**Abstract**

The document presents an analysis of a weather dataset, focusing on understanding and interpreting various weather attributes.

The primary objectives include importing and exploring the dataset, performing sanity checks, and conducting Exploratory Data Analysis (EDA). The analysis encompasses handling missing values, outliers, and data cleaning and preprocessing.

Regression analysis is performed to assess relationships between weather features and target variables.

The findings are visualized using scatter plots and histograms in Python, and further insights are generated through Power BI visualizations.

This comprehensive analysis provides valuable insights into weather patterns and their interactions, supporting informed decision-making and further research.

**Source : kaggle.com**

**Table of Contents**

|  |  |
| --- | --- |
| **1**. | **Introduction**  - Objective  - Dataset Overview  - About the Dataset  - Input Variables |
| **2.** | **Data Import and Exploration**  - Import Libraries  - Read Dataset  - Head and Tail |
| **3**. | **Sanity Check of Data**  - Naming Columns as Given in Dataset  - Removing Date Column  - Shape  - Info |
| **4.** | **Finding Missing Values**  - Missing Values  - Percentage of Missing Values |
| **5.** | **Exploratory Data Analysis (EDA)**  - Descriptive Statistics for Numeric Values  - Descriptive Statistics for Object Values  - Unique Values  - Boxplot to Identify Outliers |
| **6.** | **Data cleaning and Preprocessing**  - Missing Value Treatment  - Outlier Treatment  - Duplicate Value Treatment  - Encoding  - Normalization |
| **7.** | **Correlation**  - Heatmap |
| **8**. | **Regression Analysis**  - Import Libraries  - Choose Target and Features  - Split the data  - Train the Regression  - Evaluate the Model |
| **9**. | **Visualization**  - Scatter Plot  - Histogram |
| **10.** | **Visualization of the Weather Dataset**  - Using Power BI |
| **11.** | **Conclusion** |

***1.Introduction***

**1.1 Objective**

The objective of this analysis is to understand and model weather patterns using the provided dataset. The aim to explore relationships between weather variables, clean and preprocess the data, and perform regression analysis to gain insights into how different weather attributes impact precipitation.

**1.2 Dataset Overview**

**1.2.1 About the Dataset**

The dataset includes various weather attributes recorded over a period. Specifically, it contains data on precipitation, temperature, wind speed, and weather conditions.

**1.2.2 Input Variables**

1. Date
2. Precipitation
3. Temp\_max(Maximum Temperature)
4. Temp\_min(Minimum Temperature)
5. Wind(Wind Speed)
6. Weather(Weather Conditions)
   * drizzle
   * fog
   * rain
   * snow
   * sun -> Number of varibles - 5 (Precipitation, Temp\_max, Temp\_min, Wind, Weather)

***2.Data Import and Exploration***

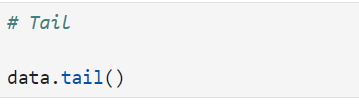
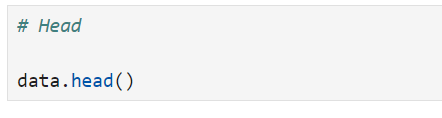
**2.1 Import Libraries**

1. ***Pandas*** - Provides data structures and functions for efficient data manipulation and analysis.
2. ***NumPy***- Supports numerical operations and efficient array handling.
3. ***Matplotlib.pyplot*** - Enables creation of static, interactive, and animated visualizations in Python
4. ***Seaborn***- Simplifies the creation of informative and attractive statistical graphics based on Matplotlib.

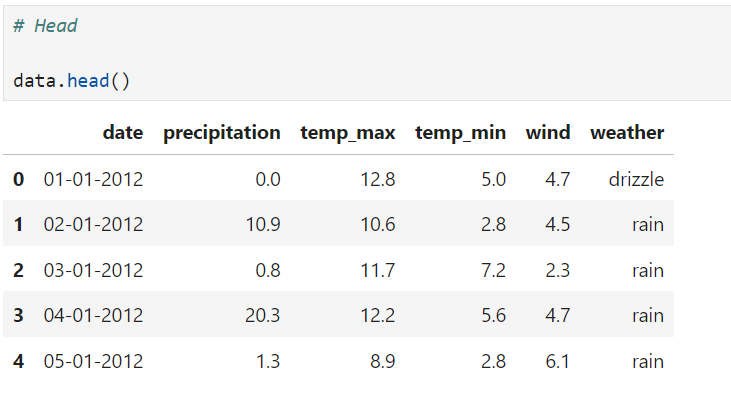
**2.2 Read Dataset**



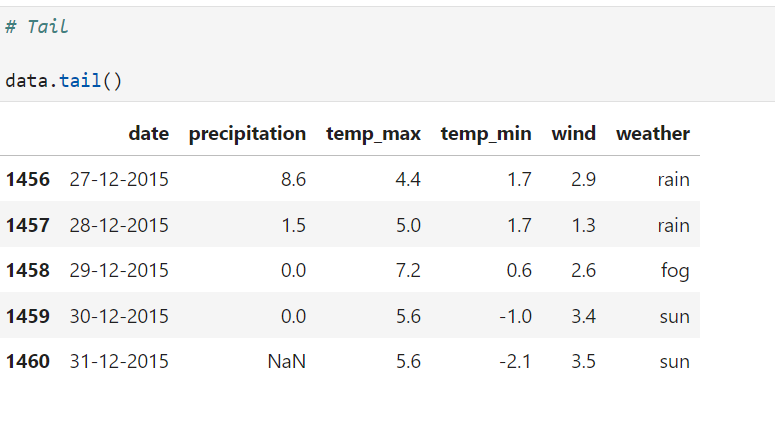
* 1. **Head and Tail**

****

***Head***- Shows the first few rows of the DataFrame.

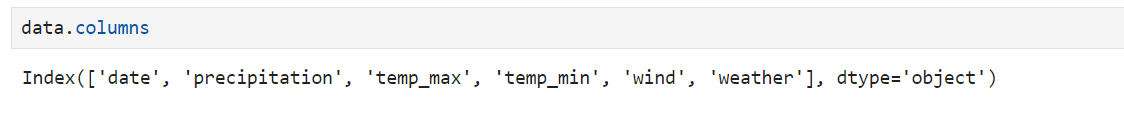


***Tail***- Displays the last few rows of the DataFrame.



***3.Sanity Check of Data***

* To ensure the dataset is accurate and reliable by identifying and addressing errors, inconsistencies, and anomalies before further analysis.

**3.1 Naming the Columns as given in dataset**

The data.columns output reveals that the dataset has 6 columns:

**1.date** --> The date of the weather observation.

**2.precipitation** --> The amount of precipitation recorded.

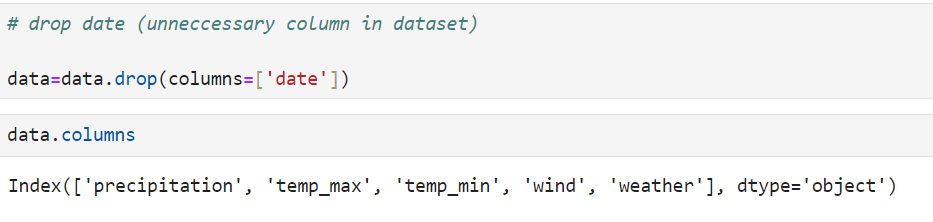
**3.temp\_max** --> The maximum temperature recorded.

**4.temp\_min** --> The minimum temperature recorded.

**5.wind** --> The wind speed recorded.

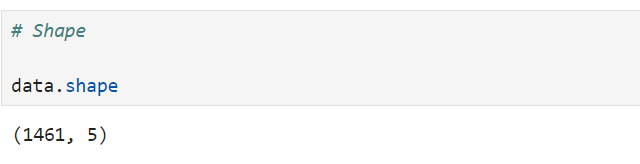
**6.weather**--> The weather condition(e.g., drizzle, fog, rain, snow, sun).

**3.2 Removing Date Column**

* To eliminate non-essential information that does not contribute to the analysis of weather conditions, simplifying the dataset for further processing and analysis.

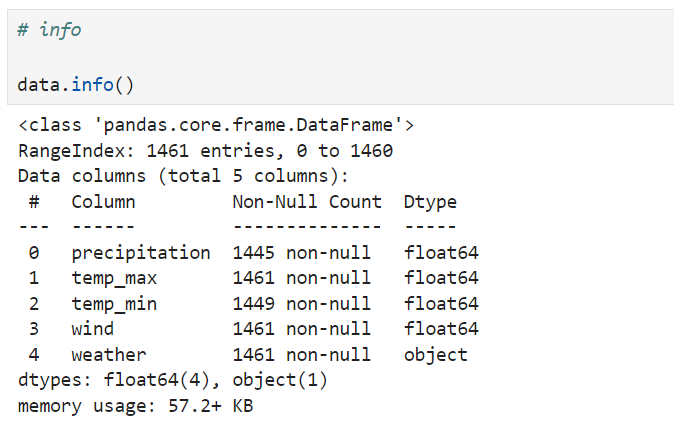
**3.3 Shape**

* To determine the dimensions of the dataset, including the number of rows and columns, which helps in understanding its structure and ensuring it aligns with expectations.

  
The data.shape output indicates that the dataset has 1461 rows and 5 columns.

**3.4 Info**

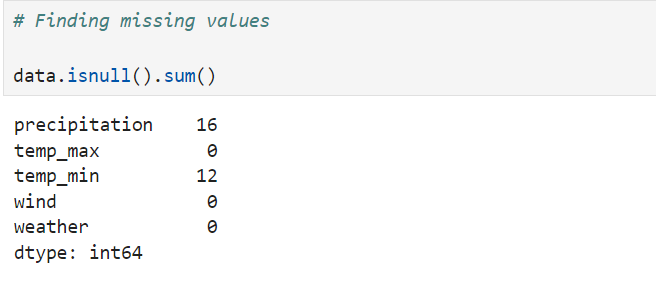
* To provide a summary of the dataset's structure, including data types, non-null counts, and column names, which helps in assessing data quality and preparing for analysis.



