegression()

    LR.fit(X\_train, y\_train)

    linear\_regression\_score = cross\_val\_score(LR, X\_train, y\_train, cv=10).mean()

    print("KfoldCrossVal mean score using Linear regression is %s" % linear\_regression\_score)

    poly\_model = make\_pipeline(PolynomialFeatures(degree), LinearRegression())

    poly\_model.fit(X\_train, y\_train)

    polynomial\_regression\_score = cross\_val\_score(poly\_model, X\_train, y\_train, cv=10).mean()

    print("KfoldCrossVal mean score using Polynomial regression (degree {}) is {}".format(degree, polynomial\_regression\_score))

polynomial\_regression(X\_1, y\_1, degree=2)

**StepWise Regression:**

import statsmodels.api as sm

import pandas as pd

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, KFold

from sklearn.metrics import mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

from sklearn.base import BaseEstimator, RegressorMixin

class StatsmodelsWrapper(BaseEstimator, RegressorMixin):

    def \_\_init\_\_(self, model):

        self.model = model

        self.model\_fit = None

    def fit(self, X, y):

        self.fitted\_model = sm.OLS(y, sm.add\_constant(X)).fit()

        return self

    def predict(self, X):

        if self.fitted\_model is not None:

            return self.fitted\_model.predict(sm.add\_constant(X))

def stepwise\_regression(X, y,

                        initial\_list=[],

                        threshold\_in=0.01,

                        threshold\_out=0.05,

                        verbose=True):

    included = list(initial\_list)

    while True:

        changed = False

        # forward step

        excluded = list(set(X.columns) - set(included))

        new\_pval = pd.Series(index=excluded, dtype=float)

        for new\_column in excluded:

            model\_columns = included + [new\_column]

            X\_model = sm.add\_constant(X[model\_columns])

            model = sm.OLS(y, X\_model).fit()

            new\_pval[new\_column] = model.pvalues[new\_column]

        best\_pval = new\_pval.min()

        if best\_pval < threshold\_in:

            best\_feature = new\_pval.idxmin()

            included.append(best\_feature)

            changed = True

            if verbose:

                print('Add  {:30} with p-value {:.6}'.format(best\_feature, best\_pval))

        # backward step

        X\_model = sm.add\_constant(X[included])

        model = sm.OLS(y, X\_model).fit()

        # use all coefs except intercept

        pvalues = model.pvalues.iloc[1:]

        worst\_pval = pvalues.max()  # null if pvalues is empty

        if worst\_pval > threshold\_out:

            changed = True

            worst\_feature = pvalues.idxmax()

            included.remove(worst\_feature)

            if verbose:

                print('Drop {:30} with p-value {:.6}'.format(worst\_feature, worst\_pval))

        if not changed:

            break

    # Scatter plot for Stepwise Regression

    X\_stepwise = sm.add\_constant(X[included])

    model\_stepwise = sm.OLS(y, X\_stepwise).fit()

    predictions\_stepwise = model\_stepwise.predict(X\_stepwise)

    plt.figure(figsize=(18, 10))

    plt.scatter(predictions\_stepwise, y, alpha=0.3)

    plt.xlabel('Predictions (Stepwise Regression)')

    plt.ylabel('AQI')

    plt.title('Stepwise Regression')

    plt.show()

    # Print metrics

    print(X\_stepwise.shape, X.shape, y.shape)

    print('r2\_Square (Stepwise): %.2f ' % r2\_score(y, predictions\_stepwise))

    print('MSE (Stepwise): %.2f ' % np.sqrt(mean\_squared\_error(y, predictions\_stepwise)))

    print("KfoldCrossVal mean score using Stepwise regression is %s" % cross\_val\_score(StatsmodelsWrapper(model\_stepwise), X\_stepwise, y, cv=10).mean())

    return included, StatsmodelsWrapper(model\_stepwise)

def cross\_val\_stepwise(X, y):

    # SPLIT TEST AND TRAIN

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

    # One Hot Encoding

    X\_train = pd.get\_dummies(X\_train)

    X\_test = pd.get\_dummies(X\_test)

    # Stepwise Regression

    selected\_features, stepwise\_model = stepwise\_regression(X\_train, y\_train)

    # Cross-validation

    kfold = KFold(len(X), shuffle=True)

    mse\_scores = cross\_val\_score(stepwise\_model, X\_train[selected\_features], y\_train, cv=10, scoring='neg\_mean\_squared\_error')

    r2\_scores = cross\_val\_score(stepwise\_model, X\_train[selected\_features], y\_train, cv=10, scoring='r2')

    # print("KfoldCrossVal mean score using Step-Wise regression is %s" % mse\_scores.mean())

    # Calculate R-squared

    r2\_mean = r2\_scores.mean()

