Udacity Data Analyst Nanodegree

Project 1 : **Explore Weather Trends**

Date of submission: 16. January 2020 *Student: Chris Catacata*

Extract the data

The task was to export temperature data for the world and the closest big city to where I live. A list of countries and their respective cities can be found in the `city_list` table. To find out the cities in Germany, this was the SQL script I executed :

```
-- Cities in Germany (1) --
select *
from city_list
where country = 'Germany';
```

This yields three cities : `Berlin, Hamburg` and `Munich`. Of these cities, the one closest to where I live is **Munich**. To get the temperature data for Munich, I ran this script :

```
-- Temperature over time in Munich (2) --
select *
from city_data
where city = 'Munich';
```

The global temperature data in the `global_data` table is also necessary for comparison. To extract this data, I used this script :

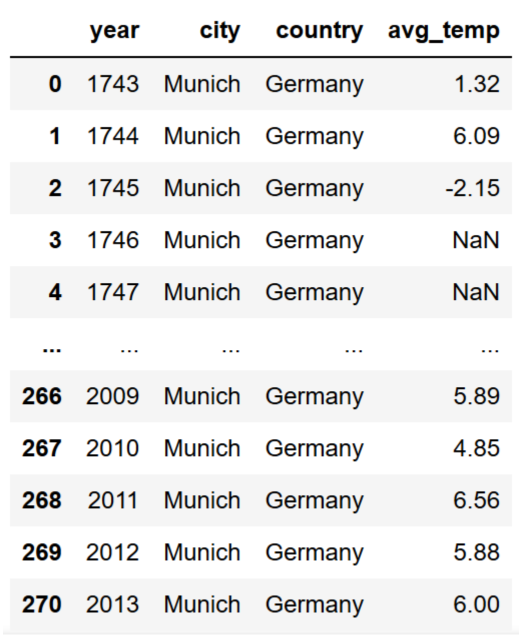
```
-- Global temperature (3) --
select *
from global_data;
```

Open the data

I saved the results of scripts (2) and (3) as CSV files. I have used Pandas from time to time (but not in a data analysis context) and decided to challenge myself in this project. The CSV files were loaded as dataframes. I created a Jupyter notebook where the CSV files were loaded.

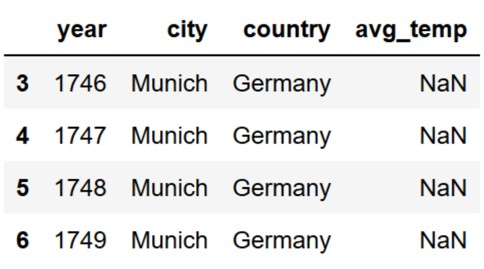
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

munich_df = pd.read_csv(r".\MunichData.csv", header = 0, delimiter = ',')
global_df = pd.read_csv(r".\GlobalData.csv", header = 0, delimiter = ',')

