Exploratory Data Analysis (EDA) of Weather dataset

Written by:
Pratik Antoni Patekar (1001937948).

Other team members:
Harshini Kandimalla (1001960046)
Pratik Dhanraj Chavan (1001963580)

Table of contents

Introduction:

In this assignment, we are working on the Weather dataset which consists of 4463 weather records which include 17 different parameters such Date, Time, Outside temperature, High temperature, Low temperature and so on.

Glimpse of the dataset columns:
The dataset consists of following columns:

1. Date
2. Time
3. Temp Humidity Index
4. Outside Temperature
5. WindChill
6. Hi Temperature
7. Low Temperature
8. Outside Humidity
9. DewPoint
10. WindSpeed
11. Hi
12. Wind Direction
13. Rain
14. Barometer
15. Inside Temperature
16. Inside Humidity
17. ArchivePeriod

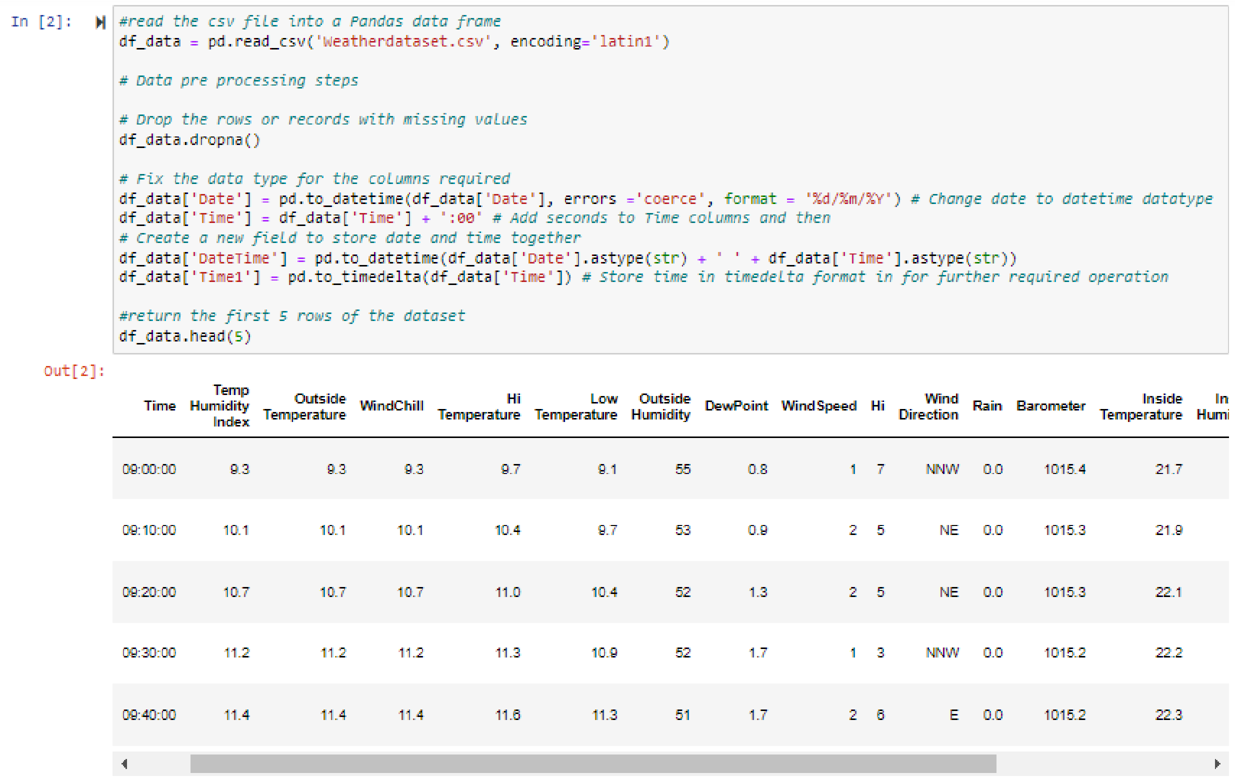
The packages that we used for the completing the assignment are as follows:

1. **Array processing:**
    a. Numpy
2. **Data analysis, wrangling and common exploratory operations:**
    a. Pandas
    b. Itertools
3. **Visualization:**
    a. Matplotlib
    b. Seaborn

Reading and pre-processing the dataset:
The dataset is stored in the ‘Weatherdataset.csv’ file. After reading the csv file, we checked for missing values and dropped if any were present. Luckily, there are no missing values found in the Weather dataset file.

