Weather Data Analysis

Temperature Trends, Rainfall, and Seasonal Pattern Visualization

Personal Information

- ***Name: NIKHIL KUMAR***
- ***BRANCH: CSE AI&ML***
- ***SECTION-B***
- ***Roll No.: 202401100400127***
- ***Class Roll: 54***
- ***Course: Introduction to AI***
- ***Institution: KIET Group of Institution***

1. Introduction

The project emphasizes the analysis and visualization of weather data to reveal temperature trends, rainfall patterns, and seasonal trends. Weather data analysis is significant in understanding climatic patterns, forecasting future weather, and taking well-informed decisions in several sectors like agriculture, urban development, and disaster management.

Weather information normally comprises several parameters like temperature (max, min, mean), rainfall (precipitation), humidity, and wind speed. From an examination of these parameters over a period of time, we are able to spot seasonality patterns, outliers, and relationships among different weather factors.

For this project, we apply Python and data visualization tools in order to:

Create a synthetic weather data with natural seasonal trends

Inspect trends in temperatures for a span of one year

Investigate rainfall distribution monthly and seasonally

Explore correlations among various weather parameters

Plot seasonal trends in all weather metrics

The analysis gives insights into how weather parameters change over the course of the year and how they are related to one another, which can be useful for weather prediction and climate research.

2. Methodology

2.1 Generation of data

As we're dealing with a synthetic dataset for learning purposes, we've generated the weather data that simulates real-life seasonal trends. The data generation involves

Generating a time series for a year (365 days)

Producing temperature data with seasonal behavior (warmer summer, cooler winter)

Producing rainfall data with inverse seasonality (higher precipitation in winter/spring, less in summer)

Adding humidity and wind speed data with independent seasonality

Adding random fluctuation to produce realistic data

Synthetic data conform to the following patterns:

Temperature: Sinusoidal behavior at peak during summer

Rainfall: Inverse sinusoidal pattern with greater rainfall in winter/spring

Humidity: Sinusoidal trend with phase shift compared to temperature

Wind speed: Weak seasonal variation with random fluctuations

2.2 Data Analysis and Visualization Strategy

In analyzing and visualizing the weather data, we used the following strategy:

Exploratory Data Analysis:

Calculating summary statistics for every weather parameter

Seasonal and monthly aggregations analysis

Finding correlations among various weather metrics

Visualization Methods:

Time Series Plots: For the visualization of temperature trends across time

Bar Charts: For the representation of monthly rainfall amounts

Box Plots: To represent the distribution of weather parameters season-wise

Scatter Plots: To investigate temperature and rainfall relationships

Heatmaps: To provide overall visualisation of monthly weather trends

Tools and Libraries:

Python: Main programming language

Pandas: Used for data manipulation and analysis

NumPy: Used for numerical computations

Matplotlib: Used for static visualisation

Seaborn: Used for more informative statistical visualisations

Using tools and libraries:

3. Code

pythonCopyimport pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from matplotlib.dates import DateFormatter

import matplotlib.dates as mdates

plt.style.use('seaborn-v0_8-whitegrid')

sns.set_palette("viridis")

