DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
BACHELORS IN COMPUTER SYSTEMS ENGINEERING
Course Code: CS-324
Course Title: Machine Learning

Complex Engineering Problem

TE Batch 2022, Spring Semester 2025
Grading Rubric
TERM PROJECT

Group Members:

Student No.	**Name**	**Roll No.**
S1		
S2		
S3		

CRITERIA AND SCALES				**Marks Obtained**		
				S1	**S2**	**S3**
Criterion 1: Does the application meet the desired specifications and produce the desired outputs? (CPA-1, CPA-2, CPA-3) **[8 marks]**						
1	2	3	4			
The application does not meet the desired specifications and is producing incorrect outputs.	The application partially meets the desired specifications and is producing incorrect or partially correct outputs.	The application meets the desired specifications but is producing incorrect or partially correct outputs.	The application meets all the desired specifications and is producing correct outputs.			
Criterion 2: How well is the code organization? **[2 marks]**						
1	2	3	4			
The code is poorly organized and very difficult to read.	The code is readable only to someone who knows what it is supposed to be doing.	Some part of the code is well organized, while some part is difficult to follow.	The code is well organized and very easy to follow.			
Criterion 3: Does the report adhere to the given format and requirements? **[6 marks]**						
1	2	3	4			
The report does not contain the required information and is formatted poorly.	The report contains the required information only partially but is formatted well.	The report contains all the required information but is formatted poorly.	The report contains all the required information and completely adheres to the given format.			
Criterion 4: How does the student performed individually and as a team member? (CPA-1, CPA-2, CPA-3) **[4 marks]**						
1	2	3	4			
The student did not work on the assigned task.	The student worked on the assigned task, and accomplished goals partially.	The student worked on the assigned task, and accomplished goals satisfactorily.	The student worked on the assigned task, and accomplished goals beyond expectations.			

Final Score = (Criteria_1_score) + (Criteria_2_score) + (Criteria_3_score) + (Criteria_4_score)
= ________________________

____
Teacher's Signature

Table of Content

Introduction

This project aims to utilize a structured weather dataset to implement machine learning algorithms and predict the weather. The data is available in the form of structured tables. The primary goal is to predict the 'Summary' attribute, which represents the weather conditions for a given time, based on various weather-related features such as temperature, humidity, pressure, wind, and precipitation. The dataset includes 27 unique classes in the 'Summary' attribute, making this a multiclass classification problem. Weather prediction plays an important role in various fields, including agriculture, transportation, and event planning. It helps people make informed decisions about their day based on expected weather conditions.

Dataset Description

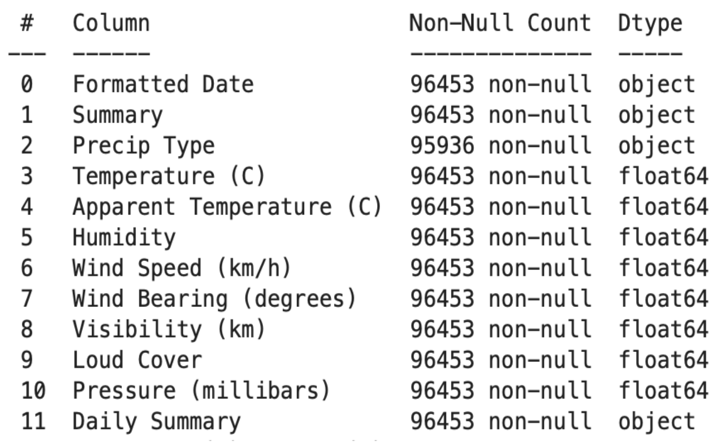
The dataset used in this project is sourced from Kaggle, titled **"Weather Dataset"**. It contains historical weather observations recorded over time, with multiple meteorological features and a categorical label that indicates the overall weather condition. The dataset is structured for a supervised classification task, where the goal is to predict the "**Summary**", which is a categorical variable describing the general weather condition, based on the other weather features. Hence, Summary becomes our target variable**.** The project is a multi-class classification problem since our target label has multiple classes.

Dataset Size: 96,453 rows × 12 columns

Target Label: 'Summary'

Features:

1. **Formatted Date:** Date and Time of the weather.
2. **Precip Type:** Type of precipitation (rain or snow)
3. **Temperature (°C):** Actual temperature
4. **Apparent Temperature (°C):** Feels-like temperature
5. **Humidity:** Relative humidity
6. **Wind Speed (km/h):** Speed of the wind
7. **Wind Bearing (°):** Direction of the wind in degrees
8. **Visibility (km):** Visibility distance
9. **Loud Cover:**
10. **Pressure:** Atmospheric pressure
11. **Daily Summary:** Detailed Summary in text



```
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Formatted Date            96453 non-null  object
 1   Summary                   96453 non-null  object
 2   Precip Type               95936 non-null  object
 3   Temperature (C)           96453 non-null  float64
 4   Apparent Temperature (C)  96453 non-null  float64
 5   Humidity                  96453 non-null  float64
 6   Wind Speed (km/h)         96453 non-null  float64
 7   Wind Bearing (degrees)    96453 non-null  float64
 8   Visibility (km)           96453 non-null  float64
 9   Loud Cover                96453 non-null  float64
 10  Pressure (millibars)      96453 non-null  float64
 11  Daily Summary             96453 non-null  object
```

Data Preprocessing

Removal of Duplicate Rows

The dataset was first checked for duplicate rows, which can introduce bias or overfitting in models if repeated multiple times. They can distort model training by giving certain data points more weight than others. The check identified 24 duplicated entries, which were removed to maintain the integrity and uniqueness of the training data.

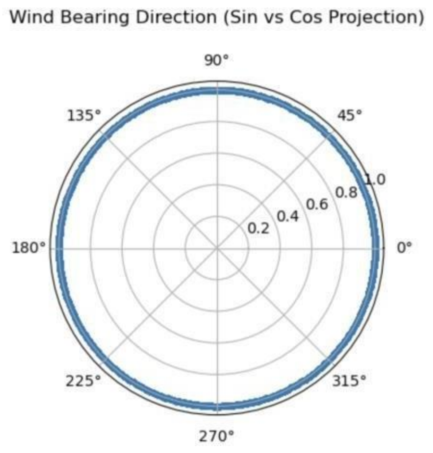
Removal of Unwanted Summary Classes

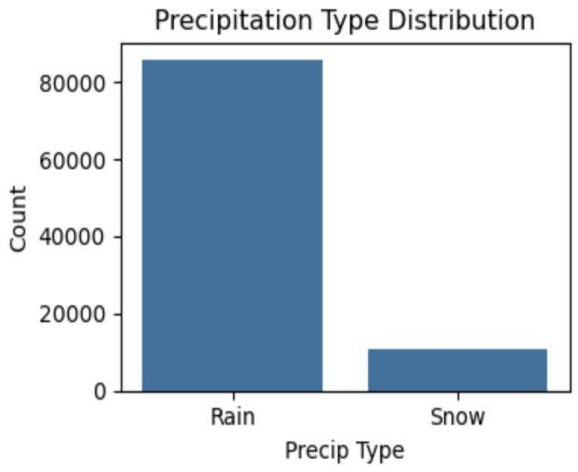
The target variable Summary initially contained 27 weather condition classes. Some of these classes such as "Windy", "Humid and Overcast", "Windy and Foggy", "Windy and Dry", "Dangerously Windy and Partly Cloudy", and "Breezy and Dry" were removed because they were infrequent, rare, and introduced noise due to the lack of sample size. They were chosen since they contained less than 10 samples. Removing these classes helped reduce complexity and improved model focus on the most relevant and well-represented weather summaries, thus enhancing classification reliability and reducing class imbalance.

```
Windy                                   8
Humid and Overcast                      7
Windy and Foggy                         4
Windy and Dry                           1
Dangerously Windy and Partly Cloudy     1
Breezy and Dry                          1
```

Handling Missing Values and Data Type Conversion

1. There were 1281 samples with zeros in the "Pressure" column, which is physically impossible. So, they were replaced with the mean pressure value to correct anomalies and avoid misleading the model.
2. The "Precip Type" column had 517 missing values that were filled with the majority class "rain". This prevented data loss and maintained the class distribution balance. The "Precip Type" feature was then converted to binary by mapping "rain" to 0 and "snow" to 1, converting a categorical variable into a numeric format that is required by machine learning algorithms.
3. For the "Wind Bearing (degrees)" feature, a transformation was applied to convert the directional data into two new features representing the sine and cosine of the wind bearing in radians. This allowed the cyclical nature of wind direction to be captured effectively by the models. The original degree and radian columns were dropped afterward.

