

```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import r2_score, root_mean_squared_error,
mean_absolute_error
import pickle
```

```
df=pd.read_csv('Datasets/final.csv')
df.head(5)
```

	Datetime	City	PM2.5	PM10	NO	NO2	NOx	NH3
0	2020-03-12 13:00:00	Aizawl	25.0	31.11	7.14	1.86	11.28	24.00
1	2020-03-12 14:00:00	Aizawl	19.0	29.17	7.32	1.15	10.85	27.59
2	2020-03-12 15:00:00	Aizawl	24.0	30.00	7.14	1.04	10.51	31.13
3	2020-03-12 16:00:00	Aizawl	25.0	32.08	7.20	1.19	10.74	33.31
4	2020-03-12 17:00:00	Aizawl	33.0	41.00	7.22	1.37	10.93	30.05

	S02	03	Benzene	Toluene	AQI	AQI_Bucket
0	4.31	0.76	1.5	4.33	51.0	Satisfactory
1	4.65	0.07	1.5	4.33	52.0	Satisfactory
2	4.83	0.67	1.5	4.33	52.0	Satisfactory
3	5.26	0.05	1.5	4.33	53.0	Satisfactory
4	5.39	0.02	1.5	4.33	54.0	Satisfactory

```
df.shape
```

```
(239322, 15)
```

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

Datetime	10654
City	0
PM2.5	0
PM10	0
NO	0
NO2	0
NOx	0
NH3	0

```
C0          0
S02         0
O3          0
Benzene     0
Toluene     0
AQI         0
AQI_Bucket  0
dtype: int64
```

```
df.describe()
```

	PM2.5	PM10	NO	N02 \
count	239322.000000	239322.000000	239322.000000	239322.000000
mean	61.813510	125.092535	18.531356	33.339782
std	62.803755	103.770181	33.829649	25.601033
min	0.010000	0.010000	0.010000	0.010000
25%	25.550000	58.000000	3.800000	15.340000
50%	44.120000	95.800000	7.980000	26.880000
75%	72.980000	153.190000	16.770000	43.930000
max	999.990000	1000.000000	498.970000	380.020000

	NOx	NH3	C0	S02 \
count	239322.000000	239322.000000	239322.000000	239322.000000
mean	36.709078	23.418141	1.039834	11.270571
std	40.002762	18.960396	1.464409	10.426656
min	0.010000	0.010000	0.010000	0.010000
25%	14.780000	11.140000	0.500000	5.530000
50%	24.490000	18.050000	0.740000	8.640000
75%	41.900000	30.600000	1.090000	13.330000
max	493.400000	485.820000	47.420000	199.930000

	O3	Benzene	Toluene	AQI
count	239322.000000	239322.000000	239322.000000	239322.000000
mean	38.721834	4.141433	10.064720	143.422594
std	29.183399	21.101855	23.906228	99.630428
min	0.010000	0.010000	0.010000	8.000000
25%	17.810000	0.630000	1.820000	76.000000
50%	31.050000	1.500000	4.330000	110.000000
75%	52.430000	3.630000	10.910000	173.000000
max	497.620000	498.070000	498.070000	762.000000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 239322 entries, 0 to 239321
Data columns (total 15 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Datetime    228668 non-null object
1   City        239322 non-null object
```

```

2   PM2.5      239322 non-null float64
3   PM10      239322 non-null float64
4   NO        239322 non-null float64
5   NO2       239322 non-null float64
6   NOx       239322 non-null float64
7   NH3       239322 non-null float64
8   CO        239322 non-null float64
9   SO2       239322 non-null float64
10  O3        239322 non-null float64
11  Benzene   239322 non-null float64
12  Toluene   239322 non-null float64
13  AQI       239322 non-null float64
14  AQI_Bucket 239322 non-null object

```

dtypes: float64(12), object(3)

memory usage: 27.4+ MB

#Handle missing Datetime values

```
df = df.dropna(subset=['Datetime'])
```

Convert Datetime to datetime format

```
df['Datetime'] = pd.to_datetime(df['Datetime'])
```

Extract temporal features

