**COVER PAGE**

**DATA SCIENCE TOOLBOX PYTHON PROGRAMMING**

**PROJECT REPORT**

(Project Semester January-April 2025)

**MADRID DAILY WEATHER ANALYSIS**

**Submitted by**

**NAME:** P.HEMANTH

**REGISTRATION NO:** 12304744

**PROGRAMME AND SECTION:** K23GN30

**COURSE CODE** **:** CSE375

Under the Guidance of

**MRS. AASHIMA (UID: 28968)**

**Discipline of CSE/IT**

**Lovely School of Computer Science**

**Lovely Professional University, Phagwara**

**CERTIFICATE**

This is to certify that **P.Hemanth** bearing Registration no. **12304744** has completed **CSE375** project titled, **“Mrs.Aashima”** under my guidance and supervision. To the best of my knowledge, the present work is the result of his/her original development, effort and study.

**Signature and Name of the Supervisor**

**Designation of the Supervisor**

**School of Computer Science**

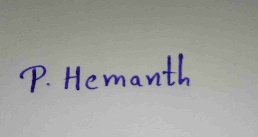
Lovely Professional University

Phagwara, Punjab.

Date:

**DECLARATION**

I, P.Hemanth student of CSE (Program name) under CSE/IT Discipline at, Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own intensive work and is genuine.



Date:11-04-2025 Signature

Registration No: 12304744 Name of the student: P.Hemanth

|  |  |
| --- | --- |
| **Title** | **Pg.no** |
| **1. Introduction** | **5** |
| **2. Source of Dataset** | **5** |
| **3. EDA Process** | **6** |
| **4. Analysis on Dataset** | **7** |
| **4.1 Temperature Distribution** | **7** |
| **4.2 Correlation Between Temperatures** | **8** |
| **4.3 Monthly Average Analysis** | **9** |
| **4.4 Scatter Plot: Min vs Max Temperature** | **10** |
| **4.5 Pairwise Temperature Comparison** | **11** |
| **4.6 Monthly Record Distribution** | **12** |
| **4.7 Time Series Analysis** | **13** |
| **5. Predictive Modeling** | **14** |
| **5.1 Manual Linear Regression** | **14** |
| **6. Conclusion** | **15** |
| **7. Future Scope** | **15** |
| **8. References** | **15** |
| **9.Source code** | **16 - 19** |

**1. Introduction**

In today’s data-driven world, the ability to extract meaningful insights from raw data is a crucial skill. This project focuses on exploring and analyzing weather data from Madrid, Spain, covering the years 1997 to 2015. It demonstrates the end-to-end data analysis process, including data cleaning, exploratory data analysis (EDA), statistical analysis, visualization, and basic predictive modeling using Python.

**What is EDA?**

Exploratory Data Analysis (EDA) is the process of examining datasets to summarize their main characteristics, often using visual methods. EDA is crucial because it helps us:

* Detect anomalies or outliers
* Understand data distributions and relationships
* Formulate hypotheses for further analysis
* Prepare data for modelling

**Why EDA is Important**

EDA ensures that we don’t enter the modeling phase blindly. It lays the groundwork for more complex analysis and provides a visual understanding of patterns and trends within the dataset.

**How We Are Achieving EDA in This Project**

We used Python libraries like pandas, NumPy, seaborn, and matplotlib to:

* Handle missing values
* Generate summary statistics
* Visualize distributions and relationships
* Create insightful plots (boxplots, heatmaps, bar charts, scatter plots, etc.)

**2. Source of Dataset**

The dataset used in this project is publicly available and was downloaded from:

* **Source**: [Kaggle / Custom Weather Archive]
* **File Name**: Madrid Daily Weather 1997-2015.csv
* **Data Range**: January 1, 1997 - December 31, 2015
* **Primary Attributes**: Date, Min Temperature, Max Temperature, Mean Temperature, and other weather indicators.