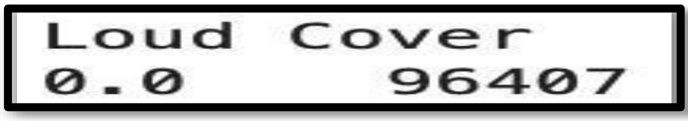




Dropping Redundant Columns

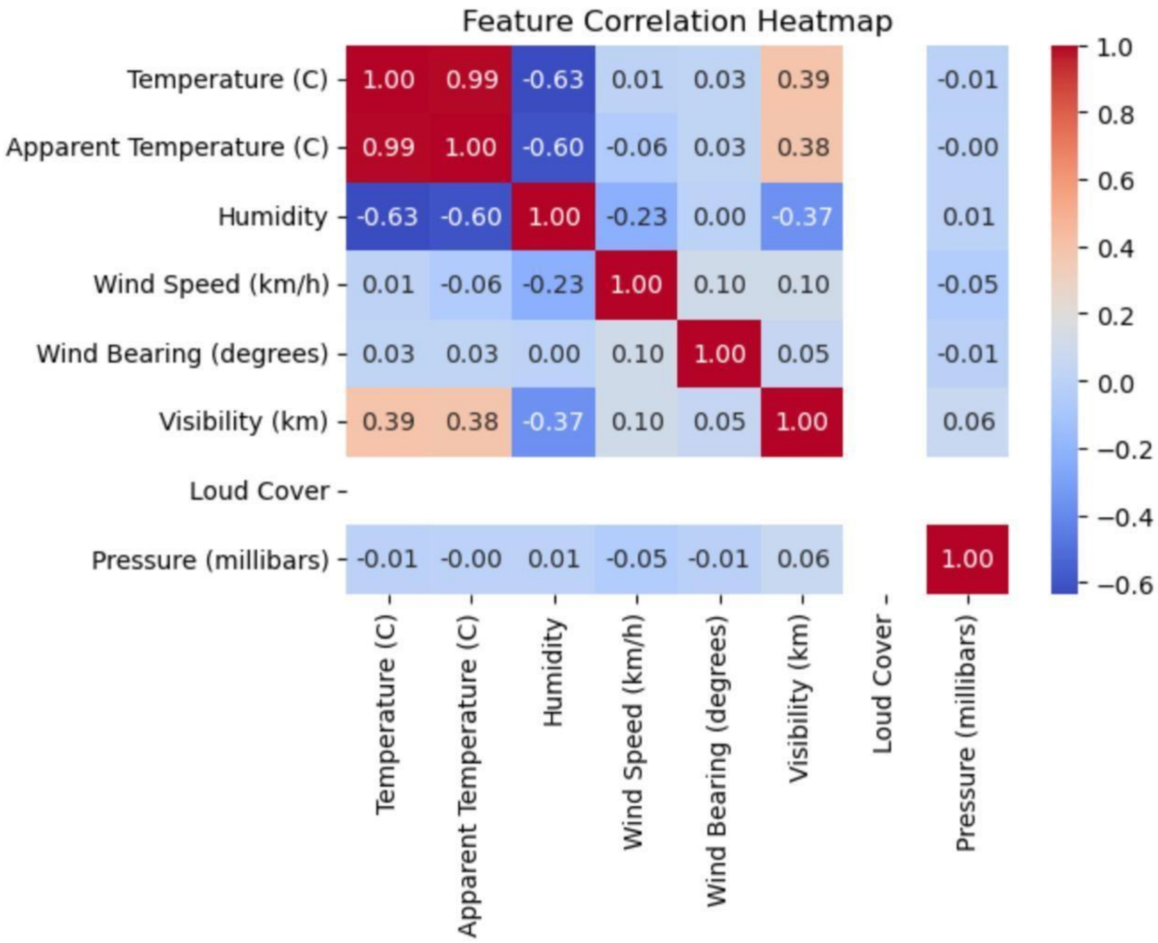
After analyzing all features, it was known that some features were redundant and non-informative. They were increasing complexity and making the features higher dimensional, so they were dropped.

1. **Daily Summary:** The feature has text-based descriptions and overlaps with the target “Summary”.
2. **Formatted Date:** This was dropped because date/time information is not directly needed in the models. Since the target “summary” is derived from meteorological conditions at a given time, other variables already reflect time-specific conditions in a more useful way.
3. **Loud Cover:** The feature was removed from the dataset as it contained a constant value (zero) for all samples, providing no variability or meaningful information for the classification task.



```
Loud Cover
0.0         96407
```

4. **Apparent Temperature (C):** It was dropped because it was highly correlated with "Temperature (C)". The correlation coefficient between these two features was extremely high and close to 1, indicating that they provided nearly identical information. Keeping both could lead to multicollinearity, negatively affecting model performance and interpretability.



	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)
Temperature (C)	1.00	0.99	-0.63	0.01	0.03	0.39		-0.01
Apparent Temperature (C)	0.99	1.00	-0.60	-0.06	0.03	0.38		-0.00
Humidity	-0.63	-0.60	1.00	-0.23	0.00	-0.37		0.01
Wind Speed (km/h)	0.01	-0.06	-0.23	1.00	0.10	0.10		-0.05
Wind Bearing (degrees)	0.03	0.03	0.00	0.10	1.00	0.05		-0.01
Visibility (km)	0.39	0.38	-0.37	0.10	0.05	1.00		0.06
Loud Cover								
Pressure (millibars)	-0.01	-0.00	0.01	-0.05	-0.01	0.06		1.00

Label Encoding

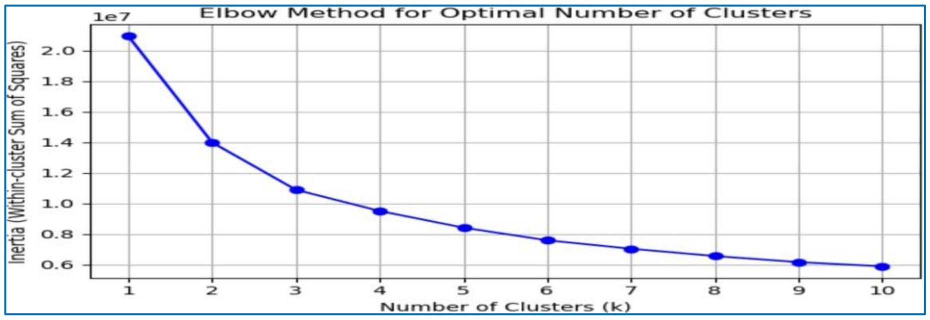
To prepare the target variable for a supervised classification task, the textual descriptions in the "Summary" column were converted into numerical labels using manual label encoding. The new category is called Weather Category. This step is critical for transforming categorical target data into a machine-learning-friendly numeric format, enabling effective multi-class classification. This involved grouping logically similar weather descriptions into three broader categories:

Class 0 – Clear/Partly Cloudy Conditions:
Includes summaries like Clear, Partly Cloudy, Dry, and similar weather types with minimal cloud cover and no precipitation. This has a total of 71782 samples.

Class 1 – Foggy/Misty Conditions:
Covers weather types such as Foggy, Drizzle, or Humid and Mostly Cloudy, which generally indicate low visibility or light atmospheric moisture. This has a total of 17349 samples.

Class 2 – Overcast/Rainy/Windy Conditions:
Represents more adverse weather including Rain, Overcast, Light Rain, and Windy and Overcast, signifying heavier cloud cover, wind, and precipitation. This has a total of 7276 samples.

The number of these classes was determined by several factors, including domain knowledge of weather categories which helped to identify natural groupings in weather patterns. Similar weather was clustered together based on visibility, precipitation, and sky conditions. Another way to determine how many classes to use was the Elbow Method plot using KMeans clustering. It further supported that 3 is an appropriate number of clusters for this variable.



Handling Class Imbalance

The original distribution of the target variable Weather Category was imbalanced, with Class 1 being underrepresented. Synthetic Minority Oversampling Technique (SMOTE) was applied to over sample class 1 to 17000 samples. SMOTE generates synthetic examples of the minority class rather than simply duplicating existing ones. By applying SMOTE, we ensure better class representation during training, leading to more balanced learning and improved generalization.

Standardization

To prepare the dataset for effective model training, standardization was applied to selected numerical features: Temperature (C), Humidity, Wind Speed (km/h), Visibility (km), and Pressure (millibars). This was achieved using StandardScaler, which transforms each feature to have a mean of 0 and a standard deviation of 1. Standardization is a critical preprocessing step, particularly when using algorithms like Logistic Regression and Multilayer Perceptron (MLP), both of which are sensitive to the scale of input features due to their reliance on gradient-based optimization. Although Random Forest is generally scale-invariant, applying uniform scaling ensures consistent treatment of features and supports a streamlined workflow when comparing models. The scaler was fit on the training data and applied to both training and test sets to prevent data leakage and maintain model generalization.

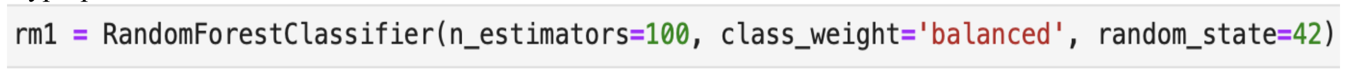
Model Selection

Non-Parametric Algorithm: Random Forest Classifier

For the non-parametric model, the Random Forest Classifier was chosen because of its strong performance in handling high-dimensional feature spaces and its flexibility with minimal preprocessing. Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mode of their predictions. This majority voting mechanism enhances stability and accuracy, particularly in noisy datasets. Additionally, Random Forests are inherently well-suited for multi-class classification tasks and are less prone to overfitting compared to single decision trees. Different models were created by changing hyperparameters and achieved a well generalized model through fine tuning.

