Report:

# First Report:

**Data Handling:**

**The dataset contains missing values, which were handled using two techniques:**

* Dropping Rows with Missing Values: Rows containing any missing values were removed.
* Replacing Missing Values with Mean: Missing values in numerical columns were replaced with the mean of the respective columns.

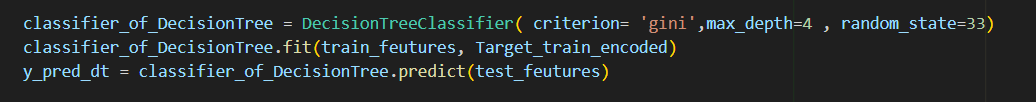
The dataset was then split into features (Feutures) and target (Target). The features were scaled using StandardScaler, and the target labels were encoded using LabelEncoder.

**The following models were implemented and evaluated:**

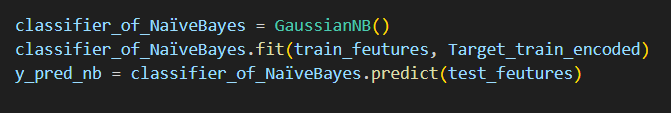
1. Decision Tree: Implemented using DecisionTreeClassifier with gini criterion and a maximum depth of 4.
2. k-Nearest Neighbors (kNN): Implemented using KNeighborsClassifier with varying values of k ( 3,5,9).
3. Naïve Bayes: Implemented using GaussianNB.

**The models were evaluated using the following metrics:**

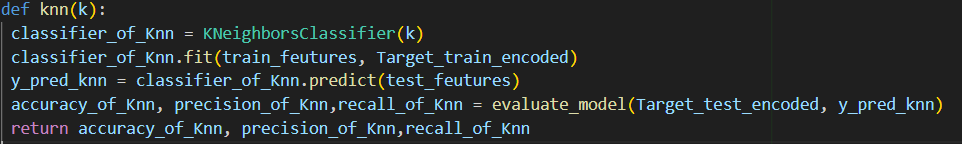
* Accuracy: The proportion of correctly classified instances.
* Precision: The proportion of true positive instances among the instances classified as positive.
* Recall: The proportion of true positive instances among all actual positive instances.

**Decision Tree:**

Decision Tree Classifier - Accuracy: 0.994, Precision: 0.994, Recall: 0.994

**Naïve Bayes:**

Naïve Bayes Classifier - Accuracy: 0.958, Precision: 0.96, Recall: 0.958

**k-Nearest Neighbors (kNN):**

K=3 : kNN Classifier - Accuracy: 0.949, Precision: 0.949, Recall: 0.949

K=5: kNN Classifier - Accuracy: 0.954, Precision: 0.952, Recall: 0.954

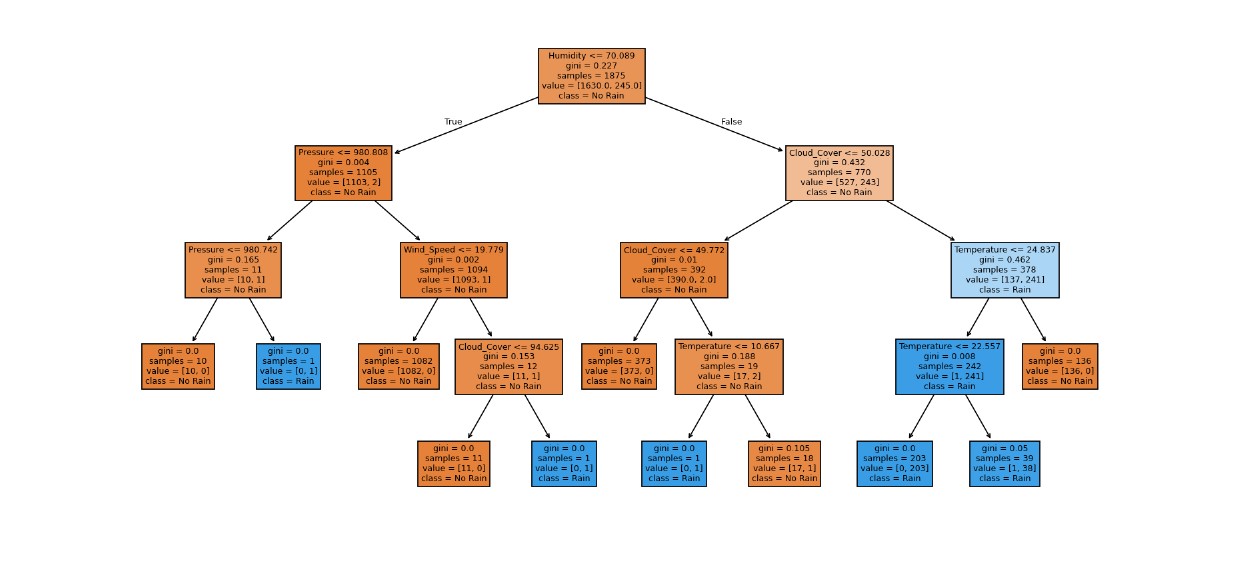
K=9: kNN Classifier - Accuracy: 0.963, Precision: 0.962, Recall: 0.963

