WEATHER ANALYSIS USING PYTHON

Prepared in partial fulfilment of the summer internship program on data anaylsis

AT

Under the guidance of

T.Srikanya,APSSDC

K.Meenakshi,APSSDC

Submitted by

Kondapaturi Venkata Tanuja-22NE1A05A0
Kakarparthi Jaya Venkata Lakshmi-22NE1A0575
Kodari Alekhya-22NE1A0588
Nallamothu Divija-22NE1A05C7

TIRUMALA ENGINEERING COLLEG,JONNALAGADDA,NARASARAOPET

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to all those who have contributed to the successful completion of my summer internship project at Andhra Pradesh Skill Development Corporation (APSSDC). This opportunity has been an enriching and transformative experience for me, and I am truly thankful for the support, guidance, and encouragement I have received along the way. First and foremost, I extend my sincere regards to Mrs T.Srikanya and Mrs k. Meenakshi ,my supervisors and mentors, for providing me with valuable insights, constant guidance, and unwavering support throughout the duration of the internship. Their expertise and encouragement have been instrumental in shaping the direction of this project. I would like to thank the entire team at Andhra Pradesh Skill Development Corporation (APSSDC) for fostering a collaborative and innovative environment. The camaraderie, knowledge sharing, and feedback I received from my colleagues significantly contributed to the development and success of this project. In conclusion, I am honoured to have been a part of this internship program, and I look forward to leveraging the skills and knowledge gained to contribute positively to future endeavours.

Thank you.

Sincererly,

K.V.Tanuja

K.V.J.Lakshmi

K.Alekhya

N.Divija

Series of contents

1.Abstract

This project aims to analyze historical weather data to uncover patterns, trends, and insights that can help in various applications such as climate research, agriculture, and disaster management. Utilizing Python and its data analysis libraries like Pandas, NumPy, and Matplotlib, the project involves cleaning and preprocessing the dataset, performing statistical analyses, and visualizing the results. By grouping, filtering, and aggregating the data, the analysis reveals important weather trends and their impact over time. The project concludes with a set of visualizations and findings that can be used for further research and decision-making processes.

1. Introduction

Weather analysis is a critical component of modern meteorology and has far-reaching implications across numerous sectors. Accurate weather forecasting plays a pivotal role in agriculture by guiding planting and harvesting decisions, in transportation by ensuring safe travel conditions, and in urban planning by helping manage resources and infrastructure effectively. The rise of big data has transformed the way weather information is processed and utilized, making it possible to analyze vast amounts of historical and real-time data to predict weather patterns with greater precision.

- **Purpose**

  The purpose of this project is to harness the capabilities of Python programming to perform a detailed analysis of weather data. By employing data analysis and visualization techniques, the project aims to uncover valuable insights from historical weather records. This includes identifying significant trends, correlations, and anomalies that could inform better forecasting models and operational strategies. The ultimate goal is to enhance the understanding of weather patterns and contribute to more informed decision-making processes in various domain

impacted by weather conditions.

- **Scope**

  This project encompasses several key phases: data collection, data preprocessing, exploratory data analysis, and visualization. The scope includes:

  Data Collection: Gathering weather data from reliable sources such as weather APIs or historical weather datasets.

  Data Preprocessing: Cleaning and preparing the data for analysis, which involves handling missing values, normalizing data, and transforming data into suitable formats.

  Exploratory Data Analysis (EDA): Analyzing the data to identify patterns, trends, and relationships. This phase involves statistical analysis and the use of visual tools to understand the data better.

  Visualization: Creating various plots and charts to visually represent the data, making it easier to interpret and communicate findings. This includes line charts, bar charts, scatter plots, and pie charts.

  By leveraging Python's data analysis libraries, this project seeks to provide a comprehensive approach to weather data analysis, demonstrating the power of data-driven insights in understanding and predicting weather phenomena.

3.System Requirements

- ***Operating System***

  To execute the weather analysis project effectively, the choice of operating system is crucial. The primary operating systems compatible with the tools and libraries used in this project include:

Windows: A widely used operating system that supports all major Python distributions and data analysis libraries. It is suitable for most users and provides a robust environment for software development and data analysis.

macOS: Known for its stability and Unix-based architecture, macOS is also fully compatible with Python and data analysis libraries. It offers a reliable platform for data-driven projects and supports various Python development tools.

Linux: Preferred by many developers for its open-source nature and powerful command-line capabilities. Linux distributions such as Ubuntu, Fedora, and CentOS are well-suited for running Python applications and managing data analysis workflows.

Each of these operating systems provides the necessary environment to run Python and related libraries, though some specific configurations might be required for optimal performance.

- ***Software***

The software requirements are essential for developing and running the weather analysis project. These include:

Python: The core programming language used for data analysis in this project. Python 3.8 or later is recommended for compatibility with the latest libraries and features. Python should be installed along with a package manager such as pip to handle library installations.

Integrated Development Environment (IDE): An IDE facilitates coding and debugging. Recommended IDEs include:

Jupyter Notebook: Ideal for interactive coding and visualization. It allows for creating and sharing documents that contain live code, equations, visualizations, and narrative text.

Data Analysis Libraries: Essential Python libraries for this project include:

Pandas: A powerful library for data manipulation and analysis. It provides data structures and functions needed to work with structured data, including DataFrames for handling and analyzing tabular data.

NumPy: A fundamental library for numerical computations. It provides support for arrays and matrices, along with mathematical functions to perform operations on these structures.