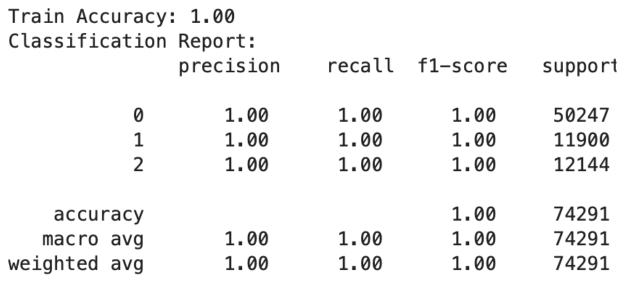
Model RM1 (Baseline Configuration):

Hyperparameters:

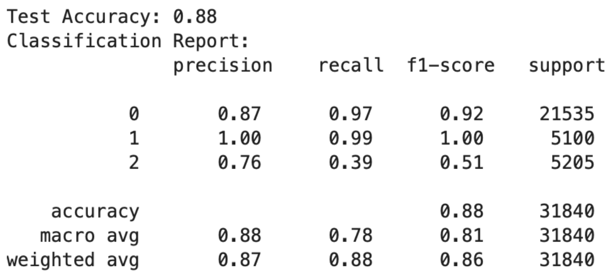


```
rm1 = RandomForestClassifier(n_estimators=100, class_weight='balanced', random_state=42)
```

This model achieved perfect accuracy on the training set, indicating possible overfitting, although test accuracy remained reasonably high. The baseline model overfits.



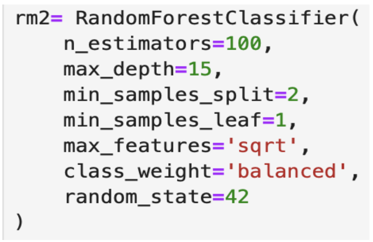
```
Train Accuracy: 1.00
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     50247
           1       1.00      1.00      1.00     11900
           2       1.00      1.00      1.00     12144

    accuracy                           1.00     74291
   macro avg       1.00      1.00      1.00     74291
weighted avg       1.00      1.00      1.00     74291
```



```
Test Accuracy: 0.88
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.97      0.92     21535
           1       1.00      0.99      1.00      5100
           2       0.76      0.39      0.51      5205

    accuracy                           0.88     31840
   macro avg       0.88      0.78      0.81     31840
weighted avg       0.87      0.88      0.86     31840
```

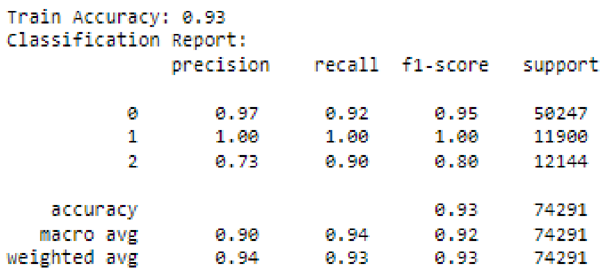
Model RM2 (Depth-Constrained):

Hyperparameters:

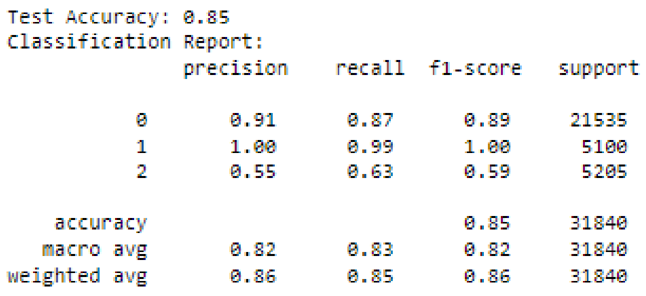


```
rm2= RandomForestClassifier(
    n_estimators=100,
    max_depth=15,
    min_samples_split=2,
    min_samples_leaf=1,
    max_features='sqrt',
    class_weight='balanced',
    random_state=42
)
```

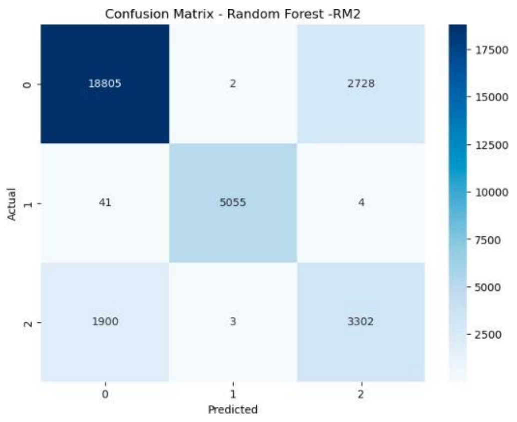
This configuration aimed to reduce overfitting by controlling tree depth and increasing bias. While it generalized slightly less effectively than RM1, it offered more regularization. It uses an ensemble of one hundred decision trees, each limited to a maximum depth to prevent overfitting and maintain manageable complexity. The model ensures that splits in the trees require a minimum number of samples, allowing it to capture important patterns without becoming too sensitive to noise. By selecting a subset of features at each split, it promotes diversity among the trees



```
Train Accuracy: 0.93
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.92      0.95     50247
           1       1.00      1.00      1.00     11900
           2       0.73      0.90      0.80     12144

    accuracy                           0.93     74291
   macro avg       0.90      0.94      0.92     74291
weighted avg       0.94      0.93      0.93     74291
```



```
Test Accuracy: 0.85
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.87      0.89     21535
           1       1.00      0.99      1.00      5100
           2       0.55      0.63      0.59      5205

    accuracy                           0.85     31840
   macro avg       0.82      0.83      0.82     31840
weighted avg       0.86      0.85      0.86     31840
```



Model RM3 (Fine-Tuned with More Estimators):
A more fine-tuned variant, RM3 balanced training performance and generalization well, offering a compromise between RM1 and RM2.
Hyperparameters:

```
rm3 = RandomForestClassifier(
    n_estimators=150,
    max_depth=18,
    min_samples_split=3,
    min_samples_leaf=2,
    max_features='log2',
    random_state=42,
    criterion='gini',
    class_weight='balanced'
)
```

```
Train Accuracy: 0.98
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98     50247
           1       1.00      1.00      1.00     11900
           2       0.89      0.97      0.93     12144

    accuracy                           0.98     74291
   macro avg       0.96      0.98      0.97     74291
weighted avg       0.98      0.98      0.98     74291
```

```
Test Accuracy: 0.87
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.91      0.91     21535
           1       1.00      0.99      0.99      5100
           2       0.62      0.56      0.59      5205

    accuracy                           0.87     31840
   macro avg       0.84      0.82      0.83     31840
weighted avg       0.87      0.87      0.87     31840
```

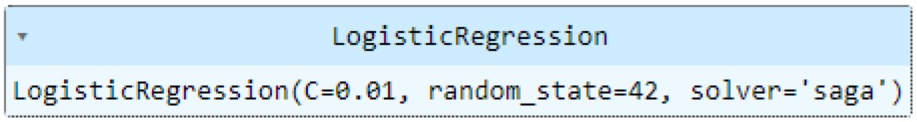
Overall, RM1 achieved the highest test accuracy; however, its perfect training accuracy indicates potential overfitting, suggesting that the model may have memorized the training data rather than generalized well. RM3 provided a better balance between training and test performance, indicating improved generalization while maintaining strong predictive power. RM2, with more constrained hyperparameters, demonstrated effective regularization by reducing overfitting, albeit with a modest decrease in accuracy, highlighting the trade-off between bias and variance in model tuning.

Parametric Algorithm: Logistic Regression

Logistic Regression was selected as the parametric model due to its interpretability and efficiency in binary classification tasks. It models the probability of a binary outcome based on one or more predictor variables, making it suitable for datasets where the relationship between features and the target variable is approximately linear. Beyond its simplicity, Logistic Regression provides well-calibrated probability estimates, which can be thresholded flexibly depending on the cost of false positives versus false negatives in your application. Its coefficients directly quantify the change in log-odds per unit change in each feature, enabling straightforward feature importance analysis and hypothesis testing. Moreover, by incorporating regularization (L1 or L2 penalties), Logistic Regression can mitigate overfitting, perform implicit feature selection, and control model complexity, essential when working with high-dimensional data. Although inherently a binary classifier, it extends naturally to multiclass problems via one-vs-rest or multinomial (softmax) formulations, offering a coherent, easy-to-interpret framework across both binary and multiclass tasks.

Model LR1 (Baseline Configuration):

HyperParameters:

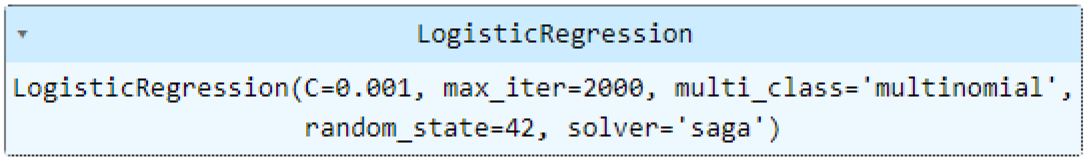


```
LogisticRegression
LogisticRegression(C=0.01, random_state=42, solver='saga')
```

Model LR1 supports strong L_2 regularization (C=0.01) and the 'saga' solver to aggressively shrink coefficients, effectively controlling variance and producing identical train and test accuracies (~0.84) without extreme overfitting.

