BRAC
UNIVERSITY

Inspiring Excellence

CSE422: Artificial Intelligence

Project Report

Project Title: Weather Type Classification

Group No: 08

Section: 11

ID	NAME
21201327	MD. KAWSAR HABIB
21201532	TASMIN AHMED ONI

Table of Contents

Introduction

This project aims to explore the application of machine learning techniques to classify weather conditions using a synthetic weather dataset. The primary objective is to develop, evaluate, and optimize classification models capable of accurately predicting weather types such as Rainy, Sunny, Cloudy, and Snowy based on various weather-related features.

The project addresses the problem of effectively handling diverse data characteristics, including outliers and categorical variables, while leveraging the dataset for meaningful classification tasks. The inclusion of intentional outliers in the dataset provides a unique challenge, enabling experimentation with outlier detection and data preprocessing techniques critical for real-world applications.

The motivation behind this project stems from the increasing reliance on machine learning for weather forecasting and its broader implications in agriculture, disaster management, and climate research. By simulating these challenges in a synthetic environment, this project serves as an educational tool to refine practical skills in data preprocessing, feature engineering, and model evaluation. Additionally, it underscores the importance of data quality and preparation in achieving robust predictive performance.

Dataset Description

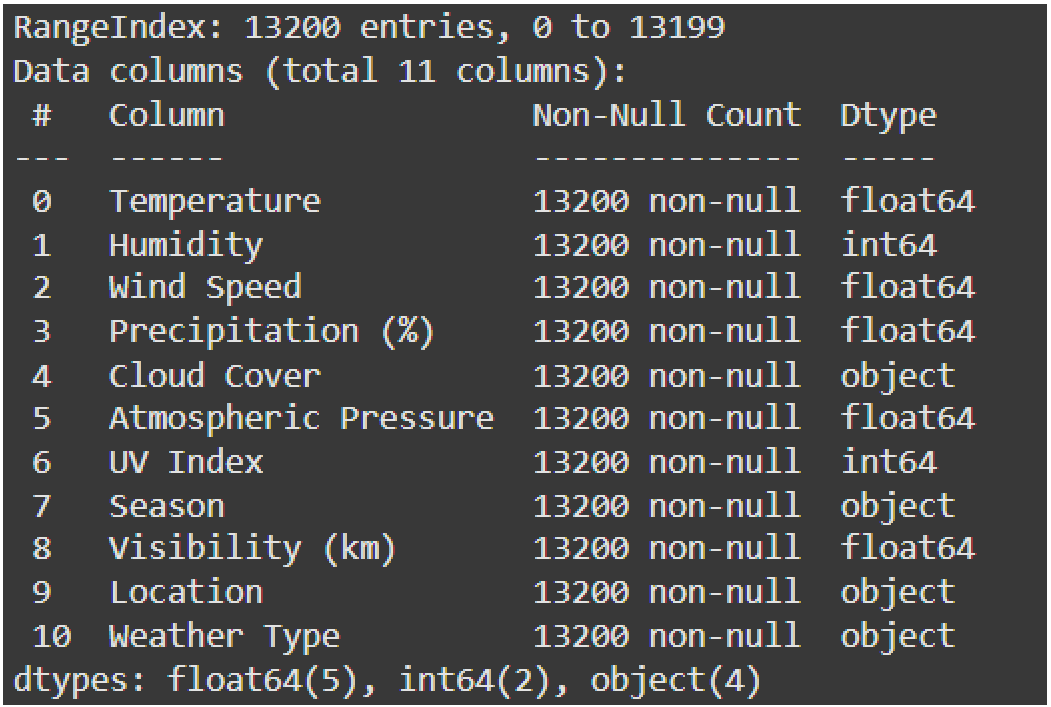
Link: https://www.kaggle.com/datasets/nikhil7280/weather-type-classification/data

Ipynb file: weathertype.ipynb

There are 11 features in the dataset:

- Temperature
- Humidity
- Wind Speed
- Precipitation (%)

- Cloud Cover
- Atmospheric Pressure
- UV Index
- Season
- Visibility (km)
- Location
- Weather Type (target variable)



