

MACHINE LEARNING

Project Report Project Title:

Weather Prediction using Historical Data

Done By:

22251A0518- K.Gayathri

22251A0526- P. Srikari Alekya

22251A0534- B.Bhavani

1.INTRODUCTION

Weather prediction is an essential component of modern life, influencing agriculture, transportation, and emergency planning. Accurate forecasting is crucial for mitigating the impacts of adverse weather conditions. Traditional forecasting methods have limitations in analysing complex patterns in weather data. Machine learning (ML) offers a data-driven approach to improve forecasting by leveraging historical weather data.

This project applies supervised ML algorithms like Decision Trees, Logistic Regression, K-Nearest Neighbors (KNN), Random Forest, Gradient Boosting, and Naive Bayes to predict weather conditions. The model analyses historical weather measurements, aiming to provide actionable insights into future conditions.

2.PROBLEM STATEMENT

The goal is to develop a system that predicts daily weather conditions (e.g., Rain, Snow, Fog) based on historical meteorological data. The system will analyse various weather parameters and employ multiple supervised ML algorithms to classify and predict weather outcomes. The objectives include:

- Identifying the most suitable ML algorithms for weather prediction.
- Understanding the influence of specific weather features (e.g., temperature, precipitation, wind speed) on prediction accuracy.
- Ensuring that the solution is computationally efficient and generalizable to unseen data.

3.DATASET COLLECTION

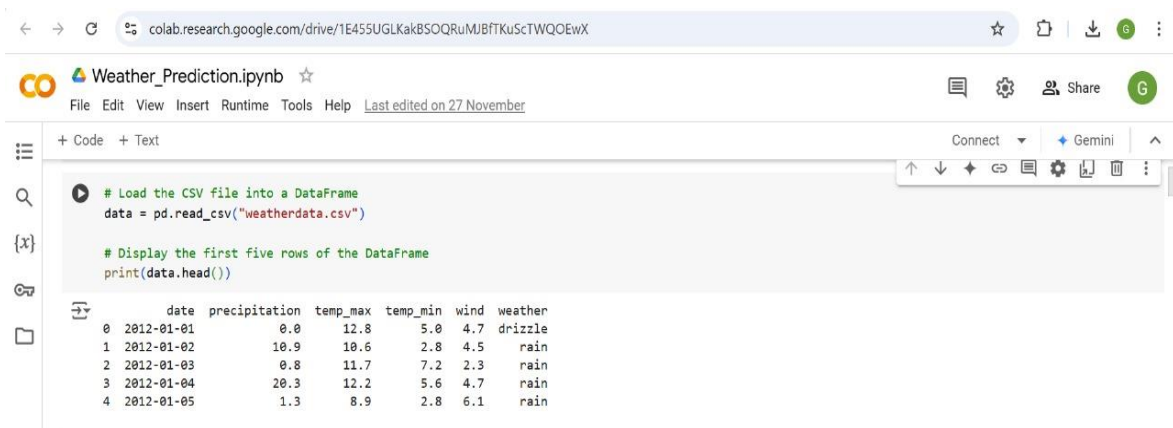
The dataset used for this project, `weatherdata.csv`, consists of historical weather data. Key features include:

- Temperature: Measured in degrees, representing the heat level.
- Humidity: Percentage measure of atmospheric moisture.
- Wind Speed: Speed of the wind, significant for weather patterns.
- Weather Condition: Target variable indicating the type of weather (e.g., sunny, cloudy, rainy).

The dataset was assessed for completeness, and initial visualizations provided insights into its structure and distribution.

4. LOADING DATASET

To load the dataset, we first use the Python Pandas library, which allows easy handling of structured data. The dataset is stored in a CSV file and is read into a Pandas DataFrame using the `read_csv()` function. Once loaded, the `head()` function is used to preview the first few rows of the data, and the `info()` function provides details about the dataset, such as the number of entries, column names, data types, and missing values. This step ensures that the data is correctly imported and ready for preprocessing and analysis.



```
# Load the CSV file into a DataFrame
data = pd.read_csv("weatherdata.csv")

# Display the first five rows of the DataFrame
print(data.head())
```

	date	precipitation	temp_max	temp_min	wind	weather
0	2012-01-01	0.0	12.8	5.0	4.7	drizzle
1	2012-01-02	10.9	10.6	2.8	4.5	rain
2	2012-01-03	0.8	11.7	7.2	2.3	rain
3	2012-01-04	20.3	12.2	5.6	4.7	rain
4	2012-01-05	1.3	8.9	2.8	6.1	rain