* The data.info() output reveals that the dataset is a Pandas DataFrame with 1461 entries, indexed from 0 to 1460.
* It contains 5 columns: precipitation, temp\_max, temp\_min, wind, and weather.
* The precipitation column has 1445 non-null entries, while temp\_max, wind, and weather columns have 1461 non-null entries. The temp\_min column has 1449 non-null entries.
* The first four columns (precipitation, temp\_max, temp\_min, wind) are of type float64, representing numerical data.
* The weather column is of type object, indicating categorical weather conditions.
* The dataset's memory usage is approximately 57.2 KB.

***4.Finding Missing Values***

**4.1 Missing values**

* ****To identify and assess missing data in the dataset, which is crucial for ensuring data completeness and accuracy before proceeding with analysis.

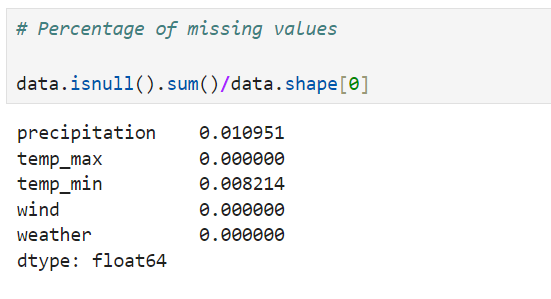
**Precipitation --> 16 missing values**

temp\_max --> No missing values

**temp\_min --> 12 missing values**

wind --> No missing values

weather --> No missing values

****The dataset has missing values in the **precipitation and temp\_min columns.**

**precipitation --> 1.10%**

temp\_max --> 0.00%

**temp\_min --> 0.82%**

wind --> 0.00%

weather --> 0.00%

The dataset has a small percentage of missing values in the **precipitation and temp\_min columns**.

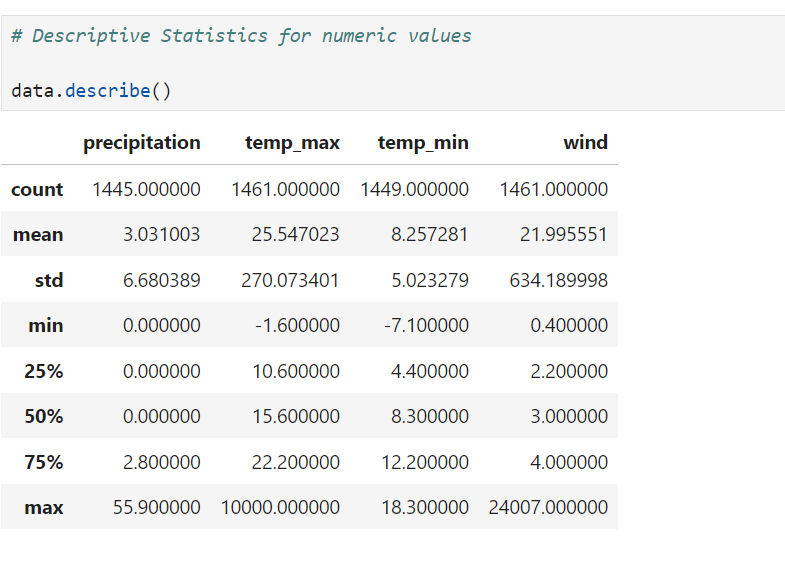
***5.Exploratory Data Analysis (EDA)***

* To perform a preliminary examination of the dataset, identifying key patterns and basic statistics to understand its structure and content.

## **Investigating the data**

**5.1 Descriptive Statistics for numeric values**

* Provides a summary of key statistical measures such as mean, median, standard deviation, and range for numeric data, helping to understand the distribution and variability of the dataset

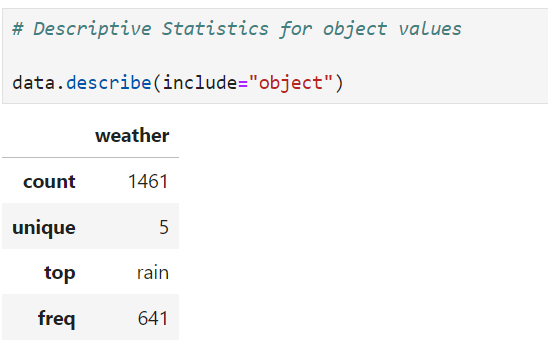


The data.describe() output provides statistical summary of the numerical columns in the dataset:

|  |  |
| --- | --- |
| precipitation:  Mean: 3.03  Std Dev: 6.68  Min: 0.00  Max: 55.90 | temp\_min:  Mean: 8.26  Std Dev: 5.02  Min: -7.10  Max: 18.30 |
| temp\_max:  Mean: 25.55  Std Dev: 270.07  Min: -1.60  Max: 10000.00 | wind:  Mean: 22.00  Std Dev: 634.19  Min: 0.40  Max: 24007.00 |

This summary provides an overview of the central tendency, spread, and range of the numerical variables in the dataset.

**5.2 Descriptive Statistics for object values**

* Provides insights into the frequency and distribution of categorical data, including counts and unique values, helping to understand the composition and diversity of the dataset.

The data.describe(include="object") output provides a summary of the categorical column Weather:

**Count**  --> 1461

**Unique Values** --> 5

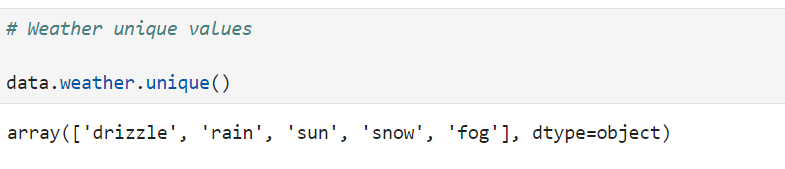
**Most Common Value (Top)** --> rain`

**Frequency of Most Common Value** --> 641

This summary provides an overview of the categorical distribution in the weather column, showing the most common weather condition and its frequency.

**5.2.1 Unique Values**

* To identify all distinct values in a categorical dataset, helps in understanding the diversity and variability of object features.



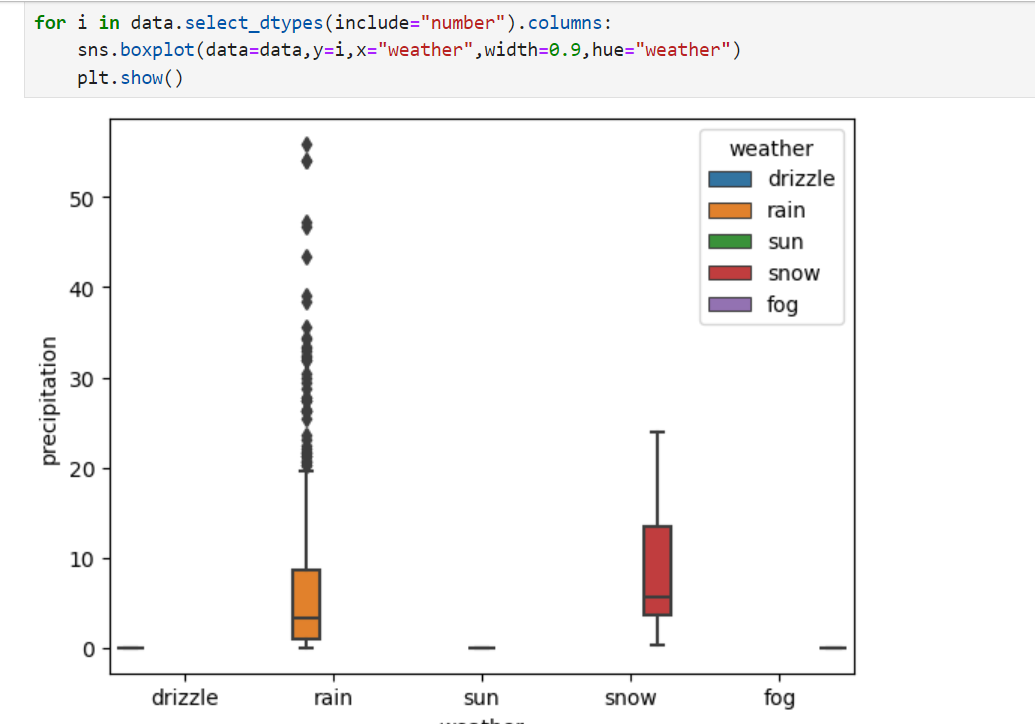
The weather column contains unique values:

* drizzle
* rain
* sun
* snow
* fog

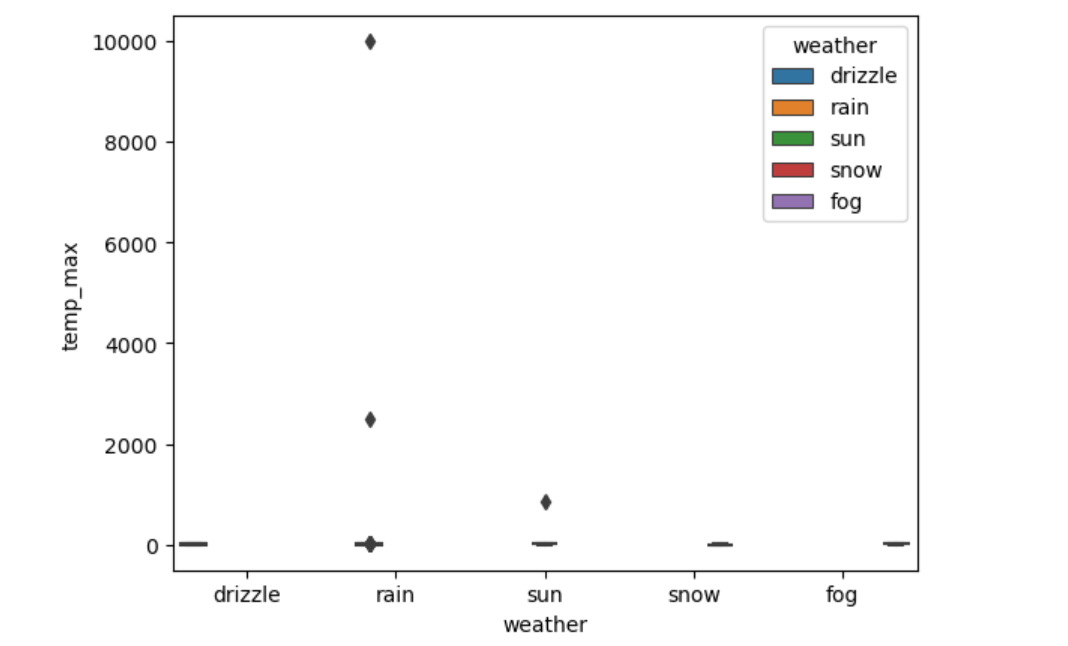
**5.3 Boxplot to Identify the Outliers**

* To visualize the distribution of data and detect outliers by showing the range, quartiles, and extreme values in the dataset.

