Weather forecasting using Machine Learning

Introduction

Weather forecasting plays a crucial role in **smart agriculture**, helping farmers make informed decisions about irrigation, planting, and harvesting. However, traditional weather forecasts often lack the accuracy needed for **hyper-local conditions**, leading to inefficient resource usage and potential crop losses. By leveraging **machine learning**, we can develop **data-driven predictive models** that analyze historical weather patterns and provide more precise forecasts. This project aims to build a **machine learning model** that predicts whether it will rain based on factors such as **temperature, humidity, wind speed, and atmospheric pressure**. By improving rain predictions, this model can assist farmers in optimizing their agricultural practices and minimizing risks associated with unpredictable weather changes.

Problem Statement

Farmers heavily rely on **weather forecasts** to plan their agricultural activities, but traditional forecasting methods often fail to provide **accurate, hyper-local predictions**. This inaccuracy can lead to **inefficient irrigation**, **crop damage**, and **financial losses** due to unexpected weather changes. To address this issue, a **data-driven approach** using **machine learning** can help improve rain predictions by analyzing historical weather patterns. However, several challenges arise in this process, including **data inconsistencies** such as **missing values, incorrect entries, and formatting errors**. Additionally, achieving a balance between **model accuracy and computational efficiency** is crucial to ensure the system remains reliable and practical for real-world applications. Overcoming these challenges will enable the development of a robust model that provides **accurate and actionable insights for farmers**.

Dataset description

The dataset contains **311 days of weather observations**.
Features

1. **avg_temperature**: Average temperature (°C)
2. **humidity**: Humidity (%)
3. **avg_wind_speed**: Wind speed (km/h)
4. **rain_or_not**: Binary label (1 = rain, 0 = no rain)
5. **date**: Date of observation
6. **cloud_cover**: cloud_cover(%)
7. **pressure**: Air pressure: (hPa)

Data Preprocessing

Handled Missing Values using Mean Imputation

- We identified **15 rows** with missing values.
- Instead of removing them, we replaced the missing values with the **mean of the respective column**.
- **Why?** Removing rows would reduce the dataset size, potentially **losing valuable information**. Using the mean ensures we **retain the data while maintaining statistical consistency**.

Converted date column into datetime format

- Transformed the `date` column into a proper **datetime format**.
- **Why?** This allows us to extract useful features like **month and day**, which can help detect **seasonal patterns in rain occurrence**.

Extracted New Features: `month` and `day`

- Derived **`month`** and **`day`** from the `date` column.
- **Why?** Seasonal variations affect rain patterns, so having these features can improve prediction accuracy.

Dropped the `date` column

- Once we extracted **`month` and `day`**, we removed the original `date` column.
- **Why?** The `date` column itself is not useful as a direct input for machine learning models.

Standardized Numerical Features using `StandardScaler()`

- Applied **StandardScaler** to transform numerical columns (e.g., `avg_temperature`, `humidity`, `avg_wind_speed`, `pressure`).
- **Why?** Different features have different units and scales, which can negatively impact machine learning models. Standardization ensures that all features contribute **equally** to model training.

Converted `rain_or_not` into Binary Format (0 & 1)

- The `rain_or_not` column was already in binary form (1 = rain, 0 = no rain).
- **Why?** This ensures compatibility with **classification models**, which require numeric labels.

Handled Outliers in `avg_wind_speed`

- Used **graphs (boxplots, histograms) to identify outliers** in `avg_wind_speed`.

- **Replaced extreme outliers with the mean value** of the column.
- **Why?** Outliers can negatively affect model performance, making it biased towards extreme values. Instead of removing data, we replaced extreme values with a reasonable estimate.

Other Approaches We Could’ve Taken

One alternative approach would have been **removing rows** with missing values. Instead of imputing the missing data, we could have dropped those rows. However, given that we only have 300 records, removing even 15 rows would significantly reduce the training data, which would likely impact the performance of our model.

Another option would have been to use the median or mode for imputation instead of the mean. The median is a better choice when dealing with skewed data, and the mode would be suitable for categorical values. However, the mean was selected because it works well when the data is normally distributed, making it a good fit for our dataset.

Lastly, we could have used a log transformation to handle outliers, reducing the impact of extreme values. While this approach works well for skewed distributions, replacing outliers with the mean was simpler and effective in our case, making it the preferred choice for handling outliers.