Matplotlib: A plotting library for creating static, interactive, and animated visualizations. It is used to generate various types of plots, including line charts, bar charts, and scatter plots.

Seaborn: A data visualization library built on top of Matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics.

Additional Tools:

Git: For version control, allowing for tracking changes and collaborating with others. GitHub or GitLab can be used for repository hosting.

Virtual Environment: Tools like venv or conda to create isolated environments for managing dependencies and preventing conflicts between packages.

- ***Hardware***

The hardware requirements for running the weather analysis project are generally modest, but specific configurations can enhance performance:

Processor: A modern multi-core processor (e.g., Intel i5/i7 or AMD Ryzen) is recommended to handle data processing and computations efficiently. While Python's performance is often CPU-bound, multi-core processors can facilitate faster execution of parallel tasks.

Memory (RAM): At least 8 GB of RAM is recommended for handling large datasets and performing complex calculations. For larger datasets or more intensive analysis, 16 GB or more may be beneficial.

Storage: Sufficient storage space is needed to store datasets, code, and outputs. An SSD with at least 50 GB of free space is recommended to ensure quick access to files and smooth performance during data analysis.

These system requirements ensure that the weather analysis project can be executed efficiently and effectively, providing a smooth development and analysis experience.

4.Architecture of the Project

- ***Data Analysis***

1. **Statistical Analysis**:

Descriptive Statistics: Calculating measures such as mean,

median, standard deviation to understand data distribution.

Correlation Analysis: Assessing relationships between different weather variables.

2. Predictive Modeling:

Machine Learning: Implementing algorithms for forecasting weather patterns. Techniques such as regression, time series analysis, or machine learning models (e.g., Random Forest, ARIMA) may be used.

Model Evaluation: Evaluating model performance using metrics like accuracy, precision, recall, and RMSE.

- ***Data Visualization***

1. **Plotting Libraries**:

Matplotlib: Used for creating a wide range of static plots including line charts, bar charts, and histograms.

Seaborn: Provides high-level functions for statistical visualizations such as heatmaps and pair plots.

Plotly: For creating interactive visualizations like scatter plots and 3D plots.

2. Visualization Design:

Chart Types: Choosing appropriate chart types based on the analysis requirements (e.g., time series plots for trend analysis, scatter plots for relationship analysis).

Customizations: Enhancing visualizations with titles, labels, legends, and color schemes to improve readability and interpretability.

- ***Deployment***

1. **Environment Setup:**

Virtual Environments: Creating isolated environments using venv or conda to manage dependencies and ensure compatibility.

Containerization: Using Docker to containerize the application for consistent deployment across different environments.

2. **Hosting:**

Web Hosting: Deploying the application to web servers or cloud platforms (e.g., AWS, Azure) to make it accessible to users.

- ***Maintenance and Updates***

1. Monitoring:

Performance Monitoring: Tracking system performance and handling potential issues such as bottlenecks or failures.

Data Updates: Regularly updating the dataset with new information and re-running analyses as needed.

2. Continuous Improvement:

Feature Enhancements: Adding new features or improving existing ones based on user feedback and emerging needs.

Bug Fixes: Addressing any bugs or issues that arise during usage.

This architecture ensures that the weather analysis project is organized and modular, facilitating efficient data handling, comprehensive analysis, and clear presentation of results.

5.Uses of Data Analysis Libraries

Data analysis libraries play a crucial role in modern data science and analytics projects by providing powerful tools and functionalities for handling, analyzing, and visualizing data. In the context of a weather analysis project, these libraries facilitate various tasks, from data collection and cleaning to advanced analytics and visualization. Here's an in-depth look at the uses of key data analysis libraries:

- ***Pandas***

1. **Data Manipulation and Cleaning:**

DataFrame Management: Pandas provides the DataFrame object, which is essential for storing and manipulating structured data. It allows for easy indexing, filtering, and transformation of data.

Handling Missing Values: Functions like dropna() and fillna() help in managing missing or incomplete data by either removing affected rows or filling them with appropriate values.

Data Merging and Joining: Pandas supports merging and joining of datasets using functions such as merge(), concat(), and join(), making it easier to combine data from multiple sources.

2. **Data Aggregation and Grouping**:

GroupBy Operations: The groupby() method enables the aggregation of data based on specific criteria, allowing for operations such as summing, averaging, or counting grouped data.

Pivot Tables: Creating pivot tables using pivot_table() provides a way to summarize and analyze data by organizing it into a tabular format.

3. **Data Transformation**:

Applying Functions: The apply() method allows for the application of custom functions across DataFrame columns or rows, facilitating complex transformations.

Reshaping Data: Pandas provides tools like melt() and pivot() for reshaping data to suit different analytical needs.

- **NumPy**

1. **Numerical Computation:**

Array Operations: NumPy's ndarray object supports efficient storage and manipulation of large arrays and matrices of numerical data.

Mathematical Functions: It provides a wide range of mathematical functions, such as mean(), median(), std(), and correlation(), essential for statistical analysis and calculations.

2. Linear Algebra:

Matrix Operations: NumPy includes functions for performing matrix operations, such as multiplication, inversion, and decomposition, which are fundamental for advanced mathematical and statistical analysis.

3. Random Sampling:

Generating Random Data: Functions like randn() and random() are used to generate random samples, which can be useful for simulations and modeling.

