

Phase-3

Student Name: Abinaya B

Register Number: 410723104003

Institution: Dhanalakshmi College of Engineering

Department: Computer Science and Engineering

Date of Submission: 13-05-2025

Github Repository Link: <https://github.com/Abinaya02-06/nm-abinaya>

Predicting air quality levels using advanced Machine Learning algorithms for Environmental Insights

1. Problem Statement

This project aims to accurately predict air quality levels by leveraging advanced machine learning models on environmental and meteorological data. Develop a machine learning model to accurately forecast Air Quality Index (AQI) and pollutant concentrations, enabling early warnings, informed decision-making, and improved public health.

2. Abstract

- To develop a predictive model for air quality using historical environmental data, focusing on key pollutants such as PM2.5, PM10, NO2, and CO levels.
- The primary objective is to accurately forecast air quality index (AQI) values to support timely decision-making and public health advisories.
- The approach involves data collection from publicly available air quality datasets, followed by preprocessing, feature selection, and model training using machine learning algorithms such as Random Forest and Gradient Boosting.
- This work contributes to proactive air quality management and can be integrated into real-time monitoring systems.

3. System Requirements

- **Hardware:**

Processor: Intel Core i5 or higher (or equivalent AMD processor)

RAM: Minimum 8 GB (16 GB recommended for large datasets)

Storage: At least 50 GB free space (for datasets, models, and software)

- **Software:**

Programming Language: Python

Data Processing: Pandas, NumPy

Visualization: Matplotlib, Seaborn, Plotly

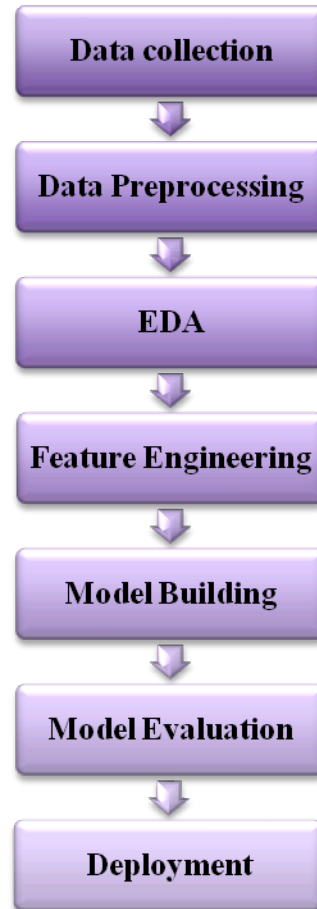
Machine Learning: Scikit-learn, XGBoost.

Environment: jupyter Anaconda, google colab

4. Objectives

The objective of air quality prediction using machine learning is to accurately forecast pollution levels (like PM2.5, PM10, NO2, CO, etc.) in the air based on various environmental, weather, and traffic-related features. Estimate future levels of pollutants such as PM2.5, PM10, NO2, SO2, CO, and O3. This helps in early warning systems for public health. Use feature importance to identify which factors (e.g., temperature, humidity, wind speed, traffic levels) most affect air quality. Provide forecasts to the public via apps or websites so they can take precautions.

5. Flowchart of Project Workflow:



6. Dataset Description

- **Source:** Kaggle-AirQualityPrediction
- **Type :** public
- **Size and structure:** 4000 rows and 23 columns
- Dataset link: <https://www.kaggle.com/datasets/khushikyad001/air-quality-data/data>

```
df.head()
```

	Date	Time	CO(GT)	NOx(GT)	NO2(GT)	O3(GT)	SO2(GT)	PM2.5	PM10	Temperature	...	WindDirection	CO_NOx_Ratio	NOx_NO2_Ratio	Temp_Humidity_Index	AirQualityIndex
0	2024-01-01	00:00	3.807947	172.026768	144.333317	118.120832	1.215679	147.349671	208.803124	28.564580	...	209.984267	0.022008	1.183671	3.541778	343.353046
1	2024-01-01	01:00	9.512072	241.824266	137.769318	15.325830	1.016178	40.979839	145.595579	6.793192	...	319.534890	0.039173	1.742635	0.727989	206.282028
2	2024-01-01	02:00	7.346740	228.288118	20.055086	44.377036	24.140910	72.594740	26.155000	24.436552	...	274.644300	0.032042	10.842422	7.378322	140.170920
3	2024-01-01	03:00	6.026719	47.016072	184.591909	139.488603	2.435392	134.339724	276.367944	26.463951	...	312.266023	0.125515	0.253330	21.684266	307.928588
4	2024-01-01	04:00	1.644585	45.625591	114.125968	95.634768	48.752095	99.007422	294.295449	10.530331	...	21.392120	0.035272	0.396310	9.627596	370.134556

5 rows × 23 columns

7. Data Preprocessing

- Remove duplicates and handled missing values.
- Encoded categorical variables using Label Encoding.
- Scaled numeric features using Standard Scaler.
- Description of data:

```
df.describe()
```

