Air Quality Index (AQI) Prediction - Google Colab Project Notebook

```
# 📌 Phase 1: Setup and Data Loading
!pip install seaborn xgboost plotly
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, classification_report, confusion_matrix
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
import xgboost as xgb
from google.colab import files
uploaded = files.upload()
df = pd.read_csv('city_day.csv')  # Replace with your actual file name
df.head()
```

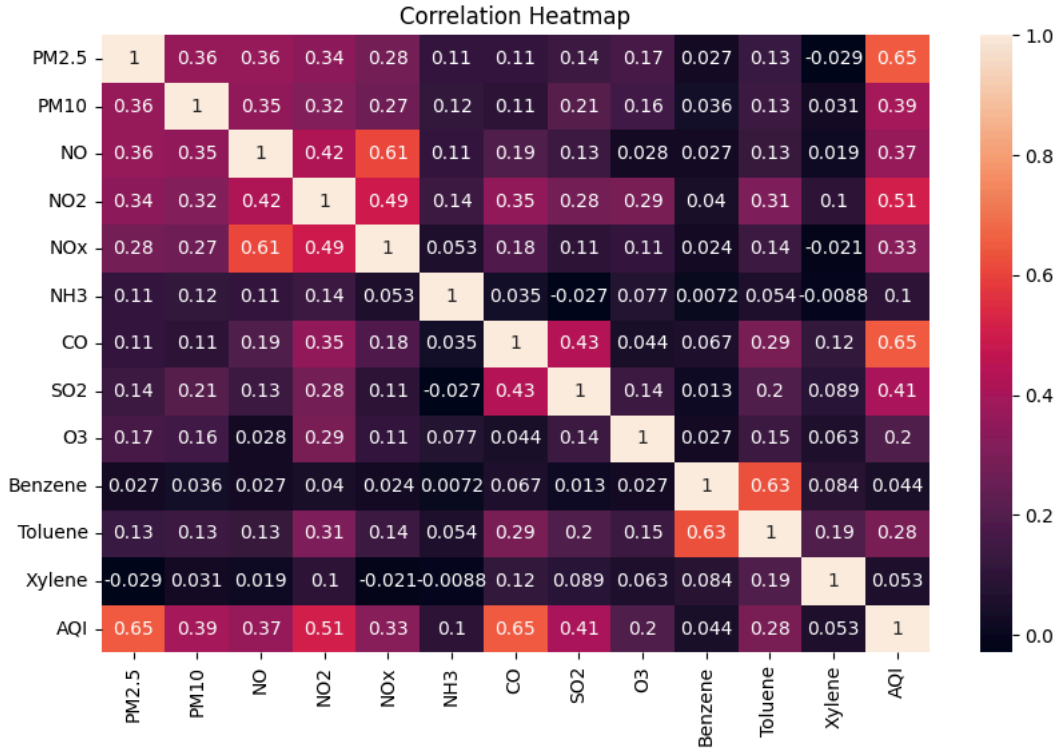
```
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: xgboost in /usr/local/lib/python3.11/dist-packages (2.1.4)
Requirement already satisfied: plotly in /usr/local/lib/python3.11/dist-packages (5.24.1)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.11/dist-packages (from seaborn) (2.0.2)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.11/dist-packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.11/dist-packages (from seaborn) (3.10.0)
Requirement already satisfied: nvidia-nccl-cu12 in /usr/local/lib/python3.11/dist-packages (from xgboost) (2.21.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from xgboost) (1.15.2)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.11/dist-packages (from plotly) (9.1.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from plotly) (24.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (11.2.1
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seabor
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving city_day.csv.zip to city_day.csv.zip
```

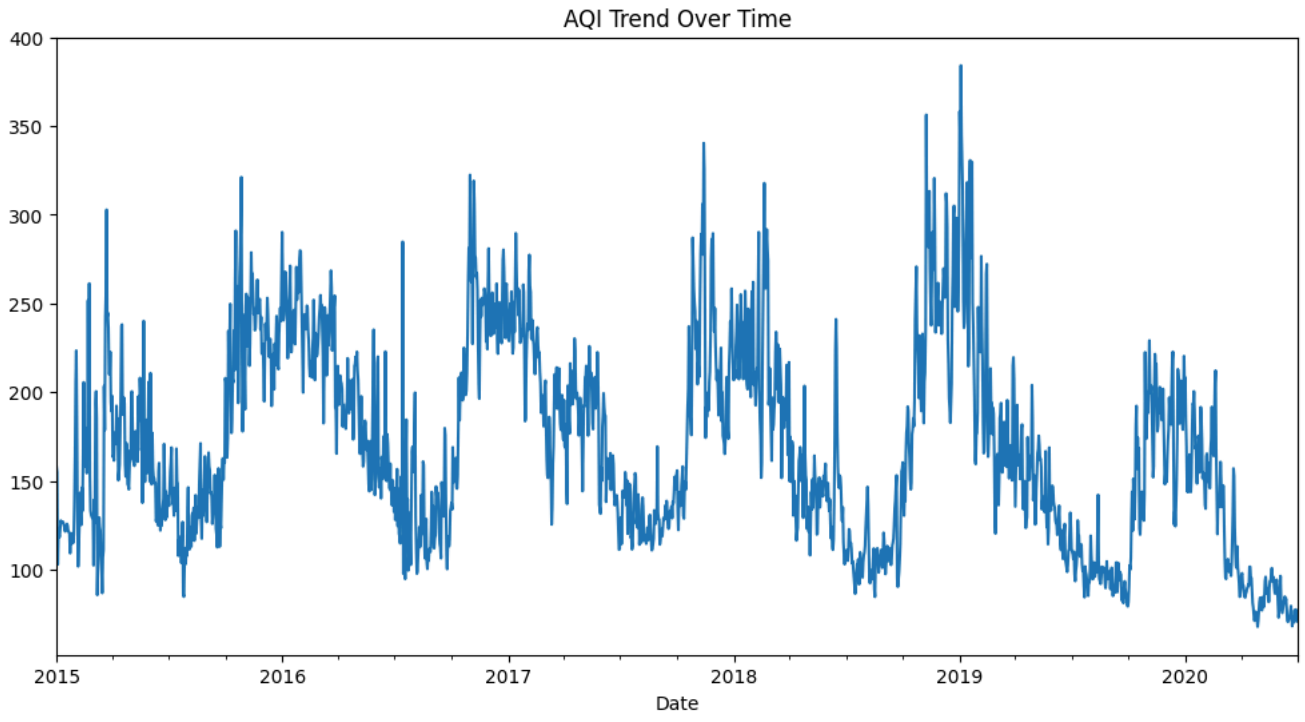
	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
0	Ahmedabad	2015-01-01	NaN	NaN	0.92	18.22	17.15	NaN	0.92	27.64	133.36	0.00	0.02	0.00	NaN	NaN
1	Ahmedabad	2015-01-02	NaN	NaN	0.97	15.69	16.46	NaN	0.97	24.55	34.06	3.68	5.50	3.77	NaN	NaN
2	Ahmedabad	2015-01-03	NaN	NaN	17.40	19.30	29.70	NaN	17.40	29.07	30.70	6.80	16.40	2.25	NaN	NaN
3	Ahmedabad	2015-01-04	NaN	NaN	1.70	18.48	17.97	NaN	1.70	18.59	36.08	4.43	10.14	1.00	NaN	NaN

```
df.ffill(inplace=True)
df.bfill(inplace=True)

print(df.columns)

location_column_name = 'City'
df.drop_duplicates(subset=['Date', location_column_name], inplace=True)
df['Date'] = pd.to_datetime(df['Date'])

label_encoder = LabelEncoder()
try:
    df['AQI_Category_Encoded'] = label_encoder.fit_transform(df['AQI_Bucket'])  # Use correct label column
except KeyError:
    print("Column 'AQI_Bucket' not found in the DataFrame. Please check the column name.")

df_with_dummies = pd.get_dummies(df, columns=[location_column_name])

# Drop all non-numeric or unwanted columns before scaling
columns_to_drop = ['AQI', 'AQI_Bucket', 'AQI_Category_Encoded', 'Date']