- ***Matplotlib***

1. **Data Visualization**:

Plotting: Matplotlib is widely used for creating static, high-quality visualizations such as line plots, bar charts, histograms, and scatter plots.

Customization: It offers extensive customization options, including titles, labels, legends, and colors, enabling the creation of detailed and informative plots.

2. Subplots and Grids:

Multi-Plot Layouts: The subplot() function allows for the arrangement of multiple plots in a single figure, making it easier to compare different data visualizations side by side.

- ***Seaborn***

1. Statistical Visualizations:

Enhanced Plots: Seaborn provides high-level functions for creating more complex statistical visualizations, such as heatmaps, pair plots, and violin plots.

Theme and Color Management: It includes built-in themes and color palettes that improve the aesthetic quality of plots and make them more interpretable.

2. Data Exploration:

Visualization of Relationships: Functions like pairplot() and jointplot() help in exploring relationships between multiple variables and understanding their distributions.

- ***Plotly***

1. Interactive Visualizations:

Dynamic Charts: Plotly is known for its interactive charts, including scatter plots, line plots, and 3D plots. These charts

allow users to interact with the data by zooming, panning, and hovering over points.

Web Integration: Plotly integrates well with web frameworks, enabling the creation of interactive dashboards and web applications.

2. Complex Visualizations:

3D and Geo-Plots: It supports advanced visualizations such as 3D scatter plots and geographic maps, which are useful for presenting data with multiple dimensions.

- ***Statsmodels***

1. Statistical Analysis:

Regression Models: Statsmodels provides tools for estimating and interpreting various regression models, including linear regression, logistic regression, and time series models.

Hypothesis Testing: It supports a range of statistical tests, such as t-tests and chi-square tests, for hypothesis testing and validating analytical findings.

2. Time Series Analysis:

ARIMA Models: For forecasting time series data, Statsmodels includes functions for fitting ARIMA (AutoRegressive Integrated Moving Average) models and other time series forecasting methods.

6.Coding

The Weather Dataset

Description:

The Weather Dataset is a time-series dataset with per-hour

information about the weather condition at a particular location. It records Temperature, dew Point temperature, Relative humidity, Wind speed, Visibility, Pressure and conditions.

This Dataset available as a csv file. We are going to analyze this dataset using Pandas dataframe.

Workflow:

- Import library
- Analyzing the data
- solve few questions of Data Analytics

Code:

```
# import library
import pandas as pd
df=pd.read_csv(r"C:\Users\TANUJA\Downloads\Weather_Data.csv")
df
```

Analyzing the data

.head()

Its show the first N rows in the data(by default N=5)

code:

```
# Display the first few rows of the dataset
print(df.head())
```

.shape

its shows total number of rows and columns of the dataframe.

code:

```
df.shape
```

.index

This attribute provide the index of the dataframe.

Code:

```
df.index
```

columns

Its shows the name of each columns.

code:

df.columns

.dtypes

Its shows datatypes of each column.

code:

df.dtypes

unique()

In a column,its shows alll the unique values. It can be applied on a sinngle column only.Not on the whole dataframe.

code:

df['Weather'].unique()

.nunique()

Its shows the total number of unique values in each column.It can be applied on a single column as on whole dataframe.

code:

df.nunique()

.count

It show the total no. of non-null values in each column.It can be applied on a single column as well as whole dataframe.

code:

df.count()

.value_counts()

In a column ,it shows all the unique values with their count. It can be applied on single column only.

code:

df['Weather'].value_counts()

.info()

It provides basic information about the datframe.

code:

df.info()

info

provides basic information about the dataframe

code:

df.info()

Q.1 Find all the unique 'Wind speed' values in the data.

code:

df.head(2)

df.nunique()

df["Wind Speed_km/h"].nunique()

df["Wind Speed_km/h"].unique()

Q.2 Find the number of time when the 'Weather is exactly clear'.

code:

df.head(2)

#1: value_counts()

df.Weather.value_counts()

#2. Filtering

df.head(2)

df[df.Weather== 'Clear']

#3. groupby()

df.head(2)

df.groupby('Weather').get_group('Clear')

Q.3 Find the number of time when the 'Weather is exactly 4km/h'.

Code:

df[df['Wind Speed_km/h']==4]

df[df['Wind Speed_km/h']==4].count()

Q.4 Find out all the null values in the data.

code:

df.isnull().sum()

df.notnull().sum()

Q.5 Rename the column name 'Weather ' of the dataframe to 'Weather condition'.

Code:

df.head(2)

df.rename(columns={'Weather':'Weather Condition'}) *# only for this comands*

df.rename(columns={'Weather':'Weather Condition'},inplace=**True**) *# change column name permantly*

Q.6 What is the mean 'visibility'?.

code:

df.head(2)

df.Visibility_km.mean()

Q.7 What is the standard deviation of 'Pressure ' in this data?

Code:

df.head()

df.Press_kPa.std() *# nospace so using '.'*

Q.8 What is the variance of 'Relative Humidity ' in this data?

code:

df['Rel Hum_%'].var() *#space between name so using []*

Q.9 Find all instances when 'Snow' was recorded.

code:

#1:value_counts()

df.head(2)

df['Weather Condition'].value_counts()

#2: Filtering

df[df['Weather Condition']=='Snow']

#3: str.contains

df[df['Weather Condition'].str.contains('Snow')].tail(10)

```
df[df['Weather Condition'].str.contains('Snow')].head(10)
df[df['Weather Condition'].str.contains('Snow')].tail(10)
```

Q.10 Find all instances when 'Wind speed is above 24' and 'Visibility'is 25 .