**3. EDA Process**

**3.1 Data Cleaning & Preprocessing**

* Loaded dataset using pandas.read\_csv()
* +Removed extra white spaces from column names
* Converted 'CET' column to datetime format
* Handled missing values using:
  + Row drop for invalid date entries
  + Mean substitution for numeric columns
* Extracted new time features: Year, Month, Day

**3.2 Summary Statistics**

* Used df.describe() to view distributions of temperature-related columns
* Used df.info() to verify data types and detect missing data
* Created correlation matrices to understand relationships between variables

**4. Analysis on Dataset**

**4.1 Temperature Distribution**

**i. Introduction**

This section analyzes how temperatures (min, max, mean) are distributed over the dataset timeframe.

**ii. General Description**

We examine the spread and central tendency of daily temperatures to understand climate behavior in Madrid.

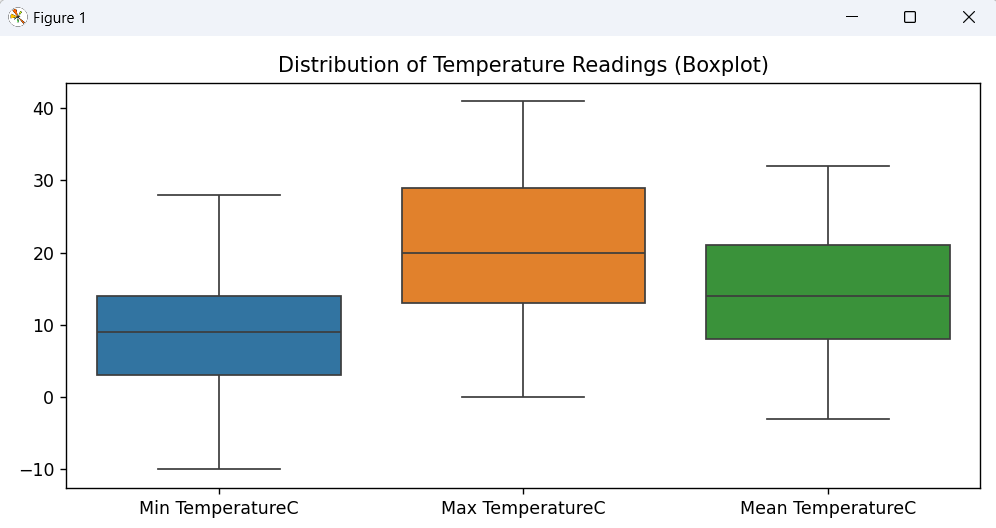
**iii. Functions and Formulas**

* Used df[['Min TemperatureC', 'Max TemperatureC', 'Mean TemperatureC']].describe()
* Visualized with seaborn boxplots

**iv. Analysis Results**

The boxplot shows consistent seasonal trends and identifies some outliers.

**v. Visualization**



**4.2 Correlation Between Temperatures**

**i. Introduction**

We explore how different temperature measures correlate.

**ii. General Description**

Strong relationships between temperature types may indicate consistency in weather recordings.

