EXPLORE WEATHER TRENDS

Udacity: Data Analyst Nanodegree

February 4, 2020

By

Rohith Sowmithra

Hyderabad, India

Overview

In this project I have extracted the global temperature data, city-wise temperature data from Udacity portal, analyzed the two datasets, cleaned them up and compared my city temperature trend with global temperature trend.

Goals

1. Extract temperature data from Udacity portal.

2. Clean up the extracted data and plot the comparison chart.

3. Write observations from the plots drawn.

Tools Used

1. SQL: To extract data from the database.
2. Python: To perform data cleaning, compute moving averages and plotting the charts.
3. Jupyter Notebook: To write python code.

Sequence of steps to achieve Goals

We can achieve our goals using two approaches.

1. Write sql query to fetch the data specific to your city and compare it with global temperature trend by plotting a line chart.
2. Write sql query to fetch all cities data, ask the user for his country and city, separate out the data specific to his city and compare it with global temperature trend by plotting a comparison line chart.

In this report implementation of second approach has been explained as implementing first approach is already a part of it.

Step-1 Extract temperature data from Udacity portal

I have used basic sql queries to extract the data from datasets. Following are the queries used.

1. select * from city_data

2. select * from global_data

Above queries will fetch all the data from the mentioned two tables. Download the results generated by these two queries into two separate files namely cuty_temp.csv and global_temp.csv.

3. select city_data.year,city_data.city,city_data.country,city_data.avg_temp as city_avg_temp, global_data.avg_temp as global_avg_temp from city_data INNER JOIN global_data on city_data.year = global_data.year

Above query will get the data by joining the two tables based on "year" field. Download the result and name it as "city_global_temp.csv".

Step-2 Clean up the extracted data and plot the comparison chart

To achieve this, I have used Python Programming language. There are a wide variety of IDEs available for python programming, but I chose Jupyter Notebook for better readability.

Following are the steps involved in achieving the second goal.

1. I've used the following python libraries, read the csv files generated by the sql queries specified in above section.

```
# Import the needed libraries and read the csv files

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from scipy.stats import pearsonr

city_df = pd.read_csv("city_temp.csv")

global_df = pd.read_csv("global_temp.csv")

city_global_df = pd.read_csv("city_global_temp.csv")
```

2. Once the files are read see the table structure of each file as shown below.

```
# Display the number of observations in each file

print(f"Number of rows and columns in city_temp file are {city_df.shape}\n")

print(f"Number of observations in city_temp file are: {city_df.shape[0]}\n")

print(f"Number of rows and columns in global_temp file are {global_df.shape}\n")

print(f"Number of observations in global_temp file are: {global_df.shape[0]}\n")

print(f"Number of rows and columns in global_temperatures file are {city_global_df.shape}\n")

print(f"Number of observations in global_temperatures file are: {city_global_df.shape[0]}\n")
```

```
Number of rows and columns in city_temp file are (70792, 4)

Number of observations in city_temp file are: 70792

Number of rows and columns in global_temp file are (266, 2)

Number of observations in global_temp file are: 266

Number of rows and columns in global_temperatures file are (70239, 5)

Number of observations in global_temperatures file are: 70239
```

3. Ask the user to input his country name as shown below.

```
# Ask to input the country name

your_country = str(input("Enter your country name: "))
```

```
Enter your country name: India
```

4. Now fetch the data specific to India into a separate dataframe as shown below.

```
# Separate out the country data from all cities data and show the statistics

filtered_country_df = city_df[city_df['country'] == your_country].copy()

print(f"Top 5 observations in {your_country}")

print("============================================")

print(f"{filtered_country_df.head(5)}\n")

print(f"Number of observations in {your_country} are: {filtered_country_df.shape[0]}\n")

print(f"Number of null observations in {your_country} are: {filtered_country_df['avg_temp'].isnull().sum()}\n")

print(f"Number of null observations in global file are: {global_df['avg_temp'].isna().sum(axis=0)}")
```

```
Top 5 observations in India
============================================
      year  city country  avg_temp
1055  1796  Agra   India     25.05
1056  1797  Agra   India     26.71
1057  1798  Agra   India     24.19
1058  1799  Agra   India     25.31
1059  1800  Agra   India     25.25

Number of observations in India are: 4756

Number of null observations in India are: 236

Number of null observations in global file are: 0
```