	CO(GT)	NOx(GT)	NO2(GT)	O3(GT)	SO2(GT)	PM2.5	PM10	Temperature	Humidity	Pressure	...	WindDirection	CO_NOx_Ratio	NOx_NO2_Ratio	Temp_Humidity_In
count	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	...	4000.000000	4000.000000	4000.000000	4000.00
mean	5.025385	148.126633	100.213189	89.914815	26.081045	104.765999	153.591417	17.305228	54.626284	999.862679	...	179.571724	0.082564	3.412176	9.41
std	2.874632	85.999247	57.074947	52.003484	14.059684	56.344868	83.080911	12.943632	25.844003	28.897118	...	104.738760	0.215593	7.996062	8.98
min	0.100115	1.009185	1.010513	1.055442	1.012370	5.009384	10.031967	-4.996963	10.000498	950.018004	...	0.033694	0.000389	0.006021	-4.82
25%	2.514242	73.636615	51.326622	44.179487	14.220565	56.544378	82.200105	6.092531	31.970628	975.018939	...	86.409827	0.017033	0.719052	2.55
50%	5.054973	146.440690	99.508855	88.956924	26.321359	105.502686	154.714484	17.184773	55.113650	999.857722	...	179.401393	0.034021	1.467460	6.97
75%	7.524652	221.823697	149.666167	136.333683	37.833728	153.751364	222.969347	28.573093	76.311009	1024.763352	...	272.370032	0.065973	2.789423	14.75
max	9.997205	299.838744	199.934968	179.986544	49.993700	199.980691	299.911727	39.987944	99.981043	1049.926869	...	359.984504	4.078615	131.169948	39.37

8 rows × 21 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 4000 entries, 0 to 3999
```

```