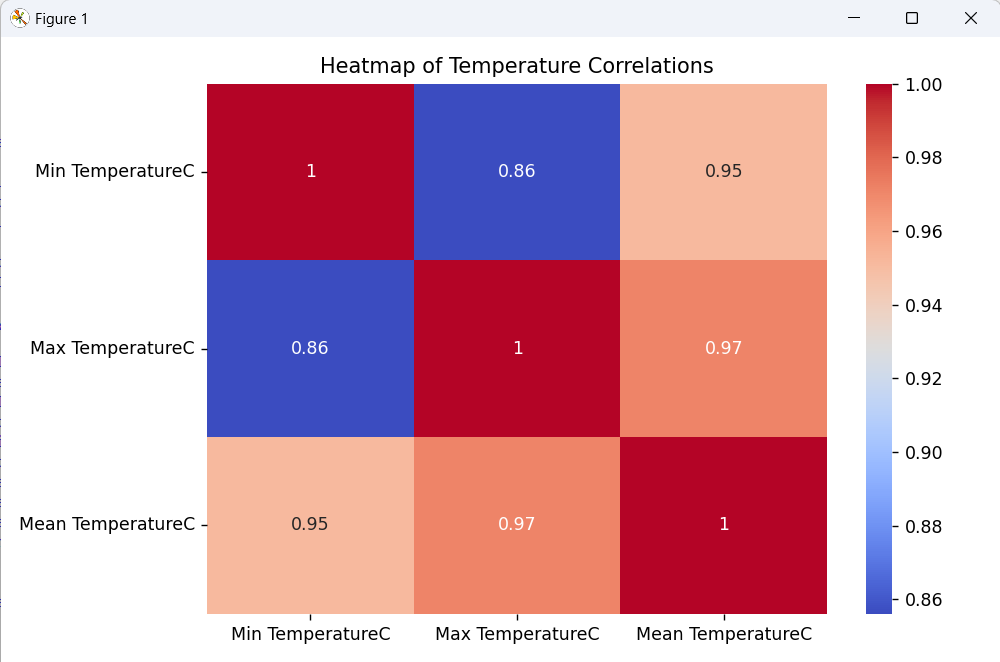
**iii. Functions and Formulas**

* Used .corr() to generate correlation matrix
* Visualized with seaborn heatmap

**iv. Analysis Results**

Strong positive correlation observed, especially between Min and Mean temperatures.

**v. Visualization**



**4.3 Monthly Average Analysis**

**i. Introduction**

We explore seasonal trends through monthly average temperatures.

**ii. General Description**

Seasonality affects climate and helps in planning agricultural or energy-based decisions.

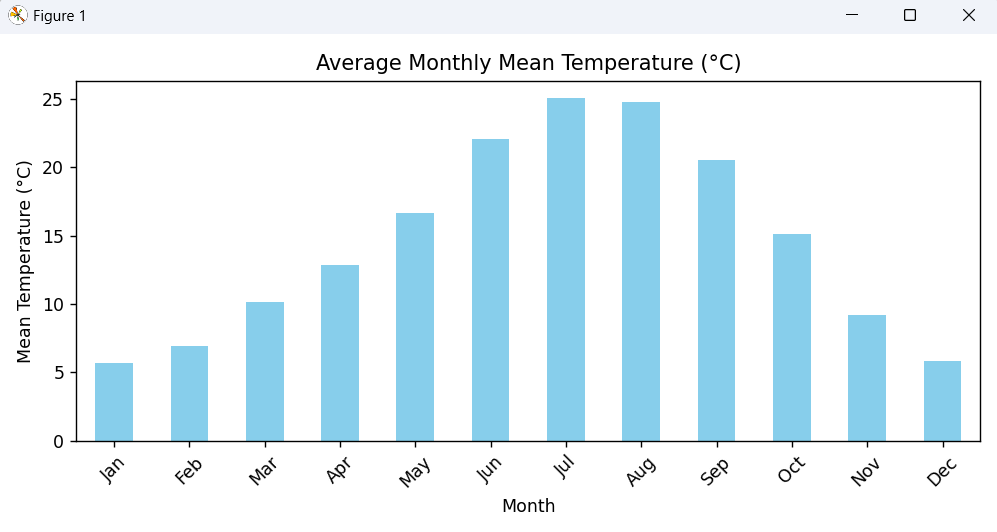
**iii. Functions and Formulas**

* Grouped by Month: df.groupby('Month')['Mean TemperatureC'].mean()
* Bar chart for visualization

**iv. Analysis Results**

Clear upward trend from Jan to July, then downward, consistent with Mediterranean climate.

**v. Visualization**



**4.4 Scatter Plot: Min vs Max Temperature**

**i. Introduction**

Shows relationship between daily minimum and maximum temperatures.