Code:

```
df[(df['Wind Speed_km/h'] > 24) & (df['Visibility_km']==25)]
```

Q.11 What is the minimum value of each column against each 'Weather condition'?

code:

```
df.head(2)
df.groupby('Weather Condition').min()
```

Q.12 What is maximum value of each column against each 'Weather condition'

```
df.head(2)
df.groupby('Weather Condition').max()
```

Q.13 Show All the records where weather condition is Fog.

Code:

```
df[df["Weather Condition"]=='Fog']
```

Q.14 Find all instances when 'Weather is clear' or 'Visibility is above 40'.

Code:

```
df[(df['Weather Condition']=='Clear') |
(df['Visibility_km']>40)].head(10)
df[(df['Weather Condition']=='Clear') | (df['Visibility_km']>40)].tail(10)
```

Q.15 Find all instances when:

a) 'Weather is clear' and 'Relative humidity is greater than 50' or

b) 'Visibility' is above 40

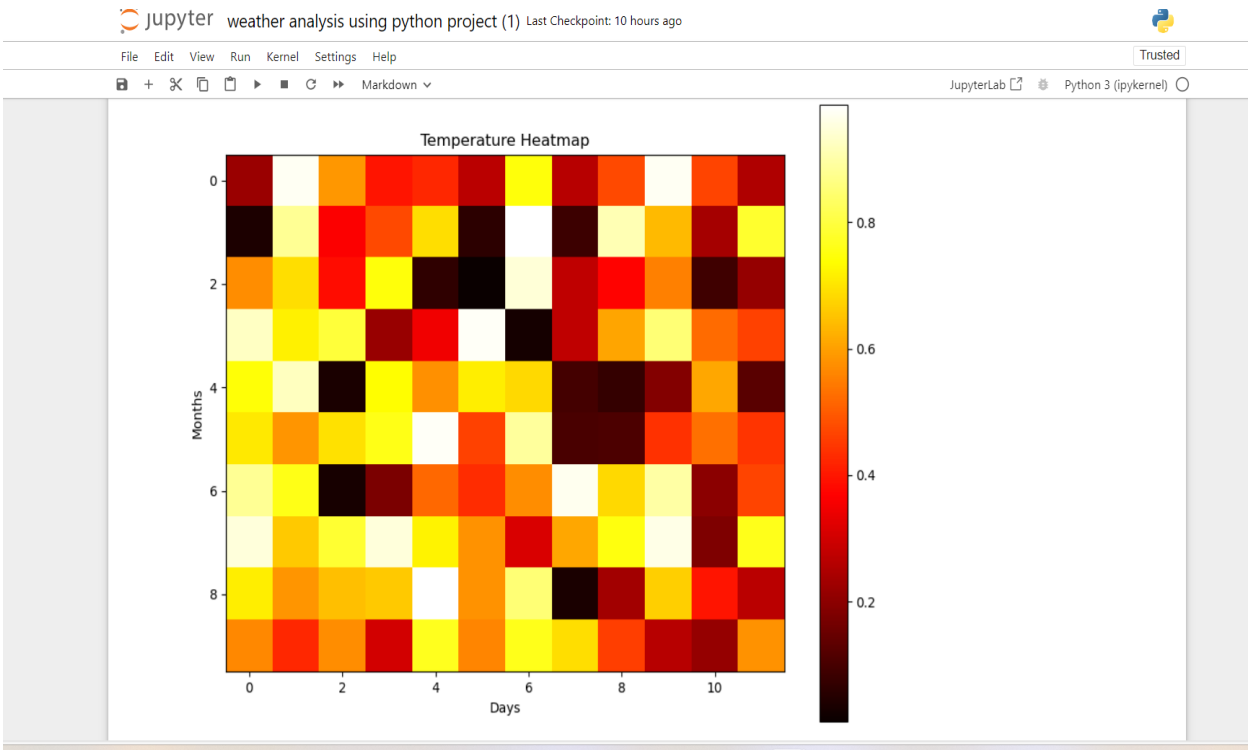
code:

```
df[(df['Weather Condition']=='Clear' ) & (df['Rel Hum_%'] >50) |
(df['Visibility_km']> 40)]
```