    # print("KfoldCrossVal mean R-squared using Step-Wise regression is %s" % r2\_mean)

    return mse\_scores, r2\_scores

# Example usage

X = dataset[['si', 'ni', 'rpi', 'spi']]

y = dataset['AQI']

# Cross-validation with Stepwise Regression

mse\_scores\_stepwise, r2\_scores\_stepwise = cross\_val\_stepwise(X, y)

**Logistic Regression:**

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report, accuracy\_score

from sklearn.model\_selection import train\_test\_split, KFold, cross\_val\_score

import matplotlib.pyplot as plt

import statsmodels.api as sm

def logistic\_regression(x, y):

    x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.25, random\_state=0)

    sc = StandardScaler()

    # Feature scaling

    x\_train = sc.fit\_transform(x\_train)

    x\_test = sc.transform(x\_test)

    # Fitting logistic regression to the training set

    classifier = LogisticRegression(random\_state=0)

    classifier.fit(x\_train, y\_train)

    # Logistic regression cross-validation

    k\_fold = KFold(n\_splits=10, shuffle=True, random\_state=0)

    cvs = cross\_val\_score(classifier, x\_train, y\_train, cv=k\_fold).mean()

    print("KfoldCrossVal mean score using Logistic regression is %s \n" % cvs)

    print("Logistic Analysis Report")

    y\_pred = classifier.predict(x\_test)

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

    print(y\_pred)

    # Accuracy score

    accuracy = accuracy\_score(y\_test, classifier.predict(x\_test))

    print("Accuracy Score: %.2f" % accuracy)

    # Probability of the dependent variable

    y\_pred\_proba = classifier.predict\_proba(x\_test)[:, 1]

    print('Probability of the dependent variable')

    print(y\_pred\_proba.mean())

    fpr, tpr, \_ = metrics.roc\_curve(y\_test,  y\_pred\_proba)

    auc = metrics.roc\_auc\_score(y\_test, y\_pred\_proba)

    plt.plot(fpr, tpr, label="data 1, auc="+str(auc))

    plt.plot([0, 1], [0, 1], 'r--')

    plt.legend(loc=4)

    plt.xlabel('False Positive Rate')

    plt.ylabel('True Positive Rate')

    plt.show()

    return accuracy, x, y  # Return accuracy, x, and y

def logit\_summary(y, X):

    logit\_model = sm.Logit(y, X)

    result = logit\_model.fit()

    print("Model Summary")

    print(result.summary2())

# Example data

# Replace this with your actual data

import numpy as np

# Generating example data for illustration purposes

np.random.seed(0)

x = np.random.rand(100, 2)

y = (x[:, 0] + x[:, 1] > 1).astype(int)

# Call the logistic\_regression function

accuracy, x\_data, y\_data = logistic\_regression(x, y)

# Print the accuracy

# Call the logistic\_regression function

r2, x\_data, y\_data = logistic\_regression(x, y)

print("R-squared (R2): %.4f" % r2)

print("Mean Squared Error (MSE): %.4f" % accuracy)

# You can use x\_data and y\_data for further analysis if needed

**Ridge Regression:**

from sklearn.linear\_model import RidgeClassifier

from sklearn.model\_selection import train\_test\_split, KFold, cross\_val\_score

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import classification\_report, mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

import statsmodels.api as sm

def ridge\_regression(x, y, alpha=1.0):

    x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.25, random\_state=0)

    sc = StandardScaler()

    # Feature scaling

    x\_train = sc.fit\_transform(x\_train)

    x\_test = sc.transform(x\_test)

    # Fitting Ridge regression to the training set

    classifier = RidgeClassifier(alpha=alpha, random\_state=0)

    classifier.fit(x\_train, y\_train)

    # Ridge regression cross-validation

    k\_fold = KFold(n\_splits=10, shuffle=True, random\_state=0)

    cvs = cross\_val\_score(classifier, x\_train, y\_train, cv=k\_fold).mean()

    print("KfoldCrossVal mean score using Ridge regression is %s \n" % cvs)

    print("Ridge Analysis Report")

    y\_pred = classifier.predict(x\_test)

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

    print(y\_pred)

    # Accuracy score

    accuracy = metrics.accuracy\_score(y\_test, y\_pred)

    print("Accuracy Score: %.2f" % accuracy)

    # Probability of the dependent variable

    y\_pred\_proba = classifier.decision\_function(x\_test)

    print('Probability of the dependent variable')

    print(y\_pred\_proba.mean())

    fpr, tpr, \_ = metrics.roc\_curve(y\_test,  y\_pred\_proba)

    auc = metrics.roc\_auc\_score(y\_test, y\_pred\_proba)

    plt.plot(fpr, tpr, label="data 1, auc="+str(auc))

    plt.plot([0, 1], [0, 1], 'r--')

    plt.legend(loc=4)

    plt.xlabel('False Positive Rate')

    plt.ylabel('True Positive Rate')

    plt.show()

