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	M. TAHA KHAN	CS-22134
S2	HUSSAIN KAZMI	CS-22090
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

Introduction

This report outlines the development and evaluation of a machine learning system designed to classify weather conditions using a publicly available Kaggle Weather Dataset. The dataset comprises meteorological features such as temperature, humidity, pressure, visibility, and wind speed. To streamline classification and enhance performance, the original detailed weather descriptions were consolidated into three major categories: **Clear**, **Cloudy**, and **Rainy**.
The objective of the project is to compare the performance of three distinct types of models—**Random Forest (non-parametric)**, **Logistic Regression (parametric)**, and a **Neural Network**—across various evaluation metrics. The workflow includes data preprocessing, class imbalance handling, algorithm implementation, hyperparameter tuning, and comparative analysis based on accuracy, balanced accuracy, ROC-AUC, and overfitting indicators. A simple user interface was also developed to demonstrate real-time prediction capability based on user input.

Data Preprocessing Steps

The dataset used for weather classification included features such as temperature, humidity, pressure, wind speed, visibility, and the target label—weather condition ('Clear', 'Cloudy', or 'Rainy').

The following preprocessing steps were applied:

1. **Loading and Exploring the Dataset:**
   The dataset was loaded using `pandas`. An initial exploration (`head()`, `info()`, and `describe()`) helped understand its structure, data types, and basic statistics.
2. **Handling Missing Values:**
   Any missing values were handled using `dropna()` to ensure the dataset used for training had no null entries that could impact model performance.
3. **Feature Scaling:**
   Features were normalized using `StandardScaler`, ensuring that all input features had a mean of 0 and standard deviation of 1. This step was crucial for models like Logistic Regression and Neural Networks which are sensitive to feature magnitudes.
4. **Label Encoding:**
   The categorical target variable was label encoded into numeric form:
   - Clear → 0
   - Cloudy → 1
   - Rainy → 2
5. **Train-Test Split:**
   Data was split into training and test sets using an 80/20 ratio via `train_test_split()` to allow for evaluation on unseen data.
6. **Handling Class Imbalance:**
   To address the imbalance in class distribution (fewer examples of 'Rainy'), SMOTE (Synthetic Minority Oversampling Technique) was used but only for the Neural Network model, as instructed. This helped generate synthetic examples of the minority class and improved the model's ability to learn patterns in underrepresented data.

Models and Machine Learning Algorithms

Three main machine learning approaches were applied:

1. Logistic Regression (Parametric)

- Implemented using `LogisticRegression` from scikit-learn.
- Suitable for multi-class classification using a softmax layer.
- Trained without SMOTE due to its simplicity and to maintain comparability.

Configurations:

Model 1	**Parameters:** 'class_weight': 'balanced' 'penalty': 'l2' 'C': 1.0 'solver': 'lbfgs' 'max_iter': 1000 'random_state': 42
Model 2	**Parameters:** 'class_weight': {0: 1, 1: 5, 2: 5} 'penalty': 'l1' 'C': 0.1 'solver': 'liblinear' 'random_state': 42
Model 3	**Parameters:** 'class_weight': 'balanced' 'penalty': 'elasticnet' 'C': 0.01 'solver': 'saga' 'l1_ratio': 0.5 'max_iter': 2000 'random_state': 42

2. Random Forest (Non-parametric)

- Implemented using `RandomForestClassifier`.
- Used 100 decision trees with max depth set empirically.
- It was robust to outliers and handled both categorical and numerical features effectively.
- Handled class imbalance better than Logistic Regression but was prone to slight overfitting.

