CHAPTER - 1

INTRODUCTION

Data analytics is the pursuing an extract meaning from raw data using specialized computer systems. These systems transform, organize, and model the data to draw conclusions and identify patterns. A term Analytics is the systematic computational analysis of data or statistics. It is used for the discovery, interpretation, and communication of meaningful patterns in data. It also logically applying data patterns towards effective decision-making. It can be valuable in areas rich with recorded information; analytics relies on the simultaneous application of statistics, computer programming and operations research to quantify performance.

1.1Project Overview

AS a name says the work Live weather forecast analysis will do the work of analysing the current weather data. It not only fetches the current data it also gives us a future weather report of upcoming 4 days. In this work we can manipulate the area, place, time and get the data according to that location and time. This will locate the area by Latitude and Longitude values. Here we have used many python libraries which help us to extract information from raw data. Here we used Tomorrow.io to get a live weather report.

Weather forecast analysis is statistical analysis. working on this dataset weather can be explored and can be plotted on graphs like bar plot, pie chart and heat map allows the user to get information and helps to understand in easy way

1.2Data Collection

The dataset used in this project came from Tomorrow.io Weather API v4, one of the most reliable free-to-use weather API. The latest version comes with all-in-one endpoint, you can easily retrieve all the data that you need by making a single call to the endpoint. Total we get 98 hours of weather data from current date and time to next 4 days.

How we collect the data?

Tomorrow.io API offer a wide selection of weather data feeds sourced by our own proprietary models as well as from many external and public data vendors. Access to the Tomorrow.io API requires a valid access key with the right permissions, allowing it to be used to make requests to specific endpoints.

To get access key sign up for a new Tomorrow.io account. Free plan allows up to:

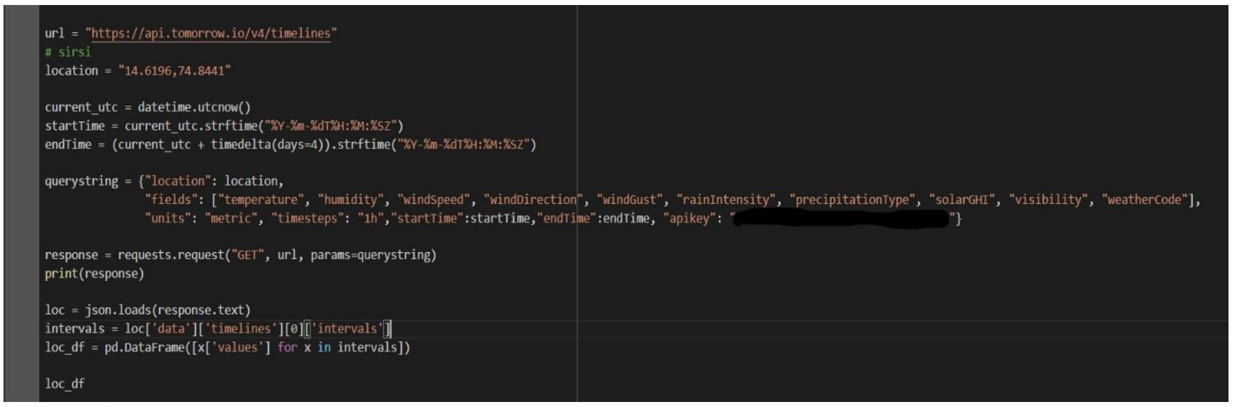
- 500Calls / per day
- 25Calls / per hour
- 2Calls / per second

Once account created, click on the Development menu. A unique Secret Key will be provided there which will be used for calling the weather API.

The API accepts the following input parameters:

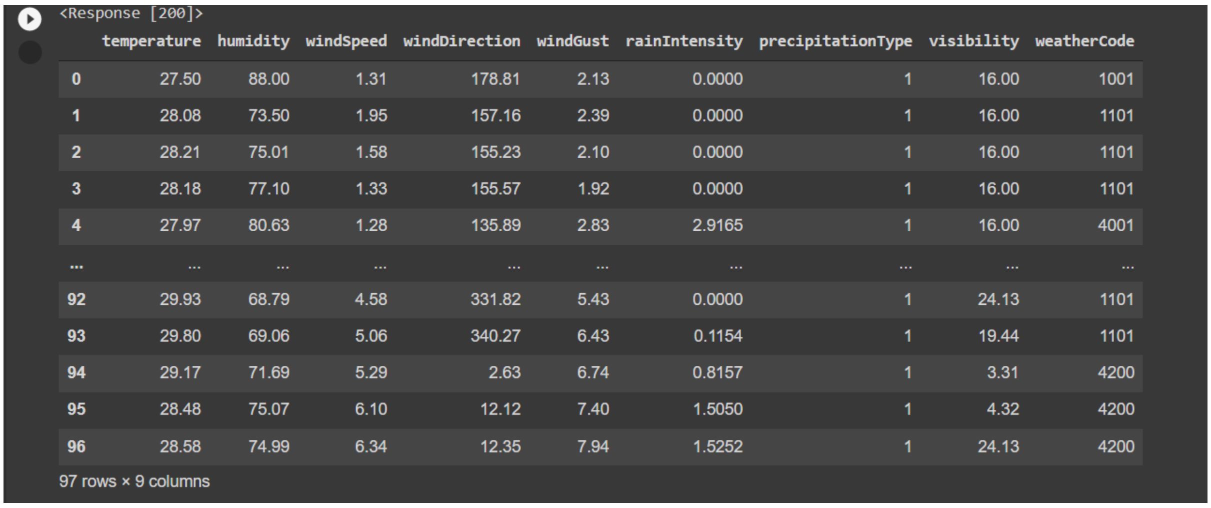
- location - ID of a pre-defined location or latitude longitude string.
- fields - A list of fields representing the data layer.
- units - Unit system. Accepts either metric or imperial.
- timestep - Timesteps of the timelines. Default value it 1h. Accepts best, 1m, 5m, 15m, 30m, 1h, 1d or current.
- startTime - Start time in ISO 8601 format. e.g. 2022–01–20T14:09:50Z.
- endTime - End time in ISO 8601 format. e.g. 2022–01–24T14:09:50Z.
- timezone - Time zone of the datetime. Default to UTC

