**Final Project Documentation**

Weather Prediction ML Model

Supervised by : ENG/ Mohamed Ebrahim

**Project Overview**

We will develop a predictive model utilizing data from the [**Visual Crossing Weather**](https://www.visualcrossing.com/) API, which is the easiest-to-use and lowest-cost source for historical and forecast weather data. These models offer geographic-specific weather predictions with varying resolutions and available variables. By selecting and comparing individual weather models, we will collect daily weather data to train a machine learning model capable of predicting temperature or classifying weather conditions (e.g., rainy or sunny). This approach leverages real-time, interpolated data to create accurate weather predictions.

**Motivation**

Accurate weather predictions are important for a variety of sectors such as agriculture, disaster management, transportation, and tourism. Leveraging multiple weather models allows for more accurate and geographically specific predictions.

**Data Source**

**Visual Crossing API**

* **API Overview**: Visual Crossing offers both historical and real-time weather data, providing high-resolution weather information across the globe. It supports features such as temperature, precipitation, humidity, and other meteorological variables.
* **Data Access**: Each member of the team is allotted 1,000 rows of data per day, which is used to collect data from multiple geographic locations or time periods. This allows us to compile a diverse dataset over time, despite daily row limits.

**Data Collection**

**Data Collection Process**: The team collectively gathers weather data daily, with each member responsible for querying up to 1,000 rows of data from the Visual Crossing API. This approach enables the collection of weather data from different locations or time intervals.

* **Variables Collected**: We retrieve various weather features such as temperature, weather condition (e.g., clear, cloudy, rainy), humidity, wind speed, and precipitation levels.

**Data Features**

The dataset collected from the Visual Crossing API consists of various weather-related features that will be used to train and evaluate our machine learning model. Below is a detailed explanation of each feature:

**Feature Descriptions:**

1. **name**: The location name where the weather data is collected (e.g., Egypt).
2. **datetime**: The date for which the weather data is recorded in YYYY-MM-DD format.
3. **tempmax**: The maximum temperature recorded on the day (in degrees Celsius).
4. **tempmin**: The minimum temperature recorded on the day (in degrees Celsius).
5. **temp**: The average temperature recorded on the day (in degrees Celsius).
6. **feelslikemax**: The maximum "feels-like" temperature based on wind chill and humidity (in degrees Celsius).
7. **feelslikemin**: The minimum "feels-like" temperature based on wind chill and humidity (in degrees Celsius).
8. **feelslike**: The average "feels-like" temperature for the day (in degrees Celsius).
9. **dew**: Dew point, representing the temperature at which air becomes saturated with moisture (in degrees Celsius).
10. **humidity**: The average relative humidity for the day (in percentage).
11. **precip**: The amount of precipitation recorded during the day (in millimeters).
12. **precipprob**: The probability of precipitation occurring on that day (in percentage).
13. **precipcover**: The percentage of time during the day that precipitation was observed.
14. **preciptype**: The type of precipitation observed (e.g., rain, snow).
15. **snow**: The amount of snow recorded (in millimeters).
16. **snowdepth**: The depth of snow on the ground (in millimeters).
17. **windgust**: The maximum wind gust speed recorded during the day (in kilometers per hour).
18. **windspeed**: The average wind speed during the day (in kilometers per hour).
19. **winddir**: The wind direction in degrees (0° corresponds to north, 180° to south).
20. **sealevelpressure**: The sea-level atmospheric pressure recorded (in hectopascals or millibars).
21. **cloudcover**: The percentage of the sky covered by clouds during the day.
22. **visibility**: The average visibility recorded (in kilometers).
23. **solarradiation**: The amount of solar radiation received (in watts per square meter).
24. **solarenergy**: The amount of solar energy received during the day (in kilowatt-hours per square meter).
25. **uvindex**: The ultraviolet index indicating the strength of UV radiation.
26. **severerisk**: A numerical value indicating the risk of severe weather events.
27. **sunrise**: The time of sunrise for the day in YYYY-MM-DDTHH:MM:SS format.
28. **sunset**: The time of sunset for the day in YYYY-MM-DDTHH:MM:SS format.
29. **moonphase**: The phase of the moon for that day (e.g., full moon, new moon), represented as a fraction (0 = new moon, 1 = full moon).
30. **conditions**: A brief textual description of the weather conditions (e.g., Clear, Partially Cloudy).
31. **description**: A more detailed description of the weather conditions for the day.
32. **icon**: A simplified icon representing the weather conditions (e.g., clear-day, partly-cloudy-day).
33. **stations**: The identifiers of the weather stations used to collect the data for that day (e.g., "HECA,62366099999").

**Data Loading and Exploration**

Before proceeding with data preprocessing, it is crucial to first load the data and perform an exploratory analysis. This step helps us understand the dataset, its structure, and any potential issues that may need to be addressed before preparing the data for modeling.

**- Data Loading**

The dataset is collected from the Visual Crossing API in CSV format. Each team member has access to 1,000 rows of weather data per day, and we merge these datasets to create a unified weather dataset for analysis. Our data is (5369 Rows , and 34 Columns)

**Data Processing**

**1- Data Cleaning**: Handling missing values and correcting inconsistencies in the dataset and remove unnecessary columns

* We dropped (‘Unnames’ , ‘severerisk’ , ‘name’ , 'stations'

,’snow’,‘snowdepth’,‘preciptype’ , 'description',

'tempmax','tempmin','feelslikemin','feelslike','feelslikemax')

* we handled missing values in ‘windgust’ by filling it by median
* Then we dropped one row that has missing value in ‘Visibility’
* we checked for Duplication , and there was no duplication in our data

**2- Convert Data Types:**

* Convert 'datetime' column to actual datetime format
* Convert 'sunset' column to actual datetime format
* Convert 'sunrise' column to actual datetime format

**3- removing/clamping outliers**

**4- Correlation Analysis**

Correlation analysis is a critical part of exploratory data analysis that helps us understand the linear relationships between numerical features in the dataset. By calculating pairwise correlations between all numerical variables, we can identify which features are strongly related and potentially redundant, or which features may have the strongest impact on the target variable (e.g., temperature).