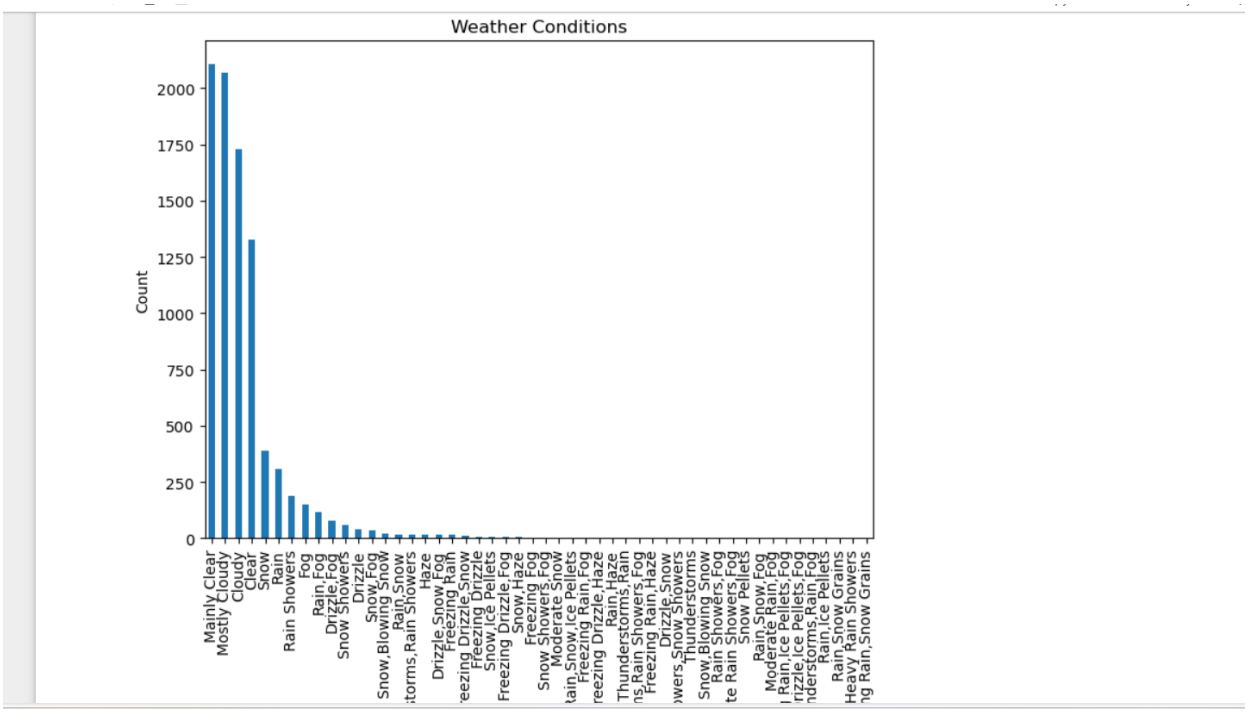
7.Data Visulization

Code:

```
# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import random
# Create a sample dataset
data = np.random.rand(10, 12)  # 10 rows (months) x 12 columns (days)
index = pd.date_range('2024-01-01', periods=10, freq='M')
columns = pd.date_range('2024-01-01', periods=12, freq='D')
df = pd.DataFrame(data, index=index, columns=columns)
# Create a heatmap
plt.figure(figsize=(10, 8))
plt.imshow(df, cmap='hot', interpolation='nearest')
plt.title('Temperature Heatmap')
plt.xlabel('Days')
plt.ylabel('Months')
plt.colorbar()
plt.show()
```

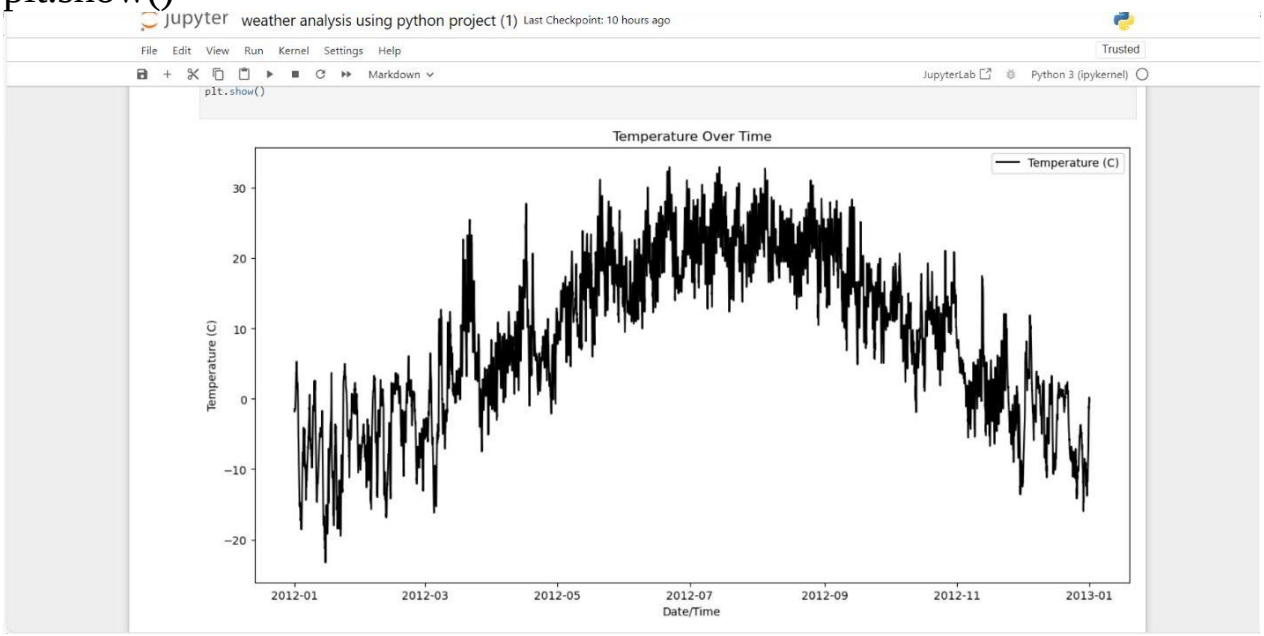



```
# create a BAR CHART
# Import libraries
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv(r"C:\Users\TANUJA\Downloads\Weather_Data.csv")
df['Weather'].value_counts().plot(kind='bar', figsize=(8, 6))
plt.title('Weather Conditions')
plt.xlabel('Weather')
plt.ylabel('Count')
plt.show()
```