How we Load the Data



```python
url = "https://api.tomorrow.io/v4/timelines"
# sirsi
location = "14.6196,74.8441"

current_utc = datetime.utcnow()
startTime = current_utc.strftime("%Y-%m-%dT%H:%M:%SZ")
endTime = (current_utc + timedelta(days=4)).strftime("%Y-%m-%dT%H:%M:%SZ")

querystring = {"location": location,
               "fields": ["temperature", "humidity", "windSpeed", "windDirection", "windGust", "rainIntensity", "precipitationType", "solarGHI", "visibility", "weatherCode"],
               "units": "metric", "timesteps": "1h","startTime":startTime,"endTime":endTime, "apikey": "[redacted]"}

response = requests.request("GET", url, params=querystring)
print(response)

loc = json.loads(response.text)
intervals = loc['data']['timelines'][0]['intervals']
loc_df = pd.DataFrame([x['values'] for x in intervals])

loc_df
```

Fig. 1.1: Loading the Data



```
<Response [200]>
    temperature  humidity  windSpeed  windDirection  windGust  rainIntensity  precipitationType  visibility  weatherCode
0         27.50     88.00       1.31         178.81      2.13         0.0000                  1       16.00         1001
1         28.08     73.50       1.95         157.16      2.39         0.0000                  1       16.00         1101
2         28.21     75.01       1.58         155.23      2.10         0.0000                  1       16.00         1101
3         28.18     77.10       1.33         155.57      1.92         0.0000                  1       16.00         1101
4         27.97     80.63       1.28         135.89      2.83         2.9165                  1       16.00         4001
...         ...       ...        ...            ...       ...            ...                ...         ...          ...
92        29.93     68.79       4.58         331.82      5.43         0.0000                  1       24.13         1101
93        29.80     69.06       5.06         340.27      6.43         0.1154                  1       19.44         1101
94        29.17     71.69       5.29           2.63      6.74         0.8157                  1        3.31         4200
95        28.48     75.07       6.10          12.12      7.40         1.5050                  1        4.32         4200
96        28.58     74.99       6.34          12.35      7.94         1.5252                  1       24.13         4200
97 rows × 9 columns
```

Fig. 1.2 Data Set

Data Fields	Description	Data types	SI Unit
Temperature	Temperature	Number	°C
humidity	Percent relative humidity from 0 - 100%	Number	%
windspeed	Wind speed	Number	m/s
Wind Direction	Wind direction in polar degrees 0-360 where 0 is North	number or NULL (when there is no wind)	degrees
Wind Gust	Wind gust speed	Number	m/s
Rain Intensity	measure of the amount of rain that falls over time	Number	m/hr
Precipitation type	The type of precipitation	Number	
visibility	Visibility distance	Number	km
Weather Code	A textual field that conveys the weather conditions.	Number	

Table 1.1 Data Fields

CHAPTER 2

LITERATURE SURVEY

Data pre-processing, the initial part of the project is to understand implementation and usage of various python- built modules. The above process helps us to understand why different modules are helpful rather than implementing those functions from scratch by the developer. These various modules provide better code representation and user understandability. The following libraries are used such as pandas, matplotlib, requests, json, pytz, seaborn etc.

2.1. Library/Module Requirements

- Pandas is defined as an open-source library that provides high-performance data manipulation in Python. The name of Pandas is derived from the word Panel Data, which means an Econometrics from Multidimensional data. Pandas came into the picture and enhanced the capabilities of data analysis. It can perform five significant steps required for processing and analysis of data irrespective of the origin of the data, i.e., load, manipulate, prepare, model, and analyse, and we can install it on command prompt by "pip install pandas" command.

- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible and can create publication quality plots which make interactive figures that can zoom, pan, update and customize visual style and layout also export to many file formats, and Embed in JupyterLab and Graphical User Interfaces, we can install it on command prompt by "pip install matplotlib" command.

- Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and colour palettes to make statistical plots more. attractive and aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs, so that we can switch between different visual representations for same variables for better understanding of dataset, we can install it one command prompt by "pip install seaborn" command.

- Requests allows you to send HTTP/1.1 requests extremely easily. There's no need to manually add query strings to your URLs, or to form-encode your POST data. Keep-alive and HTTP connection pooling are 100% automatic
- The full-form of JSON is JavaScript Object Notation. It means that a script (executable) file which is made of text in a programming language, is used to store and transfer the data. Python supports JSON through a built-in package called json. To use this feature, we import the json package in Python script. It's pretty easy to load a JSON object in Python. Python has a built-in package called json, which can be used to work with JSON data. It's done by using the JSON module, which provides us with a lot of methods which among loads() and load() methods are going to help us to read the JSON file.
- Pytz brings the Olson tz database into Python and thus supports almost all time zones. This module serves the date-time conversion functionalities and helps user serving international client's base. It enables time-zone calculations in our Python applications and also allows us to create timezone aware datetime instances. we can install it one command prompt by "pip install pytz" command.

2.2. Hardware & Software Requirements

SI. NO	Components	Specification
01	PC/Laptop	14' Colour display
02	RAM	4GB
03	HDD	500GB
04	Processor	I3, AMD or more

Table 2.1 Hardware-Requirements

Software	Specification
Operating System	Windows-10
Python Editor	Jupiter Notebook /Google Colab

Table, 2.2: Software-Requirements

2.3. Tools/ Languages/Platform

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet. The platform looks like

Jupyter Notebook can be installed by using either of the two ways described below:

- Using Anaconda: Install Python and Jupyter using the Anaconda Distribution, which includes Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.
- Using PIP: Install Jupyter using the PIP package manager used to install and manage software packages/libraries written in Python.

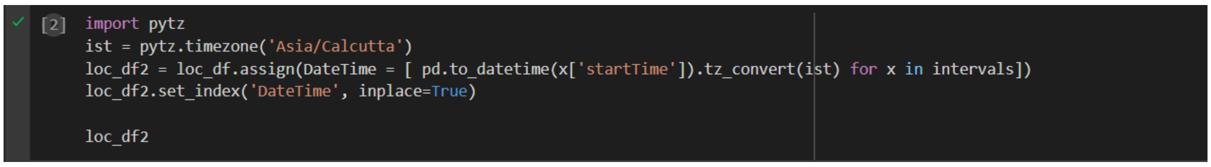
Command Prompt is a command line interpreter application available in most Windows operating systems. It's used to execute entered commands. Most of those commands automate tasks via scripts and batch files, perform advanced administrative functions, and troubleshoot or solve certain kinds of Windows issues.

CHAPTER -3

DATA CLEANING AND WRANGLING MECHANISMS

Data Cleaning or cleansing is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.

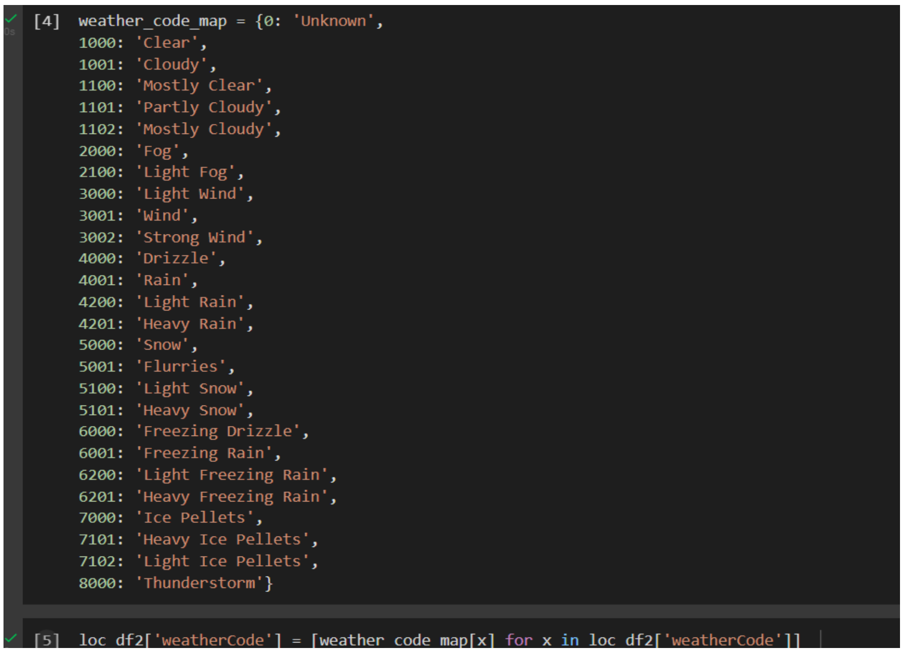
In Dataset total we have 97 rows and 9 columns without any Nan values, we get to know this by NumPy attribute shape which gives (97,9). Hence, we modify this dataset according to requirements for effective prediction and get a result.



```
[2]  import pytz
     ist = pytz.timezone('Asia/Calcutta')
     loc_df2 = loc_df.assign(DateTime = [ pd.to_datetime(x['startTime']).tz_convert(ist) for x in intervals])
     loc_df2.set_index('DateTime', inplace=True)

     loc_df2
```

