**Student Name:** SANTHOSH S

**Register Number:** 410723104076

**Institution:** DHANALAKSHIMI COLLEGE OF ENGINEERING

**Department:** COMPUTER SCIENCE AND ENGINEERING

**Date of Submission:** 14/05/2025

**Github Repository Link: https://github.com/Santhosh-OFFX/NM-santhosh**

## 1. Problem Statement

*Rapid urbanization and industrial growth have led to significant air pollution in many regions, particularly in developing countries like India. Traditional methods of monitoring air quality are often inadequate due to their limited spatial coverage and inability to provide timely forecasts. This project aims to develop a machine learning-based system that can accurately predict air quality levels, specifically the Air Quality Index (AQI), to inform public health decisions and environmental policies*

## 2. Abstract

*Air pollution poses significant health risks, particularly in rapidly urbanizing regions like India. Traditional air quality monitoring*

*methods often lack the temporal and spatial resolution needed for effective decision-making. This project employs machine learning algorithms to predict the Air Quality Index (AQI) using historical data on various pollutants and meteorological parameters.*

*The study utilizes datasets from cities such as Visakhapatnam and Chennai, incorporating factors like PM2.5, PM10, $NO_2$, CO, $SO_2$, temperature, humidity, and wind speed. Advanced models like Cat Boost and X G Boost are trained and evaluated for their predictive accuracy. The results demonstrate that these models can accurately forecast AQI levels, providing valuable insights for public health and environmental policy.*

*By leveraging machine learning, this approach offers a scalable and efficient method for air quality prediction, aiding in proactive environmental management and policy-making.*

## 3. System Requirements

### Hardware Requirements

- **CPU**: Intel Core i5 or AMD Ry zen 5 (≥4 cores)
- **RAM**: 8 GB (16 GB recommended)
- **Storage**: 100 GB HDD (500 GB SSD recommended)
- **GPU**: Optional, but beneficial for deep learning tasks (e.g., NVIDIA GTX 1050 or better)
- **Operating System**: Windows 10/11 (64-bit), macOS 10.12+, or Linux (Ubuntu preferred)

### Software Requirements :

- **Programming Language**: Python 3.8 or higher

- **Development Environment**: Jupiter Notebook, Google Collab, or VS Code
- **Libraries**:
  - **Data Science**: Pandas, NumPy, Matplotlib, Seaborn
  - **Machine Learning**: Scikit-learn, XG Boost, Light GBM, Cat Boost
  - **Deep Learning**: TensorFlow, Keras
- **Version Control**: Git (for code versioning and collaboration)
-

## 4. Objectives

### 1. Develop Predictive Models for AQI Forecasting

- Create machine learning models to predict the Air Quality Index (AQI) using historical data.
- Utilize algorithms such as Random Forest, XG Boost, Cat Boost, and LSTM to compare performance.

### 2. Identify Key Factors Influencing Air Quality

- Analyze the impact of pollutants like PM2.5, PM10, $NO_2$, CO, $SO_2$, and meteorological parameters (temperature, humidity, wind speed) on AQI levels.
- Employ feature selection techniques to determine the most significant contributors.

