**Air Quality Prediction System**

**Introduction**

Air pollution is a major environmental challenge that affects public health and climate conditions worldwide. Predicting air quality helps authorities take preventive measures to minimize exposure to hazardous pollutants. This report discusses the implementation of an **Air Quality Prediction System** using **CSV files** for data storage and analysis.

**Objectives**

The primary objectives of the Air Quality Prediction System are:

* To collect and process air pollution data stored in CSV files.
* To develop a machine learning model for air quality prediction.
* To analyze air quality trends based on historical data.
* To provide insights for policymaking and public awareness.

**Data Collection and CSV File Usage**

The system uses air quality data collected from monitoring stations or sensors. These datasets are typically stored in CSV files, which include key parameters such as:

* **Date & Time**: Timestamp of the measurement.
* **Location**: Geographic coordinates or station ID.
* **Pollutant Levels**: Concentrations of pollutants like **PM2.5, PM10, NO2, SO2, CO, and O3**.
* **Meteorological Data**: Factors like temperature, humidity, wind speed, and pressure.

Using **Python**, the CSV files can be processed using libraries such as **pandas** for data cleaning, transformation, and visualization.

**Methodology**

**1. Data Preprocessing**

* Load CSV files using pandas.read\_csv().
* Handle missing values and remove inconsistencies.
* Convert timestamps into useful formats for trend analysis.

**2. Feature Selection**

Selecting key features that influence air quality, such as pollutant concentrations and meteorological parameters.

**3. Model Development**

Applying **machine learning algorithms** like:

* **Linear Regression**: For trend analysis.
* **Decision Trees & Random Forest**: For pollutant level classification.
* **Neural Networks**: For deep-learning-based prediction.

**4. Training and Evaluation**

* Train the model using past air quality data from CSV files.
* Evaluate performance using metrics like **Mean Absolute Error (MAE)** and **Root Mean Square Error (RMSE)**.

**5. Prediction and Visualization**

* Forecast air quality for future timestamps.
* Display results using graphs, heatmaps, and dashboards.

**Implementation Tools**

* **Python**: Primary programming language.
* **Pandas & NumPy**: For data handling.
* **Matplotlib & Seaborn**: For visualization.
* **Scikit-learn & TensorFlow**: For machine learning.
* **Jupyter Notebook**: For interactive coding and analysis.