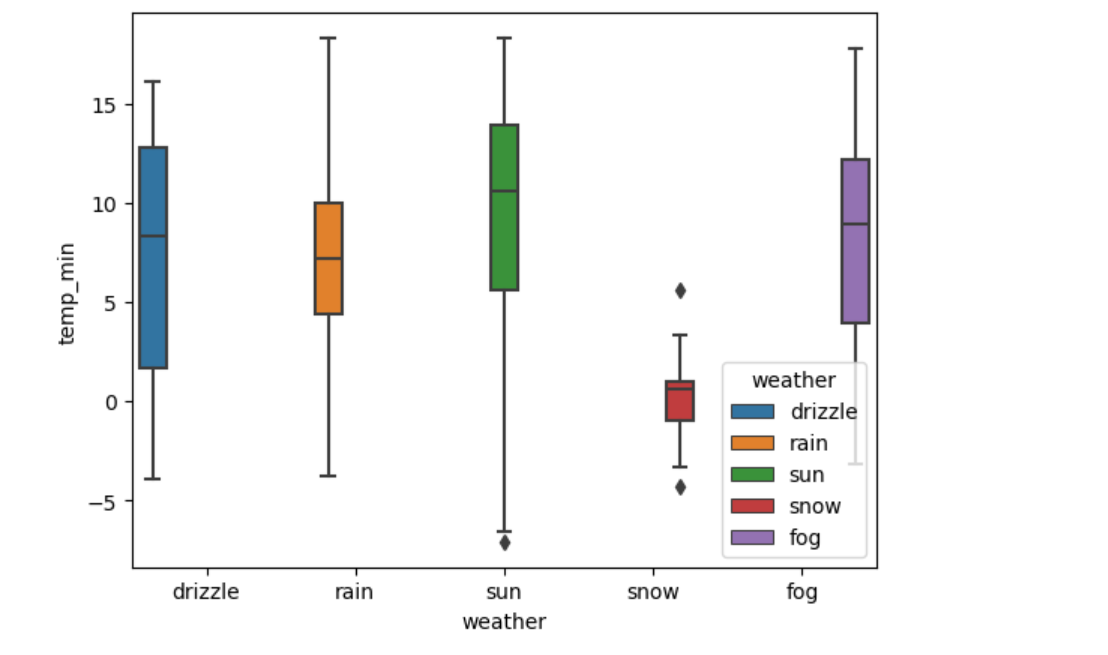
**Boxplot to Identify the Outliers(Precipitation)**

****

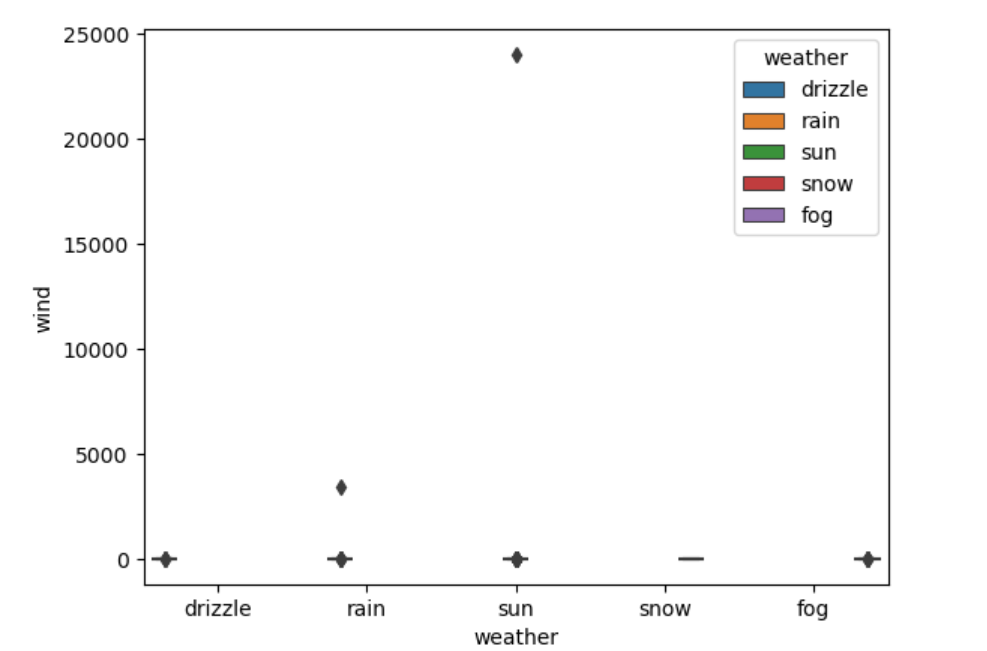
**Boxplot to Identify the Outliers(Temp\_max)**

****

**Boxplot to Identify the Outliers(Temp\_min)**

****

**Boxplot to Identify the Outliers(Wind)**

****

***6.Data Cleaning and preprocessing***

* To prepare the dataset for analysis by addressing issues such as missing values, outliers, and inconsistencies, ensuring data quality and accuracy.

**6.1 Missing Value Treatment**

* To address and handle missing data in the dataset by applying methods such as imputation or removal, ensuring completeness and reliability for accurate analysis.



The code fills missing values in the **precipitation and temp\_min** columns by replacing them with the median value of each column:

**Precipitation** --> Missing values are filled with the median value of the precipitation column.

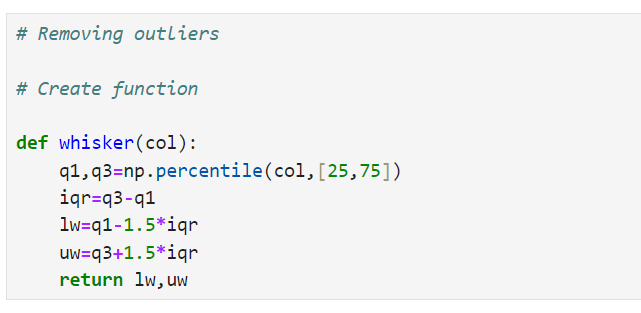
**temp\_min** --> Missing values are filled with the median value of the temp\_min column.

This approach helps maintain the central tendency of the data while

handling missing values.

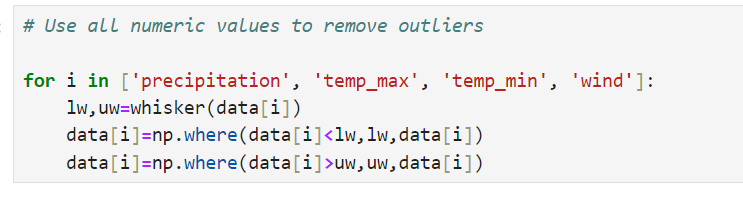
**6.2 Outlier Treatment**

* To identify and address outliers in the dataset by methods such as removal or transformation, ensuring that they do not skew or distort the analysis.



The code removes outliers from the numerical columns (precipitation, temp\_max, temp\_min, wind) using the interquartile range (IQR) method:

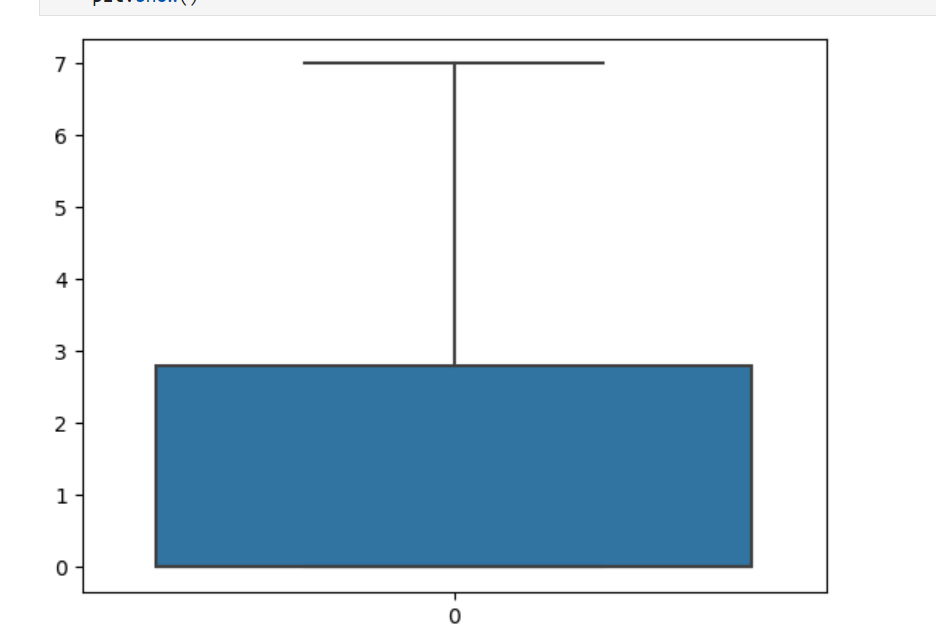
Define the whisker function: Calculates the lower and upper bounds for outliers using the IQR method.