```
df['hour'] = df['Datetime'].dt.hour
```

```
df['day'] = df['Datetime'].dt.day
```

```
df['month'] = df['Datetime'].dt.month
```

```
df['weekday'] = df['Datetime'].dt.weekday
```

```
df['weekofyear'] = df['Datetime'].dt.isocalendar().week
```

```
df['dayofyear'] = df['Datetime'].dt.dayofyear
```

```
df.info()
```

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

```
Index: 228668 entries, 0 to 228667
```

```
Data columns (total 21 columns):
```

#	Column	Non-Null Count	Dtype
0	Datetime	228668 non-null	datetime64[ns]
1	City	228668 non-null	object
2	PM2.5	228668 non-null	float64
3	PM10	228668 non-null	float64
4	NO	228668 non-null	float64
5	NO2	228668 non-null	float64
6	NOx	228668 non-null	float64
7	NH3	228668 non-null	float64
8	CO	228668 non-null	float64
9	SO2	228668 non-null	float64
10	O3	228668 non-null	float64
11	Benzene	228668 non-null	float64
12	Toluene	228668 non-null	float64

```
13  AQI                228668 non-null  float64
14  AQI_Bucket        228668 non-null  object
15  hour              228668 non-null  int32
16  day               228668 non-null  int32
17  month             228668 non-null  int32
18  weekday           228668 non-null  int32
19  weekofyear        228668 non-null  UInt32
20  dayofyear         228668 non-null  int32
dtypes: UInt32(1), datetime64[ns](1), float64(12), int32(5), object(2)
memory usage: 33.4+ MB
```

```
#One-hot encode cities
```

```
cities = df['City'].unique()
city_dummies = pd.get_dummies(df['City'], prefix='City')
df = pd.concat([df, city_dummies], axis=1)
```

```
# Drop unnecessary columns
```

```
df.drop(columns=['Datetime', 'City', 'AQI_Bucket'], inplace=True)
```

```
# Handle any remaining missing values
```

```
df.fillna(df.median(), inplace=True)
```

```
# Define features and target variable
```

```
X = df.drop(columns=['AQI'])
y = df['AQI']
```

```
#Splitting the dataset into 80-20 for train-test data
```

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

```
# Scale features
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# Linear Regression Model
```

```
lr=LinearRegression()
lr.fit(X_train_scaled,y_train)
print("Accuracy:- ",lr.score(X_test_scaled,y_test)*100)
```

```
Accuracy:- 78.41448881273578
```

```
# Decision Tree Model
```

```
dtm=DecisionTreeRegressor()
dtm.fit(X_train_scaled,y_train)
print("Accuracy:- ", dtm.score(X_test_scaled,y_test)*100)
```

```
Accuracy:- 82.17392214009945
```

```
# Random Forest Model
```

```
rfm=RandomForestRegressor(n_estimators=50,max_depth=10)
```

```

rfm.fit(X_train_scaled,y_train)
print("Accuracy:- ", rfm.score(X_test_scaled,y_test)*100)

Accuracy:- 85.90274367991026

# XGBoost Model
xgb = XGBRegressor(random_state=42)
xgb.fit(X_train_scaled, y_train)
print("Accuracy:- ", xgb.score(X_test_scaled,y_test)*100)

Accuracy:- 89.94964099538343

# Performance Metrics
y_pred = xgb.predict(X_test_scaled)
print('Root Mean Squared Error:', root_mean_squared_error(y_test,
y_pred))
print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))
print('R2 Score:', r2_score(y_test, y_pred))
print('Accuracy:', xgb.score(X_test_scaled,y_test)*100)

Root Mean Squared Error: 31.565413603954052
Mean Absolute Error: 21.128566954014637
R2 Score: 0.8994964099538343
Accuracy: 89.94964099538343