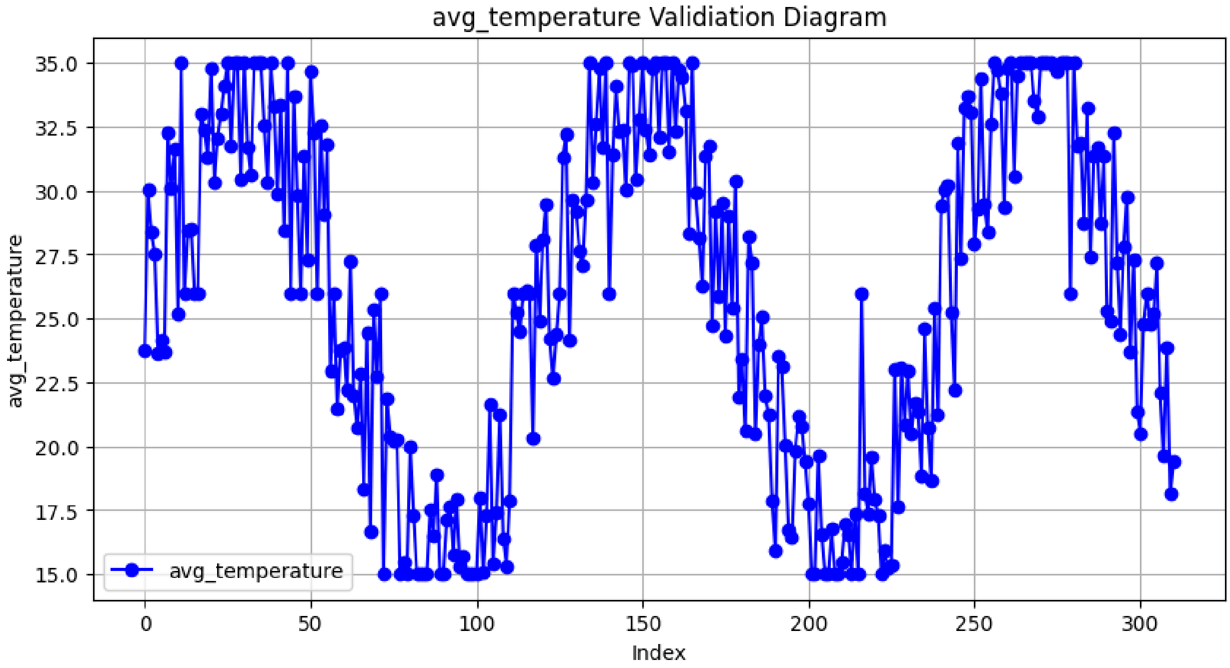
Exploratory Data Analysis (EDA) Report

Introduction

The goal of this Exploratory Data Analysis (EDA) is to better understand the dataset and uncover any hidden patterns, trends, or relationships between variables. Our primary focus is on the weather-related features such as average temperature, pressure, humidity, cloud cover, and wind speed, and how they correlate with the likelihood of rain. This process helps us identify outliers, analyze data distributions, and gain insights into feature relationships.

2. Initial Overview and Data Visualization

We started by visualizing the time series data of the **average temperature** to understand the trends over time. The plot below shows the trend of `avg_temperature` over the course of the dataset.



The graph visually represents fluctuations in temperature, which can be compared with other features (such as humidity, cloud cover, etc.) to observe if there are any coinciding patterns with rain occurrences.

Outlier Detection and Feature Analysis

Using the time series graph for the various features, we detected **outliers** in `avg_wind_speed`. These outliers can have a significant impact on modeling, as extreme values can skew predictions. To assess this more rigorously, we used **boxplots** to inspect feature distributions and identify unusual data points.

We applied this analysis to the following features:

- **Pressure**
- **Average Temperature**
- **Humidity**
- **Cloud Cover**
- **Average Wind Speed**

Boxplots for each feature gave us a clear visual representation of the central tendency, spread, and outliers. This allowed us to understand the range of values, identify skewed distributions, and determine whether some variables require transformations or outlier removal.