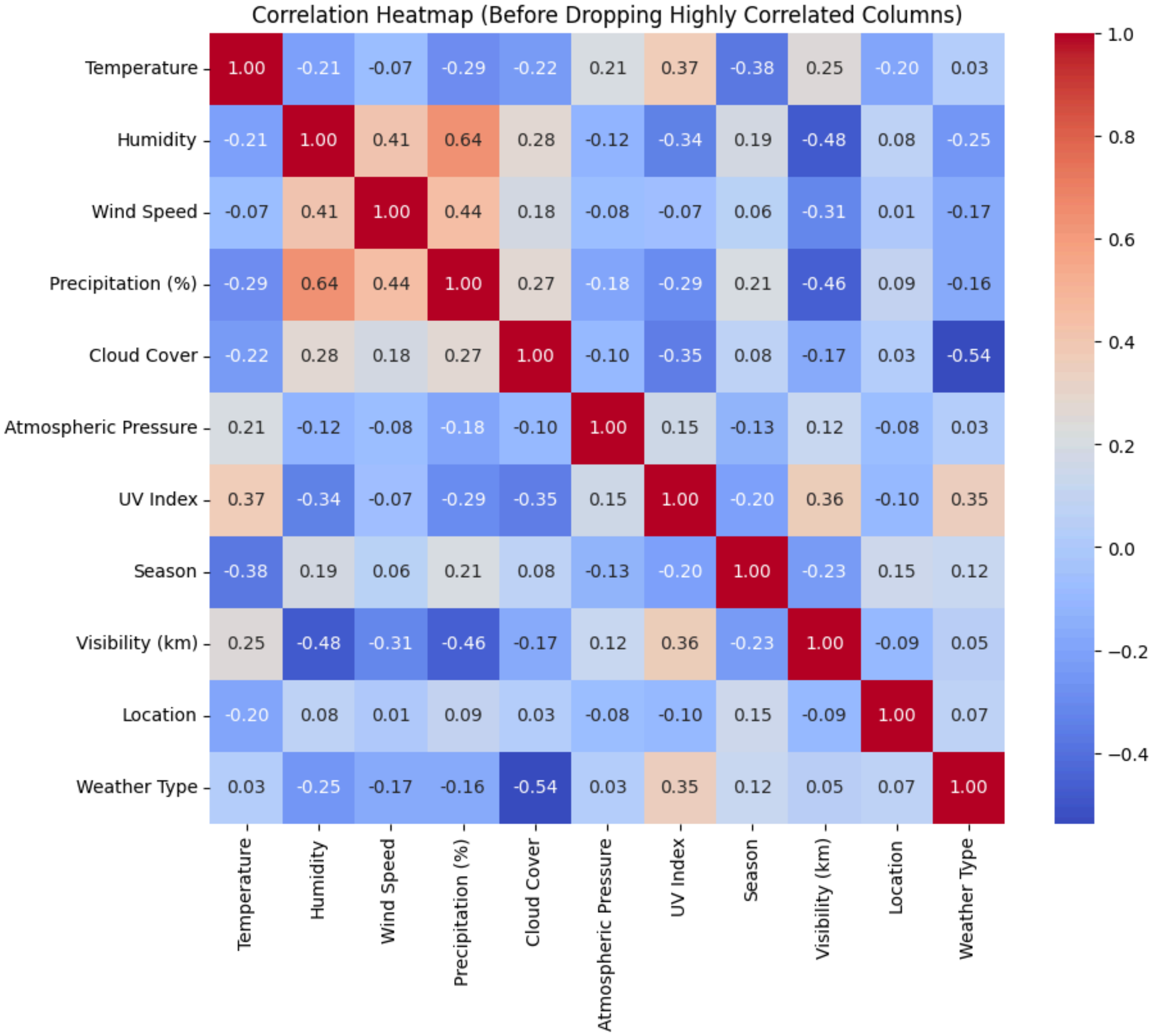
```
RangeIndex: 13200 entries, 0 to 13199
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Temperature           13200 non-null  float64
 1   Humidity              13200 non-null  int64  
 2   Wind Speed            13200 non-null  float64
 3   Precipitation (%)     13200 non-null  float64
 4   Cloud Cover           13200 non-null  object 
 5   Atmospheric Pressure  13200 non-null  float64
 6   UV Index              13200 non-null  int64  
 7   Season                13200 non-null  object 
 8   Visibility (km)       13200 non-null  float64
 9   Location              13200 non-null  object 
 10  Weather Type          13200 non-null  object 
dtypes: float64(5), int64(2), object(4)
```

The above figure is a small representation of the dataset.

This is a classification problem because the target variable, Weather Type, is categorical and classifies the data into specific categories (Rainy, Sunny, Cloudy, and Snowy). There are 13,200 data points, as seen from the weather_df.info() output.

