Github Link: [Https://github.com/pavithra 88888/project-predicting-air-quality-using-ml]

predicting air quality levels using advanced machine learning algorithms for environmental insights

phase-3

1. Problem Statement

Air pollution poses a significant threat to public health and the environment, with increasing urbanization and industrial activities exacerbating air quality degradation worldwide. Accurate prediction of air quality levels is essential for early warnings, policy-making, and reducing the exposure of populations to harmful pollutants. However, traditional methods for air quality monitoring are often reactive, costly, and limited in spatial coverage.

This project aims to develop an advanced machine learning-based system to predict air quality levels (e.g., AQI or concentrations of pollutants like PM2.5, PM10, NO2, O3, CO) using environmental data such as meteorological parameters, traffic data, industrial emissions, and historical air quality records. By leveraging cutting-edge machine learning algorithms, the goal is to deliver real-time and accurate air quality forecasts that can assist governments, environmental agencies, and the public in making informed decisions.

2. Abstract

Air pollution is a critical environmental challenge affecting human health, ecosystems, and climate worldwide. Accurate forecasting of air quality is essential for proactive environmental management, public health protection, and regulatory compliance. This study explores the application of advanced machine learning algorithms to predict air quality levels by analyzing a range of environmental and atmospheric variables such as temperature, humidity, wind speed, traffic density, and historical pollutant concentrations.

The proposed approach involves data preprocessing, feature engineering, and the implementation of various machine learning models, including Random Forest, Gradient Boosting, and Long Short-Term Memory (LSTM) networks. These models are trained and evaluated using real-world datasets collected from air quality monitoring stations and open-source meteorological data. Performance metrics such as RMSE, MAE, and R^2 are used to assess model accuracy and robustness.

3. System Requirements

A. Hardware Requirements

- **Processor:** Intel Core i5/i7 or AMD Ryzen 5/7 (or higher)
- **RAM:** Minimum 8 GB (16 GB recommended for deep learning models)
- **Storage:** Minimum 256 GB SSD (512 GB recommended for handling large datasets)

- **Graphics Card (Optional):** NVIDIA GPU with CUDA support (e.g., GTX 1660, RTX 3060 or higher) for training deep learning models efficiently

B. Software Requirements

- **Operating System:** Windows 10/11, macOS, or Linux (Ubuntu 18.04 or newer)
- **Programming Language:** Python 3.7 or higher
- **Integrated Development Environment (IDE):** Jupyter Notebook, Visual Studio Code, or PyCharm

C. Libraries and Frameworks

- **Data Handling & Preprocessing:**
    - Pandas
    - NumPy
    - Scikit-learn
- **Data Visualization:**
    - Matplotlib
    - Seaborn
    - Plotly
- **Machine Learning Algorithms:**
    - Scikit-learn (Random Forest, Gradient Boosting, etc.)
    - XGBoost or LightGBM
- **Deep Learning (Optional):**
    - TensorFlow or PyTorch (for LSTM, CNN, or hybrid models)
- **Other Tools:**
    - Air quality datasets APIs (e.g., OpenAQ, U.S. EPA AirNow)
    - Weather APIs (e.g., OpenWeatherMap) for meteorological data

4. Objectives

The primary objective of this project is to design and implement a machine learning-based system capable of accurately predicting air quality levels to support environmental monitoring and decision-making. The specific objectives are as follows:

4.1. Develop an Accurate Prediction Model

- To build machine learning models capable of forecasting air quality indices (AQI) and individual pollutant concentrations (e.g., PM2.5, PM10, NO_2, CO, O_3).

- To explore and compare different algorithms such as Random Forest, Gradient Boosting, Support Vector Machines, and deep learning models (e.g., LSTM).

4.2. Analyze Key Environmental Factors

- To identify and evaluate the impact of various meteorological and anthropogenic factors (e.g., temperature, humidity, wind speed, traffic, emissions) on air quality.

- To perform feature selection and engineering to improve model performance and interpretability.

4.3. Integrate Real-World Datasets

- To collect and preprocess data from reliable sources such as environmental monitoring stations, weather services, and public datasets.

