Python Report on Assignment-1

Student Name and ID of the member submitting the assignment:
Sri Archana Taduri - 1001964029

Student Name and ID of the remaining members:
Jasti Kavya Sri - 1002029982
Ritish Alapati - 1001989611

In this assignment we have conducted guided exploration/aggregation/descriptive operations over the given Weather dataset(Weatherdataset.csv).

Overall we have performed 4-Tasks which are explained below in brief.

Required Python Packages

Here we have imported the below libraries :

a. **matplotlib inline** - This is a special Python command to prepare the notebook for matplotlib numpy - for the array processing

b. **For data analysis**, wrangling and common exploratory operations we use the b below libraries
   - pandas
   - from pandas, imported Series & DataFrame
   - from itertools, imported chain

c. **Finally for visualization**, we imported two libraries
   - Matplotlib for basic viz and
   - seaborn for more stylish figures

Reading Dataset

Here we read the “Weatherdataset.csv” file into the dataframe “df_data” and have displayed first 5 rows of the dataset using the method head().

```
In [114]: #read the csv file into a Pandas data frame
          df_data = pd.read_csv('Weatherdataset.csv', encoding='latin1',parse_dates=["Date"],dayfirst=True)

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

Out[114]:

	Date	Time	Temp Humidity Index	Outside Temperature	WindChill	Hi Temperature	Low Temperature	Outside Humidity	DewPoint	WindSpeed	Hi	Wind Direction	Rain	Barometer	Inside Temperature
0	2006-05-31	09:00	9.3	9.3	9.3	9.7	9.1	55	0.8	1	7	NNW	0.0	1015.4	21.7
1	2006-05-31	09:10	10.1	10.1	10.1	10.4	9.7	53	0.9	2	5	NE	0.0	1015.3	21.9
2	2006-05-31	09:20	10.7	10.7	10.7	11.0	10.4	52	1.3	2	5	NE	0.0	1015.3	22.1
3	2006-05-31	09:30	11.2	11.2	11.2	11.3	10.9	52	1.7	1	3	NNW	0.0	1015.2	22.2
4	2006-05-31	09:40	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

Here we have used some pandas in-built functions to explore the Data provided in the csv file.

Task-1a : we used the **info()** method to get the technical details of the csv file like rows, columns, non-null count, data type and RAM usage etc,. as well.

```
Task 1-a: Details of df_data data frame are:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4463 entries, 0 to 4462
Data columns (total 17 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Date                   4463 non-null   datetime64[ns]
 1   Time                   4463 non-null   object
 2   Temp Humidity Index    4463 non-null   float64
 3   Outside Temperature    4463 non-null   float64
 4   WindChill              4463 non-null   float64
 5   Hi Temperature         4463 non-null   float64
 6   Low Temperature        4463 non-null   float64
 7   Outside Humidity       4463 non-null   int64
 8   DewPoint               4463 non-null   float64
 9   WindSpeed              4463 non-null   int64
 10  Hi                     4463 non-null   int64
 11  Wind Direction         4463 non-null   object
 12  Rain                   4463 non-null   float64
 13  Barometer              4463 non-null   float64
 14  Inside  Temperature    4463 non-null   float64
 15  Inside  Humidity       4463 non-null   int64
 16  ArchivePeriod          4463 non-null   int64
dtypes: datetime64[ns](1), float64(9), int64(5), object(2)
memory usage: 592.9+ KB
Task 1-b: Number of rows:4463 and number of columns:17
```

Task 1-b : Used the **len()** method to find the number of rows and columns. We can also use the shape() method to get the same output.
Task 1-c : Used the **describe()** method to get the descriptive details like dispersion, central tendency and shape of the dataset distribution excluding the Nan.
Task 1-d : Used the **mean()** function to find the average temperature of all the days.
Task 1-e : Used the **groupby()** method along with **grouper** to group all the days in a week which helped in taking the average temperature of the whole week.