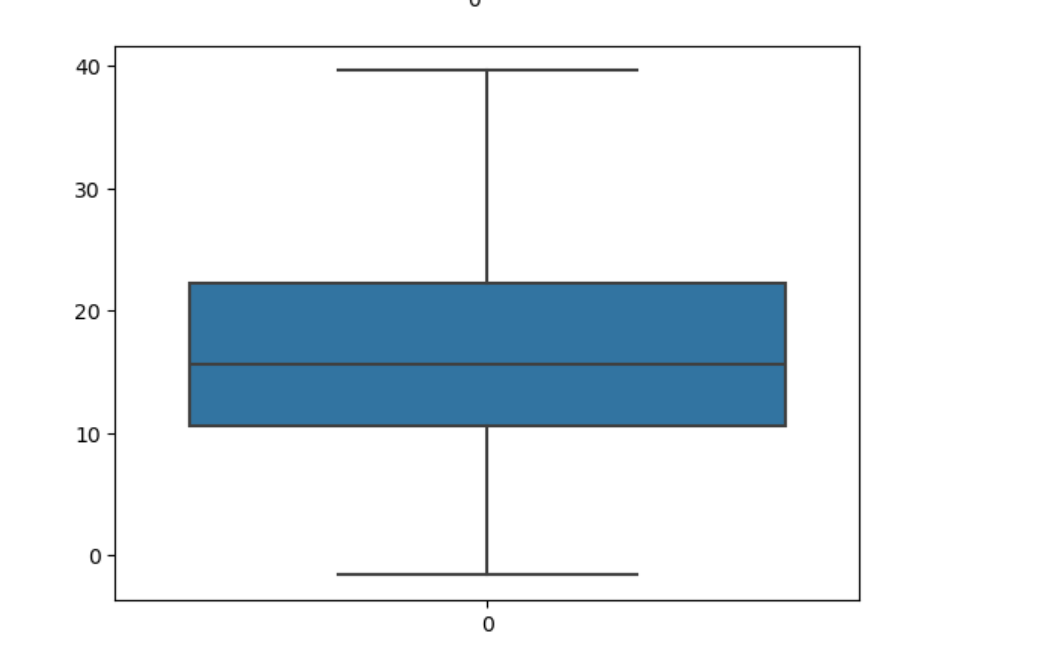


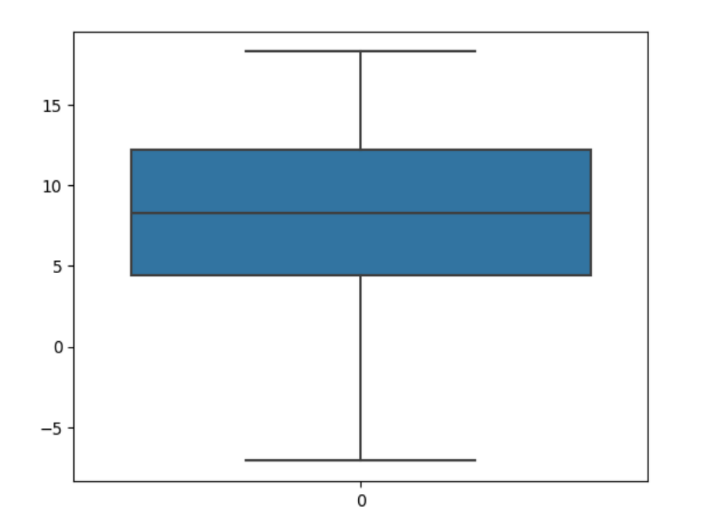
**Remove outliers:**

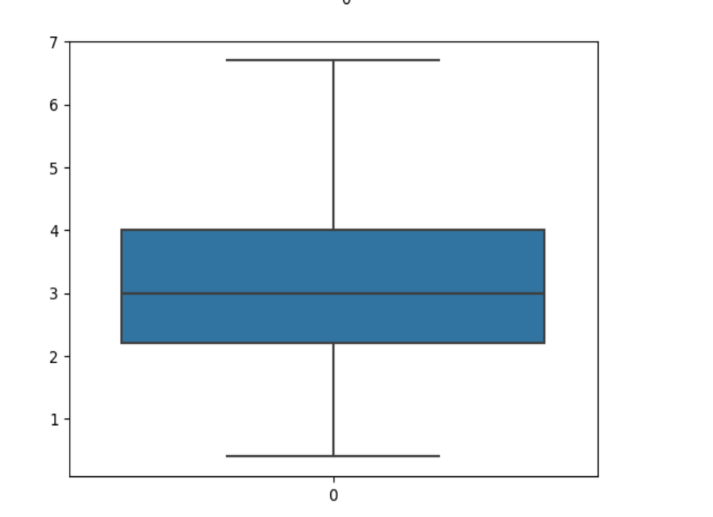
Apply the whisker function to each numeric column and replace values outside the bounds with the corresponding boundary value.



 **Precipitation** **Temp\_max**



**Temp\_min Wind**

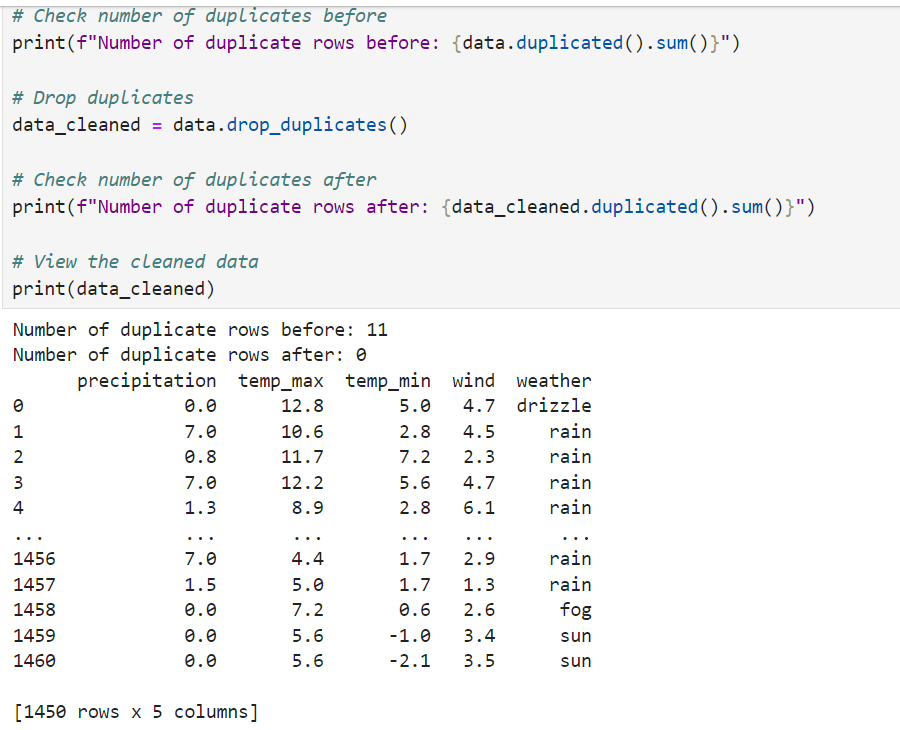


**Visualize the results:**

Use boxplots to check the distribution of data after outlier removal.

**6.3 Duplicate values Treatment**

* To identify and remove duplicate records in the dataset, ensuring data integrity and accuracy for reliable analysis.

****

**Checking Number of Duplicate Rows Before Removal**: The dataset initially had 11 duplicate rows

print(f"Number of duplicate rows before: {data.duplicated().sum()}")

**Removing Duplicate Rows**: Duplicate rows were removed from the dataset, resulting in a cleaned DataFrame

data\_cleaned = data.drop\_duplicates()

**Checking Number of Duplicate Rows After Removal:** After removing duplicates, the dataset no longer contains any duplicate rows:

print(f"Number of duplicate rows after: {data\_cleaned.duplicated().sum()}")

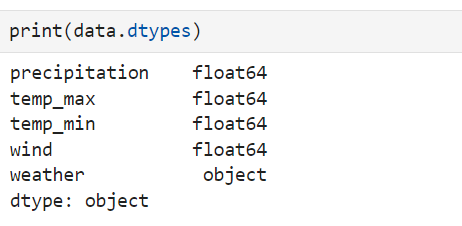
**Viewing the Cleaned Data:** The cleaned DataFrame, which now contains 1450 rows and 5 columns, is displayed:

print(data\_cleaned)

The process successfully removed all duplicate rows from the dataset, resulting in a clean dataset with 1450 unique rows. The dataset columns include precipitation, temp\_max, temp\_min, wind, and weather

**6.4 Encoding**

* To convert categorical data into numerical format, enabling the use of machine learning algorithms and statistical analysis that require numerical inputs.



The dataset contains the following columns with their respective data types:

**precipitation**  --> float64

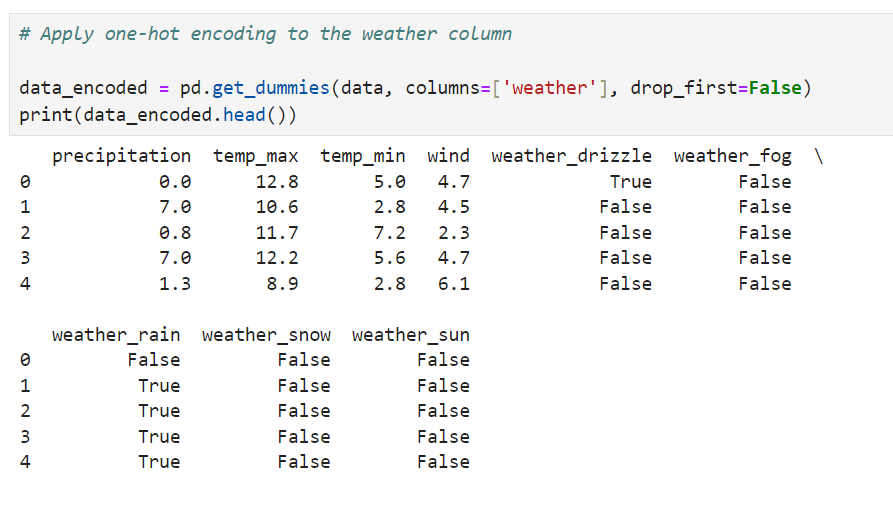
**temp\_max** --> float64

**temp\_min** -->float64

**wind**  --> float64

**weather**  -->object

The dataset consists of numerical columns (float64) for measurements and a categorical column (object) for weather conditions.



One-hot encoding is applied to the weather column to convert categorical values into binary (0 or 1) columns.

The resulting DataFrame, data\_encoded, includes the original numerical columns and new binary columns for each weather condition.

weather\_drizzle --> drizzle (1 for presence, 0 for absence).