**CODE :**

**import pandas as pd**

**import numpy as np**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**import plotly.express as px**

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

**from sklearn.ensemble import RandomForestRegressor**

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

**from sklearn.preprocessing import StandardScaler**

**# Step 1: Load Data**

**df = pd.read\_csv("/content/city\_day.csv.zip")**

**# Step 2: Basic Cleaning**

**df = df.dropna(subset=['AQI']) # Remove rows without AQI**

**df.fillna(0, inplace=True) # Fill missing pollution values with 0**

**# Step 3: Feature Selection**

**features = ['PM2.5', 'PM10', 'NO2', 'CO', 'SO2', 'O3', 'NH3']**

**X = df[features]**

**y = df['AQI']**

**# Step 4: Train/Test Split**

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

**# Step 5: Scaling Features**

**scaler = StandardScaler()**

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

**X\_test\_scaled = scaler.transform(X\_test)**

**# Step 6: Train the Model**

**model = RandomForestRegressor(n\_estimators=100, random\_state=42)**

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

**# Step 7: Predict & Evaluate**

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

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

**rmse = np.sqrt(mse)**

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

**print(f"\nModel Evaluation:")**

**print(f"RMSE: {rmse:.2f}")**

**print(f"R² Score: {r2:.2f}")**

**# Step 8: City-Based AQI Prediction and Visualizations**

**print("\n--- Predict AQI for a City ---")**

**city\_input = input("Enter the name of the city: ").strip().title()**

**if city\_input not in df['City'].unique():**

**print("City not found in the dataset.")**

**print("Available cities in the dataset:")**

**print(df['City'].unique())**

**else:**

**city\_df = df[df['City'] == city\_input].copy()**

**city\_df['Date'] = pd.to\_datetime(city\_df['Date'])**

**latest\_record = city\_df.sort\_values(by='Date', ascending=False).iloc[0]**

**city\_features = latest\_record[features].values.reshape(1, -1)**

**city\_features\_scaled = scaler.transform(city\_features)**

**predicted\_aqi = model.predict(city\_features\_scaled)[0]**

**print(f"\nLatest available pollution data for {city\_input} (on {latest\_record['Date'].date()}):")**

**for feat in features:**

**print(f"{feat}: {latest\_record[feat]}")**

**print(f"\nPredicted AQI for {city\_input}: {predicted\_aqi:.2f}")**

**# Visualization 1: City Comparison (Bar Plot)**

**plt.figure(figsize=(12, 6))**

**avg\_aqi\_cities = df.groupby('City')['AQI'].mean().sort\_values(ascending=False)**

**top\_cities = avg\_aqi\_cities.head(10)**

**sns.barplot(x=top\_cities.index, y=top\_cities.values, palette='viridis')**

**plt.axhline(y=avg\_aqi\_cities[city\_input], color='r', linestyle='--', label=f'{city\_input} AQI')**

**plt.title(f'Top 10 Cities by Average AQI (Comparison with {city\_input})')**

**plt.xticks(rotation=45)**

**plt.ylabel('Average AQI')**

**plt.legend()**

**plt.tight\_layout()**

**plt.show()**

**# Visualization 2: Air Components Pie Chart**

**pollution\_components = latest\_record[features]**

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