```
Task 1-b: Number of rows:4463 and number of columns:17
Task 1-c: Descriptive details of 'Wind Direction' column are

count     4463
unique      17
top        SSW
freq      1276
Name: Wind Direction, dtype: object
Task 1-d: The average temp for each day:

Date
2006-05-31    11.950000
2006-06-01    12.520139
2006-06-02    12.960417
2006-06-03    12.840278
2006-06-04    12.013889
2006-06-05    12.186806
```

```
2006-06-30    14.709722
2006-07-01    11.605556
Name: Outside Temperature, dtype: float64
Task 1-e: The average temp for each week:

Date
2006-06-04    12.498048
2006-06-11    14.464881
2006-06-18    14.549206
2006-06-25    12.189573
2006-07-02    13.934109
Freq: W-SUN, Name: Outside Temperature, dtype: float64
```

Task 1-f : Here we have used the **min()** and **max()** functions to find the minimum and maximum of values of the temperature in the column and have subtracted the values to find the maximum temperature difference in a day.

Task 1-h : Here we have used the **min()** to the minimum temperature and functions, **nsmallest()** - which gives the list of 'n' smallest values in a Series, **drop_duplicates()** - which is used to drop the duplicate values in a series along with the **groupby()** method to find the second minimum value in a Series.

Subtracting the minimum value and second minimum value in a Series will give us the minimum temperature difference of that day.

Task 1-i : Here we have made use of the **unique()** function to find the unique values in each column of a given dataset. We wrote a for loop to iterate through the dataset Series number of times.

```
Freq: W-SUN, Name: Outside Temperature, dtype: float64
Task 1-f: Maximum temperature difference each day for all the days of the months:

Date
2006-05-31     7.1
2006-06-01     8.6
2006-06-02     9.5
2006-06-03    12.4
```

```
Task 1-h: Minimum temperature difference each day for all the days of the months:
         Date  Outside Temperature
0  2006-05-31                  0.1
1  2006-06-01                  0.2
2  2006-06-02                  0.1
3  2006-06-03                  0.1
4  2006-06-06                  0.1
```

```
Task 1-i: The unique values for each column are:


Unqie values in col  Date :
 ['2006-05-31T00:00:00.000000000' '2006-06-01T00:00:00.000000000'
 '2006-06-02T00:00:00.000000000' '2006-06-03T00:00:00.000000000'
 '2006-06-04T00:00:00.000000000' '2006-06-05T00:00:00.000000000'
 '2006-06-06T00:00:00.000000000' '2006-06-07T00:00:00.000000000'
```

Task 2: Aggregation & Filtering & Rank

Task 2-A - We have generated an output file named “Outside_Temperature.txt” which has

a. The average time of hottest daily temperature (over month)
b. Most commonly occurring hottest time of the day
c. Top Ten hottest times on distinct days, sorted by date order

We used two data frames grouped by “Date” field and are taking the maximum of the day temperature. These frames are later merged together to get the desired output.
We made use of the functions normalize(),value_counts(),sort_values(),drop_duplicates() to find out the mean of the hottest temperature on average and also on a daily basis.

And finally we used the file handling functions in Pandas like f.write() and f.close to write and close the file.

Task 2-B - We have generated an output file named “Hi Temperature.txt” which has all 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.

As mentioned we have taken two variables to store the start and end date of June and then compared the created data frame dates which are falling in the first nine days of June.

If the days are in between 1st and 9th June, the we are checking if the Hi Temperature value is in between 21.3 and 23.3(i.e., +/-1 degree of 22.3) and if the Low Temperature value is in between 10.1 and 10.5(i.e., +/- 0.2 degree higher or lower of 10.3).

If yes, then we are writing to the output file Hi Temperature.txt.

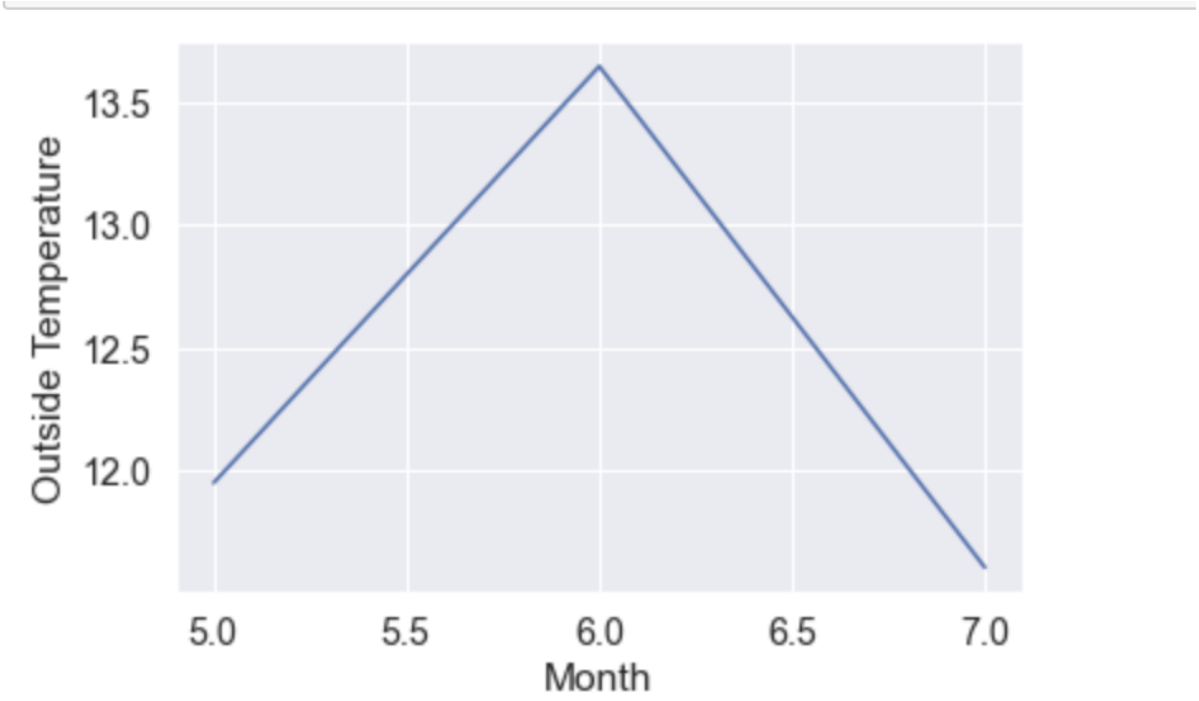
Task 3: Visualization

For Visualization in Task-3, we have used the seaborn to plot the data.

Task 3-a: Visualize the temperature for each month