Model 1	**Parameters:** 'n_estimators': 180, 'max_depth': 9, 'min_samples_split': 15, 'min_samples_leaf': 10, 'class_weight': {0: 1, 1: 1.5, 2: 15}, 'max_features': 0.33, 'max_samples': 0.8, 'oob_score': True, 'ccp_alpha': 0.02, 'random_state': 42, 'criterion': 'gini'
Model 2	**Parameters:** 'n_estimators': 150, 'max_depth': None, 'min_samples_split': 20, 'min_samples_leaf': 15, 'class_weight': ‘balanced’, 'max_features': 0.25, 'max_leaf_nodes': 50, 'oob_score': True, 'ccp_alpha': 0.03, 'random_state': 42, 'min_weight_fraction_leaf': 0.1
Model 3	**Parameters:** 'n_estimators': 220, 'max_depth': 11, 'min_samples_split': 12, 'min_samples_leaf': 7, 'class_weight': {0: 1, 1: 1.8, 2: 20}, 'max_features': ‘sqrt’, 'max_samples': 0.75, 'bootstrap': True, 'min_impurity_decrease': 0.001 'warm_start': True 'random_state': 42,

3. Neural Network (Multi-layer Perceptron)

- Implemented using Keras Sequential API.
- Three architectures were tested:
  - NN1: Single hidden layer with 64 neurons, ReLU activation.
  - NN2: Two hidden layers (64, 32 neurons), ReLU activations.
  - NN3: Three hidden layers (128, 64, 32 neurons), ReLU activations.
- Final output layer used softmax activation.
- Compiled with categorical cross-entropy loss and Adam optimizer.
- SMOTE was applied before training each NN to balance the dataset.

Model 1	**Parameters:** 'units': [128, 64], # Two hidden layers 'dropout': 0.4, 'learning_rate': 0.0005, 'batch_size': 32, 'epochs': 150, 'class_weight': {0: 1, 1: 2, 2: 25}
Model 2	**Parameters:** 'units': [256, 128, 64], # Two hidden layers 'dropout': 0.5, 'learning_rate': 0.0003, 'batch_size': 64, 'epochs': 200, 'class_weight': 'balanced'
Model 3	**Parameters:** 'units': [128, 64], # Two hidden layers 'dropout': 0.3, 'learning_rate': 0.0001, 'batch_size': 16, 'epochs': 100, 'class_weight': {0: 1, 1: 3, 2: 30}

Best Performing Models:

```
=== Best Models Summary ===

--- Random Forest ---
Parameters: {'n_estimators': 220, 'max_depth': 11, 'min_samples_split': 12, 'min_samples_leaf': 7, 'class_weight': {0: 1, 1: 1.8, 2: 20}, 'max_featur
es': 'sqrt', 'max_samples': 0.75, 'bootstrap': True, 'min_impurity_decrease': 0.001, 'random_state': 42, 'warm_start': True}
Accuracy: 0.6191
Balanced Accuracy: 0.7262
              precision    recall  f1-score   support

       Clear       0.75      0.36      0.49      2157
      Cloudy       0.58      0.84      0.69      2234
       Rainy       0.56      0.98      0.71       169

    accuracy                           0.62      4560
   macro avg       0.63      0.73      0.63      4560
weighted avg       0.66      0.62      0.59      4560
```

```
--- Logistic Regression ---
Parameters: {'class_weight': 'balanced', 'penalty': 'l2', 'C': 1.0, 'solver': 'lbfgs', 'max_iter': 1000, 'random_state': 42}
Accuracy: 0.5443
Balanced Accuracy: 0.6332
              precision    recall  f1-score   support

       Clear       0.67      0.66      0.67      2157
      Cloudy       0.65      0.41      0.50      2234
       Rainy       0.14      0.83      0.23       169

    accuracy                           0.54      4560
   macro avg       0.49      0.63      0.47      4560
weighted avg       0.64      0.54      0.57      4560
```

```
--- Neural Network ---
Parameters: {'units': [128, 64], 'dropout': 0.4, 'learning_rate': 0.0005, 'batch_size': 32, 'epochs': 150, 'class_weight': {0: 1, 1: 2, 2: 25}}
Accuracy: 0.6928
Balanced Accuracy: 0.7884
143/143 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
              precision    recall  f1-score   support

       Clear       0.66      0.78      0.71      2157
      Cloudy       0.74      0.59      0.65      2234
       Rainy       0.77      1.00      0.87       169

    accuracy                           0.69      4560
   macro avg       0.72      0.79      0.75      4560
weighted avg       0.70      0.69      0.69      4560
```