5. Ask the user to input his city name.

```
# Ask to input the city name

your_city = str(input("Enter your city name: "))
```

```
Enter your city name: Hyderabad
```

6. Fetch the records of your city into a separate dataframe as shown below.

```
# Separate out the city data from country data and show the statistics

filtered_city_df = filtered_country_df[filtered_country_df['city'] == your_city].copy()

print(f"Top 5 observations in {your_city}")

print("============================================")

print(f"{filtered_city_df.head(5)}\n")

print(f"Number of observations in {your_city} are: {filtered_city_df.shape[0]}\n")

print(f"Number of null observations in {your_city} are: {filtered_city_df['avg_temp'].isnull().sum()}\n")

print(f"Number of null observations in global file are: {global_df['avg_temp'].isna().sum(axis=0)}")
```

```
Top 5 observations in Hyderabad
============================================
       year       city country  avg_temp
26146  1796  Hyderabad   India     26.53
26147  1797  Hyderabad   India     27.48
26148  1798  Hyderabad   India     26.20
26149  1799  Hyderabad   India     26.84
26150  1800  Hyderabad   India     26.88

Number of observations in Hyderabad are: 218

Number of null observations in Hyderabad are: 7

Number of null observations in global file are: 0
```

7. Find the mean and standard deviation of the chosen city and global_temp file as shown below.

```
# Get the statistics of your city and global temperatures

city_mean = filtered_city_df['avg_temp'].mean()

city_stddev = filtered_city_df['avg_temp'].std()

global_mean = global_df['avg_temp'].mean()

global_stddev = global_df['avg_temp'].std()

data = np.array([[city_mean,city_stddev],[global_mean,global_stddev]])

pd.DataFrame(data,columns=["Mean","Std Dev"],index=["City","Global"])
```

	Mean	Std Dev
City	26.861564	0.542281
Global	8.369474	0.584747

Our city temperature is way too much hotter than global temperature.

8. As shown in above screenshot, we see that there are 7 missing values in the city chosen. There are many strategies that can be followed to fill-in the missing values, but, I've chosen to replace missing values with mean which is a bit appropriate here. Number of missing values/null observations become zero after replacement which is shown below.

```
# Fill missing values with mean

filtered_city_df.fillna(city_mean,inplace=True)

print(f"Number of observations in {your_city} are: {filtered_city_df.shape[0]}\n")

print(f"Number of null observations in {your_city} are: {filtered_city_df['avg_temp'].isnull().sum()}\n")
```

```
Number of observations in Hyderabad are: 218

Number of null observations in Hyderabad are: 0
```

9. Now compute simple moving averages for the “avg_temp” column in both the datasets. I’ve chosen 10-year moving average as values lower than 10 are giving more coarser graphs.

```
# Compute moving average of 10 yrs for Your city temperature and display observations

filtered_city_df["city_10yr_moving_avg"] = filtered_city_df["avg_temp"].rolling(window=10).mean()

print(filtered_city_df.head(15))
```

```
# Compute moving average of 10 yrs for Global temperature and display observations

global_df["global_10yr_moving_avg"] = global_df["avg_temp"].rolling(window=10).mean()

print(global_df.head(15))
```

10. Join both the datasets on the field “year” as our city has data from 1796 and global_temp data starts from 1750. In order to maintain consistency we join them based on year as shown below.