In this project, we excluded non-numerical (object) columns from the correlation analysis and used a heatmap to visually represent the correlation matrix. Here's the process we followed:

* Step 1: Calculate Correlations  
  We computed the correlation matrix for all numerical features in the dataset using the corr() method. This matrix displays the correlation coefficients between pairs of variables, with values ranging from -1 (perfect negative correlation) to +1 (perfect positive correlation). A value of 0 indicates no linear correlation between the variables.
* Step 2: Visualizing Correlations  
  A heatmap was used to visualize the correlation matrix, making it easier to interpret relationships between the features. The heatmap color scheme is set to "coolwarm," where:
  + Dark Red represents strong positive correlations close to +1 (e.g., when one variable increases, the other also tends to increase).
  + Dark Blue represents strong negative correlations close to -1 (e.g., when one variable increases, the other tends to decrease).
  + White or light colors represent correlations close to 0, indicating little to no linear relationship between the variables.
* Step 3: Drop Features that Have No Linear Relation

We dropped ‘precipcover’ , ‘precipprob’ and ‘precip’ all data is 0.0

**5- Feature Engineering: Extracting Date Components**

In this step, we perform feature engineering by creating new features from the existing datetime column to enhance the model’s ability to capture temporal patterns in the weather data. Since weather patterns can vary by season, month, or even day, breaking down the datetime column into individual components such as day, month, and year provides the model with more granular information about the time-related patterns in the data.

We extracted the following features from the datetime column:

* Day: Extracted the day of the month from the datetime column.
* Month: Extracted the month from the datetime column, which helps in capturing seasonal variations (e.g., summer vs. winter weather patterns).
* Year: Extracted the year, which can be useful in identifying long-term trends or yearly fluctuations in weather conditions.

**6- Repeated Correlation Analysis after Removing Weak Relationships**

After the initial correlation analysis, we identified several pairs of variables that had weak or no significant linear relationships (represented by white or light colors in the heatmap). These variables contributed little to understanding the relationships in the dataset and could introduce noise into the machine learning model. To optimize the dataset, we decided to remove features with low correlation values close to zero and then recompute the correlation matrix.

The second correlation heatmap now focused on stronger relationships, highlighting the key variables that are more informative for predicting weather conditions.

**Reduced Noise**: By removing the weak relationships, we reduced the potential noise in the dataset, which could have negatively affected model performance.

**Improved Interpretability**: The new heatmap is easier to interpret, as it focuses on the more meaningful correlations between the remaining variables. For example, the relationships between features like temp, humidity, and windgust became clearer.

**Visualizing Feature Distributions**

To better understand the distribution of each numerical feature in our dataset, we created **histograms** for all the numerical variables. This is an important step in data exploration, as it helps us visualize how the values of each feature are spread out, detect any outliers, and identify skewness in the data, which may affect model performance.

**Purpose of Histograms:**

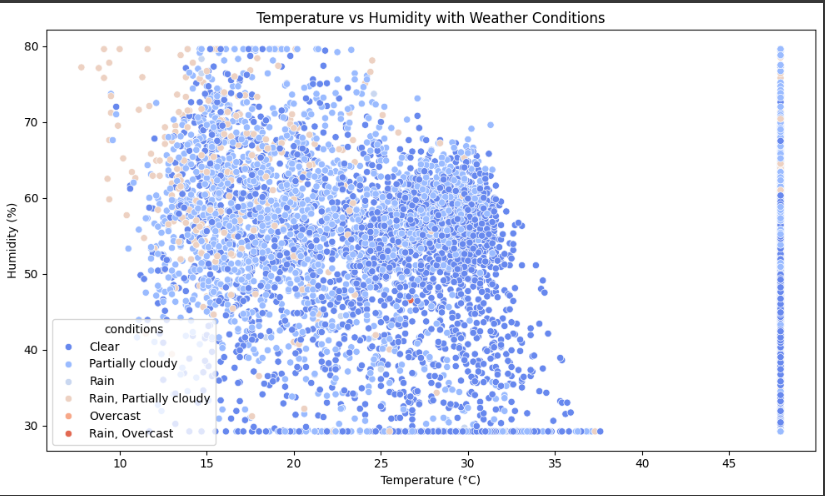
* **Identify Data Distributions**: The histograms show the frequency of values for each numerical feature. For example, you can see whether the temperature values are normally distributed or skewed towards lower or higher values.
* **Detect Outliers**: Outliers, or extreme values that do not fit within the typical range of data, become visible in the histograms as bars that are far from the majority of the data points.
* **Check for Skewness**: Some features may have skewed distributions (i.e., more values on one side of the distribution than the other). For instance, wind speed may have more lower values and fewer extreme values, indicating a right-skewed distribution.

The insights gained from these histograms are crucial for guiding further data preprocessing. For example:

* **Outlier Handling**: If we detect outliers in features like temperature or wind gusts, we may choose to handle them by capping extreme values or removing them altogether.
* **Normalization or Transformation**: If certain features (e.g., precipitation or wind speed) are heavily skewed, we may apply transformations such as log scaling to make the data more normally distributed, which can improve model performance.