Model LR2 (Regularized Configuration):

HyperParameters:

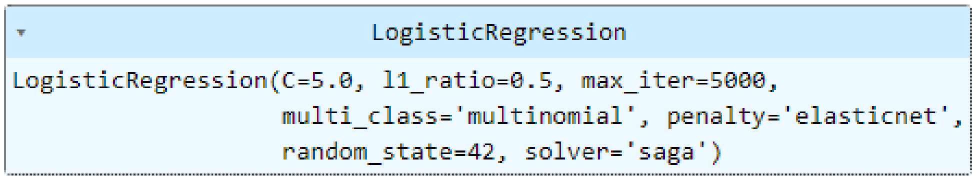


```
LogisticRegression
LogisticRegression(C=0.001, max_iter=2000, multi_class='multinomial',
                   random_state=42, solver='saga')
```

On the **training set**, LR2 attained **83% accuracy** with a **macro ROC-AUC** of **0.896**, and on the **test set**, it replicated both metrics (0.83 accuracy; 0.896 AUC), indicating that the combined effect of stronger regularization and a true multinomial formulation produced a well-generalized model with minimal overfitting.

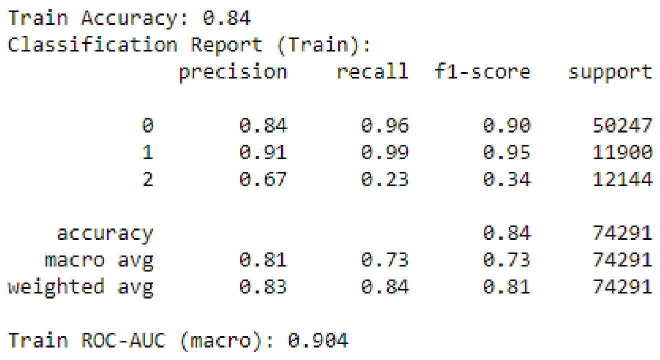
Model LR3 (Optimized Configuration):

HyperParameters:

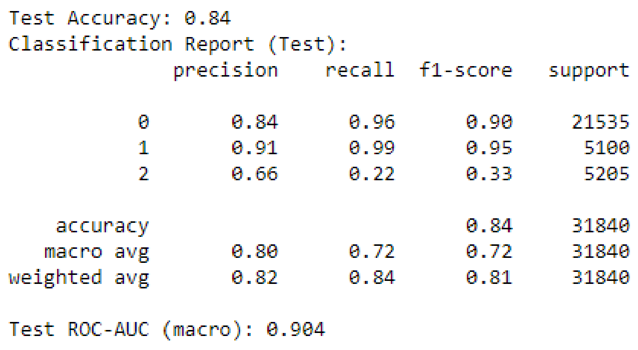


```
LogisticRegression
LogisticRegression(C=5.0, l1_ratio=0.5, max_iter=5000,
                   multi_class='multinomial', penalty='elasticnet',
                   random_state=42, solver='saga')
```

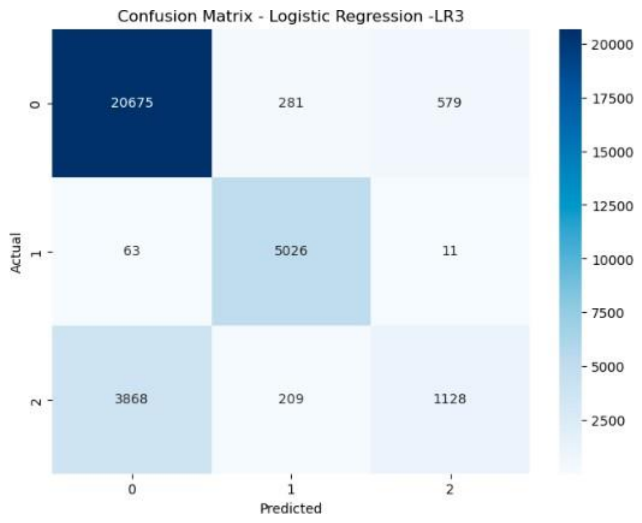
Classification Metric:



```
Train Accuracy: 0.84
Classification Report (Train):
              precision    recall  f1-score   support

           0       0.84      0.96      0.90     50247
           1       0.91      0.99      0.95     11900
           2       0.67      0.23      0.34     12144

    accuracy                           0.84     74291
   macro avg       0.81      0.73      0.73     74291
weighted avg       0.83      0.84      0.81     74291

Train ROC-AUC (macro): 0.904
```



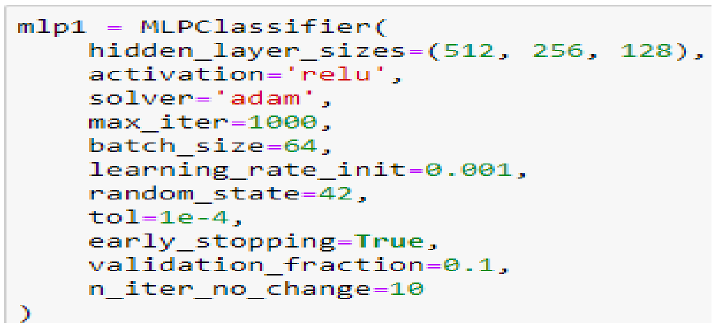
```
Test Accuracy: 0.84
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.84      0.96      0.90     21535
           1       0.91      0.99      0.95      5100
           2       0.66      0.22      0.33      5205

    accuracy                           0.84     31840
   macro avg       0.80      0.72      0.72     31840
weighted avg       0.82      0.84      0.81     31840

Test ROC-AUC (macro): 0.904
```



Neural Network Algorithm: Multi-Layer Perceptron

Model MLP1 (Baseline Configuration):

Hyperparameter:



```
mlp1 = MLPClassifier(
    hidden_layer_sizes=(512, 256, 128),
    activation='relu',
    solver='adam',
    max_iter=1000,
    batch_size=64,
    learning_rate_init=0.001,
    random_state=42,
    tol=1e-4,
    early_stopping=True,
    validation_fraction=0.1,
    n_iter_no_change=10
)
```

MLP1 is a three-layer feedforward neural network implemented via Scikit-learn's **MLPClassifier**, featuring hidden layers of sizes 512, 256, and 128 with ReLU activations, optimized using the Adam solver with early stopping. Trained over up to 1,000 epochs on mini-batches of 64 samples and validated on 10% of the training set, it achieved **88% training accuracy** (macro-F1 ≈ 0.83) and **86% test accuracy** (macro-F1 ≈ 0.80), reflecting strong generalization.

Model LR2 (Regularized Configuration):
Hyperparameter:

```
hidden_layer_sizes=(512, 256, 128, 64),
activation='relu',
solver='adam',
alpha=1e-5,
learning_rate_init=5e-4,
batch_size=128,
max_iter=1500,
early_stopping=True,
random_state=42  )
```

MLP2’s deeper architecture (512→256→128→64) combined with light L_2 regularization and large batches allowed it to learn richer feature hierarchies while maintaining stable convergence due to a reduced learning rate (5×10^{-4}) and early stopping over 1,500 epochs without overfitting.

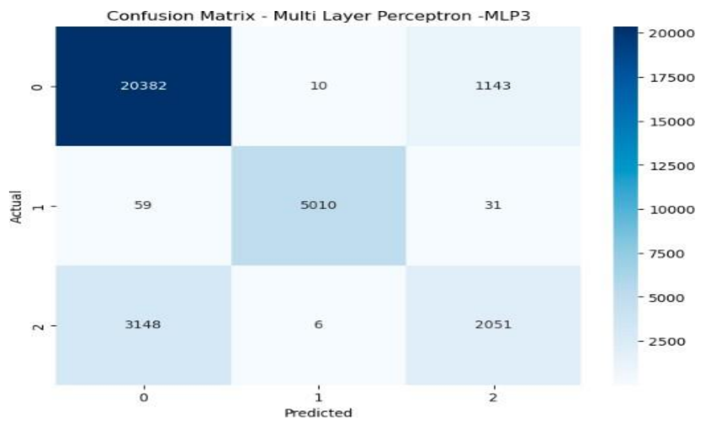
MLP3 (Configuration & Rationale):
Hyperparameter:

```
mlp3 = MLPClassifier(
    hidden_layer_sizes=(1024,512),
    activation='relu',
    solver='adam',
    learning_rate='adaptive',
    learning_rate_init=1e-2,
    momentum=0.9,
    alpha=1e-3,
    batch_size=64,
    max_iter=2000,
    early_stopping=True,
    n_iter_no_change=10,
    validation_fraction=0.1,
    random_state=42
)
```

MLP3 is a two-layer feedforward network with 1,024 and 512 ReLU neurons, trained using the Adam optimizer with an **adaptive** learning rate that begins at 0.01 and decays when validation performance stalls. A **momentum** of 0.9 smooths gradient updates, while a moderate **L_2 penalty** (α=1e-3) helps prevent overfitting in its wide layers. Training proceeds in mini-batches of 64 samples, with **early stopping** triggered if no improvement occurs over 10 validation-set epochs (10% split), and runs for up to 2,000 iterations. MLP3’s accuracy 87% on the training set and 86% on the test set indicates a well-regularized model that generalizes almost as well as it fits.

```
Train Accuracy: 0.87
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.95      0.91     50247
           1       1.00      0.99      0.99     11900
           2       0.66      0.41      0.51     12144

    accuracy                           0.87     74291
   macro avg       0.84      0.78      0.80     74291
weighted avg       0.85      0.87      0.85     74291
```

```
Test Accuracy: 0.86
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.96      0.91     21535
           1       0.99      0.97      0.98      5100
           2       0.67      0.38      0.48      5205

    accuracy                           0.86     31840
   macro avg       0.84      0.77      0.79     31840
weighted avg       0.85      0.86      0.85     31840
```



Tuning and Model Evaluation Interface:

To enable weather-category predictions, we implemented a lightweight command-line function, **simple_predict(),** which walks the user through seven input prompts, applies the same preprocessing pipeline used during training, and then outputs each model's prediction.