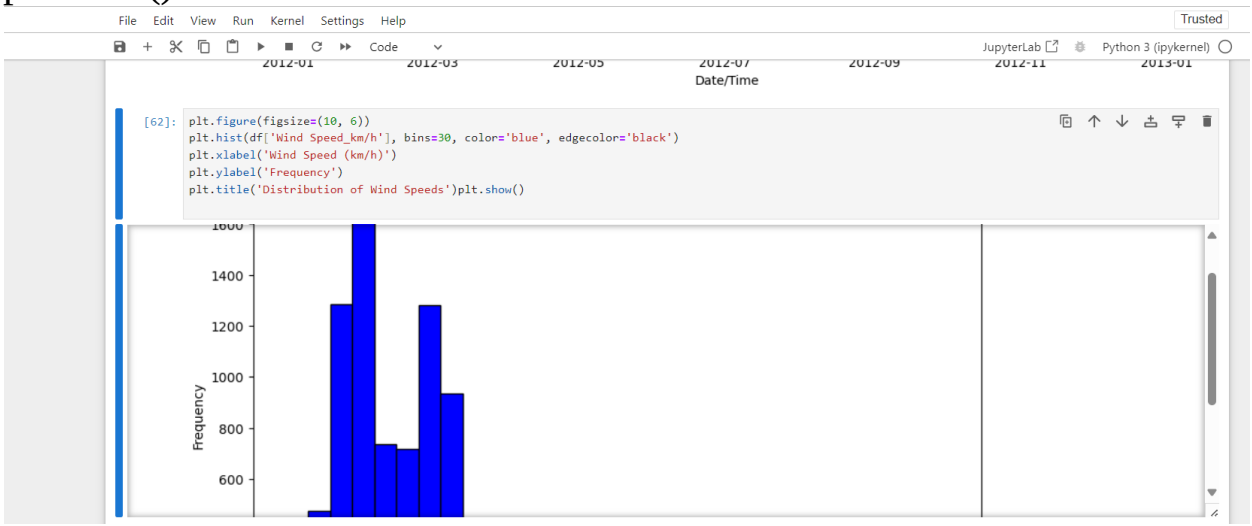



```
#create a LINE GRAPH
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
# Load dataset
df=pd.read_csv(r"C:\Users\TANUJA\Downloads\Weather_Data.csv")
# Ensure the 'Date/Time' column is parsed as datetime
df['Date/Time'] = pd.to_datetime(df['Date/Time'])
plt.figure(figsize=(14, 7))
plt.plot(df['Date/Time'], df['Temp_C'], label='Temperature (C)',
color='black')
plt.xlabel('Date/Time')
plt.ylabel('Temperature (C)')
plt.title('Temperature Over Time')
plt.legend()
```

plt.show()

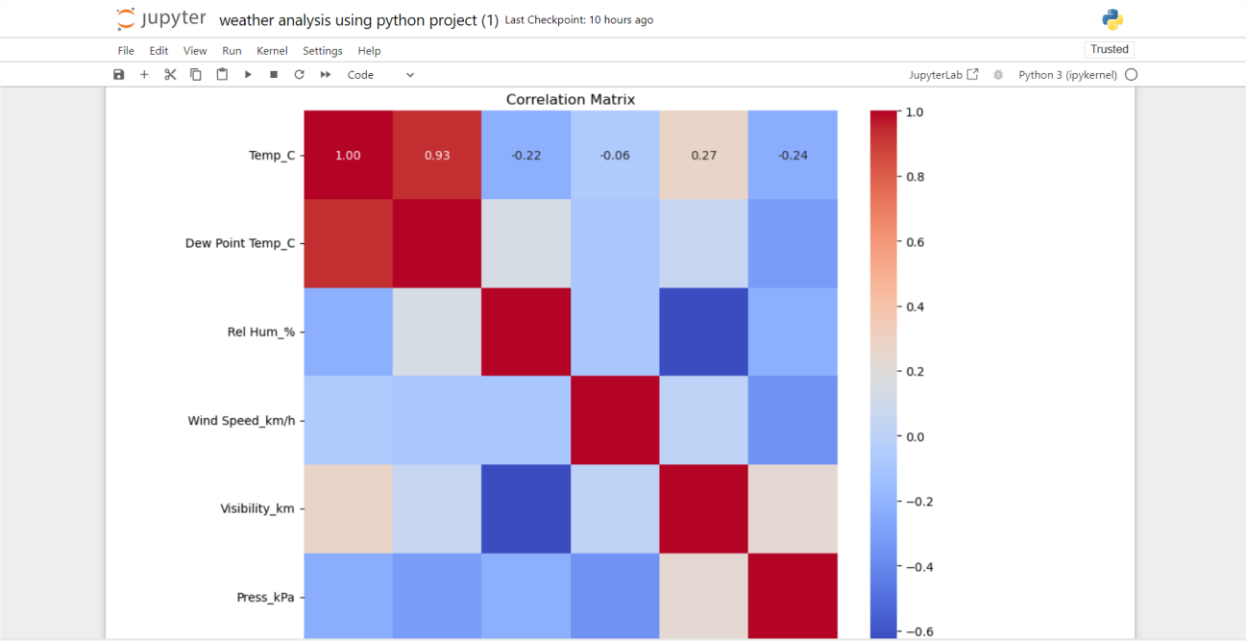


plt.figure(figsize=(10, 6))
plt.hist(df['Wind Speed_km/h'], bins=30, color='blue', edgecolor='black')
plt.xlabel('Wind Speed (km/h)')
plt.ylabel('Frequency')
plt.title('Distribution of Wind Speeds')
plt.show()