Data columns (total 23 columns):
```

#	Column	Non-Null Count	Dtype
0	Date	4000 non-null	object
1	Time	4000 non-null	object
2	CO(GT)	4000 non-null	float64
3	NOx(GT)	4000 non-null	float64
4	NO2(GT)	4000 non-null	float64
5	O3(GT)	4000 non-null	float64
6	SO2(GT)	4000 non-null	float64
7	PM2.5	4000 non-null	float64
8	PM10	4000 non-null	float64
9	Temperature	4000 non-null	float64
10	Humidity	4000 non-null	float64
11	Pressure	4000 non-null	float64
12	WindSpeed	4000 non-null	float64
13	WindDirection	4000 non-null	float64
14	CO_NOx_Ratio	4000 non-null	float64
15	NOx_NO2_Ratio	4000 non-null	float64
16	Temp_Humidity_Index	4000 non-null	float64
17	AirQualityIndex	4000 non-null	float64
18	CO_MA3	4000 non-null	float64
19	NO2_MA3	4000 non-null	float64
20	O3_MA3	4000 non-null	float64
21	DayOfWeek	4000 non-null	int64
22	Hour	4000 non-null	int64

```
dtypes: float64(19), int64(2), object(2)
```

```
memory usage: 718.9+ KB
```

- **Handle missing values:**

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

Date	0
Time	0
CO(GT)	0
NOx(GT)	0
NO2(GT)	0
O3(GT)	0
SO2(GT)	0
PM2.5	0
PM10	0
Temperature	0
Humidity	0
Pressure	0
WindSpeed	0
WindDirection	0
CO_NOx_Ratio	0
NOx_NO2_Ratio	0
Temp_Humidity_Index	0
AirQualityIndex	0
CO_MA3	0
NO2_MA3	0
O3_MA3	0
DayOfWeek	0
Hour	0
dtype:	int64

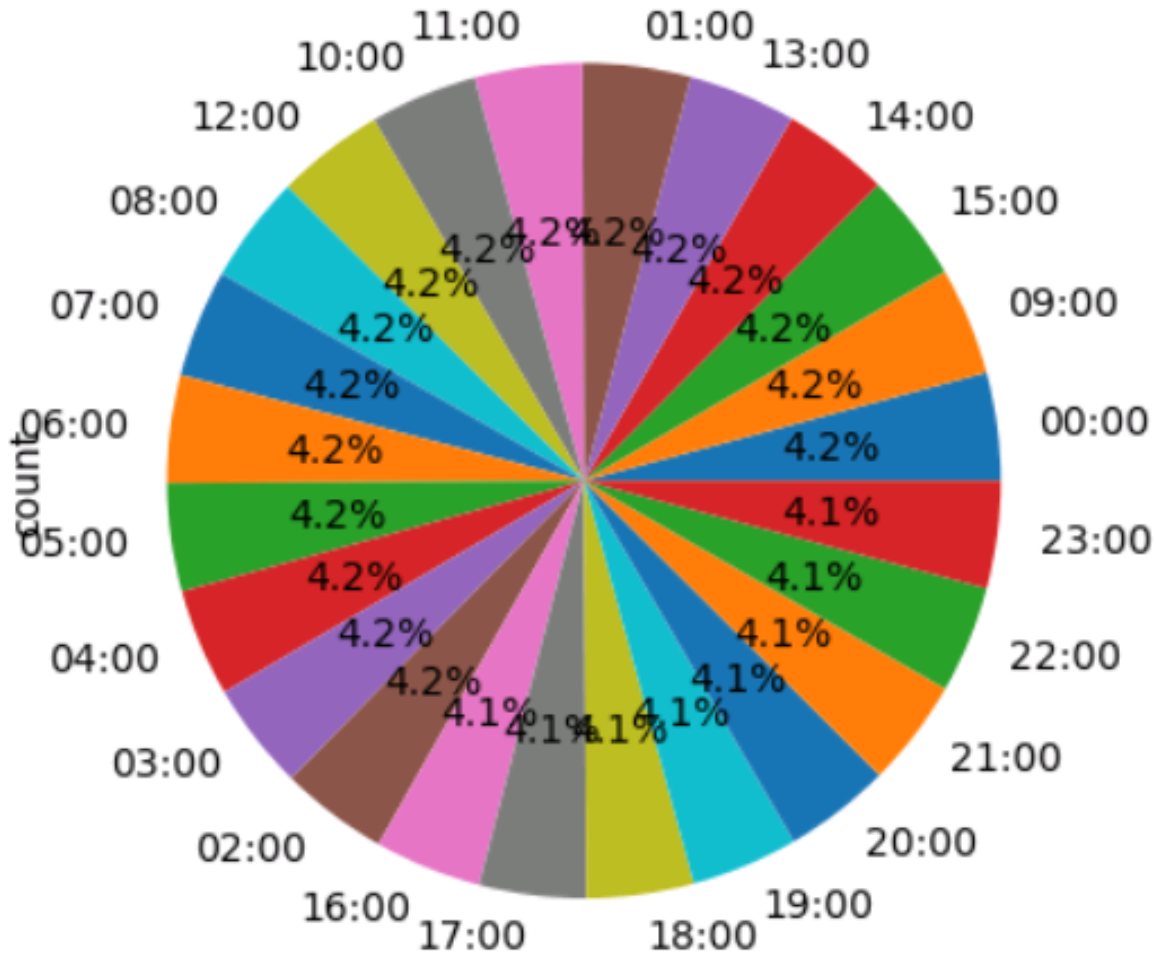
- Code to check for Duplicate values:

```
: print(df.duplicated().sum())
```

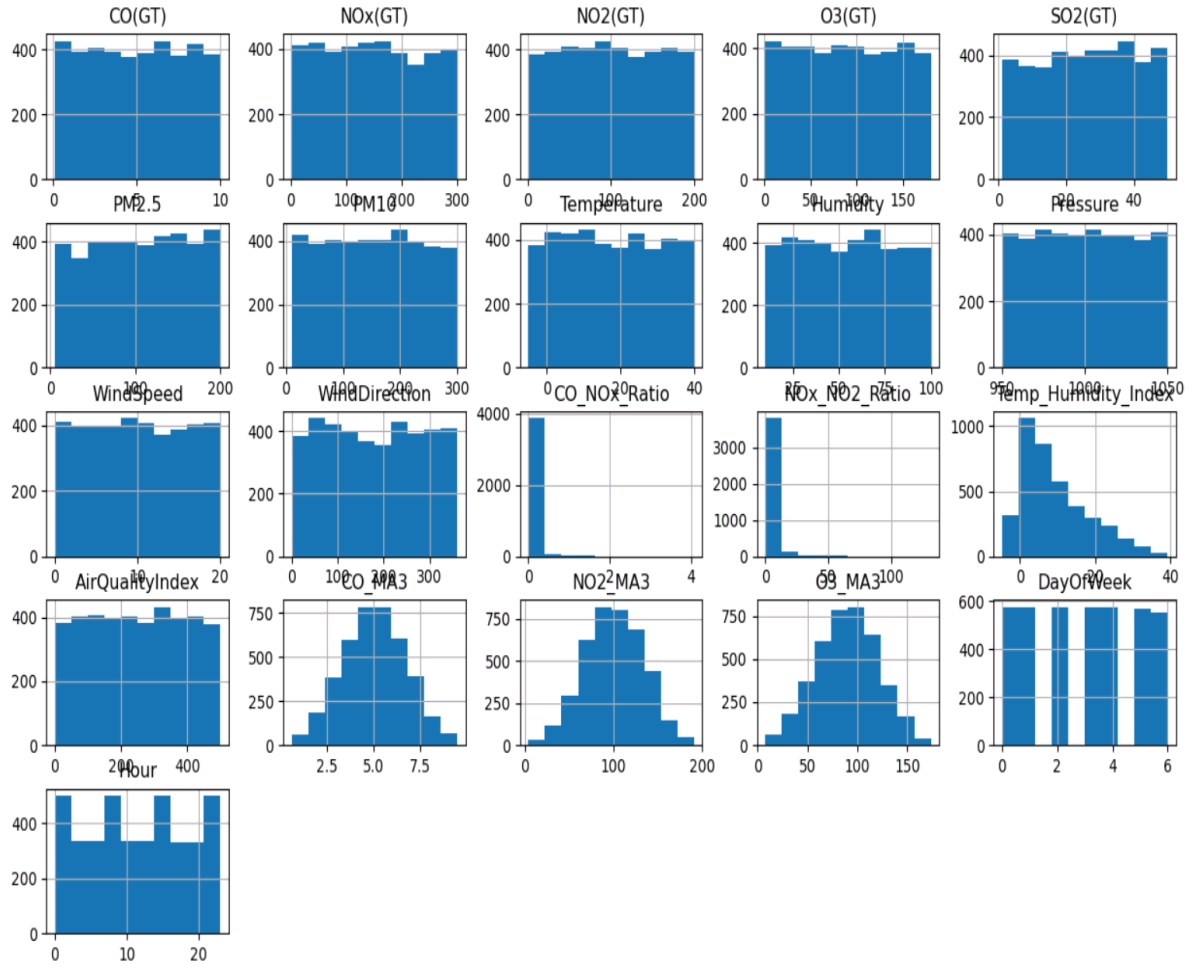
0

8. Exploratory Data Analysis (EDA)

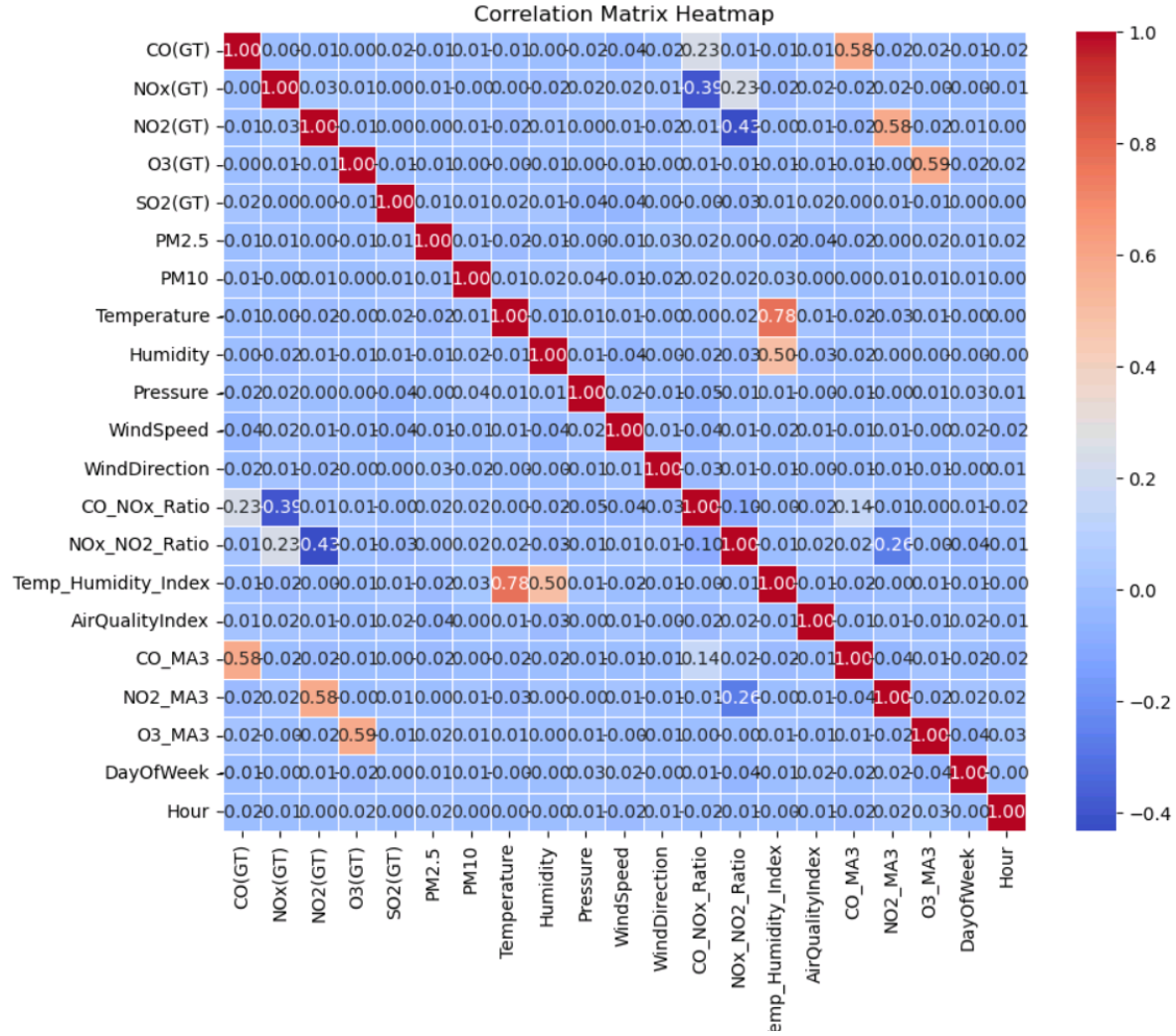
- Pie Chart Visualisation:



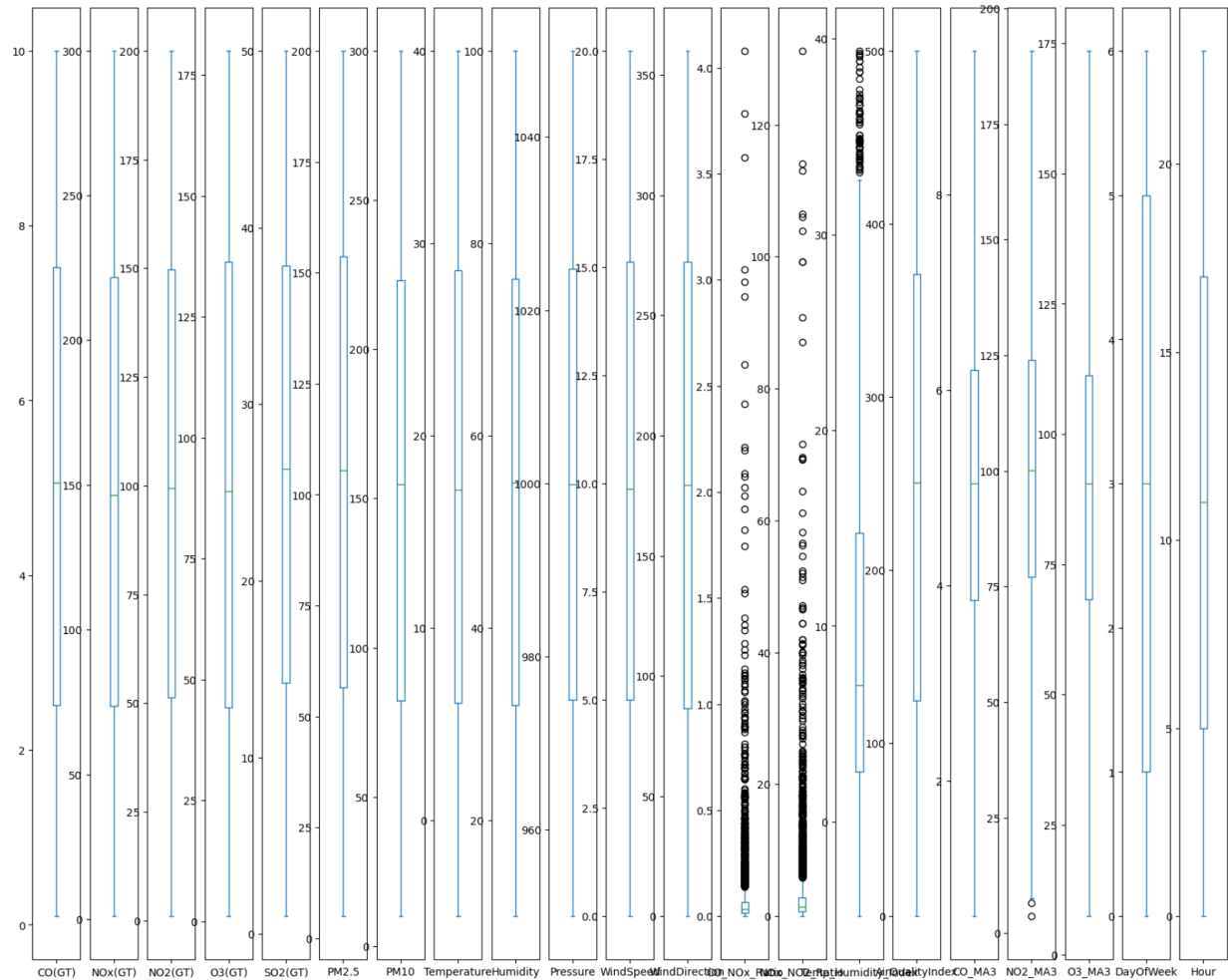
- Histogram visualization:



- **Correlation Matrix Heatmap:**



- Box Plot Visualisation:



9. Feature Engineering

- Create new features to improve model accuracy. Bar plots or box plots of

newly created features (e.g., pollutant ratios like PM2.5/PM10).

- plot pollution level by hour of day, day of week. Helps see if certain times are more polluted.
- Combined weighted score of multiple pollutants.