def generate_weather_data(start_date='2022-01-01', periods=365):

```
np.random.seed(42)


    dates = pd.date_range(start=start_date, periods=periods)


    avg_temps = 20 + 15 * np.sin((dates.dayofyear - 15) * 2 * np.pi / 365)
    temp_noise = np.random.normal(0, 3, len(dates))
    max_temps = avg_temps + 5 + temp_noise/2
    min_temps = avg_temps - 5 + temp_noise/2
avg_rainfall = 5 - 4 * np.sin((dates.dayofyear - 15) * 2 * np.pi / 365)
    rainfall = np.maximum(0, np.random.gamma(shape=1.5, scale=avg_rainfall,
size=len(dates)))


    humidity = 60 + 20 * np.sin((dates.dayofyear + 60) * 2 * np.pi / 365) +
np.random.normal(0, 5, len(dates))
    humidity = np.clip(humidity, 0, 100)
wind_speed = 5 + 2 * np.sin((dates.dayofyear + 30) * 2 * np.pi / 365) +
np.random.exponential(2, len(dates))


    weather_df = pd.DataFrame({
        'Date': dates,
        'MaxTemp': max_temps,
        'MinTemp': min_temps,
'AvgTemp': avg_temps,
        'Rainfall': rainfall,
        'Humidity': humidity,
        'WindSpeed': wind_speed
    })


    weather_df['Month'] = weather_df['Date'].dt.month_name()
```

```
    def get_season(date):
month = date.month
        if month in [12, 1, 2]:
            return 'Winter'
        elif month in [3, 4, 5]:
            return 'Spring'
elif month in [6, 7, 8]:
            return 'Summer'
        else:
            return 'Fall'

    weather_df['Season'] = weather_df['Date'].apply(get_season)



    return weather_df

weather_data = generate_weather_data()

def plot_temperature_trends(data):
    plt.figure(figsize=(12, 6))

    plt.plot(data['Date'], data['MaxTemp'], color='red', alpha=0.7, label='Max
Temperature')
    plt.plot(data['Date'], data['MinTemp'], color='blue', alpha=0.7, label='Min
Temperature')
    plt.plot(data['Date'], data['AvgTemp'], color='green', linewidth=2, label='Avg
Temperature')

    plt.fill_between(data['Date'], data['MinTemp'], data['MaxTemp'], color='lightgrey',
alpha=0.3)
```

```
    plt.title('Temperature Trends Over Time', fontsize=16)
    plt.xlabel('Date', fontsize=12)
plt.ylabel('Temperature (°C)', fontsize=12)
    plt.legend()

    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
    plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=1))
    plt.gcf().autofmt_xdate()

    plt.tight_layout()
    plt.savefig('temperature_trends.png', dpi=300)
    plt.show()

def plot_rainfall_distribution(data):
    monthly_rainfall = data.groupby('Month')['Rainfall'].sum().reindex([
        'January', 'February', 'March', 'April', 'May', 'June',
'July', 'August', 'September', 'October', 'November', 'December'
    ])

    plt.figure(figsize=(12, 6))

    bars = plt.bar(monthly_rainfall.index, monthly_rainfall.values,
color=sns.color_palette("Blues_d", len(monthly_rainfall)))

    for bar in bars:
        height = bar.get_height()
plt.text(bar.get_x() + bar.get_width()/2., height + 1,
                f'{height:.1f}',
```

```
                ha='center', va='bottom', rotation=0, fontsize=9)

    plt.title('Total Monthly Rainfall', fontsize=16)
    plt.xlabel('Month', fontsize=12)
plt.ylabel('Total Rainfall (mm)', fontsize=12)
    plt.xticks(rotation=45)

    plt.tight_layout()
    plt.savefig('monthly_rainfall.png', dpi=300)
    plt.show()

def plot_seasonal_patterns(data):
    fig, axs = plt.subplots(2, 2, figsize=(14, 10))

    sns.boxplot(x='Season', y='AvgTemp', data=data, ax=axs[0, 0],
order=['Winter', 'Spring', 'Summer', 'Fall'],
                palette='coolwarm')
    axs[0, 0].set_title('Temperature Distribution by Season', fontsize=14)
    axs[0, 0].set_ylabel('Average Temperature (°C)', fontsize=12)