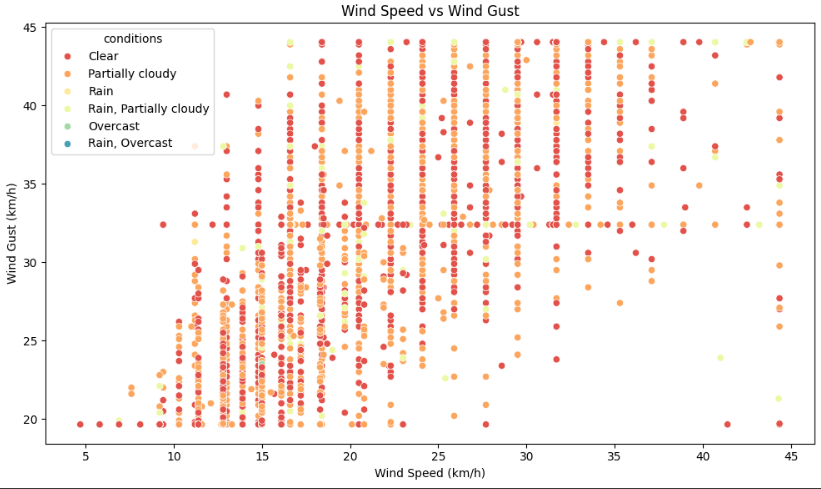
### **Data Analysis**

#### **1. Temperature vs. Humidity with Weather Conditions**



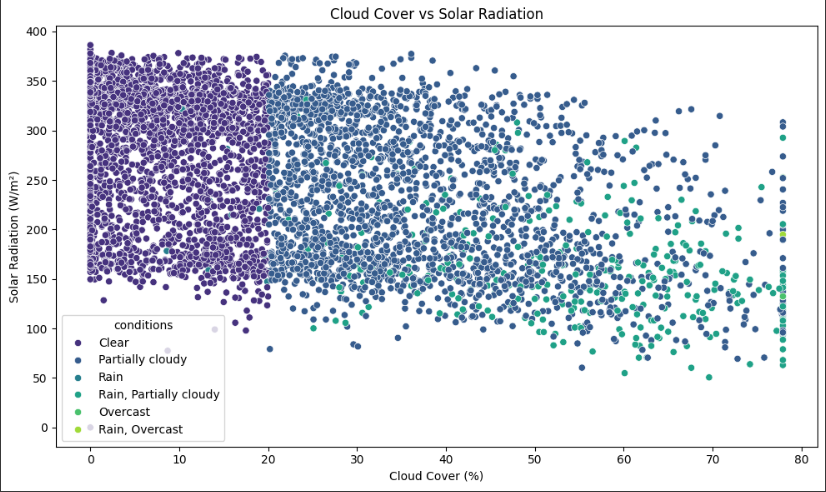
* **Correlation:** A positive correlation was observed between temperature and humidity, indicating that warmer temperatures tend to have higher humidity levels.
* **Weather Influence:** Weather conditions significantly impact the temperature-humidity relationship. Rainy and overcast conditions generally lead to higher humidity levels compared to clear or partially cloudy skies.

#### **2. Wind Speed vs. Wind Gust with Weather Conditions**



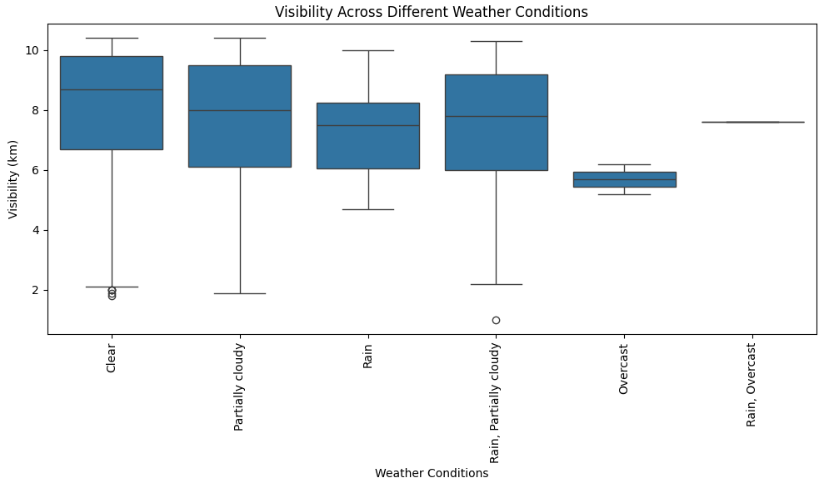
* **Correlation:** A strong positive correlation was found between wind speed and wind gust, suggesting that higher wind speeds are associated with stronger gusts.
* **Weather Influence:** Weather conditions influence the relationship. Rainy and overcast conditions tend to have a stronger association between higher wind speeds and stronger gusts.

