nment3-jaamie-maarsh-joy-martin-2

November 1, 2023

IE6600 Computisation and Visualisation - Assignment 3

[1]:
```
import pandas as pd
import plotly.express as px
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
#This command is to ignore all the warnings
warnings.filterwarnings("ignore")
```

[2]:
```
# Defining the file paths as lists
file_paths = [
    '/Users/jaamiemaarshj/Desktop/  DAE Course Materials/Computization and␣
 ↪Visualisation/Assignment-3/Beijing.csv',
    '/Users/jaamiemaarshj/Desktop/  DAE Course Materials/Computization and␣
 ↪Visualisation/Assignment-3/Chengdu.csv',
    '/Users/jaamiemaarshj/Desktop/  DAE Course Materials/Computization and␣
 ↪Visualisation/Assignment-3/Shanghai.csv',
    '/Users/jaamiemaarshj/Desktop/  DAE Course Materials/Computization and␣
 ↪Visualisation/Assignment-3/Guangzhou.csv',
    '/Users/jaamiemaarshj/Desktop/  DAE Course Materials/Computization and␣
 ↪Visualisation/Assignment-3/Shenyang.csv'
]

# reading of the datasets into a DataFrames
City_datasets = [pd.read_csv(file, low_memory=False) for file in file_paths]

Cities = []
for df, file in zip(City_datasets, file_paths):
# Extract the city name by from the file path by splitting them into lists and␣
 ↪then choosing the last item on the list
# ignoring the ".csv" portion of the datasets.
    df['city'] = file.split('/')[-1][:-4]
    Cities.append(df)

# Concatenate the datasets
merged_Cities_df = pd.concat(Cities)
```

```
merged_Cities_df.tail(5)
```

[2]:
```
        year  month  day  hour  season     PM  DEWP   HUMI    PRES  TEMP   Iws  \
52579   2015     12   31    19     4.0  166.0 -10.0  92.42  1031.0  -9.0   2.0
52580   2015     12   31    20     4.0  259.0 -10.0  79.10  1030.0  -7.0   5.0
52581   2015     12   31    21     4.0  368.0 -10.0  79.10  1030.0  -7.0   8.0
52582   2015     12   31    22     4.0  319.0 -10.0  79.10  1028.0  -7.0  11.0
52583   2015     12   31    23     4.0  275.0  -9.0  79.26  1028.0  -6.0  12.0

        precipitation  Iprec      city
52579             0.0    0.0  Shenyang
52580             0.0    0.0  Shenyang
52581             0.0    0.0  Shenyang
52582             NaN    NaN  Shenyang
52583             0.0    0.0  Shenyang
```

[3]:
```
#using the backup/copy of the existing file for date time updation
Updated_Cities=merged_Cities_df.copy()
#adding a seperate field to the created table
Updated_Cities['date'] = pd.
 ↪to_datetime(Updated_Cities[['year','month','day','hour']])
Updated_Cities.head()
```

[3]:
```
   year  month  day  hour  season  PM  DEWP  HUMI    PRES  TEMP    Iws  \
0  2010      1    1     0     4.0 NaN -21.0  43.0  1021.0 -11.0   1.79
1  2010      1    1     1     4.0 NaN -21.0  47.0  1020.0 -12.0   4.92
2  2010      1    1     2     4.0 NaN -21.0  43.0  1019.0 -11.0   6.71
3  2010      1    1     3     4.0 NaN -21.0  55.0  1019.0 -14.0   9.84
4  2010      1    1     4     4.0 NaN -20.0  51.0  1018.0 -12.0  12.97

   precipitation  Iprec     city                date
0            0.0    0.0  Beijing 2010-01-01 00:00:00
1            0.0    0.0  Beijing 2010-01-01 01:00:00
2            0.0    0.0  Beijing 2010-01-01 02:00:00
3            0.0    0.0  Beijing 2010-01-01 03:00:00
4            0.0    0.0  Beijing 2010-01-01 04:00:00
```

[4]:
```
Updated_Cities.isna().sum()
```

[4]:
```
year                 0
month                0
day                  0
hour                 0
season               1
PM               95562
DEWP              1240
HUMI              1568
```

```
PRES              1581
TEMP              1238
Iws               1243
precipitation    20212
Iprec            20212
city                 0
date                 0
dtype: int64
```

