**Source Code : Air Quality Levels Using Advanced Machine Learning**

**Algorithms**

# Import necessary libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import GradientBoostingRegressor

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

# 1. Load Dataset

df = pd.read\_csv("updated\_pollution\_dataset (1).csv")

print("Initial shape:", df.shape)

print(df.head())

# 2. Data Cleaning

df.dropna(inplace=True) # Drop missing values for simplicity

df = df.drop\_duplicates()

print("Cleaned shape:", df.shape)

# 3. EDA

print(df.describe())

print(df.info())

# Correlation heatmap

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

sns.heatmap(df.corr(), annot=True, cmap="coolwarm")

plt.title("Feature Correlation")

plt.show()

# Histogram for numeric columns

df.hist(figsize=(12, 10), bins=20)

plt.tight\_layout()

plt.show()

# 4. Feature Engineering

# Example: Encoding categorical features if any

df = pd.get\_dummies(df, drop\_first=True)

# Separate features and target (assuming the target is 'pollution')

print(df.columns)

target\_col = 'PM2.5'

if target\_col not in df.columns:

  raise ValueError(f"Target column '{target\_col}' not found.Update the script accordingly.")

X = df.drop(target\_col,axis=1)

y = df[target\_col]

# Scale features

print(X.dtypes)

# Select only numeric columns from X before scaling

print(X.dtypes)

# Select only numeric columns from X before scaling

X = X.select\_dtypes(include=np.number)

print("Shape of X after removing non-numeric columns:", X.shape)

print(X.dtypes) # Print dtypes again to confirm

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# 5. Train/Test Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

# 6. Gradient Boosting Model

model = GradientBoostingRegressor(n\_estimators=100, learning\_rate=0.1, max\_depth=3, random\_state=42)

model.fit(X\_train, y\_train)

# 7. Model Evaluation

y\_pred = model.predict(X\_test)

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

r2 = r2\_score(y\_test, y\_pred)

print(f"\nModel Evaluation:")

print(f"Mean Squared Error: {mse:.2f}")

print(f"R^2 Score: {r2:.2f}")