INTERNSHIP PROJECT SUBMISSION

NEEL DHAYGUDE

neeldhaygude7@gmail.com

1st February 2025

COMPANY: PM ACCELERATOR

TOPIC: WEATHER REPORTS DATASET ANALYSIS AND PREDICTION

PM ACCELERATOR MISSION

Our Mission

By making industry-leading tools and education available to individuals from all backgrounds, **we level the playing field for future PM leaders.** This is the PM Accelerator motto, as we grant aspiring and experienced PMs what they need most – Access. We introduce you to industry leaders, **surround you with the right PM ecosystem**, and discover the new world of AI product management skills.

INTRODUCTION AND MOTIVATION

This is a basic weather data representation, visualization project including facilities:

1. **OVERALL TEMPRATURE, LOCATION (LATITUDE, LONGITUDE), AIR QUALITY RELATION AND VISUAL REPRESENTATION.**
2. **COUNTRYWISE REPRESENTATION OF TEMPRATURE AND AIR QUALITY**
3. **GLOBAL TRENDS OF TEMPRATURE AND AIR QUALITY**
4. **LOCAL TRENDS OF TEMPRATURE AND AIR QUALITY**
5. **GLOBAL PREDICTION OF TEMPRATURE AND AIR QUALITY**
6. **LOCAL PREDICTION OF TEMPRATURE AND AIR QUALITY**

EACH FUNCTION IS BUILT AND INDEPENDENTLY TESTED ON A LARGE DATABASE.

REASONS:

1. **UPDATING THE DATASET MODIFIES DATA REALTIME**
2. **IT CAN BE MADE APP BASED AS EACH FACILITY WORK INDEPENDENTLY.**
3. **DYNAMIC DATA RETRIVAL ALSO SHOWCASES FLEXIBILITY OF MODEL**

ENVIRONMENT USED TO CODE:

JUPYTER NOTEBOOK (MULTIPLE CODING PANNELS THAT ARE RELATED AND INDEPENDENT AT SAME TIME)

MODULES AND USE

MODULES	USE
sklearn.model_selection import train_test_split	Used to fix train and test portion of data (larger the train more accurate the result) to train model
import pandas as pd	DATA HANDLING AND MANIPULATION
import numpy as np	NUMERICAL (STATISTICAL, ARTHMATICAL) OPERATIONS

import tensorflow as tf	MACHINE LEARNING APPLICATION IN MODEL LIKE DATA RECOGNITION, MODEL MAKING
import matplotlib.pyplot as plt	Plotting basic data on scatterplots and operations like slope calculations on it
Sns by seaborn	Plotting customer attractive and more informative graphs like boxplot, violin graph, etc
from sklearn.preprocessing import StandardScaler from sklearn.linear_model import LinearRegression	Standard scalar handles **outliners, missing values all at once.** By taking in consideration variance and mean and then, linear regression checks the trends of this provided data.
from datetime import datetime, timedelta	HANDLING DATE AND TIME IN DATA FOR PRESENT AND FUTURE CONVERSION IN PREDICTION

MODEL EXPLAINATION (PART BY PART) AND CODE

1. **SETTING ENVIRONMENT**

I DOWNLOADED THE INSTRUCTED DATABASE AND STORED IT AS A ZIP FILE IN MY PC.

THIS CODE IS ABOUT IMPORTING THE FILE IN THE MODEL AND REPRESENTING A SMAPP PART FOR USER TO SEE THE STRUCTURE.

```
from sklearn.model_selection import train_test_split

import os

import zipfile

import pandas as pd

import pandas as pd

import numpy as np

from sklearn.linear_model import LinearRegression
```

```
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras import datasets, layers, models
from sklearn.model_selection import train_test_split
import seaborn as sns


zip_file_path = r"C:\Users\Neel\Downloads\archive (8).zip"
extract_to = r"C:\Users\Neel\Downloads\extracted_files"
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
  zip_ref.extractall(extract_to)
for root, dirs, files in os.walk(extract_to):
  for file in files:
    if file.endswith('.csv'):
      csv_file_path = os.path.join(root, file)
      df = pd.read_csv(csv_file_path)
      print(f"Loaded data from: {csv_file_path}")
      print(df.head())
```