**ii. General Description**

Can help detect unusual climate days or data inconsistencies.

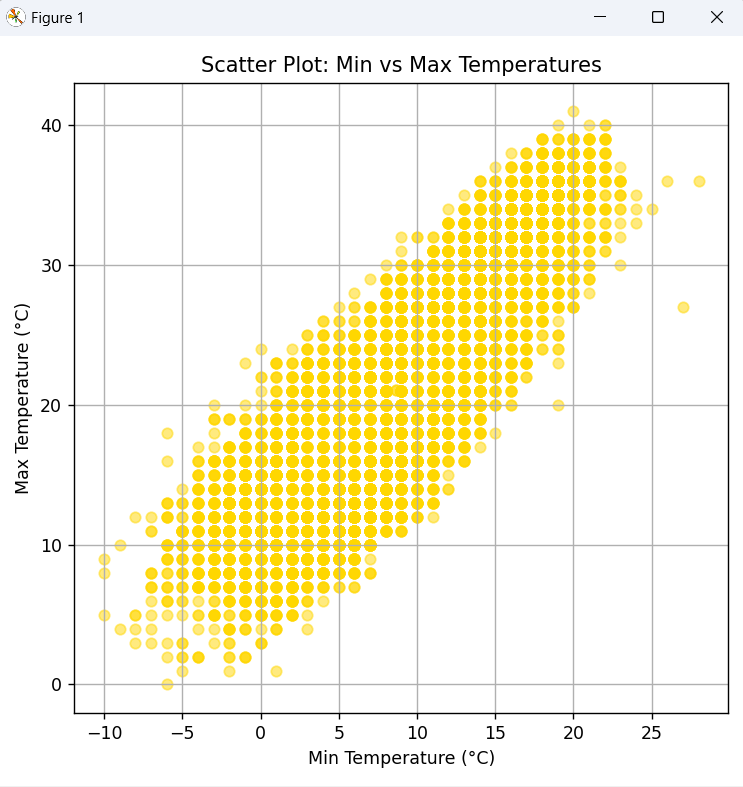
**iii. Functions and Formulas**

* Used matplotlib scatter() plot

**iv. Analysis Results**

Strong linear pattern, consistent min-max temp difference across dataset.

**v. Visualization**



**4.5 Pairwise Temperature Comparison**

**i. Introduction**

Pair plot allows visual inspection of relationships among all temperature types.

**ii. General Description**

Useful in detecting clusters, linearity, and distributions.