    # Calculate MSE and R2

    mse = mean\_squared\_error(y\_test, y\_pred)

    r2 = r2\_score(y\_test, y\_pred)

    return mse, r2, x\_test, y\_test  # Return MSE, R2, x\_test, and y\_test

def ridge\_logit\_summary(y, X, alpha=1.0):

    ridge\_logit\_model = sm.Logit(y, X)

    ridge\_result = ridge\_logit\_model.fit\_regularized(alpha=alpha)

    print("Ridge Model Summary")

    print(ridge\_result.summary2())

# Example usage

ridge\_alpha = 1.0

mse, r2, x\_test\_data, y\_test\_data = ridge\_regression(X, y, alpha=ridge\_alpha)

ridge\_logit\_summary(y\_test\_data, x\_test\_data, alpha=ridge\_alpha)

# Print the MSE and R2

print("Mean Squared Error (MSE): %.4f" % mse)

print("R-squared (R2): %.4f" % r2)

#yakiy67969@alibrs.com itsnotmyaccount

**8.5 Implementing Large Language Models:**

import os

import sys

from langchain.document\_loaders import CSVLoader

from langchain.indexes import VectorstoreIndexCreator

from langchain.llms import OpenAI

from langchain.chat\_models import ChatOpenAI

os.environ["OPNAI\_API\_KEY"] = "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"

query = sys.argv[1]

# Create a CSVLoader with the specified CSV file

loader = CSVLoader('cleaned\_data.csv')

index = VectorstoreIndexCreator().from\_loaders([loader])

print(index.query(query, llm=ChatOpenAI()))