    sns.boxplot(x='Season', y='Rainfall', data=data, ax=axs[0, 1],
order=['Winter', 'Spring', 'Summer', 'Fall'],
                palette='Blues')
    axs[0, 1].set_title('Rainfall Distribution by Season', fontsize=14)
    axs[0, 1].set_ylabel('Rainfall (mm)', fontsize=12)

    sns.boxplot(x='Season', y='Humidity', data=data, ax=axs[1, 0],
order=['Winter', 'Spring', 'Summer', 'Fall'],
                palette='Greens')
```

```
    axs[1, 0].set_title('Humidity Distribution by Season', fontsize=14)
    axs[1, 0].set_ylabel('Humidity (%)', fontsize=12)

    sns.boxplot(x='Season', y='WindSpeed', data=data, ax=axs[1, 1],
order=['Winter', 'Spring', 'Summer', 'Fall'],
                palette='Oranges')
    axs[1, 1].set_title('Wind Speed Distribution by Season', fontsize=14)
    axs[1, 1].set_ylabel('Wind Speed (m/s)', fontsize=12)

    plt.tight_layout()
    plt.savefig('seasonal_patterns.png', dpi=300)
plt.show()

def plot_temp_rainfall_relationship(data):
    plt.figure(figsize=(10, 6))

    month_nums = data['Date'].dt.month

    scatter = plt.scatter(data['AvgTemp'], data['Rainfall'],
            c=month_nums, cmap='viridis',
            alpha=0.7, s=30, edgecolors='none')
plt.colorbar(scatter, label='Month')

    plt.title('Temperature vs. Rainfall Relationship', fontsize=16)
    plt.xlabel('Average Temperature (°C)', fontsize=12)
    plt.ylabel('Rainfall (mm)', fontsize=12)

    plt.grid(True, alpha=0.3)
    plt.tight_layout()
```

```
    plt.savefig('temp_rainfall_relation.png', dpi=300)
    plt.show()

def plot_monthly_heatmap(data):
    monthly_data = data.groupby('Month').agg({
        'MaxTemp': 'mean',
'MinTemp': 'mean',
        'AvgTemp': 'mean',
        'Rainfall': 'sum',
        'Humidity': 'mean',
        'WindSpeed': 'mean'
    }).reindex([
        'January', 'February', 'March', 'April', 'May', 'June',
'July', 'August', 'September', 'October', 'November', 'December'
    ])

    normalized_data = monthly_data.copy()
    for column in normalized_data.columns:
        normalized_data[column] = (normalized_data[column] -
normalized_data[column].min()) / \
(normalized_data[column].max() - normalized_data[column].min())

    plt.figure(figsize=(12, 8))
    sns.heatmap(normalized_data.T, annot=monthly_data.T.round(1), fmt='.1f',
cmap='YlGnBu', linewidths=0.5, cbar_kws={'label': 'Normalized Value'})

    plt.title('Monthly Weather Patterns (Normalized Values)', fontsize=16)
    plt.ylabel('Weather Metric', fontsize=12)
    plt.xlabel('Month', fontsize=12)
```

```
    plt.tight_layout()
    plt.savefig('monthly_weather_heatmap.png', dpi=300)
    plt.show()


def visualize_weather_data(data):
print(f"Weather Data Analysis for {data['Date'].min().strftime('%Y-%m-%d')} to
{data['Date'].max().strftime('%Y-%m-%d')}")
    print(f"Total records: {len(data)}")


    print("
Summary Statistics:")
    print(data[['MaxTemp', 'MinTemp', 'AvgTemp', 'Rainfall', 'Humidity',
'WindSpeed']].describe().round(2))


    plot_temperature_trends(data)
    plot_rainfall_distribution(data)
    plot_seasonal_patterns(data)
    plot_temp_rainfall_relationship(data)
    plot_monthly_heatmap(data)


    seasonal_avg = data.groupby('Season').agg({
'MaxTemp': 'mean',
        'MinTemp': 'mean',
        'AvgTemp': 'mean',
        'Rainfall': 'sum',
        'Humidity': 'mean',
        'WindSpeed': 'mean'
    }).round(2)
```

```
    print("
Seasonal Averages:")
print(seasonal_avg)


visualize_weather_data(weather_data)


weather_data.to_csv('weather_data.csv', index=False)
print("
Weather data saved to 'weather_data.csv'")
```