# Train Linear Regression models for pollutant forecasting
pollutant_cols = ['PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO',
'SO2', 'O3', 'Benzene', 'Toluene']
time_cols = ['hour', 'day', 'month', 'weekday', 'weekofyear',
'dayofyear']
pollutant_models = {}
for pollutant in pollutant_cols:
    model = LinearRegression()
    model.fit(df[time_cols], df[pollutant])
    pollutant_models[pollutant] = model

# Save models and scaler
with open('model.sav', 'wb') as f:
    pickle.dump(xgb, f)
with open('scaler.pkl', 'wb') as f:
    pickle.dump(scaler, f)

# Function to forecast pollutants
def forecast_pollutants(pollutant_models, input_datetime):
    dt = pd.to_datetime(input_datetime)
    features = pd.DataFrame([
        dt.hour, dt.day, dt.month,
        dt.weekday(), dt.isocalendar().week, dt.dayofyear
    ], columns=['hour', 'day', 'month', 'weekday', 'weekofyear',
'dayofyear'])
    predictions = {p: max(0, model.predict(features)[0]) for p, model

```

```

in pollutant_models.items(){}
    return predictions

# Function to forecast AQI
def predict_aqi_with_city(model, scaler, input_datetime, city,
**pollutants):
    cities = ['Aizawl', 'Amaravati', 'Amritsar', 'Bengaluru',
'Chandigarh', 'Chennai',
'Coimbatore', 'Delhi', 'Gurugram', 'Hyderabad',
'Jaipur', 'Kolkata',
'Patna', 'Shillong', 'Talcher', 'Visakhapatnam']

    #Extracting temporal features
    dt = pd.to_datetime(input_datetime)
    temporal_features = [
        dt.hour, dt.day, dt.month,
        dt.weekday(), dt.isocalendar().week, dt.dayofyear
    ]

    #Create city one-hot encoding
    city_data = [1 if c == city else 0 for c in cities]

    #Pollutant features
    pollutant_features = [
        pollutants.get('PM2.5', 0),
        pollutants.get('PM10', 0),
        pollutants.get('NO', 0),
        pollutants.get('NO2', 0),
        pollutants.get('NOx', 0),
        pollutants.get('NH3', 0),
        pollutants.get('CO', 0),
        pollutants.get('SO2', 0),
        pollutants.get('O3', 0),
        pollutants.get('Benzene', 0),
        pollutants.get('Toluene', 0)
    ]

    #Combine all features
    input_features = pollutant_features + temporal_features +
city_data

    #Scale and predict
    input_scaled = scaler.transform([input_features])
    predicted_aqi = model.predict(input_scaled)[0]

    #Classification of AQI Categories
    if predicted_aqi <= 50:
        bucket = "Good"
    elif predicted_aqi <= 100:
        bucket = "Satisfactory"

```

```

elif predicted_aqi <= 200:
    bucket = "Moderate"
elif predicted_aqi <= 300:
    bucket = "Poor"
elif predicted_aqi <= 400:
    bucket = "Very Poor"
else:
    bucket = "Severe"
return round(predicted_aqi, 2), bucket

#Predict AQI, its category and pollutant levels
city = 'Jaipur'
input_date = '2025-05-24'

#Forecast pollutant levels
predicted_pollutants = forecast_pollutants(pollutant_models,
input_date)

#Predict AQI using forecasted pollutants
aqi, bucket = predict_aqi_with_city(
    model=xgb,
    scaler=scaler,
    input_datetime=input_date,
    city=city,
    **predicted_pollutants
)

print(f"Predicted AQI for {city} on {input_date} is {aqi} µg/m³")
print("Air Quality Category:", bucket)
print(f"Forecasted pollutant levels are:")
for pollutant, value in predicted_pollutants.items():
    print(f" {pollutant}: {round(value, 2)} µg/m³")

```

Predicted AQI for Jaipur on 2025-05-24 is 104.70999908447266 µg/m³

Air Quality Category: Moderate

Forecasted pollutant levels are:

```

PM2.5: 45.89 µg/m³
PM10: 102.51 µg/m³
NO: 12.92 µg/m³
NO2: 22.52 µg/m³
NOx: 27.46 µg/m³
NH3: 20.35 µg/m³
CO: 0.9 µg/m³
SO2: 9.84 µg/m³
O3: 28.9 µg/m³
Benzene: 3.45 µg/m³
Toluene: 8.26 µg/m³

```

C:\Users\Guraa\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-

```
packages\Python311\site-packages\sklearn\utils\validation.py:2739:
UserWarning: X does not have valid feature names, but StandardScaler
was fitted with feature names
```

```
warnings.warn(
```

```
df.sample(10)
```

	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2
03 \								
24970	43.89	192.69	17.15	9.88	27.04	12.99	0.21	19.57
9.81								
168714	62.71	137.46	10.95	32.01	32.91	19.69	0.77	13.46
66.03								
213063	162.50	367.00	22.60	141.78	93.77	17.92	2.48	37.35
35.55								
170263	14.13	33.61	8.66	17.70	21.37	23.98	1.17	18.70
64.56								
186447	58.83	119.10	2.40	33.97	36.39	31.47	0.40	18.43
55.06								
112576	144.79	245.63	31.23	73.84	71.10	39.84	1.17	26.63
19.91								
23486	45.69	75.20	12.69	9.21	21.39	10.98	1.01	6.14
19.19								
86589	489.19	890.94	57.28	115.36	82.02	83.57	2.14	25.93
35.66								
17846	58.05	109.36	25.31	15.62	40.93	0.56	1.67	1.63
15.27								
149139	36.21	80.30	4.30	27.58	15.58	11.13	0.52	2.39
16.18								

	Benzene	...	City_Coimbatore	City_Delhi	City_Gurugram	\
24970	2.60	...	False	False	False	
168714	1.11	...	False	False	False	
213063	8.10	...	False	False	False	
170263	0.56	...	False	False	False	
186447	2.48	...	False	False	False	
112576	4.19	...	False	True	False	
23486	2.66	...	False	False	False	
86589	6.80	...	False	True	False	
17846	2.50	...	False	False	False	
149139	0.41	...	False	False	False	

	City_Hyderabad	City_Jaipur	City_Kolkata	City_Patna
City_Shillong \				
24970	False	False	False	False
False				
168714	False	True	False	False
False				
213063	False	False	False	False
False				

170263	False	True	False	False
False				
186447	False	False	True	False
False				
112576	False	False	False	False
False				
23486	False	False	False	False
False				
86589	False	False	False	False
False				
17846	False	False	False	False
False				
149139	True	False	False	False
False				

	City_Talcher	City_Visakhapatnam
24970	False	False
168714	False	False
213063	False	True
170263	False	False
186447	False	False
112576	False	False
23486	False	False
86589	False	False
17846	False	False
149139	False	False

[10 rows x 34 columns]

df.info()

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

Index: 228668 entries, 0 to 228667

Data columns (total 34 columns):

#	Column	Non-Null Count	Dtype
---	-----	-----	-----
0	PM2.5	228668 non-null	float64
1	PM10	228668 non-null	float64
2	NO	228668 non-null	float64
3	NO2	228668 non-null	float64
4	NOx	228668 non-null	float64
5	NH3	228668 non-null	float64
6	CO	228668 non-null	float64
7	SO2	228668 non-null	float64
8	O3	228668 non-null	float64
9	Benzene	228668 non-null	float64
10	Toluene	228668 non-null	float64
11	AQI	228668 non-null	float64
12	hour	228668 non-null	int32
13	day	228668 non-null	int32

```

14 month                228668 non-null int32
15 weekday              228668 non-null int32
16 weekofyear           228668 non-null UInt32
17 dayofyear            228668 non-null int32
18 City_Aizawl          228668 non-null bool
19 City_Amaravati       228668 non-null bool
20 City_Amritsar        228668 non-null bool
21 City_Bengaluru       228668 non-null bool
22 City_Chandigarh     228668 non-null bool
23 City_Chennai         228668 non-null bool
24 City_Coimbatore      228668 non-null bool
25 City_Delhi           228668 non-null bool
26 City_Gurugram        228668 non-null bool
27 City_Hyderabad       228668 non-null bool
28 City_Jaipur          228668 non-null bool
29 City_Kolkata         228668 non-null bool
30 City_Patna           228668 non-null bool
31 City_Shillong        228668 non-null bool
32 City_Talcher         228668 non-null bool
33 City_Visakhapatnam   228668 non-null bool
dtypes: UInt32(1), bool(16), float64(12), int32(5)
memory usage: 31.6 MB

```