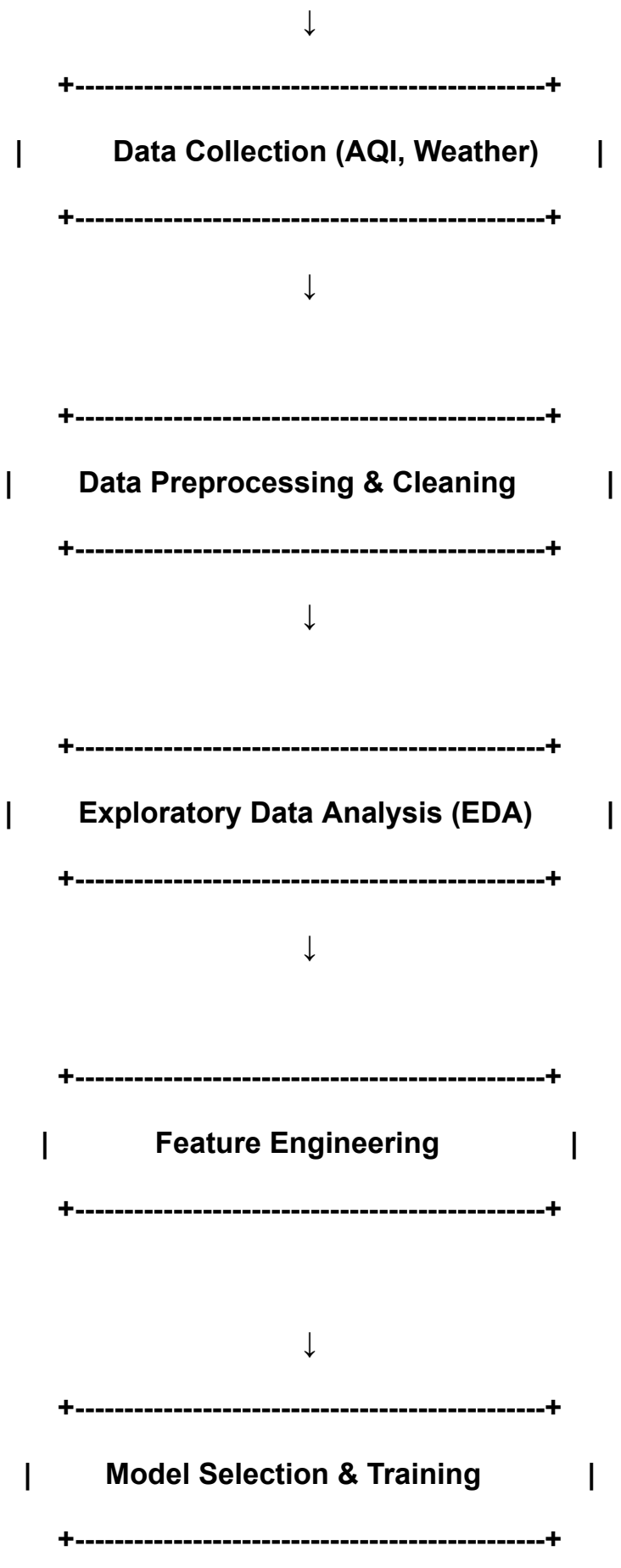
- To ensure data quality through cleaning, normalization, and handling of missing or inconsistent values.

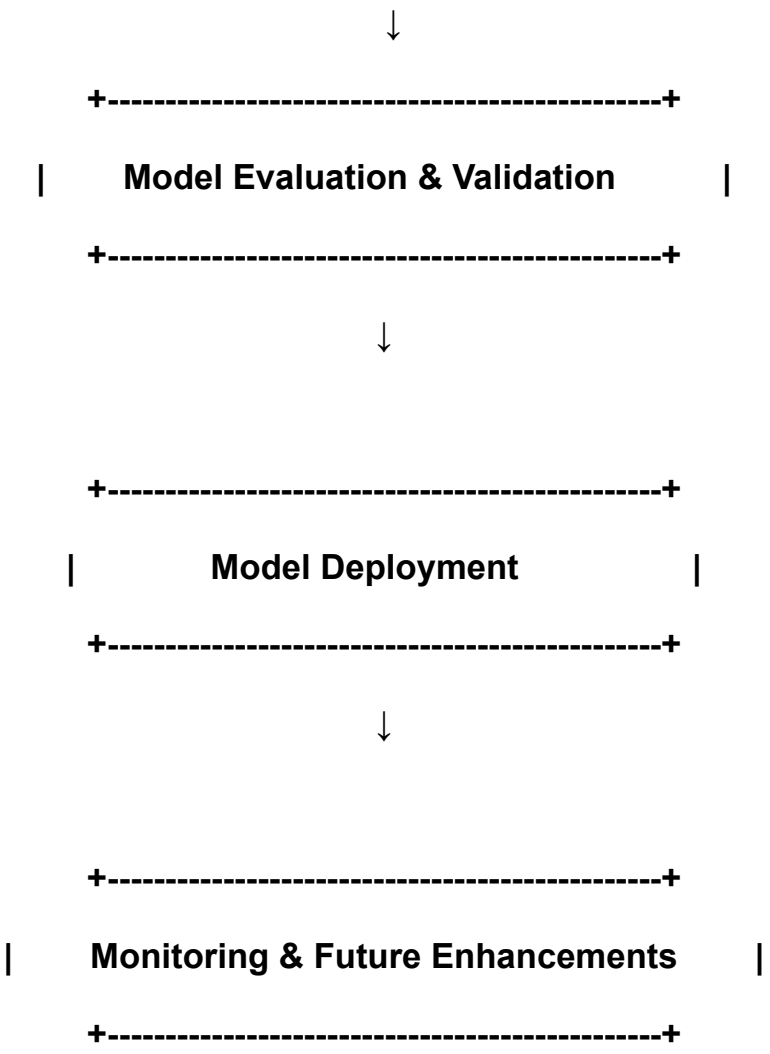
4.4. Validate and Optimize Model Performance

To use metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R^2 Score for evaluating model accuracy.