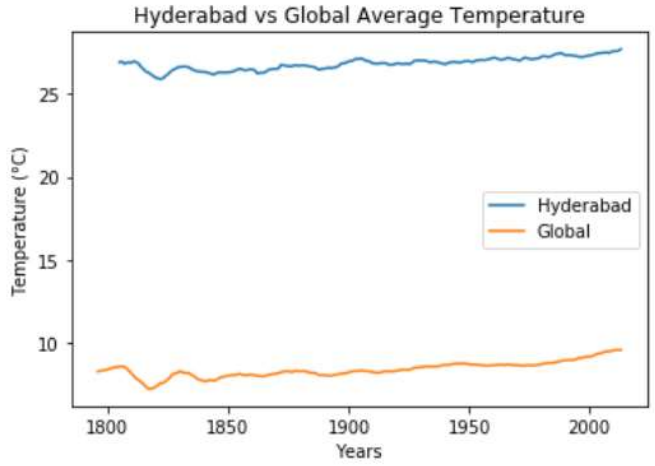
```
combined_dataset = pd.merge(filtered_city_df,global_df,on = "year")

print(combined_dataset)
```

```
    year       city country  avg_temp_x  city_10yr_moving_avg  avg_temp_y  \
0   1796  Hyderabad   India   26.530000                   NaN        8.27
1   1797  Hyderabad   India   27.480000                   NaN        8.51
2   1798  Hyderabad   India   26.200000                   NaN        8.67
3   1799  Hyderabad   India   26.840000                   NaN        8.51
4   1800  Hyderabad   India   26.880000                   NaN        8.48
5   1801  Hyderabad   India   26.050000                   NaN        8.59
6   1802  Hyderabad   India   27.440000                   NaN        8.58
7   1803  Hyderabad   India   27.220000                   NaN        8.50
8   1804  Hyderabad   India   27.650000                   NaN        8.84
9   1805  Hyderabad   India   27.020000             26.931000        8.56
10  1806  Hyderabad   India   26.890000             26.967000        8.43
11  1807  Hyderabad   India   26.150000             26.834000        8.28
```

From above figure we see that dataset starts from the year 1796 instead of 1750 due to inner join operation.

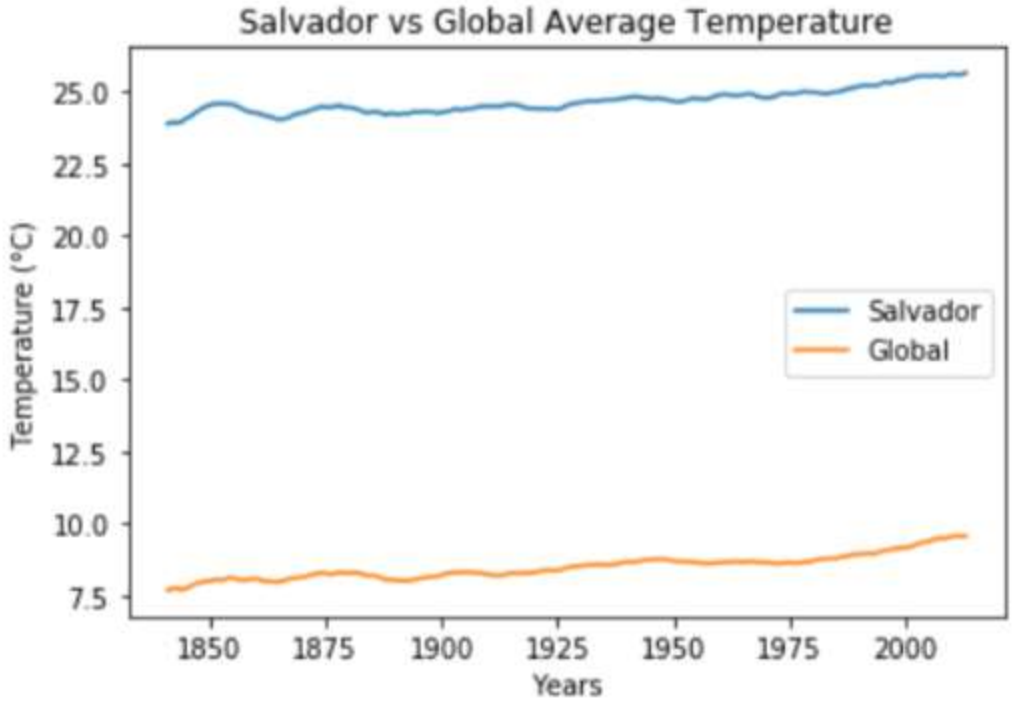
11. Now plot a comparison line chart “City vs Global Temperature” as shown below.

```
# Plot a comparison line chart: Your City vs Global Temperature

plt.plot(combined_dataset['year'], combined_dataset['city_10yr_moving_avg'], label=your_city)

plt.plot(combined_dataset['year'], combined_dataset['global_10yr_moving_avg'], label='Global'

plt.legend()

plt.xlabel("Years")

plt.ylabel("Temperature (°C)")

plt.title(f"{your_city} vs Global Average Temperature")

plt.show()
```