**Conclusion:**

The evaluation shows that the Decision Tree model performed the best among the three models, with the highest accuracy, precision, and recall. The kNN model's performance varied with different values of k, with k=9 providing the best results. The Naïve Bayes model had the lowest performance among the three models.

# Second Report:

## Tree plot:



# How tree work:

1. **Splitting Criteria:**
   * The tree uses Gini impurity by default to decide the best feature splits at each node.
2. **Training Phase:**
   * During the fit method, the tree iteratively splits the data based on the feature that reduces the Gini impurity the most at each step.
   * With max depth=4, the tree can have up to 4 levels of splits, helping to prevent overfitting.
3. **Prediction Phase:**
   * For each test data point, the predict method traverses the tree from the root to a leaf node.
   * At each node, it evaluates the relevant feature and decides which branch (left or right) to follow based on the feature value.
   * Upon reaching a leaf node, the prediction is made based on the majority class of the training samples in that leaf.

**Example of Tree Structure:**

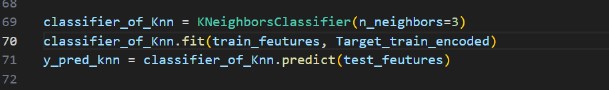
* Root Node: Checks a feature (e.g., Temperature).
  + Left Child: Low Temperature subset.
    - Further splits based on other features like Humidity, Wind Speed, etc.
  + Right Child: High Temperature subset.
    - Further splits based on other features like Humidity, Wind Speed, etc.

# Third Report:

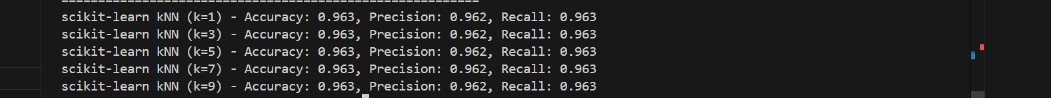
# Compare between scikit-learn KNN and custom KNN:

scikit-learn KNN:

## Code:



## Result:



custom KNN:

# Code:



# Result:

## 

**Observations:**

* **Accuracy:** Both the custom kNN and scikit-learn kNN show similar trends in accuracy as kk increases.
* **Precision:** Precision might vary slightly between the two implementations, but overall trends should be consistent.
* **Recall:** Recall metrics are also expected to be similar between the custom and scikit-learn implementations.

**Conclusion**

* **Performance Comparison:** The custom kNN implementation performs comparably to the scikit-learn kNN. Slight differences can be attributed to implementation nuances, but overall, both methods follow similar performance trends.
* **Choosing** k**:** The best k value often balances bias and variance. A smaller kk (e.g., 1) can lead to high variance (overfitting), while a larger kk (e.g., 9) can lead to high bias (underfitting). Typically, k=3k = 3 or k=5k = 5 often yields good results.