# Show Munich temperature data as a dataframe
munich_df

# Show global temperature data as a dataframe
global_df
```

Before plotting, I wanted to inspect the data by showing the dataframe. In Figure 1, there are missing (NaN) values in the `avg_temp` column. To check how many missing values there are, I used the following method : `munich_df.isnull().sum()`. Indeed, this shows that there are four missing values in the `avg_temp` column. Specifically, there are no temperature records from 1746 to 1749 as shown in Figure 2. Similar inspection to the `global_df` dataframe shows no missing values.



	year	city	country	avg_temp
0	1743	Munich	Germany	1.32
1	1744	Munich	Germany	6.09
2	1745	Munich	Germany	-2.15
3	1746	Munich	Germany	NaN
4	1747	Munich	Germany	NaN
...	...	...	...	...
266	2009	Munich	Germany	5.89
267	2010	Munich	Germany	4.85
268	2011	Munich	Germany	6.56
269	2012	Munich	Germany	5.88
270	2013	Munich	Germany	6.00

Figure 1 – Munich temperature data as Pandas dataframe



	year	city	country	avg_temp
3	1746	Munich	Germany	NaN
4	1747	Munich	Germany	NaN
5	1748	Munich	Germany	NaN
6	1749	Munich	Germany	NaN

Figure 2 – Missing data in the `munich_df` dataframe using munich_df[munich_df['avg_temp'].isnull()]

Since the missing data only comprises 1.4% of all the average temperatures, I decided to start the comparison by dropping all the rows before 1750, i.e. the missing values and the three prior rows. This allows for a better comparison with the global temperature data. In the end, both dataframes were cleaned using `munich_df = munich_df['munich_df['year'] >= 1750]` and `global_df = global_df[(global_df['year'] >= 1750) (global_df['year'] <= 2013)] (Since the temperature data for Munich is only up to 2013..`

Create the line chart

After cleaning the data, I used the Pandas function `rolling` to compute the moving average. I decided to compute this moving average on a decade-basis (using the argument `window = 10`). I applied this to both the Munich and global temperature dataframes. Figure 3 shows the comparison between the Munich and global temperature data as plotted line graphs.

Make observations