# https://blog.langchain.dev/retrieval/

**CHAPTER – 9**

**RESULT**

9.1 Regression:

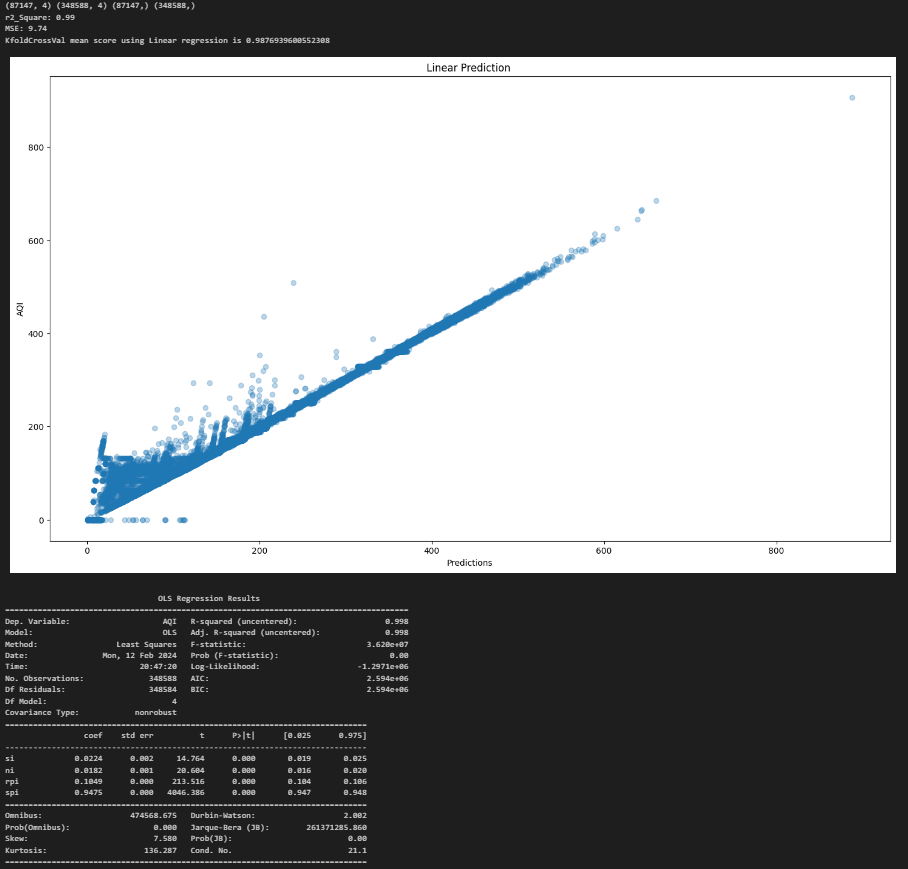
****

Fig: 9.1.1 Linear Regression

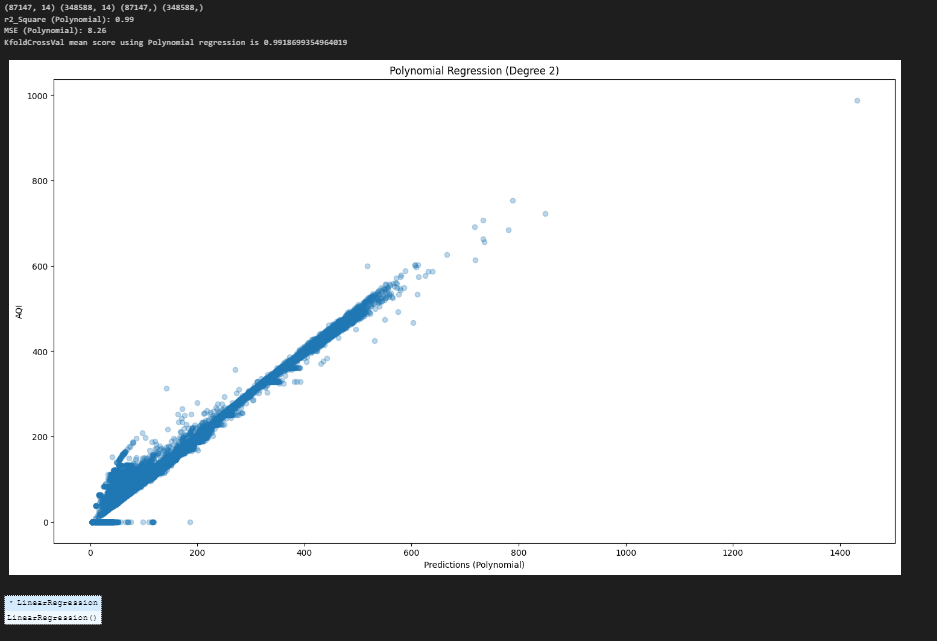


Fig: 9.1.2 Polynomial Regression

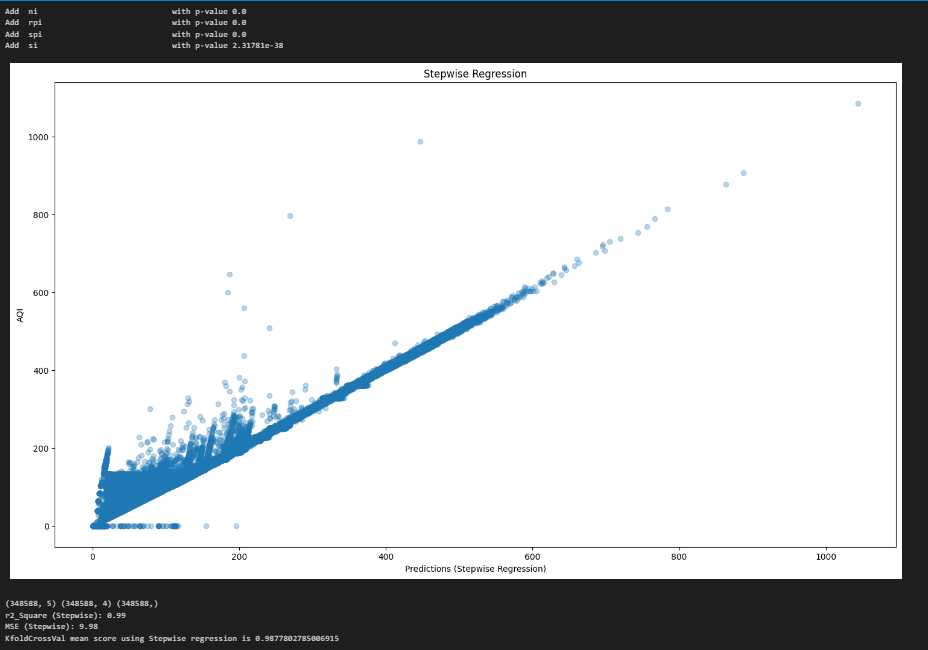
****

Fig: 9.1.3 StepWise Regression

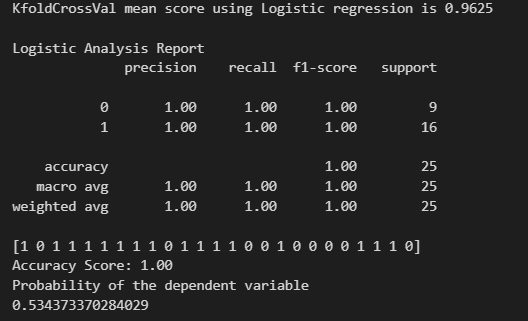
­

Fig: 9.1.4 Logistic Regression

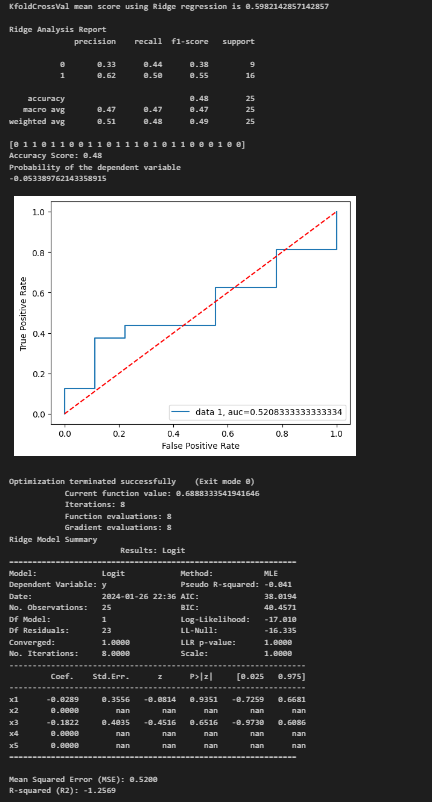


Fig: 9.1.5 Ridge Regression­­­­

9.2 LLM Model:

To calculate the Air Quality Index (AQI) for each column in the dataset, we need to use the respective formula for each pollutant. Here are the calculations and the updated dataset:

For Sulfur Dioxide (SO2):

AQI = (100 \* C) / I

Where C is the concentration of SO2 and I is the standard for SO2 (24-hour average, 80 ╡g/m│)

For Nitrogen Dioxide (NO2):

AQI = (100 \* C) / I

Where C is the concentration of NO2 and I is the standard for NO2 (24-hour average, 100 ╡g/m│)

For Respirable Suspended Particulate Matter (RSPM):

AQI = (100 \* C) / I

Where C is the concentration of RSPM and I is the standard for RSPM (24-hour average, 100 ╡g/m│)

For Suspended Particulate Matter (SPM):

AQI = (100 \* C) / I

Where C is the concentration of SPM and I is the standard for SPM (24-hour average, 200 ╡g/m│)

For PM2.5:

AQI = (100 \* C) / I

Where C is the concentration of PM2.5 and I is the standard for PM2.5 (24-hour average, 60 ╡g/m│)

Here is the updated dataset with the AQI values added:

state: Andhra Pradesh