[5]:

```
Updated_Cities.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 262920 entries, 0 to 52583
Data columns (total 15 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   year           262920 non-null  int64
 1   month          262920 non-null  int64
 2   day            262920 non-null  int64
 3   hour           262920 non-null  int64
 4   season         262919 non-null  float64
 5   PM             167358 non-null  float64
 6   DEWP           261680 non-null  float64
 7   HUMI           261352 non-null  float64
 8   PRES           261339 non-null  float64
 9   TEMP           261682 non-null  float64
 10  Iws            261677 non-null  float64
 11  precipitation  242708 non-null  float64
 12  Iprec          242708 non-null  float64
 13  city           262920 non-null  object
 14  date           262920 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(9), int64(4), object(1)
memory usage: 32.1+ MB
```

Question 1: Perform data cleaning, impute missing values, do feature engineering for “Season” feature map it with seasons. (10 Marks)

[6]:

```
# Assuming you have already defined merged_df
#new_df = merged_df.copy()  # Create a copy to avoid modifying the original␣
 ↪DataFrame

# Handling of missing values in the 'Seasons' column
Updated_Cities['season'].fillna(0, inplace=True)

# updating the missing values in the respective column with its mean values
mean_PM = int(Updated_Cities['PM'].mean())
Updated_Cities['PM'].fillna(mean_PM, inplace=True)
```

```python
# Forward fill to fill in missing values in rest of the columns
Updated_Cities.fillna(method='ffill', inplace=True)

# Customizing a function to map numerical seasons with categorical names
def mapping_season(x):
    if x == 1:
        return "Spring"
    if x == 2:
        return "Summer"
    if x == 3:
        return "Autumn"
    if x == 4:
        return "Winter"

# Apply the season mapping function to the 'season' column
Updated_Cities['season'] = Updated_Cities['season'].apply(mapping_season)

# Verification of data and its counts
display(Updated_Cities['season'].value_counts())
print("The total seasons are:", Updated_Cities.shape)

# Cross verification of the dataframe
display(Updated_Cities)
```

```
Spring    66240
Summer    66240
Autumn    65520
Winter    64919
Name: season, dtype: int64
```

```
The total seasons are: (262920, 15)
```

```
       year  month  day  hour  season     PM  DEWP   HUMI    PRES  TEMP  \
0      2010      1    1     0  Winter   73.0 -21.0  43.00  1021.0 -11.0
1      2010      1    1     1  Winter   73.0 -21.0  47.00  1020.0 -12.0
2      2010      1    1     2  Winter   73.0 -21.0  43.00  1019.0 -11.0
3      2010      1    1     3  Winter   73.0 -21.0  55.00  1019.0 -14.0
4      2010      1    1     4  Winter   73.0 -20.0  51.00  1018.0 -12.0
…       …     …    …    …       …     …     …      …       …     …
52579  2015     12   31    19  Winter  166.0 -10.0  92.42  1031.0  -9.0
52580  2015     12   31    20  Winter  259.0 -10.0  79.10  1030.0  -7.0
52581  2015     12   31    21  Winter  368.0 -10.0  79.10  1030.0  -7.0
52582  2015     12   31    22  Winter  319.0 -10.0  79.10  1028.0  -7.0
52583  2015     12   31    23  Winter  275.0  -9.0  79.26  1028.0  -6.0

         Iws  precipitation  Iprec     city                date
0       1.79            0.0    0.0  Beijing 2010-01-01 00:00:00
1       4.92            0.0    0.0  Beijing 2010-01-01 01:00:00
```

```
2        6.71             0.0     0.0    Beijing 2010-01-01 02:00:00
3        9.84             0.0     0.0    Beijing 2010-01-01 03:00:00
4       12.97             0.0     0.0    Beijing 2010-01-01 04:00:00
…       …             …    …       …                   …
52579    2.00             0.0     0.0   Shenyang 2015-12-31 19:00:00
52580    5.00             0.0     0.0   Shenyang 2015-12-31 20:00:00
52581    8.00             0.0     0.0   Shenyang 2015-12-31 21:00:00
52582   11.00             0.0     0.0   Shenyang 2015-12-31 22:00:00
52583   12.00             0.0     0.0   Shenyang 2015-12-31 23:00:00

[262920 rows x 15 columns]
```

Insights: The "Updated_Cities" dataset holds the atmospheric data for the above mentioned 5 cities of various parameters based on their seasons. It is also found that there is a lot of missing data in

Question 2: Generate a line chart showing the temperature (y-axis) and dates (x-axis) for one of the five cities. Is there a noticeable seasonal pattern?