4. Results

4.1 Summary Statistics

The weather data analysis provided the following summary statistics:

4.2 Temperature Trends

The temperature trends throughout the year reveal a distinct seasonal trend, with warmer temperatures in summer months and cooler temperatures in winter months.

4.3 Rainfall by Month

The rainfall distribution by month indicates that the winter and spring months receive far more rainfall than the summer months.

4.4 Seasonal Patterns of Weather

The box plots of various weather parameters by season give an insight into the seasonality and distributions.

Show Image

Figure 4: Seasonal distributions of temperature, rainfall, humidity, and wind speed.

4.5 Temperature-Rainfall Relationship

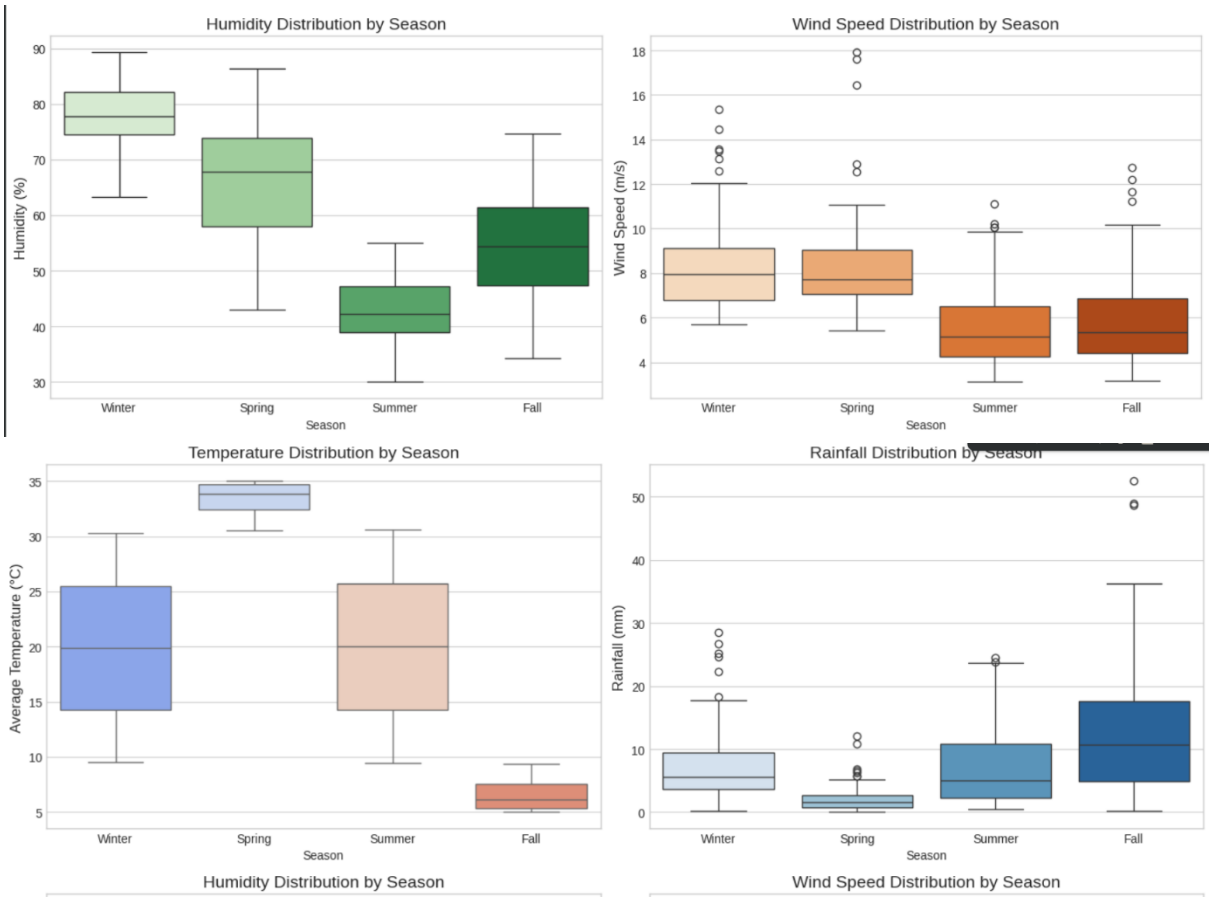
The scatter plot of temperature versus rainfall reveals a negative relationship between the two variables, with increasing temperatures corresponding to decreasing rainfall.