```
df.describe()
```

	PM2.5	PM10	NO	NO2 \
count	228668.000000	228668.000000	228668.000000	228668.000000
mean	61.895028	125.245385	18.554708	33.380458
std	63.151386	104.326682	34.234616	25.783807
min	0.010000	0.010000	0.010000	0.010000
25%	25.500000	57.900000	3.750000	15.290000
50%	44.100000	95.650000	7.880000	26.790000
75%	73.030000	153.392500	16.640000	43.980000
max	999.990000	1000.000000	498.970000	380.020000

	NOx	NH3	CO	S02 \
count	228668.000000	228668.000000	228668.000000	228668.000000
mean	36.767091	23.438328	1.040646	11.269589
std	40.354862	19.041401	1.473336	10.519521
min	0.010000	0.010000	0.010000	0.010000
25%	14.700000	11.120000	0.490000	5.500000
50%	24.490000	18.040000	0.740000	8.600000
75%	41.880000	30.600000	1.090000	13.310000
max	493.400000	485.820000	47.420000	199.930000

	O3	Benzene	Toluene	AQI \
count	228668.000000	228668.000000	228668.000000	228668.000000
mean	38.757483	4.144182	10.089447	143.599292
std	29.490377	21.152200	23.991399	99.784499
min	0.010000	0.010000	0.010000	8.000000

25%	17.600000	0.630000	1.820000	76.000000
50%	30.850000	1.500000	4.330000	110.000000
75%	52.770000	3.620000	10.910000	173.000000
max	497.620000	498.070000	498.070000	762.000000

	hour	day	month	weekday
weekofyear \				
count	228668.000000	228668.000000	228668.000000	228668.000000
228668.0				
mean	11.526357	15.794134	6.230041	3.011777
25.334953				
std	6.929618	8.814977	3.556015	1.995459
15.476489				
min	0.000000	1.000000	1.000000	0.000000
1.0				
25%	6.000000	8.000000	3.000000	1.000000
12.0				
50%	12.000000	16.000000	6.000000	3.000000
24.0				
75%	18.000000	23.000000	10.000000	5.000000
40.0				
max	23.000000	31.000000	12.000000	6.000000
53.0				

	dayofyear
count	228668.000000
mean	174.303873
std	108.646932
min	1.000000
25%	78.000000
50%	164.000000
75%	274.000000
max	366.000000

df.shape

(228668, 34)

df.isnull().sum()

PM2.5	0
PM10	0
NO	0
NO2	0
NOx	0
NH3	0
CO	0
S02	0
O3	0
Benzene	0