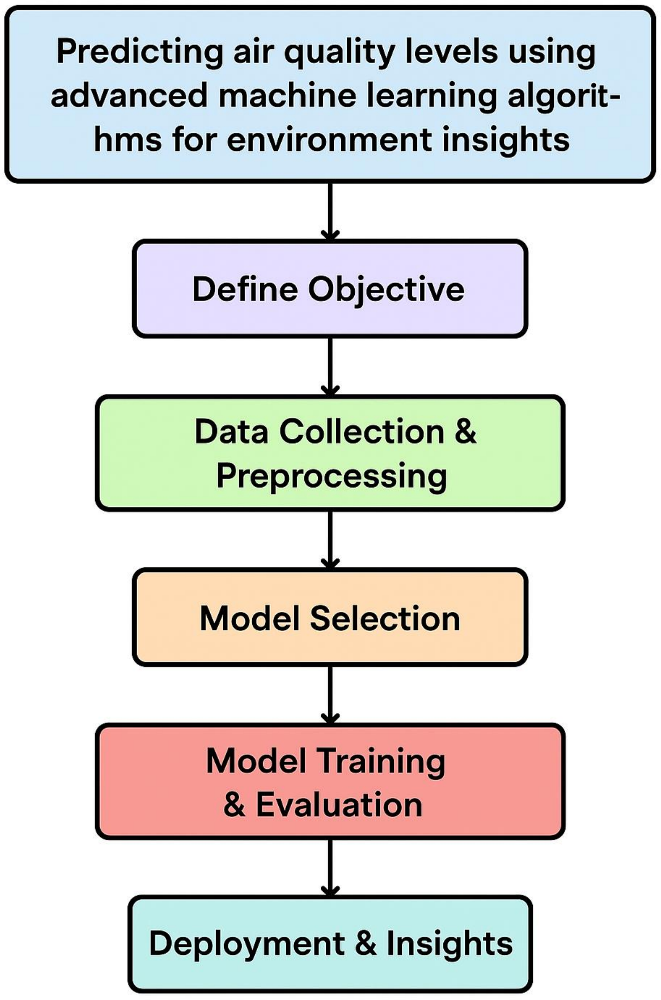
### 3. Evaluate and Optimize Model Performance

- Assess models using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and $R^2$ score.
- Implement hyperparameter tuning to enhance model accuracy.

### 4. Provide Real-Time AQI Predictions and Insights

- Develop a user-friendly interface (e.g., web application) for real-time AQI predictions.
- Offer actionable insights to inform public health decisions and environmental policies.

## 5. Flowchart of Project Workflow


Predicting air quality levels using advanced machine learning algorit-hms for environment insights
Define Objective
Data Collection & Preprocessing
Model Selection
Model Training & Evaluation
Deployment & Insights

## 6. Dataset Description

Key Attributes

| Attribute | Description |
|---|---|
| City | Name of the city (e.g., Chennai, Visakhapatnam) |
| Date & Time | Timestamp of the observation (hourly) |
| PM2.5 | Particulate Matter ≤ 2.5 µm (µg/m$^3$) |
| PM10 | Particulate Matter ≤ 10 µm (µg/m$^3$) |
| $NO_2$ | Nitrogen Dioxide concentration (µg/m$^3$ or ppb) |
| $SO_2$ | Sulfur Dioxide concentration (µg/m$^3$ or ppb) |
| CO | Carbon Monoxide concentration (ppm) |
| $O_3$ | Ozone concentration (ppb or µg/m$^3$) |
| Temperature | Ambient temperature (°C) |
| Humidity | Relative humidity (%) |
| Wind Speed | Wind speed (m/s) |
| Wind Direction | Wind direction (°) |
| Rainfall | Precipitation (mm) |
| Solar Radiation | Solar radiation (W/m$^2$) |
| Barometric Pressure | Atmospheric pressure (hPa) |

## 7. Data Preprocessing

### 1. Data Cleaning

- **Handle Missing Values**: Identify and address missing data using imputation techniques such as k-Nearest Neighbors (k-NN) or by removing rows with missing values.
- **Remove Duplicates**: Eliminate any duplicate entries to ensure data integrity.

- **Correct Errors**: Fix any inconsistencies or errors in the dataset, such as negative values for parameters that cannot be negative.

## 2. Feature Engineering

- **Select Relevant Features**: Identify and retain features that significantly impact air quality, such as PM2.5, PM10, $NO_2$, CO, $SO_2$, and meteorological parameters.
- **Create New Features**: Generate additional features that might enhance model performance, like time-based variables (e.g., hour of the day, day of the week).
- **Encode Categorical Variables**: Convert categorical variables into numerical formats using techniques like Label Encoding or One-Hot Encoding.

## 3. Normalization and Scaling

- **Normalize Data**: Apply normalization techniques to scale features to a [0, 1] range, ensuring that all features contribute equally to the model.
- **Standardize Data**: Use standardization (Z-score normalization) to transform features so that they have a mean of 0 and a standard deviation of 1, which is particularly useful for algorithms sensitive to feature scaling.