[7]:
```
#Creating specific columns to incorporate date and time values
Updated_Cities['date'] = pd.to_datetime(Updated_Cities['year'].map(str)+'/
 ↪'+Updated_Cities['month'].map(str)+'/'+Updated_Cities['day'].map(str))
Updated_Cities['date_time'] = Updated_Cities['date']+pd.
 ↪TimedeltaIndex(Updated_Cities['hour'],unit='H')
Updated_Cities.head()
```

[7]:
```
   year  month  day  hour  season    PM  DEWP  HUMI    PRES  TEMP    Iws  \
0  2010      1    1     0  Winter  73.0 -21.0  43.0  1021.0 -11.0   1.79
1  2010      1    1     1  Winter  73.0 -21.0  47.0  1020.0 -12.0   4.92
2  2010      1    1     2  Winter  73.0 -21.0  43.0  1019.0 -11.0   6.71
3  2010      1    1     3  Winter  73.0 -21.0  55.0  1019.0 -14.0   9.84
4  2010      1    1     4  Winter  73.0 -20.0  51.0  1018.0 -12.0  12.97

   precipitation  Iprec     city       date           date_time
0            0.0    0.0  Beijing 2010-01-01 2010-01-01 00:00:00
1            0.0    0.0  Beijing 2010-01-01 2010-01-01 01:00:00
2            0.0    0.0  Beijing 2010-01-01 2010-01-01 02:00:00
3            0.0    0.0  Beijing 2010-01-01 2010-01-01 03:00:00
4            0.0    0.0  Beijing 2010-01-01 2010-01-01 04:00:00
```

[8]:
```
Temperature_Season_Beijing_df =␣
 ↪Updated_Cities[Updated_Cities['city']=='Beijing']
print("Answer 2: The line chart for the temperature trends in Beijing ")
print("----------------------------------------------------------------")
fig = px.
 ↪line(x=Temperature_Season_Beijing_df['date_time'],y=Temperature_Season_Beijing_df['TEMP'],␣
 ↪title='Temperature trends across years and seasons in Beijing',labels={'x':
 ↪'Time','y':'Temperature'}, line_shape='linear')
```

```
fig.show()
```

```
Answer 2: The line chart for the temperature trends in Beijing
---------------------------------------------------------------
```

Insights: The temperature trend forms a sinusoidal wave pattern wherein they repeat the pattern repetatively. For instance, considering the data after 2016, the temperature would be the lowest in the month of January and would reach its peak in July and then gradually reduce.

Question 3: Create a boxplot showing the temperature values aggregated by month for one of the five cities.

[45]:
```
#considering the city as the same as above - Beijing
print("Answer 3: The box plot for the month wise temperature distribution in␣
 ↪Beijing ")
print("---------------------------------------------------------------")
fig = px.
 ↪box(Temperature_Season_Beijing_df,x='month',y='TEMP',title='Temperature␣
 ↪distribution month-wise - Beijing',labels={'TEMP':'Temperature'})
fig.show()
```

```
Answer 3: The box plot for the month wise temperature distribution in Beijing
---------------------------------------------------------------
```