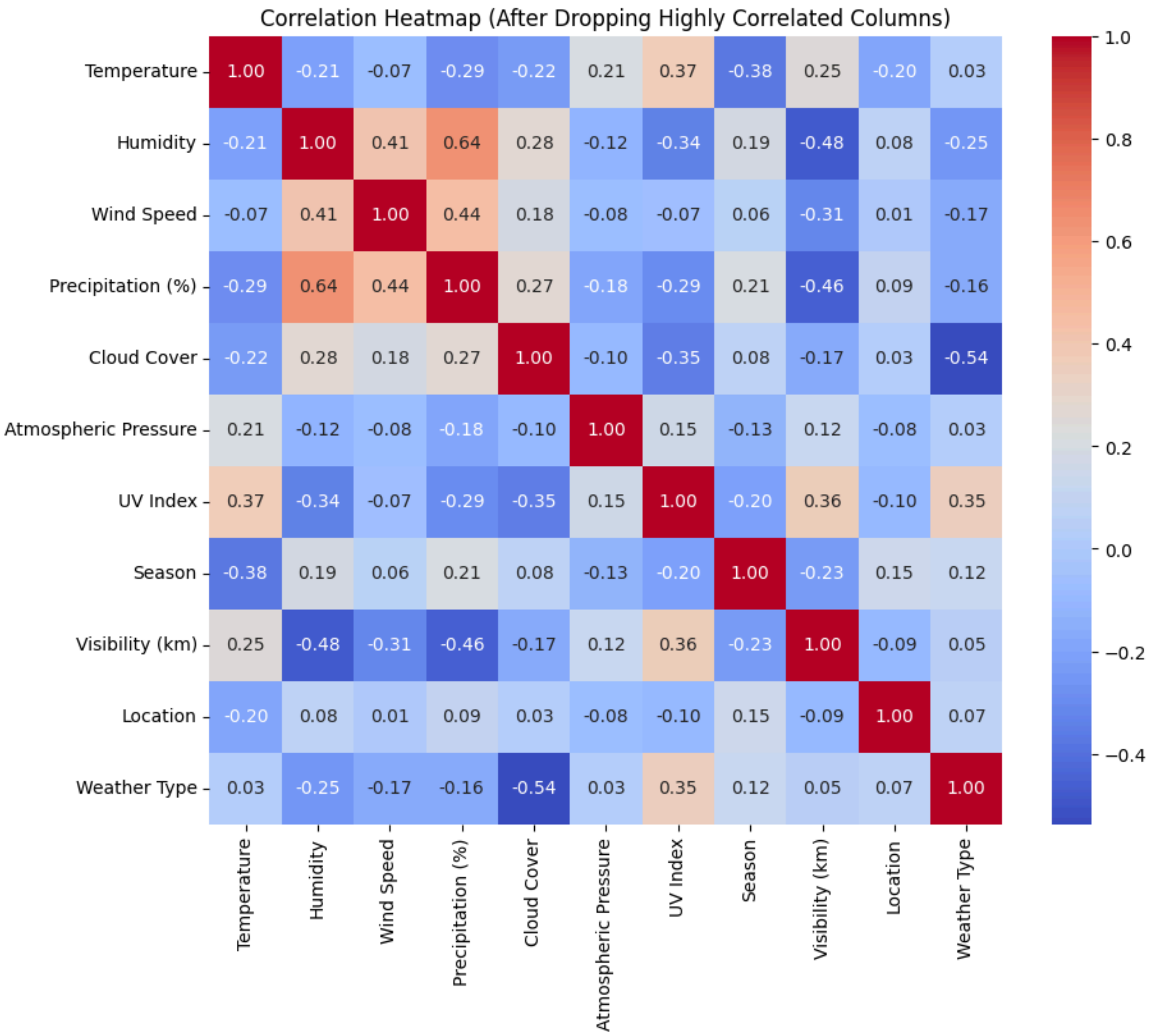
The dataset has a mix of quantitative and categorical features:

- Quantitative Features: Temperature, Humidity, Wind Speed, Precipitation, Atmospheric Pressure, UV Index, Visibility.
- Categorical Features: Cloud Cover, Season, Location, Weather Type (target variable).



	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Location	Weather Type
Temperature	1.00	-0.21	-0.07	-0.29	-0.22	0.21	0.37	-0.38	0.25	-0.20	0.03
Humidity	-0.21	1.00	0.41	0.64	0.28	-0.12	-0.34	0.19	-0.48	0.08	-0.25
Wind Speed	-0.07	0.41	1.00	0.44	0.18	-0.08	-0.07	0.06	-0.31	0.01	-0.17
Precipitation (%)	-0.29	0.64	0.44	1.00	0.27	-0.18	-0.29	0.21	-0.46	0.09	-0.16
Cloud Cover	-0.22	0.28	0.18	0.27	1.00	-0.10	-0.35	0.08	-0.17	0.03	-0.54
Atmospheric Pressure	0.21	-0.12	-0.08	-0.18	-0.10	1.00	0.15	-0.13	0.12	-0.08	0.03
UV Index	0.37	-0.34	-0.07	-0.29	-0.35	0.15	1.00	-0.20	0.36	-0.10	0.35
Season	-0.38	0.19	0.06	0.21	0.08	-0.13	-0.20	1.00	-0.23	0.15	0.12
Visibility (km)	0.25	-0.48	-0.31	-0.46	-0.17	0.12	0.36	-0.23	1.00	-0.09	0.05
Location	-0.20	0.08	0.01	0.09	0.03	-0.08	-0.10	0.15	-0.09	1.00	0.07
Weather Type	0.03	-0.25	-0.17	-0.16	-0.54	0.03	0.35	0.12	0.05	0.07	1.00