#### **3. Cloud Cover vs. Solar Radiation**



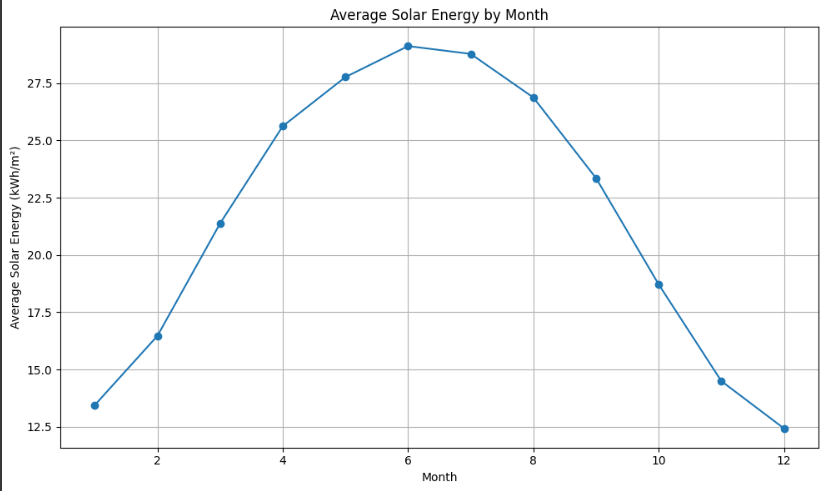
* **Negative Correlation:** A clear negative correlation was observed between cloud cover and solar radiation, indicating that increased cloud cover reduces the amount of solar energy reaching the surface.
* **Weather Influence:** Weather conditions significantly impact the relationship. Clear skies allow for maximum solar radiation, while cloudy and rainy conditions reduce solar energy significantly.

#### **4. Visibility Across Different Weather Conditions**



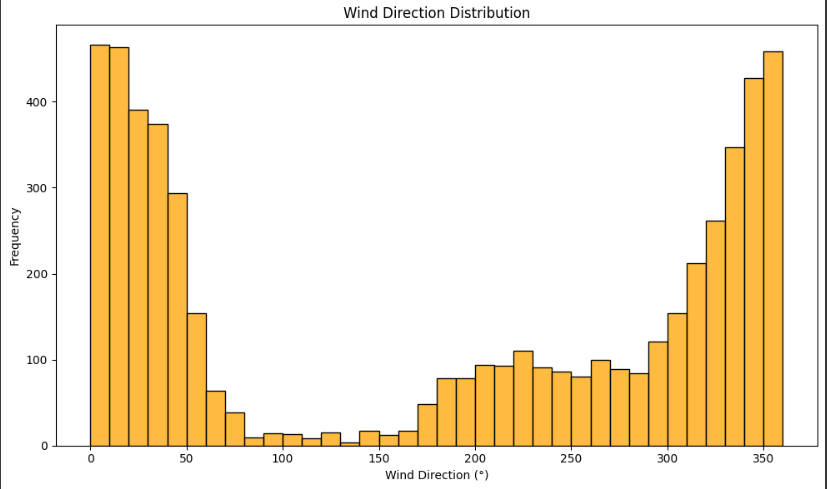
* **Median Visibility:** The median visibility tends to be highest under clear conditions and decreases progressively with increasing cloud cover and precipitation.
* **Variability:** The box plots show that visibility can vary within each weather category. Clear conditions have a wider range of visibility compared to overcast conditions.
* **Outliers:** Occasional instances of exceptionally low visibility were observed, even under favorable weather conditions.

#### **5. Solar Energy by Month**



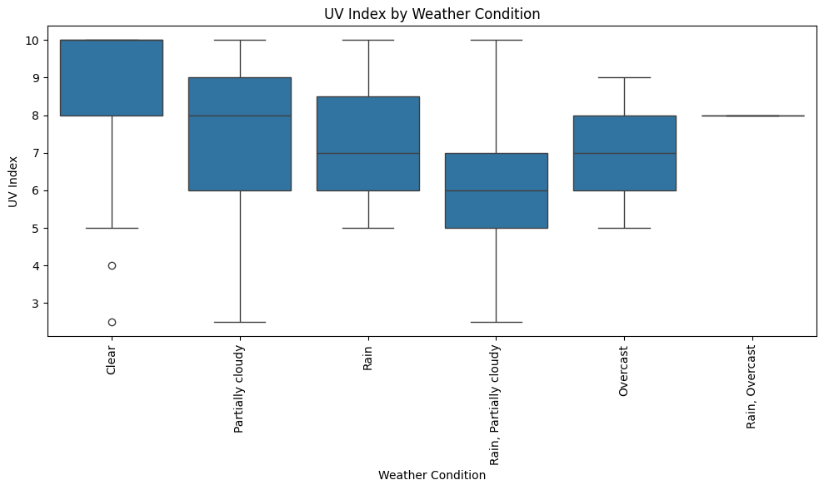
* **Seasonal Trend:** A clear seasonal trend was identified, with higher solar energy levels during summer months and lower levels during winter months.
* **Monthly Variability:** The box plots show that solar energy can vary within each month, with wider ranges in summer months compared to winter months.
* **Outliers:** Exceptional instances of high or low solar energy levels were observed.

#### **6. Wind Direction Distribution**



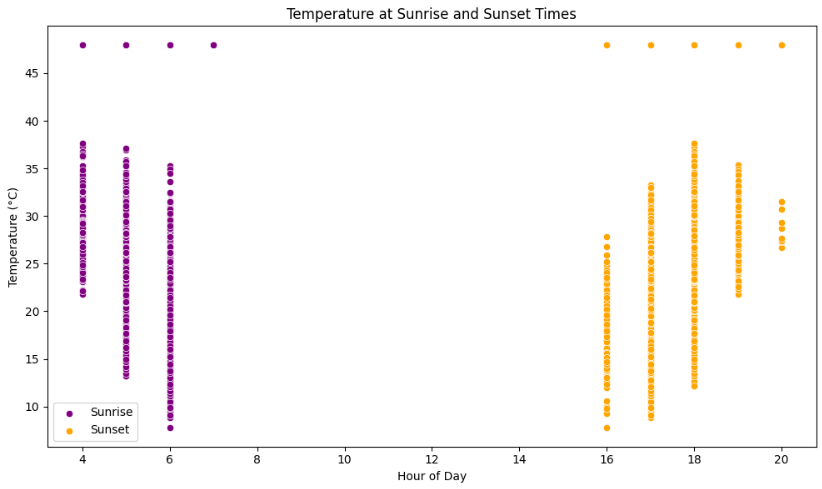
* **Prevailing Winds:** Dominant wind directions were identified, indicating the most frequent wind patterns.
* **Variability:** The spread of the bars in the histogram shows the variability in wind direction.
* **Wind Rose:** A wind rose could be used to visualize the wind direction data more intuitively.