The next pre-processing included setting the data format right for required columns like Date and Time (as they are required for performing aggregation in further tasks). The Date field is stored in ‘datetime’ format and the Time field is stored as a string itself after adding “:00”. We also added two new columns ‘DateTime’ and ‘Time’ as they will make future aggregation and visualization tasks easy to code.

The sample head of the dataset is then displayed using *df_data.head(5)* command.



```
In [2]:
#read the csv file into a Pandas data frame
df_data = pd.read_csv('Weatherdataset.csv', encoding='latin1')

# Data pre processing steps

# Drop the rows or records with missing values
df_data.dropna()

# Fix the data type for the columns required
df_data['Date'] = pd.to_datetime(df_data['Date'], errors ='coerce', format = '%d/%m/%Y') # Change date to datetime datatype
df_data['Time'] = df_data['Time'] + ':00' # Add seconds to Time columns and then
# Create a new field to store date and time together
df_data['DateTime'] = pd.to_datetime(df_data['Date'].astype(str) + ' ' + df_data['Time'].astype(str))
df_data['Time1'] = pd.to_timedelta(df_data['Time']) # Store time in timedelta format in for further required operation

#return the first 5 rows of the dataset
df_data.head(5)
```

Out[2]:

Time	Temp Humidity Index	Outside Temperature	WindChill	Hi Temperature	Low Temperature	Outside Humidity	DewPoint	WindSpeed	Hi	Wind Direction	Rain	Barometer	Inside Temperature	In Humi
09:00:00	9.3	9.3	9.3	9.7	9.1	55	0.8	1	7	NNW	0.0	1015.4	21.7	
09:10:00	10.1	10.1	10.1	10.4	9.7	53	0.9	2	5	NE	0.0	1015.3	21.9	
09:20:00	10.7	10.7	10.7	11.0	10.4	52	1.3	2	5	NE	0.0	1015.3	22.1	
09:30:00	11.2	11.2	11.2	11.3	10.9	52	1.7	1	3	NNW	0.0	1015.2	22.2	
09:40:00	11.4	11.4	11.4	11.6	11.3	51	1.7	2	6	E	0.0	1015.2	22.3	

Task 1: Statistical Exploratory Data Analysis (EDA):
In this part we explored the dataset a bit further in depth. Other than just understanding the number of columns and rows, we understood what are the data types of each column, re-check if any column contains null values, what are the unique values of different fields and how can they be found, and so on.

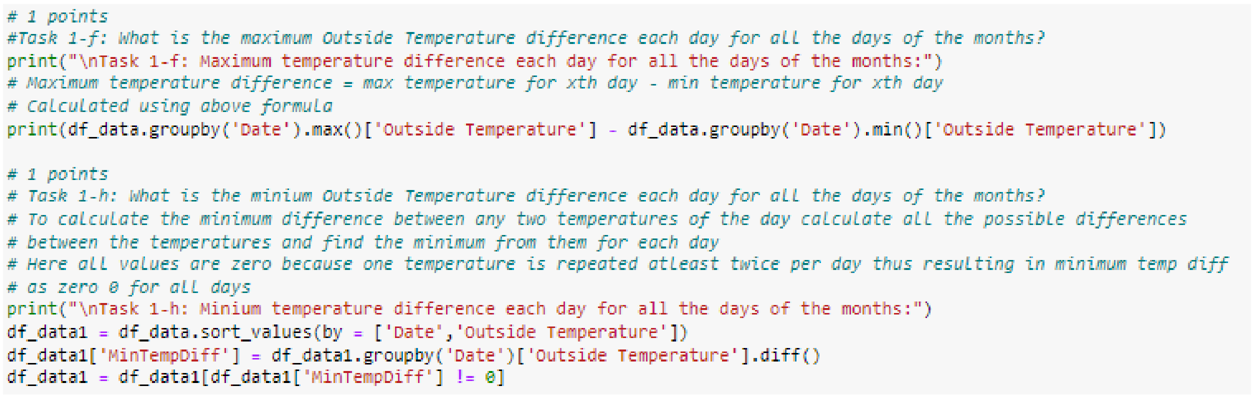
Task 1-f asked to find the maximum Outside Temperature difference each day for all the days of the months which can be easily calculated using the following formula:

$$Maximum\ temperature\ difference = max\ temperature\ for\ x^{th}\ day - min\ temperature\ for\ x^{th}\ day$$

Task 1-h was an interesting task as we had to find the minimum Outside Temperature difference each day for all the days of the month which needs a better understanding.
For example, to find the minimum difference between values 10, 11, 11.5, 13, 15.
Most people tend to say that the answer is $15 - 13 = 2$ which is not correct. The actual answer must be $11.5 - 11 = 0.5$.