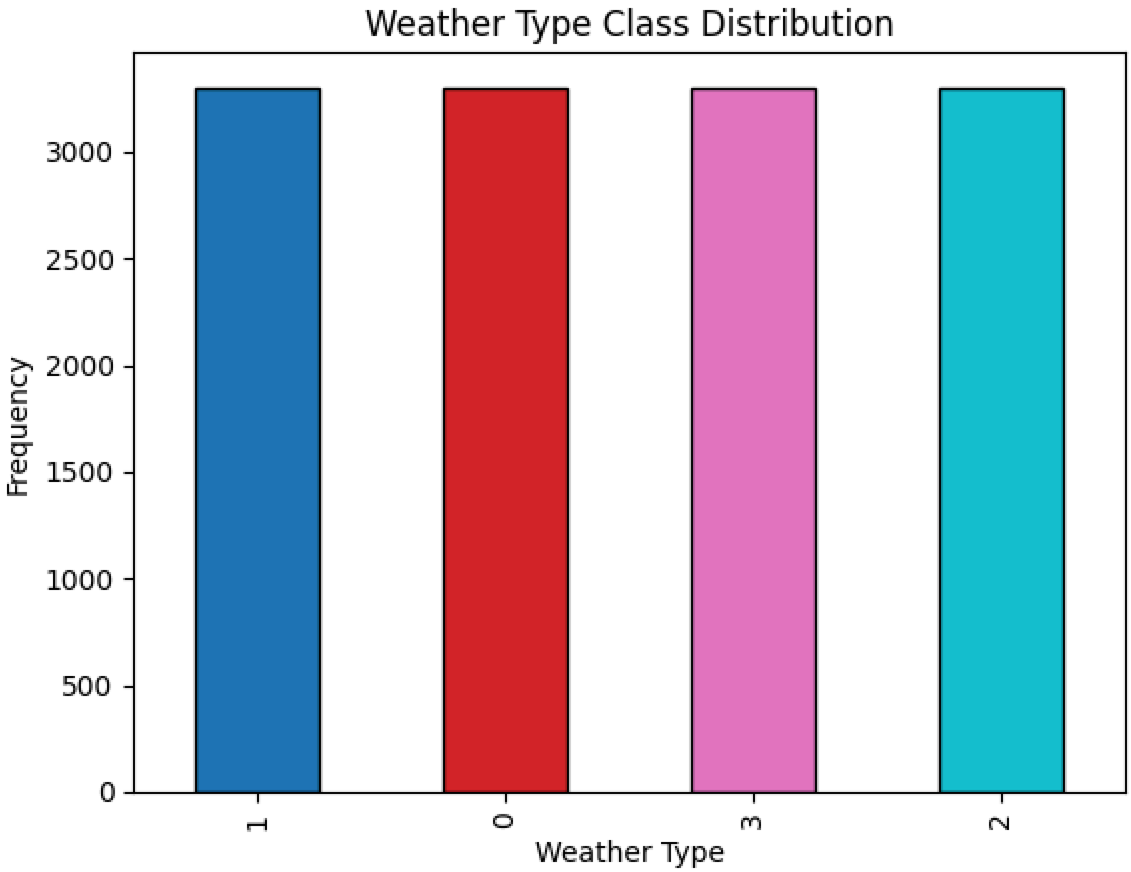
Above Picture is the correlation(before Dropping) of all the features (input and output features)



Above Picture is the correlation(After Dropping) of all the features (input and output features)

We implemented correlation heatmap, which represents the correlation matrix. It’s used to discover relationships between variables, select features for the model, detect multicollinearity, and understand key drivers.

The dataset has an equal number of instances(3300 Instance) for all unique classes in the "Weather Type" output feature.



Above graph is the bar chart

Dataset Pre-Processing

Encoding

As the dataset has no Null values that is why we directly jumped into the encoding part. So, Label encoding is used in the dataset because it is simple and efficient. It converts each category in columns like "Cloud Cover," "Season," "Location," and "Weather Type" into a unique number. This method works well because these columns have a limited number of categories, like "clear," "partly cloudy," or "overcast" for "Cloud Cover." It's also a good choice for models like decision trees or random forests, which can handle these numbers directly without any issues.

