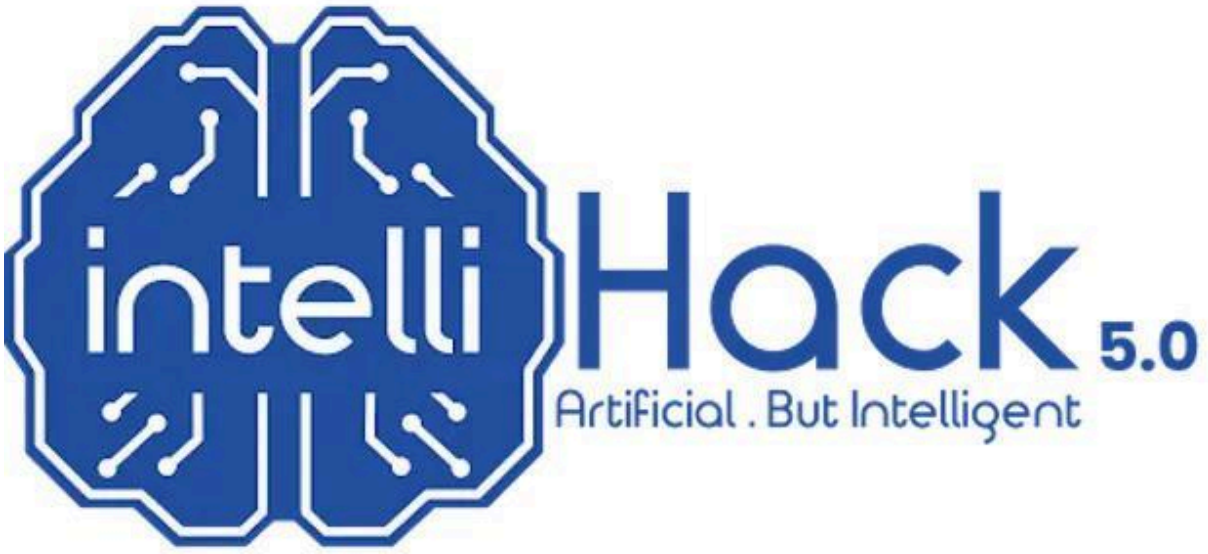
INTELLIHACK 5.0

Task 01 - Part 1



Team Cognic AI

Weather Forecasting

Introduction

Accurate weather forecasting is vital for smart agriculture, helping farmers plan irrigation, planting, and harvesting. Traditional forecasts often lack hyper-local precision. This report details the preprocessing and exploratory data analysis (EDA) performed on a 310-day weather dataset to ensure data quality for training a machine learning model to predict rainfall.

Data Preprocessing

Data Loading and Initial Inspection

	type	count	nunique	null	mode	least_frequent	mean	min	max
date	object	311	311	0	NaN	NaN	NaN	NaN	NaN
avg_temperature	float64	296	243	15	35.0	NaN	25.983840	15.000000	35.000000
humidity	float64	296	239	15	30.0	NaN	55.041385	30.000000	90.000000
avg_wind_speed	float64	296	296	15	NaN	NaN	7.556636	0.069480	56.636041
rain_or_not	object	311	2	0	Rain	No Rain	NaN	NaN	NaN
cloud_cover	float64	296	296	15	NaN	NaN	49.834827	0.321826	99.834751
pressure	float64	311	311	0	NaN	NaN	1001.059119	951.240404	1049.543752

The dataset, consisting around 300 days of weather observations, was loaded from an Excel file. An initial inspection was conducted to check its structure, verify column names, and identify missing values

Date Formatting and Duplicate Handling

To ensure consistency, the date column was converted into a proper datetime format, correcting any parsing errors. Invalid date entries were identified and removed, and the dataset was sorted chronologically. A time series check confirmed whether the data had regular daily intervals.

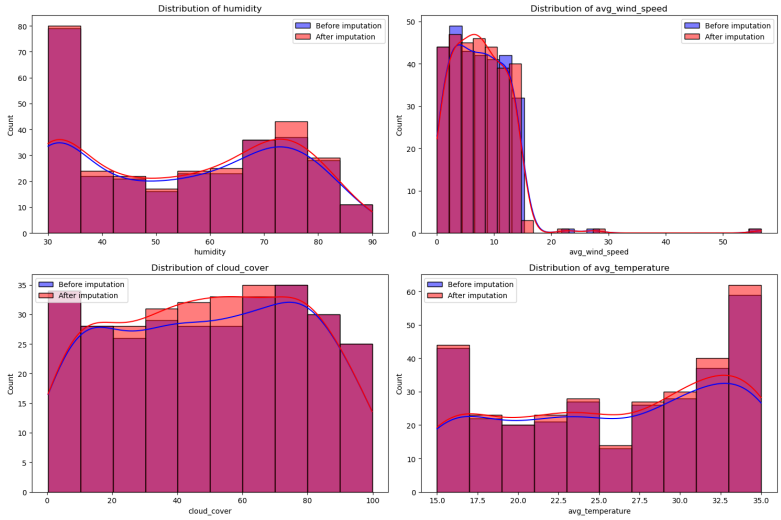
Additional date-related features, such as year, month, day, day of the week, and weekend indicators, were extracted for potential time-based analysis.

