# -\*- coding: utf-8 -\*-

"""program.ipynb

Automatically generated by Colab.

Original file is located at

https://colab.research.google.com/drive/1HfUpNpDXqiUbvWRDR4L0ayt3o7D-Fs6a

"""

# Install dependencies (if needed)

!pip install pandas scikit-learn matplotlib termcolor --quiet

# Imports

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression

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

import matplotlib.pyplot as plt

from termcolor import colored # For colored output in the terminal

# Step 1: Load the dataset (city\_day.csv)

df = pd.read\_csv("city\_day.csv")

# Step 2: Drop null values

df = df.dropna(subset=['AQI', 'City', 'Date'])

# Step 3: Convert Date column to datetime

df['Date'] = pd.to\_datetime(df['Date'])

# Step 4: Take user input

city\_input = input("Enter City Name: ")

num\_days = int(input("Enter Number of Future Days to Predict: "))

# Step 5: Filter city data

city\_df = df[df['City'].str.lower() == city\_input.lower()].sort\_values('Date')

if len(city\_df) < 10:

print(f"\n❌ Invalid city name Not enough data for '{city\_input}'. Try another city.")

else:

# Step 6: Prepare AQI with lags

aqi\_df = city\_df[['Date', 'AQI']].copy()

aqi\_df['AQI\_1'] = aqi\_df['AQI'].shift(1)

aqi\_df['AQI\_2'] = aqi\_df['AQI'].shift(2)

aqi\_df['AQI\_3'] = aqi\_df['AQI'].shift(3)

aqi\_df = aqi\_df.dropna()

# Step 7: Train the model

X = aqi\_df[['AQI\_1', 'AQI\_2', 'AQI\_3']]

y = aqi\_df['AQI']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, shuffle=False)

model = LinearRegression()

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

# Step 8: Predict future AQI

last\_known\_values = list(aqi\_df['AQI'].values[-3:])

future\_predictions = []

for \_ in range(num\_days):

input\_features = np.array(last\_known\_values[-3:]).reshape(1, -1)

next\_aqi = model.predict(input\_features)[0]

future\_predictions.append(next\_aqi)

last\_known\_values.append(next\_aqi)

# Step 9: Convert AQI to category

def get\_aqi\_category(aqi):

if aqi <= 50:

return "Good"

elif aqi <= 100:

return "Satisfactory"

elif aqi <= 200:

return "Moderate"

elif aqi <= 300:

return "Poor"

elif aqi <= 400:

return "Very Poor"

else:

return "Severe"

# Step 10: Calculate percentages of Good and Poor air quality

good\_count = 0

poor\_count = 0

for i, aqi in enumerate(future\_predictions):

category = get\_aqi\_category(aqi)

if category == "Good":

good\_count += 1

elif category == "Poor":

poor\_count += 1

good\_percentage = (good\_count / num\_days) \* 100

poor\_percentage = (poor\_count / num\_days) \* 100

# Step 11: Display results with color formatting

print(f"\n📍 AQI Prediction for {city\_input.title()} - Next {num\_days} Days:\n")

for i, aqi in enumerate(future\_predictions):

category = get\_aqi\_category(aqi)

print(colored(f"Day {i+1}: AQI = {round(aqi, 2)} ==> {category}", 'green' if category == 'Good' else 'red'))

print(f"\n🟢 Percentage of 'Good' Days: {round(good\_percentage, 2)}%")

print(f"🔴 Percentage of 'Poor' Days: {round(poor\_percentage, 2)}%")

# Step 12: Show Trend - Increasing or Decreasing AQI

trend = []

for i in range(1, len(future\_predictions)):

if future\_predictions[i] > future\_predictions[i - 1]:

trend.append("Increasing")

elif future\_predictions[i] < future\_predictions[i - 1]:

trend.append("Decreasing")

else:

trend.append("Stable")

print("\n🌡️ Air Quality Trend:")

for i, t in enumerate(trend):

print(f"Day {i + 2}: {t}")

# Step 13: Enhanced Plot

plt.figure(figsize=(10, 5))

plt.plot(range(1, num\_days + 1), future\_predictions, marker='o', color='blue', linestyle='--', linewidth=2, markersize=8)

plt.title(f"Predicted AQI for Next {num\_days} Days - {city\_input.title()}", fontsize=16, fontweight='bold')

plt.xlabel("Days Ahead", fontsize=12)

plt.ylabel("Predicted AQI", fontsize=12)

plt.xticks(range(1, num\_days + 1), rotation=45, fontsize=10)

plt.yticks(fontsize=10)

plt.grid(True, which='both', linestyle=':', linewidth=0.5, color='gray')

# Adding annotations

for i, aqi in enumerate(future\_predictions):

plt.text(i + 1, aqi + 5, f"{round(aqi, 2)}", ha='center', fontsize=10, color='black')

plt.tight\_layout()

plt.show()