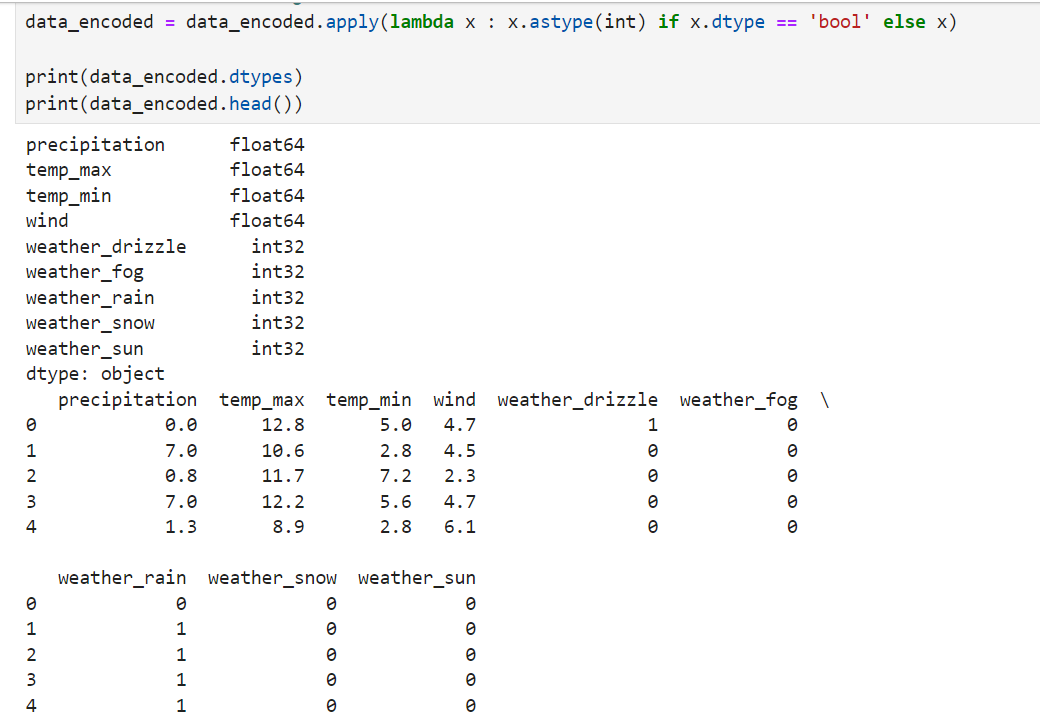
weather\_fog --> fog (1 for presence, 0 for absence).

weather\_rain --> rain (1 for presence, 0 for absence).

weather\_snow --> snow (1 for presence, 0 for absence).

weather\_sun --> sun (1 for presence, 0 for absence).

But it is present in Boolean values(False or True), have to change them to integers



The code converts boolean columns in the data\_encoded DataFrame to integer data types. This transformation is performed to prepare the data for machine learning algorithms, which typically require numerical input.

**Data Types After Conversion:**

|  |  |
| --- | --- |
| Precipitation --> float64  temp\_max --> float64  temp\_min --> float64  wind -->float64 | weather\_drizzle --> int32  weather\_fog --> int32  weather\_rain --> int32  weather\_snow --> int32  weather\_sun --> int32 |

Resulting DataFrame:

The DataFrame now includes integer columns for weather conditions:

weather\_drizzle: Integer column (0 or 1) for drizzle.

weather\_fog: Integer column (0 or 1) for fog.

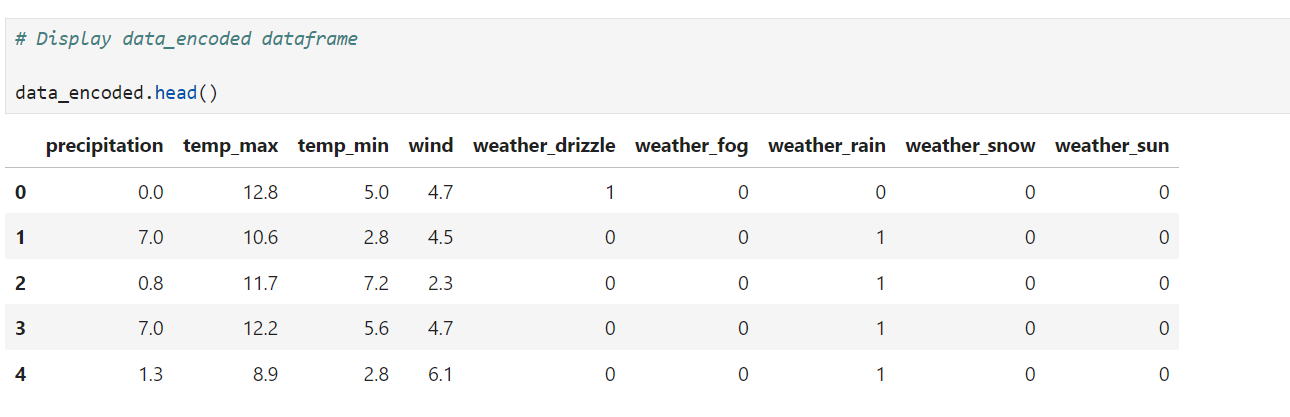
weather\_rain: Integer column (0 or 1) for rain.

weather\_snow: Integer column (0 or 1) for snow.

weather\_sun: Integer column (0 or 1) for sun.

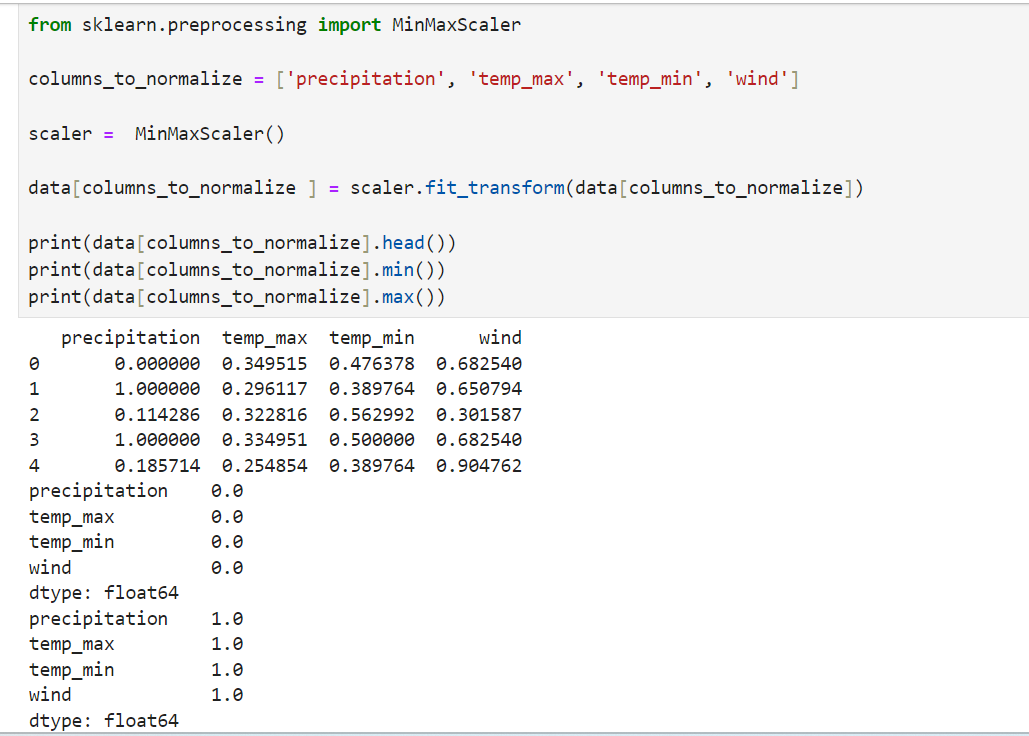
This conversion ensures that all columns are in a numerical format suitable for machine learning models.

**Displaying the first few rows of the DataFrame :**



**6.5 Normalization**

* To scale numeric data to a standard range, improving the performance and convergence of machine learning algorithms by ensuring that features contribute equally.

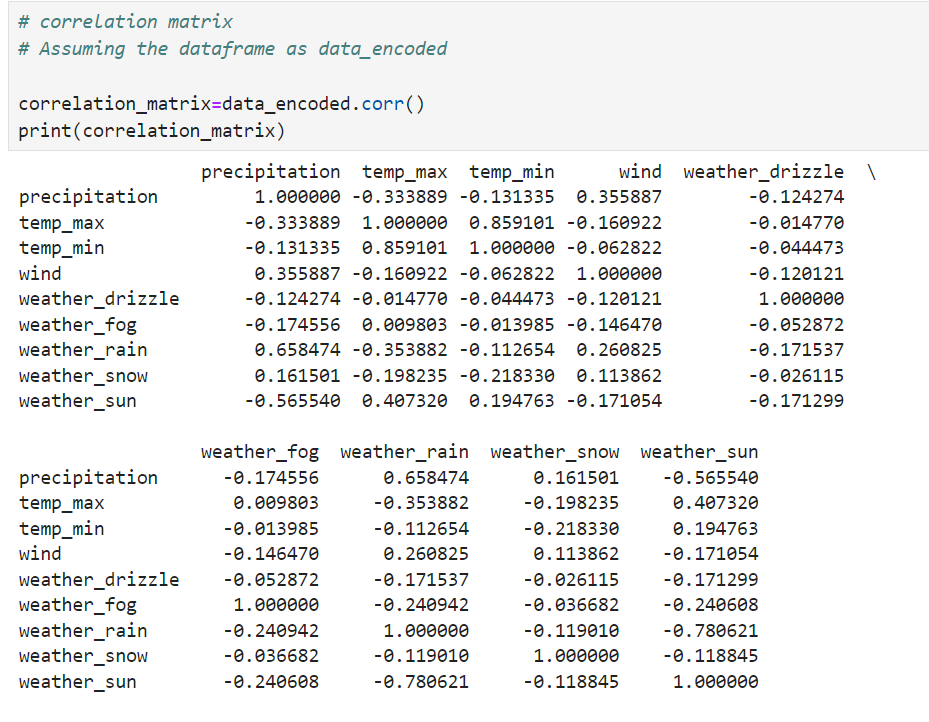


In the code, the sklearn library is used specifically for data normalization through Min-Max scaling:

1. ***Normalization***----> The MinMaxScaler from sklearn.preprocessing is used to normalize numerical features to a range between 0 and 1. This is essential for many machine learning algorithms, which perform better or converge faster when features are on a similar scale.
2. ***MinMaxScaler*** ----> Scales the specified columns (precipitation, temp\_max, temp\_min, wind) to a range between 0 and 1. This process ensures that all numerical features are on the same scale, which is crucial for improving the performance and convergence of many machine learning algorithms.