Fig. 3.1 manipulating and adding datetime

In the above code we use pytz module to date-time conversion functionalities and helps user to understand data easier. Here we are using Pytz to convert UTC time format to Indian standard time format of Calcutta. and we add Date and Time column to Dataset to perform Time series analysis and in the next step Date Time column is made as index column to make analysis easier.

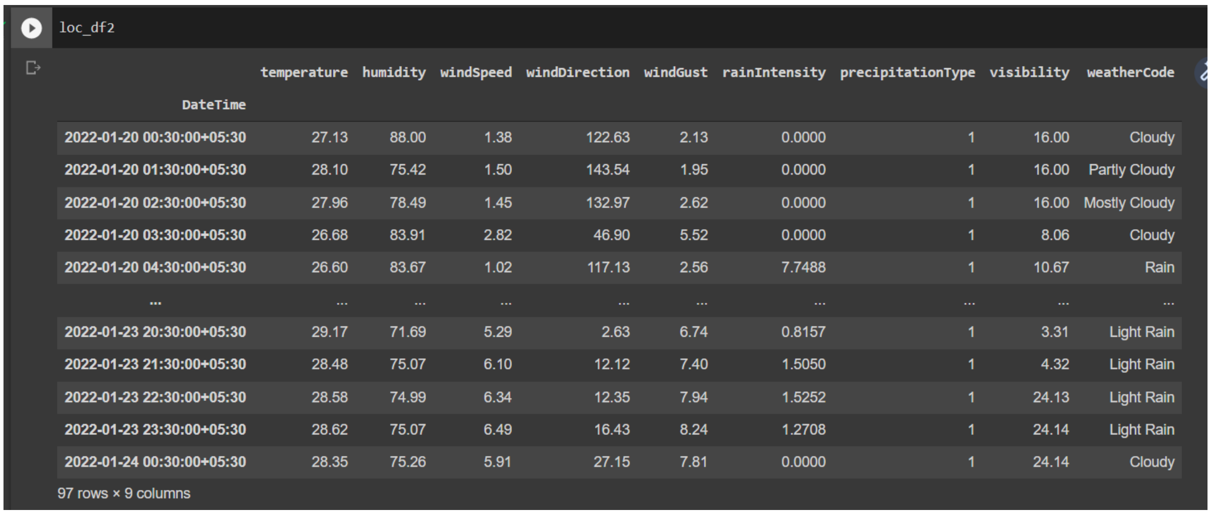


```
[4]  weather_code_map = {0: 'Unknown',
     1000: 'Clear',
     1001: 'Cloudy',
     1100: 'Mostly Clear',
     1101: 'Partly Cloudy',
     1102: 'Mostly Cloudy',
     2000: 'Fog',
     2100: 'Light Fog',
     3000: 'Light Wind',
     3001: 'Wind',
     3002: 'Strong Wind',
     4000: 'Drizzle',
     4001: 'Rain',
     4200: 'Light Rain',
     4201: 'Heavy Rain',
     5000: 'Snow',
     5001: 'Flurries',
     5100: 'Light Snow',
     5101: 'Heavy Snow',
     6000: 'Freezing Drizzle',
     6001: 'Freezing Rain',
     6200: 'Light Freezing Rain',
     6201: 'Heavy Freezing Rain',
     7000: 'Ice Pellets',
     7101: 'Heavy Ice Pellets',
     7102: 'Light Ice Pellets',
     8000: 'Thunderstorm'}

[5]  loc_df2['weatherCode'] = [weather_code_map[x] for x in loc_df2['weatherCode']]
```

Fig. 3.2 Updating Weather Code

In next process of cleaning we update the weather code weather code is passed in numerical code format. Each numerical code of weather Code represents meaning full weather situation to update exact value in dataset first we create python dictionary in which key represents weather code and its value represents actual weather value. Then we map each weather code in dataset and update the dataset.