location: Hyderabad

type: Residential, Rural and other Areas

Sulfur dioxide: 3.6

nitrogen dioxide: 18.6

respirable suspended particulate matter: 78.18282385

suspended particulate matter: 200.2603783

PM2.5: 20.22555556

date: 01-05-1990

AQI (SO2): 4.5

AQI (NO2): 18.6

AQI (RSPM): 78.2

AQI (SPM): 100.1

AQI (PM2.5): 33.7

state: Andhra Pradesh

location: Hyderabad

type: Industrial Area

Sulfur dioxide: 4

nitrogen dioxide: 8.9

respirable suspended particulate matter: 78.18282385

suspended particulate matter: 200.2603783

PM2.5: 19.91388889

date: 01-05-1990

AQI (SO2): 5

AQI (NO2): 8.9

AQI (RSPM): 78.2

AQI (SPM): 100.1

AQI (PM2.5): 33.2

state: Andhra Pradesh

location: Hyderabad

type: Industrial Area

Sulfur dioxide: 4.7

nitrogen dioxide: 7.5

respirable suspended particulate matter: 78.18282385

suspended particulate matter: 200.2603783

PM2.5: 22.3

date: 01-03-1990

AQI (SO2): 5.9

AQI (NO2): 7.5

AQI (RSPM): 78.2

AQI (SPM): 100.1

AQI (PM2.5): 37.2

state: Andhra Pradesh

location: Hyderabad

type: Residential, Rural and other Areas

Sulfur dioxide: 3.5

nitrogen dioxide: 12.1

respirable suspended particulate matter: 78.18282385

suspended particulate matter: 135

PM2.5: 21.78388889

date: 01-07-1990

AQI (SO2): 4.4

AQI (NO2): 12.1

AQI (RSPM): 78.2

AQI (SPM): 67.5

AQI (PM2.5): 36.3

Please note that the AQI values provided here are calculated based on the given concentrations and standard values. The AQI is rounded to one decimal place.