```
#standard scaler
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df[['Temperature', 'Pressure', 'Humidity']] = scaler.fit_transform(df[['Temperature', 'Pressure', 'Humidity']])
df[['Temperature', 'Pressure', 'Humidity']].head()
```

	Temperature	Pressure	Humidity
0	0.864726	1.436303	-1.636355
1	-0.816229	1.560145	-1.701307
2	0.562484	-1.712565	-0.930544
3	0.719234	-0.620507	1.063046
4	-0.502728	1.159175	1.414653

10. Model Building

Air quality prediction is a critical task that involves forecasting the concentration of pollutants in the air, such as particulate matter (PM), nitrogen dioxide (NO₂), ozone (O₃), and others. Select and implement at least 2 Machine learning. E.g., Logistic Regression, Decision Tree, Random Forest, KNN, etc.

```
# Random Forest Algorithm
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

df = pd.read_csv("AirQualityData.csv")

le = LabelEncoder()
for col in df.select_dtypes(include='object').columns:
    df[col] = le.fit_transform(df[col])

X = df.drop('Time', axis=1)
y = df['Time']

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

rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

Accuracy: 0.94625

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	43
1	1.00	1.00	1.00	35
2	0.97	1.00	0.99	37
3	1.00	0.97	0.98	30
4	0.94	1.00	0.97	33
5	0.97	0.97	0.97	30
6	1.00	0.93	0.96	44
7	0.96	0.96	0.96	25
8	0.90	1.00	0.95	37
9	0.85	0.85	0.85	34
10	0.91	0.81	0.86	37
11	0.77	0.84	0.81	32
12	0.92	0.92	0.92	24
13	1.00	0.94	0.97	35
14	0.82	0.96	0.89	28
15	0.91	0.88	0.90	34
16	0.91	0.91	0.91	32
17	0.91	0.86	0.89	36
18	0.97	0.92	0.95	39
19	1.00	1.00	1.00	33
20	1.00	1.00	1.00	32
21	1.00	1.00	1.00	27
22	1.00	1.00	1.00	32
23	1.00	1.00	1.00	31
accuracy			0.95	800
macro avg	0.95	0.95	0.95	800
weighted avg	0.95	0.95	0.95	800

#Scaler standarisation:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
mae = mean_absolute_error(y_test,y_pred)
mse = mean_squared_error(y_test,y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("R-squared (R²):",r2)
```

Mean Absolute Error (MAE): 3.8

Mean Squared Error (MSE): 22.525

R-squared (R²): 0.5378960499472312

#KNN Algorithm

```
#svm algorithm with confusion matrix
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder

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)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

Accuracy: 0.00375
Confusion Matrix:
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

#SVM algorithm


```
#svm algorithm with confusion matrix
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder

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)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

Accuracy: 0.00375
Confusion Matrix:
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

11. Model Evaluation

- **Evaluation metrics:**

- accuracy:94%
- F1-score:0.95

- **Visuals:**

- Confusion matrix
- Classification report
- Precision

- Recall

- **K means cluster:**

```
#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_)

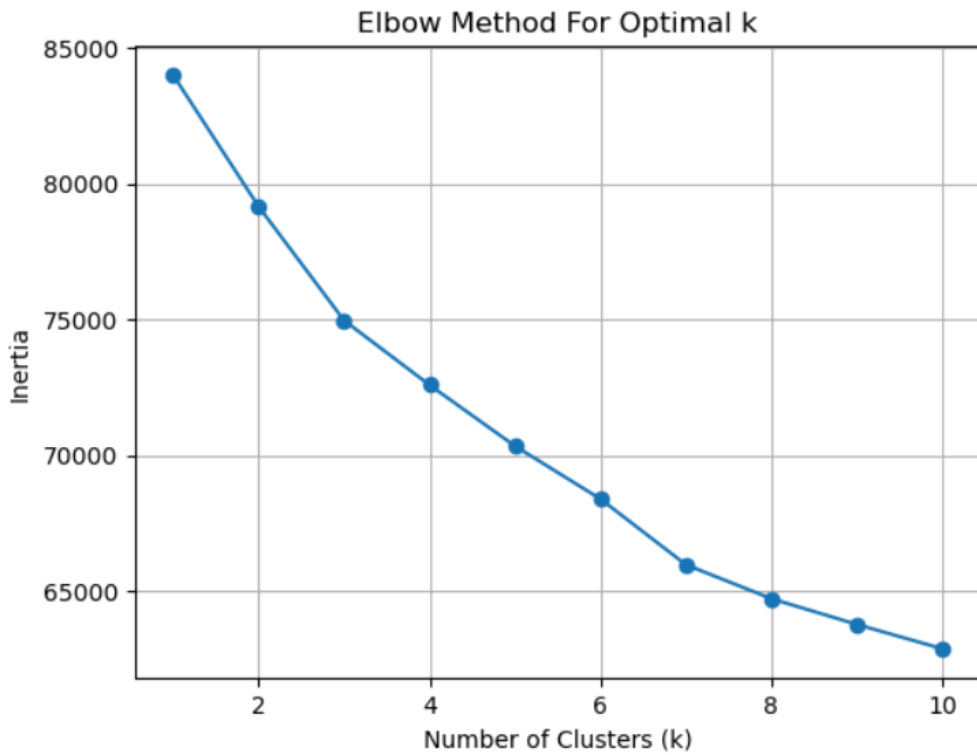
plt.plot(k_range, inertia, marker='o')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method For Optimal k')
plt.grid(True)
plt.show()

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)