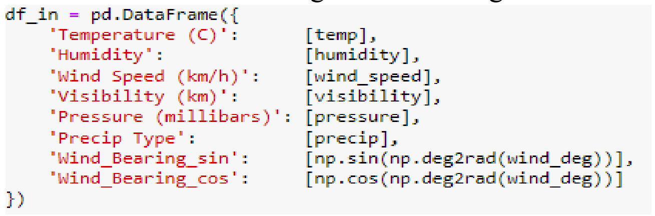
User Input Collection

The function first requests the following real-time measurements:

- **Temperature (°C)**
- **Relative Humidity (0–1)**
- **Wind Speed (km/h)**
- **Visibility (km)**
- **Pressure (millibars)**
- **Precipitation Type**
- **Wind Bearing (°)**

Feature Engineering & DataFrame Construction

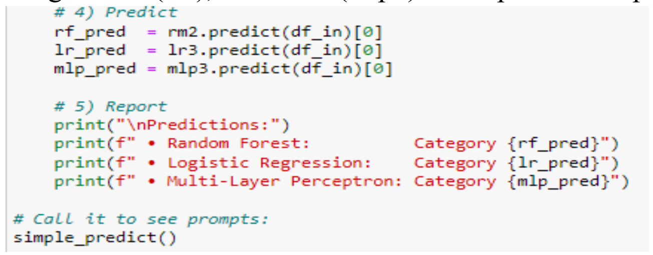
The raw inputs are assembled into a single-row Pandas DataFrame, ensuring each column exactly matches the feature names used during model training:



```
df_in = pd.DataFrame({
    'Temperature (C)':      [temp],
    'Humidity':             [humidity],
    'Wind Speed (km/h)':    [wind_speed],
    'Visibility (km)':      [visibility],
    'Pressure (millibars)': [pressure],
    'Precip Type':          [precip],
    'Wind_Bearing_sin':     [np.sin(np.deg2rad(wind_deg))],
    'Wind_Bearing_cos':     [np.cos(np.deg2rad(wind_deg))]
})
```

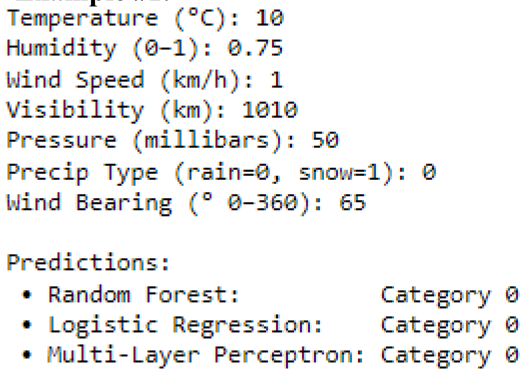
Model Inference & Reporting

Finally, simple_predict() calls .predict() on each serialized model instance—Random Forest (rm1), Logistic Regression (lr3), and MLP (mlp1)—and prints their predicted class indices:



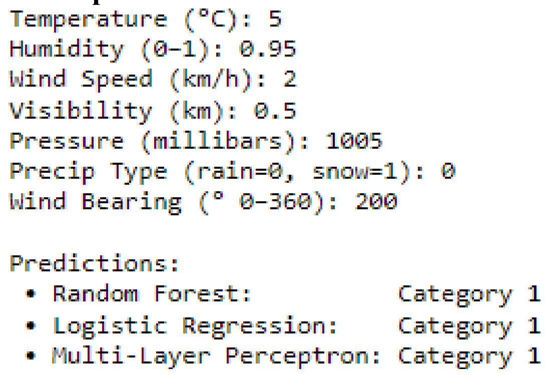
```
    # 4) Predict
    rf_pred  = rm2.predict(df_in)[0]
    lr_pred  = lr3.predict(df_in)[0]
    mlp_pred = mlp3.predict(df_in)[0]

    # 5) Report
    print("\nPredictions:")
    print(f" • Random Forest:          Category {rf_pred}")
    print(f" • Logistic Regression:    Category {lr_pred}")
    print(f" • Multi-Layer Perceptron: Category {mlp_pred}")

# Call it to see prompts:
simple_predict()
```

Example #1:



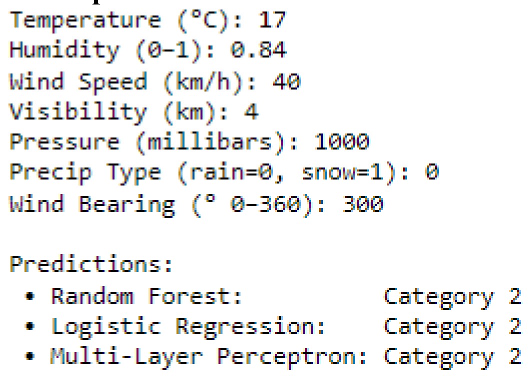
```
Temperature (°C): 10
Humidity (0-1): 0.75
Wind Speed (km/h): 1
Visibility (km): 1010
Pressure (millibars): 50
Precip Type (rain=0, snow=1): 0
Wind Bearing (° 0-360): 65

Predictions:
 • Random Forest:          Category 0
 • Logistic Regression:    Category 0
 • Multi-Layer Perceptron: Category 0
```

Example #2:



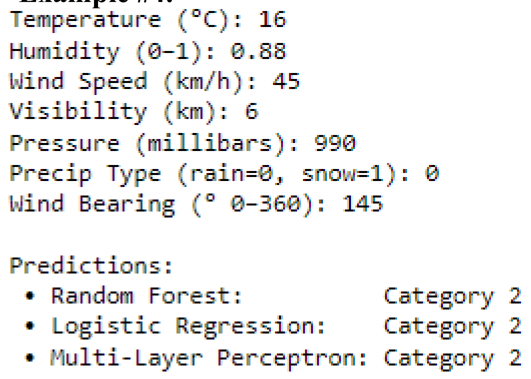
```
Temperature (°C): 5
Humidity (0-1): 0.95
Wind Speed (km/h): 2
Visibility (km): 0.5
Pressure (millibars): 1005
Precip Type (rain=0, snow=1): 0
Wind Bearing (° 0-360): 200

Predictions:
 • Random Forest:          Category 1
 • Logistic Regression:    Category 1
 • Multi-Layer Perceptron: Category 1
```

Example #3:



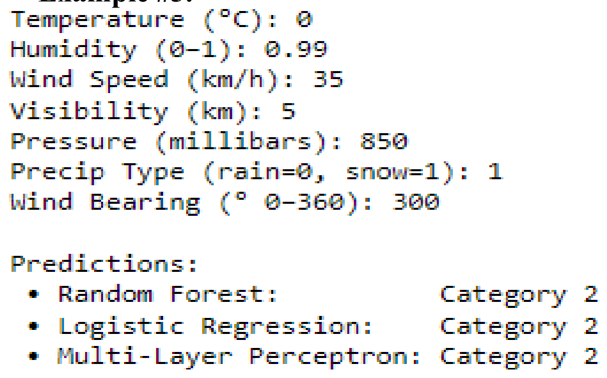
```
Temperature (°C): 17
Humidity (0-1): 0.84
Wind Speed (km/h): 40
Visibility (km): 4
Pressure (millibars): 1000
Precip Type (rain=0, snow=1): 0
Wind Bearing (° 0-360): 300

Predictions:
 • Random Forest:          Category 2
 • Logistic Regression:    Category 2
 • Multi-Layer Perceptron: Category 2
```

Example #4:



```
Temperature (°C): 16
Humidity (0-1): 0.88
Wind Speed (km/h): 45
Visibility (km): 6
Pressure (millibars): 990
Precip Type (rain=0, snow=1): 0
Wind Bearing (° 0-360): 145

Predictions:
 • Random Forest:          Category 2
 • Logistic Regression:    Category 2
 • Multi-Layer Perceptron: Category 2
```