After all Cleaning Process we get Dataset as in the below table:



```
loc_df2
```

	temperature	humidity	windSpeed	windDirection	windGust	rainIntensity	precipitationType	visibility	weatherCode
DateTime									
2022-01-20 00:30:00+05:30	27.13	88.00	1.38	122.63	2.13	0.0000	1	16.00	Cloudy
2022-01-20 01:30:00+05:30	28.10	75.42	1.50	143.54	1.95	0.0000	1	16.00	Partly Cloudy
2022-01-20 02:30:00+05:30	27.96	78.49	1.45	132.97	2.62	0.0000	1	16.00	Mostly Cloudy
2022-01-20 03:30:00+05:30	26.68	83.91	2.82	46.90	5.52	0.0000	1	8.06	Cloudy
2022-01-20 04:30:00+05:30	26.60	83.67	1.02	117.13	2.56	7.7488	1	10.67	Rain
...	...	...	...	...	...	...	...	...	...
2022-01-23 20:30:00+05:30	29.17	71.69	5.29	2.63	6.74	0.8157	1	3.31	Light Rain
2022-01-23 21:30:00+05:30	28.48	75.07	6.10	12.12	7.40	1.5050	1	4.32	Light Rain
2022-01-23 22:30:00+05:30	28.58	74.99	6.34	12.35	7.94	1.5252	1	24.13	Light Rain
2022-01-23 23:30:00+05:30	28.62	75.07	6.49	16.43	8.24	1.2708	1	24.14	Light Rain
2022-01-24 00:30:00+05:30	28.35	75.26	5.91	27.15	7.81	0.0000	1	24.14	Cloudy

97 rows × 9 columns

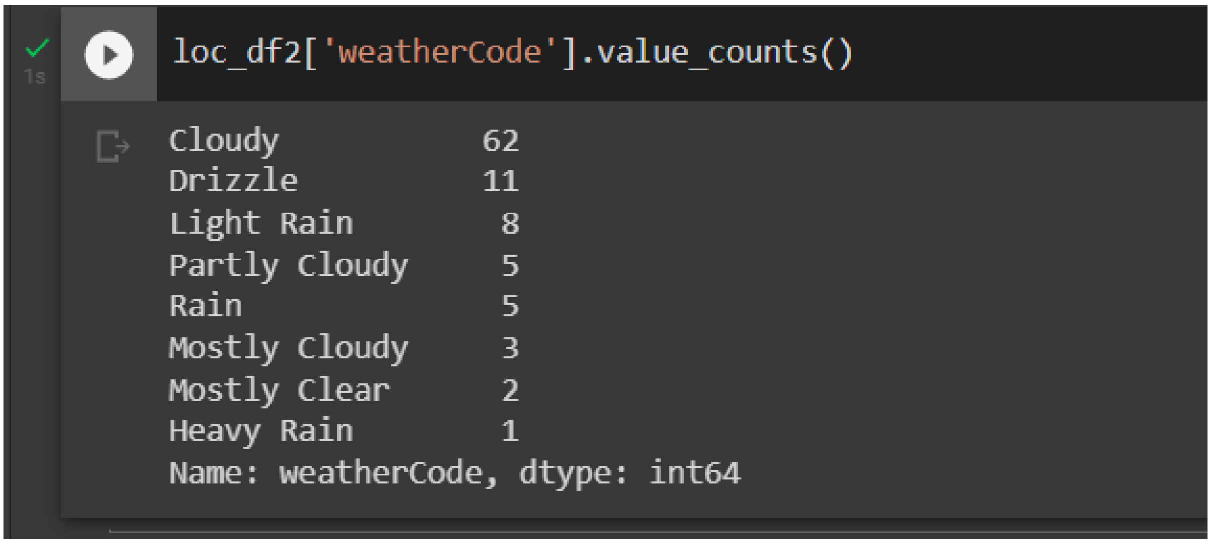
Fig. 3.3 Data Set after Cleaning

CHAPTER 4

DATA ANALYSIS AND VISUALIZATION

Data Analysis is a mechanism that how we analysis the given data in effective way and comparison of given records in Dataset. Some of the scenarios that we performed in our project has given below:

After all the process of data **cleaning we start to analyse the data**

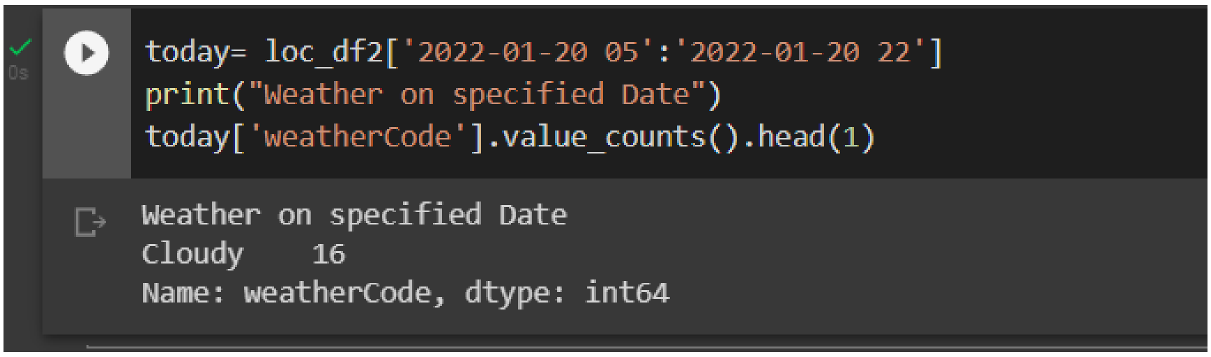


```
loc_df2['weatherCode'].value_counts()
```

```
Cloudy           62
Drizzle          11
Light Rain        8
Partly Cloudy     5
Rain              5
Mostly Cloudy     3
Mostly Clear      2
Heavy Rain        1
Name: weatherCode, dtype: int64
```

Fig. 4.1 Weather Condition Count

We are Checking the weather condition counts that are in dataset that represents weather condition for next 4 days.