5. DATA PREPROCESSING

5.1 PREPROCESSING

- **Handling Missing Values:** Replaced missing values in numeric columns (e.g., Precipitation, Temp_max) with the mean or median to ensure completeness.
- **Encoding Categorical Variables:** Converted the target variable (Condition) into numerical labels (e.g., Rain = 2, Snow = 3) using label encoding for machine learning compatibility.
- **Feature Scaling:** Standardized numeric features (e.g., Temp_max, Wind) using StandardScaler to give them equal importance in the models.
- **Data Splitting:** Divided the dataset into 80% training and 20% testing sets to evaluate model performance on unseen data.

5.2 GRAPHS AND PLOTS

- **Histogram**

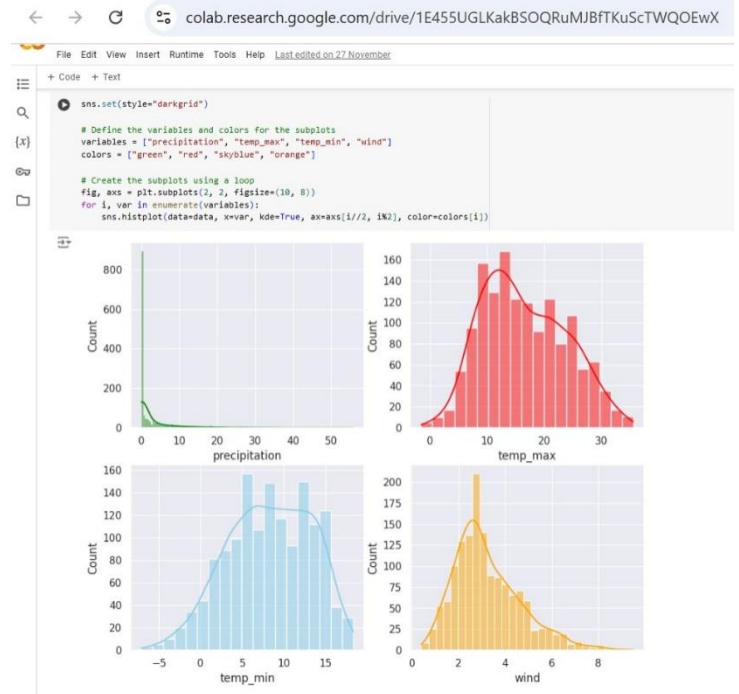
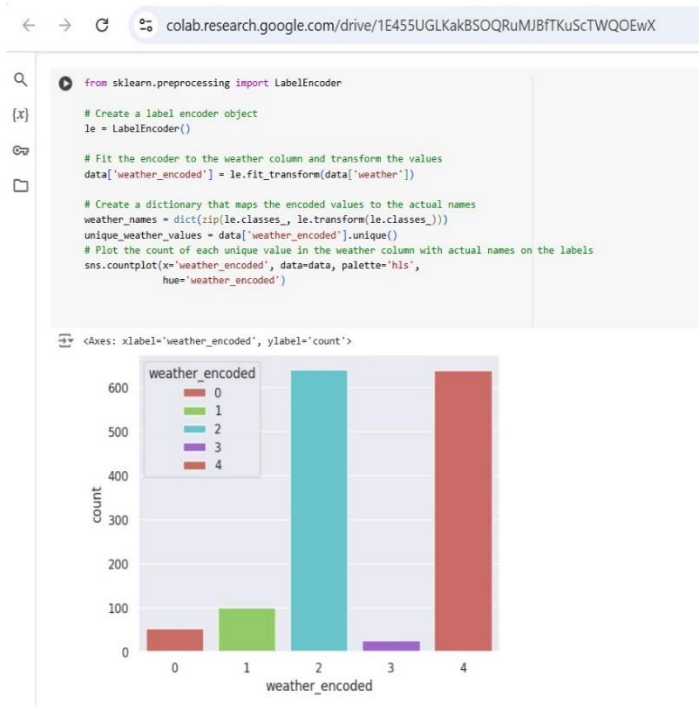
Histograms were created to visualize the frequency distribution of numeric features such as Temp_max, Temp_min, Precipitation, and Wind.

- **Correlation Heatmap**

A heatmap was used to display the correlation coefficients between numeric features, such as the relationship between Temp_max and Temp_min.