**plt.pie(pollution\_components, labels=features, autopct='%1.1f%%',**

**colors=sns.color\_palette('pastel'), startangle=90)**

**plt.title(f'Air Pollution Composition in {city\_input}')**

**plt.show()**

**# Visualization 3: Time Series of AQI for the selected city**

**plt.figure(figsize=(12, 6))**

**city\_df.set\_index('Date')['AQI'].plot()**

**plt.title(f'AQI Trend Over Time in {city\_input}')**

**plt.ylabel('AQI')**

**plt.xlabel('Date')**

**plt.grid(True)**

**plt.show()**

**# Visualization 4: Feature Importance from Random Forest**

**feature\_importance = pd.Series(model.feature\_importances\_, index=features).sort\_values(ascending=False)**

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

**sns.barplot(x=feature\_importance, y=feature\_importance.index, palette='coolwarm')**

**plt.title('Feature Importance in AQI Prediction')**

**plt.xlabel('Importance Score')**

**plt.ylabel('Pollution Components')**

**plt.show()**

**# Visualization 5: Correlation Heatmap**

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

**corr\_matrix = city\_df[features + ['AQI']].corr()**

**sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', center=0)**

**plt.title(f'Pollution Component Correlation in {city\_input}')**

**plt.show()**

**# Step 9: Visualization of AQI on India Map using Mapbox**

**city\_coords = {**

**'Delhi': [28.6139, 77.2090],**

**'Mumbai': [19.0760, 72.8777],**

**'Chennai': [13.0827, 80.2707],**

**'Kolkata': [22.5726, 88.3639],**

**'Bangalore': [12.9716, 77.5946],**

**'Hyderabad': [17.3850, 78.4867],**

**'Ahmedabad': [23.0225, 72.5714],**

**'Pune': [18.5204, 73.8567],**

**'Lucknow': [26.8467, 80.9462],**

**'Kanpur': [26.4499, 80.3319],**

**'Jaipur': [26.9124, 75.7873],**

**'Patna': [25.5941, 85.1376],**

**'Bhopal': [23.2599, 77.4126],**

**'Indore': [22.7196, 75.8577],**

**'Nagpur': [21.1458, 79.0882]**

**}**

**avg\_aqi = df.groupby("City")["AQI"].mean().reset\_index()**

**avg\_aqi['lat'] = avg\_aqi['City'].map(lambda x: city\_coords.get(x, [None, None])[0])**

**avg\_aqi['lon'] = avg\_aqi['City'].map(lambda x: city\_coords.get(x, [None, None])[1])**

**avg\_aqi = avg\_aqi.dropna(subset=['lat', 'lon'])**

**fig = px.scatter\_mapbox(**

**avg\_aqi,**

**lat="lat",**

**lon="lon",**

**color="AQI",**

**size="AQI",**

**hover\_name="City",**

**color\_continuous\_scale="Plasma",**

**size\_max=20,**

**zoom=4,**

**center={"lat": 22.9734, "lon": 78.6569},**

**title="Average AQI Across Indian Cities"**

**)**

**fig.update\_layout(mapbox\_style="open-street-map")**

**fig.show()**

**# Additional Visualization: AQI Distribution by City**

**plt.figure(figsize=(14, 8))**

**top\_cities = df['City'].value\_counts().index[:10] # Top 10 cities with most data**

**sns.boxplot(data=df[df['City'].isin(top\_cities)], x='City', y='AQI', palette='Set3')**

**plt.title('AQI Distribution by City')**

**plt.xticks(rotation=45)**

**plt.tight\_layout()**

**plt.show()**

**# Additional Visualization: Monthly AQI Trends**

**df['Month'] = pd.to\_datetime(df['Date']).dt.month**

**plt.figure(figsize=(12, 6))**

**sns.lineplot(data=df, x='Month', y='AQI', hue='City',**

**estimator='mean', ci=None, palette='tab20', legend=False)**

**plt.title('Monthly AQI Trends Across Cities')**

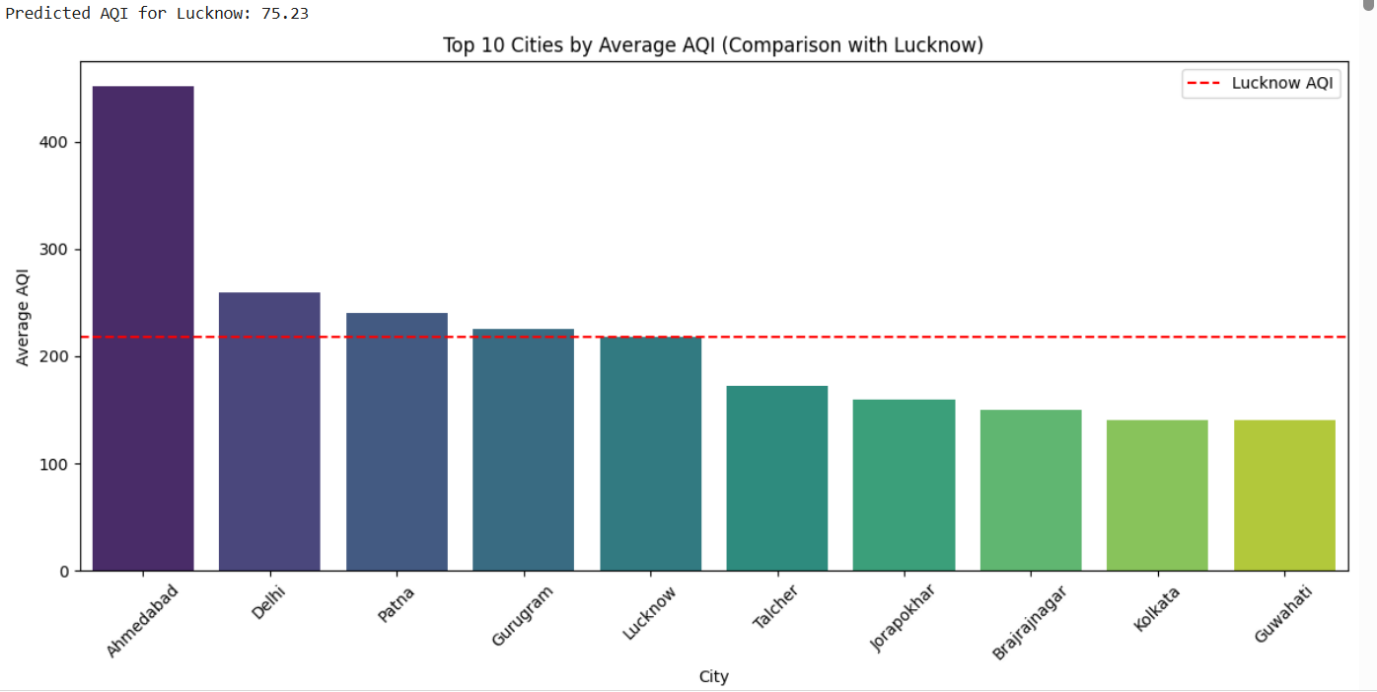
**plt.xticks(range(1, 13), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',**