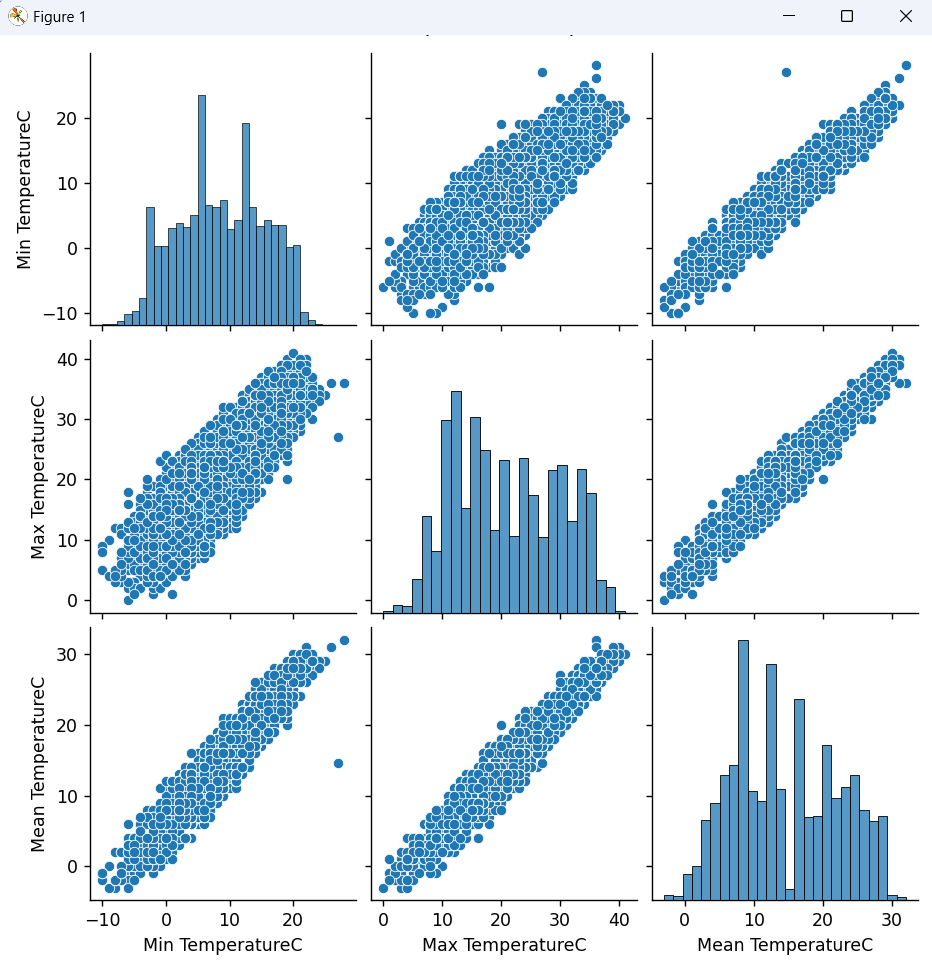
**iii. Functions and Formulas**

* Used seaborn pairplot()

**iv. Analysis Results**

Well-aligned pairwise scatter patterns confirm correlation.

**v. Visualization**



**4.6 Monthly Record Distribution**

**i. Introduction**

Pie chart visualization of dataset entry distribution across months.

**ii. General Description**

Ensures uniformity and helps validate data quality.