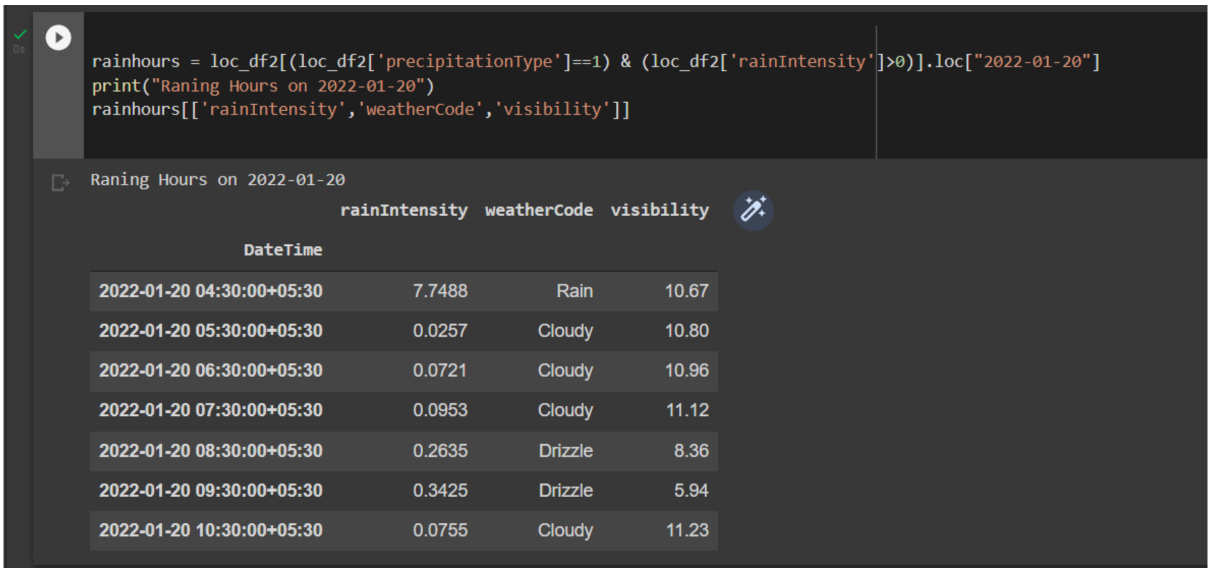


```
today= loc_df2['2022-01-20 05':'2022-01-20 22']
print("Weather on specified Date")
today['weatherCode'].value_counts().head(1)
```

```
Weather on specified Date
Cloudy    16
Name: weatherCode, dtype: int64
```

Fig. 4.2 Weather Between date and time

In the above image we are checking weather Condition between Specific date and time



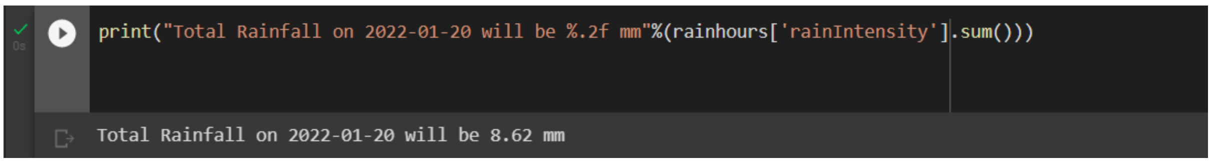
```
rainhours = loc_df2[(loc_df2['precipitationType']==1) & (loc_df2['rainIntensity']>0)].loc["2022-01-20"]
print("Raning Hours on 2022-01-20")
rainhours[['rainIntensity','weatherCode','visibility']]
```

```
Raning Hours on 2022-01-20
```

DateTime	rainIntensity	weatherCode	visibility
2022-01-20 04:30:00+05:30	7.7488	Rain	10.67
2022-01-20 05:30:00+05:30	0.0257	Cloudy	10.80
2022-01-20 06:30:00+05:30	0.0721	Cloudy	10.96
2022-01-20 07:30:00+05:30	0.0953	Cloudy	11.12
2022-01-20 08:30:00+05:30	0.2635	Drizzle	8.36
2022-01-20 09:30:00+05:30	0.3425	Drizzle	5.94
2022-01-20 10:30:00+05:30	0.0755	Cloudy	11.23

Fig. 4.3 Raining Hours

We are analysing the Raining Hours on Specific Day. It results a table with raining hours, Rain intensity along with visibility.



```
print("Total Rainfall on 2022-01-20 will be %.2f mm"%(rainhours['rainIntensity'].sum()))
```

```
Total Rainfall on 2022-01-20 will be 8.62 mm
```

Fig. 4.4 Total Rain fall in a day

Here we are Calculating the total amount of rain fall in between Specific Date and time

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide accessible way to see and understand trends, outliers, and patterns in data. In Weather data analysis we used Matplotlib and Seaborn for Plotting the graphs.