#### **7. UV Index by Weather Condition**



* **Median UV Index:** The median UV index tends to be slightly higher under clear conditions compared to other weather categories.
* **Variability:** The box plots show that UV index can vary within each weather category. Clear conditions have a wider range of UV index values compared to overcast conditions.
* **Outliers:** Occasional instances of exceptionally high UV index values were observed.

#### **8. Temperature at Sunrise and Sunset Times**



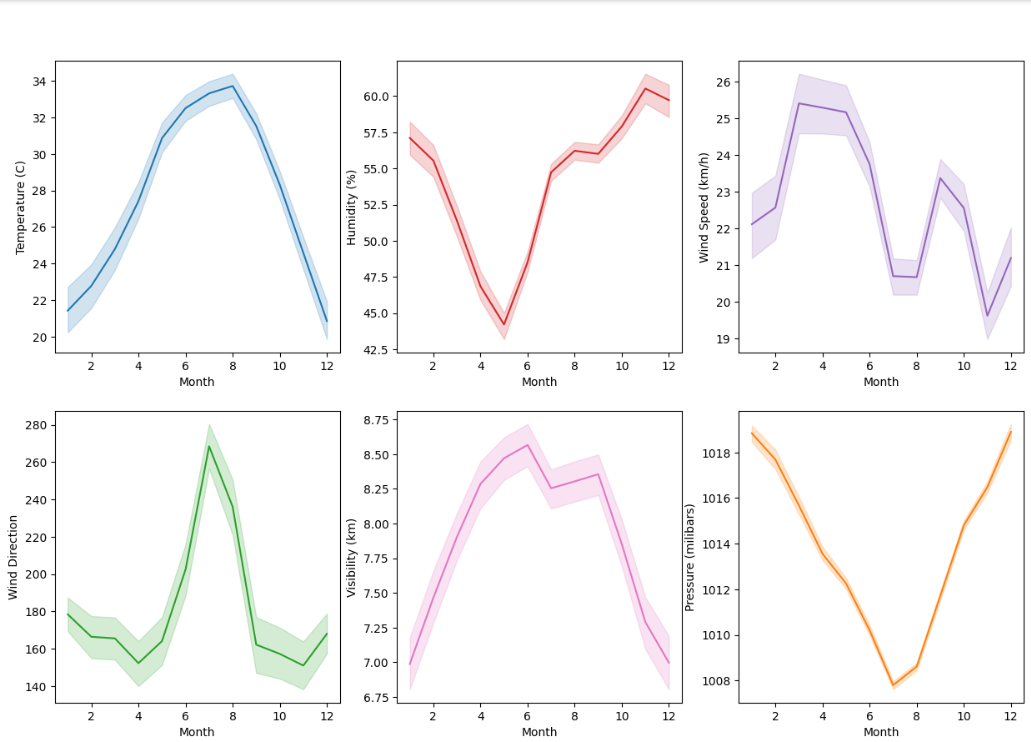
* **Temperature Variation:** The scatter plot indicates that temperatures at sunrise and sunset can vary, with general trends observed within specific ranges.
* **Seasonal Patterns (Potential):** Seasonal variations in temperature might be evident if the data spans multiple years or seasons.
* **Daylight Hours Correlation (Potential):** Analyzing the relationship between temperature and daylight hours could provide further insights into temperature variations.

### **9. Temperature Trend Over Time**

#### **Observations:**

* **Seasonal Patterns:** The plot clearly shows distinct seasonal variations in temperature, with warmer temperatures observed during summer months and cooler temperatures during winter months.

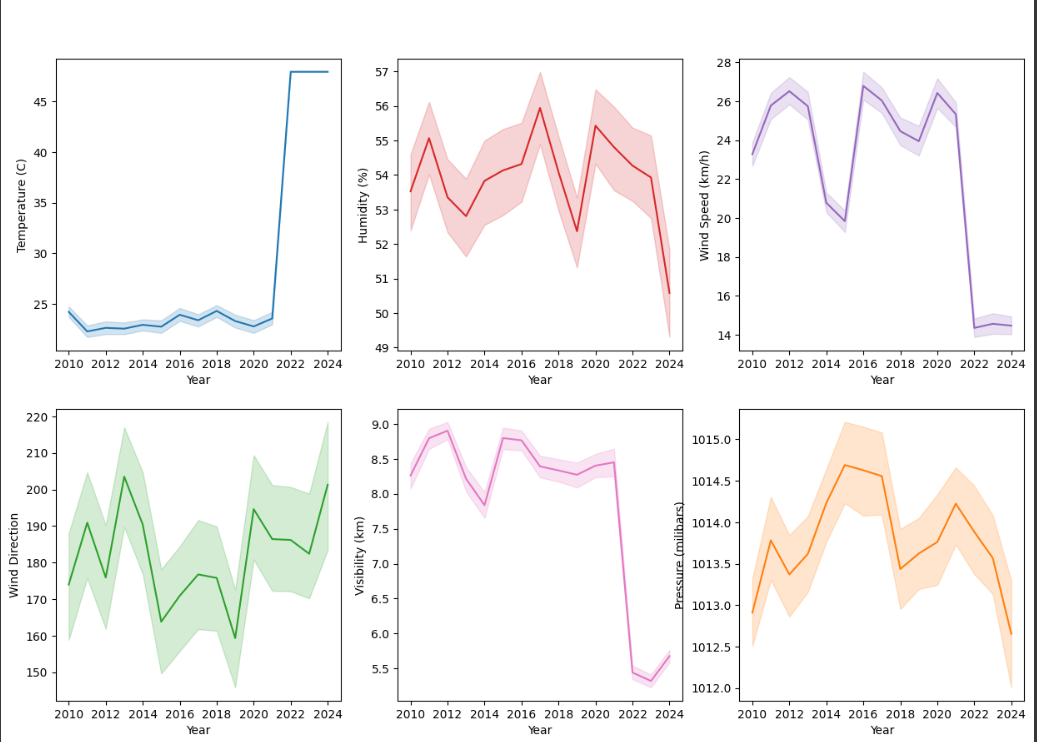
### **9.1 Year-over-Year Changes in Recorded Features(Month):**