***7.Correlation***

* To measure and visualize the strength and direction of relationships between numeric variables, aiding in understanding how variables interact with one another.



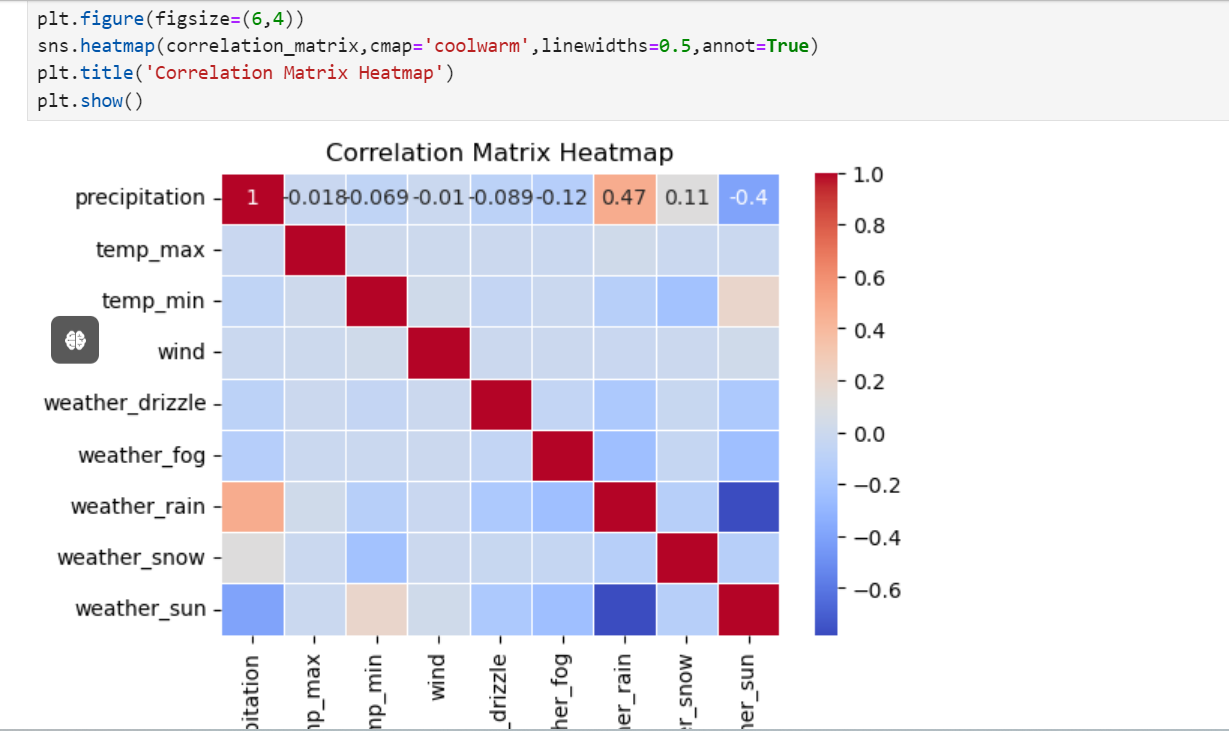
The correlation matrix shows the pairwise correlations between numerical features and encoded weather conditions in the dataset.

Correlation values range from -1 to 1, where 1 indicates a perfect positive correlation,

-1 indicates a perfect negative correlation, and

0 indicates no correlation.

**7.1 Heatmap**

* ****To visually represent the correlation matrix between numeric variables, making it easier to identify patterns and relationships through color-coded values.

The sns.heatmap() function with the annot=True parameter creates a heatmap of the correlation matrix s, where the correlation coefficients are displayed as annotations within the heatmap cells.

**Color Intensity** -----> The color intensity represents the strength of the correlation, with darker colors indicating stronger correlations.

**Annotations** -----> The numeric values inside the heatmap cells show the exact correlation coefficients between pairs of features.

**Correlation Insights** -----> This visualization provides a clear, visual representation of the correlations between numerical features, making it easier to identify strong or weak relationships.

This heatmap is useful for quickly assessing feature relationships and understanding how different variables are interrelated in the dataset.

**Interpretation**

**precipitation:** Shows a positive correlation with weather\_rain (0.658) and a negative correlation with weather\_sun (-0.566).

**temp\_max:** Highly positively correlated with temp\_min (0.859) and negatively correlated with weather\_rain (-0.354).

**temp\_min:** Positively correlated with temp\_max (0.859) and negatively correlated with weather\_rain (-0.113).

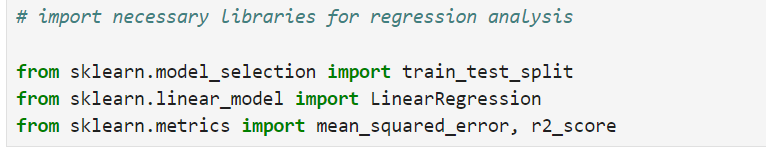
**wind:** Shows a positive correlation with weather\_rain (0.261) and a negative correlation with weather\_sun (-0.171).

**Encoded Weather Columns:** Exhibit varying degrees of correlation with numerical features, weather\_rain and weather\_sun showing strong correlations with other features.

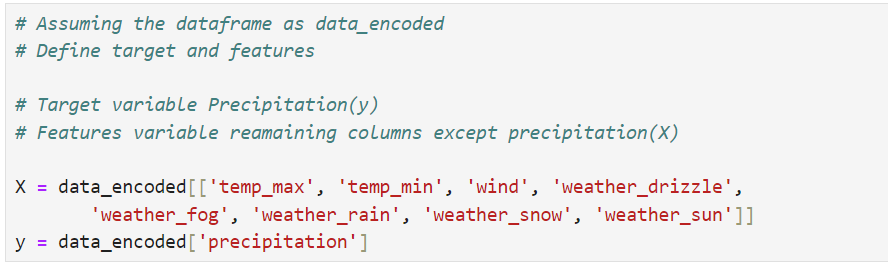
***8. Regression Analysis***

* To model and understand the relationship between dependent and independent variables, predicting outcomes and identifying trends based on the data.

**8.1 Import Libraries**

1. ***train\_test\_split*** - Splits the dataset into training and testing sets for model evaluation.
2. ***LinearRegression*** - Applies linear regression to model relationships between variables.
3. ***mean\_squared\_error, r2\_score*** - Evaluates model performance using prediction accuracy and variance explanation.

**8.2 Choosing Target and Features**

****

In the data\_encoded DataFrame:

Target Variable (y) --> precipitation

This is the variable, aim to predict or model.

Feature Variables (X):

These are the predictors or independent variables used to build the model. They include:

temp\_max

temp\_min

wind

weather\_drizzle

weather\_fog

weather\_rain

weather\_snow

weather\_sun

**X--->** Contains all the columns except precipitation. It includes numerical and one-hot encoded categorical features.

**y--->** Contains only the precipitation column, which is the target variable to predict.

This separation of features and target variable is crucial for training predictive models.

**8.3 Split the data**

The dataset was divided into training and testing sets using the train\_test\_split function from scikit-learn:

Training Set (X\_train, y\_train):

Contains 80% of the data used to train the model.

Testing Set (X\_test, y\_test):

Contains 20% of the data used to evaluate the model's performance.

X\_train --> Features for the training set.

X\_test--> Features for the testing set.

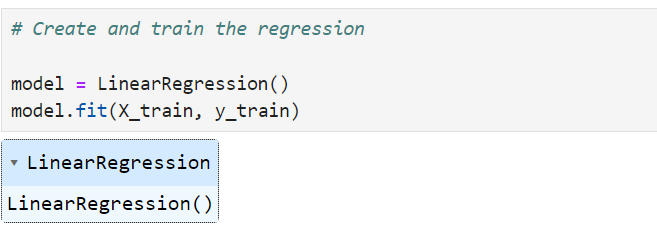
y\_train --> Target variable for the training set.

y\_test:] --> Target variable for the testing set.

test\_size=0.2 --> Specifies that 20% of the data is reserved for testing.

random\_state=42 --> Ensures reproducibility of the split by setting a fixed random seed.

**8.4 Train the model**



Model Creation and Training

A Linear Regression model was created and trained using the training data:

Model--> Linear Regression

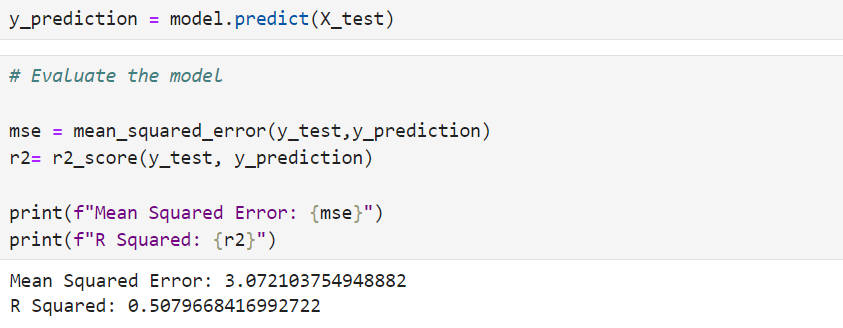
Training Process--> The model was trained on the training set (X\_train, y\_train).

LinearRegression()--> Initializes the Linear Regression model.

model.fit(X\_train, y\_train)--> Trains the model using the training feature set (X\_train) and target variable (y\_train).

This step fits the linear regression model to the training data, allowing it to learn the relationship between the features and the target variable.

**8.5 Evaluate the model**

****

Model Evaluation

The performance of the trained Linear Regression model was assessed using the testing data:

Predictions --> The model's predictions on the test set were obtained.

Evaluation Metrics

Mean Squared Error (MSE) --> Measures the average squared difference between the actual and predicted values.