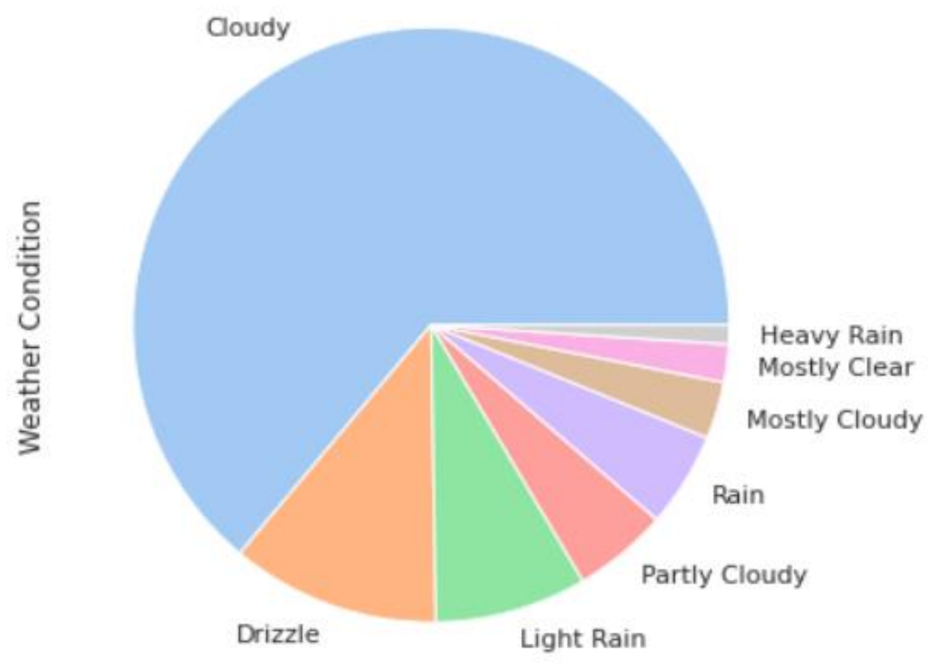
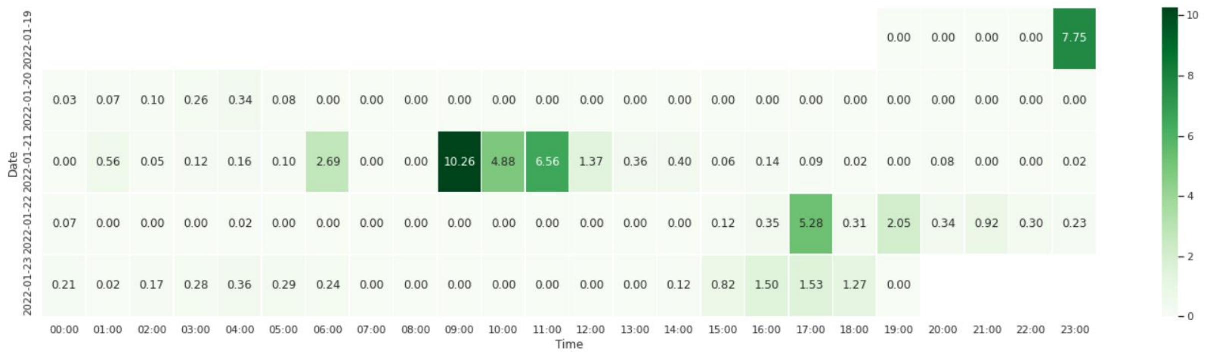


Fig. 4.5 Weather Condition

Here in this Pie Chart Represents weather condition for next 4 days.



Date	00:00	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00
2022-01-19																				0.00	0.00	0.00	0.00	7.75
2022-01-20	0.03	0.07	0.10	0.26	0.34	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2022-01-21	0.00	0.56	0.05	0.12	0.16	0.10	2.69	0.00	0.00	10.26	4.88	6.56	1.37	0.36	0.40	0.06	0.14	0.09	0.02	0.00	0.08	0.00	0.00	0.02
2022-01-22	0.07	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.12	0.35	5.28	0.31	2.05	0.34	0.92	0.30	0.23
2022-01-23	0.21	0.02	0.17	0.28	0.36	0.29	0.24	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.12	0.82	1.50	1.53	1.27	0.00				

Fig. 4.6 Rain Intensity

This is heat map representation of Rain fall intensity where X axis represents Time and Y axis represents the date.