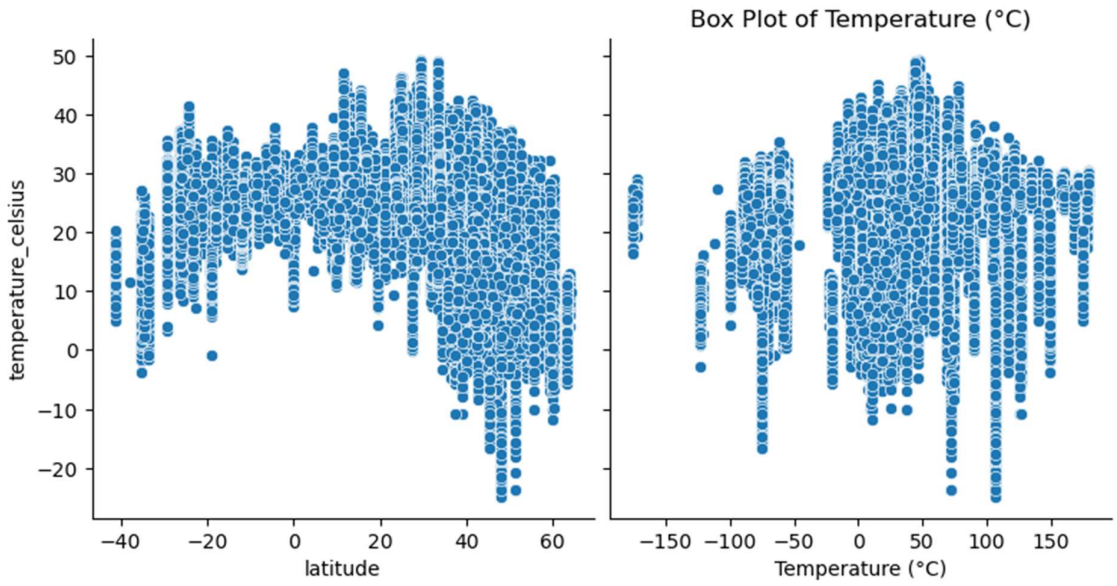
2. FIRST LET US PLOT BASIC TEMPERATURE AND AIRQUALITY GRAPHS

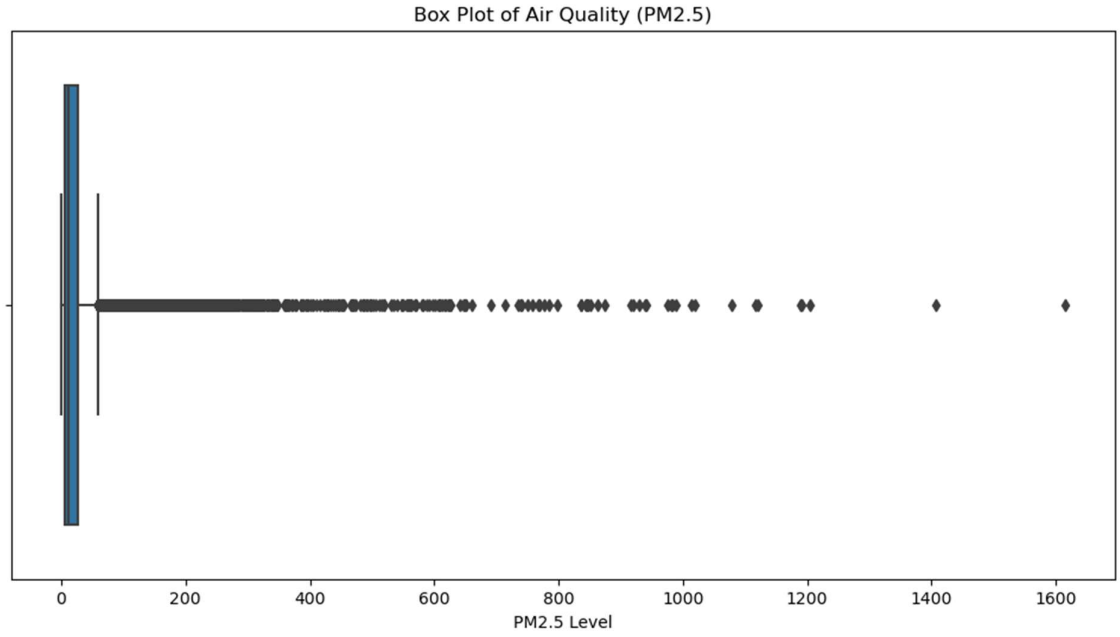
Here I have shown

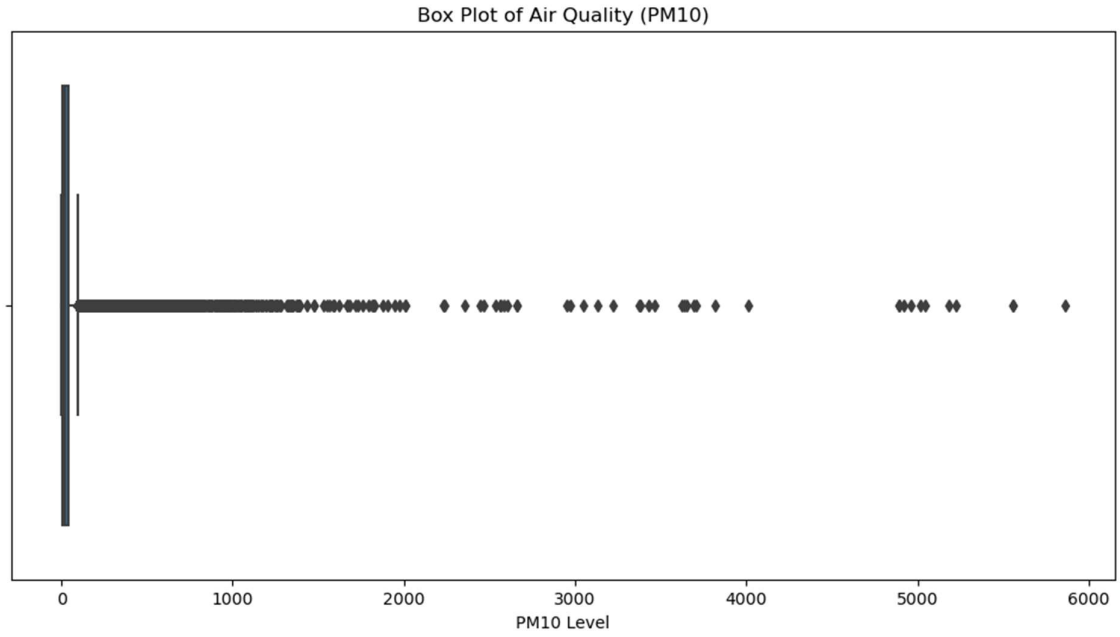
- **temperature vs latitude pairplot**
- **temperature vs longitude pairplot**
- **boxplot (shows frequency of values, more the value bigger the box) of temperature and air quality**
- **a pairplot of temperature vs air quality**

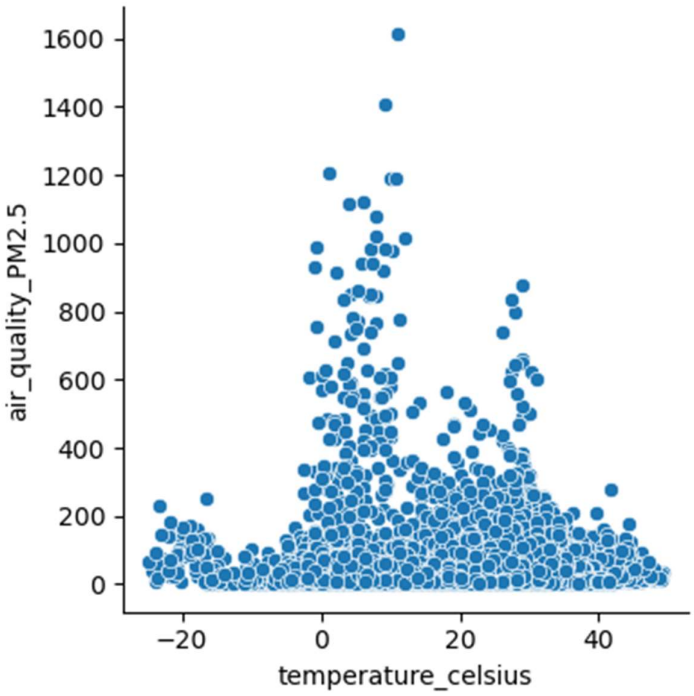
```
sns.pairplot(df, x_vars=["latitude", "longitude"], y_vars="temperature_celsius", height=4)
```

```
print("\n")

print("\nThis is a pairplot of temprature changes in various pocations\n")

print("--------------------\n")

print("\n")

plt.title("Box Plot of Temperature (°C)")

plt.xlabel("Temperature (°C)")

plt.show()

print("\nThis is a box plot of temprature ranges tells which temp values are dominant and common\n")

print("--------------------\n")

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

sns.boxplot(x=df["air_quality_PM2.5"])

plt.title("Box Plot of Air Quality (PM2.5)")

plt.xlabel("PM2.5 Level")

plt.show()

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

sns.boxplot(x=df["air_quality_PM10"])

plt.title("Box Plot of Air Quality (PM10)")

plt.xlabel("PM10 Level")

plt.show()

print("\nThis is a similar box plot of air quality ranges tells which air quality values are dominant and common\n")

print("--------------------\n")

sns.pairplot(df, x_vars=["temperature_celsius"], y_vars="air_quality_PM2.5", height=4)

print("\nFINALLY A PAIRPLOT COMPARING BOTH OF THEM\n")

print("--------------------\n")
```


Box Plot of Temperature (°C)
temperature_celsius
50
40
30
20
10
0
−10
−20
−40
−20
0
20
40
60
latitude
−150
−100
−50
0
50
100
150
Temperature (°C)


Box Plot of Air Quality (PM2.5)
0
200
400
600
800
1000
1200
1400
1600
PM2.5 Level