For doing this, we have created two new data frames, one for Month and one for Outside Temperature which we have merged to form a new dataframe using the frames function.
We did it using the inner join method of pandas and then took the average of the same using mean() function.

Finally we plotted the line plot of the average temperature of each month using sns.lineplot() function from Seaborn.



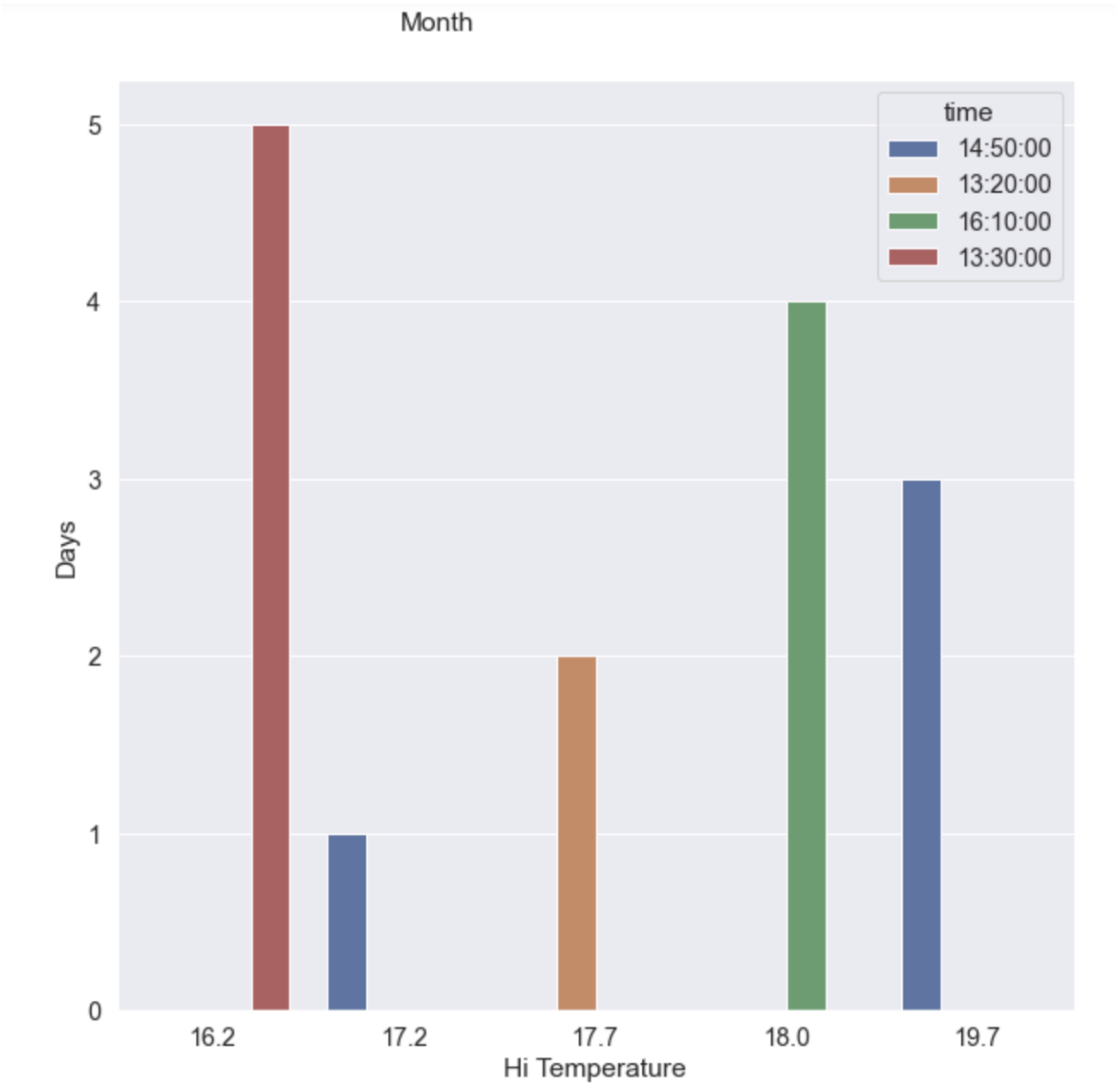
Task 3-b :
Display the time period on a bar plot which has highest temperature for the first 5 days of every month

Firstly, we took the days from the “Date” column using the .dt.day attribute and time from the “Time” column from the dataset using to_datetime and dt.time attributes. This is done so as to check/compare the day at which the highest temperature was recorded and also note the time period at the same time.

After that we checked if a day is falling in between the first five days of the month by using the loc[] function which is used to get the index of the value and stored in a dataframe.

Grouped by “Date” field, we took the max temperature of the day and merged this data frame with the dataframe above using the inner join.

From the merged data frames we have dropped the duplicate values so as to take only unique values and plot the Highest temperature values in a Month.



Task 4:

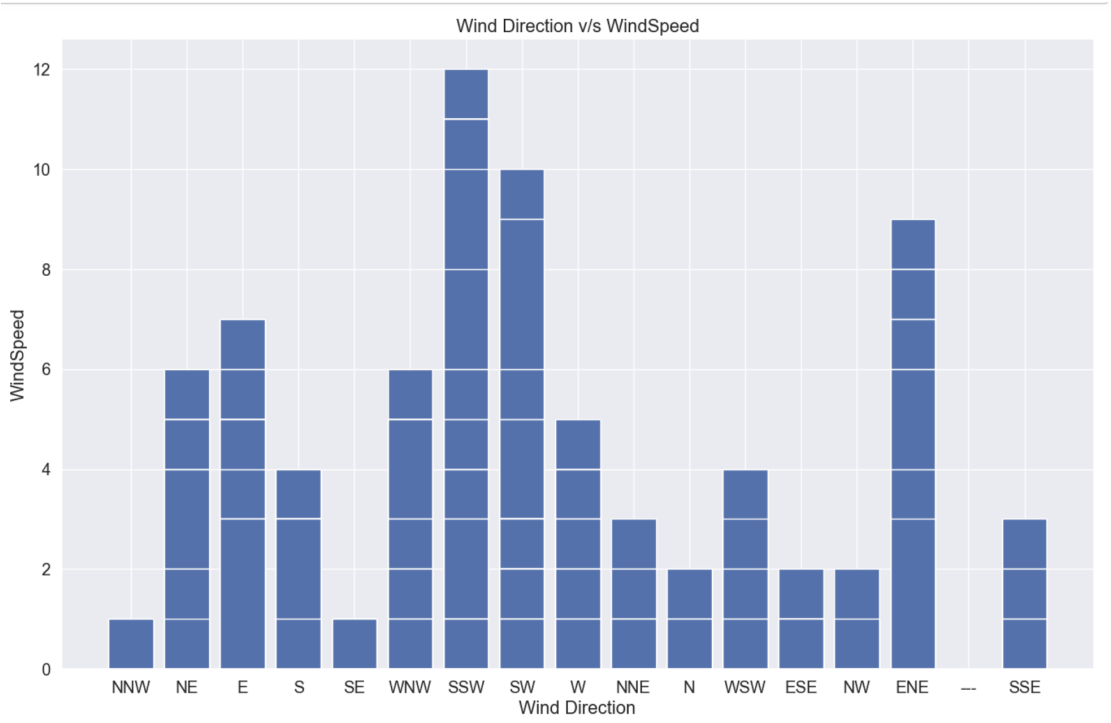
For Visualization in task4, we have used Matplotlib and Seaborn for plotting the data.

#Interesting Information-1: We have interestingly observed below mentioned findings based on two columns namely Wind Direction and WindSpeed. Used the below Matplotlib functions to plot the bar graph.

plt.figure(): used this to create a figure by setting the figsize (width & height) as (16,10) and dots-per-inch(dpi) is fixed to 100 for a better glance.
plt.bar(): By using this function, we are directly comparing the columns 'Wind Direction' and 'WindSpeed' to show categorical data as rectangular bars with the value they represent from the given dataset.