To fine-tune hyperparameters and implement cross-validation for robust performance.

5. Flowchart of the Project Workflow


Data Collection (AQI, Weather)
Data Preprocessing & Cleaning
Exploratory Data Analysis (EDA)
Feature Engineering
Model Selection & Training



6. Dataset Description

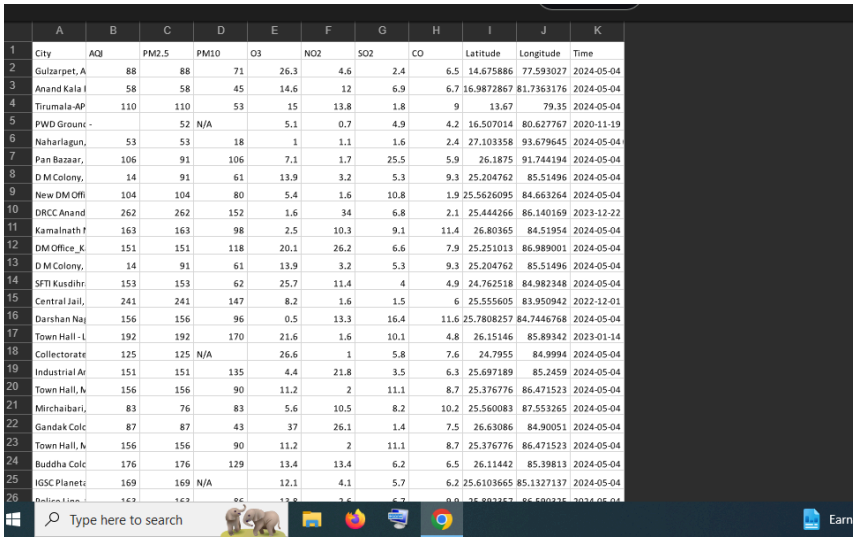
To accurately predict air quality levels, multiple datasets are integrated, encompassing air pollutant concentrations, meteorological variables, and temporal information. The following are the primary datasets used:

6.1. Air Quality Dataset

- **Source:** OpenAQ, U.S. EPA AirNow, or national environmental monitoring agencies
- **Attributes:**

- `PM2.5` (µg/m³): Fine particulate matter
- `PM10` (µg/m³): Coarse particulate matter
- `NO2` (ppb): Nitrogen Dioxide
- `CO` (ppm): Carbon Monoxide
- `O3` (ppb): Ozone
- `SO2` (ppb): Sulfur Dioxide
- `AQI`: Air Quality Index (calculated or provided)
- `Timestamp`: Date and time of measurement
- `Location ID`: Monitoring station identifier
- `Latitude/Longitude`: Geolocation of the station

Sample dataset (df.head())



