Question 1

Part - 1

Weather Forecasting

Team Name: **DataMavericks**

Table of Contents

1 INTRODUCTION

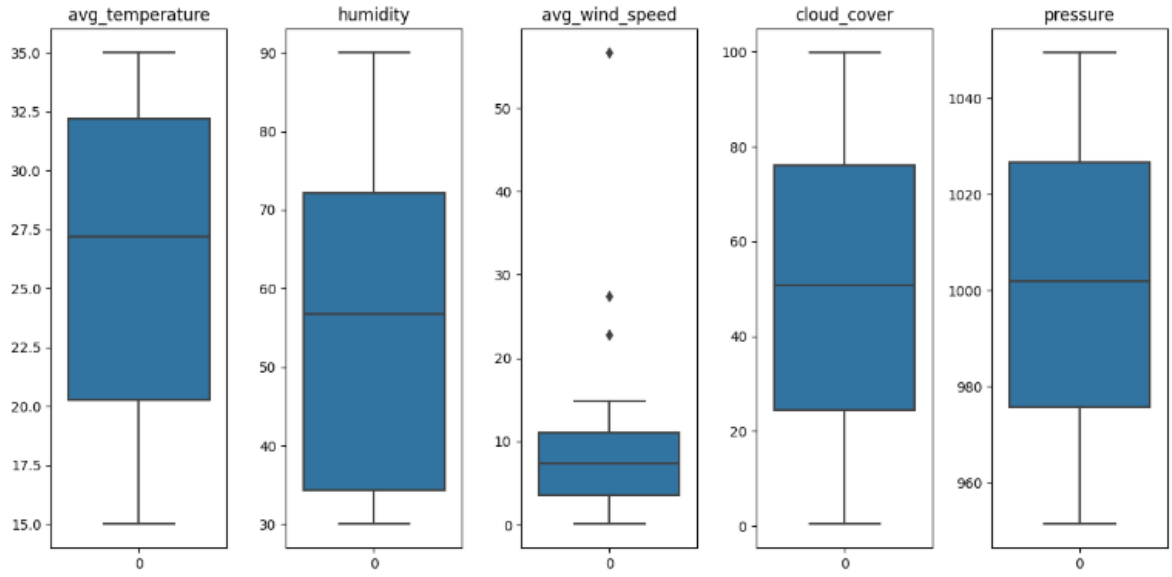
The problem is a binary classification task, where the model needs to classify weather conditions into two categories: "rain" or "no rain." The report will discuss the methodologies used for data preprocessing, model selection, and evaluation, as well as the results obtained from different models (logistic regression, decision trees, random forests, and gradient boosting). Also optimizing these models through hyperparameter tuning to select the best model for prediction. The final output will provide the probability of rain for the next 21 days.

2 METHODOLOGY

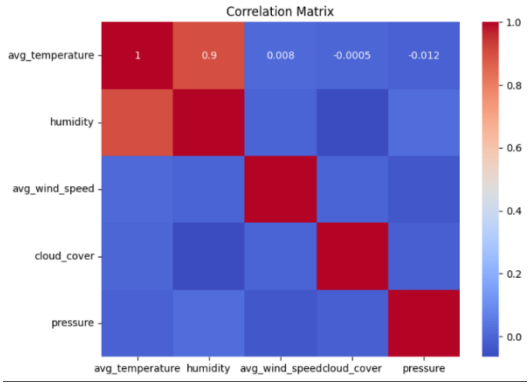
2.1 Exploratory Data Analysis (EDA)

The dataset used for this project consists of 311 entries across seven features: date, avg_temperature, humidity, avg_wind_speed, rain_or_not, cloud_cover, and pressure. The shape of the dataset is (311, 7). Upon initial inspection, the dataset contains null values. These missing values need to be addressed through appropriate methods.

The dataset exhibits class imbalance, with a bias towards the "rain" class, indicating that there are more instances of rainy days than non-rainy days. This imbalance can affect the performance of machine learning models, necessitating techniques like oversampling the minority class or undersampling the majority class to balance the dataset. Visualizing the dataset using box plots revealed the presence of outliers in some features.



Correlation analysis between features can provide insights into which variables are most strongly related to the target variable (rain_or_not). This can guide feature engineering and model selection processes.



2.2 Data Preprocessing

Data preprocessing is a crucial step in preparing the dataset for training machine learning models. It involves handling missing values, encoding categorical variables, and removing irrelevant features.