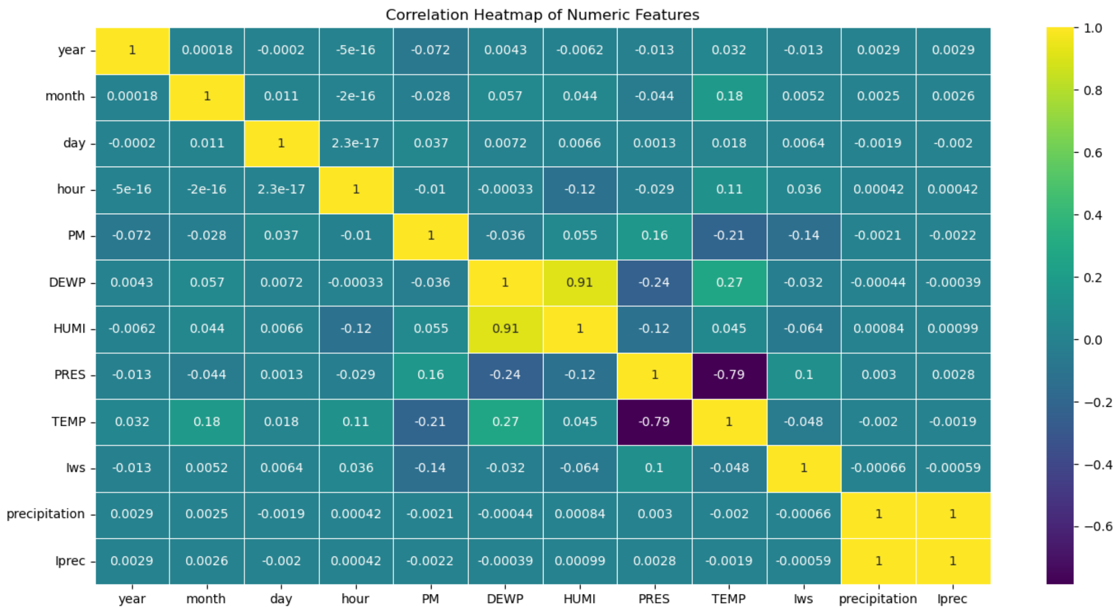
Insights: As found above, the temperature distribution across the months follows a sinusoidal wave pattern wherein the temperature raises from January and then starts reducing from July. Also, since the median is almost distributed in the centre for all which determines the temperature for their respective month where the majority number of times that value is achieved.

Question 4: Create a heatmap to generate correlation between numeric features

[11]:
```
print("Answer 4: The heatmap determining the correlation between numeric␣
 ↪features ")
print("---------------------------------------------------------------")
plt.figure(figsize=(16,8))
sns.heatmap(Updated_Cities.corr(),cmap='viridis', annot=True, linewidth=.5)
plt.title("Correlation Heatmap of Numeric Features")
```

```
Answer 4: The heatmap determining the correlation between numeric features
---------------------------------------------------------------
```

[11]:
```
Text(0.5, 1.0, 'Correlation Heatmap of Numeric Features')
```



Insights: 1) The pressure(PRES) and the temperature are inversely proportional wherein when one parameter increases the other decreases. 2) there is a heavy correlation between humidity (HUMI) and precipitation wherein they are found to be dependent of each other. 3) PM and temperature parameters fall on the each ends of the spectrum

Question 5: Create a scatter plot using two features of your choice. Choose a pair of features that you believe have some correlation between them. Based on your visualization, do they seem to be correlated?

[12]:
```python
print("Answer 5: The correlation scatter plot between features ")
print("-----------------------------------------------------------------")
fig = px.scatter(Updated_Cities,x='PM',y='TEMP',title=' correlation between the Particle concentration mass and the Temperature',labels={"PM":"Particle mass (ug/m3)","TEMP":"Temperature (Celcius)"})
fig.show()
```

```
Answer 5: The correlation scatter plot between features
-----------------------------------------------------------------
```

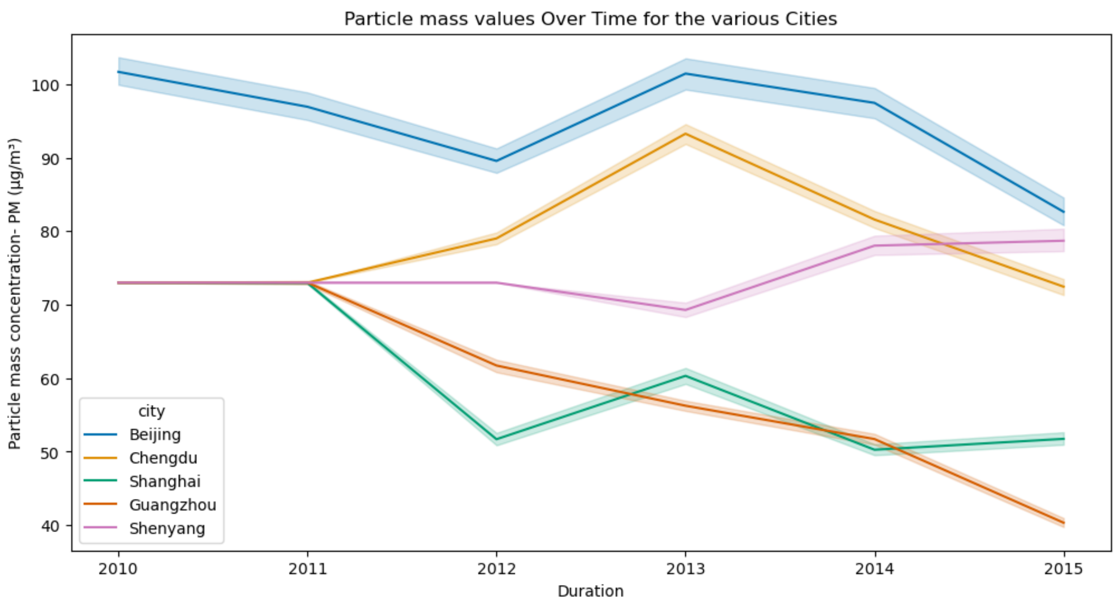
Insights: The mass of the particle kept on increasing when the temperature is decreasing and vice versa, inversely proportional. It is also observed that the majority of the mass is found between is less than 400 (ug/m3)