#### **Observations:**

* **Temperature:** The temperature plot shows a clear seasonal pattern with higher temperatures during summer months and lower temperatures during winter months. There appears to be a slight upward trend in average temperatures over the years, suggesting potential warming.
* **Humidity:** The humidity plot indicates a similar seasonal pattern with higher humidity levels during summer months and lower levels during winter months. There might be a slight downward trend in average humidity, but further analysis is needed to confirm.
* **Wind Speed:** The wind speed plot shows a relatively consistent pattern throughout the year, with some seasonal variations. There doesn't seem to be a significant trend in wind speed over time.
* **Wind Direction:** The wind direction plot shows a relatively consistent distribution, suggesting no major changes in prevailing wind directions.
* **Visibility:** The visibility plot shows a slight upward trend, indicating improved visibility over time. This could be attributed to factors like air quality improvements or changes in land use.
* **Pressure:** The pressure plot shows a relatively stable trend, with some seasonal variations. There doesn't seem to be a significant trend in pressure over time.

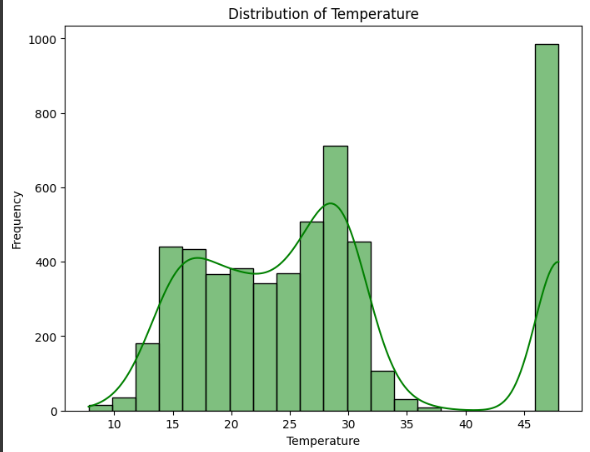
### **9.2 Year-over-Year Changes in Recorded Features(Year):**



#### **Observations:**

* **Temperature:** The temperature plot shows a clear upward trend, particularly from 2020 onwards. This suggests a significant increase in average temperatures over the recent years.
* **Humidity:** The humidity plot shows a slight downward trend, indicating a decrease in average humidity levels.
* **Wind Speed:** The wind speed plot shows a relatively stable trend with some minor fluctuations. There doesn't seem to be a significant change in average wind speeds.
* **Wind Direction:** The wind direction plot shows a consistent distribution, suggesting no major changes in prevailing wind directions.
* **Visibility:** The visibility plot shows a slight upward trend, indicating improved visibility over time. This could be attributed to factors like air quality improvements or changes in land use.
* **Pressure:** The pressure plot shows a relatively stable trend with some minor fluctuations. There doesn't seem to be a significant change in average pressure.

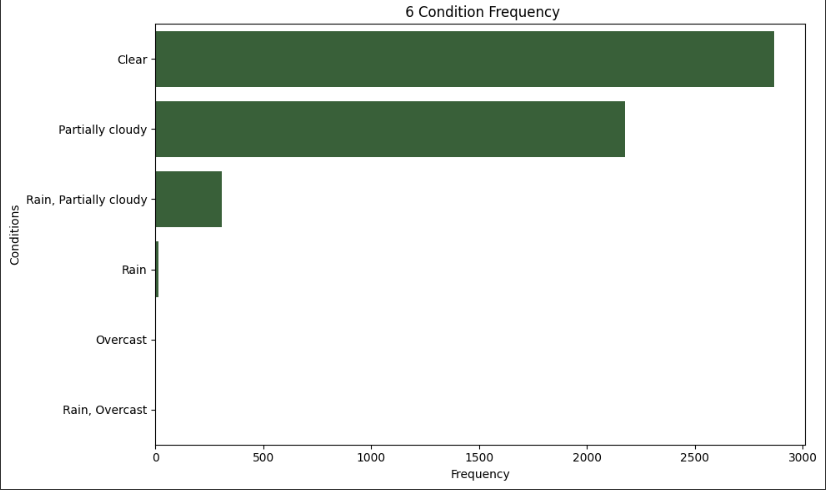
### **10. Distribution of Temperature:**



### **Observations:**

* **Skewness:** The distribution is slightly skewed to the right, indicating a slight positive skew. This means there are a few higher temperature values that are pulling the mean to the right.