After handling date formatting, duplicate records were checked and removed to prevent redundancy in the dataset. No duplicate dates were found in the data set.

Handling Missing Data

Missing values were first identified and analyzed before applying appropriate imputation techniques:

- Temperature: Missing values were imputed using time-based interpolation, leveraging the chronological order of data for more accurate estimations.
- Humidity, Wind Speed, and Cloud Cover: These variables were imputed using K-Nearest Neighbors (KNN) imputation, ensuring values were estimated based on similar observations in the dataset.
- Remaining Numeric Features: Any remaining missing values in numerical columns were filled using median imputation to maintain data consistency.

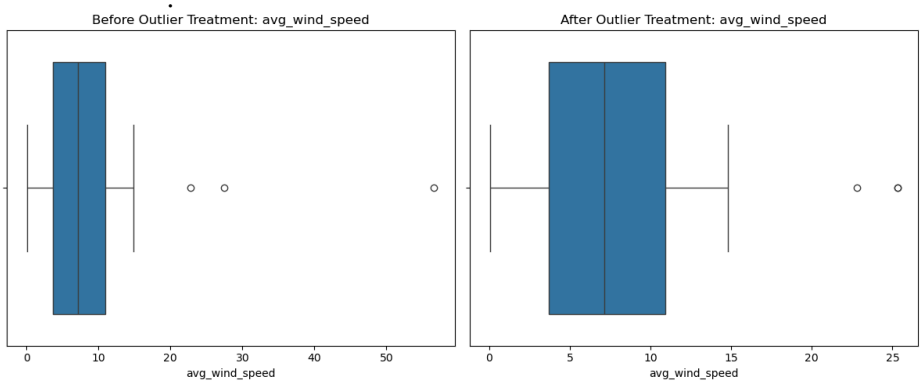


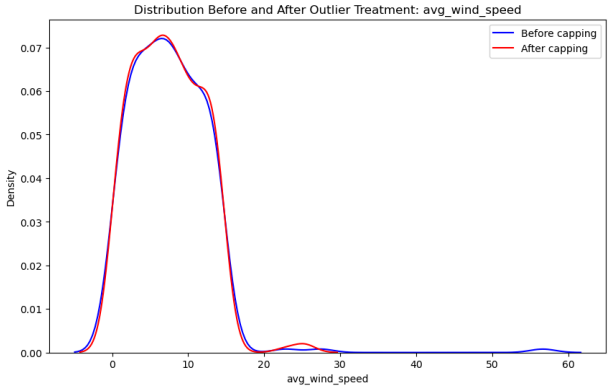
comparison of distributions before and after imputation was performed using histograms.

Handling Outliers

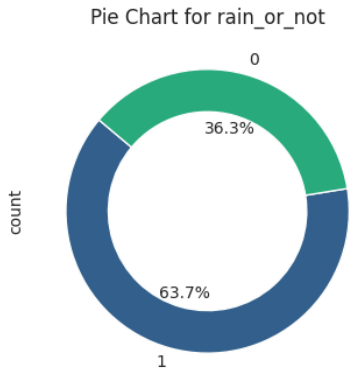
To ensure data quality, outliers in numerical features were identified and capped using statistical methods. The Interquartile Range (IQR) method was primarily used to detect and cap extreme values, while the Z-score method was applied where necessary. For most features, outliers were capped at 1.5 times the IQR, but for wind speed, a higher threshold (2.0) was used to accommodate natural variations The process included visualizing distributions before and after treatment to assess the impact of capping

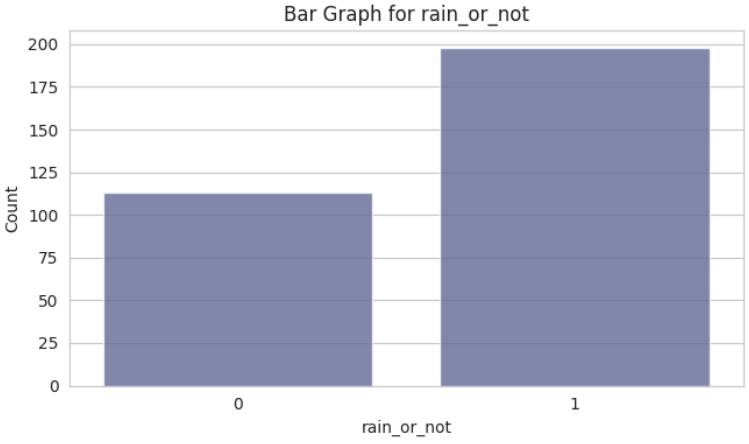
.