df_for_scaling = df_with_dummies.drop(columns=columns_to_drop, axis=1, errors='ignore')

scaled_features = MinMaxScaler().fit_transform(df_for_scaling)

X = pd.DataFrame(scaled_features, columns=df_for_scaling.columns)
```

```
Index(['City', 'Date', 'PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2',
       'O3', 'Benzene', 'Toluene', 'Xylene', 'AQI', 'AQI_Bucket',
       'AQI_Category_Encoded', 'day', 'month', 'year', 'season'],
      dtype='object')
```

```
# 📌 Phase 3: Exploratory Data Analysis (EDA)
plt.figure(figsize=(10, 6))
# Select only numerical features for correlation
numerical_df = df.select_dtypes(include=np.number)  # Select numerical columns
sns.heatmap(numerical_df.corr(), annot=True)  # Calculate correlation on numerical data
plt.title('Correlation Heatmap')
plt.show()
plt.figure(figsize=(12, 6))
df.groupby('Date')['AQI'].mean().plot()  # Changed 'timestamp' to 'Date'
plt.title('AQI Trend Over Time')
plt.show()
```






```
# 📌 Phase 4: Feature Engineering
df['day'] = df['Date'].dt.day  # Changed 'timestamp' to 'Date'
df['month'] = df['Date'].dt.month  # Changed 'timestamp' to 'Date'
df['year'] = df['Date'].dt.year  # Changed 'timestamp' to 'Date'
```

```
df['season'] = df['month'] % 12 // 3 + 1


# 📌 Phase 5: Modeling - Regression

# --- Explicitly select numerical features ---
X_reg = df.select_dtypes(include=np.number)  # Select only numerical columns

# --- Drop target and unnecessary columns ---
X_reg = X_reg.drop(columns=['AQI', 'AQI_Category_Encoded', 'day', 'month', 'year', 'season'], errors='ignore') # Dropped 'AQI', 'Date', a



#X_reg = df[['PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'O3', 'Benzene', 'Toluene', 'Xylene', 'day', 'month', 'year', 'seas

y_reg = df['AQI']
X_train, X_test, y_train, y_test = train_test_split(X_reg, y_reg, test_size=0.2, random_state=42)
model_rf = RandomForestRegressor()
model_rf.fit(X_train, y_train)
y_pred_rf = model_rf.predict(X_test)
print('MAE:', mean_absolute_error(y_test, y_pred_rf))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred_rf)))
print('R2 Score:', r2_score(y_test, y_pred_rf))
```

```
MAE: 19.51535981734534
RMSE: 41.3259287073762
R2 Score: 0.9041882759414485
```

```
# 📌 Phase 7: Feature Importance
# Assuming model_rf is defined and trained in a previous cell
# If not, make sure to train the model before this cell
try:
    feat_imp = pd.Series(model_rf.feature_importances_, index=X_train.columns)
    feat_imp.nlargest(10).plot(kind='barh')
    plt.title('Top 10 Feature Importances')
    plt.show()
except NameError:
    print("Error: model_rf is not defined. Please ensure the model is trained in a previous cell.")
```