Question 6: Create a single plot that illustrates the value of the PM column over time for each of the four cities. Color and label each city differently so that they can be distinguished easily

[9]:
```python
print("Answer 6: The correlation line plot between features ")
print("-----------------------------------------------------------------")
plt.figure(figsize=(12,6))
```

```
sns.lineplot(data=Updated_Cities,x='year',y='PM',hue='city', palette
    ='colorblind')
plt.title('Particle mass values Over Time for the various Cities')
plt.xlabel('Duration')
plt.ylabel('Particle mass concentration- PM (µg/m³)')
plt.show()
```

```
Answer 6: The correlation scatter plot between features
---------------------------------------------------------------
```



Insights: Observations from the chart is that the Beijing data is the most consistant across the entire time frame scale and also has the maximum overall PM value ranging upto nearly 1000 µg/m³. On the other hand, the PM for Guangzhou has been the less constantly across the 2010-2016 time frame when resulted in the healthier air compared to the other cities.

Question 7: How do meteorological factors (DEWP, HUMI, PRES, TEMP) correlate with PM levels? Create scatter plots to explore relationships.

[62]:
```
import plotly.graph_objs as go
from plotly.subplots import make_subplots

# Assuming you have a DataFrame 'df'

# Create a subplot grid with 2 rows and 2 columns
fig = make_subplots(rows=2, cols=2, subplot_titles=('DEWP vs. PM', 'HUMI vs.
    PM', 'PRES vs. PM', 'TEMP vs. PM'))

# plotting for DEWP vs. PM
```

```
scatter_PM_DEW = go.Scatter(x=Updated_Cities['DEWP'], y=Updated_Cities['PM'],␣
 ↪mode='markers', name='DEWP vs. PM')
fig.add_trace(scatter_PM_DEW, row=1, col=1)

# plotting for HUMI vs. PM
scatter_HUMI_PM = go.Scatter(x=Updated_Cities['HUMI'], y=Updated_Cities['PM'],␣
 ↪mode='markers', name='HUMI vs. PM')
fig.add_trace(scatter_HUMI_PM, row=1, col=2)

# plotting for PRES vs. PM
scatter_PRES_PM = go.Scatter(x=Updated_Cities['PRES'], y=Updated_Cities['PM'],␣
 ↪mode='markers', name='PRES vs. PM')
fig.add_trace(scatter_PRES_PM, row=2, col=1)

# plotting for TEMP vs. PM
scatter_TEMP_PM = go.Scatter(x=Updated_Cities['TEMP'], y=Updated_Cities['PM'],␣
 ↪mode='markers', name='TEMP vs. PM')
fig.add_trace(scatter_TEMP_PM, row=2, col=2)

# defining axis labels for each of the scatterplots
fig.update_xaxes(title_text='Dew Point (°C)', row=1, col=1)
fig.update_xaxes(title_text='Humidity ', row=1, col=2)
fig.update_xaxes(title_text='Pressure (hPa)', row=2, col=1)
fig.update_xaxes(title_text='Temperature (°C)', row=2, col=2)
fig.update_yaxes(title_text='PM Level (µg/m³)', row=1, col=1)

print("Answer 7: The correlation scatter plot between features ")
print("----------------------------------------------------------------")

# Set layout title
fig.update_layout(title_text='Scatter Plots of Atmospheric Factors vs its PM␣
 ↪Levels')

# Show the subplot grid
fig.show()
```

```
Answer 7: The correlation scatter plot between features
----------------------------------------------------------------
```

Insights: from the scatterplots, the PM are concentrated within an area or a certain range of values which means that they are constant apart from a few outlier values.