#### **11. Condition Frequency:**



#### **Observations:**

* **Dominant Conditions:** The plot clearly shows that "Clear" and "Partially cloudy" are the most frequent weather conditions, with significantly higher frequencies compared to the other categories.
* **Less Frequent Conditions:** "Rain" and "Overcast" conditions are much less frequent, with "Rain" being the least common.

### **Combined Conditions:** The categories "Rain, Partially cloudy" and "Rain, Overcast" have relatively low frequencies, indicating that combinations of rain and cloudiness are less common.

### 

### 

### **Modeling**

#### **Data Preparation**

The modeling phase of the project involved preparing the data for various regression algorithms. The initial step was to separate the target variable (temp) from the feature set. The features were stored in the variable **X**, while the target variable was stored in **y**. The dataset was then divided into training and testing sets using an 80-20 split. This allows us to train the model on a subset of the data while reserving a portion for evaluating its performance.

#### **1. Linear Regression**

The first model evaluated was the Linear Regression model. After training the model on the training data, predictions were made on the test set. The performance of the model was measured using two metrics:

* **R² Score**: This statistic indicates how well the model explains the variance in the target variable. A score close to 1 indicates a good fit.
* **Mean Squared Error (MSE)**: This measures the average squared difference between the actual and predicted values, with a lower value indicating better model performance.

The results showed that the Linear Regression model achieved an R² score of **0.96251** and an MSE of **4.63278**.

#### **2. K-Nearest Neighbors (KNN)**

The next model tested was the K-Nearest Neighbors regressor. Similar to the previous model, it was trained using the training data and evaluated on the test set. The KNN model achieved an R² score of **0.96321** and an MSE of **4.54734**, indicating a slight improvement in performance compared to Linear Regression.

#### **3. Support Vector Regression (SVR)**

The Support Vector Regression model was also evaluated. This model utilizes a different approach to make predictions based on the support vectors derived from the training data. However, it resulted in a lower R² score of **0.68788** and a higher MSE of **38.57323**, suggesting it was not as effective for this particular dataset.

#### **4. Decision Tree Regression**

A Decision Tree Regressor was then applied. This model partitions the feature space into regions that predict the target variable. The Decision Tree model yielded an R² score of **0.98281** and an MSE of **2.12432**, demonstrating excellent performance.

#### **5. Random Forest Regression**

Finally, a Random Forest Regressor was employed. This ensemble method constructs multiple decision trees and averages their predictions to enhance model robustness. The Random Forest model achieved the highest performance with an R² score of **0.99320** and an MSE of **0.84081**.

### **Models Comparison**

To summarize the performance of all models, the results were compiled into a comparative table. The models were evaluated based on their R² scores and Mean Squared Errors, allowing for a clear visual representation of which models performed best.

A bar plot was generated to visually compare the R² scores and MSE across the different models, highlighting the effectiveness of the Random Forest Regressor as the most reliable model in this analysis.

This comparative analysis not only provides insights into each model's performance but also assists in selecting the most suitable model for future predictions regarding temperature based on the features considered in the dataset.

### **Streamlit Interface**

The Streamlit app allows users to interactively predict the temperature by inputting various weather-related data points. Below is a detailed explanation of each component of the interface:

### **1. Title and Subheader**

Upon launching the application, the user is greeted with a title and a subheader:

* **Title**: "Temperature Prediction Application"
  + This clearly defines the purpose of the application.
* **Subheader**: "Enter weather conditions to predict temperature:"
  + The subheader guides the user to input weather data for the temperature prediction.

### **2. User Input Fields**

The app uses several input fields to allow the user to provide relevant weather conditions that will be used for temperature prediction. The inputs are captured using Streamlit’s number\_input widget, which ensures that only numerical data is entered, with default values provided for each field.

Below is a description of each input field:

* **Dew Point**:
  + Represents the atmospheric temperature below which water droplets begin to condense and dew can form.
  + Input Range: Positive or negative values (default value: 10.0°C).
* **Humidity (%)**:
  + Percentage of water vapor in the air.
  + Input Range: 0 to 100% (default value: 60%).
* **Wind Gust Speed (km/h)**:
  + The highest speed reached by the wind in short bursts.
  + Input Range: Positive values (default value: 32.4 km/h).
* **Wind Speed (km/h)**:
  + The sustained speed of the wind.
  + Input Range: Positive values (default value: 15.0 km/h).
* **Wind Direction (degrees)**:
  + The direction from which the wind is blowing, measured in degrees from north.
  + Input Range: 0 to 360 degrees (default value: 200.0°).
* **Sea Level Pressure (hPa)**:
  + The atmospheric pressure at sea level, usually measured in hectopascals.
  + Input Range: Positive values (default value: 1015.0 hPa).
* **Cloud Cover (%)**:
  + The percentage of the sky covered by clouds.
  + Input Range: 0 to 100% (default value: 20%).
* **Visibility (km)**:
  + How far one can clearly see, usually measured in kilometers.
  + Input Range: Positive values (default value: 8.0 km).
* **Solar Radiation (W/m²)**:
  + The power per unit area received from the Sun in the form of electromagnetic radiation.
  + Input Range: Positive values (default value: 160.0 W/m²).
* **Solar Energy (MJ/m²)**:
  + The energy per unit area derived from solar radiation.
  + Input Range: Positive values (default value: 14.0 MJ/m²).
* **UV Index**:
  + A measure of the intensity of ultraviolet (UV) radiation from the Sun.
  + Input Range: 0 to 11 (default value: 6.0).
* **Day**:
  + The day of the month.
  + Input Range: 1 to 31 (default value: 1).
* **Month**:
  + The month of the year.
  + Input Range: 1 to 12 (default value: 1).
* **Year**:
  + The year within the range of available data.
  + Input Range: 2010 to 2024 (default value: 2023).