	City	AQI	PM2.5	PM10	O3	NO2	SO2	CO	Latitude	Longitude	Time
2	Gulzarpet, A	88	88	71	26.3	4.6	2.4	6.5	14.675886	77.593027	2024-05-04
3	Anand Kala I	58	58	45	14.6	12	6.9	6.7	16.9872867	81.7363176	2024-05-04
4	Tirumala-AP	110	110	53	15	13.8	1.8	9	13.67	79.35	2024-05-04
5	PWD Ground -		52	N/A	5.1	0.7	4.9	4.2	16.507014	80.627767	2020-11-19
6	Naharlagun,	53	53	18	1	1.1	1.6	2.4	27.103358	93.679645	2024-05-04
7	Pan Bazaar,	106	91	106	7.1	1.7	25.5	5.9	26.1875	91.744194	2024-05-04
8	D M Colony,	14	91	61	13.9	3.2	5.3	9.3	25.204762	85.51496	2024-05-04
9	New DM Offi	104	104	80	5.4	1.6	10.8	1.9	25.5626095	84.663264	2024-05-04
10	DRCC Anand	262	262	152	1.6	34	6.8	2.1	25.444266	86.140169	2023-12-22
11	Kamalnath I	163	163	98	2.5	10.3	9.1	11.4	26.80365	84.51954	2024-05-04
12	DM Office_K	151	151	118	20.1	26.2	6.6	7.9	25.251013	86.989001	2024-05-04
13	D M Colony,	14	91	61	13.9	3.2	5.3	9.3	25.204762	85.51496	2024-05-04
14	SFTI Kusdihr	153	153	62	25.7	11.4	4	4.9	24.762518	84.982348	2024-05-04
15	Central Jail,	241	241	147	8.2	1.6	1.5	6	25.555605	83.950942	2022-12-01
16	Darshan Naj	156	156	96	0.5	13.3	16.4	11.6	25.7808257	84.7446768	2024-05-04
17	Town Hall - L	192	192	170	21.6	1.6	10.1	4.8	26.15146	85.89342	2023-01-14
18	Collectorate	125	125	N/A	26.6	1	5.8	7.6	24.7955	84.9994	2024-05-04
19	Industrial Ar	151	151	135	4.4	21.8	3.5	6.3	25.697189	85.2459	2024-05-04
20	Town Hall, N	156	156	90	11.2	2	11.1	8.7	25.376776	86.471523	2024-05-04
21	Mirchaibari,	83	76	83	5.6	10.5	8.2	10.2	25.560083	87.553265	2024-05-04
22	Gandak Colc	87	87	43	37	26.1	1.4	7.5	26.63086	84.90051	2024-05-04
23	Town Hall, N	156	156	90	11.2	2	11.1	8.7	25.376776	86.471523	2024-05-04
24	Buddha Colc	176	176	129	13.4	13.4	6.2	6.5	26.11442	85.39813	2024-05-04
25	IGSC Planeta	169	169	N/A	12.1	4.1	5.7	6.2	25.6103665	85.1327137	2024-05-04

6.2. Meteorological Dataset

- **Source:** OpenWeatherMap API, NOAA, or local meteorological stations
- **Attributes:**
  - `Temperature (°C)`

- `Relative Humidity (%)`
- `Wind Speed (m/s)`
- `Wind Direction (degrees)`
- `Atmospheric Pressure (hPa)`
- `Precipitation (mm)` (if available)
- `Timestamp`: Synchronized with pollution data

6.3. Temporal Features

- **Generated From Timestamps:**
  - `Hour of Day`
  - `Day of Week`
  - `Month`
  - `Weekend/Holiday` Indicator (if applicable)
  - `Season` (Winter, Spring, etc.)

---

7. Data Preprocessing

Data preprocessing is a crucial step in preparing raw data for effective machine learning model training. The quality of the data directly impacts the accuracy and reliability of the predictions. This stage involves cleaning, transforming, and engineering features from the collected datasets.

7.1. Data Cleaning

- **Handling Missing Values:**

    - Impute missing pollutant and weather values using interpolation, mean, median, or forward-fill methods.
    - Remove rows with excessive missing or invalid entries.
- **Removing Duplicates:**
    - Eliminate duplicate records based on timestamp and station ID.
- **Outlier Detection and Treatment:**
    - Detect anomalies using Z-score, IQR, or domain thresholds (e.g., PM2.5 > 1000 µg/m³).
    - Apply clipping, replacement, or removal based on context.

7.2. Data Integration

- **Merge Multiple Sources:**
    - Join air quality data with meteorological and traffic datasets using timestamp and location.
- **Time Alignment:**
    - Resample data to a consistent interval (e.g., hourly, daily).
    - Align time zones across sources.

7.3. Feature Engineering

- **Temporal Features:**
    - Extract `hour`, `day`, `month`, `weekday`, and `season` from timestamps.
- **Lag Features:**
    - Include previous hour/day pollutant concentrations as features to capture temporal dependencies.
- **Rolling Statistics:**

    - Use moving averages, standard deviations over time windows (e.g., 3-hour rolling PM2.5).

- **Categorical Encoding:**

    - Convert categorical features like `day of week` into one-hot or label encoded format.

7.4. Feature Scaling

- **Normalization/Standardization:**

    - Apply Min-Max scaling or Z-score normalization to numerical features to ensure consistency across feature ranges, especially for distance-sensitive models (e.g., SVM, KNN).

7.5. Data Splitting

- **Train-Test Split:**

    - Typically split data into 70–80% training and 20–30% testing sets.

    - Use time-based splitting if data is sequential to preserve temporal structure.

- **Validation Set:**

    - Further divide the training set for hyperparameter tuning (e.g., cross-validation).