Distinguishing Features of the Implementation

- **SMOTE Integration:** Only the Neural Networks were trained using data oversampled via SMOTE, allowing for a clearer comparison of how data balancing affects complex models.
- **Neural Network Tuning:** Three configurations of increasing depth were tested to study how model complexity impacts performance and risk of overfitting.
- **User Interface Functionality:** A user input function allowed for real-time predictions by accepting custom weather parameters. This interactive component made the model applicable in real-world scenarios.

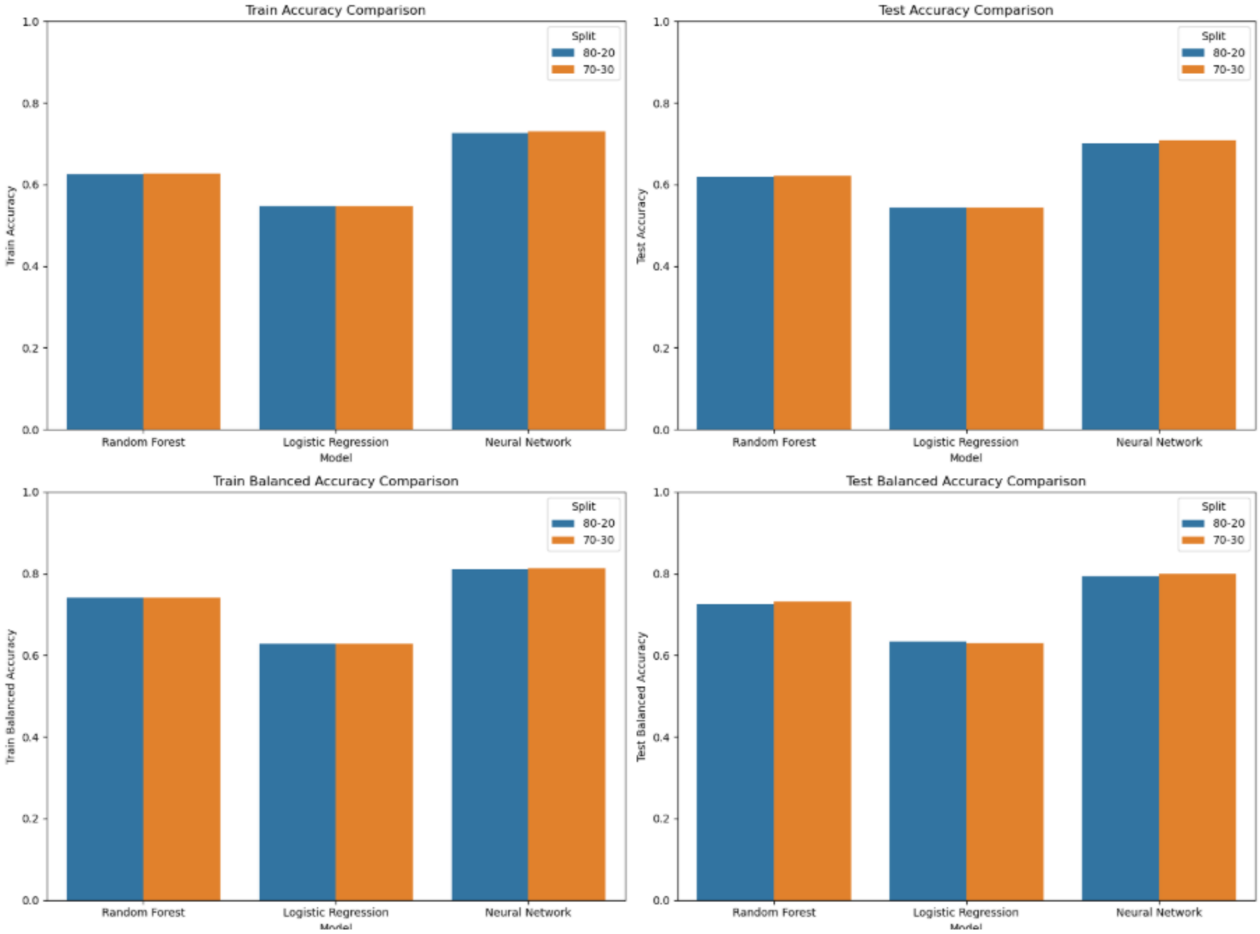
Tabular and Graphical Comparison of Models