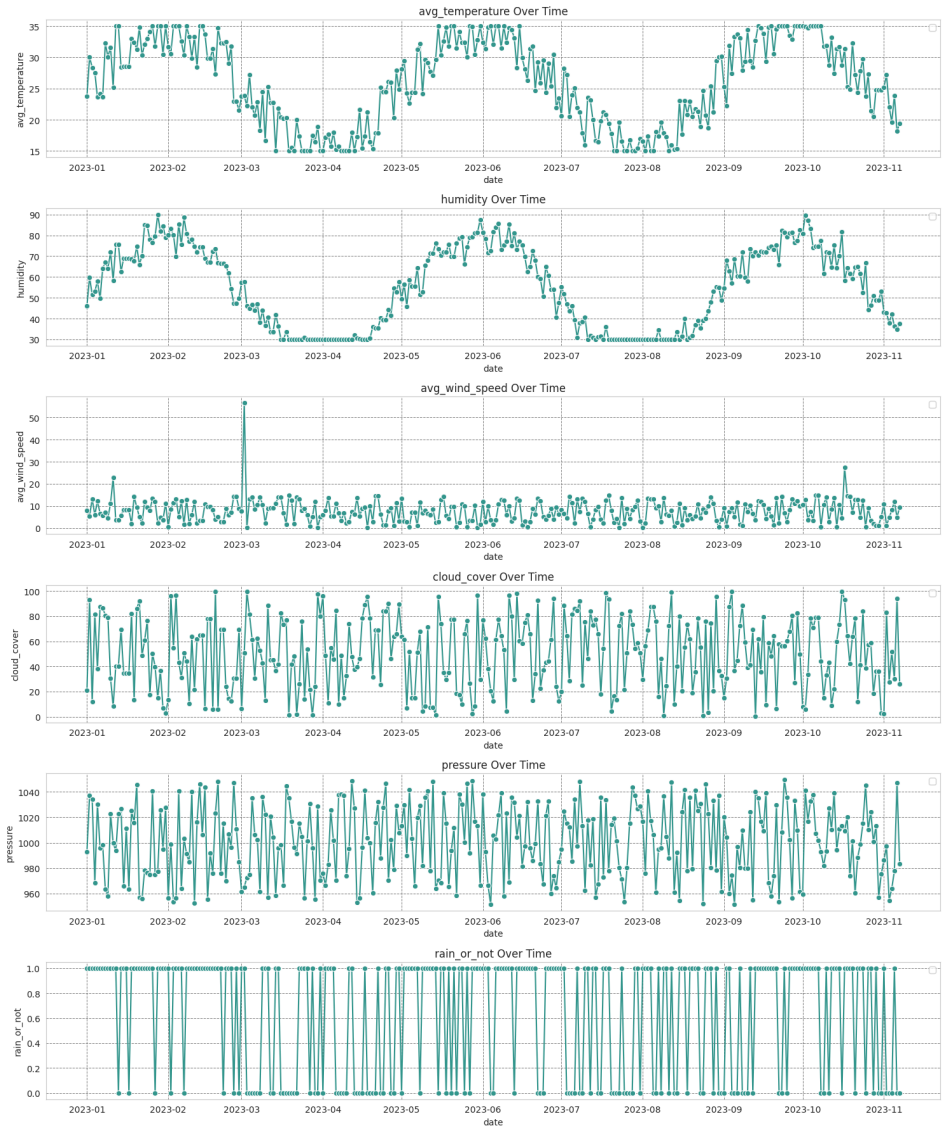
Exploratory Data Analysis





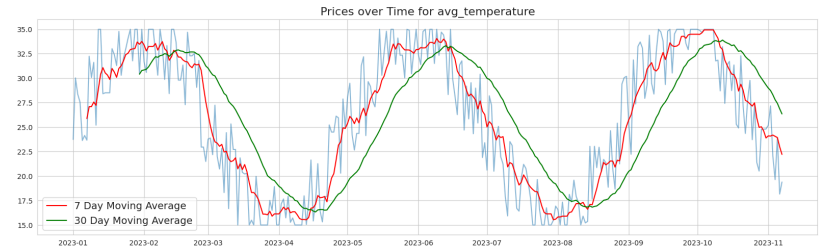
The target variable in this study is **"Rain or Not"**, which is a binary classification label indicating whether it is raining (1) or not (0). The objective of this analysis is to predict this label based on various meteorological features.

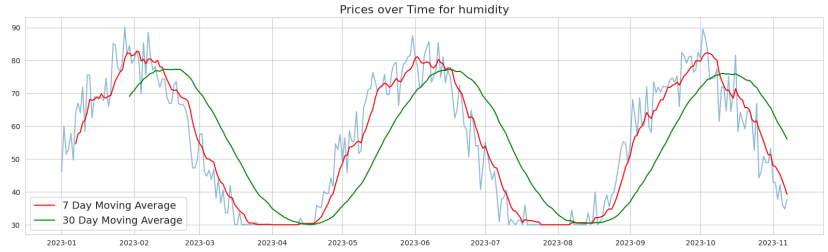
An important characteristic of the dataset is the **class imbalance**, where the majority of the records belong to the **rain** class.