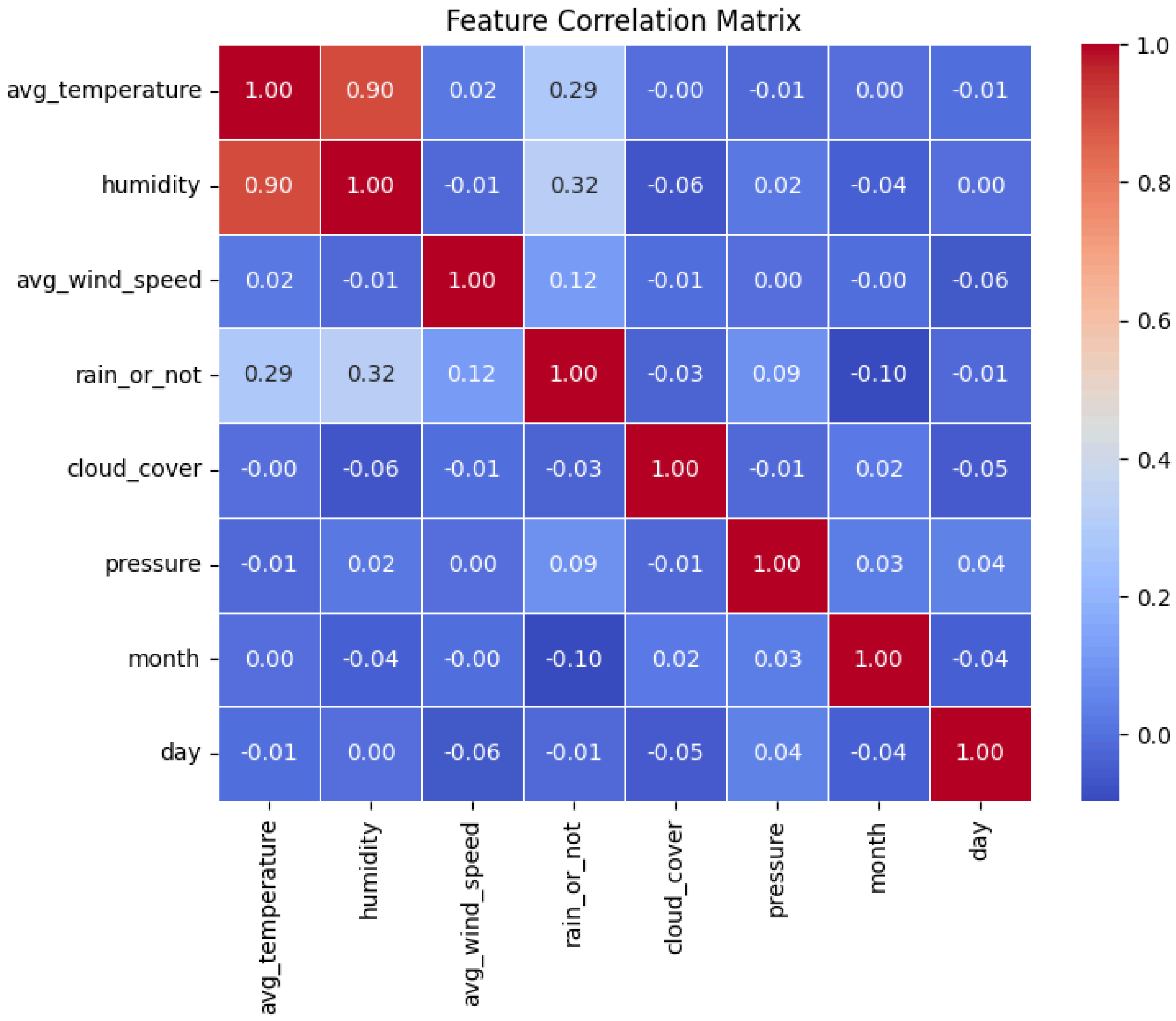
Data Distribution Analysis

Next, we investigated the overall data distribution using **histograms**. This helped us detect irregularities in the data spread, such as months with insufficient data. Upon inspection, we observed that the months of **November** and **December** had fewer data points, which could affect the generalizability of our model for these months.

This discovery was important for our analysis as it highlighted the need to either collect more data for these months or handle the imbalanced dataset in a way that does not impact the accuracy of the rain prediction model.