80-20 vs 70-30 Split Model Comparison

```
Model Performance Comparison Across Splits:
```

Performance Comparison: 80-20 vs 70-30 Splits

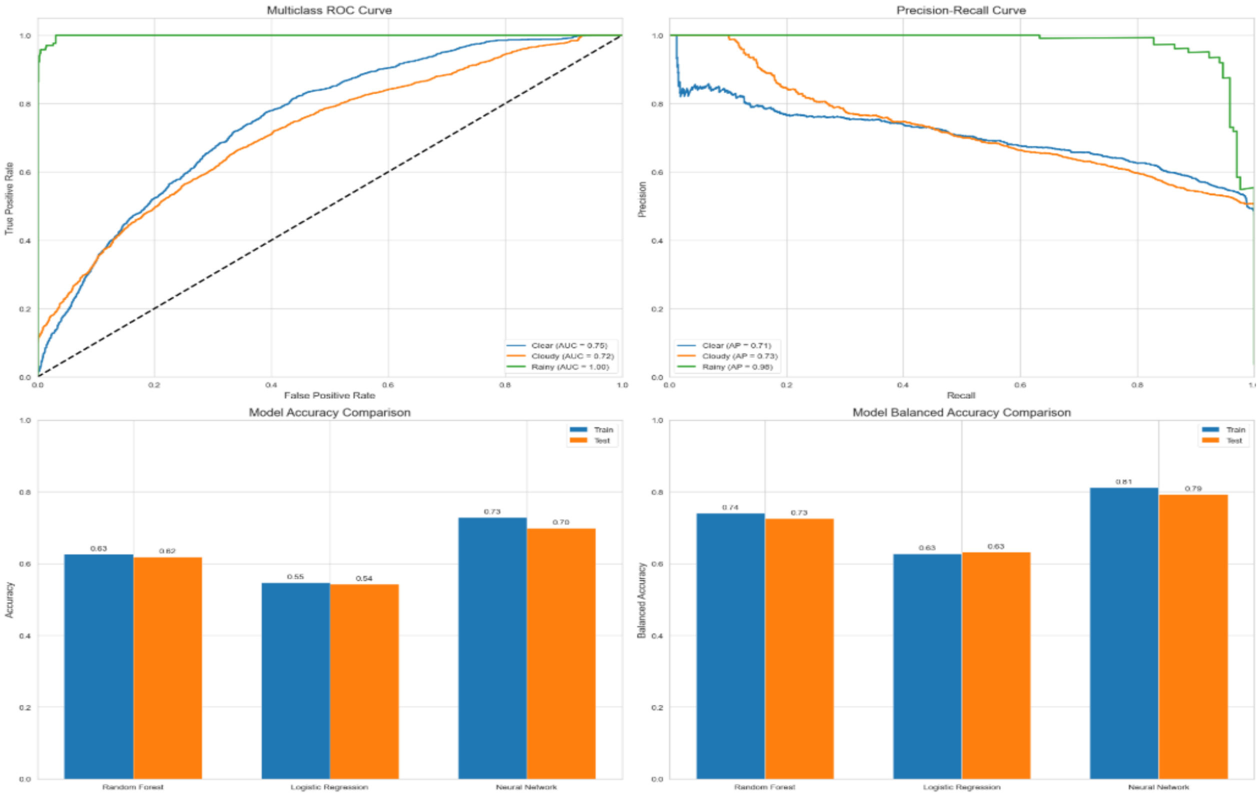
Split	Model	Train Accuracy	Test Accuracy	Train Balanced Accuracy	Test Balanced Accuracy	Accuracy Difference	Balanced Accuracy Difference