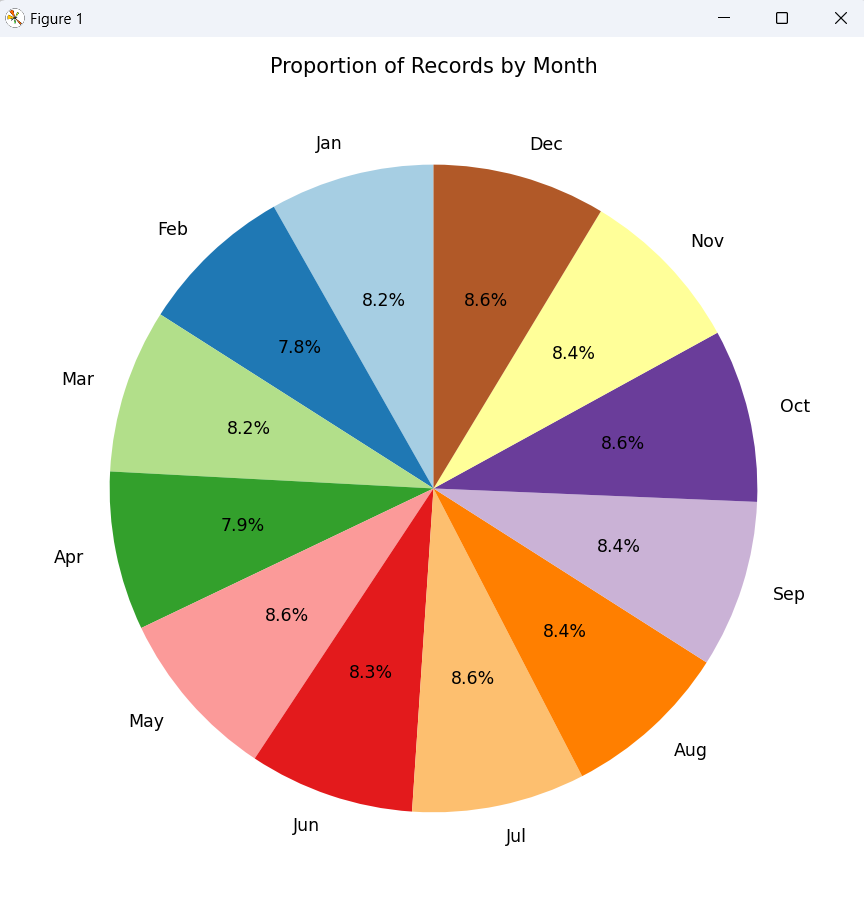
**iii. Functions and Formulas**

* Used value\_counts() on Month
* Used matplotlib pie chart

**iv. Analysis Results**

Fairly equal representation across months; reliable dataset.

**v. Visualization**



**4.7 Time Series Analysis**

**i. Introduction**

Temperature trend over the years.

**ii. General Description**

Helpful to analyze climate change, warming patterns.

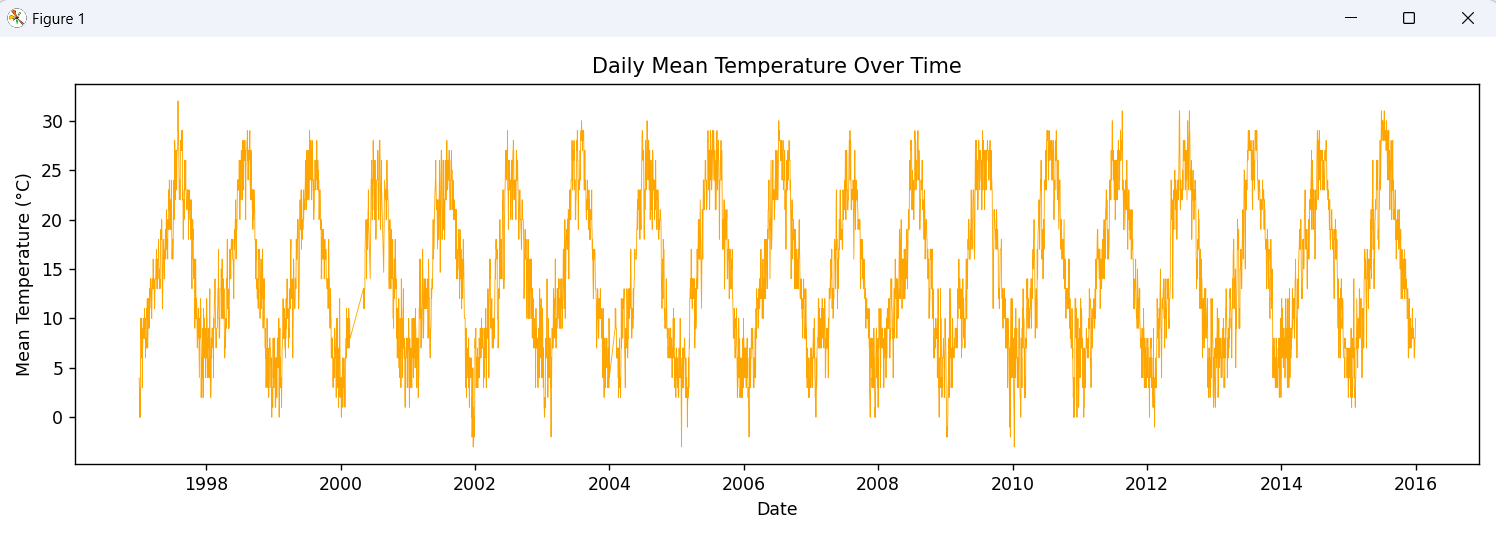
**iii. Functions and Formulas**

* Line plot of Date vs Mean Temperature

**iv. Analysis Results**

Seasonal cyclic pattern is evident, with minor year-to-year variations.