Correlation Analysis

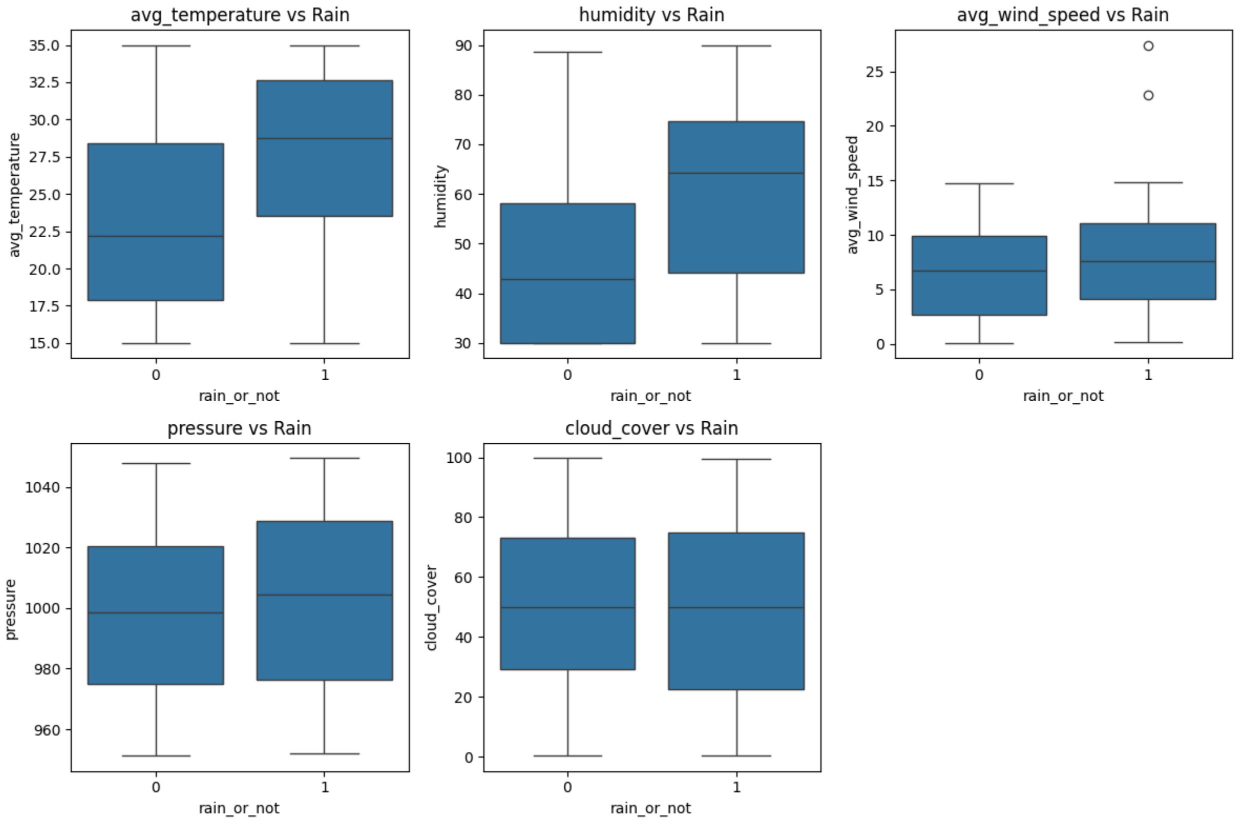
One of the critical steps in EDA is to understand the relationships between the features. We used a **correlation heatmap** to analyze how different features interact with each other.



From the heatmap, we observed that **average temperature** and **humidity** have a strong positive correlation. Both of these features also show a positive correlation with the likelihood of rain (`rain_or_not`). This is an important insight, as it suggests that higher temperatures and humidity levels are generally associated with a higher probability of rain.