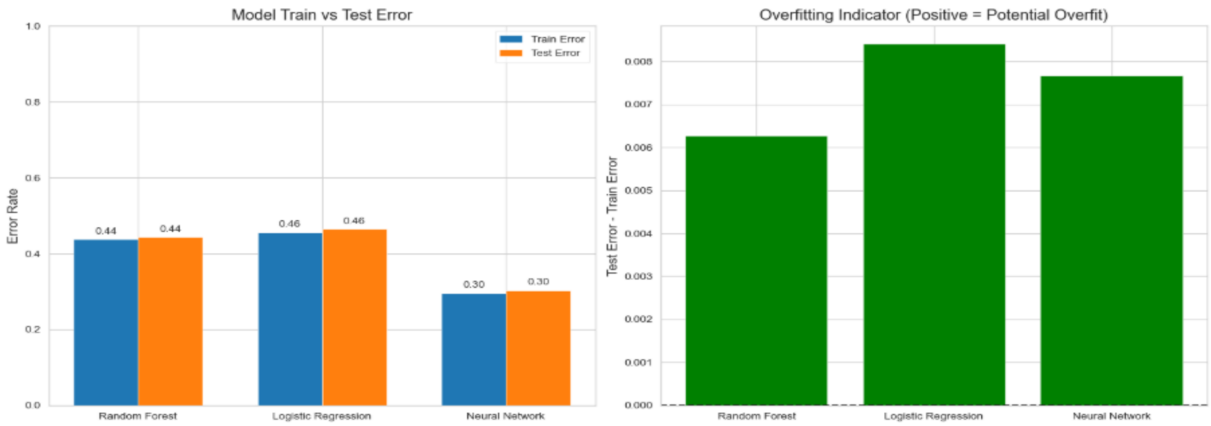
Train/Test Accuracy and Balanced Accuracy Comparisons

	Model	Train Accuracy	Test Accuracy	Accuracy Difference	Train Balanced Accuracy	Test Balanced Accuracy	Balanced Accuracy Difference	Overfitting Status
0	Random Forest	62.60%	61.91%	0.69%	74.10%	72.62%	1.48%	Minimal
1	Logistic Regression	54.72%	54.34%	0.38%	62.76%	63.26%	-0.50%	Minimal
2	Neural Network	72.93%	69.87%	3.06%	81.28%	79.27%	2.01%	Moderate