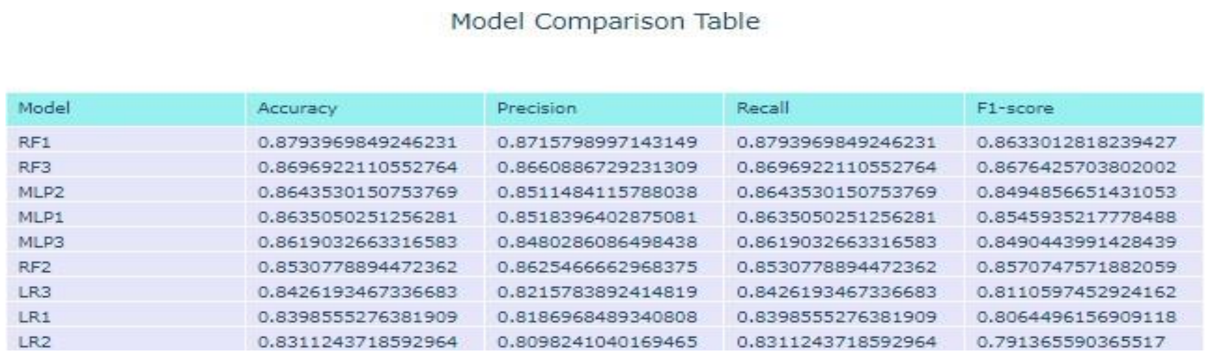
Example #5:



```
Temperature (°C): 0
Humidity (0-1): 0.99
Wind Speed (km/h): 35
Visibility (km): 5
Pressure (millibars): 850
Precip Type (rain=0, snow=1): 1
Wind Bearing (° 0-360): 300

Predictions:
 • Random Forest:          Category 2
 • Logistic Regression:    Category 2
 • Multi-Layer Perceptron: Category 2
```

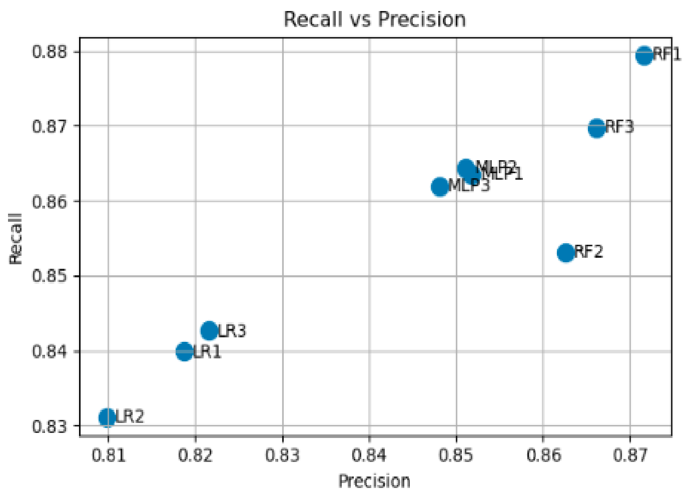
Results and Analysis

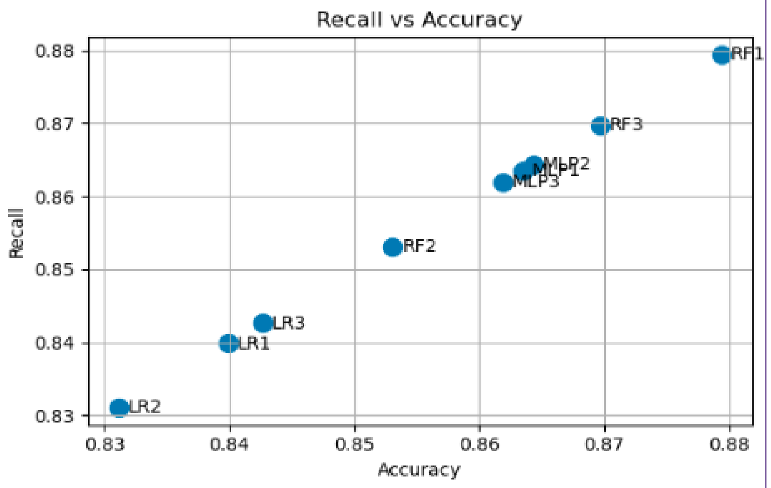
- ### Tabular Comparison of All Models

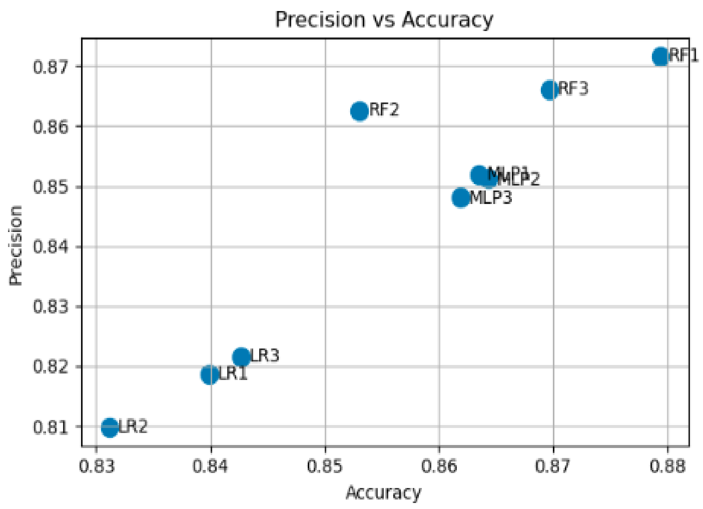

Model Comparison Table

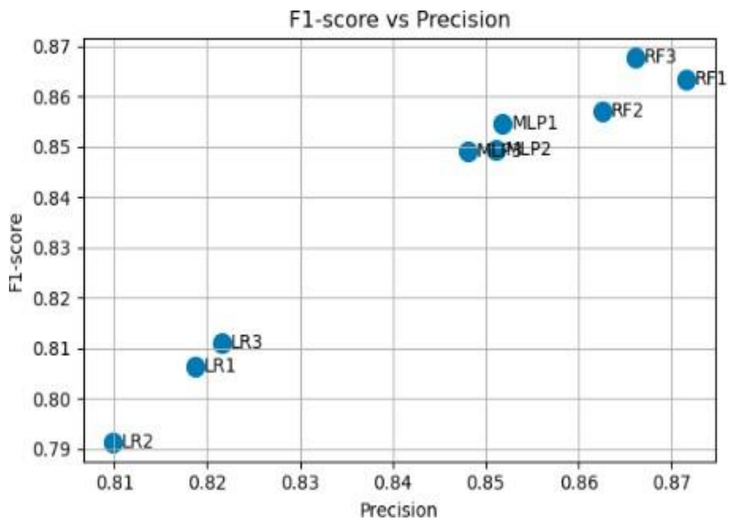
Model	Accuracy	Precision	Recall	F1-score
RF1	0.8793969849246231	0.8715798997143149	0.8793969849246231	0.8633012818239427
RF3	0.8696922110552764	0.8660886729231309	0.8696922110552764	0.8676425703802002
MLP2	0.8643530150753769	0.8511484115788038	0.8643530150753769	0.8494856651431053
MLP1	0.8635050251256281	0.8518396402875081	0.8635050251256281	0.8545935217778488
MLP3	0.8619032663316583	0.8480286086498438	0.8619032663316583	0.8490443991428439
RF2	0.8530778894472362	0.8625466662968375	0.8530778894472362	0.8570747571882059
LR3	0.8426193467336683	0.8215783892414819	0.8426193467336683	0.8110597452924162
LR1	0.8398555276381909	0.8186968489340808	0.8398555276381909	0.8064496156909118
LR2	0.8311243718592964	0.8098241040169465	0.8311243718592964	0.791365590365517