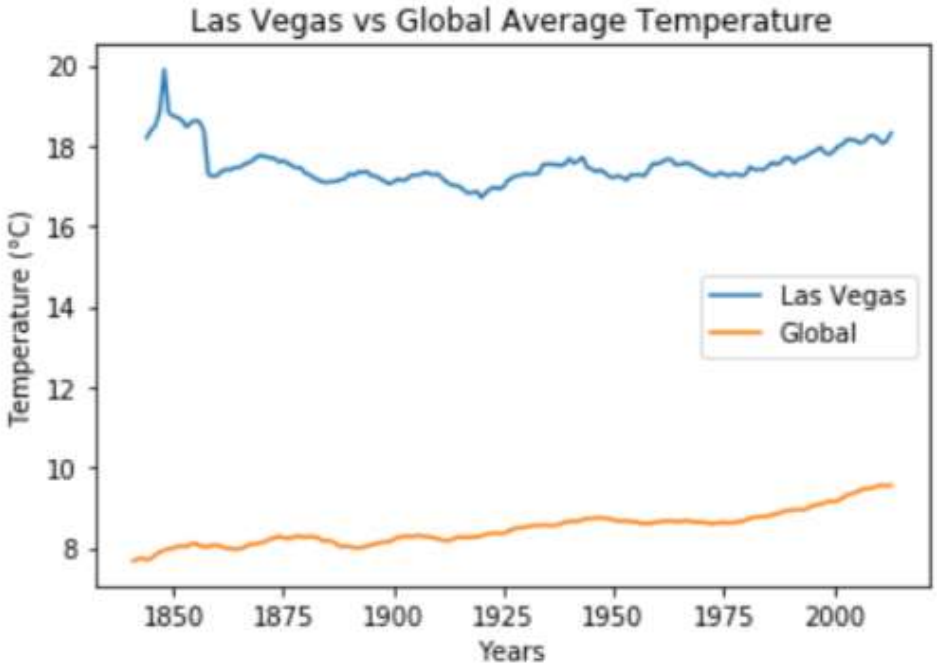


12. We can also compute correlation coefficient between city avg_temp and global avg_temp fields and find which city has more affinity with global temperature. I have used pearson correlation coefficient for this and observed the following.
    - Maximum correlation is obtained for: ('Brazil', 'Salvador', 0.8710937673186244)
    - Minimum correlation is obtained for: ('United States', 'Las Vegas', 0.09240067152608472)
    - Correlation for my city (Hyderabad) is ('India', 'Hyderabad', 0.7243811325384892)

13. Comparison line chart for the city ('Salvador') with maximum correlation value is shown below.



14. Comparison line chart for the city (Las Vegas') with maximum correlation value is shown below.



Observations

1. From Hyderabad vs Global Temperature graph, we see that average temperature of both city and global increases with time, which means that earth is becoming hotter every year.
2. Global temperature is continuously increasing since 1975. Similar is the case with Hyderabad city.
3. Comparison graph clearly depicts that Hyderabad city temperature is extremely hot than Global temperature.
4. 72% of the time Hyderabad's temperature has similar trend (Upward/Downward) when compared to Global temperature.
5. 87% of the time Salvador's (City in Brazil) temperature has similar trend with global temperature while only 9% of the time Las Vegas's (City in the United States) trend matches with global temperature.

Note: Image used in this report has been copied from google for reference.