**v. Visualization**



**5. Predictive Modeling**

**5.1 Manual Linear Regression**

**i. Introduction**

Basic linear regression to predict Mean Temperature using Min and Max Temperatures.

**ii. General Description**

Implemented without sklearn to demonstrate understanding of regression math.

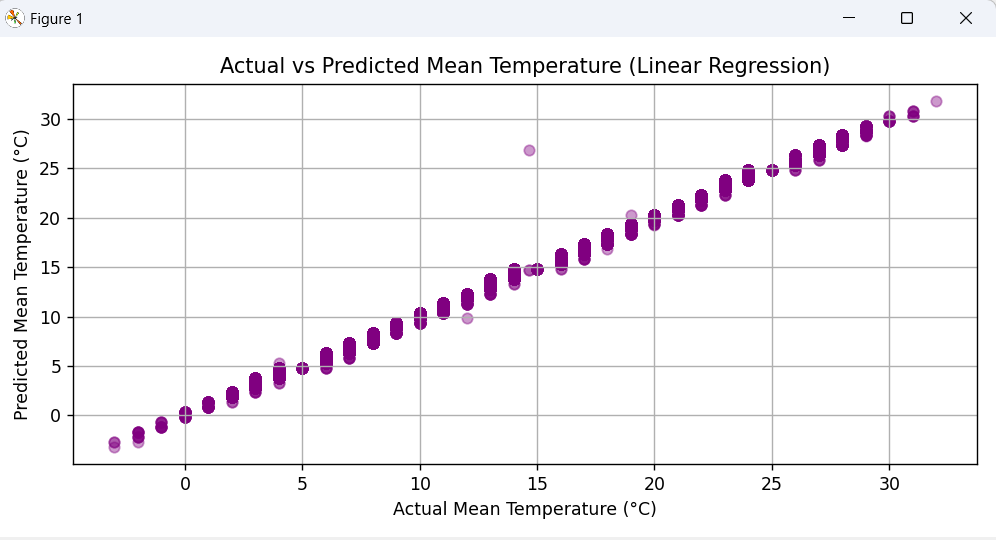
**iii. Functions and Formulas**

* Used Normal Equation:
* Calculated Mean Squared Error (MSE)

**iv. Analysis Results**

* Regression MSE: Low, indicating high prediction accuracy
* Prediction line fits closely with actual values

**v. Visualization**



**6. Conclusion**

This project demonstrated how to perform an end-to-end weather data analysis using Python. We began with data cleaning, then moved to exploratory and statistical analysis, followed by insightful visualizations and a predictive regression model. Key takeaways include:

* Strong seasonal temperature patterns
* High correlation between temperature measures
* Effective use of Python tools for EDA and modeling

**7. Future Scope**

* Apply advanced machine learning models (e.g., Random Forest, SVR)
* Incorporate humidity, precipitation, and wind speed in modeling
* Deploy as a web application dashboard
* Perform anomaly detection or climate change study over decades

**8. References**

* Pandas Documentation
* [Matplotlib Documentation](https://matplotlib.org/)
* Seaborn Documentation
* [NumPy Documentation](https://numpy.org/)
* Dataset Source: Kaggle / Custom Archive

**9 Source code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# 1. Data Cleaning & Preprocessing

df = pd.read\_csv(r"C:\\Users\\heman\\Downloads\\Madrid+Daily+Weather+1997-2015.csv\\Madrid Daily Weather 1997-2015.csv")

df.columns = df.columns.str.strip()

df['CET'] = pd.to\_datetime(df['CET'], errors='coerce')

df.dropna(subset=['CET'], inplace=True)

df.fillna(df.mean(numeric\_only=True), inplace=True)

df['Month'] = df['CET'].dt.month

df['Year'] = df['CET'].dt.year

df['Day'] = df['CET'].dt.day

# 2. Statistical Analysis

print("🔹 Dataset Info:")

print(df.info())

print("\n🔹 Summary Statistics:")

print(df.describe())

print("\n🔹 Missing Values Per Column:")

print(df.isnull().sum())

print("\n🔹 Correlation Matrix:")

print(df[['Min TemperatureC', 'Max TemperatureC', 'Mean TemperatureC']].corr())