Box Plot of Air Quality (PM10)
0
1000
2000
3000
4000
5000
6000
PM10 Level

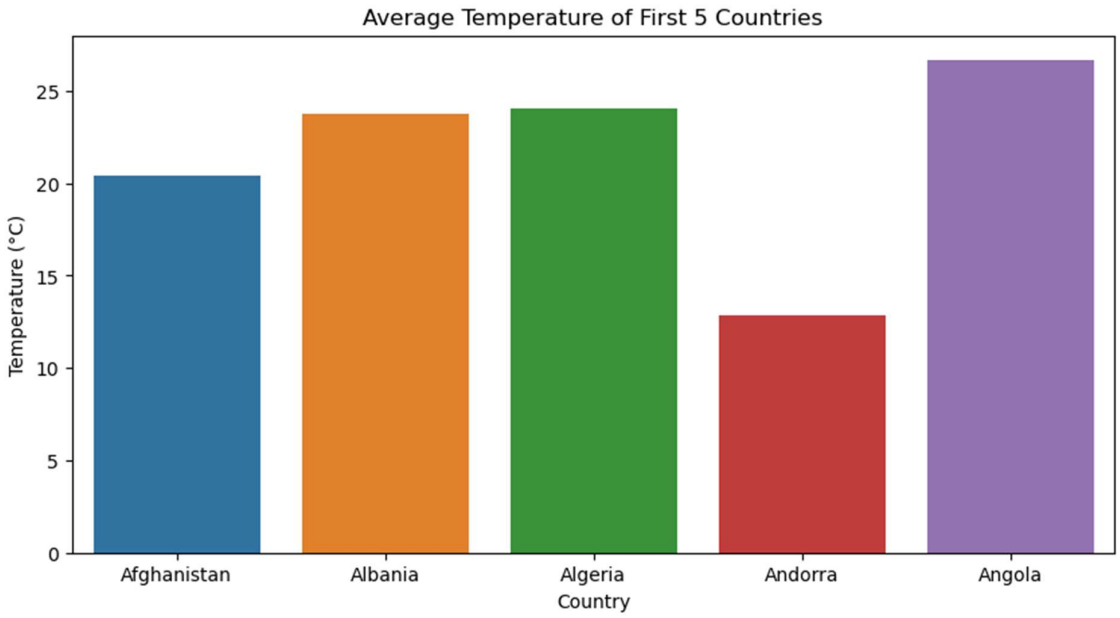

1600
1400
1200
1000
800
600
400
200
0
air_quality_PM2.5
−20
0
20
40
temperature_celsius

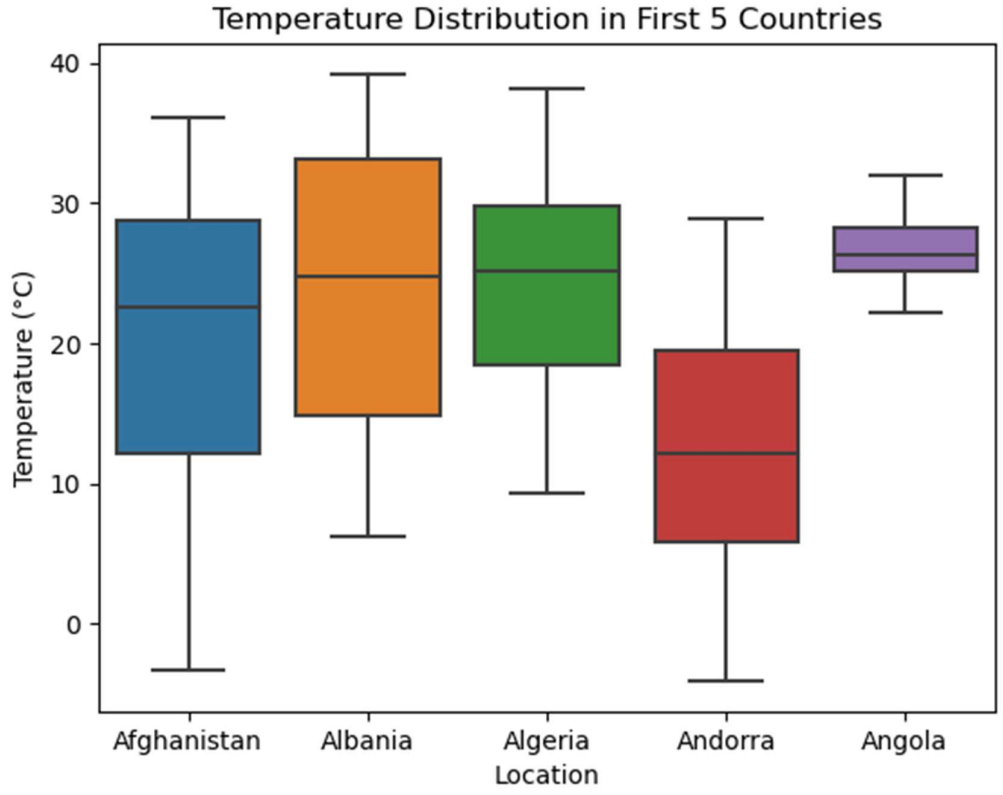
3. COUNTRY BASED GRAPHS (KEEPING FIRST 5 COUNTRIES IN MIND DUE TO LARGE DATASET)

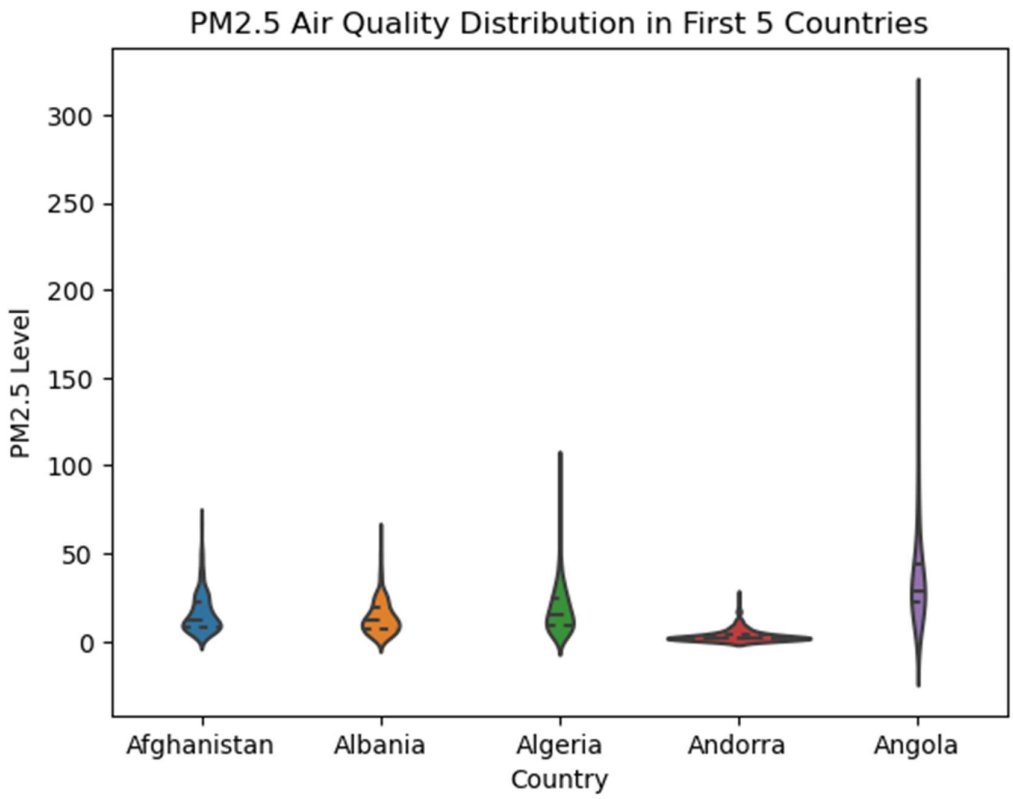
GIVEN THE LARGE DATASET TO SHOW COUNTRY VISE DATA (SELECTIVE DATA HANDLING USAGE) I USED 5 TOP COUNTRIES AND MADE