## 4. Data Splitting

- **Split Dataset**: Divide the dataset into training and testing sets, commonly using an 80/20 or 70/30 split, to evaluate model performance effectively.
- **Ensure Temporal Consistency**: For time-series data, ensure that the training set precedes the testing set to simulate real-world prediction scenarios.

## 5. Outlier Detection

- **Identify Outliers**: Detect outliers using statistical methods or visualization techniques like box plots.

- **Handle Outliers**: Decide whether to remove, cap, or transform outliers based on their impact on model performance.

**6. Data Transformation**

- **Feature Transformation**: Apply transformations like logarithmic or polynomial transformations to skewed features to improve model accuracy.
- **Time Series Decomposition**: Decompose time-series data into trend, seasonality, and residual components if necessary for certain models.

## 8. Exploratory Data Analysis (EDA)

**1. Data Overview**

- **Shape and Structure**: Examine the number of rows and columns to understand the dataset's size.
- **Data Types**: Identify the types of each feature (e.g., numerical, categorical).

**2. Missing Values**

- **Detection**: Check for any missing or null values in the dataset.
- **Handling**: Decide on strategies to address missing data, such as imputation or removal.

**3. Descriptive Statistics**

- **Summary Statistics**: Calculate measures like mean, median, standard deviation, and percentiles for numerical features.
- **Categorical Analysis**: Determine the frequency distribution of categorical variables.

**4. Data Visualization**

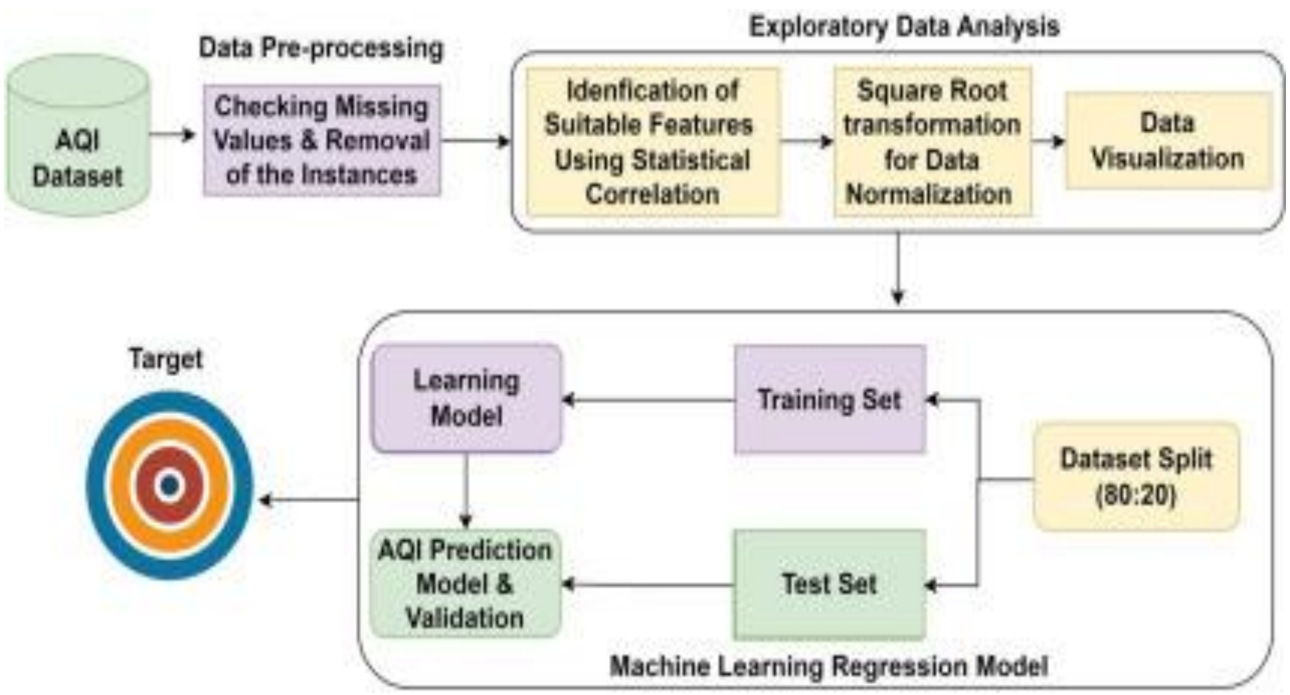
- **Histograms**: Visualize the distribution of individual numerical features.
- **Box Plots**: Identify outliers and understand the spread of the data.
- **Pair Plots**: Examine relationships between pairs of numerical variables.
- **Correlation Heatmap**: Assess the strength and direction of relationships between numerical features.