7.6. Data Transformation (Optional)

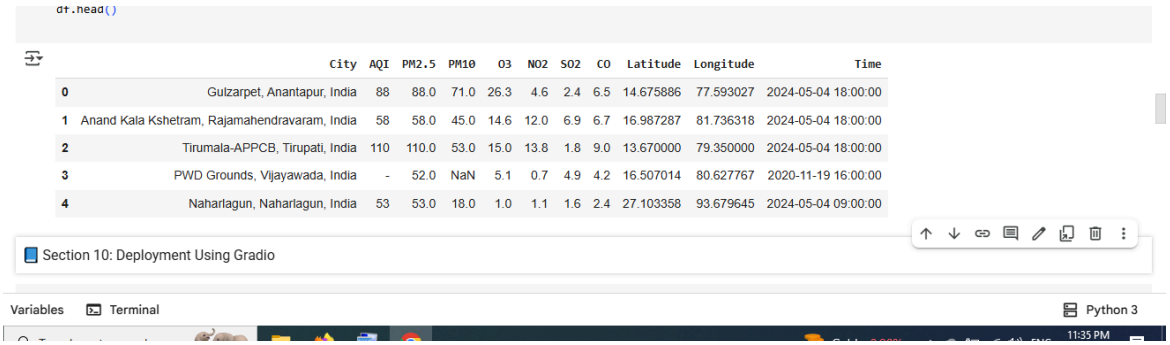
- **Log Transformation:**

    - Apply to skewed pollutant variables (e.g., PM2.5) to improve normality.

- **Dimensionality Reduction:**

    - Use PCA if needed to reduce multicollinearity and compress feature

    - ● Scaling:



```
df.head()
```

	City	AQI	PM2.5	PM10	O3	NO2	SO2	CO	Latitude	Longitude	Time
0	Gulzarpet, Anantapur, India	88	88.0	71.0	26.3	4.6	2.4	6.5	14.675886	77.593027	2024-05-04 18:00:00
1	Anand Kala Kshetram, Rajamahendravaram, India	58	58.0	45.0	14.6	12.0	6.9	6.7	16.987287	81.736318	2024-05-04 18:00:00
2	Tirumala-APPCB, Tirupati, India	110	110.0	53.0	15.0	13.8	1.8	9.0	13.670000	79.350000	2024-05-04 18:00:00
3	PWD Grounds, Vijayawada, India	-	52.0	NaN	5.1	0.7	4.9	4.2	16.507014	80.627767	2020-11-19 16:00:00
4	Naharlagun, Naharlagun, India	53	53.0	18.0	1.0	1.1	1.6	2.4	27.103358	93.679645	2024-05-04 09:00:00

8. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a vital step in understanding the structure, patterns, and relationships in the data before applying machine learning models. It helps uncover anomalies, detect correlations, and guide feature selection.