Fig: 9.2.1 LLM Output

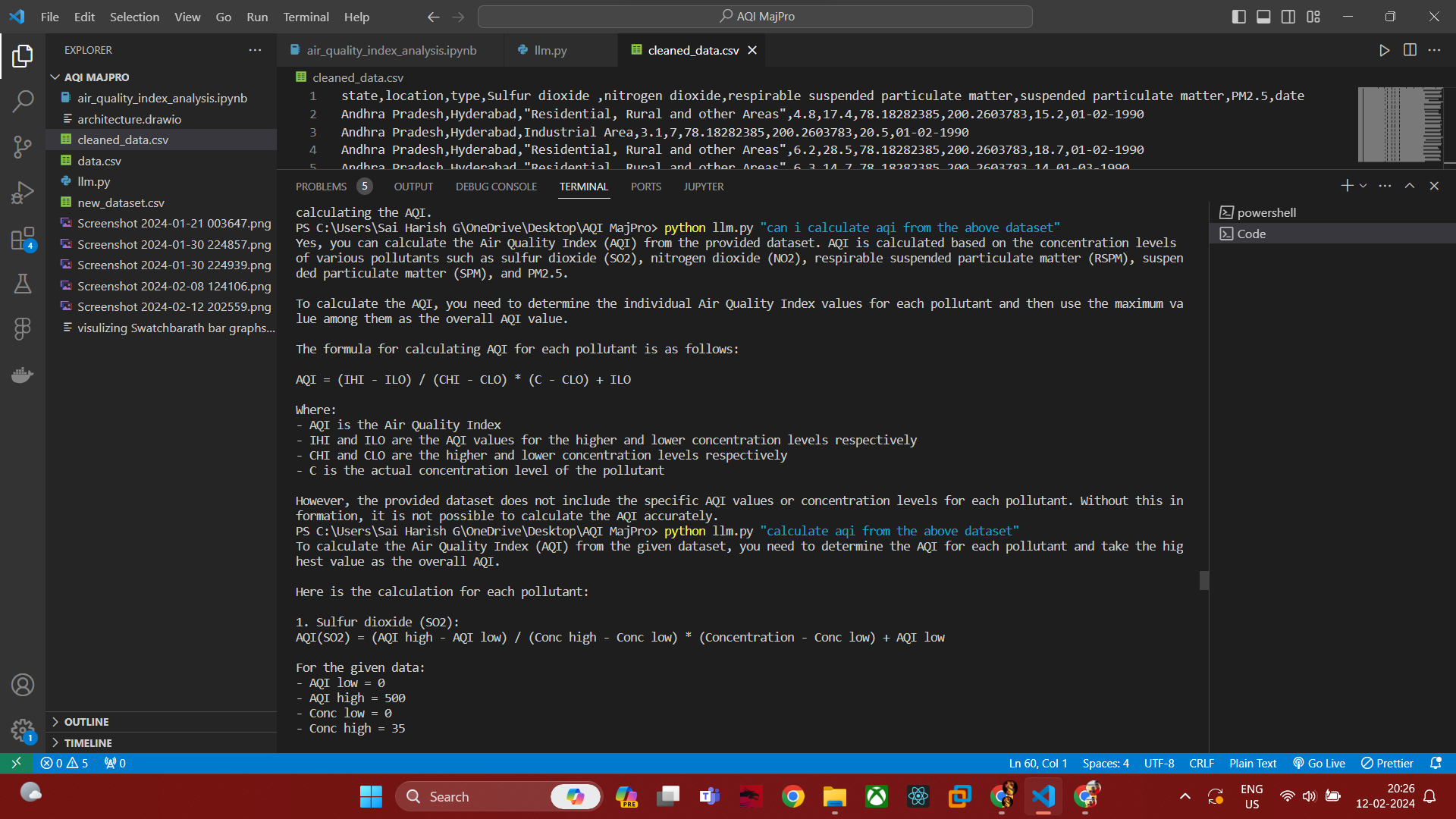


Fig: 9.2.2 LLM Output

**CHAPTER – 10**

**ANALYSIS**

Comparative Analysis for different Algorithms

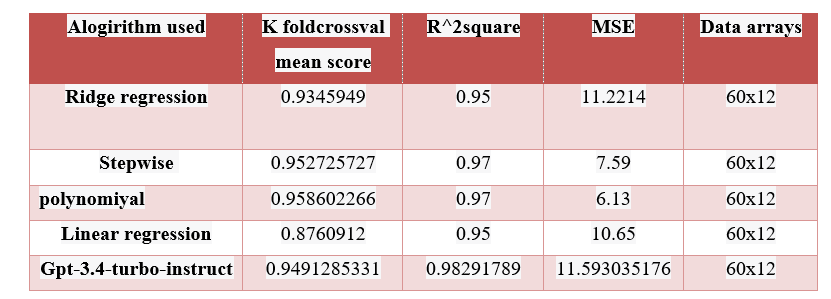


Fig: 10.1

Fig: 10.2

Fig: 10.3

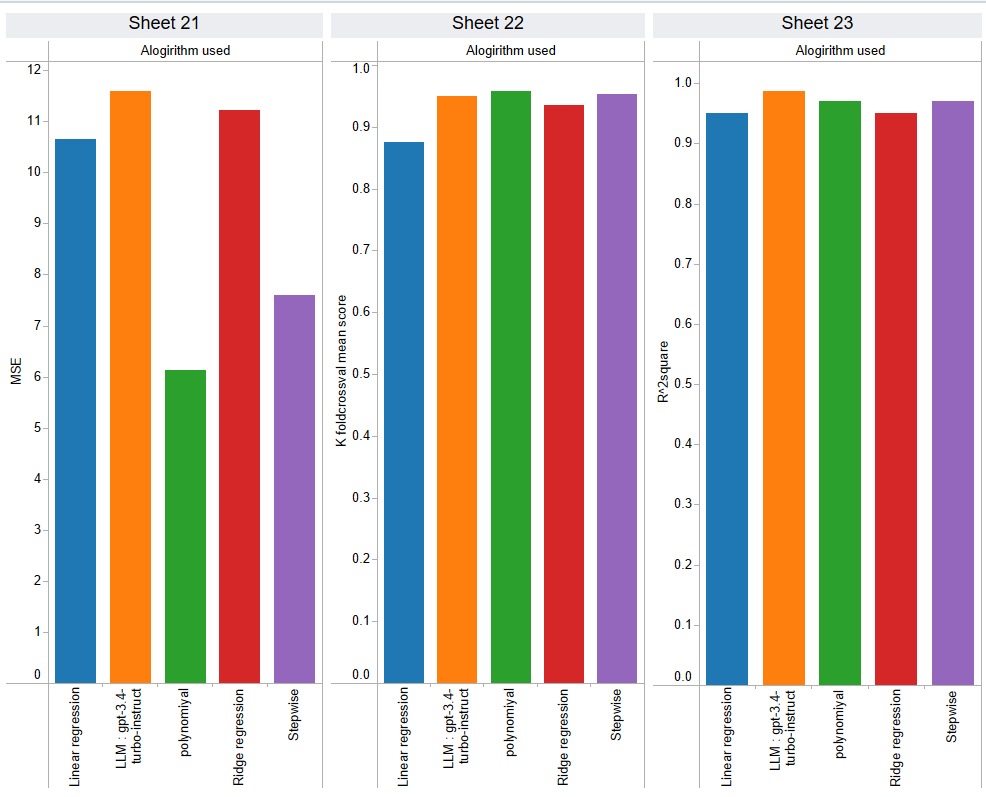
****

Fig:10.3

Fig:10.4

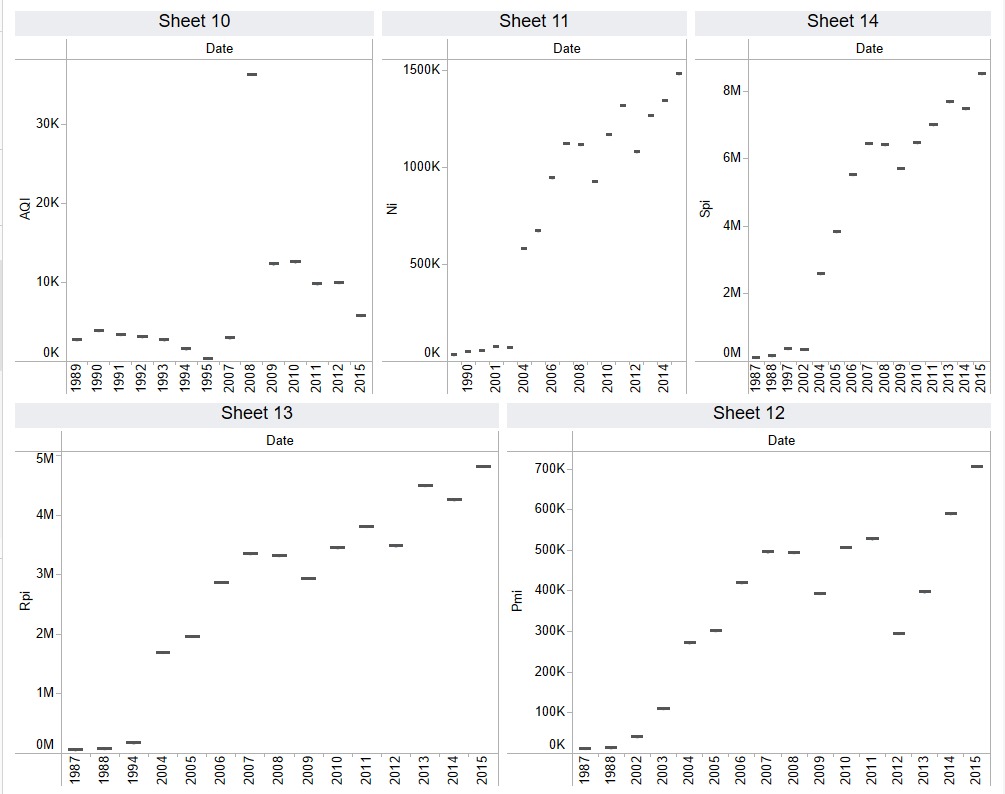
****

Fig:10.5

Fig:10.6

Fig:10.7

**CHAPTER – 11**

**CONCLUSION AND FUTURE SCOPE**

While machine learning offers promising solutions for air quality prediction, the current algorithms have limitations in predicting extreme pollution points, determining cut-offs, handling complex calculations efficiently, and incorporating temporal aspects of data. This project, focused on addressing these limitations, proposes a new approach based on:

* **Improved extreme point prediction:** Utilizing techniques like Long Short-Term Memory (LSTM) networks to capture temporal trends and predict maximum and minimum pollution levels.
* **Optimized cut-off determination:** Developing data-driven methods to establish reliable cut-offs for air quality index calculations.
* **Efficient prediction process:** Simplifying and streamlining calculations without compromising accuracy, leading to faster predictions.
* **Temporal data integration:** Effectively incorporating time-sensitive factors like weather patterns and seasonal variations into predictions.

By addressing these challenges, this project aims to deliver:

* **Enhanced prediction accuracy:** Leading to more reliable forecasts of air quality, particularly during peak pollution events or periods of exceptional cleanliness.
* **Improved public health protection:** Facilitating timely interventions and preventive measures based on accurate forecasts.
* **Better environmental management:** Enabling policymakers to develop targeted strategies for mitigating air pollution and its negative impacts.

## Future Scope

The proposed approach can be further extended by:

* **Expanding data sources:** Integrating additional data like satellite imagery and traffic information to capture broader environmental factors.
* **Real-time predictions:** Developing models that can continuously update predictions based on live sensor data, enabling real-time air quality monitoring.
* **Personalized recommendations:** Utilizing individual health data and location information to provide personalized air quality alerts and mitigation strategies.
* **Multi-city implementation:** Adapting the model to predict air quality across various cities and regions, offering insights for broader environmental management efforts.

By building upon this project, we can contribute significantly to creating a future with cleaner air, safeguarding public health and fostering a more sustainable environment for all.

**REFERENCES**

* [1] (2016). PhishMe Q1 2016 Malware Review. [Online]. Available: <https://phishme.com/project/phishme-q1-2016-malware-review/>
* [2] A. Belabed, E. Aimeur, and A. Chikh, ‘‘A personalized whitelist approach for phishing webpage detection,’’ in Proc. 7th Int. Conf. Availability, Rel. Security (ARES), Aug. 2012, pp. 249–254.