**5. Outlier Detection**

- **Methods**: Use statistical techniques like Z-scores or visualizations like box plots to detect outliers.

- **Treatment**: Decide whether to remove, cap, or transform outliers based on their impact on analysis.

## 9. Feature Engineering


Data Pre-processing
AQI Dataset
Checking Missing Values & Removal of the Instances
Exploratory Data Analysis
Idenfication of Suitable Features Using Statistical Correlation
Square Root transformation for Data Normalization
Data Visualization
Target
Learning Model
Training Set
Dataset Split (80:20)
AQI Prediction Model & Validation
Test Set
Machine Learning Regression Model

### 1. Temporal Feature Encoding

- **Cyclic Encoding**: Transform time-related features like month, day, and hour into cyclic representations using sine and cosine functions to capture seasonal patterns effectively.

### 2. Numerical Feature Transformation

- **Log Transformation**: Apply logarithmic transformations to skewed features to stabilize variance and normalize distributions, enhancing model performance.

### 3. Categorical Feature Encoding

- **One-Hot Encoding**: Convert categorical variables into binary vectors to allow models to interpret them correctly.

**4. Interaction Features**

- **Polynomial Features**: Generate interaction terms between features to capture non-linear relationships that may influence air quality.

**5. Feature Scaling**

- **Normalization**: Scale features to a [0, 1] range to ensure uniformity, preventing certain features from dominating due to differing units or ranges.

**6. Handling Outliers**

- **Capping**: Limit extreme values to a specified range to reduce their impact on model training.

**7. Feature Selection**

- **Importance Ranking**: Utilize algorithms like Random Forest or XGBoost to identify and retain the most influential features, improving model efficiency and accuracy.

## 10. Model Building

**1. Model Selection**

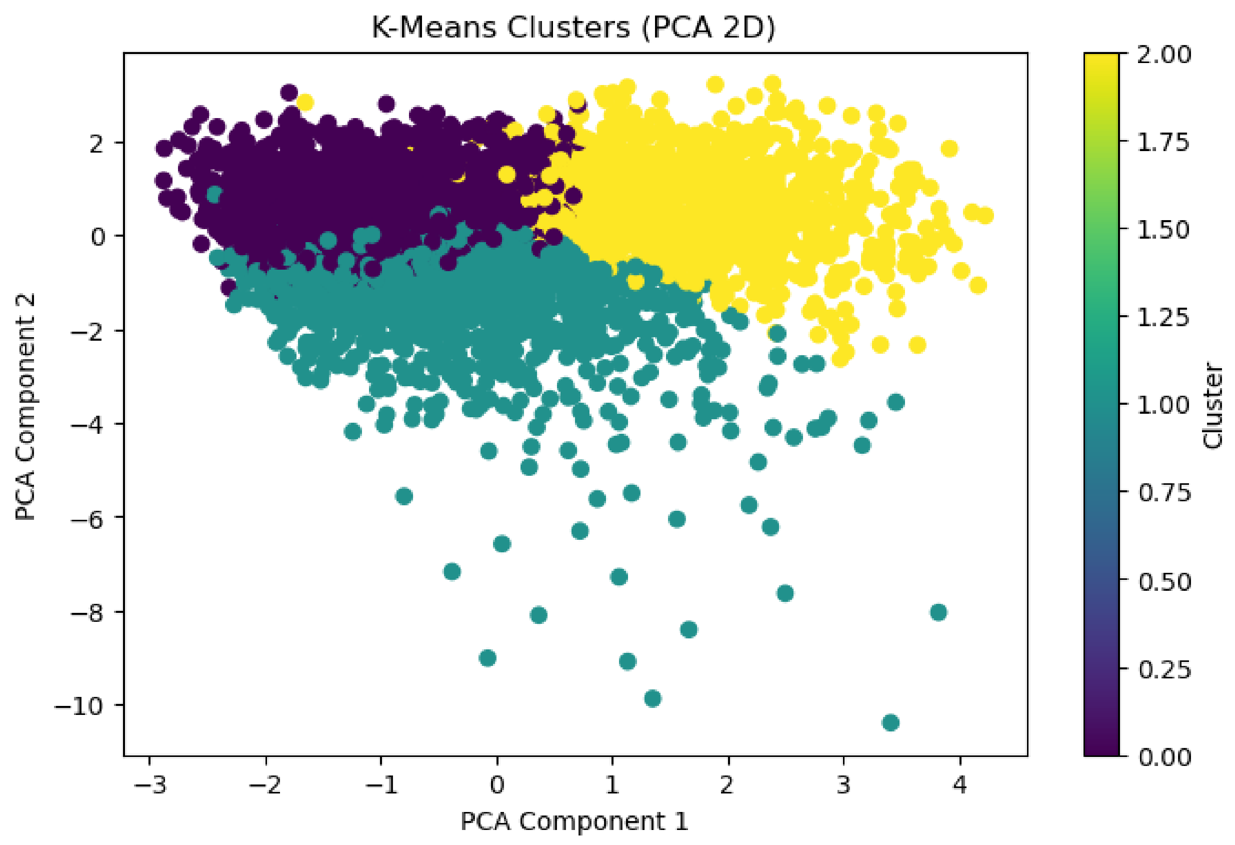
- **Regression Models**: Use algorithms like Random Forest Regressor, AdaBoost Regressor, and Support Vector Regression (SVR) for continuous AQI prediction.

