Weather Forecasting Using Machine Learning (Random Forest Model)

1. Introduction: Machine Learning and Supervised Learning

Machine Learning (ML) is a subset of artificial intelligence (AI) that enables systems to learn patterns from data and make decisions with minimal human intervention. Among its paradigms, **Supervised Learning** is one of the most widely used, where the model is trained on labeled data — meaning the input features are associated with known output labels.

In this project, we utilize **supervised classification** to predict the **weather status of the next day** based on current day's meteorological features.

2.Model Definition: Random Forest Classifier

The **Random Forest Classifier** is an ensemble learning method that builds a "forest" of decision trees. Each tree is trained on a random subset of the data and a random subset of features, introducing diversity in learning and reducing overfitting.

Key characteristics of Random Forest:

- **Bagging-based algorithm**: It uses **bootstrap aggregation** where multiple decision trees are trained on different bootstrapped samples of the training data.
- **Feature Randomness**: At each split in the tree, only a random subset of features is considered, which decorrelates the trees and improves generalization.
- **Voting mechanism**: For classification tasks, the output is the majority vote from all trees.
- **Handling of missing values and outliers**: It is robust to noisy data and does not require feature scaling or normalization.
- **Feature Importance**: It provides insight into which features contribute most to prediction.

In this project, we used:

- `random_state=42` : for reproducibility

The Random Forest was chosen after comparing initial results with other models like **XGBoost**, and it performed better on recall and overall balance, especially when paired with SMOTE for resampling

3.Dataset Overview

The dataset used contains daily weather records. After preprocessing, the selected features used for model training were:

- **month**: Extracted from the date; useful for capturing seasonal patterns.
- **day**: Day of the month; helps with monthly weather trends.
- **temperature_2m_mean**: Average daily temperature (in °C).
- **temperature_2m_max**: Maximum temperature for the day.
- **temperature_2m_min**: Minimum temperature for the day.
- **wind_speed_10m_max**: Maximum wind speed measured at 10 meters height.
- **wind_gusts_10m_max**: Maximum wind gusts for the day.
- **wind_direction_10m_dominant**: Dominant wind direction (in degrees).
- **shortwave_radiation_sum**: Total shortwave solar radiation received (in MJ/m^2); a proxy for sunlight exposure.
- **et0_fao_evapotranspiration**: Reference evapotranspiration; reflects potential evaporation and plant transpiration and correlates with weather conditions.

These features were chosen because they collectively represent atmospheric, thermal, and solar conditions that influence future weather behavior — especially the **weather status of the next day**, which is our classification target

4. Model Development Process and Challenges

The model development followed these steps:

a. **Data Cleaning (SQL + Python)**:

- Null values were checked and removed using SQL transactions to maintain data integrity.
- Invalid dates like `'1940-01-01'` were removed explicitly.
- A new label `weather_status` was derived from the `weather_code` using SQL `CASE` logic.

b. **Feature Engineering**:

- Extracted day and month from the `date`.
- Added `next_day_weather_status` using SQL's `LEAD()` function to set it as the supervised target.

c. **Data Filtering**:

To simplify the problem and improve model focus:

- Classes like `'Drizzle'` were removed to reduce noise.

- Rows with rare classes were filtered to focus the classification on dominant weather patterns.

d. **Class Imbalance Problem**:

One of the major challenges was **imbalanced classes** (e.g., 'Cloudy' dominating the data). This was tackled using **SMOTE (Synthetic Minority Oversampling Technique)** to generate synthetic examples of underrepresented classes. This balanced the training data and improved model generalization.

e. **Model Training**:

The data was split into 80% training and 20% testing. The Random Forest Classifier was trained on the resampled data.

f. **Evaluation**:

The model achieved strong accuracy, and the classification report showed improved recall and precision across multiple classes after resampling.