Correlation

In our dataset, no columns are highly correlated with each other based on the specified threshold(threshold = 0.9). So, we didn't have to drop any columns.

Scaling

To ensure our models perform more efficiently and effectively, we have used a scaling technique: StandardScaler.
The StandardScaler removes the mean and scales the data to unit variance, helping in cases where the algorithm predicts based on the weighted relationships formed between data points.

Split Train Test

Then we splitted the sections for train and test. For training, we kept 70% of the data set and the rest 30% for the test. The variables that are used, which are mainly datasets for training and testing are given as follows:

- X_train
- X_test
- y_train
- y_test

Model Selection

Random Forest Regressor:

We selected this ensemble model because it combines multiple decision trees to produce a more robust and accurate prediction. It's particularly good at handling overfitting.

Decision Tree:

This model offers a flowchart-like structure that helps us make decisions based on the historical data. It works by breaking down our dataset into smaller subsets while at the same time an associated decision tree is incrementally developed.

Logistic Regression:

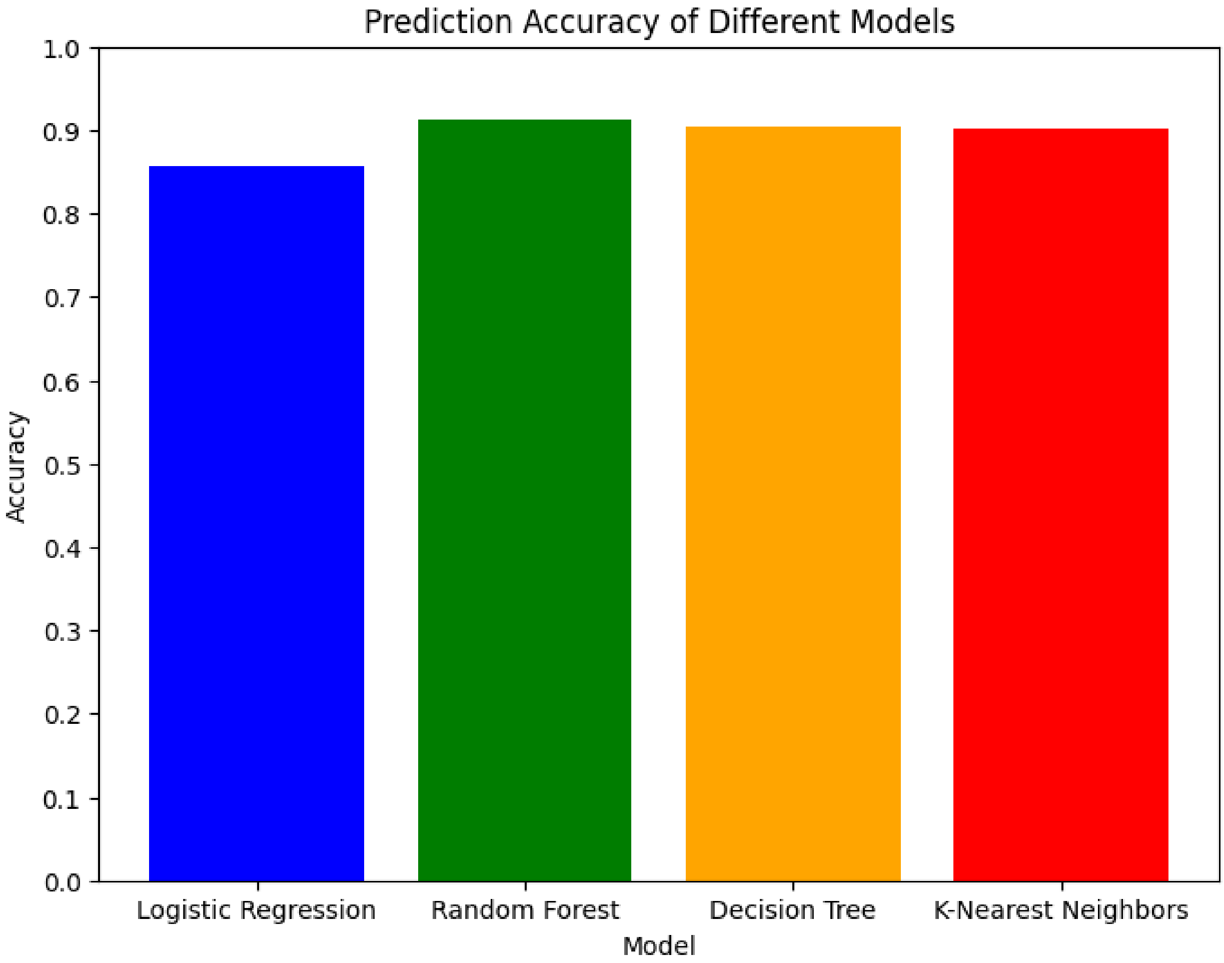
We have used the Logistic Regression model for its ability to model binary outcomes by establishing a relationship between the independent variables (such as volume, open price, and other relevant features) and the probability of a particular class or event occurring (e.g., whether the close price will increase or decrease). As a classification model, Logistic Regression provides a foundation for understanding the likelihood of events, making it useful in scenarios involving categorical outcomes. It serves as a valuable tool for performance benchmarking and comparison with more complex models in our process.