It is apparent from Figure 3 that Munich's temperature over the years has been consistently below the global average ($\bar{x}_{World} = 8.37, s_{World} = 0.58, \bar{x}_{Munich} = 4.64, s_{Munich} = 0.76$) throughout the past 250 years. Munich's average temperature (aggregated in the entire dataset) is half of the global temperature data. This might be related to its elevation (519 meters[1]). Although the temperature is lower, one can see from the graphs that Munich's temperature curve roughly follows the same trend as that of the world. This can be particularly observed in the dip and the rise in the temperature between 1800 and 1850 and in the consistent temperature rise after 1950.

1. From https://www.muenchen.de/sehenswuerdigkeiten/muenchen-in-zahlen.html

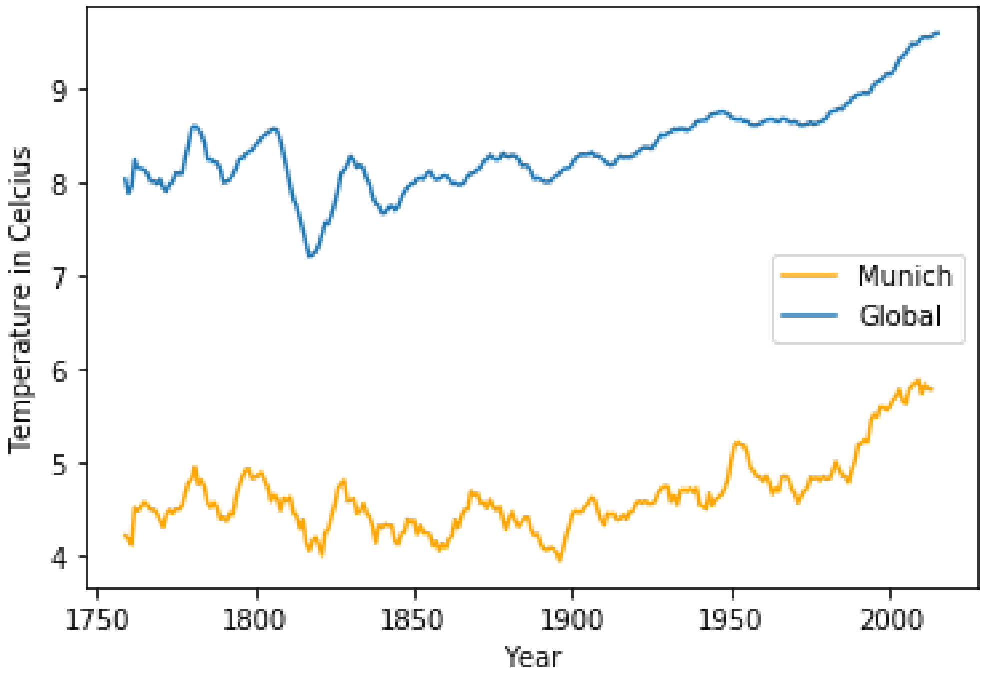


Figure 3 – Smoothed (using moving averages) plot of temperature data for Munich and the world between 1750 and 2013.

Furthermore, in both global and Munich temperatures, one can observe an exponential temperature rise after 1950. Following this trend, it would be safe to say that the world is indeed getting warmer.
Out of curiosity, I also extracted the data for the other German cities in the dataset : Berlin and Hamburg. The temperature data for Berlin and Hamburg has likewise been cleaned as in the Munich and global temperature data. Figure 4 shows the graphs of the temperature development in the aforementioned German cities as well as for the world (with a slightly thicker line). The graphs for Berlin and Hamburg are closer to the global averages throughout the years and likewise have similar fluctuations as in the Munich graph. The Pearson correlation of the global temperatures and temperatures in these German cities (ranging between 0.52 and 0.56) shows that there is a positive correlation, i.e. the temperatures in these cities increase as the global average temperature increases. This observation might seem trivial but actually provides the assertion that the global temperature is a sound point of reference for predicting temperatures all over the world.

In an attempt to predict the global temperatures for the years to come, I used the Python library `fbprophet`[2] which can be used to forecast time series data, as what has been given in this task. I had to perform some data manipulation in the source dataset (rename the columns and convert the year to a datetime object) to fit the model that this library requires. The data manipulation and visualisation done to achieve forecasted values is shown in the following script :

2. https ://facebook.github.io/prophet/

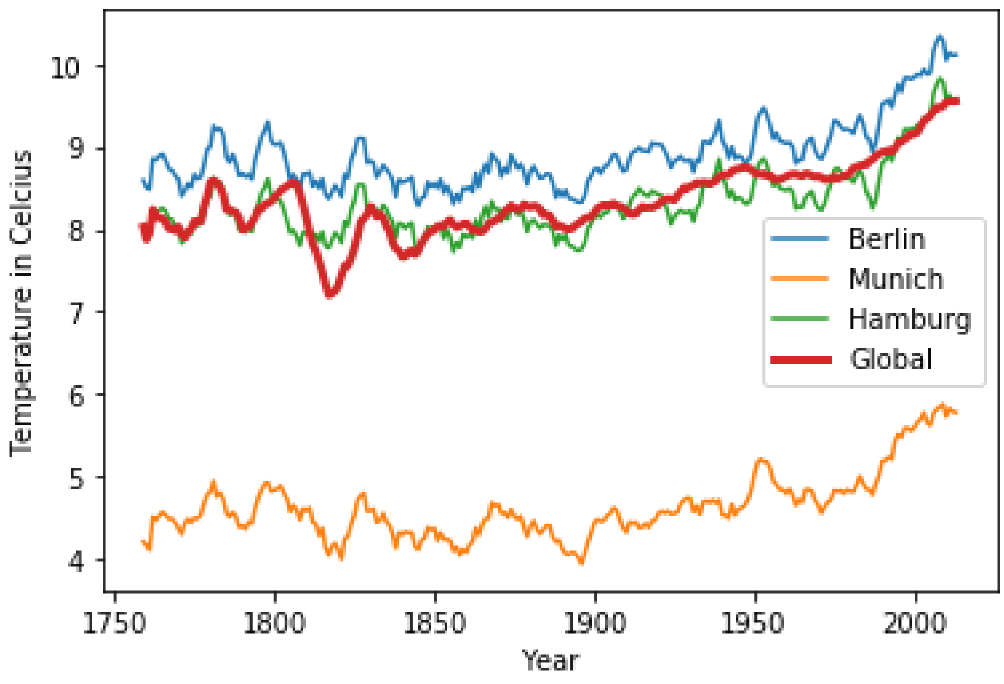


Figure 4 – Temperature over the years in German cities vs global temperature data