- **BARPLOT OF AVERAGE TEMPRATURE**
- **MULTI VALUE BOXPLOT OF TEMPRATURE RANGES**
- **VIOLIN PLOT OF AIRQUALITY DENSITIES IN EACH REGION**

```
plt.figure(figsize=(10, 5))
C5 = df["country"][:5]
df_top5 = df[df["country"].isin(C5)]
sns.barplot(x=df_top5.groupby("country")["temperature_celsius"].mean().index,
        y=df_top5.groupby("country")["temperature_celsius"].mean().values)
plt.title("Average Temperature of First 5 Countries")
plt.xlabel("Country")
plt.ylabel("Temperature (°C)")
plt.show()
sns.boxplot(x="country", y="temperature_celsius", data=df_top5)
plt.title("Temperature Distribution in First 5 Countries")
plt.xlabel("Location")
plt.ylabel("Temperature (°C)")
plt.show()
sns.violinplot(x="country", y="air_quality_PM2.5", data=df_top5, inner="quartile")
plt.title("PM2.5 Air Quality Distribution in First 5 Countries")
plt.xlabel("Country")
plt.ylabel("PM2.5 Level")
plt.show()
```


Average Temperature of First 5 Countries
Temperature (°C)
0
5
10
15
20
25
Afghanistan
Albania
Algeria
Andorra
Angola
Country


Temperature Distribution in First 5 Countries
Temperature (°C)
0
10
20
30
40
Afghanistan
Albania
Algeria
Andorra
Angola
Location



4. COUNTRY VISE AVERAGE VALUES AS TABLE

NOW FOR AN INTERACTIVE SIDE

HERE THE USER INPUTS A COUNTRY AND USING MAJORLY PANDA AND NUMPY FUNCTIONS WE FINF AVERAGE TEMPRATURE AND AIR QUALITY (PM2.5)

AND DISPLAY AS TABLE

```
def display_country_data_summary(file_path, country_name):
    df = pd.read_csv(file_path)
    country_data = df[df['country'] == country_name]
    if country_data.empty:
        print("Country not found in dataset.")
        return
    avg_temp = country_data['temperature_celsius'].mean()
    avg_aq = country_data['air_quality_PM2.5'].mean()
    lat = country_data['latitude'].iloc[0]
```

```
    lon = country_data['longitude'].iloc[0]
    summary_df = pd.DataFrame({
        "Country": [country_name],
        "Average Temperature (°C)": [round(avg_temp, 2)],
        "Average Air Quality (PM2.5)": [round(avg_aq, 2)],
        "Latitude": [lat],
        "Longitude": [lon]
    })
    print("\nCountry Data Summary:")
    print(summary_df.to_string(index=False))
file_path = r"C:\Users\Neel\Downloads\extracted_files\GlobalWeatherRepository.csv"
country_name = input("Enter country name: ")
display_country_data_summary(file_path, country_name)
```

```
Enter country name: India

Country Data Summary:
Country  Average Temperature (°C)  Average Air Quality (PM2.5)  Latitude  Longitude
  India                     30.87                       107.46      28.6       77.2
```

5. GLOBAL TRENDS IN TEMPRATURE AND AIR QUALITY

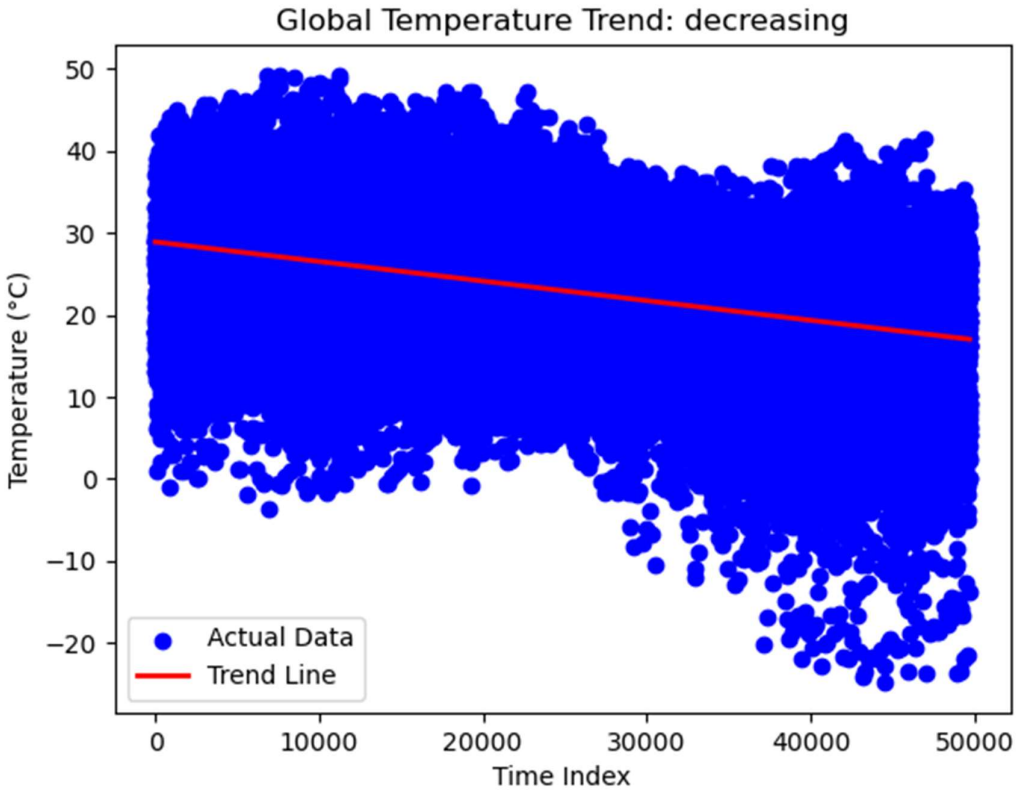
HERE COMES REGRESSION AND MODEL TRAINING:

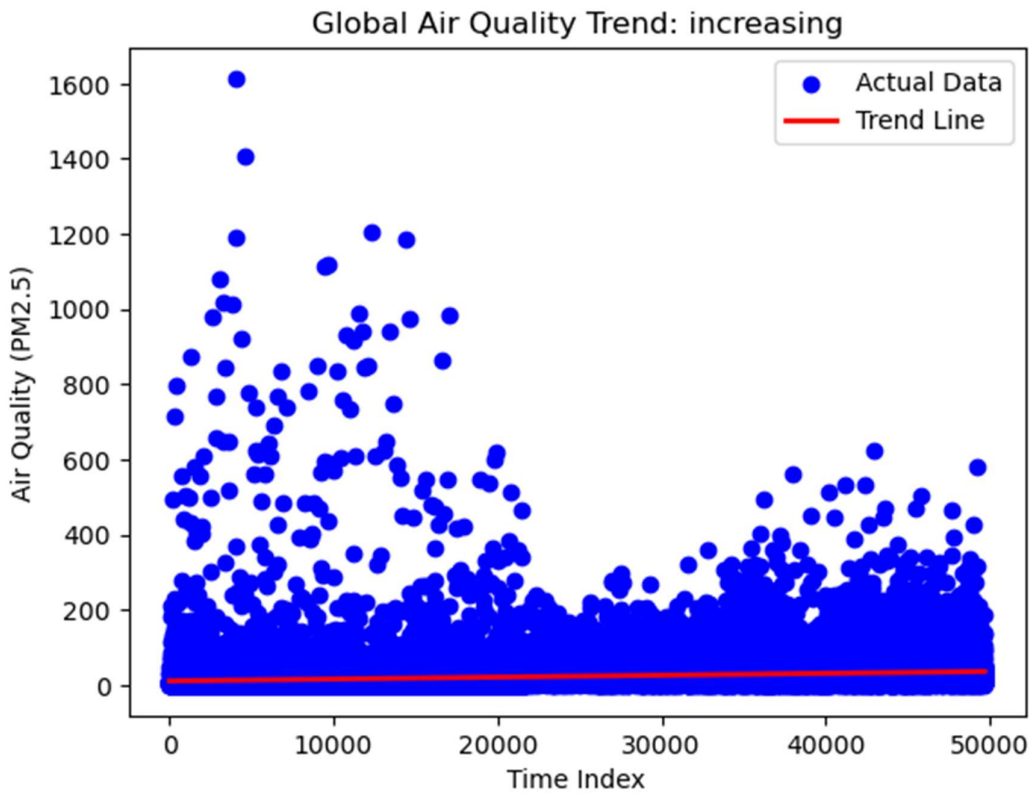
- **TO START WE JUST PLOTTED ALL THE TEMPRATURE VS TIME AND AIR QUALITY VS TIME VALUES GLOBALLY**
- **THEN WE MAKE A SCATTER PLOT LINE**
- **USING THE SLOPE WE CAN CONCLUDE IF TEMPRATURE IS INCREASING OR DECREASING**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
def predict_temperature_trend(file_path):
    df = pd.read_csv(file_path)
    df = df.sort_values(by='last_updated_epoch')
    df['time_index'] = np.arange(len(df))
    X = df[['time_index']]
    y = df['temperature_celsius']
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    model = LinearRegression()
    model.fit(X_scaled, y)
    trend_slope = model.coef_[0]
    trend = "increasing" if trend_slope > 0 else "decreasing"
    plt.scatter(X, y, color='blue', label='Actual Data')
    plt.plot(X, model.predict(X_scaled), color='red', linewidth=2, label='Trend Line')
    plt.xlabel('Time Index')
    plt.ylabel('Temperature (°C)')
    plt.title(f'Global Temperature Trend: {trend}')
    plt.legend()
    plt.show()
    print(f"Global temperature trend is {trend}.")

def predictair(file_path):
    df = pd.read_csv(file_path)
    df = df.sort_values(by='last_updated_epoch')
    df['time_index'] = np.arange(len(df))
    X = df[['time_index']]
    y = df['air_quality_PM2.5']
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    model = LinearRegression()
    model.fit(X_scaled, y)
```

```
    trend_slope = model.coef_[0]
    trend = "increasing" if trend_slope > 0 else "decreasing"
    plt.scatter(X, y, color='blue', label='Actual Data')
    plt.plot(X, model.predict(X_scaled), color='red', linewidth=2, label='Trend Line')
    plt.xlabel('Time Index')
    plt.ylabel('Air Quality (PM2.5)')
    plt.title(f'Global Air Quality Trend: {trend}')
    plt.legend()
    plt.show()
    print(f"Global air quality trend is {trend}.")