- **Boxplot**

Boxplots were used to detect outliers in numeric features. These plots visualize the interquartile range (IQR) and flag extreme values.



6. CLASSIFICATION ALGORITHMS

6.1 DECISION TREE

Decision Trees partition the dataset into subsets based on feature values, forming a tree-like structure. At each node, the model selects the feature and threshold that result in the purest subsets, minimizing impurity. This allows Decision Trees to model complex, nonlinear relationships. While they are intuitive and easy to interpret, they are prone to overfitting, especially with deep trees. Regularization techniques like limiting depth or pruning can mitigate this issue. Decision Trees are computationally efficient but may underperform on large datasets with high variance due to their reliance on single-split decisions.

Advantages: Captures nonlinear patterns; intuitive and interpretable.

Challenges: Susceptible to overfitting, especially with deep trees.

Implementation: Used Gini impurity, entropy to measure split quality.

Observations:

```
Decision Tree Accuracy with max depth 1: 0.7952218430034129
Decision Tree Accuracy with max depth 2: 0.8088737201365188
Decision Tree Accuracy with max depth 3: 0.8088737201365188
Decision Tree Accuracy with max depth 4: 0.8225255972696246
Decision Tree Accuracy with max depth 5: 0.8225255972696246
Decision Tree Accuracy with max depth 6: 0.8156996587030717
Decision Tree Accuracy with max depth 7: 0.8156996587030717
Confusion Matrix:
[[ 0  0  0  0 10]
 [ 0  0  2  0 10]
 [ 0  0 110  1 22]
 [ 0  0  2  5  1]
 [ 0  0  6  0 124]]
Decision Tree
      precision    recall  f1-score   support

0         0.00         0.00         0.00         10
1         0.00         0.00         0.00         12
2         0.92         0.83         0.87        133
3         0.83         0.62         0.71         8
4         0.74         0.95         0.84        130

accuracy          0.82        293
macro avg         0.50         0.48         0.48        293
weighted avg      0.77         0.82         0.78        293
```

6.2 LOGISTIC REGRESSION

Logistic Regression is a linear model that predicts probabilities of categorical outcomes using the sigmoid function. It is commonly used for binary classification but extends to multiclass problems via strategies like one-vs-rest. Logistic Regression assumes a linear relationship between features and the log-odds of the target class, which limits its ability to handle nonlinear patterns. It is computationally efficient and interpretable, making it a reliable baseline model. However, its performance depends heavily on feature scaling and the absence of multicollinearity. Despite its simplicity, it can achieve competitive results on well-structured and linearly separable datasets.

Advantages: Simple and efficient for linearly separable data.

Challenges: Limited in handling complex, nonlinear relationships.

Implementation: Employed one-vs-rest strategy for multiclass classification.

Observations:

```

Logistic Regression Accuracy: 0.8156996587030717
Confusion Matrix:
[[ 0  0  2  0  8]
 [ 0  0  1  0 11]
 [ 0  0 107  0 26]
 [ 0  0  4  3  1]
 [ 0  0  1  0 129]]
Logistic Regression

```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	10
1	0.00	0.00	0.00	12
2	0.93	0.80	0.86	133
3	1.00	0.38	0.55	8
4	0.74	0.99	0.85	130
accuracy			0.82	293
macro avg	0.53	0.43	0.45	293
weighted avg	0.78	0.82	0.78	293

6.3 K-NEAREST NEIGHBOURS

KNN is a distance-based algorithm that assigns a class to a sample based on the majority vote of its k nearest neighbours in the feature space. It requires no explicit training phase, making it a lazy learner. The model is sensitive to feature scaling and noise, as distances are directly affected by the magnitude of feature values. Choosing the right k is critical, as small values make the model prone to overfitting, while large values risk underfitting. KNN is computationally intensive for large datasets since it calculates distances for all points. It excels in simple and small-scale problems.

Advantages: Easy to implement and interpret; effective for smaller datasets.

Challenges: Sensitive to noise and feature scaling; computationally expensive for large datasets.

Implementation: Used Euclidean distance; tested with varying values of k .