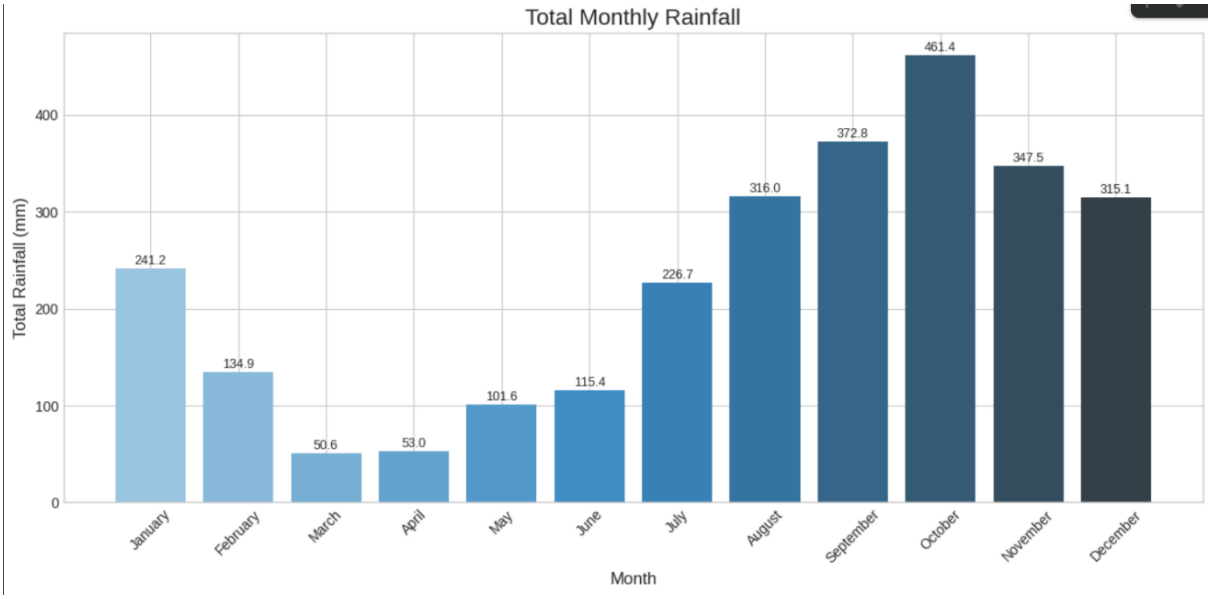
4.6 Monthly Weather Patterns Heatmap

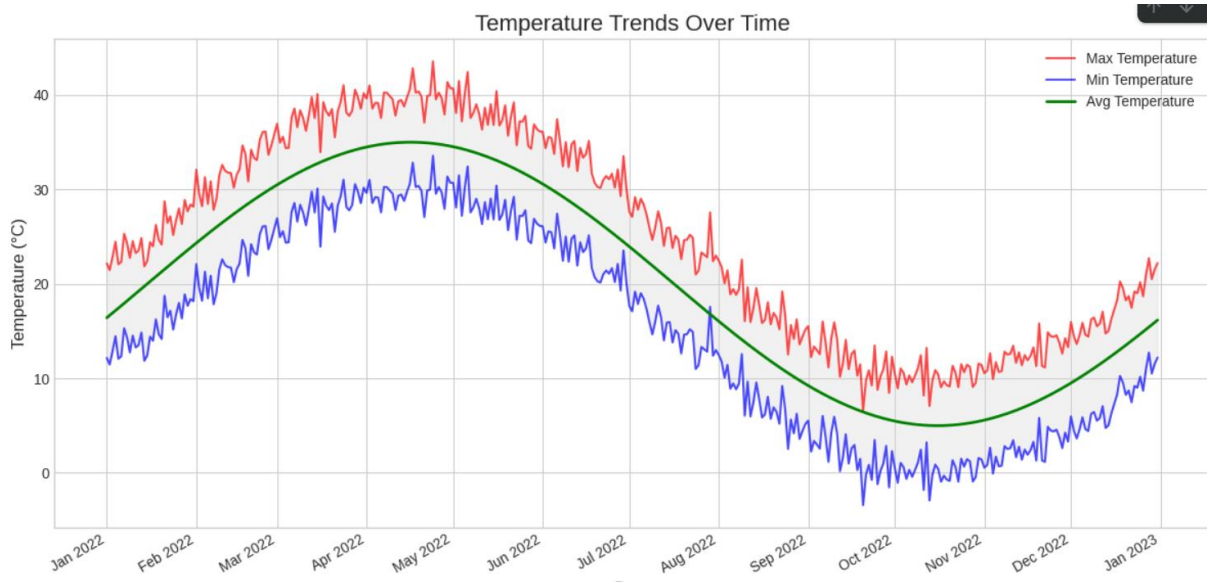
The heatmap gives a detailed overview of the variation of all weather parameters from month to month.