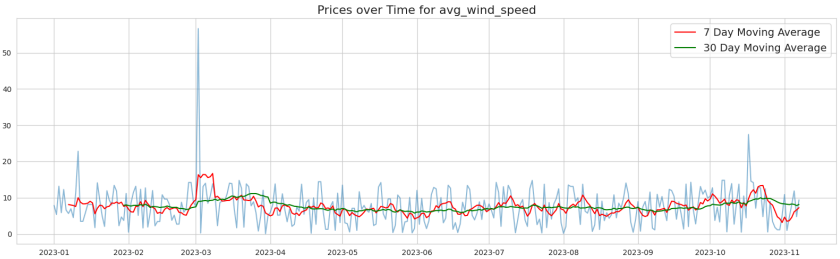


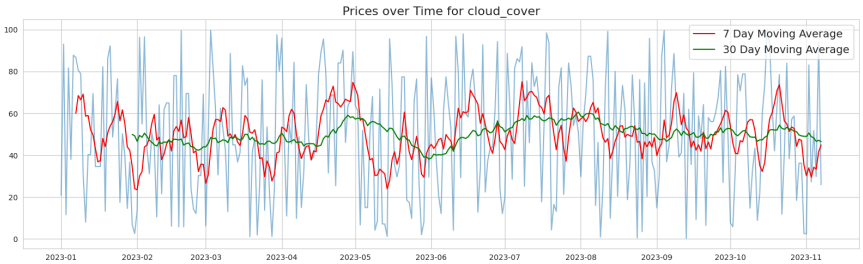
Examining the distribution of each feature over time reveals a clear **sinusoidal pattern** in average temperature and humidity, indicating seasonal variations. However, for other features, such as wind speed, cloud cover, and pressure, no distinct pattern is immediately apparent.

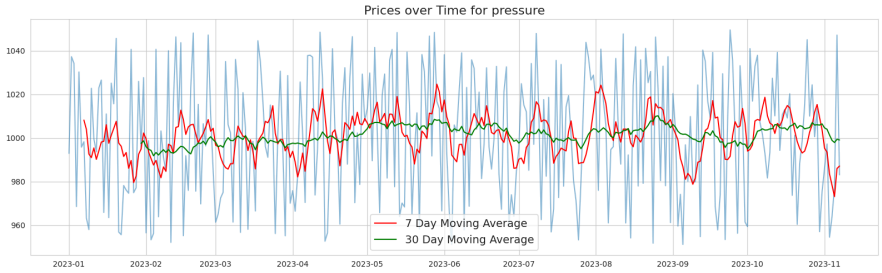
That said, hidden trends or periodic relationships may still exist within these features.