```

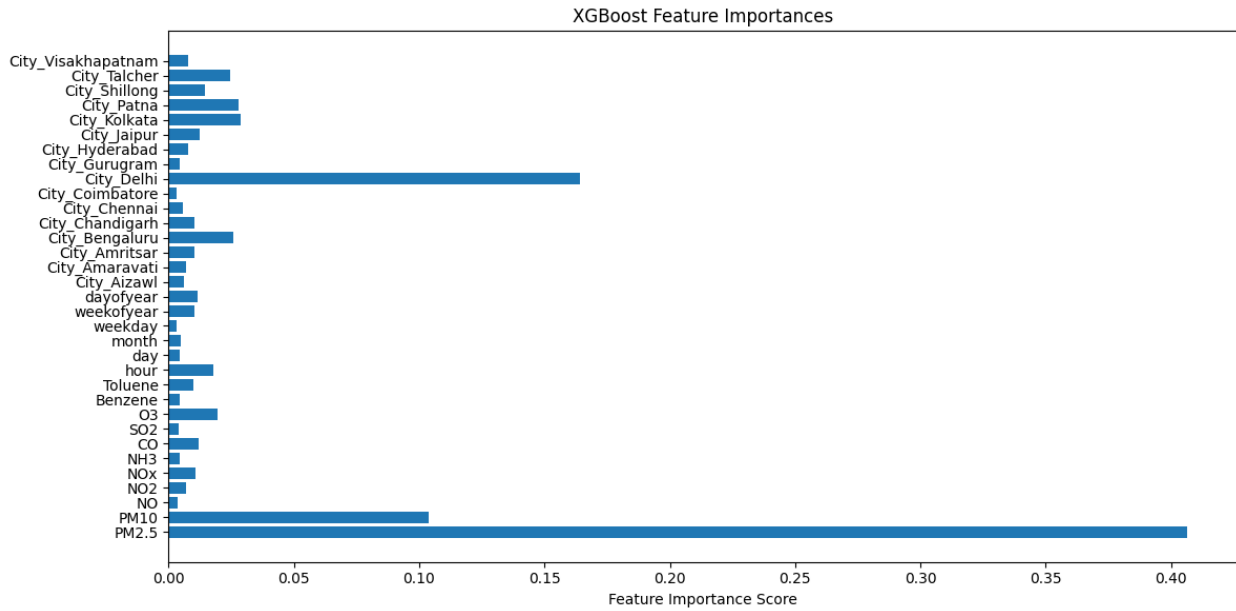
Toluene          0
AQI              0
hour            0
day            0
month          0
weekday         0
weekofyear      0
dayofyear       0
City_Aizawl     0
City_Amaravati  0
City_Amritsar   0
City_Bengaluru  0
City_Chandigarh 0
City_Chennai    0
City_Coimbatore 0
City_Delhi      0
City_Gurugram   0
City_Hyderabad  0
City_Jaipur     0
City_Kolkata    0
City_Patna      0
City_Shillong   0
City_Talcher    0
City_Visakhapatnam 0
dtype: int64

pollutant_cols = ['PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO',
                  'SO2', 'O3', 'Benzene', 'Toluene']
time_cols = ['hour', 'day', 'month', 'weekday', 'weekofyear',
             'dayofyear']
city_cols = [f'City_{city}' for city in cities] # you already defined
            'cities' earlier

all_features = pollutant_cols + time_cols + city_cols

# Plot for feature importance
plt.figure(figsize=(12, 6))
plt.barh(all_features, xgb.feature_importances_)
plt.xlabel('Feature Importance Score')
plt.title('XGBoost Feature Importances')
plt.tight_layout()
plt.show()

```



```
importance_df = pd.DataFrame({
    'Feature': all_features,
    'Importance': xgb.feature_importances_
}).sort_values(by='Importance', ascending=False)
```

```
print(importance_df.head(33))
```

	Feature	Importance
0	PM2.5	0.406175
24	City_Delhi	0.164324
1	PM10	0.103886
28	City_Kolkata	0.028956
29	City_Patna	0.028093
20	City_Bengaluru	0.025923
31	City_Talcher	0.024711
8	O3	0.019481
11	hour	0.017777
30	City_Shillong	0.014513
27	City_Jaipur	0.012269
6	CO	0.011945
16	dayofyear	0.011763
4	NOx	0.010793
19	City_Amritsar	0.010442
21	City_Chandigarh	0.010375
15	weekofyear	0.010351
10	Toluene	0.009974
26	City_Hyderabad	0.007882
32	City_Visakhapatnam	0.007791
3	NO2	0.007183
18	City_Amaravati	0.006898
17	City_Aizawl	0.006268

22	City_Chennai	0.005598
13	month	0.004908
12	day	0.004589
25	City_Gurugram	0.004519
9	Benzene	0.004511
5	NH3	0.004422
7	SO2	0.004113
2	NO	0.003491
14	weekday	0.003046
23	City_Coimbatore	0.003033