Multi-class ROC and PR Curve



Comprehensive Model Fit Analysis

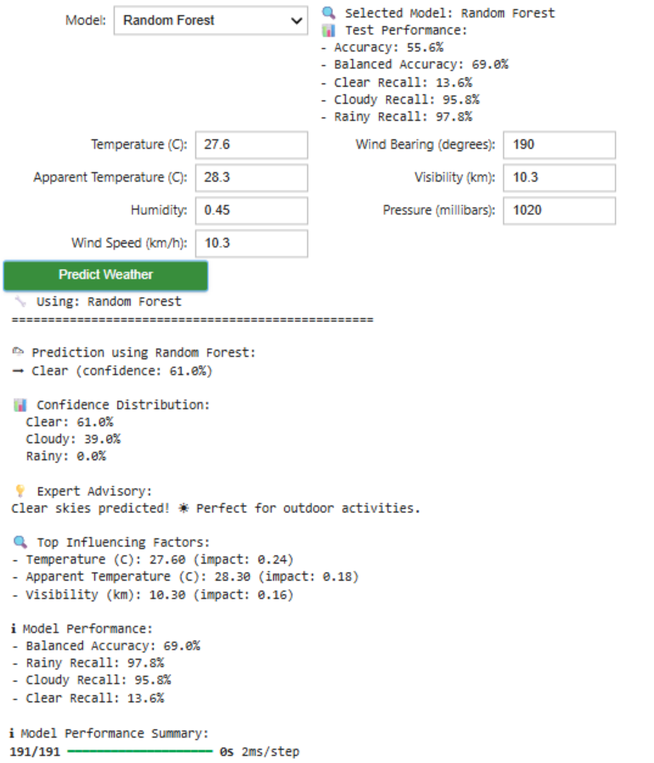


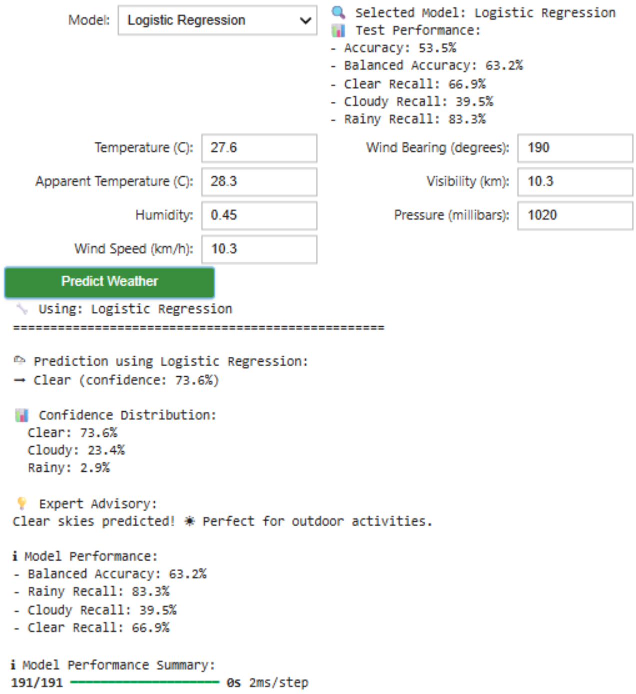

```
Model Fit Diagnostics:
```

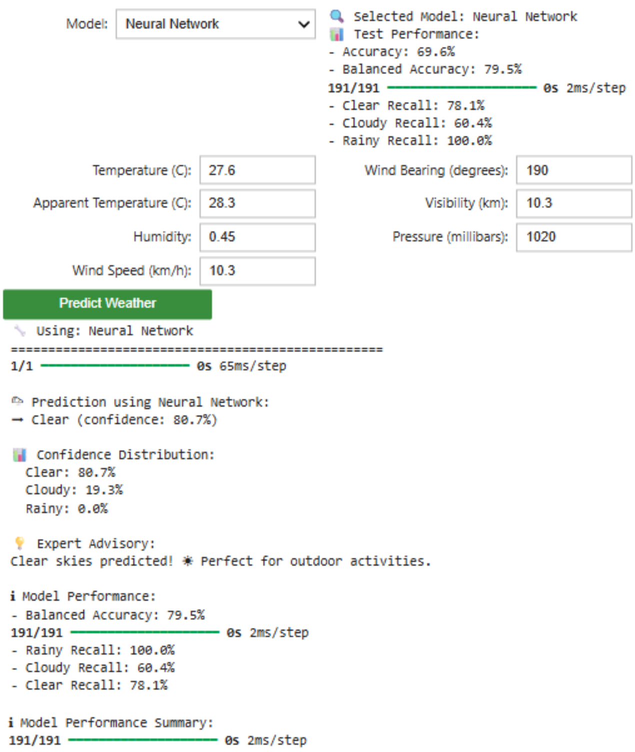
Comprehensive Model Fit Analysis

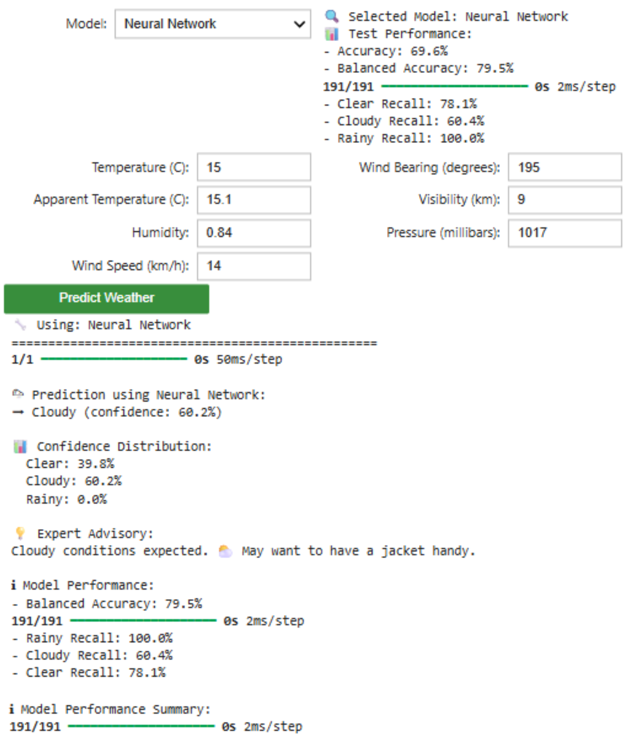
	Model	Train Accuracy	Test Accuracy	Train Error	Test Error	Error Difference	Train Balanced Accuracy	Test Balanced Accuracy	Train Log Loss	Test Log Loss	Fit Status
0	Random Forest	56.20%	55.57%	43.80%	44.43%	+0.63%	70.19%	69.04%	0.710	0.719	Good Fit
1	Logistic Regression	54.38%	53.54%	45.62%	46.46%	+0.84%	62.22%	63.24%	0.891	0.902	Good Fit
2	Neural Network	70.41%	69.64%	29.59%	30.36%	+0.77%	79.92%	79.50%	0.563	0.579	Good Fit

User Interface and Prediction









Model: Logistic Regression

🔍 Selected Model: Logistic Regression
📊 Test Performance:
- Accuracy: 53.5%
- Balanced Accuracy: 63.2%
- Clear Recall: 66.9%
- Cloudy Recall: 39.5%
- Rainy Recall: 83.3%

Temperature (C): 15
Apparent Temperature (C): 15.1
Humidity: 0.6
Wind Speed (km/h): 14
Wind Bearing (degrees): 195
Visibility (km): 8.4
Pressure (millibars): 1002