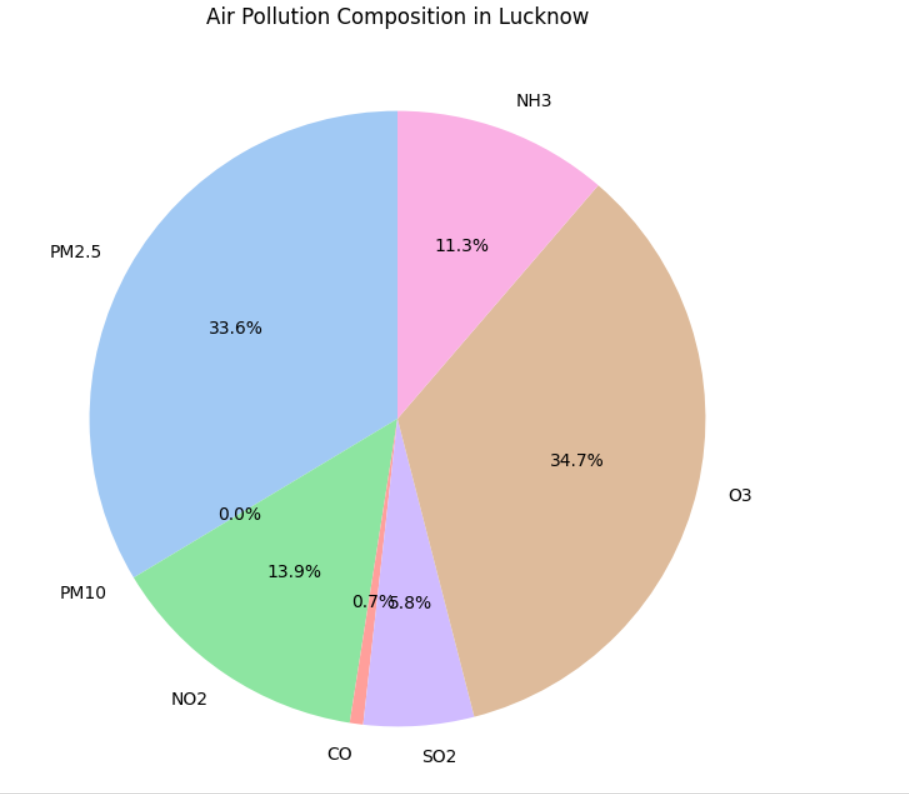
**'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])**