predict_temperature_trend(file_path)
predictair(file_path)
```





6. COUNTRY WISE TRENDS IN TEMPRATURE AMD AIR OVER TIME

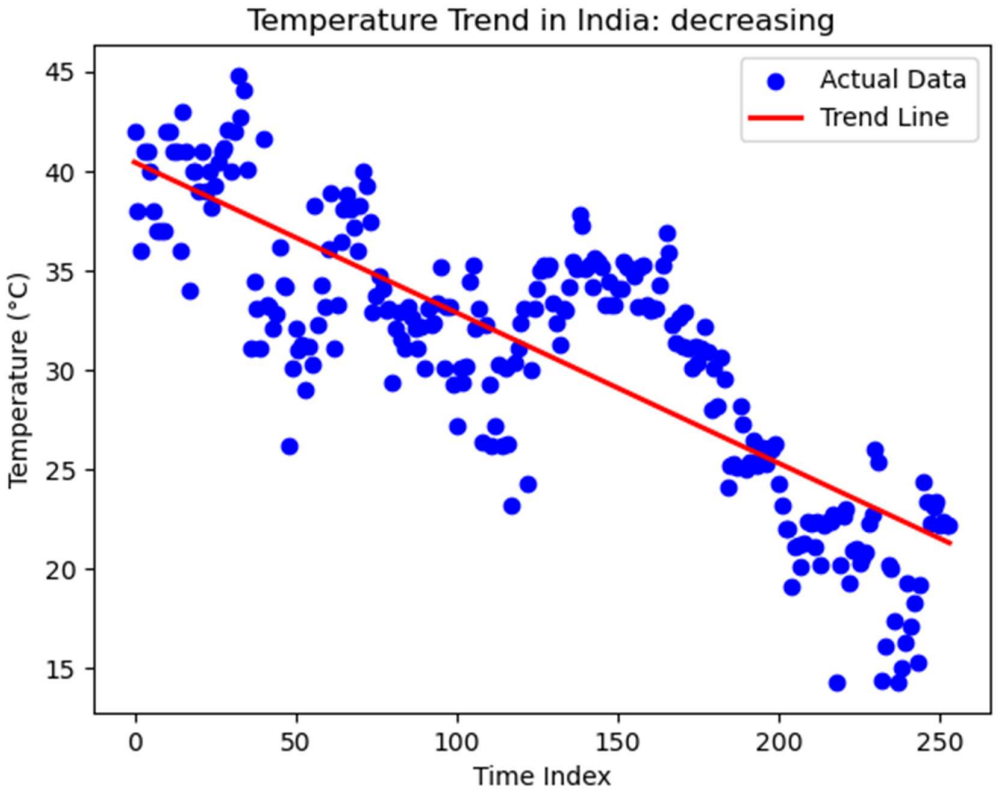
(THIS DOES THE ABOVE FUNCTION BUT ON A SELECTED COUNTRY ENTERED BY THE USER)

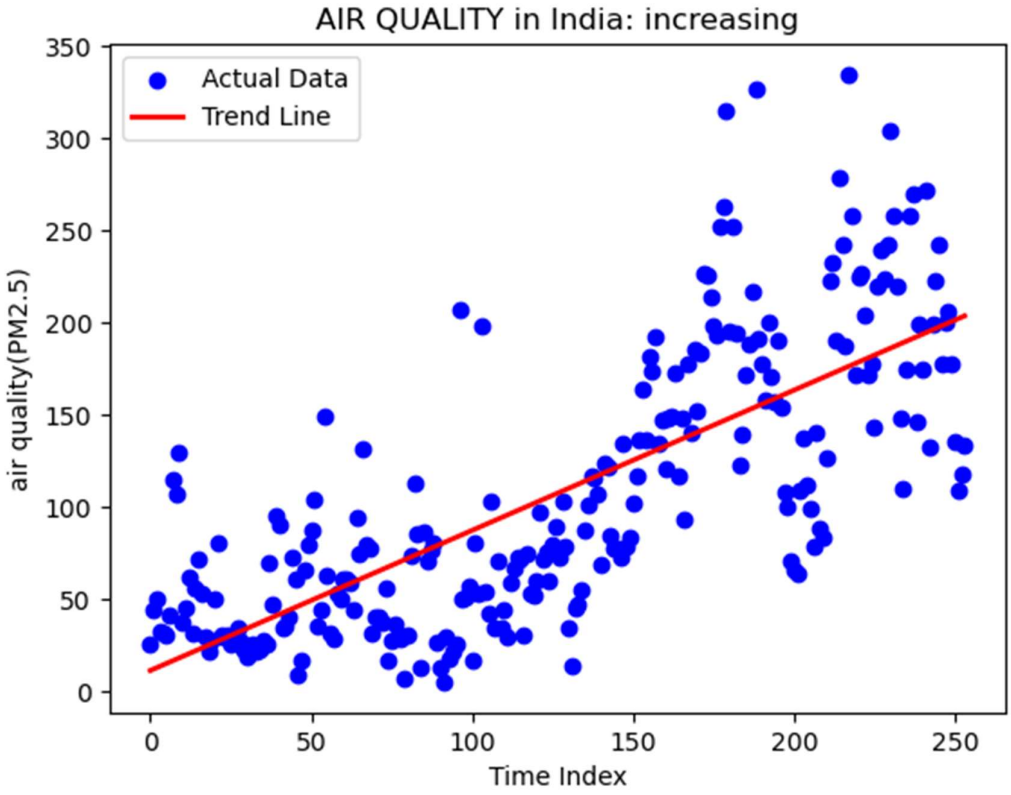
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler

def predict_25_trend(country_name, file_path):
    country_data = df[df['country'] == country_name]
    if country_data.empty:
        print("Country not found in dataset.")
```

```
        return
    country_data = country_data.sort_values(by='last_updated_epoch')
    country_data['time_index'] = np.arange(len(country_data))
    X = country_data[['time_index']]
    y = country_data['air_quality_PM2.5']
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    model = LinearRegression()
    model.fit(X_scaled, y)
    trend_slope = model.coef_[0]
    trend = "increasing" if trend_slope > 0 else "decreasing"
    plt.scatter(X, y, color='blue', label='Actual Data')
    plt.plot(X, model.predict(X_scaled), color='red', linewidth=2, label='Trend Line')
    plt.xlabel('Time Index')
    plt.ylabel('air quality(PM2.5)')
    plt.title(f'AIR QUALITY in {country_name}: {trend}')
    plt.legend()
    plt.show()
    print(f"AIR QUALITY trend in {country_name} is {trend}.")

def predict_temperature_trend(country_name, file_path):
    country_data = df[df['country'] == country_name]
    if country_data.empty:
        print("Country not found in dataset.")
        return
    country_data = country_data.sort_values(by='last_updated_epoch')
    country_data['time_index'] = np.arange(len(country_data))
    X = country_data[['time_index']]
    y = country_data['temperature_celsius']
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
```

```
    model = LinearRegression()
    model.fit(X_scaled, y)
    trend_slope = model.coef_[0]
    trend = "increasing" if trend_slope > 0 else "decreasing"
    plt.scatter(X, y, color='blue', label='Actual Data')
    plt.plot(X, model.predict(X_scaled), color='red', linewidth=2, label='Trend Line')
    plt.xlabel('Time Index')
    plt.ylabel('Temperature (°C)')
    plt.title(f'Temperature Trend in {country_name}: {trend}')
    plt.legend()
    plt.show()
    print(f"Temperature trend in {country_name} is {trend}.")

file_path = r'C:\Users\Neel\Downloads\extracted_files\GlobalWeatherRepository.csv'
country_name = input("Enter country name: ")
predict_temperature_trend(country_name, zip_file_path)
predict_25_trend(country_name, zip_file_path)
```

(assuming chosen country is india)


Temperature Trend in India: decreasing
Actual Data
Trend Line
Temperature (°C)
Time Index
45
40
35
30
25
20
15
0
50
100
150
200
250


AIR QUALITY in India: increasing
Actual Data
Trend Line
air quality(PM2.5)
Time Index
350
300
250
200
150
100
50
0
0
50
100
150
200
250

TEMPRATURE AND AIR QUALITY PREDICTION (GLOBAL)

HERE AFTER TRAINING THE MODEL BASED ON THE SLOPE OF LINE AND KNOWN DATA VALUES WE FIND THE GLOBAL TEMPRATURE POSSIBLE IN NEXT TIME SLOTS IN FUTURE

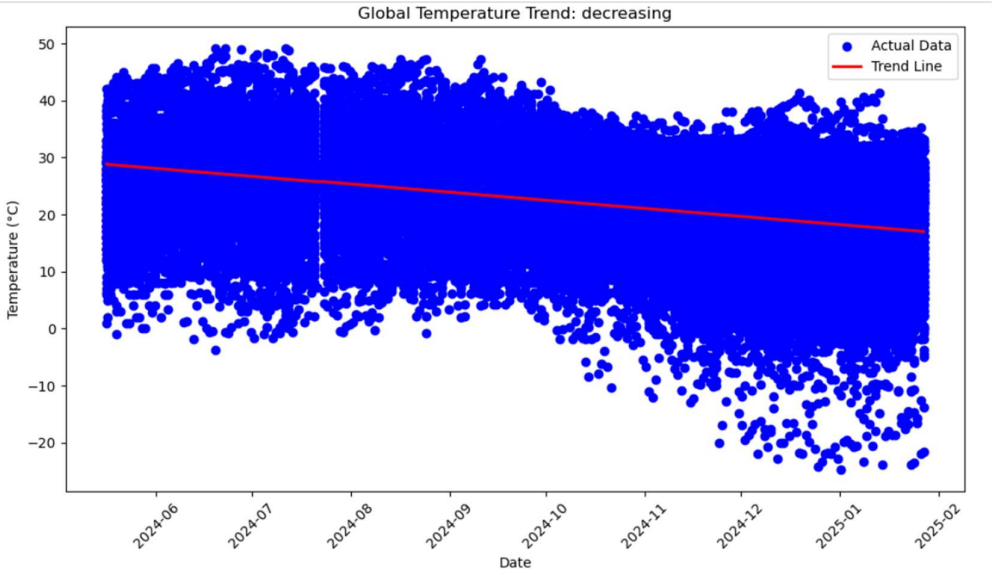
```
fut = [ld + timedelta(days=i) for i in range(1, 4)]
    fut_ind = np.array([len(df) + i for i in range(1, 4)]).reshape(-1, 1)
    fut_sca = scaler.transform(fut_ind)
    fut_pred = model.predict(fut_sca)
```