First, we checked for duplicate values in the dataset. Fortunately, there were no duplicates found, so we proceeded with the next steps. Then the target variable rain_or_not was already encoded as a binary label (1 for rain, 0 for no rain), so no additional encoding was required.

The date column was dropped from the dataset because it does not directly influence the prediction of whether it will rain or not. This decision was made based on the assumption that the other features (temperature, humidity, wind speed, cloud cover, and pressure) are more relevant to rain prediction.

Creating Copies of the DataFrame

To compare the impact of different missing value handling strategies on model performance, we created two copies of the DataFrame:

- DataFrame 1 (Drop Missing Values): This version of the DataFrame had rows with missing values removed.
- DataFrame 2 (Impute Missing Values): In this version, missing values were imputed with the mean of the respective feature.

For the DataFrame where missing values were imputed with the mean, outliers were handled by replacing them with the median of each feature. This method helps stabilize the data and reduce the influence of extreme values on the models.

2.3 Date Splitting

2.3.1 Splitting Data into X and y

After preprocessing, the dataset is split into two parts: X (features) and y (target variable). This is a standard step in machine learning pipelines, where X includes all the predictor variables (avg_temperature, humidity, avg_wind_speed, cloud_cover, and pressure), and y is the variable we want to predict (rain_or_not).

2.3.2 Splitting Data into Training and Testing Sets

The dataset is further divided into training and testing sets using train_test_split. This is done to evaluate the model's performance on unseen data. The training set is used to train the model, while the testing set is used to assess its performance.

2.3.3 Feature Standardization

Feature standardization is performed on all X values after splitting the data into training and testing sets. This is done for several reasons:

- Prevents Feature Dominance
- Improves Model Convergence
- Reduces Risk of Overfitting

Standardization is performed after splitting the data to avoid leaking information from the test set into the training set. If standardization were performed before splitting, the mean and standard deviation calculated from the entire dataset would include information from the test set, which could artificially inflate the model's performance on the test set.

2.3.4 Balancing the Training Data

The training data is balanced using techniques like SMOTE (Synthetic Minority Over-sampling Technique) to address class imbalance issues. Balancing is typically done on the training set only to maintain the natural distribution of the test set, which is used to evaluate the model's performance on unseen data.

2.4 Train and evaluate machine learning models

After preprocessing and splitting the data into training and testing sets, the next step is to train and evaluate different machine learning models. The models chosen for this task are Logistic Regression, Decision Tree Classifier, Random Forest Classifier, and Gradient Boosting Classifier. Each model has its strengths and is suited for binary classification problems like predicting whether it will rain or not.

1. Logistic Regression

Logistic regression is a simple yet effective model for binary classification tasks. It provides interpretable results, showing the relationship between each feature and the probability of rain. It is particularly useful for understanding which variables are most associated with the outcome.

2. Decision Tree Classifier

Decision trees are intuitive models that work well with categorical data and can handle missing values. They are easy to visualize and interpret, making them useful for understanding how different features contribute to the prediction.

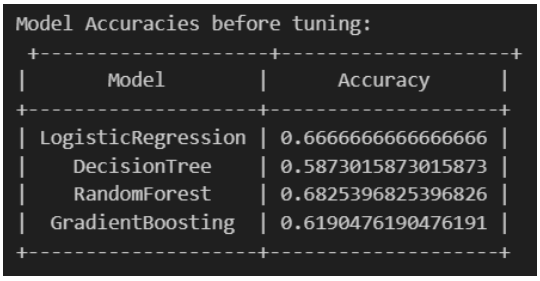
3. Random Forest Classifier

Random forests are ensemble models that combine multiple decision trees to improve the accuracy and robustness of predictions. They reduce overfitting and can handle high-dimensional data.

4. Gradient Boosting Classifier

Why Use It: Gradient boosting is another ensemble method that iteratively adds decision trees to improve predictions. It is highly effective in handling complex interactions between features and is often used in competitions for its high performance.

After training the dataset with the 4 model these are the test accuracy we got,



Model Accuracies before tuning:

Model	Accuracy
LogisticRegression	0.6666666666666666
DecisionTree	0.5873015873015873
RandomForest	0.6825396825396826
GradientBoosting	0.6190476190476191

From this before tuning the model Random Forest perform well that other

2.5 Optimize the model using hyperparameter tuning

After training the models, the next step is to optimize their performance through hyperparameter tuning. This process involves adjusting the parameters of each model to achieve the best possible results on the validation set.

Hyperparameter tuning was performed using GridSearchCV, which systematically tries out different combinations of hyperparameters to find the optimal set for each model.