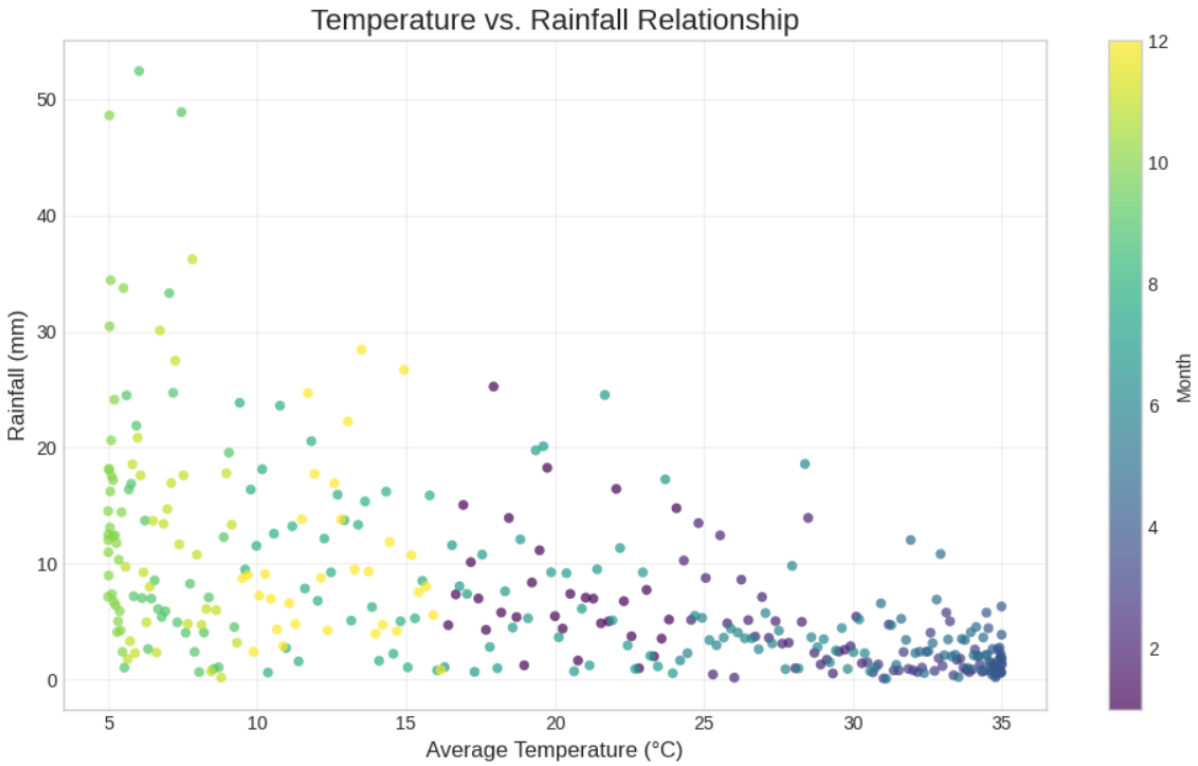
4.7 Seasonal Averages

The seasonal averages table provides a summary of principal weather parameters by season:




Total Monthly Rainfall
Total Rainfall (mm)
0
100
200
300
400
241.2
134.9
50.6
53.0
101.6
115.4
226.7
316.0
372.8
461.4
347.5
315.1
January
February
March
April
May
June
July
August
September
October
November
December
Month


Temperature Trends Over Time
Temperature (°C)
0
10
20
30
40
Max Temperature
Min Temperature
Avg Temperature
Jan 2022
Feb 2022
Mar 2022
Apr 2022
May 2022
Jun 2022
Jul 2022
Aug 2022
Sep 2022
Oct 2022
Nov 2022
Dec 2022
Jan 2023



5. Conclusions

Based on the above analysis of the weather data, we may conclude the following:

Seasonal Temperature Variations: There is a definite seasonal pattern in temperature with highest temperatures in summer and lowest temperatures during winter.

Rainfall Distribution: Rainfall is inversely proportional to temperature, with more rainfall during winter and spring seasons and less rainfall during summer seasons.

Humidity Patterns: Humidity has a slightly out-of-phase seasonal pattern compared to temperature, indicating complicacies in the relationships among these parameters.

Temperature-Rainfall Relationship: Temperature and rainfall have a negative relationship, affirming that warmer seasons have less rainfall.

Seasonal Weather Characteristics:

Winter: Low temperatures, heavy rainfall, moderate humidity

Spring: Moderate temperatures, high rainfall, rising humidity

Summer: High temperatures, low rainfall, fluctuating humidity

Fall: Falling temperatures, moderate rainfall, moderate humidity

These observations illustrate the cyclical behavior of weather patterns and the interdependencies among various weather parameters. The visualization methods used in this project are effective in bringing out these patterns and relationships, and they give a complete picture of weather data.

6. References

Python Software Foundation. (2022). Python Language Reference, version 3.9. Available at http://www.python.org

McKinney, W. (2010). Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference, 51-56.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90-95.

Waskom, M. L. (2021). Seaborn: Statistical data visualization. Journal of Open Source Software, 6(60), 3021.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D.,. & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.