plt.gca(): used this function to set the title, x and y axis labels to be displayed.
plt.show(): this function starts an event loop and checks for active figure and opens the display of the figure designed.

Output:
Based on the below graph,

- SSW direction has the highest wind speed of 12 and NNW & SE has the lowest wind speed of 1.
- It is observed that 12/16 wind directions have the wind speed is below 6.
- Highest wind speed wind directions are towards the southwest which are greater than 10.
- Average windspeed of all the wind directions is 4.8



#Interesting Information-2: By using the wind direction column, we are checking how many times it has occurred throughout the dataset. We used two lists namely wind_direction{} and plot{} as required and also we used the below mentioned Seaborn functions to plot the data.
Firstly, **Sns.set_style()** is used to style the plot with parameter-whitegrid.
Sns.set() is used to fix the font scale to 1
We have written a for loop to get the count of occurrences of wind direction and added key, values to the lists accordingly.

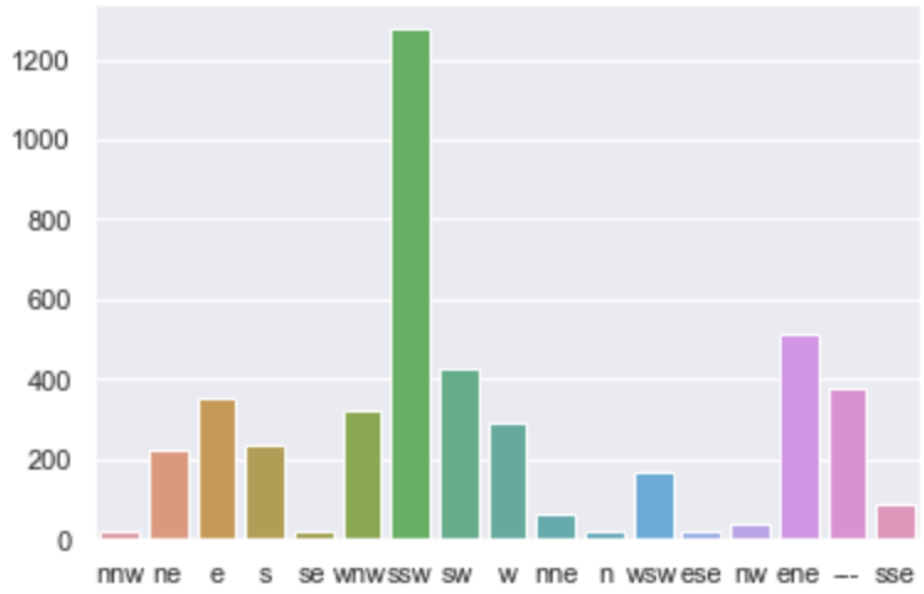
Sns.barplot() is used to aggregate the categorical data according to its mean. Here, we chose the categorical column ‘Wind Direction’ on x -axis and a numerical column ‘count’ on y-axis as the parameters.

Output:
Below graph shows the visualization of Wind Direction throughout the dataset.

- Wind Direction towards SSW is higher and NNW,SE,N,ESE has the lower occurrence.
- 1/16 directions count is less than 450.

```
Visualizing the Wind Direction by barplot

Out[120]: <AxesSubplot:>
```



References:

https://pandas.pydata.org/docs/getting_started/intro_tutorials/03_subset_data.html
https://stackoverflow.com/questions/28417293/sample-datasets-in-pandas
https://www.geeksforgeeks.org/python-pandas-dataframe/
https://www.w3schools.com/python/pandas/pandas_dataframes.asp