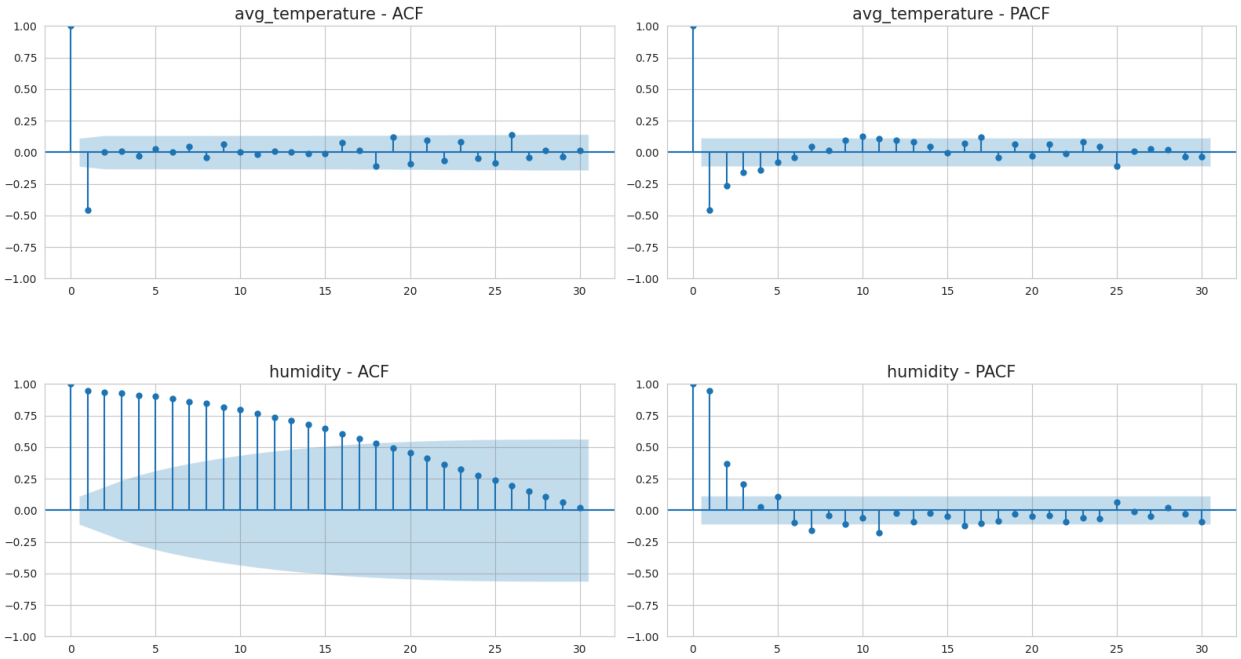


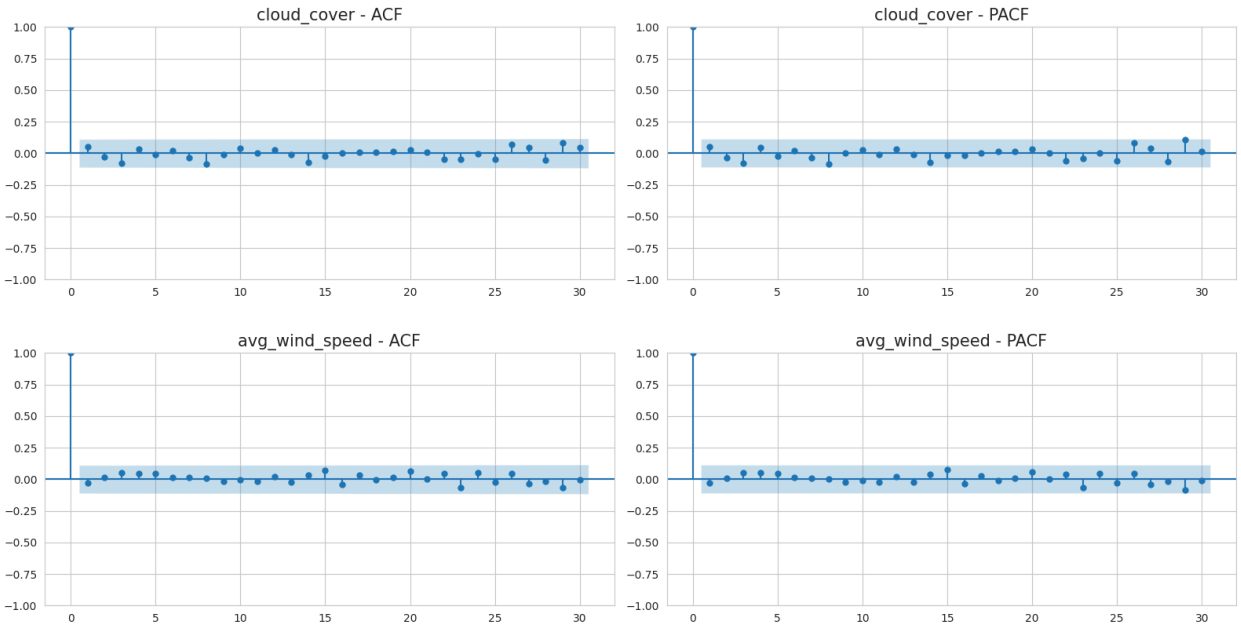




By analyzing the **moving average plots**, we can gain insights into the underlying **trends and seasonality** present in the data. These plots help in identifying long-term variations and smoothing out short-term fluctuations.

To further investigate the **temporal dependencies** in the dataset, we will examine the **Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots**. These statistical tools will help us to determine the presence of significant **lag-based correlations.**

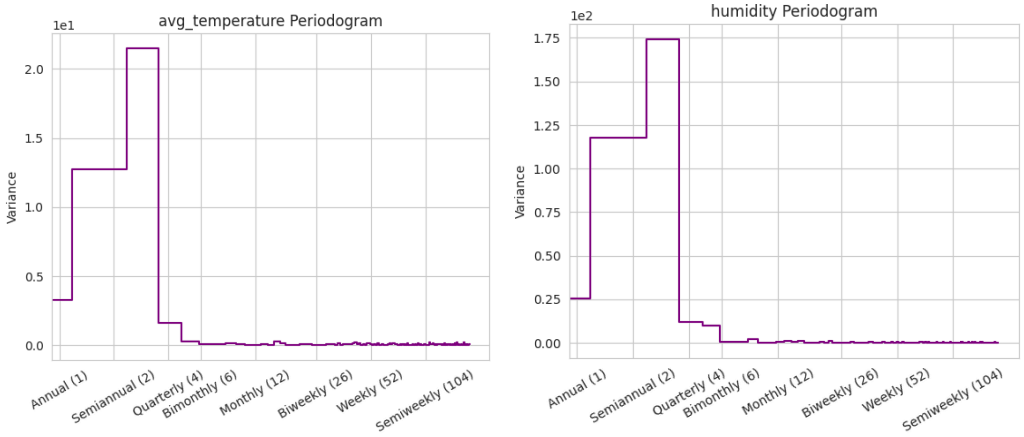


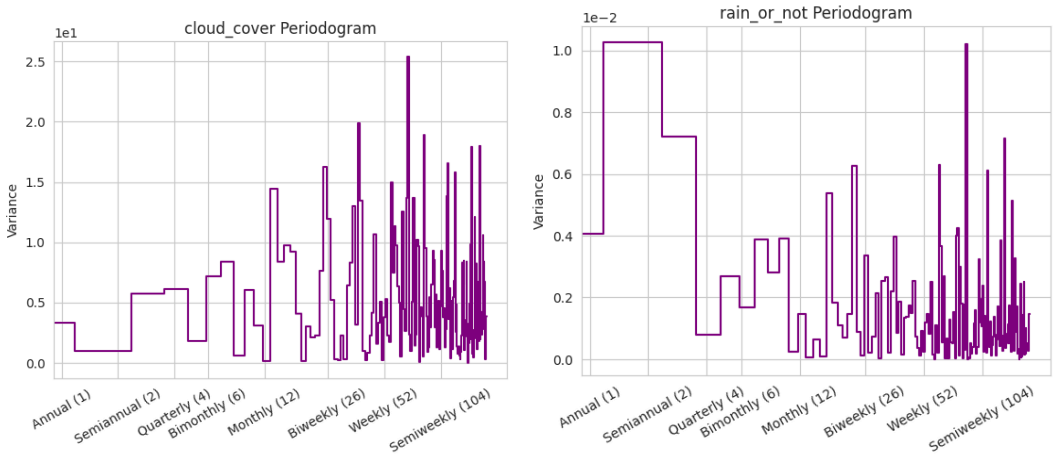


In **Average Temperature** ACF and PACF plot exhibits a significant **negative lag at 1**, with no strong correlations beyond that. This suggests that average temperature does not have a strong autocorrelation structure and may not follow a persistent time-dependent pattern.