**Default Values**: Each input field is pre-populated with a reasonable default value that can be adjusted by the user. This makes it easy for users to provide inputs without needing to start from scratch.

### **3. Prediction Button**

* **Predict Button**: After the user enters all the necessary weather data, they can press the "Predict Temperature" button to submit their inputs for processing. Once clicked, the app uses these inputs to generate a temperature prediction.

### **4. Input Data Handling**

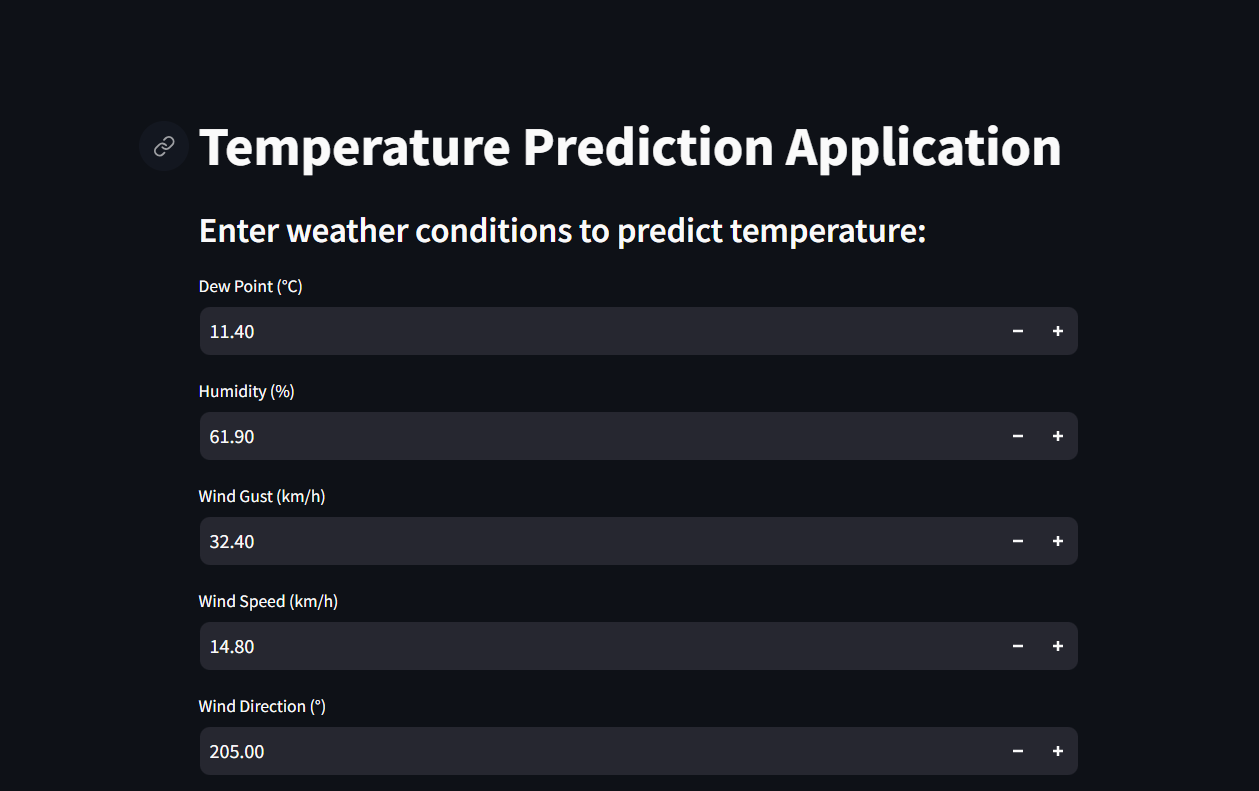
When the user clicks the **Predict Temperature** button, the following occurs:

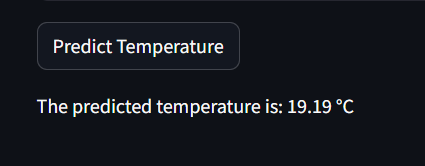
* **Input Data Collection**: The entered weather conditions are collected into a structured format (i.e., a pandas DataFrame) that matches the format expected by the machine learning model.
* **Model Prediction**: The app uses the pre-trained Random Forest Regressor model to predict the temperature. The model, which has been trained on historical weather data, takes the user’s inputs and predicts the likely temperature for the given conditions.

### **5. Displaying the Predicted Temperature**

Once the prediction is made, the app displays the result:

* **Predicted Temperature**: The temperature is shown to the user in Celsius with two decimal places for precision. This prediction represents the most likely temperature based on the weather conditions the user provided.
* **Result Example**: If the predicted temperature is, for example, 25.42°C, the app will display:
  + "The predicted temperature is: 25.42°C"





### **Performance and User Experience**

* **Real-time Interaction**: The app provides instant feedback to the user. Once the "Predict Temperature" button is clicked, the app processes the inputs and quickly displays the prediction.
* **Ease of Use**: The app's interface is intuitive, with easy-to-understand input fields and buttons. Users can adjust their inputs and re-predict the temperature as many times as they want without refreshing the app.
* **Model Accuracy**: The prediction is based on a trained Random Forest Regressor, which provides a robust and accurate prediction based on the user’s inputs.

### **Key Features Recap:**

1. **User-Friendly Input Fields**: Enter weather parameters like wind speed, humidity, solar energy, and more.
2. **Predict Button**: Quickly processes the inputs and generates a temperature prediction.
3. **Real-Time Results**: Displays predicted temperature instantly in an easy-to-read format.
4. **High Precision**: Outputs temperature with two decimal places.