* [3] Y. Cao, W. Han, and Y. Le, ‘‘Anti-phishing based on automated individual white-list,’’ in Proc. 4th ACM Workshop Digit. Identity Manage., 2008, pp. 51–60.
* [4] T.-C. Chen, S. Dick, and J. Miller, ‘‘Detecting visually similar Web pages: Application to phishing detection,’’ ACM Trans. Internet Technol., vol. 10, no. 2, pp. 1–38, May 2010.
* [5] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. C. Mitchell, ‘‘Clientside defense against Web-based identity theft,’’ in Proc. 11th Annu. Netw. Distrib. Syst. Security Symp. (NDSS), 2004, pp. 1–16
* [6] C. Inc. (Aug. 2016). Couldmark Toolbar. [Online]. Available: <http://www.cloudmark.com/desktop/ie-toolbar>
* [7] J. Corbetta, L. Invernizzi, C. Kruegel, and G. Vigna, ‘‘Eyes of a human, eyes of a program: Leveraging different views of the Web for analysis and detection,’’ in Proceedings of Research in Attacks, Intrusions and Defenses (RAID). Gothenburg, Sweden: Springer, 2014.
* [8] X. Deng, G. Huang, and A. Y. Fu, ‘‘An antiphishing strategy based on visual similarity assessment,’’ Internet Comput., vol. 10, no. 2, pp. 58–65, 2006.
* [9] Z. Dong, K. Kane, and L. J. Camp, ‘‘Phishing in smooth waters: The state of banking certificates in the US,’’ in Proc. Res. Conf. Commun., Inf. Internet Policy (TPRC), 2014, p. 16.
* [10] D. Boneh, and J. C. Mitchell, N. Chou, R. Ledesma, Y. Teraguchi, ‘‘Clientside defense against Web-based identity theft,’’ in Proc. 11th Annu. Netw. Distrib. Syst. Security Symp. (NDSS), 2004, pp. 1–16.