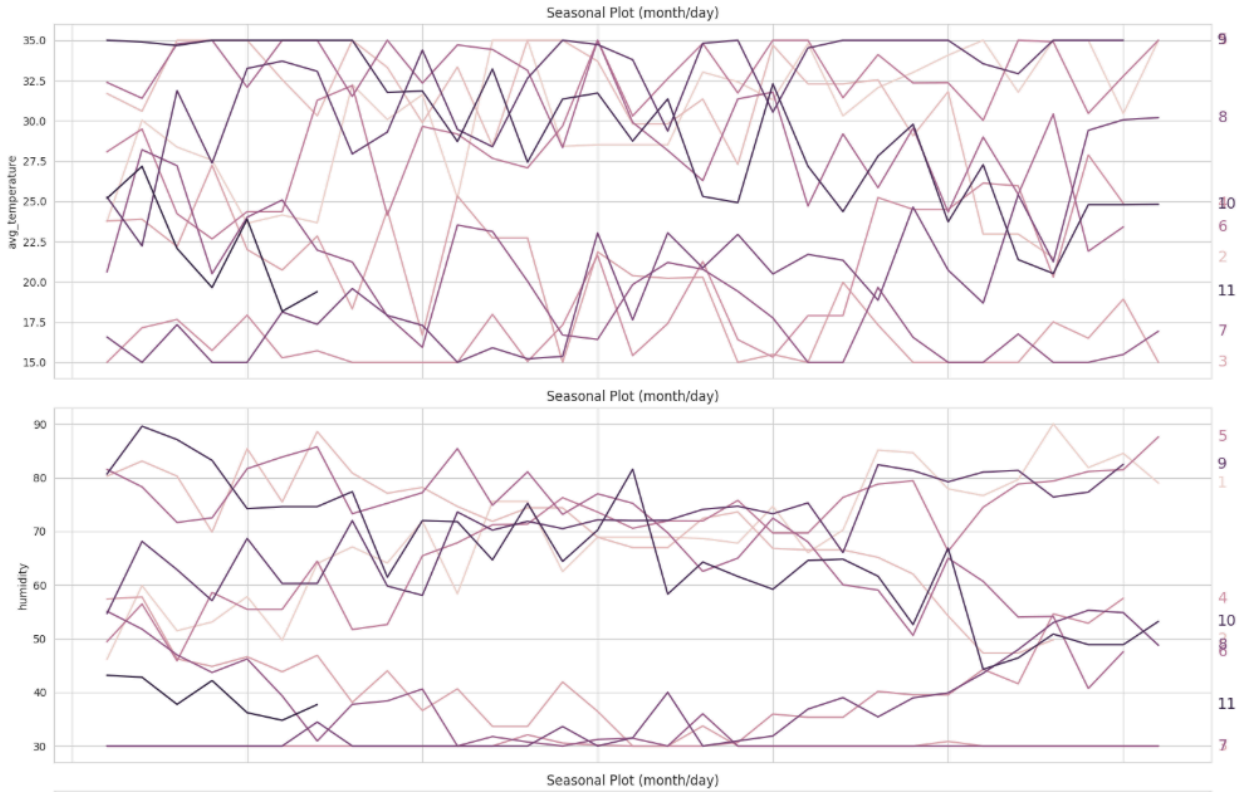
But in **Humidity** the ACF plot shows a **strong, gradually decaying correlation** across multiple lags. This suggests that humidity has a **long-term dependency**.

In other features Both the ACF and PACF plots show weak autocorrelation beyond lag 1. This indicates that they **do not exhibit strong time-series patterns** and might be more random.





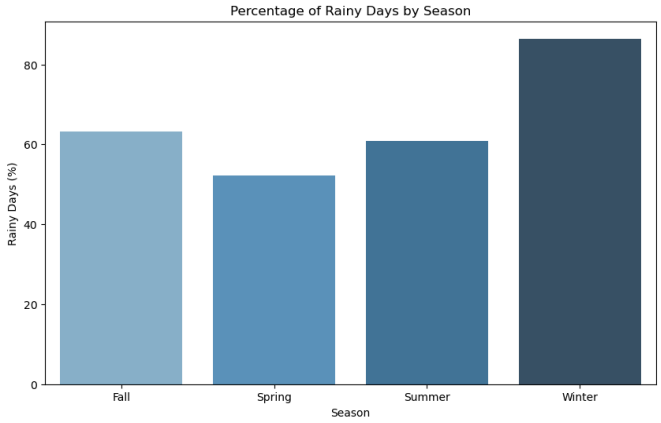
Using these periodogram plots **Temperature & Humidity** show **strong Semiannual and annual cycles**. Other than temperature and humidity there is no single dominant periodicity, and variance is spread across different time scales. **Rainfall** is mostly seasonal but has short-term variations.



As we can see in the above seasonal plots, there are two clear seasonal patterns in **temperature** and **humidity**. However, there are no clear distinct seasonal patterns in other features.