# 3. Visualization & Insights

# Boxplot

plt.figure(figsize=(8, 4))

sns.boxplot(data=df[['Min TemperatureC', 'Max TemperatureC', 'Mean TemperatureC']])

plt.title("Distribution of Temperature Readings (Boxplot)")

plt.tight\_layout()

plt.show()

# Correlation Heatmap

plt.figure(figsize=(8, 5))

sns.heatmap(df[['Min TemperatureC', 'Max TemperatureC', 'Mean TemperatureC']].corr(), annot=True, cmap='coolwarm')

plt.title("Heatmap of Temperature Correlations")

plt.tight\_layout()

plt.show()

# Monthly Average Bar Chart

monthly\_avg = df.groupby('Month')['Mean TemperatureC'].mean()

plt.figure(figsize=(8, 4))

monthly\_avg.plot(kind='bar', color='skyblue')

plt.title('Average Monthly Mean Temperature (°C)')

plt.xticks(ticks=range(12), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',

'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'], rotation=45)

plt.ylabel("Mean Temperature (°C)")

plt.tight\_layout()

plt.show()

# Scatter: Min vs Max

plt.figure(figsize=(6, 6))

plt.scatter(df['Min TemperatureC'], df['Max TemperatureC'], alpha=0.5, color='gold')

plt.title("Scatter Plot: Min vs Max Temperatures")

plt.xlabel("Min Temperature (°C)")

plt.ylabel("Max Temperature (°C)")

plt.grid(True)

plt.tight\_layout()

plt.show()

# Pair Plot

sns.pairplot(df[['Min TemperatureC', 'Max TemperatureC', 'Mean TemperatureC']])

plt.suptitle("Pairwise Temperature Comparison", y=1.02)

plt.show()

# Pie Chart: Month Distribution

month\_counts = df['Month'].value\_counts().sort\_index()

labels = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',

'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

plt.figure(figsize=(7, 7))

plt.pie(month\_counts, labels=labels, autopct='%1.1f%%', startangle=90, colors=plt.cm.Paired.colors)

plt.title("Proportion of Records by Month")

plt.tight\_layout()

plt.show()

# 4. Time Series Analysis

df.sort\_values(by='CET', inplace=True)

# Line chart: Temperature over time

plt.figure(figsize=(12, 4))

plt.plot(df['CET'], df['Mean TemperatureC'], color='orange', linewidth=0.5)

plt.title("Daily Mean Temperature Over Time")

plt.xlabel("Date")

plt.ylabel("Mean Temperature (°C)")

plt.tight\_layout()

plt.show()

# 5. Predictive Modeling (Manual Linear Regression)

X = df[['Min TemperatureC', 'Max TemperatureC']].values

y = df['Mean TemperatureC'].values

X\_b = np.c\_[np.ones((X.shape[0], 1)), X] # Add intercept

theta\_best = np.linalg.inv(X\_b.T @ X\_b) @ X\_b.T @ y

y\_pred = X\_b @ theta\_best

mse = np.mean((y - y\_pred) \*\* 2)

print(f"\n📈 Manual Linear Regression MSE: {mse:.2f}")

# Plot actual vs predicted

plt.figure(figsize=(8, 4))

plt.scatter(y, y\_pred, color='purple', alpha=0.4)

plt.title("Actual vs Predicted Mean Temperature (Linear Regression)")

plt.xlabel("Actual Mean Temperature (°C)")

plt.ylabel("Predicted Mean Temperature (°C)")

plt.grid(True)

plt.tight\_layout()

plt.show()