Predict Weather

🔧 Using: Logistic Regression
==

🌥 Prediction using Logistic Regression:
→ Cloudy (confidence: 50.1%)

📊 Confidence Distribution:
Clear: 39.4%
Cloudy: 50.1%
Rainy: 10.5%

💡 Expert Advisory:
Cloudy conditions expected. 🌥 May want to have a jacket handy.

ℹ Model Performance:
- Balanced Accuracy: 63.2%
- Rainy Recall: 83.3%
- Cloudy Recall: 39.5%
- Clear Recall: 66.9%

ℹ Model Performance Summary:
191/191 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step

Model: Logistic Regression

🔍 Selected Model: Logistic Regression
📊 Test Performance:
- Accuracy: 53.5%
- Balanced Accuracy: 63.2%
- Clear Recall: 66.9%
- Cloudy Recall: 39.5%
- Rainy Recall: 83.3%

Temperature (C): 13.5
Apparent Temperature (C): 12.9
Humidity: 0.9
Wind Speed (km/h): 12.6
Wind Bearing (degrees): 180
Visibility (km): 3.9
Pressure (millibars): 1011

Predict Weather

🔧 Using: Logistic Regression
==

🌥 Prediction using Logistic Regression:
→ Rainy (confidence: 80.0%)

📊 Confidence Distribution:
Clear: 3.7%
Cloudy: 16.2%
Rainy: 80.0%

💡 Expert Advisory:
⚠ High confidence of rain! ☂ Carry an umbrella and waterproof gear.