Question 8: How do PM levels compare between the five cities? Create bar charts or box plots for a city-to-city comparison.

[17]:
```
print("Answer 8: The correlation box plot between city comparisions ")
print("----------------------------------------------------------------")
```

```
fig = px.box(Updated_Cities,x='city',y='PM',title='PM level comparison between␣
 ↪cities',labels={'city':'Cities' , 'PM':"PM Level (µg/m³)"})
fig.show()
```

```
Answer 8: The correlation box plot between city comparisions
--------------------------------------------------------------
```

Insights: From the box plots, the median PM values are approximately found to be the same, which is around 100 µg/m³. Beijing and Shenyang has the most number of outlier values but Guangzhou seem to have a healthy air quality since there is less variation in the size of the particles.

Question 9: Create a line plot to show the seasonal distribution of precipitation levels and examine how it relates to PM levels

[61]:
```
#backup of the dataset
Cities_Copy_df=Updated_Cities.copy()

#Creating a new Date column with a specific reference point
Cities_Copy_df['Date'] = Cities_Copy_df.apply(lambda row: pd.
 ↪to_datetime(f"{int(row['year'])}-{int(row['month'])}-01"), axis=1)

#Grouping by the date column and Iprec
precip_df=Cities_Copy_df.groupby('Date')['Iprec'].sum().reset_index()

#Removing abnormal values to smoothen out data and its visualisation
precip_df=precip_df.loc[precip_df['Iprec']<10000]

print("Answer 9: The Precipitation distribution over time ")
print("--------------------------------------------------------------")

#Plotting a lineplot
img = px.line(precip_df, x='Date', y='Iprec', title='Precipitation distribution␣
 ↪over time')
img.show()

particle_mass_df=Cities_Copy_df.groupby('Date')['PM'].sum().reset_index()

#Plotting a lineplot
img = px.line(particle_mass_df, x='Date', y='PM', title='Precipitation␣
 ↪distribution over time')
img.show()
```

```
Answer 9: The Precipitation distribution over time
--------------------------------------------------------------
```

Insights: From the graphs, it is observed that the PM and the cummulative precipitation are inversely proportional is that when one is at its high the other is at its low during the same time.

Question 10:

[16]:
```python
# Importing the necessary libraries
from dash import Dash, html, dcc, callback, Output, Input
import plotly.express as px

# Initializing my app name
df_app = Dash(__name__)

# structuring the layout on how the dashboard should look like
df_app.layout = html.Div([
   html.H1(children='Temperature across various cities over the years',␣
 ↪style={'textAlign': 'center', 'fontSize': '24px','color': 'blue'}),
   dcc.Dropdown(
       options=[{'label': city, 'value': city} for city in␣
 ↪Updated_Cities['city'].unique()],
       value='Beijing',
       id='dropdown-box'
   ),
   dcc.Graph(id='graph-chart')
])

# defining the i/p and o/p of the dash
@df_app.callback(
    Output('graph-chart', 'figure'),
    Input('dropdown-box', 'value')
)
# creating a specific function to update the data
def update_graph(value):
    df_dashboard = Updated_Cities[Updated_Cities['city'] == value]
    return px.line(df_dashboard, x='date', y='TEMP')

print("Answer 10: Dashboard of the temperature across various cities ")
print("----------------------------------------------------------------")

# Executing step
if __name__ == '__main__':
    df_app.run_server(debug=True, use_reloader=False , port=8051)
```

```
Answer 10: Dashboard of the temperature across various cities
----------------------------------------------------------------
```

```
<IPython.lib.display.IFrame at 0x1696b2050>
```

Insights: As discussed above, the temperature across all the cities follow a similar sinusoidal pattern which tend to increase from January to July and then the temperature starts to decrease till the end of the year. This is periodic pattern is achieved each and every year.