USING THIS CODE WE USE THE HIGHLIGHTED FUNCTION TO MAKE THE PREDICTIONS ON GENERATED DATED BASED ON A TRAINED MODEL

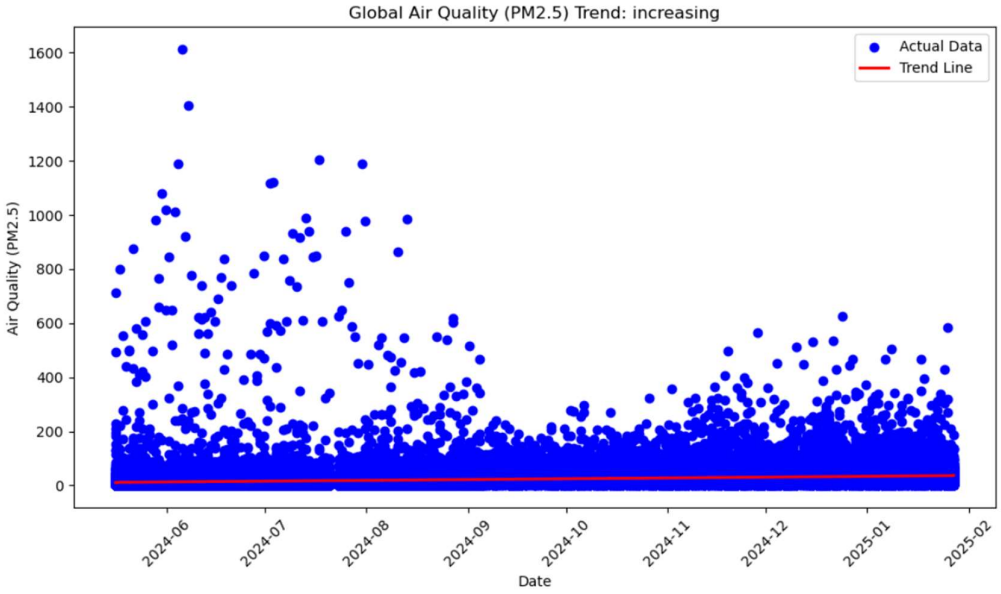
```
from datetime import datetime, timedelta


def predict_global_temperature_trend(file_path):
    df = pd.read_csv(file_path)
    df = df.sort_values(by='last_updated_epoch')
    df['datetime'] = pd.to_datetime(df['last_updated_epoch'], unit='s')
    df['time_index'] = np.arange(len(df))
    X = df[['time_index']]
    y = df['temperature_celsius']
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    model = LinearRegression()
    model.fit(X_scaled, y)
    trend_slope = model.coef_[0]
    trend = "increasing" if trend_slope > 0 else "decreasing"
    plt.figure(figsize=(10, 6))
    plt.scatter(df['datetime'], y, color='blue', label='Actual Data')
    plt.plot(df['datetime'], model.predict(X_scaled), color='red', linewidth=2, label='Trend Line')
    plt.xlabel('Date')
    plt.ylabel('Temperature (°C)')
    plt.title(f'Global Temperature Trend: {trend}')
```

```
    plt.legend()
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
    print(f"Global temperature trend is {trend}.")
    ld = df['datetime'].iloc[-1]
    fut = [ld + timedelta(days=i) for i in range(1, 4)]
    fut_ind = np.array([len(df) + i for i in range(1, 4)]).reshape(-1, 1)
    fut_sca = scaler.transform(fut_ind)
    fut_pred = model.predict(fut_sca)
    print("Predicted future temperatures:")
    for i, temp in enumerate(fut_pred):
        print(f"Date {fut[i].strftime('%Y-%m-%d %H:%M:%S')}: {temp:.2f}°C")

def predict_global_air_quality_trend(file_path):
    df = pd.read_csv(file_path)
    df = df.sort_values(by='last_updated_epoch')
    df['datetime'] = pd.to_datetime(df['last_updated_epoch'], unit='s')
    df['time_index'] = np.arange(len(df))
    X = df[['time_index']]
    y = df['air_quality_PM2.5']
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    model = LinearRegression()
    model.fit(X_scaled, y)
    trend_slope = model.coef_[0]
    trend = "increasing" if trend_slope > 0 else "decreasing"
    plt.figure(figsize=(10, 6))
    plt.scatter(df['datetime'], y, color='blue', label='Actual Data')
    plt.plot(df['datetime'], model.predict(X_scaled), color='red', linewidth=2, label='Trend Line')
    plt.xlabel('Date')
```

```
    plt.ylabel('Air Quality (PM2.5)')
    plt.title(f'Global Air Quality (PM2.5) Trend: {trend}')
    plt.legend()
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
    print(f"Global air quality trend is {trend}.")
    ld = df['datetime'].iloc[-1]
    fut = [ld + timedelta(days=i) for i in range(1, 4)]
    fut_ind = np.array([len(df) + i for i in range(1, 4)]).reshape(-1, 1)
    fut_sca = scaler.transform(fut_ind)
    fut_pred = model.predict(fut_sca)
    print("Predicted future air quality (PM2.5) values:")
    for i, aq in enumerate(fut_pred):
        print(f"Date {fut[i].strftime('%Y-%m-%d %H:%M:%S')}: {aq:.2f}")

file_path = r'C:\Users\Neel\Downloads\extracted_files\GlobalWeatherRepository.csv'
predict_global_temperature_trend(file_path)
predict_global_air_quality_trend(file_path)
```




```
Global temperature trend is decreasing.
Predicted future temperatures:
Date 2025-01-28 10:30:00: 17.01°C
Date 2025-01-29 10:30:00: 17.01°C
Date 2025-01-30 10:30:00: 17.01°C
```




```
Global air quality trend is increasing.
Predicted future air quality (PM2.5) values:
Date 2025-01-28 10:30:00: 36.27
Date 2025-01-29 10:30:00: 36.27
Date 2025-01-30 10:30:00: 36.27
```

TEMPRATURE AND AIR QUALITY PREDICTION (COUNTRY BASED)