The approach we used is to arrange the numbers in descending order i.e. $15, 13, 11.5, 11, 10$. And then use $.diff()$ function in pandas to calculate the consecutive differences i.e. $(Nan - 15),\ (15 - 13),\ (13 - 11.5),\ (11.5 - 11),\ (11 - 10) = Nan,\ 3,\ 1.5,\ 0.5,\ 1$. And now we find the minimum from these values which is nothing but 0.5. This same approach is applied to the dataframe in task 1-f.

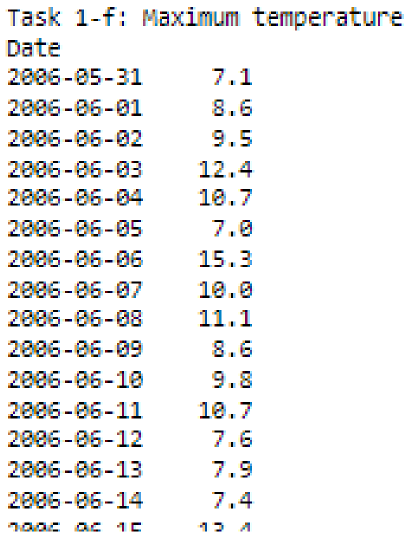
Code:



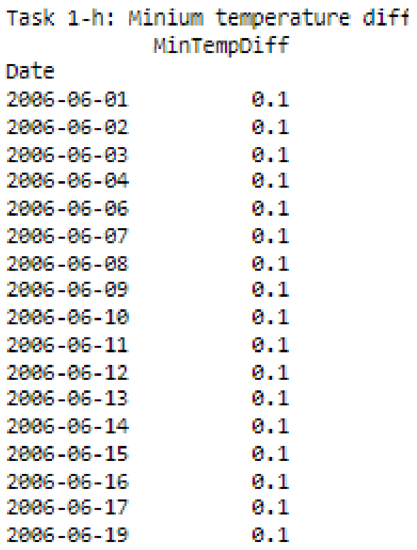
```
# 1 points
#Task 1-f: What is the maximum Outside Temperature difference each day for all the days of the months?
print("\nTask 1-f: Maximum temperature difference each day for all the days of the months:")
# Maximum temperature difference = max temperature for xth day - min temperature for xth day
# Calculated using above formula
print(df_data.groupby('Date').max()['Outside Temperature'] - df_data.groupby('Date').min()['Outside Temperature'])

# 1 points
# Task 1-h: What is the minium Outside Temperature difference each day for all the days of the months?
# To calculate the minimum difference between any two temperatures of the day calculate all the possible differences
# between the temperatures and find the minimum from them for each day
# Here all values are zero because one temperature is repeated atleast twice per day thus resulting in minimum temp diff
# as zero 0 for all days
print("\nTask 1-h: Minium temperature difference each day for all the days of the months:")
df_data1 = df_data.sort_values(by = ['Date','Outside Temperature'])
df_data1['MinTempDiff'] = df_data1.groupby('Date')['Outside Temperature'].diff()
df_data1 = df_data1[df_data1['MinTempDiff'] != 0]
```

Output:



```
Task 1-f: Maximum temperature
Date
2006-05-31     7.1
2006-06-01     8.6
2006-06-02     9.5
2006-06-03    12.4
2006-06-04    10.7
2006-06-05     7.0
2006-06-06    15.3
2006-06-07    10.0
2006-06-08    11.1
2006-06-09     8.6
2006-06-10     9.8
2006-06-11    10.7
2006-06-12     7.6
2006-06-13     7.9
2006-06-14     7.4
```



```
Task 1-h: Minium temperature diff
                MinTempDiff
Date
2006-06-01             0.1
2006-06-02             0.1
2006-06-03             0.1
2006-06-04             0.1
2006-06-06             0.1
2006-06-07             0.1
2006-06-08             0.1
2006-06-09             0.1
2006-06-10             0.1
2006-06-11             0.1
2006-06-12             0.1
2006-06-13             0.1
2006-06-14             0.1
2006-06-15             0.1
2006-06-16             0.1
2006-06-17             0.1
2006-06-19             0.1
```