Impact of Features on Rain Probability

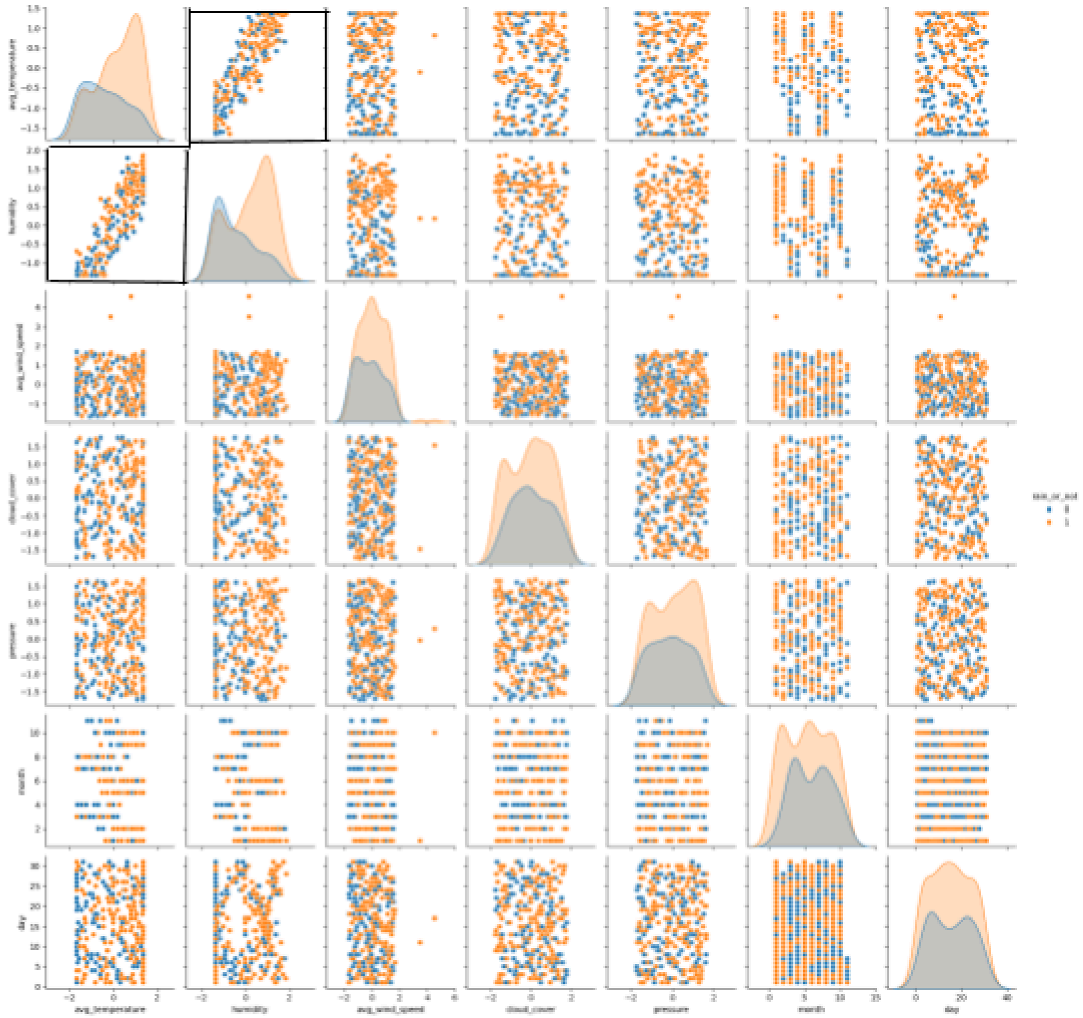
We then used **boxplots** to analyze how the various features impact the binary outcome variable, `rain_or_not`. The boxplots revealed that when **average temperature** and **humidity** were high, the likelihood of rain was significantly higher. This finding aligns with meteorological theory, where high temperatures and moisture levels often lead to precipitation.



This visual exploration is crucial for identifying the most influential features for rain prediction and can guide feature selection in future modeling steps.