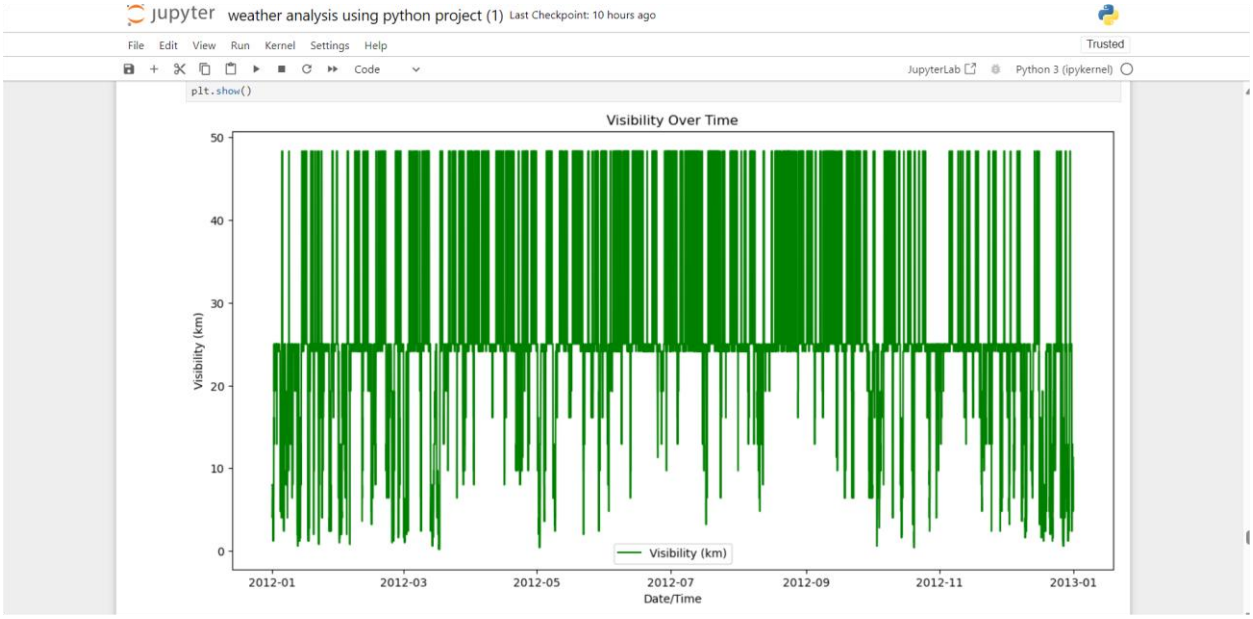


plt.figure(figsize=(10, 8))

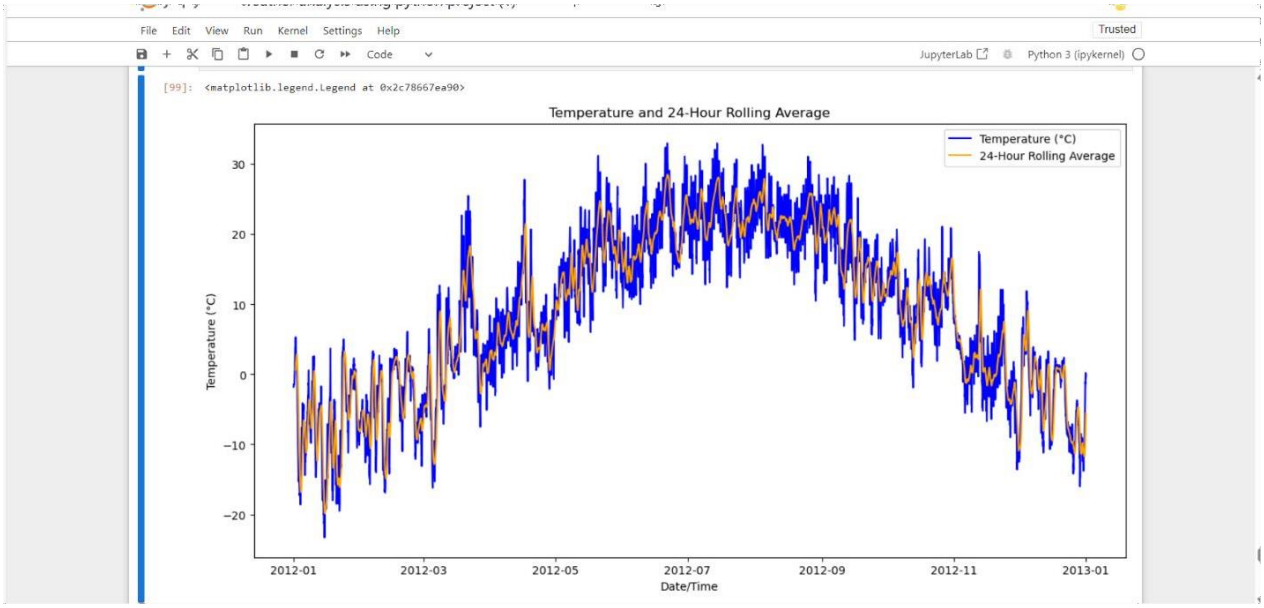
```
correlation_matrix = df[['Temp_C', 'Dew Point Temp_C', 'Rel Hum_%',
'Wind Speed_km/h', 'Visibility_km', 'Press_kPa']].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```