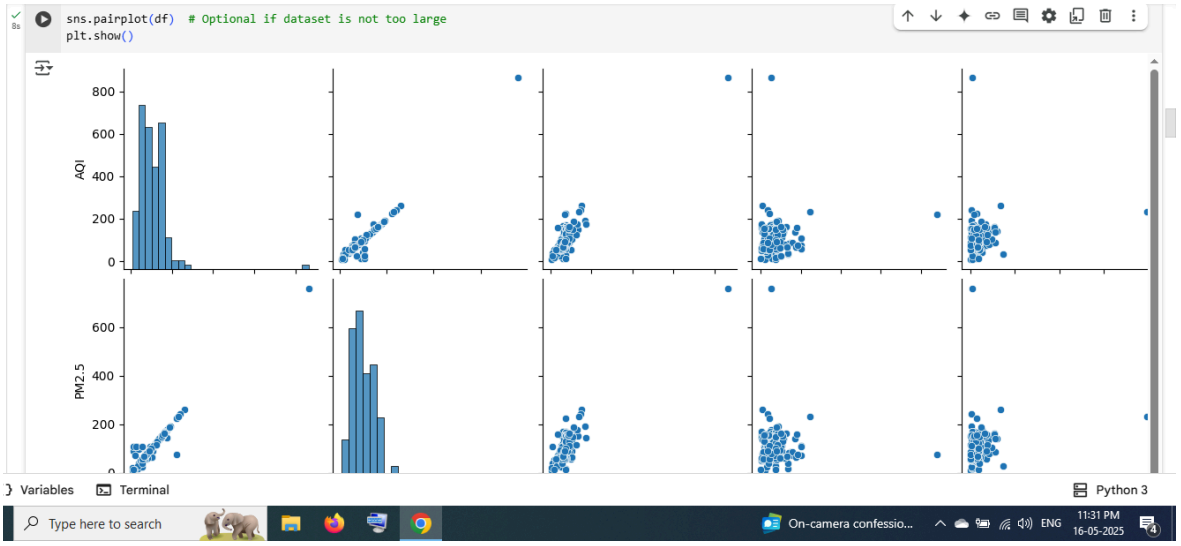
8.1. Understanding the Dataset

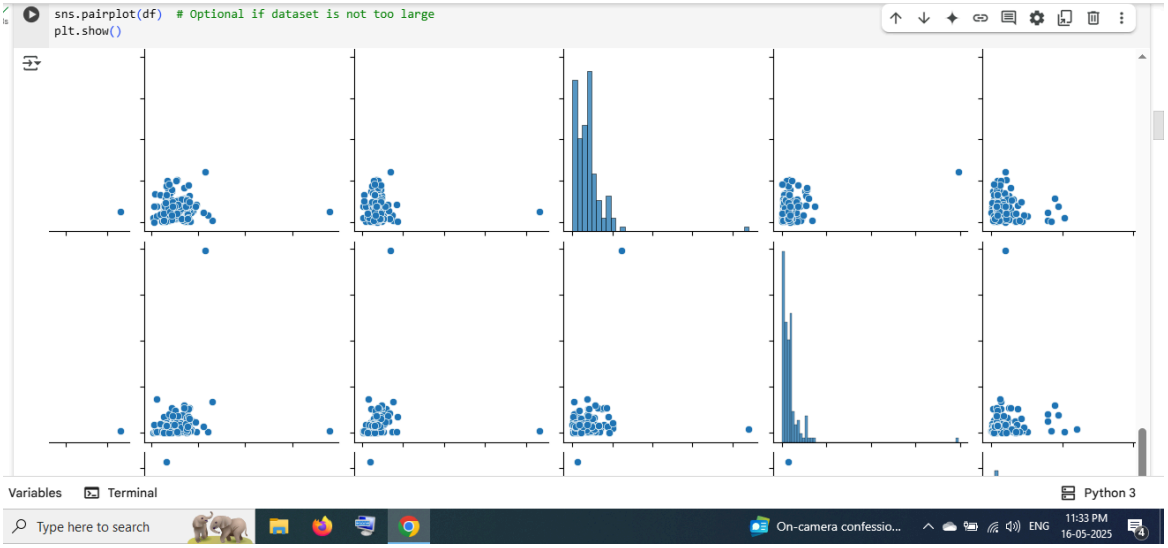
- **Descriptive Statistics:**
    - Summary of mean, median, min, max, standard deviation for each feature (e.g., PM2.5, temperature).
    - Distribution of AQI and pollutant concentrations across different locations and time periods.
- **Data Types & Missing Values:**
    - Checking for nulls, data types, and consistency.
    - Visualizing missing data using heatmaps or bar plots.

8.2. Univariate Analysis

- **Distribution Plots:**
    - Histograms and KDE plots for pollutant levels (e.g., PM2.5, NO_2) to understand skewness.
    - Box plots to detect outliers.
- **Pollutant Concentration by Time:**

- Line plots showing variation of pollutants over time (hourly, daily, monthly).
- Seasonal variation analysis.





9. Feature Engineering

Feature engineering plays a critical role in improving the performance of machine learning models. It involves transforming raw data into meaningful features that capture underlying patterns, relationships, and trends relevant to predicting air quality levels.

9.1. Temporal Features

- **Hour of the Day, Day of the Week, Month, Season:**

    - Capture periodic pollution patterns (e.g., rush hours, weekdays vs weekends, seasonal variation).

- **Holiday/Weekend Indicator:**
    - Flags days with reduced traffic or industrial activity.
- **Elapsed Time Since Last Rain (if available):**
    - Helps assess pollutant buildup or dispersion.

9.2. Lag Features

- **Previous Values of Pollutants:**
    - Include previous hour/day values of PM2.5, PM10, NO_2, etc., to model temporal dependencies.
    - Example: `PM2.5_lag_1`, `PM2.5_lag_3`, `NO2_lag_1`
- **Rolling Averages and Statistics:**
    - Moving average, rolling standard deviation over windows (e.g., 3-hour, 6-hour).
    - Helps smooth out short-term fluctuations and capture trends.

9.3. Interaction Features

- **Combined Features:**
    - Product or ratio of features like `Temperature × Humidity`, `Wind Speed / PM2.5`
    - May reveal non-linear relationships.

9.4. Categorical Encoding

- **One-Hot Encoding:**

    - For features like day of the week or weather conditions.

- **Cyclic Encoding:**
    - Encode time-based features (hour, month) using sine and cosine transforms to capture cyclical nature.

Example:

```
df['hour_sin'] = np.sin(2 * np.pi * df['hour'] / 24)
df['hour_cos'] = np.cos(2 * np.pi * df['hour'] / 24)
```

    -

9.5. Weather-Based Features

- **Pollutant Dispersion Conditions:**
    - Use wind speed/direction, temperature inversion flags (if applicable), or humidity to estimate pollutant retention.
- **Heat Index or Dew Point:**
    - Derived metrics combining temperature and humidity.