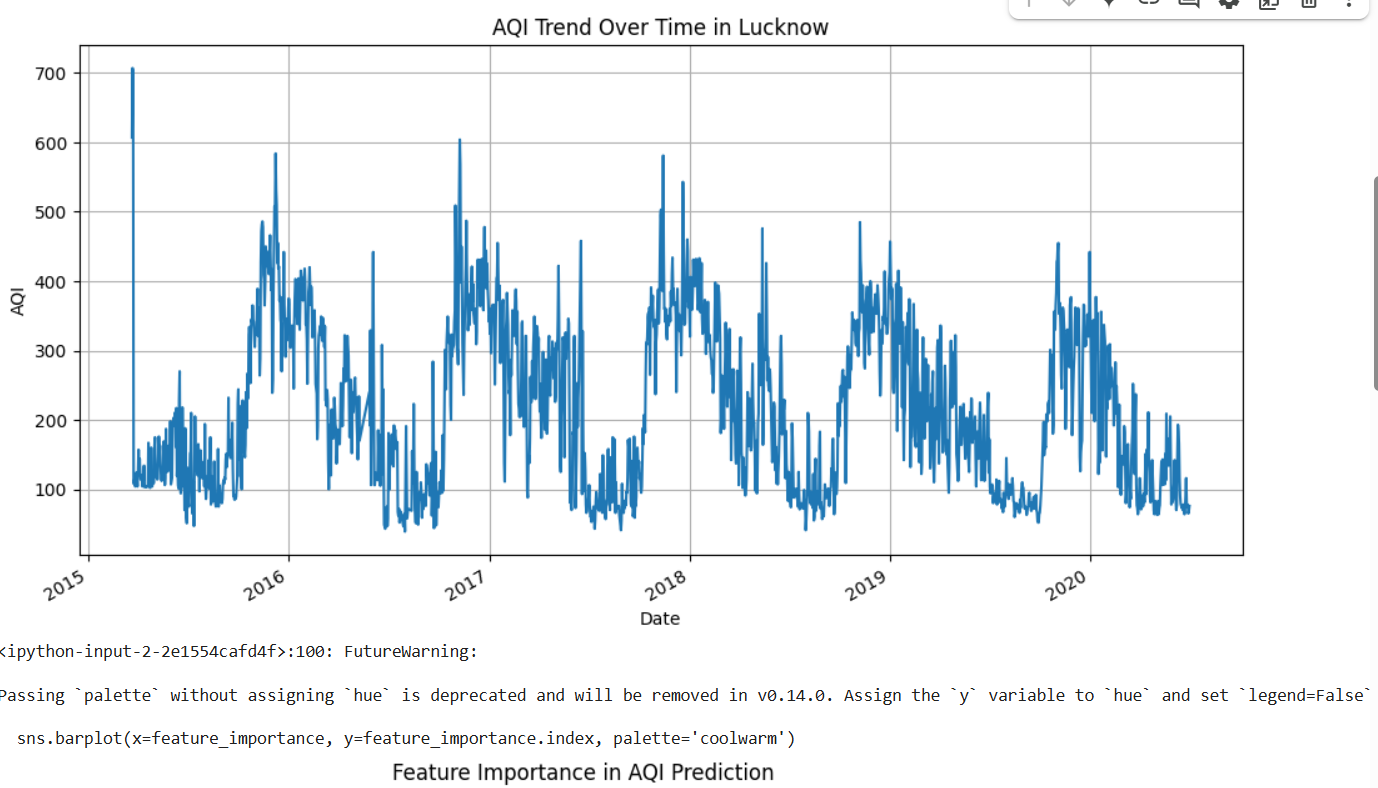
**plt.grid(True)**

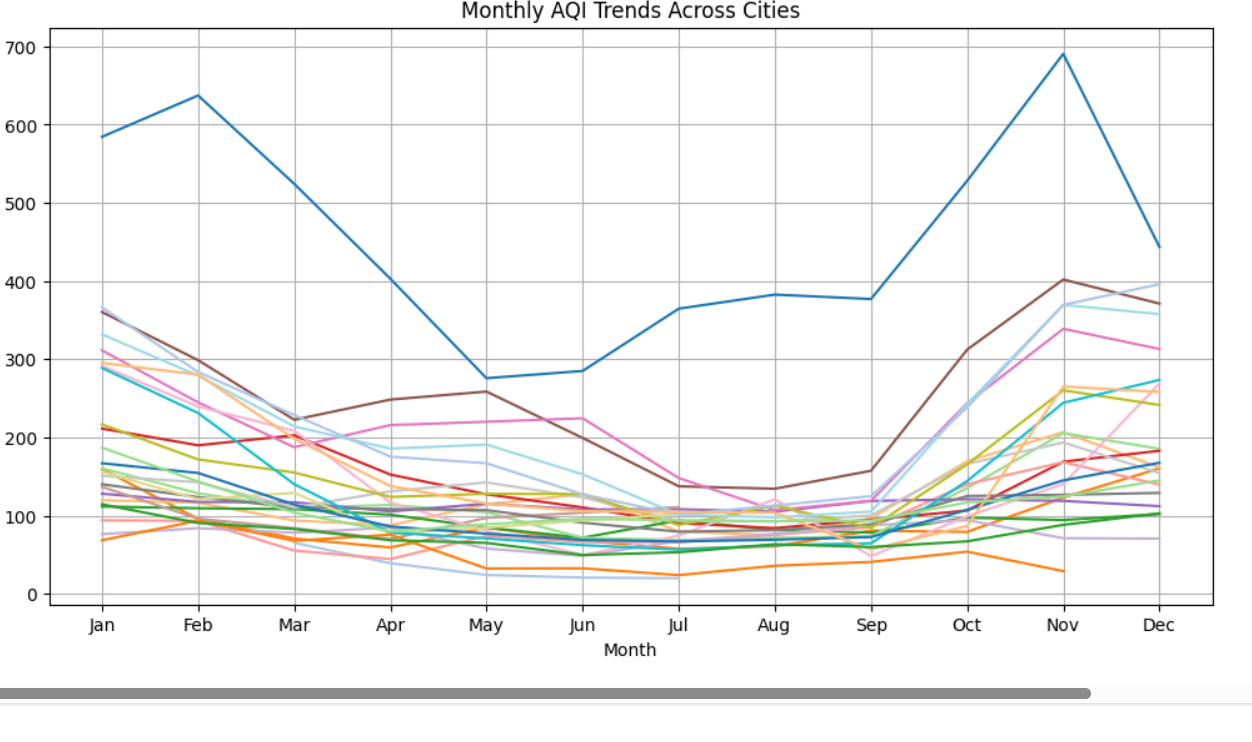
**plt.show()**

**OUTPUT :**

****

****

****

****

**Conclusion**

An Air Quality Prediction System using CSV files provides valuable insights into pollution levels and future trends. By leveraging machine learning models, policymakers and individuals can take informed actions to reduce pollution exposure. Further improvements can involve **real-time data streaming**, integration with **IoT sensors**, and deployment via **web applications**.