KNeighborsRegressor:

It was selected for its ability to find natural groupings of the data. By examining the 'k' closest neighbors, this model predicts the profit based on the proximity of similar historical data points.

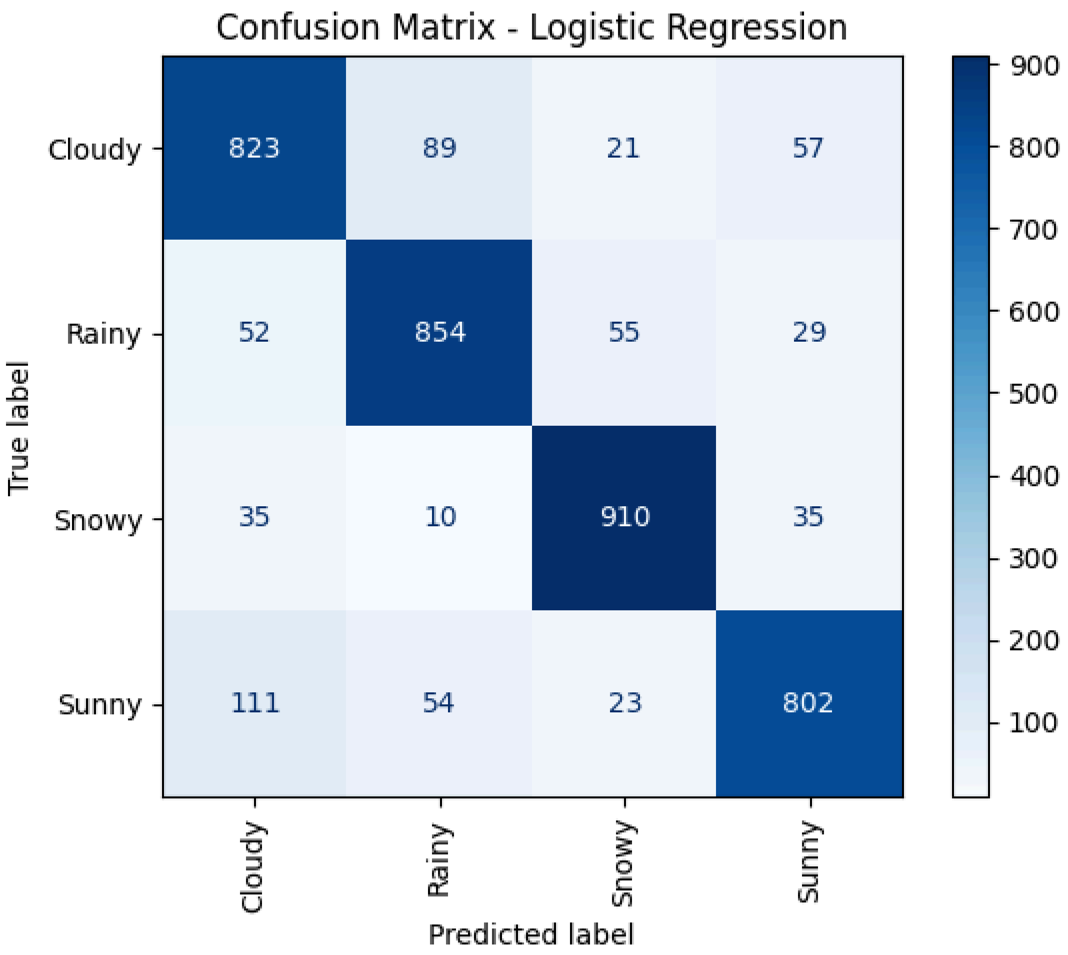
Result

Now, finally after model training and testing we established some common grounds. We could say which model is better to use for prediction analysis. Lets us go through the details