Model: Random Forest

🔍 Selected Model: Random Forest
📊 Test Performance:
- Accuracy: 55.6%
- Balanced Accuracy: 69.0%
- Clear Recall: 13.6%
- Cloudy Recall: 95.8%
- Rainy Recall: 97.8%

Temperature (C): 12.5
Apparent Temperature (C): 11.9
Humidity: 0.9
Wind Speed (km/h): 9
Wind Bearing (degrees): 210
Visibility (km): 3
Pressure (millibars): 1015

Predict Weather

🔧 Using: Random Forest
==

🌥 Prediction using Random Forest:
→ Rainy (confidence: 62.4%)

📊 Confidence Distribution:
Clear: 4.9%
Cloudy: 32.7%
Rainy: 62.4%

💡 Expert Advisory:
Possible rain expected. ☂ Consider carrying an umbrella.

🔍 Top Influencing Factors:
- Temperature (C): 12.50 (impact: 0.24)
- Apparent Temperature (C): 11.90 (impact: 0.18)
- Visibility (km): 3.00 (impact: 0.16)

ℹ Model Performance:
- Balanced Accuracy: 69.0%
- Rainy Recall: 97.8%
- Cloudy Recall: 95.8%
- Clear Recall: 13.6%

Model: Random Forest

🔍 Selected Model: Random Forest
📊 Test Performance:
- Accuracy: 55.6%
- Balanced Accuracy: 69.0%
- Clear Recall: 13.6%
- Cloudy Recall: 95.8%
- Rainy Recall: 97.8%

Model: Neural Network

🔍 Selected Model: Neural Network
📊 Test Performance:
- Accuracy: 69.6%
- Balanced Accuracy: 79.5%

191/191 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
- Clear Recall: 78.1%
- Cloudy Recall: 60.4%
- Rainy Recall: 100.0%

Temperature (C): 12.5
Apparent Temperature (C): 11.9
Humidity: 0.9
Wind Speed (km/h): 9
Wind Bearing (degrees): 210
Visibility (km): 3
Pressure (millibars): 1015

Predict Weather

🔧 Using: Neural Network
==
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step

🌥 Prediction using Neural Network:
→ Rainy (confidence: 73.9%)

📊 Confidence Distribution:
Clear: 4.7%
Cloudy: 21.4%
Rainy: 73.9%

💡 Expert Advisory:
⚠ High confidence of rain! ☂ Carry an umbrella and waterproof gear.

ℹ Model Performance:
- Balanced Accuracy: 79.5%

191/191 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
- Rainy Recall: 100.0%
- Cloudy Recall: 60.4%
- Clear Recall: 78.1%

Performance Commentary and Suggestions

Observations:

- **Logistic Regression** showed good baseline performance but struggled to capture complex nonlinear relationships.
- **Random Forest** performed better, especially in handling slightly imbalanced data without explicit SMOTE. However, it showed signs of overfitting with high training accuracy.
- **Neural Networks** with SMOTE improved prediction of the minority class ('Rainy') significantly. NN2 offered a good trade-off between complexity and performance. NN3 showed signs of **overfitting**, particularly when validation loss started increasing.

Issues Identified:

- **Underfitting**: Logistic Regression underfit due to model simplicity and linearity.
- **Overfitting**: NN3 with deeper layers showed overfitting due to increased model capacity.

Suggestions for Improvement:

- **Regularization**: Introduce dropout layers in the neural networks or apply L2 regularization to reduce overfitting, use deeper networks for better learning.
- **Hyperparameter Tuning**: Use RandomizedSearchCV for optimal parameter selection across all models.
- **Feature Engineering**: Introduce derived features (like humidity/temperature ratio, or interaction terms) to help models learn better.
- **Cross-validation**: Apply k-fold cross-validation instead of a single train-test split to ensure performance robustness.
- **Advanced Architectures**: Try LSTM or GRU-based models if time-series elements are relevant.