**2. Model Training**

- **Data Splitting**: Divide the dataset into training (80%) and testing (20%) sets.
- **Feature Scaling**: Normalize features using techniques like Standard Scaler to standardize the range of independent variables.
- **Model Fitting**: Train the selected models on the training data.

**3. Model Evaluation**

- **Performance Metrics**: Assess models using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and $R^2$ score.
- **Model Comparison**: Compare the performance of different models to select the best one.



## 11. Model Evaluation

### *1. Evaluation Metrics*

*To assess the performance of your predictive models, consider the following metrics:*

- ***R² (Coefficient of Determination)****: Indicates the proportion of variance in the dependent variable that is predictable from the independent variables.*
- ***Mean Absolute Error (MAE)****: Measures the average magnitude of the errors in a set of predictions, without considering their direction.*
- ***Mean Squared Error (MSE)****: Calculates the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value.*
- ***Root Mean Squared Error (RMSE)****: Represents the square root of the average of the squared errors, providing a measure of the average magnitude of the error.*

*These metrics provide a comprehensive understanding of model accuracy and error distribution.*

***2. Model Comparison***

*Comparing different machine learning models can help in selecting the most appropriate one for your dataset:*

- ***Support Vector Regression (SVR)****: Effective in high-dimensional spaces and for cases where the number of dimensions exceeds the number of samples.*
- ***Random Forest Regression****: An ensemble method that operates by constructing multiple decision trees during training and outputs the mean prediction of the individual trees.*
- ***Cat Boost Regression****: A gradient boosting algorithm that handles categorical features well and is known for its speed and accuracy.*