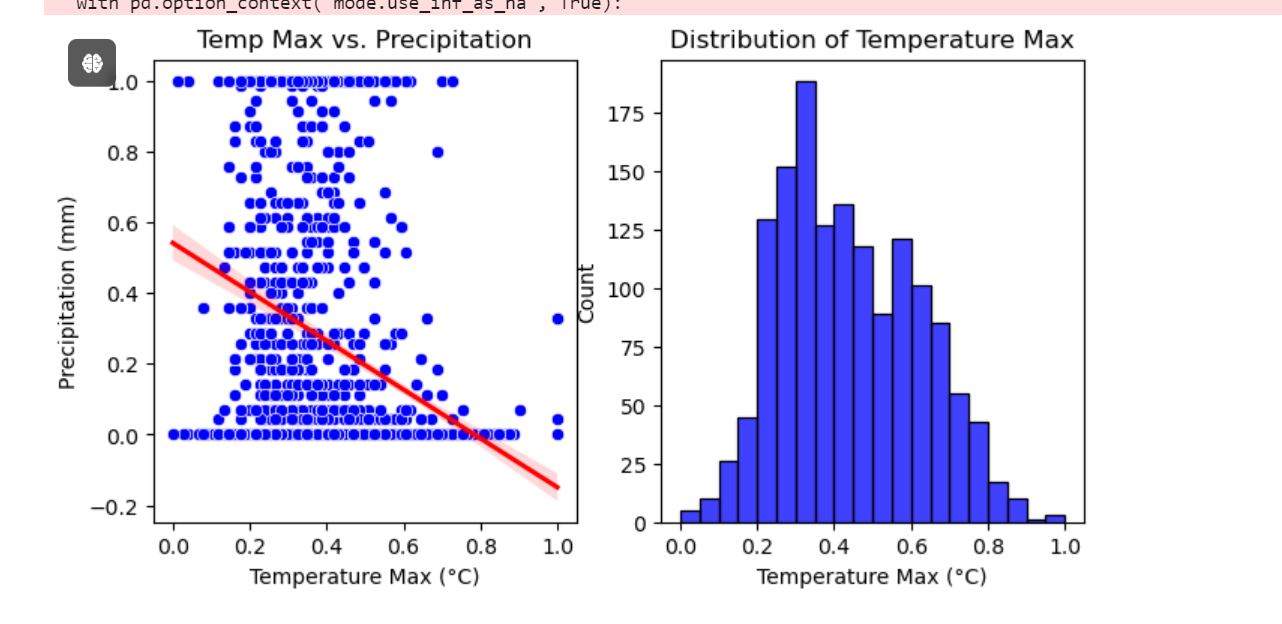
R Squared (R²)--> Represents the proportion of variance in the target variable that is predictable from the features.

***9. Visualization***

* To graphically represent data patterns and relationships, making it easier to interpret and communicate insights from the dataset.

Visualizations were created to explore the relationships between **temp\_max, temp\_min, wind, and precipitation:**

**Temp Max vs. Precipitation:**



A scatter plot with a regression line to show the relationship between **temp\_max and precipitation.**

**Scatter Plot** -----> Displays individual data points for temp\_max and precipitation.

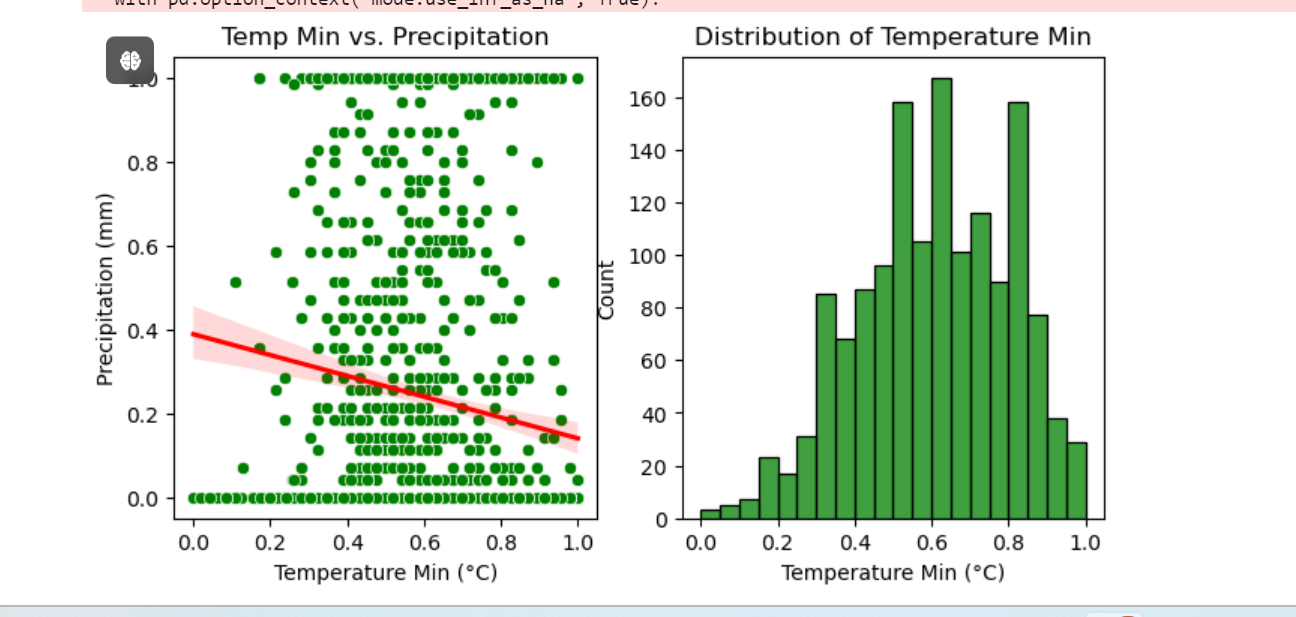
**Regression Line** ----->Added to show the trend in the data.

Distribution of Temperature Max:

A **histogram** to show the distribution of temp\_max.

**Temp Min vs. Precipitation:**

****



A scatter plot with a regression line to illustrate the relationship between **temp\_min and precipitation.**

**Scatter Plot** -----> Displays the data points for temp\_min against precipitation.

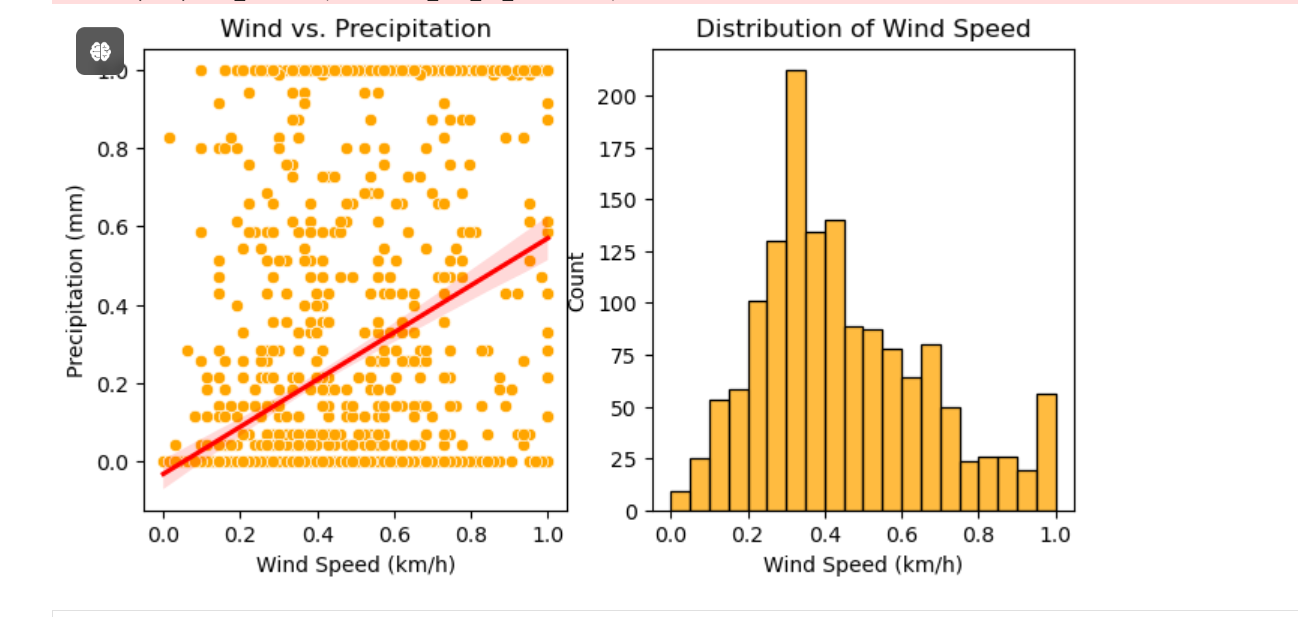
**Regression Line** -----> Shows the trend or correlation between temp\_min and precipitation.

Distribution of Temperature Min:

A **histogram** to show the distribution of temp\_min values.