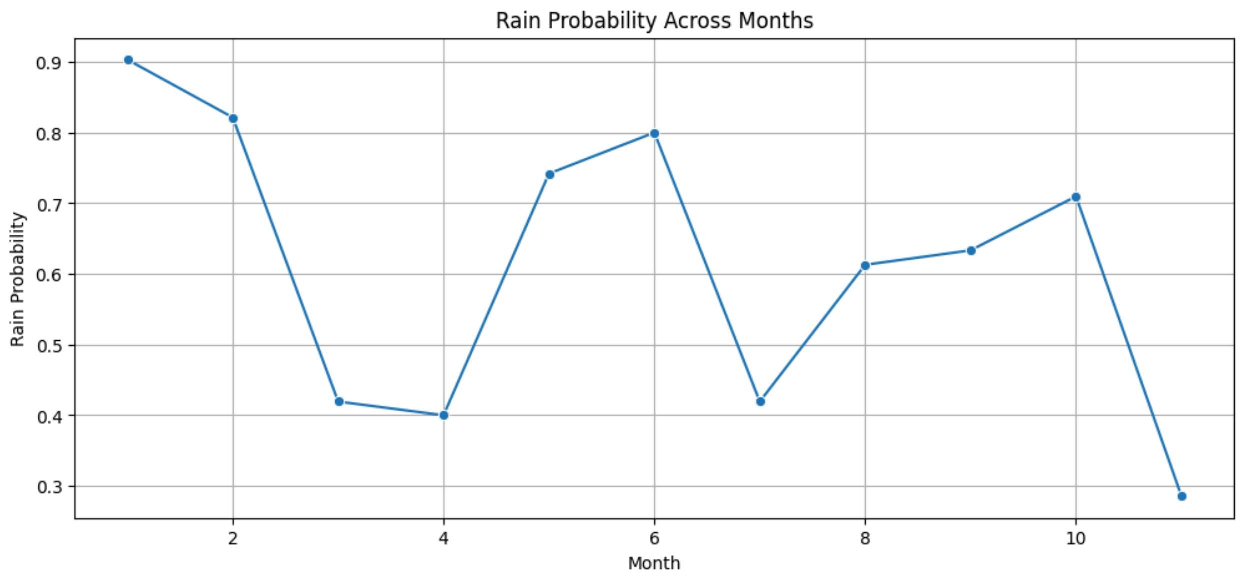
Pairplot Analysis

To further understand the relationships between multiple features at once, we utilized **pairplots**. The pairplot matrix helps visualize the interaction between different feature combinations and identifies whether any pair of variables exhibits linear or non-linear relationships. By visualizing the scatterplots and the correlation of each pair, we can identify strong relationships that may influence our target variable (i.e., the likelihood of rain).



Seasonal Trend Analysis

To analyze how rain probability varies across different seasons, we grouped the data by **month** and calculated the average rain probability for each month. This is important for understanding seasonal trends and improving the model's ability to make accurate predictions during different times of the year.



The line plot illustrates the **monthly variation in rain probability**, showing clear seasonal trends. It indicates that certain months are more likely to experience rain, which could help adjust our models for seasonal variations in precipitation.

Model Selection & Training Report

In the process of developing a machine learning model for predicting the likelihood of rain, we explored several traditional machine learning algorithms, evaluated their performance on the test data, and compared them based on key metrics such as **precision**, **recall**, **F1-score**, and **accuracy**. The models selected for evaluation are,

1. **Logistic Regression**
2. **Decision Tree Classifier**
3. **Random Forest Classifier**
4. **Gradient Boosting (XGBoost)**

Each model was trained on 70% of the data, with the remaining 30% used for testing. To ensure consistency and model optimization, we standardized the input features where necessary (Logistic Regression and XGBoost). The evaluation of the models focused on the following metrics:

- **Precision**: The proportion of true positive predictions out of all positive predictions.
- **Recall**: The proportion of true positives out of all actual positives.
- **F1-Score**: The harmonic mean of precision and recall.
- **Accuracy**: The proportion of correct predictions out of all predictions.

Training and Evaluation Metrics

Below are the detailed results for each model after evaluating them on the test dataset:

Random Forest Classifier Performance

Metric	Class 0 (No Rain)	Class 1 (Rain)	Accuracy	Macro Average	Weighted Average
Precision	0.62	0.73	0.7	0.68	0.69
Recall	0.44	0.85	0.7	0.65	0.70
F1-Score	0.52	0.78	0.7	0.65	0.69

Accuracy Score: 0.7021

- **Precision and Recall:** Random Forest achieves a reasonably good balance in precision and recall for **Class 1** (Rain) with higher recall (0.85), meaning it captures most of the rain instances. However, for **Class 0** (No Rain), precision is much lower (0.62), indicating that it occasionally misclassifies rain as no rain. The model's strong recall for rain events suggests that it is better at identifying positive rain instances, which is often crucial in weather prediction.
- **F1-Score:** The F1-score is decent for **Class 1** (0.78) and acceptable for **Class 0** (0.52). This suggests that Random Forest's ability to balance precision and recall is relatively strong.