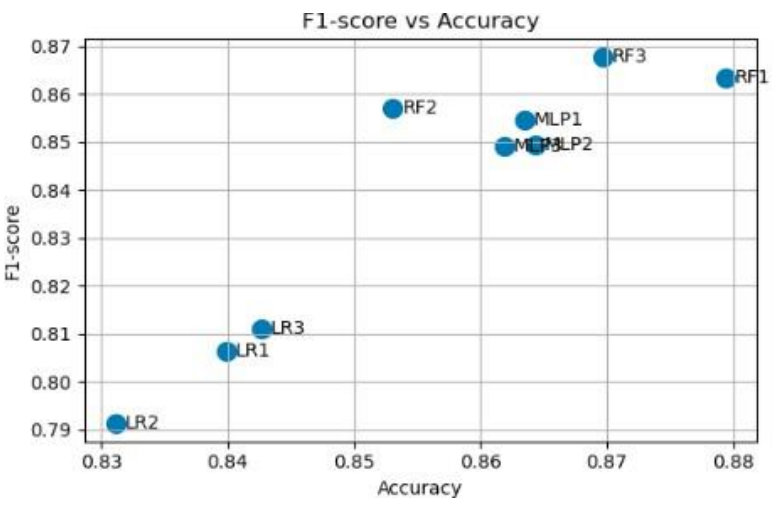
- ### Graphical Comparison

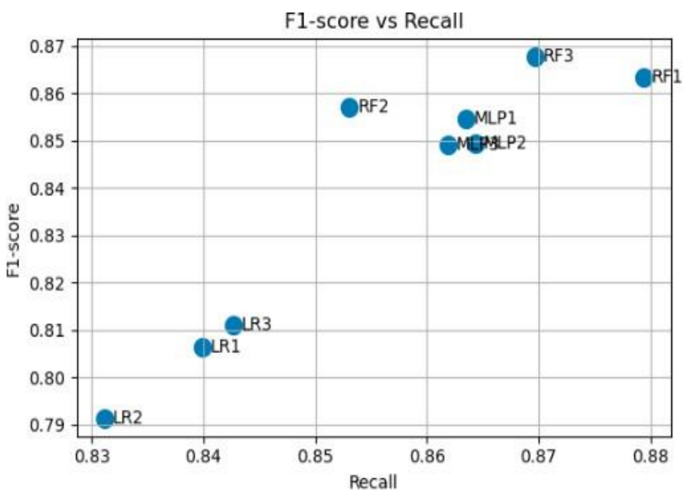






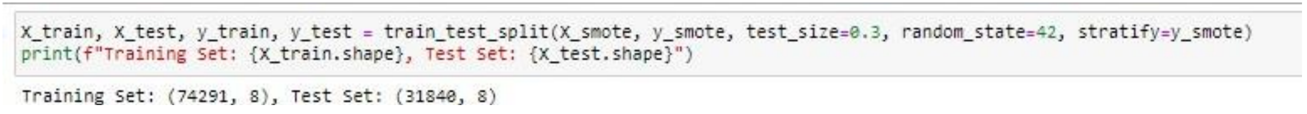






- ## Data Split Ratio Analysis

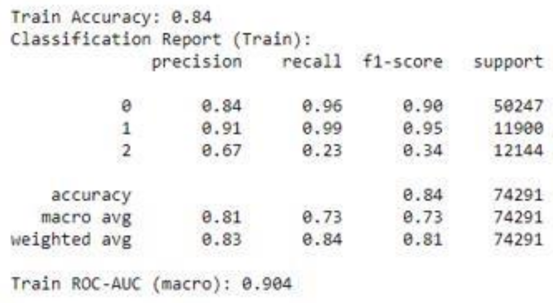
Train = 70% , Test= 30% : this is the default train split we used**.**



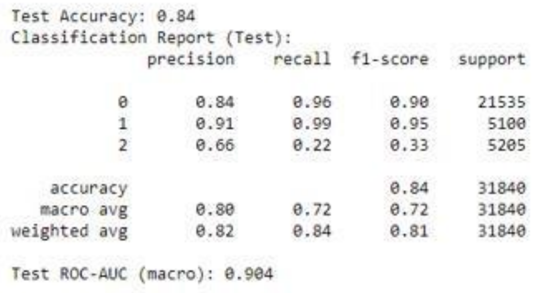
```
X_train, X_test, y_train, y_test = train_test_split(X_smote, y_smote, test_size=0.3, random_state=42, stratify=y_smote)
print(f"Training Set: {X_train.shape}, Test Set: {X_test.shape}")
```

```
Training Set: (74291, 8), Test Set: (31840, 8)
```

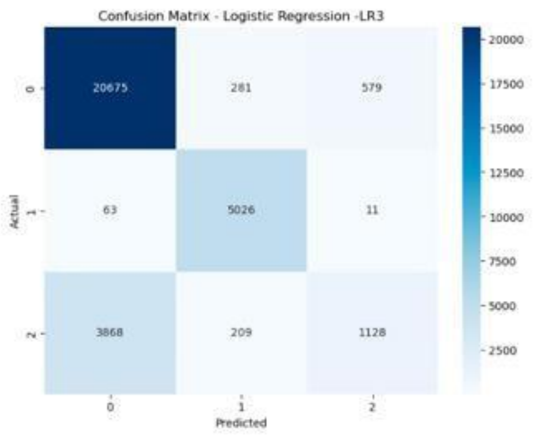
Logistic Regression:



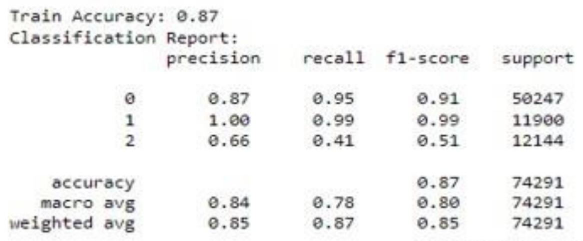
```
Train Accuracy: 0.84
Classification Report (Train):
              precision    recall  f1-score   support

           0       0.84      0.96      0.90     50247
           1       0.91      0.99      0.95     11900
           2       0.67      0.23      0.34     12144

    accuracy                           0.84     74291
   macro avg       0.81      0.73      0.73     74291
weighted avg       0.83      0.84      0.81     74291

Train ROC-AUC (macro): 0.904
```



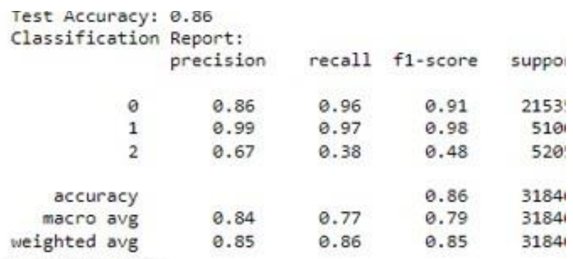
```
Test Accuracy: 0.84
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.84      0.96      0.90     21535
           1       0.91      0.99      0.95      5100
           2       0.66      0.22      0.33      5205

    accuracy                           0.84     31840
   macro avg       0.80      0.72      0.72     31840
weighted avg       0.82      0.84      0.81     31840

Test ROC-AUC (macro): 0.904
```



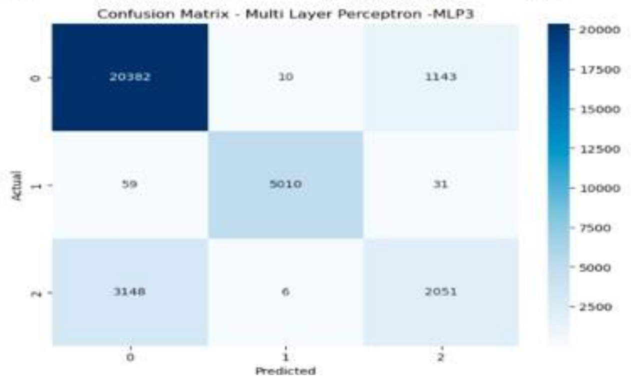
Multi-Layer Perceptron:



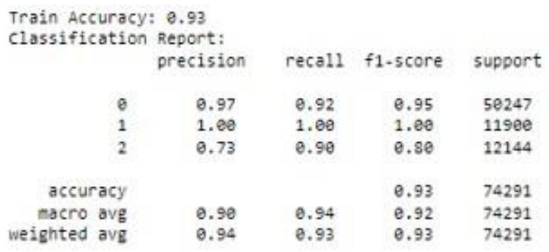
```
Train Accuracy: 0.87
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.95      0.91     50247
           1       1.00      0.99      0.99     11900
           2       0.66      0.41      0.51     12144

    accuracy                           0.87     74291
   macro avg       0.84      0.78      0.80     74291
weighted avg       0.85      0.87      0.85     74291
```



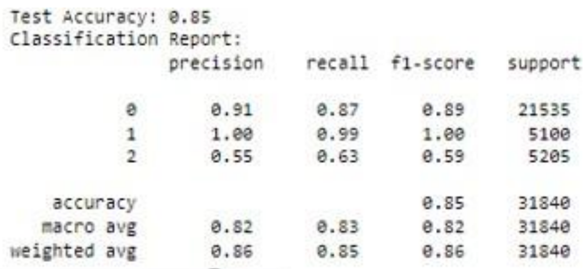
```
Test Accuracy: 0.86
Classification Report:
              precision    recall  f1-score   suppo

           0       0.86      0.96      0.91     2153
           1       0.99      0.97      0.98      510
           2       0.67      0.38      0.48      520

    accuracy                           0.86     3184
   macro avg       0.84      0.77      0.79     3184
weighted avg       0.85      0.86      0.85     3184
```



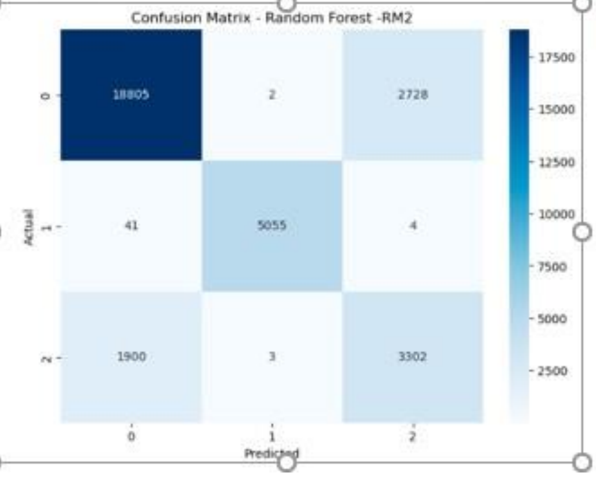
Random Forest Classifier:



```
Train Accuracy: 0.93
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.92      0.95     50247
           1       1.00      1.00      1.00     11900
           2       0.73      0.90      0.80     12144

    accuracy                           0.93     74291
   macro avg       0.90      0.94      0.92     74291
weighted avg       0.94      0.93      0.93     74291
```



```
Test Accuracy: 0.85
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.87      0.89     21535
           1       1.00      0.99      1.00      5100
           2       0.55      0.63      0.59      5205

    accuracy                           0.85     31840
   macro avg       0.82      0.83      0.82     31840
weighted avg       0.86      0.85      0.86     31840
```