**Wind Speed vs. Precipitation:**





A scatter plot with a regression line to illustrate the relationship between **wind and precipitation.**

**Scatter Plot**----> Displays the data points for wind against precipitation.

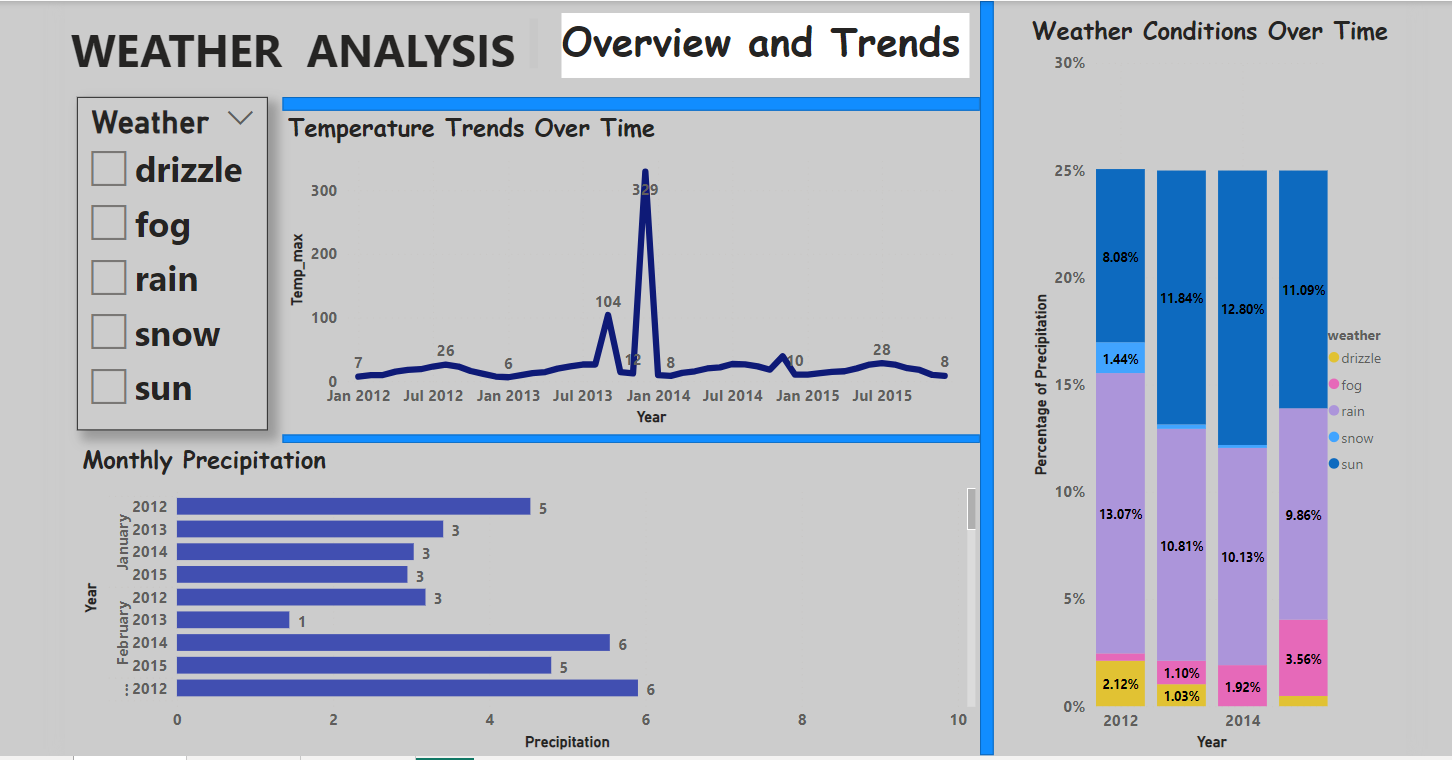
**Regression Line**----> Shows the trend or correlation between wind and precipitation.

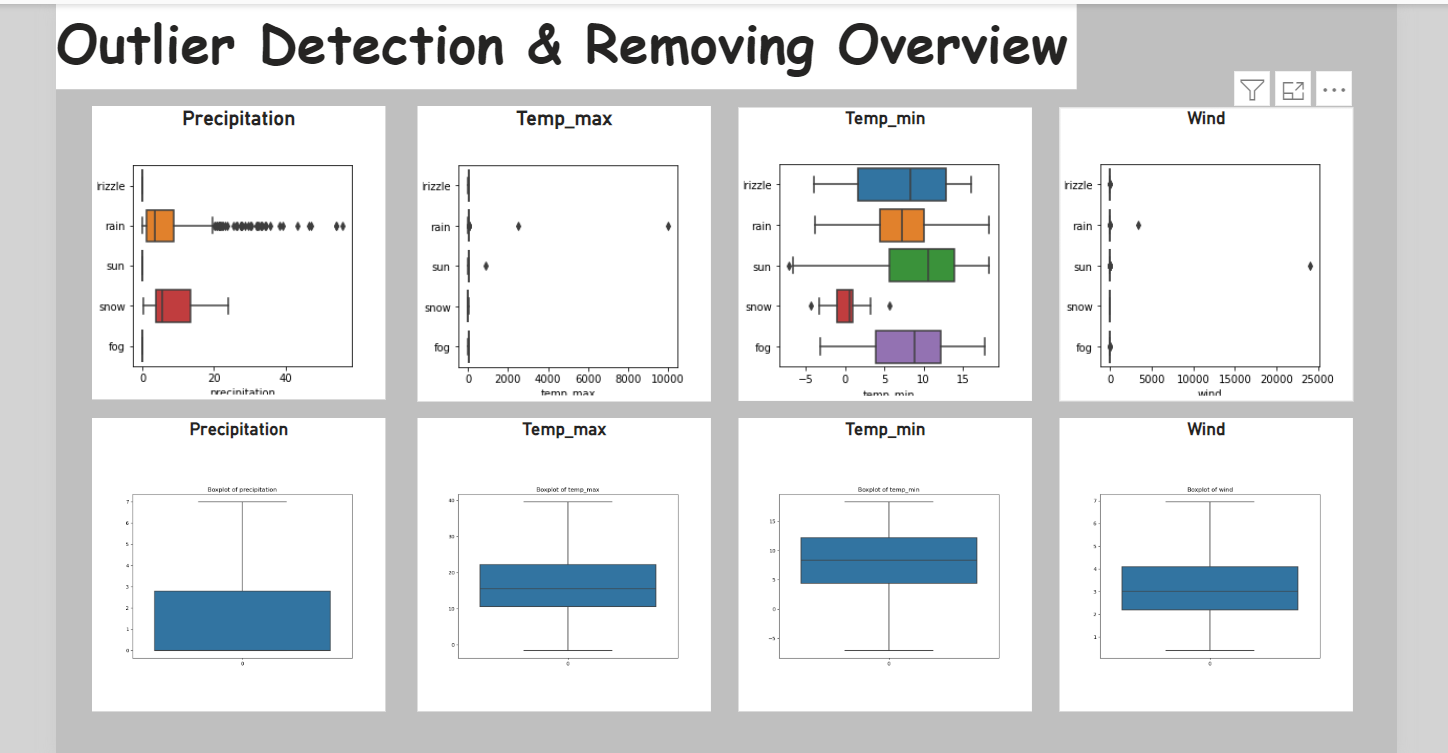
Distribution of Wind Speed:

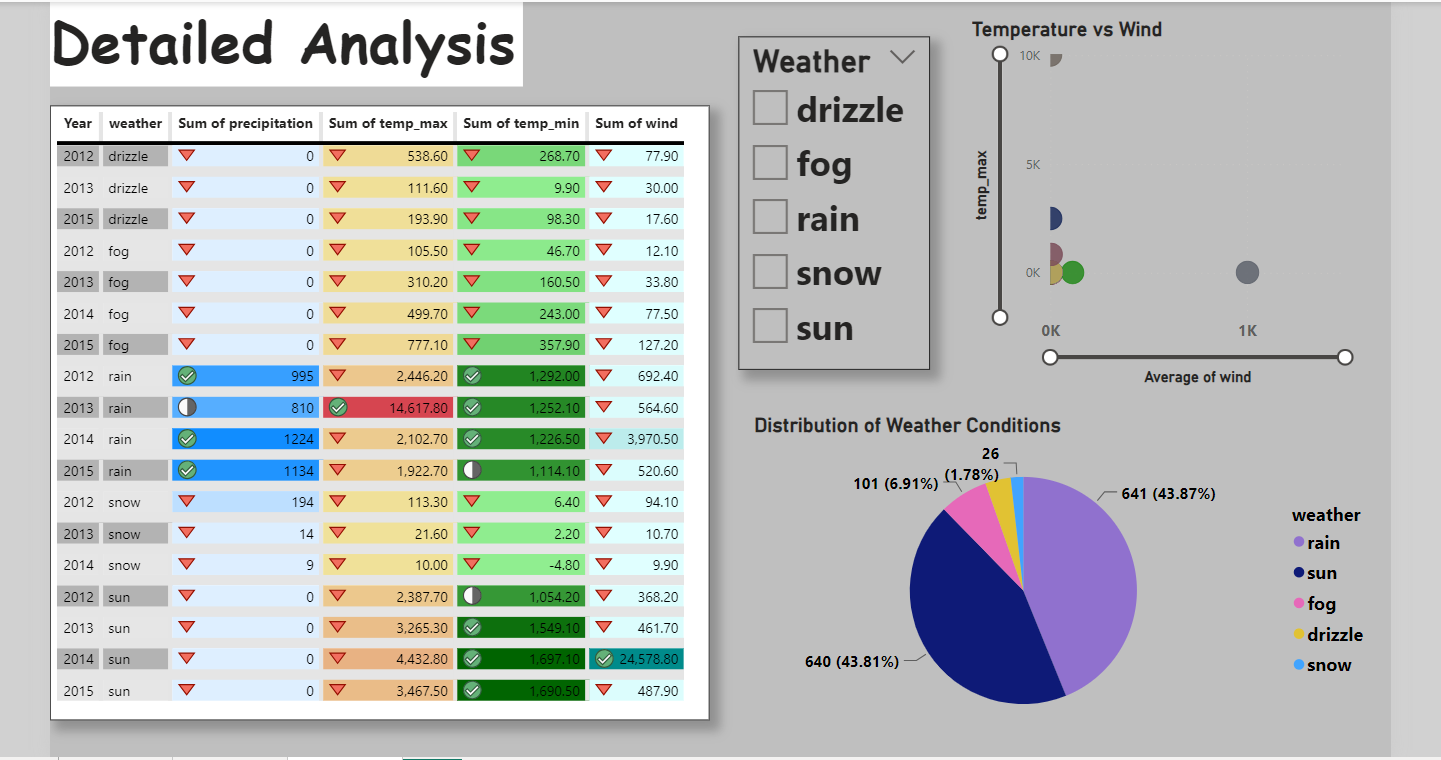
**A histogram** to show the distribution of wind speed values.

***10. Visualization of the Iris Dataset***

**10.1 Using Power BI**

******





***11. Conclusion***

The analysis of the weather dataset has successfully highlighted key patterns and relationships among various weather attributes.

By conducting thorough data cleaning and preprocessing, including handling missing values, outliers, and encoding categorical variables, the dataset was prepared for meaningful analysis.

Exploratory Data Analysis (EDA) revealed important insights into the distribution and correlation of weather features.

Regression analysis demonstrated the predictive relationships between variables, specifically how different weather conditions can influence temperature.

Visualizations in Python and Power BI provided a clear representation of the data, making it easier to interpret trends and draw conclusions. Overall, the analysis enhances understanding of weather dynamics.