Logistic Regression Performance

Metric	Class 0 (No Rain)	Class 1 (Rain)	Accuracy	Macro Average	Weighted Average
Precision	0.48	0.69	0.63	0.58	0.61
Recall	0.38	0.77	0.63	0.57	0.63
F1-Score	0.43	0.72	0.63	0.58	0.62

Accuracy Score: 0.6277

- **Precision and Recall:** Logistic Regression performs reasonably well for **Class 1** (Rain), with a good recall (0.77), but suffers from low precision (0.48) for **Class 0** (No Rain). This suggests it may incorrectly predict rain more often than it should.
- **F1-Score:** The F1-score is relatively low (0.63), indicating a moderate performance, though it does better than the Decision Tree.

Decision Tree Classifier Performance

Metric	Class 0 (No Rain)	Class 1 (Rain)	Accuracy	Macro Average	Weighted Average
Precision	0.34	0.63	0.52	0.48	0.52
Recall	0.35	0.62	0.52	0.48	0.52
F1-Score	0.35	0.62	0.52	0.48	0.52

Accuracy Score: 0.5213

- **Precision and Recall:** The Decision Tree model performs poorly with a precision of 0.34 for **Class 0** (No Rain) and a low recall for **Class 0** as well. While it does capture a somewhat balanced performance for **Class 1** (Rain), the overall lack of precision and recall for **Class 0** results in low overall performance.
- **F1-Score:** The F1-score (0.52) is quite low for both classes, suggesting the model struggles to effectively predict both rain and no-rain instances.

Gradient Boosting (XGBoost) Performance

Metric	Class 0 (No Rain)	Class 1 (Rain)	Accuracy	Macro Average	Weighted Average
Precision	0.50	0.71	0.64	0.60	0.63
Recall	0.47	0.73	0.64	0.60	0.64
F1-Score	0.48	0.72	0.64	0.60	0.64

Accuracy Score: 0.6383

- **Precision and Recall:** XGBoost shows a better performance than Decision Trees and Logistic Regression for **Class 0** (No Rain) with a precision of 0.50. It has a solid recall for **Class 1** (Rain) with a score of 0.73.
- **F1-Score:** The F1-score (0.64) is better than Logistic Regression and Decision Trees but still slightly lower than Random Forest.

Model Comparison and Selection

Upon reviewing the performance of all four models, **Random Forest** emerges as the best model for predicting rain, with an **accuracy of 0.7021** and superior performance metrics in terms of recall, precision, and F1-score, especially for **Class 1 (Rain)**. The Random Forest classifier captures the most important patterns in the data, especially with its higher recall rate for predicting rain.

Reasons for Choosing Random Forest

- **Higher Accuracy:** Random Forest achieved the highest accuracy of 0.7021 compared to all the other models, which suggests better generalization and prediction on unseen data.
- **Better Recall for Rain (Class 1):** A key focus in weather prediction is correctly identifying rain events. Random Forest performs excellently in this area with a recall of 0.85 for rain. This is particularly crucial as failing to predict rain can have significant consequences in real-world applications.
- **Balanced Performance Across Classes:** Although the precision for **Class 0 (No Rain)** is relatively low, Random Forest provides a good balance between both classes, outperforming other models in terms of the macro-average of precision and recall.

Why Not Other Models?

- **Logistic Regression:** While it performs decently, Logistic Regression fails to provide a high precision for **Class 0 (No Rain)**, leading to more false positives for no rain. The F1-score and overall accuracy are also lower than those of Random Forest.
- **Decision Tree:** This model shows the worst performance across all metrics. The low precision and recall for both classes indicate that the Decision Tree overfits or fails to generalize well. Its performance is far below that of the other models, making it unsuitable for this task.
- **XGBoost:** XGBoost performs better than Logistic Regression and Decision Tree but still falls short compared to Random Forest in terms of recall for **Class 1** and overall accuracy. Its performance is solid but not sufficient to outperform Random Forest.