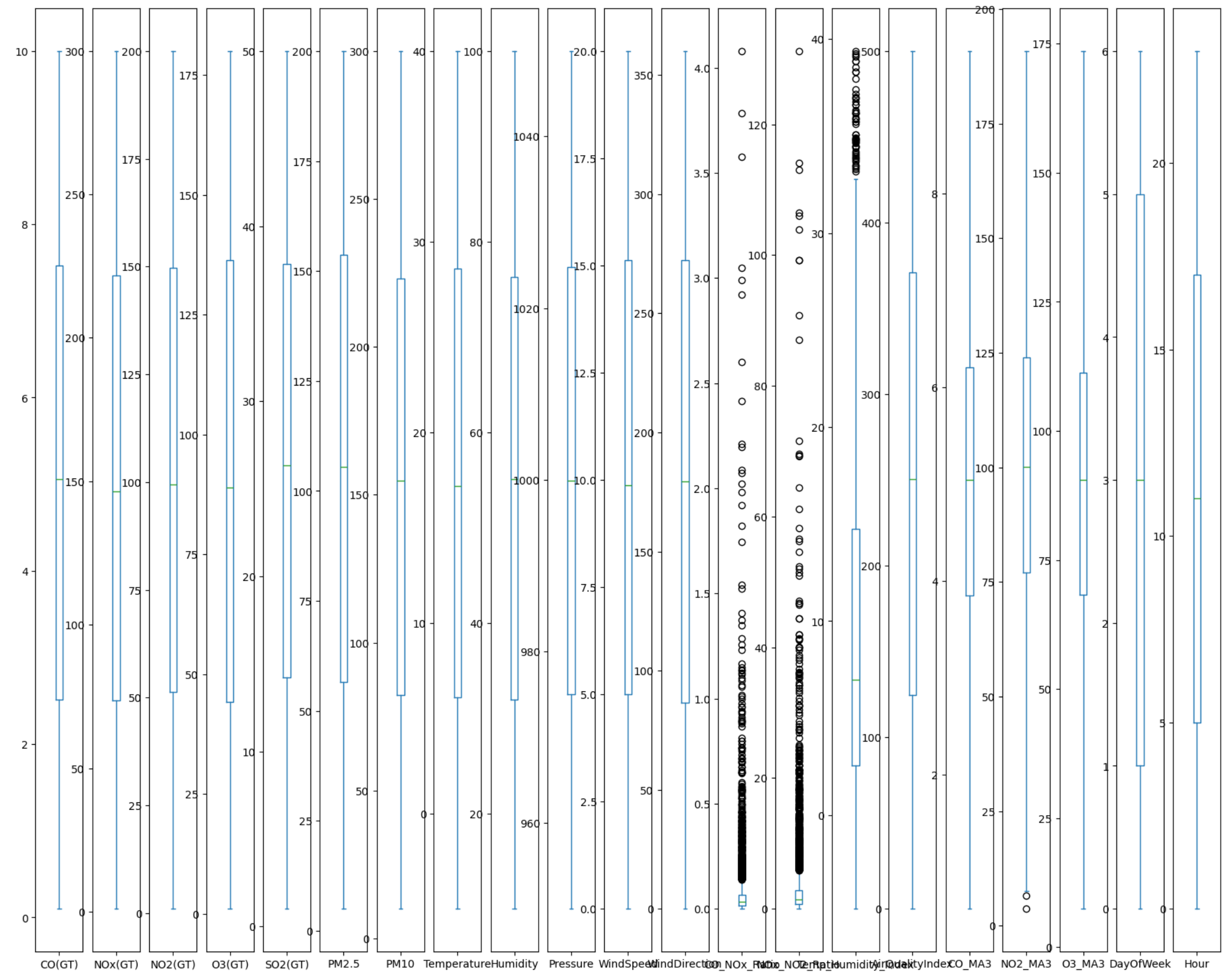
*For instance, in a study comparing these models for air quality prediction in New Delhi, Cat Boost Regression achieved the highest $R^2$ value and the lowest MAE, MSE, and RMSE, indicating superior performance over SVR and Random Forest Regression.*

***3. Visualization***

*Visualizing model performance can aid in better understanding and comparison:*

- ***Bar Charts****: Display the $R^2$, MAE, MSE, and RMSE values for each model to facilitate direct comparison.*
- ***Scatter Plots****: Plot predicted vs. actual values to assess the accuracy and identify any patterns or biases.*
- ***Residual Plots****: Help in diagnosing issues like heteroscedasticity or non-linearity in the model.*

*These visual tools are essential for interpreting model performance and making informed decisions*



## 12. Deployment

### *1. Model Deployment*

- ***Web Application****: Develop a user-friendly interface using frameworks like Flask or Stream lit to allow users to input parameters and receive air quality predictions.*
- ***Cloud Deployment****: Host the application on cloud platforms such as AWS, Azure, or Google Cloud to ensure scalability and accessibility.*
- ***Containerization****: Use Docker to containerize the application, ensuring consistency across different environments and simplifying the deployment process.*