Best Hyperparameters for Each Model

1. Logistic Regression:
   - C: 0.033
   - max_iter: 100
   - penalty: 'l1'
   - solver: 'liblinear'

2. Decision Tree Classifier:
   - criterion: 'log_loss'
   - max_depth: 30
   - min_samples_leaf: 1
   - min_samples_split: 2
   - splitter: 'best'

3. Random Forest Classifier:
   - max_depth: None
   - min_samples_leaf: 1
   - min_samples_split: 2
   - n_estimators: 100

4. Gradient Boosting Classifier:
    - learning_rate: 1
    - max_depth: 10
    - max_features: 'sqrt'
    - min_samples_split: 5
    - n_estimators: 100

After tuning the hyperparameters, the models were evaluated on the test set to assess their performance.

```
Model Accuracies after tuning:
 +--------------------+--------------------+
|       Model        |   Tuned Accuracy   |
+--------------------+--------------------+
| LogisticRegression | 0.746031746031746  |
|    DecisionTree    | 0.6190476190476191 |
|    RandomForest    | 0.6984126984126984 |
|  GradientBoosting  | 0.6507936507936508 |
+--------------------+--------------------+
```

When compare the Accuracy before and after the tuning, for all the models we can see the accuracy improvement after tuning the model.

2.5.1 Comparing Model Performance on Different DataFrames

These are the results we get from accuracy,

	Model	Null Removed Accuracy	Imputed Accuracy
0	LogisticRegression	0.683333	0.746032
1	DecisionTree	0.433333	0.619048
2	RandomForest	0.533333	0.698413
3	GradientBoosting	0.516667	0.650794

Reasoning

When rows with missing values are dropped, the dataset size is reduced, which can lead to several issues:

- Loss of Information: Dropping rows means losing potentially valuable information that could contribute to better model performance.
- Biased Sample: If the missing values are not missing completely at random (MCAR), dropping them could introduce bias into the dataset, affecting model accuracy and generalizability.

- Reduced Model Complexity: With fewer data points, models might not be able to capture complex patterns as effectively, leading to reduced performance.

Also, we check other evaluation parameter with these two different frames, here are the results,

Results for Null Removed Data:

	Model	Accuracy	F1 Score	Precision	Recall
0	Logistic Regression	0.683333	0.716418	0.800000	0.648649
1	Decision Tree	0.433333	0.514286	0.545455	0.486486
2	Random Forest	0.533333	0.621622	0.621622	0.621622
3	Gradient Boosting	0.516667	0.602740	0.611111	0.594595

Results for Null Imputed Data:

	Model	Accuracy	F1 Score	Precision	Recall
0	LogisticRegression	0.746032	0.764706	0.812500	0.722222
1	DecisionTree	0.619048	0.698795	0.617021	0.805556
2	RandomForest	0.698413	0.736842	0.700000	0.777778
3	GradientBoosting	0.650794	0.734177	0.674419	0.805556

From all this we can say null imputed data frame perform better than null remove data frame. Also for all the evaluation parameter score over 60% score. Which is good model performance.

2.5.2 Selecting the Best Model from the different 4 models

To select the best model, we have consider the following metrics and criteria:

- Accuracy: This measures how well the model predicts the correct outcome overall.
- F1 Score: This is the harmonic mean of precision and recall, providing a balanced measure of both.
- Precision: This measures how accurate the model is when it predicts a positive outcome (rain).
- Recall: This measures how well the model detects all instances of the positive outcome (rain).