```
### Predict Global Temperatures ###
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from fbprophet import Prophet
from datetime import datetime

#helper function to convert year to datetime
def convertDate(year):
    return datetime.strptime(str(year) + '-12-31', '%Y-%m-%d')

#rename columns
global_df = global_df.rename(columns = {'year':'ds', 'avg_temp':'y'}

#convert year to datetime
global_df['ds'] = global_df['ds'].apply(convertDate)

#create model to predict global temperatures in the next 10 years
model = Prophet(seasonality_mode = 'multiplicative', changepoint_prior_scale = 0.25)
model.fit(global_df)

future = model.make_future_dataframe(periods = 10, freq = 'Y')
forecast = model.predict(future)

#graph actual temperatures and forecasted temperatures
plt.plot(global_df['ds'], global_df['y'], label = 'Actual Global')
plt.plot(forecast['ds'], forecast['yhat'], label = 'Forecasted temperature')
plt.xlabel('Year')
plt.ylabel('Temperature in Celcius')
plt.legend(loc = 'lower right')
plt.show()
```

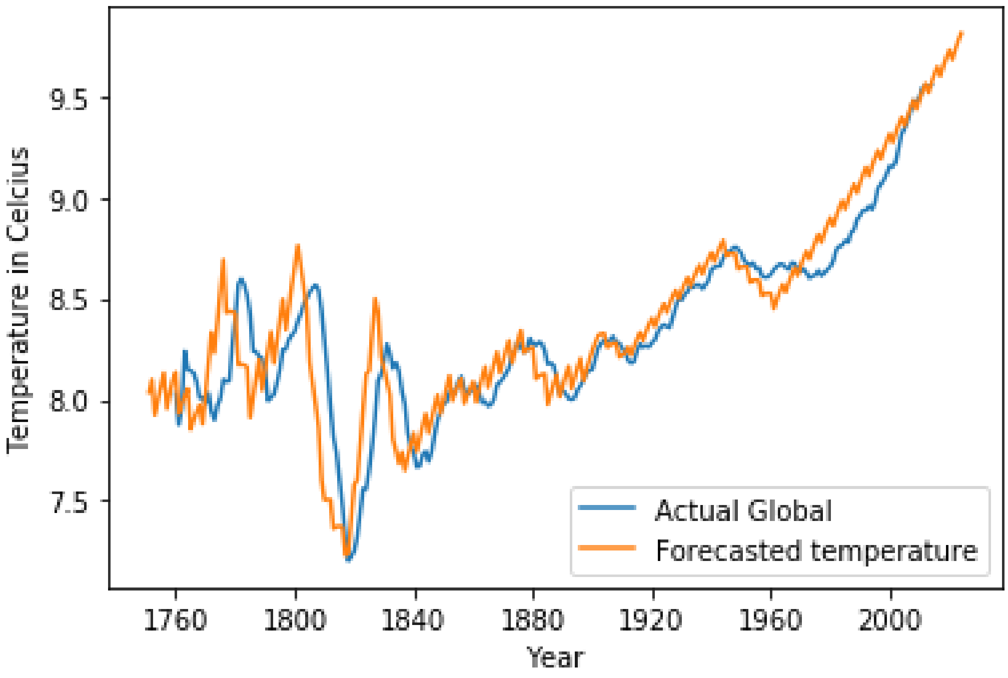
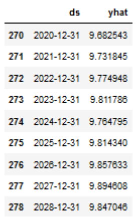


Figure 5 – Actual vs Forecasted global temperatures



	ds	yhat
270	2020-12-31	9.682543
271	2021-12-31	9.731845
272	2022-12-31	9.774948
273	2023-12-31	9.811786
274	2024-12-31	9.764795
275	2025-12-31	9.814340
276	2026-12-31	9.857633
277	2027-12-31	9.894608
278	2028-12-31	9.847046

Figure 6 – Forecasted values from 2020 to 2028

Plotting the actual and forecasted values in the same graph makes it easier to see how well the model fits. Figure 5 shows this comparison. I still do not have a good understanding of overfitting but the forecasted temperature seems to model the global temperature well (without calculating for goodness-of-fit). Based on this, the global temperatures will continually increase in the coming years. Forecasted values are shown in Figure 6.