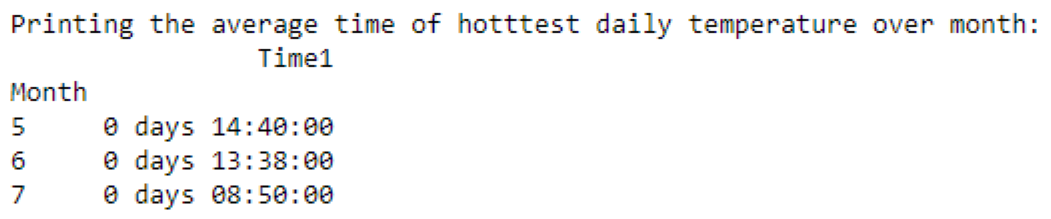
Task 2: Aggregation & Filtering & Rank

In this task, we were asked to perform some high level aggregation and filtering operations using built-in functions of the pandas package.

Task 2-A: Here we had to perform aggregation tasks using ‘Outside Temperature’ column.

Question a: *What is the average time of hottest daily temperature (over month)?*

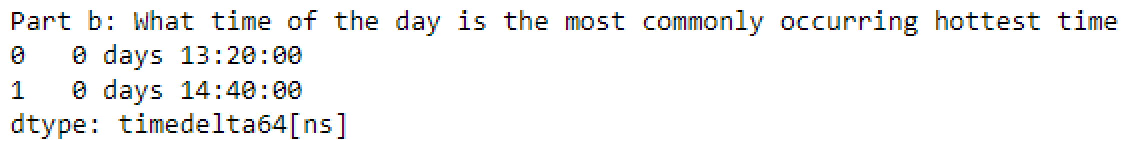
1. We first take a small subset of the dataset (named as df_data1) which include the columns which are required to perform the task.
2. We then filtered the values of df_data1 as per Date where the Outside Temperature is maximum. For this we used the ".transform*(‘max’)*“ function.
3. Then we use groupby() over Date to find the average time where the temperature is maximum for every day.
4. As in the question it is asked to find the average time over month, we calculate the same by using the groupby() again over Month. The Month can be found easily using the “pd.DatetimeIndex(out1['Date']).month” function.



```
Printing the average time of hotttest daily temperature over month:
                Time1
Month
5     0 days 14:40:00
6     0 days 13:38:00
7     0 days 08:50:00
```

Question b: *What time of the day is the most commonly occurring hottest time?*

This task basically requires us to find the hottest times of the day which have the highest frequency. This can be easily done using the “.mode()” function. The mode function returns the values from the column that have the maximum frequency.



```
Part b: What time of the day is the most commonly occurring hottest time
0   0 days 13:20:00
1   0 days 14:40:00
dtype: timedelta64[ns]
```

Question c: *Which are the Top Ten hottest times on distinct days, preferably sorted by date order?*

In task 2-A question (a) above, we already found the daily times with Hottest temperatures.

1. So in this task we used the same dataframe and sorted the data in descending order of hottest temperature so that we get 10 hottest temperatures with times at the top.
2. After that we extract the top 10 rows which are the required data and just sort them again in ascending order of the Date.

```
Part c: Which are the Top Ten hottest times on distinct days, preferably sorted by date order
         Date  Outside Temperature           Time1
3   2006-06-03                  19.6 0 days 14:55:00
6   2006-06-06                  23.2 0 days 14:20:00
7   2006-06-07                  19.6 0 days 13:00:00
8   2006-06-08                  20.7 0 days 16:35:00
10  2006-06-10                  19.9 0 days 13:05:00
11  2006-06-11                  22.4 0 days 11:10:00
12  2006-06-12                  19.4 0 days 15:00:00
15  2006-06-15                  21.1 0 days 13:55:00
17  2006-06-17                  18.9 0 days 14:50:00
28  2006-06-28                  21.4 0 days 10:45:00
```

After the data is printed, this data is written into the “Outside_Temperature.txt” file as asked in the question.

Task 2-B: Using the ‘Hi Temperature’ values produce a “Hi_Temperature.txt” file containing all of the Dates and Times where the “Hi Temperature” was within +/- 1 degree of 22.3 or the “Low Temperature” was within +/- 0.2 degree higher or lower of 10.3 over the first 9 days of June.