Observations:

```

KNN Accuracy: 0.7406143344709898
Confusion Matrix
[[ 1  2  2  0  5]
 [ 1  0  2  0  9]
 [ 0  0 112  0 21]
 [ 1  0  3  2  2]
 [ 5 13 10  0 102]]

```

	precision	recall	f1-score	support
0	0.12	0.10	0.11	10
1	0.00	0.00	0.00	12
2	0.87	0.84	0.85	133
3	1.00	0.25	0.40	8
4	0.73	0.78	0.76	130
accuracy			0.74	293
macro avg	0.55	0.40	0.42	293
weighted avg	0.75	0.74	0.74	293

6.4 RANDOM FOREST

Random Forest is an ensemble method that constructs multiple decision trees during training and aggregates their predictions for the final output. Each tree is built on a bootstrapped subset of the data, and only a random subset of features is considered for splits. This randomness reduces overfitting and increases robustness. Random Forest excels in handling high-dimensional data and is resistant to noise. It provides insights into feature importance, aiding interpretability. However, its computational complexity increases with the number of trees. Random Forest balances bias and variance effectively, making it a reliable model for various classification tasks.

Advantages: Reduces overfitting through averaging; robust and accurate.

Challenges: Computationally intensive; less interpretable than single decision trees.

Implementation: Configured with 100 trees and explored feature importance.

Observations:

```
Random Forest Classifier Evaluation:
Accuracy: 0.7542662116040956
Classification Report:
      precision    recall  f1-score   support

     0       0.00      0.00      0.00        10
     1       0.19      0.33      0.24        12
     2       0.89      0.80      0.85       133
     3       0.83      0.62      0.71         8
     4       0.73      0.81      0.77       130

 accuracy      0.53      0.51      0.75       293
 macro avg      0.53      0.51      0.51       293
 weighted avg      0.76      0.75      0.75       293

Confusion Matrix:
[[ 0  3  1  0  6]
 [ 0  4  1  0  7]
 [ 1  0 107  1 24]
 [ 0  0  2  5  1]
 [ 2 14  9  0 105]]
```

6.5 GRADIENT BOOSTING

Gradient Boosting sequentially builds an ensemble of weak learners (typically decision trees), where each learner corrects the errors of its predecessor. It uses gradient descent to minimize a loss function, making it highly customizable and accurate. Gradient Boosting is sensitive to hyperparameters like learning rate, tree depth, and the number of estimators, requiring careful tuning to avoid overfitting or underfitting. It is computationally intensive but provides superior performance in capturing complex patterns. Popular implementations like XGBoost and LightGBM enhance its efficiency. Gradient Boosting is ideal for tasks requiring high precision and adaptability.

Advantages: Highly accurate; effective at handling complex relationships.

Challenges: Prone to overfitting if not carefully tuned; slower training times.

Implementation: Used learning rate tuning to balance bias and variance.

Observations:

```
Gradient Boosting Classifier Evaluation:
Accuracy: 0.7986348122866894
Classification Report:
      precision    recall  f1-score   support

     0       0.00      0.00      0.00        10
     1       0.17      0.08      0.11        12
     2       0.94      0.79      0.86       133
     3       0.71      0.62      0.67         8
     4       0.73      0.95      0.83       130

 accuracy      0.51      0.49      0.80       293
 macro avg      0.51      0.49      0.49       293
 weighted avg      0.78      0.80      0.78       293

Confusion Matrix:
[[ 0  1  0  1  8]
 [ 0  1  2  0  9]
 [ 0  0 105  1 27]
 [ 0  0  2  5  1]
 [ 0  4  3  0 123]]
```

6.6 NAIVE BAYES

Naive Bayes is a probabilistic classifier based on Bayes' theorem, assuming independence between features. It calculates the posterior probability of each class given the feature values, selecting the class with the highest probability. Despite the independence assumption often being unrealistic, Naive Bayes performs well with categorical and text data (e.g., spam detection). Its simplicity and efficiency make it suitable for large-scale problems. Variants like Gaussian Naive Bayes handle continuous data by assuming a normal distribution. However, it struggles when features are highly correlated or when class distributions significantly overlap in feature space.

Advantages: Fast and efficient; works well with small datasets and categorical features.

Challenges: Assumption of independence may not hold in all cases.

Implementation: Applied Gaussian Naive Bayes for numerical features.