9.6. Location-Based Features (if applicable)

- **Station Metadata:**
    - Urban vs rural, elevation, proximity to industrial zones or highways.
- **Encoded Geolocation:**
    - Latitude and longitude (or clustered zones) to capture spatial variation.

9.7. Target Variable Engineering

- **Classification:**

    - Convert AQI into categories (e.g., Good, Moderate, Unhealthy).

- **Regression:**
    - Use continuous AQI or pollutant concentration values directly.

---

10. Model Building

The model building phase involves selecting, training, tuning, and evaluating machine learning algorithms to predict air quality levels accurately. A variety of models are tested to determine the most effective approach for the dataset and problem type (regression or classification).

10.1. Problem Type

- **Regression:** Predicting continuous values such as AQI, PM2.5, or PM10 concentrations.
- **Classification (optional):** Categorizing AQI levels (e.g., Good, Moderate, Unhealthy).

10.2. Model Selection

Several machine learning algorithms are explored to determine which provides the best performance:

Traditional Machine Learning Models

- **Linear Regression:** Baseline model for understanding linear relationships.
- **Decision Tree Regressor:** Interpretable, handles non-linear data well.
- **Random Forest Regressor:** Ensemble model that reduces overfitting and improves accuracy.
- **Gradient Boosting Regressor (XGBoost / LightGBM):** High-performance model for capturing complex interactions.

Deep Learning Models (Optional)

- **Artificial Neural Networks (ANN):** Can model complex nonlinear relationships.
- **Long Short-Term Memory (LSTM):** Ideal for time-series forecasting of AQI using sequential historical data.

10.3. Model Training

- **Data Splitting:** Dataset is split into training and testing sets (e.g., 80/20 or 70/30).
- **Cross-Validation:** K-fold or time-series cross-validation is used to avoid overfitting and ensure model generalization.
- **Hyperparameter Tuning:** GridSearchCV or RandomizedSearchCV is applied for finding the best parameters.

10.4. Model Evaluation

- Models are assessed using performance metrics appropriate for regression tasks:
    - **Mean Absolute Error (MAE)**
    - **Root Mean Squared Error (RMSE)**
    - **R^2 Score (Coefficient of Determination)**