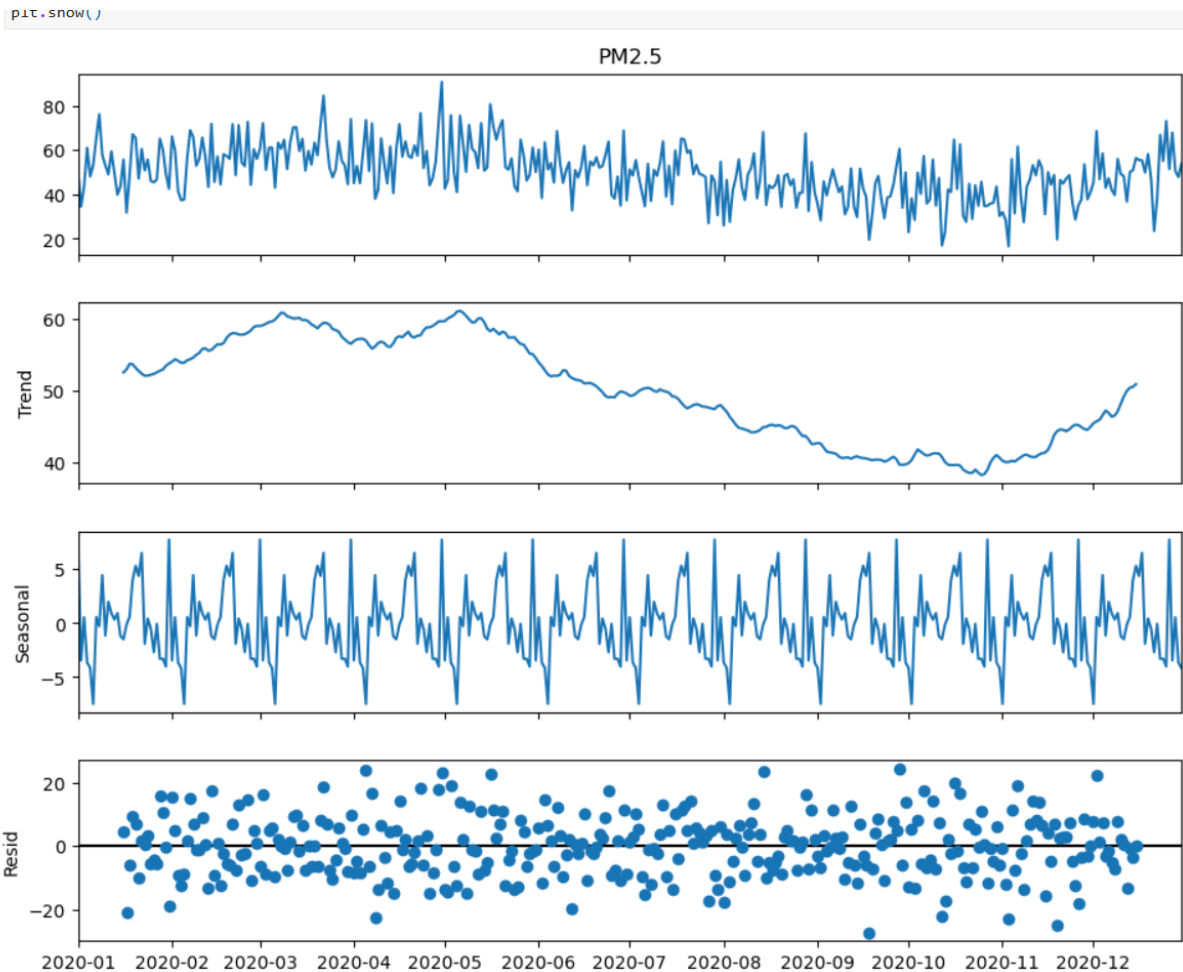
plt.figure(figsize=(8, 5))
plt.scatter(reduced_data[:, 0], reduced_data[:, 1], c=df['Cluster'], cmap='viridis')
plt.title('K-Means Clusters (PCA 2D)')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.colorbar(label='Cluster')
plt.show()
```



```
Cluster
0      1522
1      1350
2      1128
Name: count, dtype: int64
```



#Time series Composition:



12. Deployment

The trained air quality prediction model was deployed as a REST API using Flask. The deployment process involved the following steps:

- **Model Saving:**

After training, the model was saved using the joblib library for future use

without retraining.

- **API Development:**

A lightweight Flask application was created to serve predictions. The app accepts input parameters (such as temperature, humidity, etc.) via POST requests and returns the predicted PM2.5 level.

- **Testing Locally:**

The API was run locally on localhost:5000 to ensure the model correctly receives input and returns valid predictions.

13. Source code

```
import pandas as pd

from sklearn.model_selection

import train_test_split from sklearn.preprocessing

import LabelEncoder, StandardScaler

from sklearn.ensemble

import RandomForestClassifier from sklearn.metrics

import classification_report, confusion_matrix

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read_csv('AirQualityData.csv')

df.head( )

df.info( )
```

```
df.describe( )
```

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

```
df.drop_duplicates( )
```

```
df.drop_duplicates( ).sum( )
```

#Box Plot:

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('AirQualityData.csv')
```

```
df.plot(kind='box', figsize=(20,16), subplots=True, layout=(1,  
len(df.select_dtypes(include='number').columns)), sharey=False)
```

```
plt.show()
```

#Histogram:

```
import matplotlib.pyplot as plt
```

```
df.hist(figsize=(15,9))
```

```
plt.show()
```

#Pie chart:

```
import matplotlib.pyplot as plt
```

```
df['Time'].value_counts().plot(kind='pie', autopct='%1.1f%%')
```

```
plt.show()
```

#Correlation Heat Matrix:

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

corr_matrix = df.corr(numeric_only=True)

plt.figure(figsize=(10, 8))

sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f",
linewidths=0.5)

plt.title('Correlation Matrix Heatmap')

plt.show()
```

Scatter plot

```
plt.figure(figsize=(10,6))

sns.scatterplot(x=df['Date'], y=df['CO(GT)'], alpha=0.3)

plt.title('Date vs CO(GT)')

plt.xlabel('Date')

plt.ylabel('CO(GT)')

plt.show()
```

#Time series Decomposition

```
import pandas as pd

import matplotlib.pyplot as plt
```

```
from statsmodels.tsa.seasonal import seasonal_decompose

date_range = pd.date_range(start='2020-01-01', periods=365, freq='D')

import numpy as np

data = np.random.normal(loc=50, scale=10, size=365) + np.sin(np.linspace(0,
3.14*2, 365))*10

df = pd.DataFrame({'date': date_range, 'PM2.5': data })

df.set_index('date', inplace=True)

decomposition = seasonal_decompose(df['PM2.5'], model='additive', period=30)

fig = decomposition.plot()

fig.set_size_inches(10,8)

plt.show()
```

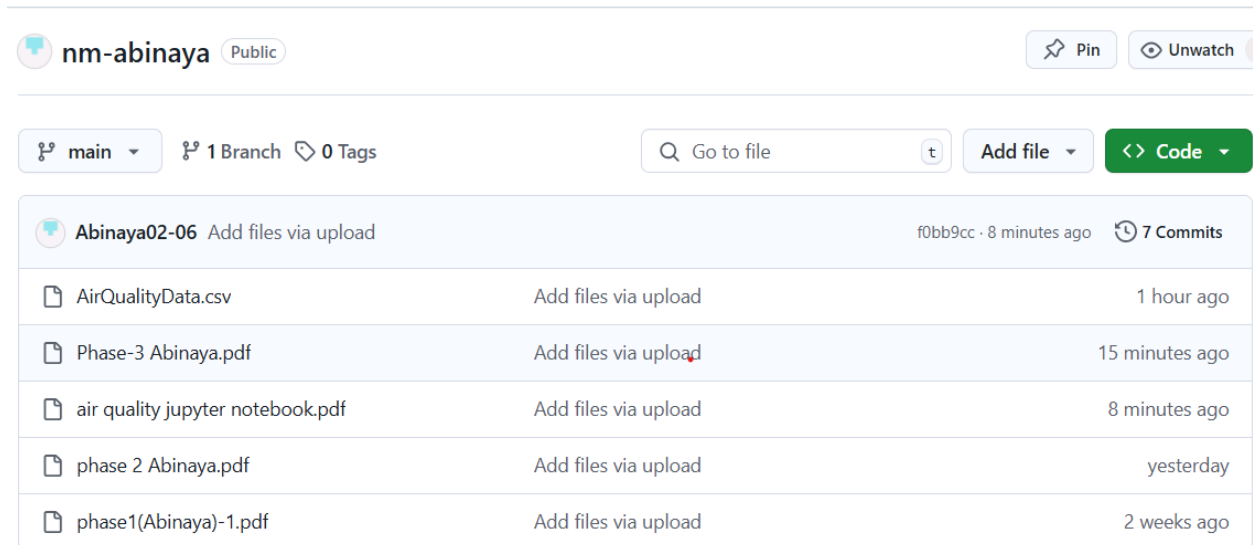
14. Future scope

- **Real-time Prediction and Monitoring:** The model can be integrated with IoT-enabled air quality sensors to provide real-time AQI forecasting and alerts for specific regions.
- **Expansion to More Pollutants and Weather Factors:** Future versions can incorporate additional pollutants (like O_3 , SO_2) and meteorological parameters (wind speed, humidity) to improve prediction accuracy.
- **Geospatial Analysis:** Adding GIS capabilities can help visualize AQI across regions, aiding in targeted environmental policy-making.
- **Mobile and Web Applications:** A user-friendly interface (dashboard or app) can be developed for the public to monitor air quality and receive health advisories.

13. Team Members and Roles

NAMES	ROLES	RESPONSIBILITY
Narmadha D	Member	Data collection and Preprocessing
Mohana priya K	Member	EDA and Feature Engineering
Abinaya B	Leader	Model building and Evaluation
Hema kanitha S	Member	Data Visualisation and Interpretation

GITHUB SCREENSHOT:



The screenshot shows a GitHub repository page for 'nm-abinaya' (Public). The repository has 1 Branch (main) and 0 Tags. The commit history shows a commit 'Abinaya02-06' made 8 minutes ago with 7 commits. The files listed in the commit are:

- AirQualityData.csv (Add files via upload, 1 hour ago)
- Phase-3 Abinaya.pdf (Add files via upload, 15 minutes ago)
- air quality jupyter notebook.pdf (Add files via upload, 8 minutes ago)
- phase 2 Abinaya.pdf (Add files via upload, yesterday)
- phase1(Abinaya)-1.pdf (Add files via upload, 2 weeks ago)

GOOGLE COLAB LINK:

https://colab.research.google.com/drive/1Dh0QAJ65_lMbJtlociST-Y6bjKGbMB5R?usp=sharing