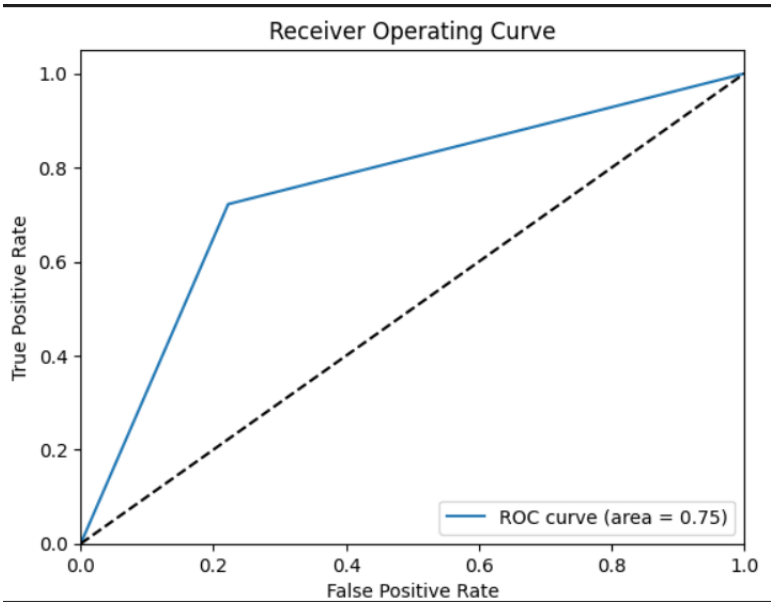
Selection Criteria

- High F1 Score: This indicates a good balance between precision and recall, which is crucial for binary classification tasks.
- High Accuracy: While important, it should be considered alongside other metrics to ensure the model is not biased towards the majority class.
- Specific Needs: Depending on your specific requirements, you might prioritize precision (avoiding false positives) or recall (avoiding false negatives).

Given the importance of both detecting rainy days accurately and avoiding false alarms**, Logistic Regression** is recommended due to its high accuracy and balanced performance metrics.

2.5.3 ROC Curve

The provided Receiver Operating Characteristic (ROC) curve evaluates the performance of the Logistic Regression model in distinguishing between the two classes: "rain" (positive class) and "no rain" (negative class).



The dashed diagonal line represents a random classifier with no discriminatory power. The area under this line is 0.5, which indicates random guessing.

The blue line represents the performance of the Logistic Regression model at various classification thresholds. A curve closer to the top-left corner indicates better performance.

Interpretation of This ROC Curve:

- The ROC curve shows that the Logistic Regression model performs better than random guessing (AUC > 0.5).
- With an AUC of 0.75, the model demonstrates a good ability to distinguish between rainy and non-rainy days, though there is room for improvement.
- The curve's shape indicates that at certain thresholds, the model achieves a good balance between TPR and FPR, making it suitable for binary classification tasks like predicting rain.

3 PREDICTION OF THE RAIN FOR NEXT 21 DAYS

The final step of the project involves predicting the probability of rain for the next 21 days using the trained Logistic Regression model. Since future weather data is not available, we simulate random weather data for each feature to make predictions.

The simulated data was stored in a DataFrame, which was then passed to the trained Logistic Regression model to predict the probability of rain for each day.

Here are the probabilities of rain for each of the next 21 days:

```
Day 1: Probability of Rain: 50.97%
Day 2: Probability of Rain: 45.26%
Day 3: Probability of Rain: 49.24%
Day 4: Probability of Rain: 48.02%
Day 5: Probability of Rain: 48.89%
Day 6: Probability of Rain: 52.30%
Day 7: Probability of Rain: 47.24%
Day 8: Probability of Rain: 50.59%
Day 9: Probability of Rain: 51.99%
Day 10: Probability of Rain: 50.31%
Day 11: Probability of Rain: 50.53%
Day 12: Probability of Rain: 52.62%
Day 13: Probability of Rain: 52.70%
Day 14: Probability of Rain: 48.28%
Day 15: Probability of Rain: 50.11%
Day 16: Probability of Rain: 51.33%
Day 17: Probability of Rain: 50.16%
Day 18: Probability of Rain: 52.89%
Day 19: Probability of Rain: 53.37%
Day 20: Probability of Rain: 49.54%
Day 21: Probability of Rain: 45.95%
```

4 CONCLUSION

This project successfully developed a machine learning pipeline to predict the probability of rain based on historical weather data. Four machine learning models (Logistic Regression, Decision Tree, Random Forest, and Gradient Boosting) were trained and evaluated. Logistic Regression emerged as the best-performing model with an accuracy of 74.60%, a high F1 score of 76.47%, and balanced precision and recall. Using simulated weather data, the model predicted rain probabilities for the next 21 days, demonstrating its practical application.

4.1 Challenges Faced During the Process

- Handling Missing Values and outliers
- Model Selection (Selecting the best model required balancing multiple metrics).
- Class Imbalance
- Simulated Data for Predictions(Since real-time weather data was unavailable, random values within realistic ranges were used for future predictions.)

4.2 Suggestions for Improving Model Performance

- Feature Engineering(Incorporate additional features such as precipitation levels, wind direction, or historical weather patterns to improve prediction accuracy.)
- Real-Time Data Integration:
- Advanced Models (Experiment with more advanced models like XGBoost or neural networks to capture complex relationships in the data.)
- Addressing Outliers Dynamically