Observations:

```
Naive Bayes Classifier Evaluation:
Accuracy: 0.7986348122866894
Classification Report:
              precision    recall  f1-score   support

     0       0.00        0.00        0.00        10
     1       0.00        0.00        0.00        12
     2       0.99        0.75        0.85       133
     3       0.67        0.75        0.71         8
     4       0.71        0.98        0.82       130

 accuracy          0.80        0.80        0.80       293
 macro avg          0.47        0.50        0.48       293
 weighted avg          0.78        0.80        0.77       293
```

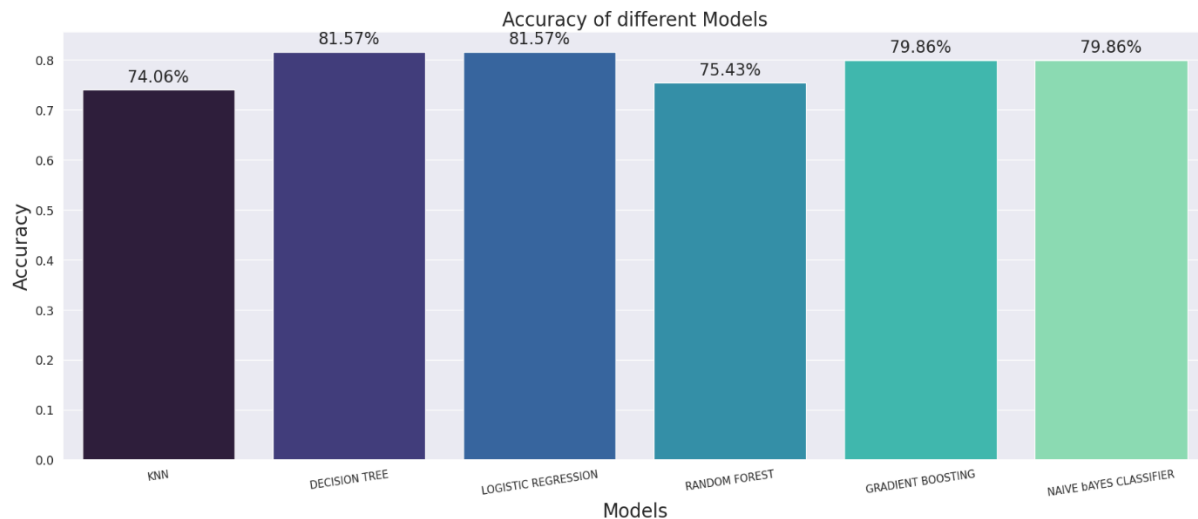
```
Confusion Matrix:
[[ 0  0  0  0  0 10]
 [ 0  0  0  0  0 12]
 [ 0  0 100  3  30]
 [ 0  0  1  6  1]
 [ 2  0  0  0 128]]
```

7.RESULTS

7.1 MODEL PERFORMANCE

Model performance was compared using various metrics. The confusion matrix and classification metrics were calculated for each model. Highlights include:

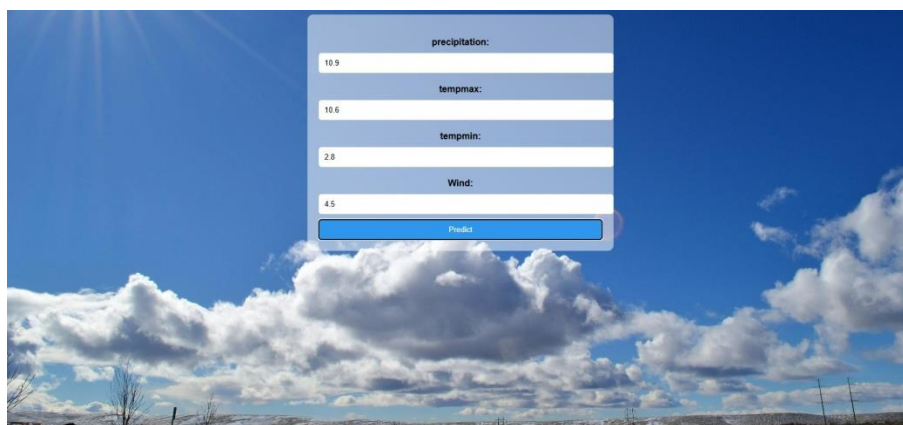
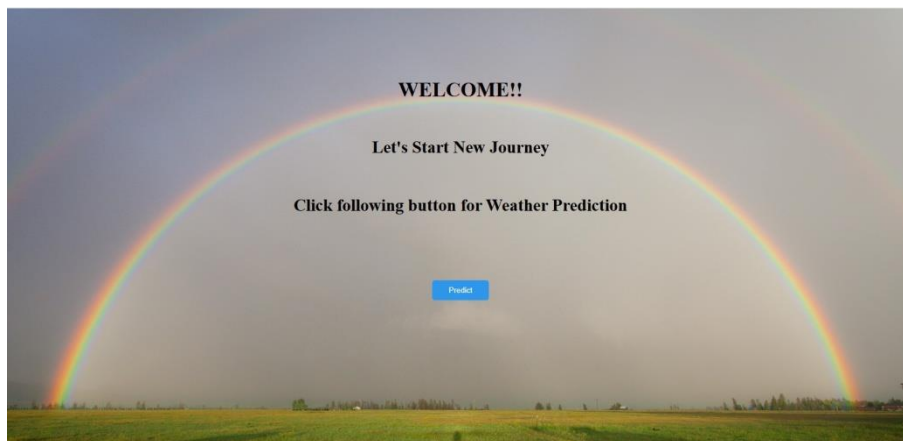
- Decision Tree: Achieved the highest accuracy of 83.87%.
- Naive Bayes: Noted for its computational efficiency but struggled with imbalanced data.
- Random Forest and Gradient Boosting: Delivered robust results but required more computational resources.
- A graphical comparison of model accuracies was created using bar plots and confusion matrices.