### *2. Monitoring and Maintenance*

- ***Performance Monitoring***: *Implement tools to monitor the application's performance, ensuring it meets desired response times and handles user load effectively.*
- ***Model Retraining***: *Regularly update the model with new data to maintain its accuracy and relevance.*
- ***User Feedback***: *Collect feedback from users to identify areas for improvement and enhance the application's functionality.*

***3. Security and Compliance***

- ***Data Security***: *Ensure that user data is handled securely, implementing encryption and secure access protocols.*
- ***Regulatory Compliance***: *Adhere to relevant regulations and standards, such as GDPR, to protect user privacy and data*

## *13. Source code*

```
import numpy as np
import pandas as pd

 Load dataset
df = pd.read_csv('AirQualityUCI.csv', header=None, skiprows=1)
col = ['DATE', 'TIME', 'CO_GT', 'PT08_S1_CO', 'NMHC_GT', 'C6H6_GT',
       'PT08_S2_NMHC', 'NOX_GT', 'PT08_S3_NOX', 'NO2_GT',
       'PT08_S4_NO2', 'PT08_S5_O3', 'T', 'RH', 'AH']
df.columns = col
df['DATE'] = pd.to_datetime(df['DATE'], format='%d-%m-%Y')
df.dropna(inplace=True)

import seaborn as sns
import matplotlib.pyplot as plt

sns.heatmap(df.corr(), annot=True)
plt.title('Correlation Heatmap')
plt.show()
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

X = df.drop(['RH'], axis=1)
y = df['RH']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

from sklearn.metrics import mean_squared_error

predictions = model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, predictions))
print(f'Root Mean Squared Error: {rmse}')

df = pd.read_csv('AirQualityData.csv')
df = df.dropna()
df['Temperature'] = df['Temperature'].astype(str)

X = df['Temperature']
y = df['Date']

le = LabelEncoder()
y = le.fit_transform(y)

vectorizer = TfidfVectorizer()
X_tfidf = vectorizer.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2,
random_state=42)

svm_clf = svm.SVC(kernel='linear')
svm_clf.fit(X_train, y_train)
```

```
y_pred = svm_clf.predict(X_test)
In [15]:
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))

#k means clustering
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

df = pd.read_csv('AirQualityData.csv')
df_numeric = df.select_dtypes(include=[np.number])

scaler = StandardScaler()
scaled_data = scaler.fit_transform(df_numeric)

inertia = []
k_range = range(1, 11)

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_data)
    inertia.append(kmeans.inertia_)

plt.plot(k_range, inertia, marker='o')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method For Optimal k')
plt.grid(True)
plt.show()
```

```
kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster'] = kmeans.fit_predict(scaled_data)

print(df[['Cluster']].value_counts())

pca = PCA(n_components=2)
reduced_data = pca.fit_transform(scaled_data)

plt.figure(figsize=(8, 5))
plt.scatter(reduced_data[:, 0], reduced_data[:, 1], c=df['Cluster'], cmap='viridis')
plt.title('K-Means Clusters (PCA 2D)')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.colorbar(label='Cluster')
plt.show()
```

## *14. Future scope*

Incorporating data from various sources such as satellite imagery, social media, and public health records can enhance the accuracy of air quality predictions. This multi-source data integration allows for a more comprehensive understanding of pollution patterns and their impacts

*Employing advanced ML methods, including deep learning and ensemble models, can improve prediction accuracy. Techniques like Long Short-Term Memory (LSTM) networks and Random Forests have shown promise in capturing complex temporal and spatial patterns in air quality data*

*Federated Learning (FL) enables collaborative model training across multiple devices without sharing raw data, addressing privacy concerns.*

*FL can be particularly useful in air quality monitoring, where data privacy is crucial.*

## 15. Team Members and Roles

Team Leader : Nithin Chander R

- Data cleaning & EDA & Data Preprocessing.

Team Member : Sajit Kumar E

- Feature engineering & Dataset Description

Team Member : Vishwanathan p

- Source code & Future scope

Team Member : Santhosh S

- EDA & Data Preprocessing.

Team Member : Rakesh S

- Documentation and reporting