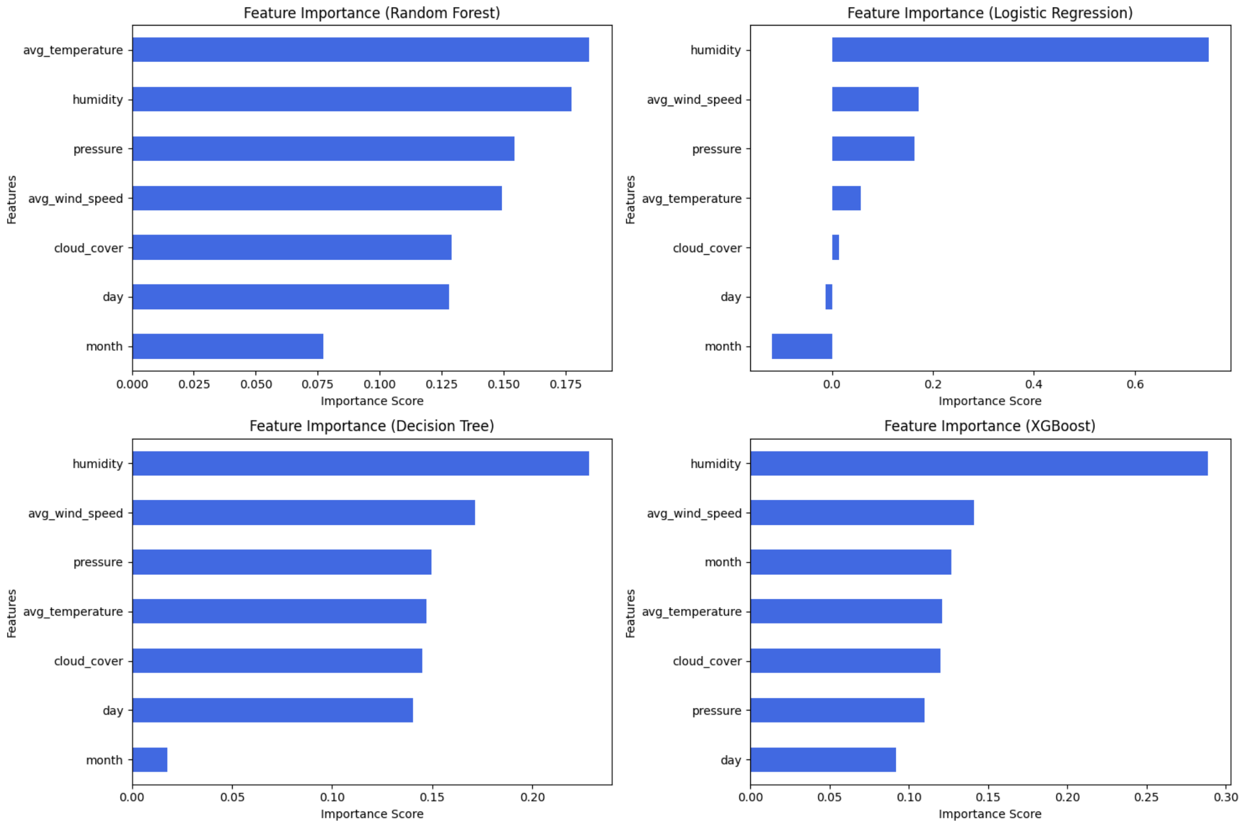
Hyperparameter tuning

Hyperparameter tuning plays a crucial role in optimizing machine learning models, as it helps improve their performance and generalization. In this case, the number of estimators (`n_estimators`) in the `RandomForestClassifier` was adjusted to **500**, and this configuration provided the best accuracy for the model. By fine-tuning the hyperparameters, we can ensure that the model is neither underfitting nor overfitting the data, leading to better predictive performance.

Cross-validation was also used to validate the model's performance across different subsets of the data, ensuring its robustness and reliability. Hyperparameter tuning and cross-validation together help in finding the optimal settings for a model, enhancing its ability to make accurate predictions on unseen data, which is essential for building reliable machine learning systems.

Feature Engineering

Random Forest models are known for their ability to handle a large number of features and deal with issues like multicollinearity and bias effectively. Since Random Forest is an ensemble method that aggregates multiple decision trees, it tends to have a lower bias and is more robust in capturing complex relationships in the data.



The fact that other models showed bias towards humidity might indicate that the feature is highly correlated with the target variable or other features. This can lead to an overemphasis on humidity by models like linear regression, which may struggle with multicollinearity, or other models that are sensitive to specific features.

It might be useful to further explore the importance of each feature in the Random Forest model using feature importance scores. This can give you a clearer idea of which features are most influential and help you decide on possible feature engineering improvements or simplifications if necessary.