```
plt.figure(figsize=(14, 7))
plt.plot(df['Date/Time'], df['Visibility_km'], label='Visibility (km)',
color='green')
plt.xlabel('Date/Time')
plt.ylabel('Visibility (km)')
plt.title('Visibility Over Time')
plt.legend()
plt.show()
```




```
import matplotlib.pyplot as plt
df=pd.read_csv(r"C:\Users\TANUJA\Downloads\Weather_Data.csv")
df['Date/Time']=pd.to_datetime(df['Date/Time'])
df.set_index('Date/Time', inplace=True)
rolling_avg_temp = df['Temp_C'].rolling(window=24).mean()
plt.figure(figsize=(14, 7))
plt.plot(df.index, df['Temp_C'], label='Temperature (°C)', color='blue')
plt.plot(df.index, rolling_avg_temp, label='24-Hour Rolling Average',
color='orange')
plt.xlabel('Date/Time')
plt.ylabel('Temperature (°C)')
plt.title('Temperature and 24-Hour Rolling Average')
plt.legend()
```



8. Adavantages of this anaylsis

- **Extensive Libraries**: Python has a rich ecosystem of libraries like NumPy, Pandas, Matplotlib, and SciPy, which are excellent for data analysis and visualization.

- **Machine Learning Integration**: Python's compatibility with machine learning libraries such as TensorFlow, scikit-learn, and Keras allows for advanced predictive modeling and analysis.

- **Ease of Use**: Python’s syntax is clear and readable, making it easier for both beginners and experienced developers to write and maintain code.

- **Community Support**: Python has a large and active community, providing ample resources, tutorials, and forums for troubleshooting and collaboration.

- **APIs and Data Access**: Python offers convenient ways to access weather data through APIs from services like OpenWeatherMap, Weather.com, and NOAA, simplifying data retrieval and integration.

- **Automated Workflows**: Python's capabilities in scripting and automation allow for the creation of automated data collection, processing, and analysis workflows, saving time and effort.

- **Visualization Tools**: Python has powerful visualization libraries such as Matplotlib, Seaborn, and Plotly, enabling the creation of detailed and interactive visual representations of weather data.

- **Flexibility and Scalability**: Python is versatile, allowing integration with other languages and systems, and scalable for large datasets and complex computations.

- **Cost-Effective**: Python is open-source and free to use, making it a cost-effective option for developing weather analysis applications.
- **Web Development**: With frameworks like Django and Flask, Python can be used to develop web applications for disseminating weather information and analysis to a wider audience.

- **Data Cleaning and Preprocessing**: Python's libraries provide robust tools for cleaning and preprocessing data, which is crucial for accurate weather analysis.

- **Real-Time Analysis**: Python's capabilities in handling real-time data streams allow for real-time weather monitoring and alert systems.

- **Interoperability**: Python can easily interface with other programming languages and tools, enhancing its flexibility in diverse computational environments.

- **Big Data Handling**: With libraries like Dask and PySpark, Python can manage and analyze large-scale weather datasets efficiently.

- **Geospatial Analysis**: Libraries such as GeoPandas, Shapely, and Folium facilitate geospatial data analysis and visualization, which is crucial for mapping weather patterns.

- **Time Series Analysis**: Python's strong support for time series analysis through libraries like Pandas and statsmodels allows for detailed trend and anomaly detection in weather data.

- **Cloud Integration**: Python integrates seamlessly with cloud services (e.g., AWS, Google Cloud, Azure), enabling scalable computing and storage solutions for extensive weather datasets.

- **Custom Algorithm Development**: Python's flexibility allows for the development and implementation of custom algorithms tailored to specific weather analysis needs.

- **Collaboration and Sharing**: Tools like Jupyter Notebooks and Google Colab make it easy to share code and analysis results with collaborators, enhancing teamwork and reproducibility.

- **Open Data Sources**: Python supports integration with numerous open weather data sources, providing a wealth of information for analysis without additional cost.

9. conclusion

In conclusion, undertaking a weather analysis project using Python offers a multitude of advantages that make it an optimal choice for researchers, data scientists, and developers. The extensive libraries and tools available in Python streamline the processes of data collection,

cleaning, analysis, and visualization. Python's ease of use, flexibility, and strong community support enhance productivity and foster innovation. By leveraging machine learning and big data capabilities, Python enables advanced predictive modeling and real-time analysis, essential for accurate weather forecasting and disaster management. Additionally, Python's ability to integrate with various APIs, cloud services, and other programming languages ensures a robust and scalable solution for comprehensive weather analysis. Overall, Python's versatility and power make it an invaluable tool for addressing the complex challenges of weather analysis and contributing to informed decision-making across various sectors.

10. Referacnce Links

LINK1-Historical Weather Data (NOAA)

https://www.ncei.noaa.gov/products/land-based-station/global- historical-climatology-network-daily

LINK-2-NOAA Climate Data Online

https://purposeontheplanet.org/climate-action-plan/?msclkid=1bdf49dee85010ffa5ee2f7be5ae54c

LINK-3:Articles related to weather data analysis.

https://www.nature.com/articles/d41586-024-02391-9

Link-4: Data set from github

https://github.com/velicki/Weather_Data_Analysis_Project/blob/main/Weather_Data.csv