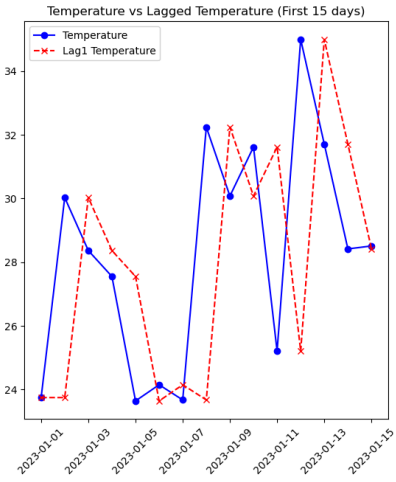
Feature Engineering

Time-Based Features



A categorical season feature (Winter, Spring, Summer, Fall) was created and one-hot encoded to capture seasonal effects.

Window-Based Features

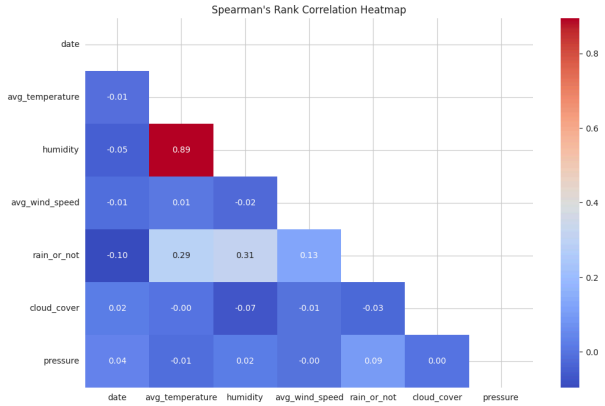


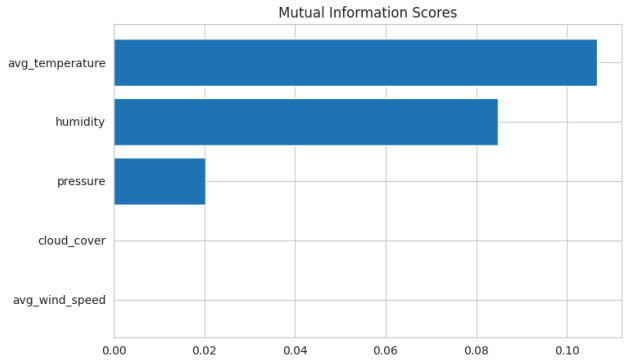
To capture temporal dependencies in the weather data, lag features were created.

Feature Crossing was done to cloud_cover, avg_wind_speed and avg_temperature to create more effective and meaningful features.

Feature Selection with Correlation

We calculated the correlation and mutual information (MI) scores of each feature with respect to the target variable to assess feature importance. These metrics help identify the most influential features in predicting the target value.





ML modeling and hyperparameter tuning

Since we are predicting for the future, we only have future dates available as inputs. Without the historical features we discussed earlier, performing a classification task for future predictions becomes challenging.

To address this, we built **ARIMA models** for each background feature, such as temperature, humidity, and others. These models help us forecast these features into the future, allowing us to use them as inputs for our classification model.

First we in each feature we check whether the time series data was stationary, we performed an Augmented Dickey-Fuller (ADF) test for that.

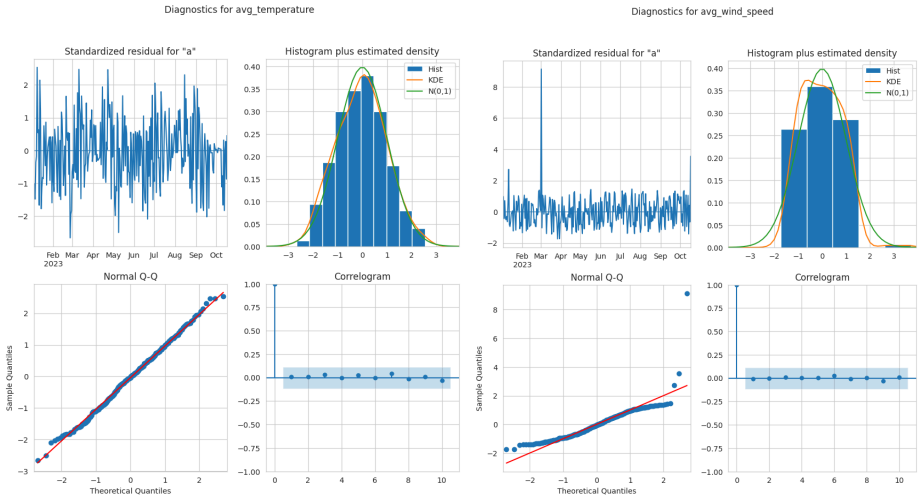
If the p-value > 0.05, the series is non-stationary, and differencing is required.

A nonseasonal ARIMA model is classified as an "ARIMA(p,d,q)" model, where:

- **p** is the number of autoregressive terms,
- **d** is the number of nonseasonal differences needed for stationarity, and
- **q** is the number of lagged forecast errors in the prediction equation.