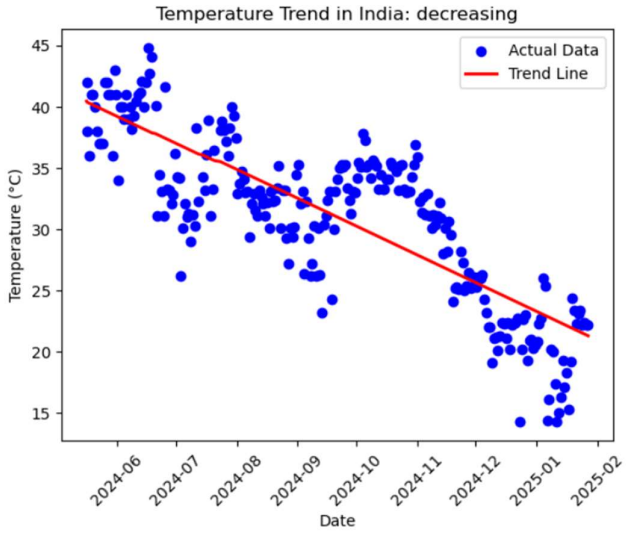
(SAME FUNCTION AS BEFORE BUT ON A USER ENTERED COUNTRY)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from datetime import datetime, timedelta

def predict_temperature_trend(file_path, country_name):
    df = pd.read_csv(file_path)
    country_data = df[df['country'] == country_name]
    if country_data.empty:
        print("Country not found in dataset.")
        return
    country_data = country_data.sort_values(by='last_updated_epoch')
    country_data['datetime'] = pd.to_datetime(country_data['last_updated_epoch'], unit='s')
    country_data['time_index'] = np.arange(len(country_data))
    X = country_data[['time_index']]
    y = country_data['temperature_celsius']
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    model = LinearRegression()
    model.fit(X_scaled, y)
    trend_slope = model.coef_[0]
    trend = "increasing" if trend_slope > 0 else "decreasing"
    plt.scatter(country_data['datetime'], y, color='blue', label='Actual Data')
    plt.plot(country_data['datetime'], model.predict(X_scaled), color='red', linewidth=2,
label='Trend Line')
    plt.xlabel('Date')
    plt.ylabel('Temperature (°C)')
    plt.title(f'Temperature Trend in {country_name}: {trend}')
```

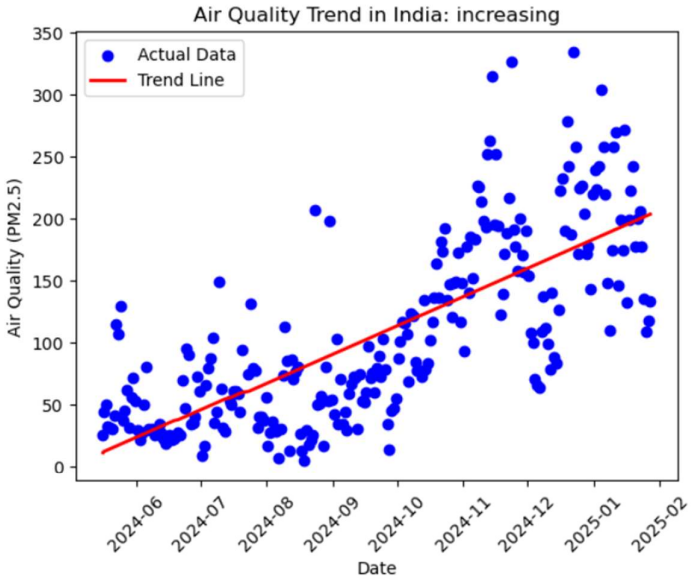
```
    plt.legend()
    plt.xticks(rotation=45)
    plt.show()
    print(f"Temperature trend in {country_name} is {trend}.")
    ld = country_data['datetime'].iloc[-1]
    fut = [ld + timedelta(days=i) for i in range(1, 4)]
    fut_ind = np.array([len(country_data) + i for i in range(1, 4)]).reshape(-1, 1)
    fut_sca = scaler.transform(fut_ind)
    fut_pred = model.predict(fut_sca)
    print("Predicted future temperatures:")
    for i, temp in enumerate(fut_pred):
        print(f"Date {fut[i].strftime('%Y-%m-%d %H:%M:%S')}: {temp:.2f}°C")

def predictair(file_path, country_name):
    df = pd.read_csv(file_path)
    country_data = df[df['country'] == country_name]
    if country_data.empty:
        print("Country not found in dataset.")
        return
    country_data = country_data.sort_values(by='last_updated_epoch')
    country_data['datetime'] = pd.to_datetime(country_data['last_updated_epoch'], unit='s')
    country_data['time_index'] = np.arange(len(country_data))
    X = country_data[['time_index']]
    y = country_data['air_quality_PM2.5']
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    model = LinearRegression()
    model.fit(X_scaled, y)
    trend_slope = model.coef_[0]
    trend = "increasing" if trend_slope > 0 else "decreasing"
    plt.scatter(country_data['datetime'], y, color='blue', label='Actual Data')
```

```
    plt.plot(country_data['datetime'], model.predict(X_scaled), color='red', linewidth=2,
label='Trend Line')
    plt.xlabel('Date')
    plt.ylabel('Air Quality (PM2.5)')
    plt.title(f'Air Quality Trend in {country_name}: {trend}')
    plt.legend()
    plt.xticks(rotation=45)
    plt.show()
    print(f"Air quality trend in {country_name} is {trend}.")
    ld = country_data['datetime'].iloc[-1]
    fut = [ld + timedelta(days=i) for i in range(1, 4)]
    fut_ind = np.array([len(country_data) + i for i in range(1, 4)]).reshape(-1, 1)
    fut_sca = scaler.transform(fut_ind)
    fut_pred = model.predict(fut_sca)
    print("Predicted future air quality (PM2.5) values:")
    for i, aq in enumerate(fut_pred):
        print(f"Date {fut[i].strftime('%Y-%m-%d %H:%M:%S')}: {aq:.2f}")

country_name = input("Enter country name: ")
predict_temperature_trend(zip_file_path, country_name)
predictair(file_path, country_name)
```

```
Enter country name: India
```




```
Temperature trend in India is decreasing.
Predicted future temperatures:
Date 2025-01-28 10:30:00: 21.14°C
Date 2025-01-29 10:30:00: 21.07°C
Date 2025-01-30 10:30:00: 20.99°C
```




```
Air quality trend in India is increasing.
Predicted future air quality (PM2.5) values:
Date 2025-01-28 10:30:00: 205.04
Date 2025-01-29 10:30:00: 205.80
Date 2025-01-30 10:30:00: 206.56
```

CONCLUSION

The internship project revolves around weather data analysis as well as forecast through data visualization, regression analysis, and machine learning. Analyzing the datasets on the Earth's temperature and air quality, a number of observations on the data were considered-trends, variable-coupled relationships, and predictions concerning global and country-specific weather conditions.

So the main goal of the project is to create an adaptive and remarkable system capable of performing real-time updates of the dataset to report on changing weather patterns, air quality, and relationships. The realized model provides for future usability through forecasting based on historical data, which will result in planning and analysis for future use.

Describe the method used by the project. The performance of big data analysis coupled with machine learning for prediction showcases the power of data-driven insights in tackling real-world issues such as climate change, air pollution, and urban planning.

While letting the aspects such as building it into an app for user accessibility and further features boosts its cut-off, it goes a milestone ahead to be built on, towards more advanced weather predictive systems.