For this task we used the technique called masking or filtering the dataset. The question can be broken down into 8 conditions that need to be applied to the dataframe. Those conditions are as follows:

1. Hi Temperature >= 21.3
2. Hi Temperature <= 23.3
3. Low Temperature >= 10.1
4. Low Temperature <= 10.5
5. Date >= 1
6. Date <= 9
7. Month == 6

This exact same conditions are applied in the code to create different masks and then these masks are then applied to the dataframe to filter the required data.

Code:

```
df_data2 = df_data[['Date', 'Time', 'Hi Temperature', 'Low Temperature']]  # get the required fields
# Add month and date fields for filtering
df_data2['Month'] = pd.DatetimeIndex(df_data2['Date']).month
df_data2['Date1'] = pd.DatetimeIndex(df_data2['Date']).day

# Create filters for filtering as per given conditions
mask1 = df_data2['Hi Temperature'] >= 21.3
mask2 = df_data2['Hi Temperature'] <= 23.3
mask3 = df_data2['Low Temperature'] >= 10.1
mask4 = df_data2['Low Temperature'] <= 10.5
mask5 = df_data2['Date1'] >= 1
mask6 = df_data2['Date1'] <= 9
mask7 = df_data2['Month'] == 6

# Apply the filters to dataframe and drop the columns that were added for creating masks/ filters
out2 = df_data2[(mask1 & mask2) | (mask3 & mask4 & mask5 & mask6 & mask7)]
out2 = out2.drop('Date1', axis = 1)
out2 = out2.drop('Month', axis = 1)
```

The output of the above code is then written into the file named “output.txt” as well.

Task 3: Visualization
In this task we were asked to perform different visualization tasks in order to get a better understanding of the data.

Task 3-A: *Visualize the temperature for each month. Think of a way to nicely visualize all the temperatures and provide a detailed explanation.*

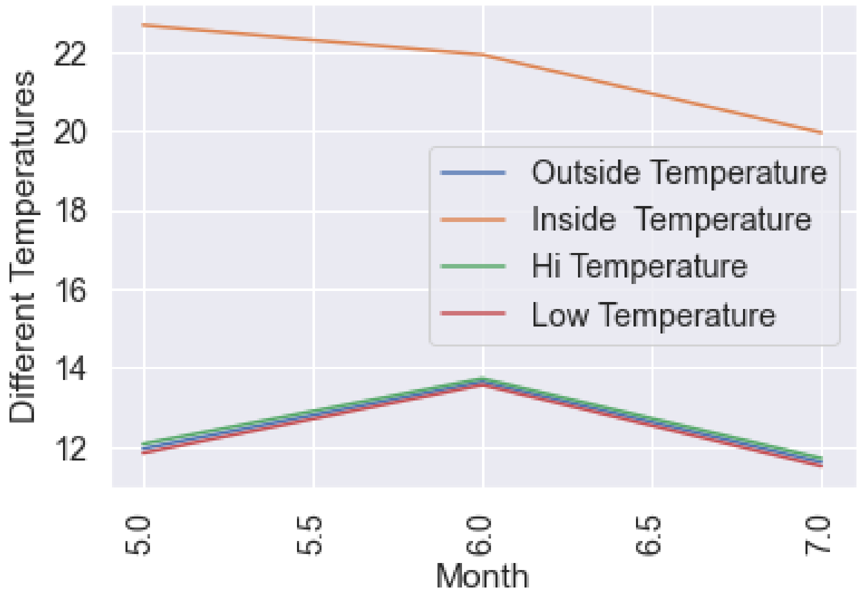
Here we are asked to plot the different temperatures over every month. The different temperatures that are present in the dataset are:

1. Outside temperature
2. Inside temperature
3. High temperature
4. Low temperature

The steps that we followed to get the required plot are as follows:

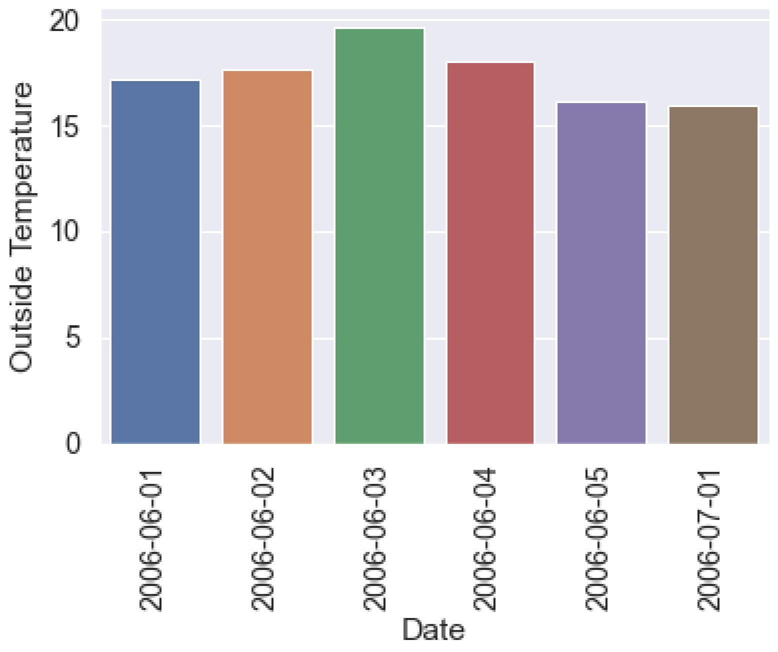
1. Get the required fields in a new dataframe
2. Add month to the new dataframe
3. Calculate the average of all temperatures (individually) over every month
4. Plot these averaged temperatures against the Month

Output:



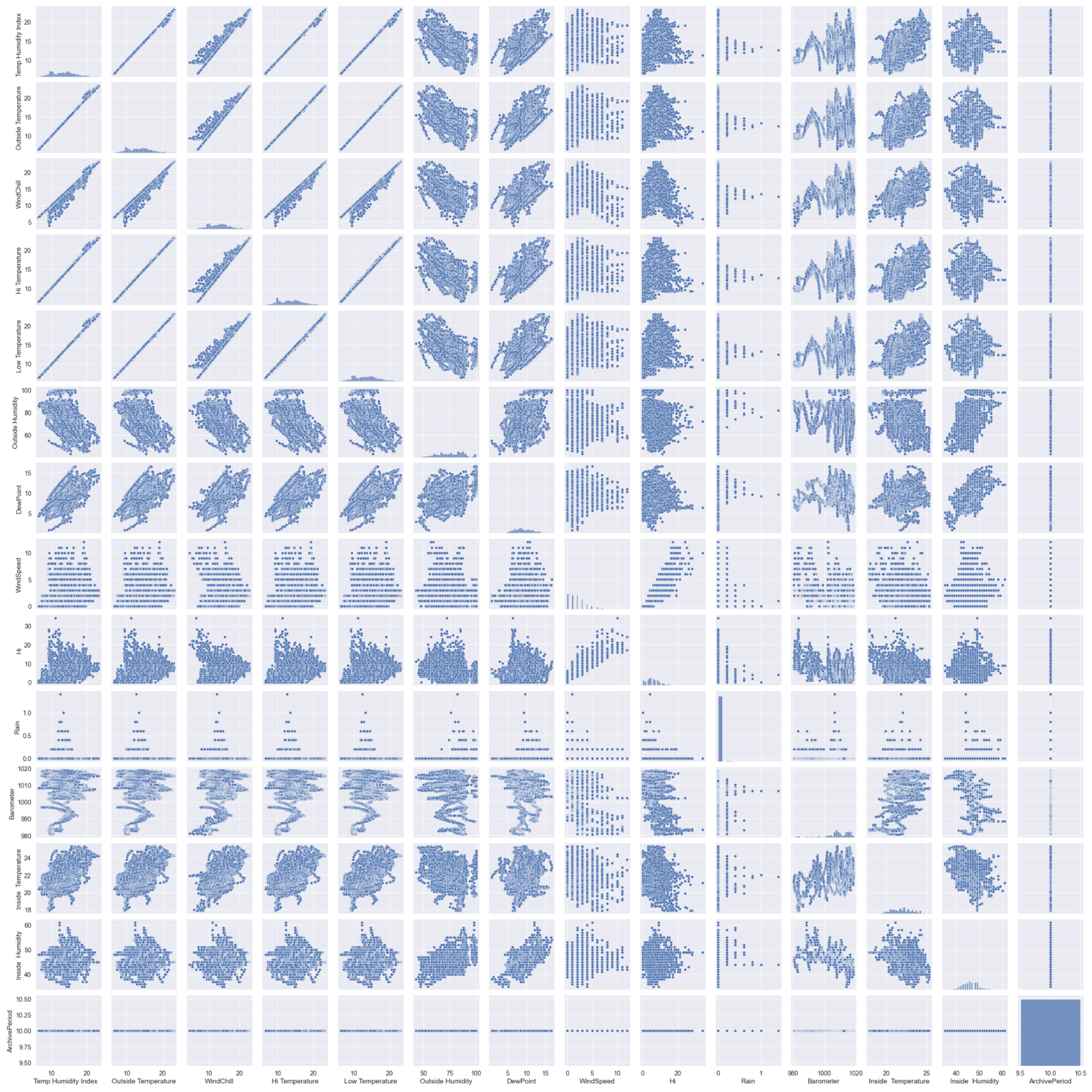
Task 3-B: *Display the time period on a bar plot which has the highest temperature for the first 5 days of every month and provide detailed explanation.*