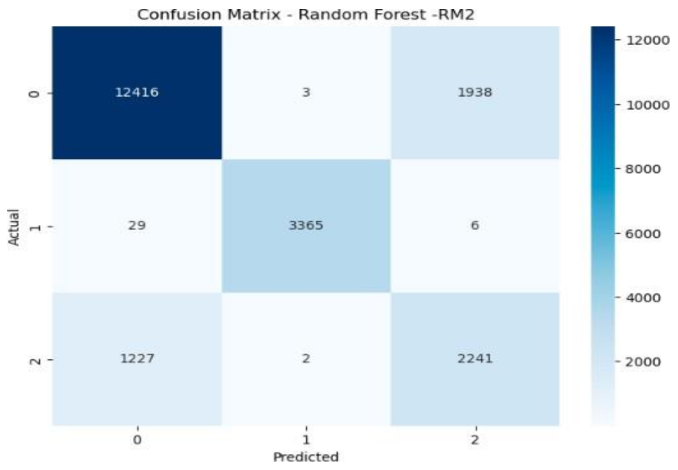


Train = 80% , Test= 20% : The train accuracy for random forest model seems to improve but not test accuracy, where logistic regression model remains the same**.** Also, in multilayer perceptron no improvement has been encountered.

Random Forest:

```
Train Accuracy: 0.92
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.91      0.94     57425
           1       1.00      1.00      1.00     13600
           2       0.70      0.89      0.78     13879

    accuracy                           0.92     84904
   macro avg       0.89      0.93      0.91     84904
weighted avg       0.93      0.92      0.92     84904
```

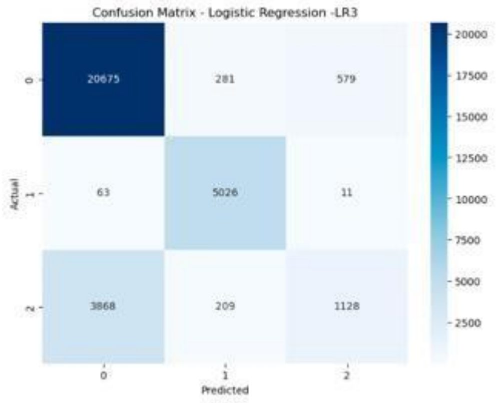
```
Test Accuracy: 0.85
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.86      0.89     14357
           1       1.00      0.99      0.99      3400
           2       0.54      0.65      0.59      3470

    accuracy                           0.85     21227
   macro avg       0.81      0.83      0.82     21227
weighted avg       0.86      0.85      0.85     21227
```



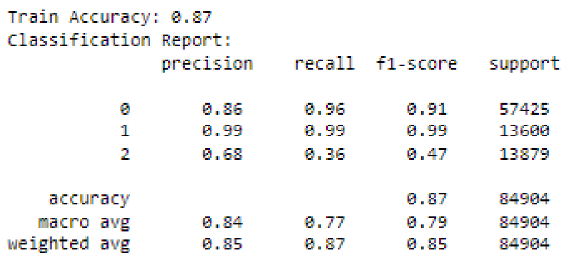
Logistic Regression:

```
Train Accuracy: 0.84
Classification Report (Train):
              precision    recall  f1-score   support

           0       0.84      0.96      0.90     50247
           1       0.91      0.99      0.95     11900
           2       0.67      0.23      0.34     12144

    accuracy                           0.84     74291
   macro avg       0.81      0.73      0.73     74291
weighted avg       0.83      0.84      0.81     74291

Train ROC-AUC (macro): 0.904
```

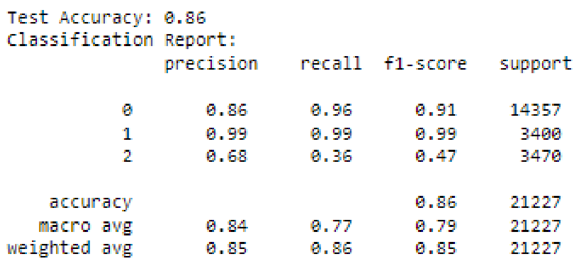
```
Test Accuracy: 0.84
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.84      0.96      0.90     21535
           1       0.91      0.99      0.95      5100
           2       0.66      0.22      0.33      5205

    accuracy                           0.84     31840
   macro avg       0.80      0.72      0.72     31840
weighted avg       0.82      0.84      0.81     31840

Test ROC-AUC (macro): 0.904
```



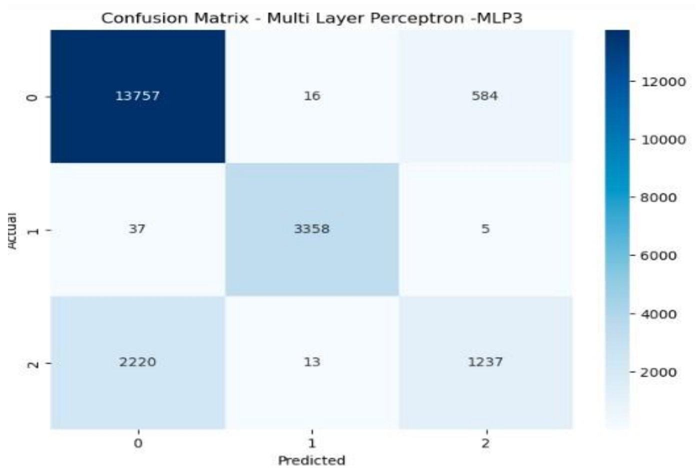
Multi-Layer Perceptron:



```
Train Accuracy: 0.87
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.96      0.91     57425
           1       0.99      0.99      0.99     13600
           2       0.68      0.36      0.47     13879

    accuracy                           0.87     84904
   macro avg       0.84      0.77      0.79     84904
weighted avg       0.85      0.87      0.85     84904
```



```
Test Accuracy: 0.86
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.96      0.91     14357
           1       0.99      0.99      0.99      3400
           2       0.68      0.36      0.47      3470

    accuracy                           0.86     21227
   macro avg       0.84      0.77      0.79     21227
weighted avg       0.85      0.86      0.85     21227
```



Performance of Machine Learning Models

The implemented machine learning system demonstrated good performance across all three model types: Random Forest (non-parametric), Logistic Regression (parametric), and Multi-Layer Perceptron (neural network). Each model exhibited different characteristics in terms of underfitting and overfitting.

For instance, the baseline Random Forest model (RM1) achieved perfect accuracy on the training set, signaling overfitting despite reasonably high-test accuracy. This was addressed in RM2 and RM3 through techniques like limiting tree depth, adjusting sample splits, and fine-tuning estimators, resulting in better generalization. Logistic Regression models were inherently resistant to overfitting due to strong L2 regularization and elastic-net penalties, consistently producing nearly identical train and test accuracies. The neural network models, particularly MLP3, employed strategies like L2 regularization, early stopping, and adaptive learning rates, which effectively mitigated overfitting while preserving high accuracy.

However, Class 2 was often misrepresented as class 0, due to similar features and less data. That can be improved by having more unique data and weather samples.

To further improve performance, techniques like cross-validation, or ensemble blending could be considered. Using more hyperparameters through grid search can be considered, specially for MLPs and Random Forests, where hyperparameter combinations can be large. Moreover, expanding the dataset or augmenting features might help reduce bias and improve model accuracy, especially under challenging or unseen weather conditions.

Conclusion

We transformed a raw weather dataset of over 96,000 records originally containing 27 summary labels into a focused three-class prediction task through cleansing (duplicate removal, anomaly correction), cyclical encoding of wind direction, and label consolidation. We addressed class imbalance using SMOTE, oversampling the minority category to ensure all models could effectively learn its decision boundary. MLP3 supports two-layer deep network (1024, 512 neurons) with an adaptive learning rate and momentum, along with moderate L_2 regularization. This configuration struck a perfect balance between capacity and control, achieving 86% test accuracy and a weighted F1 of ~0.85. RF Model RM2—configured with 200 trees, a maximum depth of 10, and a minimum of 5 samples per split proved remarkably strong. It delivered 85% test accuracy and 0.83 weighted F1, maintaining consistently high precision and recall across all three classes.
By controlling both tree complexity and sample requirements, RM2 hit the sweet spot between variance and bias, offering dependable performance in noisy or edge-case scenarios. Logistic Regression LR3, using an elastic-net penalty (C=5.0, l1_ratio=0.5), remains the go-to for transparency and speed. With 84% test accuracy and a 0.90 macro ROC-AUC, it balances interpretability and generalization. Each of these models excels in a different dimension MLP3 for nonlinear complexity, RM2 for ensemble stability, and LR3 for transparency and efficiency.