To determine the best (p, d, q) values:

- We performed a **grid search** over different values of (p, d, q).
- The model was trained for each combination, and we selected the best parameters using the **Akaike Information Criterion (AIC)**



	feature	p	d	q	RMSE	MAE	MAPE
0	avg_temperature	0	2	4	2.859417	2.392525	10.046300
1	humidity	3	2	4	7.485669	6.234228	14.057159
2	avg_wind_speed	4	1	4	5.004301	4.410429	219.378940
3	cloud_cover	3	1	4	26.419321	22.333708	236.465749
4	pressure	2	1	4	25.805348	21.346004	2.154997

Above are the values we calculated in the end.

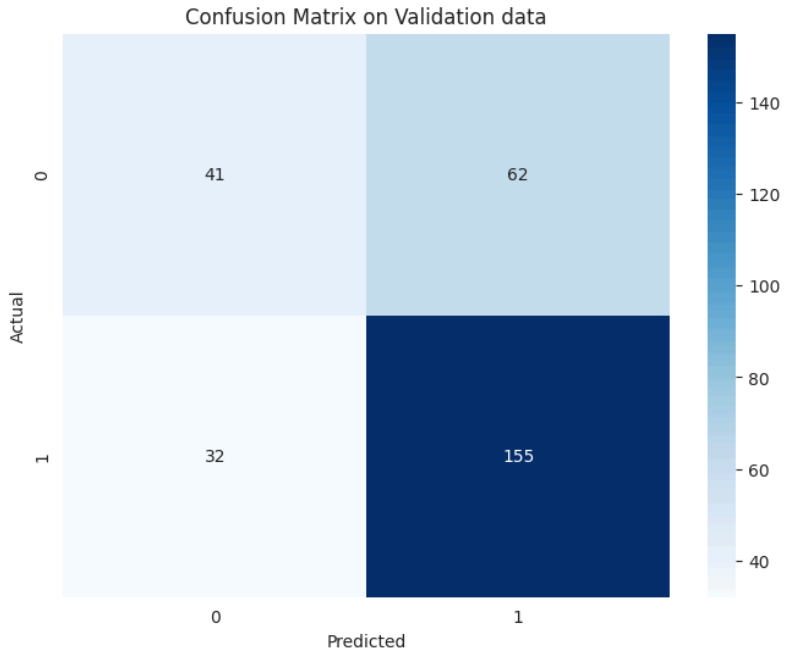
Now, we can perform a better classification task to predict the **rain or not** label. We initially trained three models: **LGBMClassifier, XGBClassifier, and CatBoostClassifier**, ensuring no data leakage. We used **10-fold cross-validation** and selected **F1-score** as the evaluation metric due to class imbalance. Among the three, **CatBoostClassifier** performed the best. We then proceeded with **hyperparameter tuning** to further optimize its performance.

We used **Optuna** for hyperparameter tuning but performed only slight adjustments to avoid overfitting due to the limited data points. During tuning, we also applied **cross-validation** to ensure the model generalizes well and does not overfit.

```
catboost_params_tuned = {
    'loss_function': 'Logloss',
    'learning_rate': 0.0010315109107573211,
    'iterations': 3815,
    'depth': 12,
    'l2_leaf_reg': 0.2798659581346516
}
```

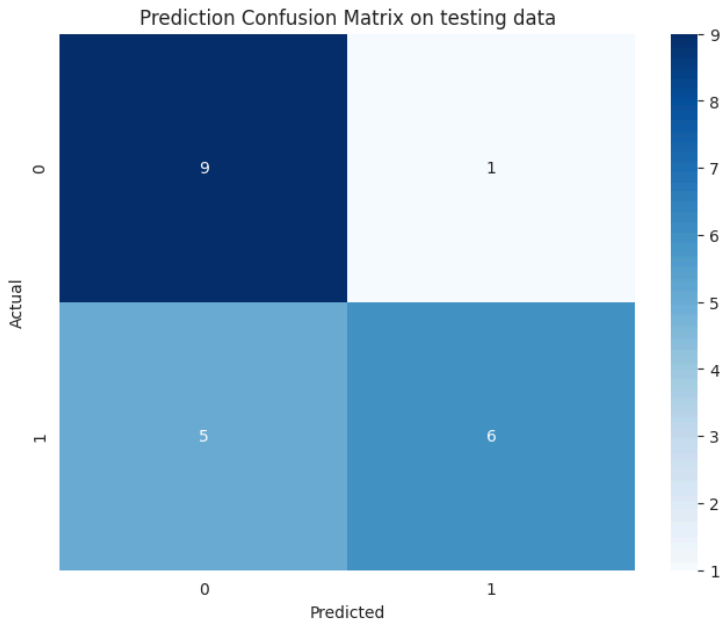
Final model results and evaluation

```
**********
Validation ACC        : 0.67586
Validation Recall     : 82.89 %
Validation F1 Score   : 76.73 %
Validation Precision  : 71.43 %
**********
```



Here, we can see that the model has performed well based on the **evaluation metrics** and **confusion matrix**. Now, let's analyze its predictions for the next **21 days**.

```
**********
Prediction ACC        : 0.71429
Prediction Recall     : 54.55 %
Prediction F1 Score   : 66.67 %
Prediction Precision  : 85.71 %
**********
```



The results are promising, with only **6 out of 21 predictions being incorrect**, indicating a strong generalization capability of the model.