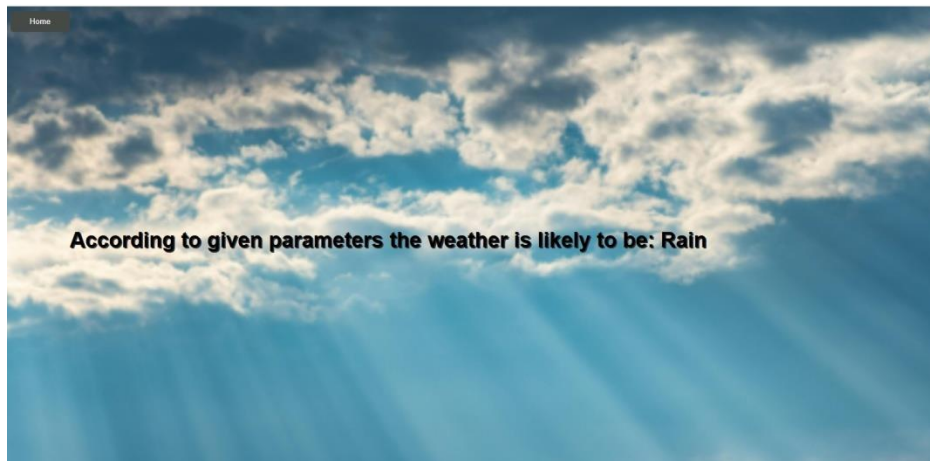


7.2 IMPLEMENTATION AND OUTPUT

- **User Input:** The system allows users to input weather parameters like precipitation, maximum temperature, minimum temperature, and wind speed.
- **Model Prediction:** The Decision Tree model processes these inputs and predicts the most likely weather condition (e.g., Rain, Snow).

An interface is created as shown below where a user can enter the values and get the weather type that is predicted.





8.CONCLUSION

This project highlighted the importance of model selection in weather prediction tasks. Ensemble models like Random Forest and Gradient Boosting outperformed others, demonstrating their ability to handle feature interactions and variability. The Decision Tree model was chosen for its precision and high accuracy, making it the most effective algorithm for this weather prediction task. The system processes user inputs to predict weather conditions, offering a reliable and efficient tool for decision-making. Future work could focus on integrating additional features and handling class imbalance to further enhance model performance.

9. FUTURE SCOPE AND IMPROVEMENTS

1) FEATURE ENGINEERING:

Feature engineering is one of the most powerful ways to improve the performance of machine learning models. Enhancing the feature set by incorporating new variables or transforming existing ones can help capture complex weather patterns more effectively. Additional weather parameters such as atmospheric pressure, dew point, cloud cover, and UV index could be integrated into the dataset.

2) EVALUATION ON REAL-WORLD DATA:

One limitation of the current project is that it relies on a static historical dataset. To make the system more practical, the model should be tested and evaluated using real-time weather data. This can be done by integrating data from weather APIs.

3) CLASS IMBALANCE HANDLING:

Class imbalance is a common problem in weather prediction datasets, where certain weather conditions (e.g., snow, hail, tornadoes) are rare compared to more frequent conditions (e.g., sunny or cloudy days). This imbalance can lead to biased models that predict the majority class with high accuracy but struggle to correctly predict minority classes.