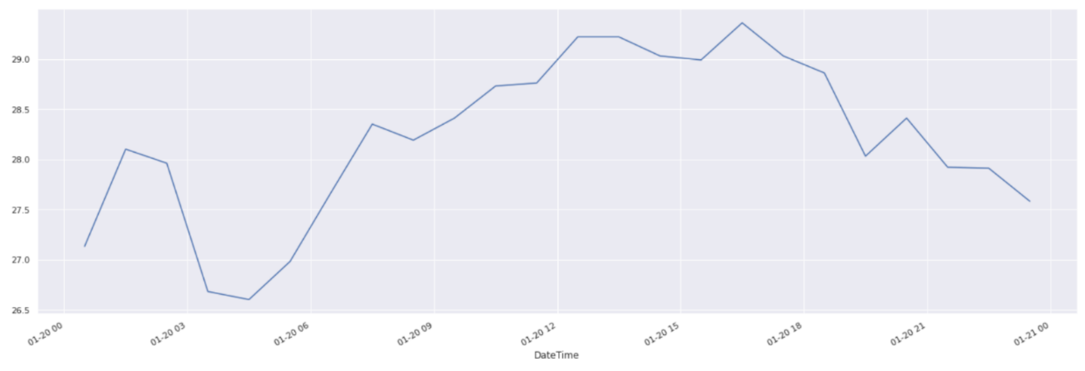


Fig. 4.7 Temperature on a Day

This is Line Plot Shows the temperature of Specific Date.

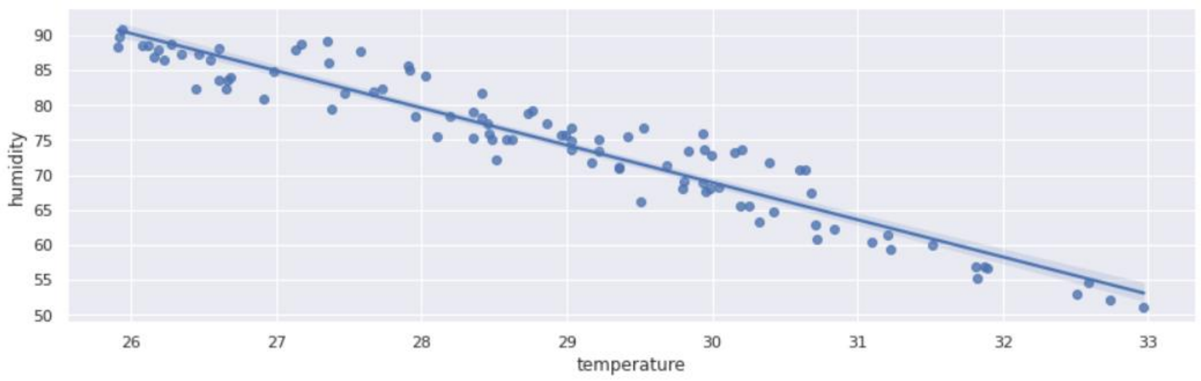


Fig. 4.8 Humidity vs Temperature

In the above Scatterplot X axis represents Temperature and Y axis represents Humidity and, in this graph, we can observe that as Temperature increases Humidity Decries.

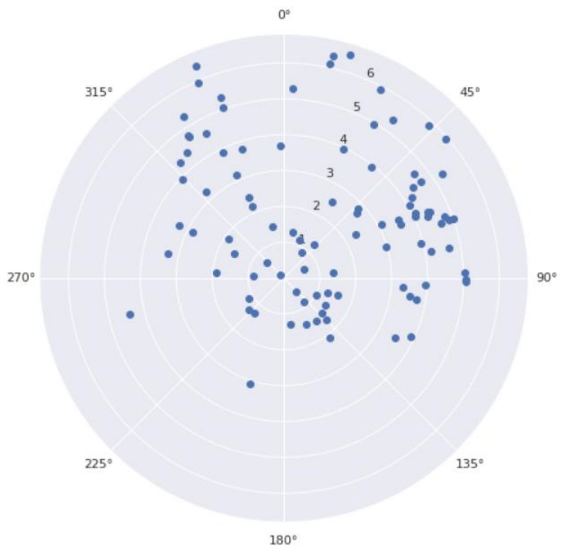


Fig. 4.9 Wind direction and wind Speed

This is Facet Grid to represent the Wind Direction and wind Speed

CHAPTER-5

CONCLUSION

REFERENCES

1. Jake Vander plas, "Python Data Science Handbook: Essential tools for working with data", O'Reilly Publishers, I Edition.
2. https://developer.tomorrow.io/
3. https://seaborn.pydata.org/examples/
4. https://levelup.gitconnected.com/weather-data-analysis-and-visualization-with-pandas-dcc9c6078065
5. https://www.youtube.com/watch?v=mKSWAlvXSmw