Example:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score

mae = mean_absolute_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
```

11. Model Evaluation

Model evaluation is a critical step to assess the predictive performance, robustness, and generalizability of the trained machine learning models. It ensures that the selected model provides accurate and reliable air quality forecasts.

11.1. Evaluation Metrics

To evaluate the models, the following metrics are commonly used for regression problems like air quality prediction:

11.2. Cross-Validation

- **K-Fold Cross-Validation:**
  The dataset is split into *k* subsets. Each subset is used once as a test set while the others serve as training data, ensuring the model's robustness across different data splits.

- **Time-Series Cross-Validation:**
  Since air quality data is temporal, training on past data and testing on future periods preserves the temporal order and avoids data leakage.

11.3. Residual Analysis

- Plot residuals (errors) to check for patterns or biases.

- Residuals should ideally be randomly distributed without obvious trends.

11.4. Feature Importance Analysis

- For tree-based models like Random Forest or XGBoost, extract and visualize feature importances.

- Helps identify key drivers of air quality changes and validates domain knowledge.

---

12. Deployment

Deployment is the phase where the trained machine learning model is integrated into a real-world environment to provide actionable air quality predictions. The goal is to make the predictive system accessible, reliable, and scalable for end-users such as environmental agencies, policymakers, and the general public.

12.1. Deployment Objectives

- Provide real-time or near-real-time air quality predictions.
- Enable easy access through web or mobile applications.
- Ensure scalability to handle data from multiple regions or sensors.
- Facilitate continuous monitoring and updating of models.

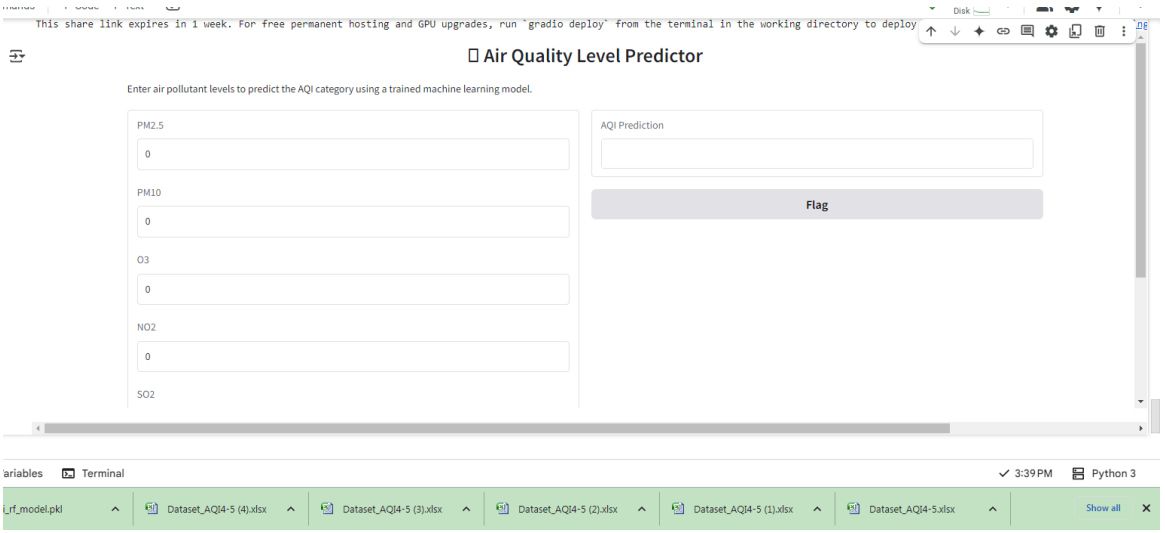
12.2. Deployment Architecture

- **Model Serving:**
  Host the trained model using platforms like Flask, FastAPI, or TensorFlow Serving to expose prediction APIs.
- **Data Pipeline:**
  Automate data ingestion from environmental sensors, weather APIs, or databases, with preprocessing steps integrated.
- **Frontend Interface:**
  Develop dashboards or mobile apps displaying predicted air quality indices, pollutant concentrations, and historical trends.

12.3. Implementation Steps

- **Export Model:**
  Save the final trained model using formats like Pickle (`.pkl`), Joblib, or ONNX for interoperability.
- **API Development:**
  Build RESTful APIs that accept input data and return air quality predictions.
- **User Interface:**
  Create visualization tools using libraries such as Dash, Streamlit, or React.js to display predictions and alerts.
- **Cloud Hosting (Optional):**
  Deploy the system on cloud services like AWS, Azure, or Google Cloud for scalability and uptime.
- **Monitoring & Logging:**
  Implement monitoring to track prediction accuracy, data anomalies, and system health. Log user interactions and prediction outcomes.`public`

- public URL: https://f11b051589882cf5be.gradio.liv



---

13. Source Code

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler

# --- Load Dataset ---
data = pd.read_csv('air_quality_data.csv')  # Replace with your dataset path

# --- Basic Preprocessing ---
# Handle missing values by forward fill
data.fillna(method='ffill', inplace=True)

# Feature Engineering: Extract temporal features
data['timestamp'] = pd.to_datetime(data['timestamp'])
data['hour'] = data['timestamp'].dt.hour
data['day_of_week'] = data['timestamp'].dt.dayofweek
data['month'] = data['timestamp'].dt.month

# Select features and target variable
features = ['PM10', 'NO2', 'CO', 'O3', 'SO2', 'temperature', 'humidity', 'wind_speed', 'hour',
'day_of_week', 'month']
X = data[features]
```

```
y = data['PM2.5']  # Target variable

# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42, shuffle=False)

# --- Model Building ---
rf = RandomForestRegressor(random_state=42)

# Hyperparameter tuning (optional)
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5]
}

grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3,
scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)

best_rf = grid_search.best_estimator_

# --- Model Evaluation ---
y_pred = best_rf.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)

print(f"MAE: {mae:.3f}")
print(f"RMSE: {rmse:.3f}")
print(f"R² Score: {r2:.3f}")

# Save the model
import joblib
joblib.dump(best_rf, 'rf_air_quality_model.pkl')
```

14. Future Scope

The field of air quality prediction using machine learning is rapidly evolving, with many opportunities for enhancement and expansion. Future work can build upon the current system to increase accuracy, scalability, and real-world impact.

14.1. Integration of Additional Data Sources

- Incorporate satellite data and remote sensing to capture wider spatial pollution trends.
- Use real-time traffic, industrial activity, and social media data for dynamic pollution monitoring.
- Integrate IoT sensor networks for hyper-local and high-frequency data collection.

14.2. Advanced Modeling Techniques

- Explore deep learning architectures such as Convolutional Neural Networks (CNNs) and Transformers for spatial-temporal modeling.
- Utilize hybrid models combining physical atmospheric simulations with machine learning predictions.
- Implement transfer learning to adapt models across different geographical regions with limited labeled data.

14.3. Real-Time Prediction and Alert Systems

- Develop streaming data pipelines for continuous real-time air quality forecasting.
- Build automated alert systems to notify vulnerable populations about hazardous air quality levels.

---

13. Team Members and Roles

Team Member	Role Description
P. Pavithra	Project Coordinator & Data Preprocessing Lead
M. Monisha	Machine Learning Model Developer
A. Revathi	Exploratory Data Analysis & Feature Engineering
M. Priyadharshini	Deployment & Documentation Specialist