Bar chart showcasing prediction accuracy of all models

Logistic Regression Confusion matrix:



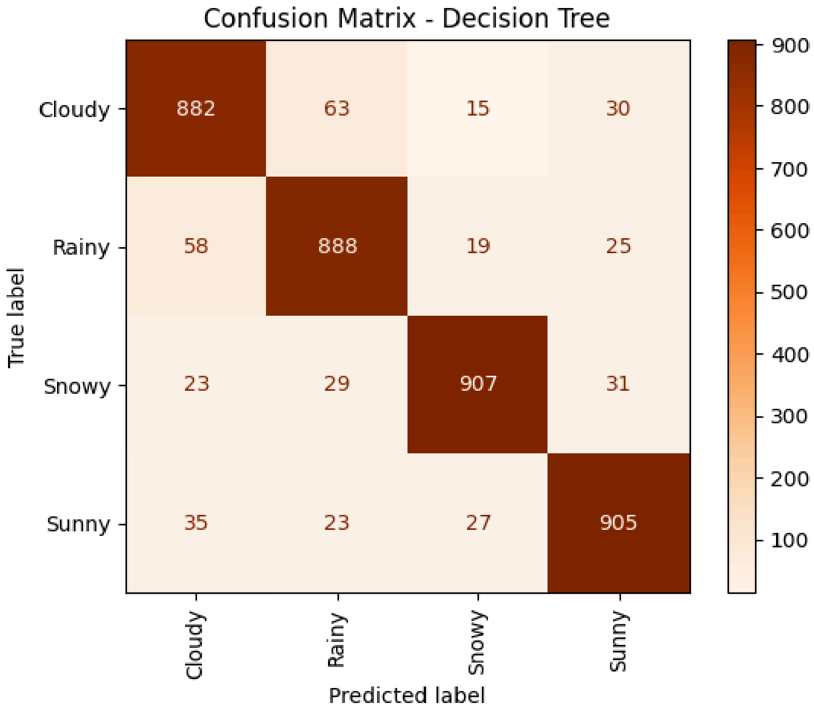
Accuracy: 0.8558080808080808

Precision: 0.8562312069373538

Recall: 0.8558080808080808

F1-Score: 0.855677503311376

Decision Tree Confusion matrix:



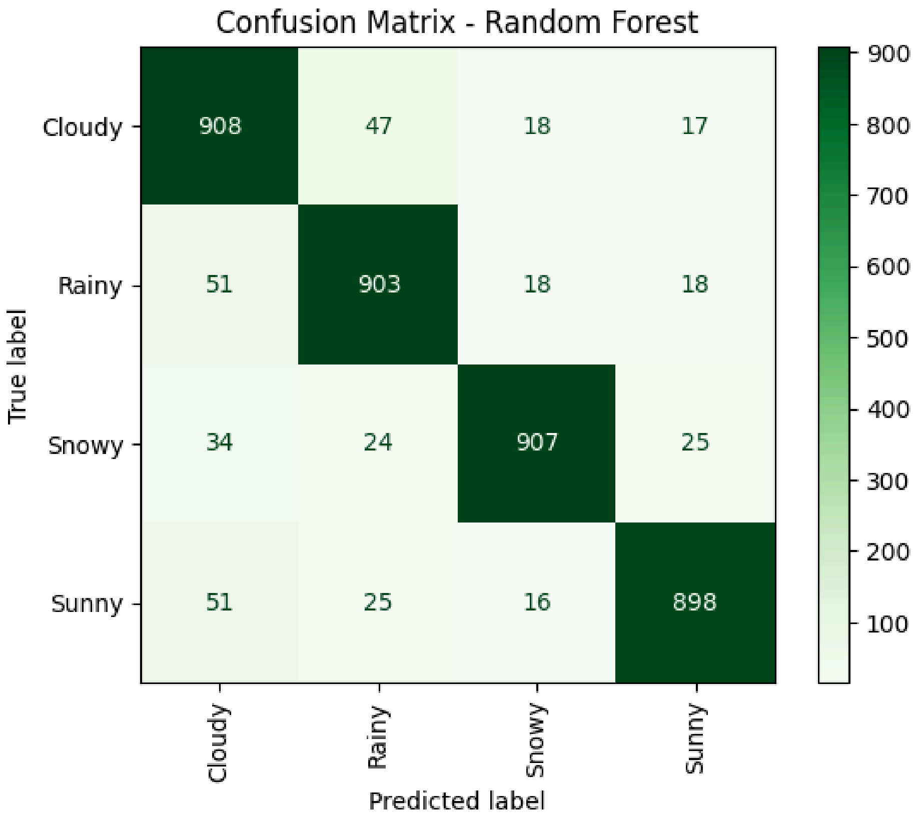
Accuracy: 0.9045454545454545

Precision: 0.9048284862442157

Recall: 0.9045454545454545

F1-Score: 0.904644596597514

<u>Random Forest Confusion matrix:</u>



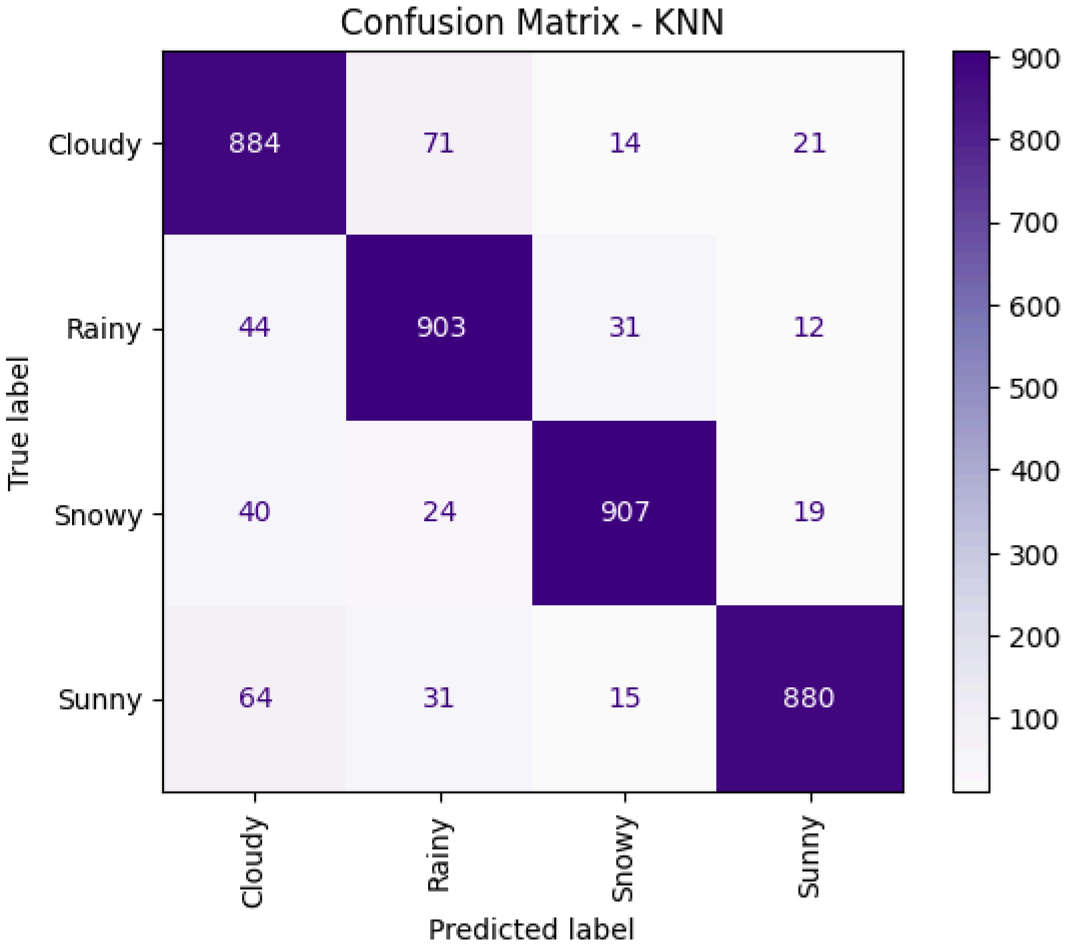
Accuracy: 0.9131313131313131

Precision: 0.9141955188473782

Recall: 0.9131313131313131

F1-Score: 0.9133802386360708

KNN Confusion matrix:



Accuracy: 0.9025252525252525

Precision: 0.90407465161879

Recall: 0.9025252525252525

F1-Score: 0.9028814502773103

Conclusion

	Accuracy	Precision	F1-Score	Recall
Logistic Regression	85.58%	85.62%	85.56%	85.58%
Decision Tree	90.45%	90.48%	90.46%	90.45%
Random Forest	91.31%	91.41%	91.33%	91.31%
KNN	90.25%	90.40%	90.28%	90.25%

The table shows that the Random Forest model performed the best, with the highest accuracy (91.31%), precision, F1-score, and recall, making it the most effective choice for this dataset. The Decision Tree and KNN models also performed well, with accuracies of 90.45% and 90.25% respectively and could serve as good alternatives. However, Logistic Regression had the lowest performance across all metrics, with an accuracy of 85.58%. Overall, Random Forest is the recommended model for this dataset.