The dataset includes the data for the last day of May i.e. 31st May 2006, all days of June and the first day of July i.e. 1st July 2006. In question we are asked to plot the highest temperatures for the first 5 days of every month. Thus we can say that the data is required only for the following dates: 06/01/2006, 06/02/2006, 06/03/2006, 06/04/2006, 06/05/2006 and 07/01/2006. For this task as well we used masking technique to filter the records with above mentioned dates. And then performed aggregation to get the max value for each day.
On using the barplot of the seaborn library we got the following plot.



Task 4: Finding interesting information
This task required us to find some interesting information i.e. relations from the dataset. The best way to find relations between the fields/ columns of the dataset is to use a pairplot or a heatmap plot of the seaborn library.
So we plotted the pairplot and heatmap for every field present in the dataset. Following are the plots and the observational conclusions.

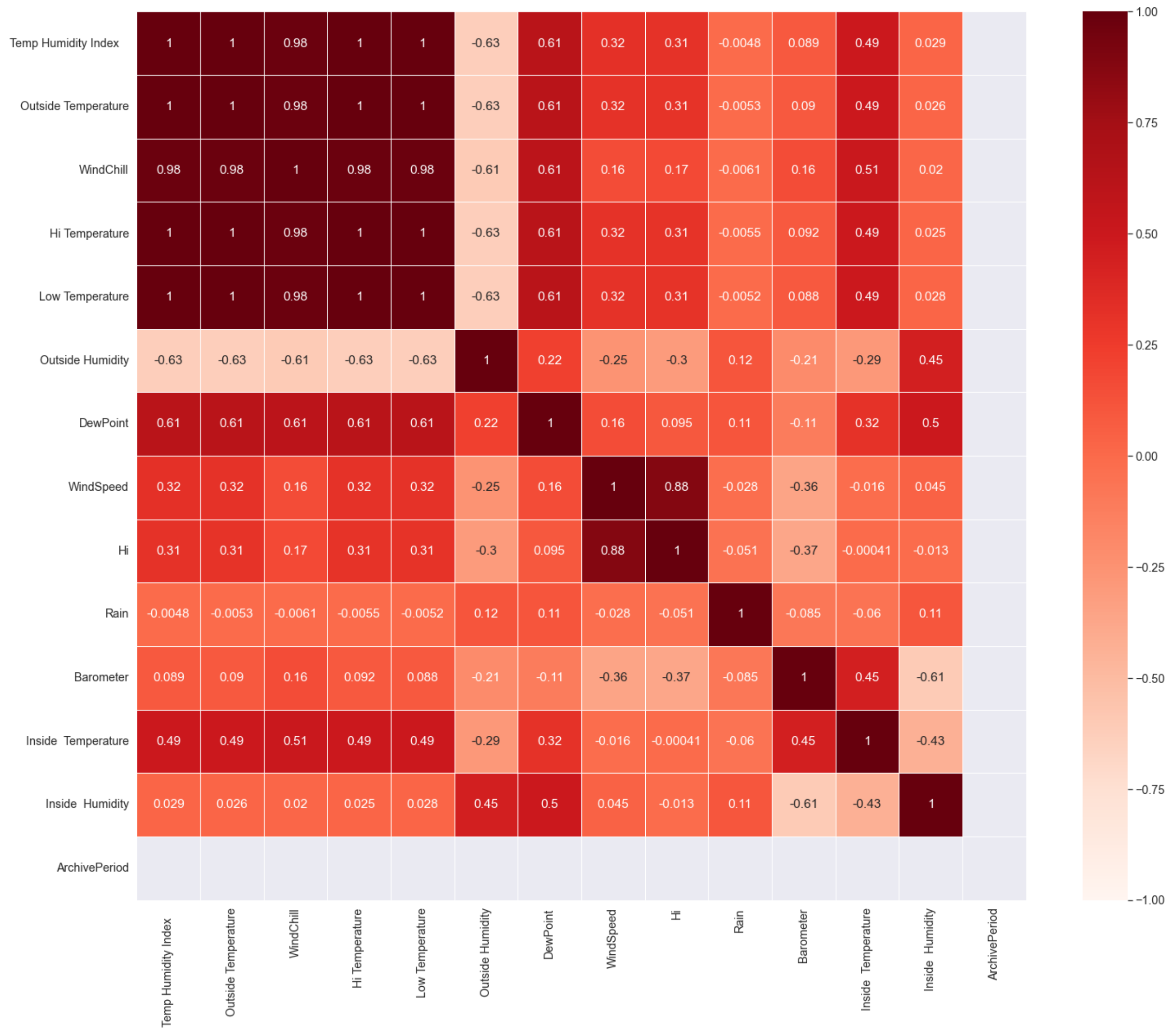
Pairplot:



In the plots below, we can see that the columns of Temp. humidity index, Outside Temperature, Windchill, High Temperature, Low Temperature all share a plot with a positive slope. From this we can understand that these fields are highly correlated to each other. Other than that we can even see that the columns WindSpeed and Hi are highly (positive) correlated to each other as well.

One thing to note over here about Outside Humidity is that, we can see it is somewhat negatively correlated to the Outside Temperature field. And similar deductions can be made for the other fields as well.

Heatmap:

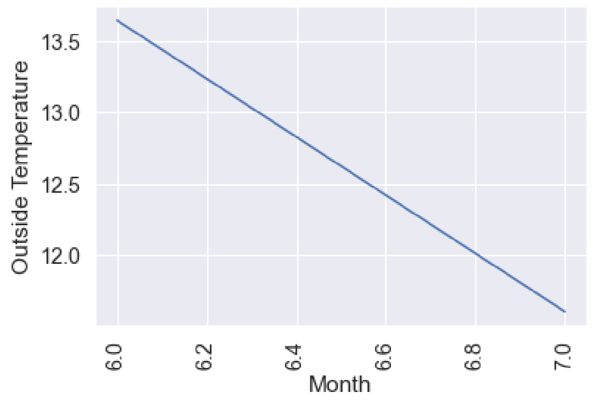


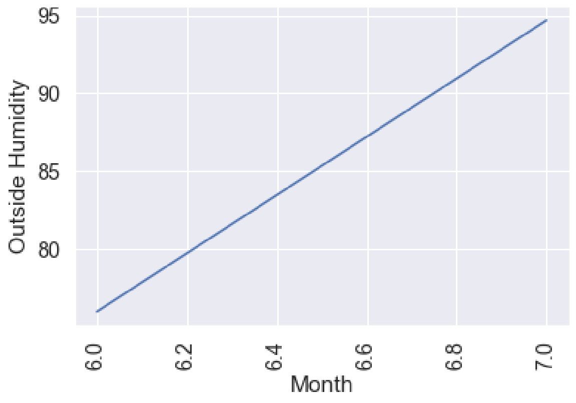
The same relationships can be deduced from the heatmap plots as well. Heat map shows the values of correlation coefficients calculated for each pair of fields on a scale of 0 to 1. The value -1 of correlation coefficient means the correlation between both columns is very high and negative and value of 1 means the correlation is very high and positive. Whereas the value of 0 means that the fields are not correlated at all.

Observation: The Outside Humidity has a negative correlation coefficient of -0.63 against Outside temperature i.e. they share an inverse relationship with each other.

A further detailed relation between outside humidity and temperature can be shown as follows. From the plots below, we can see that the average Outside temperature decreases over the month of June to July whereas on the other hand, the Outside Humidity increases.

And we know that this is true because if the Outside temperature is high, it causes more evaporation of moisture from air and thus reducing Outside Humidity and if the Outside Temperature is low then Outside Humidity will be high.





References:

1. *Python reference website: https://realpython.com/tutorials/data-science/*
2. *Pandas reference website: https://pandas.pydata.org/*
3. *Professor Neshat Beheshti lecture notes for Python Programming course.*
4. *Python code bug resolution reference: Geeks4geeks - https://www.geeksforgeeks.org/python-programming-language/ and other online source material.*

Observations by other team members:

Python is a very useful and easy to use tool for handling datasets and performing operations such as aggregation, filtering and visualization of the data.
The masking technique to filter the data in pandas dataframe was something new that we discovered.

Plotting graphs using Seaborn and Matplotlib libraries is a very powerful tool. It helps in getting several visual insights of the data such as relations between the fields present in the dataset, finding correlation and understanding the trend of the fields values.