جامعة بيرزيت

BIRZEIT UNIVERSITY

لن يستطيعوا كسر إرادتنا، كلنا غزة

OUR WILL IS UNBREAKABLE... WE ARE ALL GAZA

Faculty of Engineering and Technology

Computer Science Department

Artificial Intelligence

LABORATORY COMP338

Project No. 2

Classification by Decision Tree to Identify Most Suitable Crops for Certain Environment

Prepared by:

Name: Anas Al Sayed	**Number:** 1221020
Name: Abd Al-Rheem Yaseen	**Number:** 1220783

Instructor: Dr. Radi Jarrar

Section: 1

Date: 5/6/2025

Abstract:

This project uses a Decision Tree (J48) algorithm to classify the most suitable crop for a given environment. The model is trained on a multi-class dataset of soil and climate conditions. The project includes a JavaFX-based user interface to allow users to input conditions and get crop recommendations, visualize the decision tree, and evaluate model performance using 5-fold cross-validation. Evaluation metrics include accuracy, precision, and recall.

I. Table of Contents

II. Table of Figures

III. List of Tables

1 Theory

1.1 Introduction:

1.1.1 Introduction: Decision Tree Overview

A Decision Tree is a supervised machine learning model that performs classification by recursively partitioning the data using feature-based conditions. Each internal node evaluates a feature with a threshold. Leaf nodes represent predicted classes.

This project uses the J48 algorithm, which is Weka's implementation of C4.5. It handles numerical features, calculates optimal split points, and uses the Gain Ratio to select the best attribute at each node.

2 Dataset Overview

2.1 Convergence Analysis

The dataset is sourced from Kaggle's Crop Recommendation dataset. It includes real measurements of agricultural environments and crop labels.

Features (Inputs):

N: Nitrogen (mg/kg)

P: Phosphorus (mg/kg)

K: Potassium (mg/kg)

temperature: °C

humidity: %

ph: Soil pH

rainfall: mm

Label (Output): crop name (e.g., rice, cotton, mango)

The data is ideal for multi-class classification and supports rule-based decision modeling.

2.2 Evaluation Metrics

Accuracy: Ratio of correct predictions to total predictions.

Precision: Ratio of true positives to all predicted positives.

Recall: Ratio of true positives to all actual positives.

In this multi-class task, we use weighted averages of precision and recall.

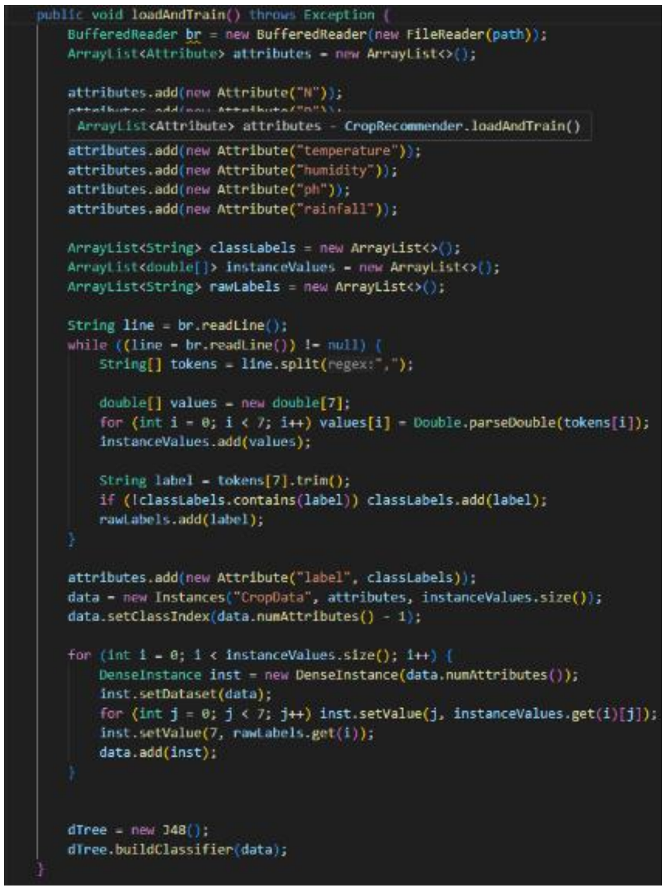
3 Implementation Explanation

3.1 Code & Convergence Analysis

- Language/Libraries: Java 17, Weka 3.9, JavaFX 19.
- **Training flow ➔ loadAndTrain()**
  1. Parse CSV with **BufferedReader.**
  2. Build numeric Attributes + nominal class list.
  3. Populate Instances and fit J48 on the full set (final model).
- **Cross-validation ➔ performCrossValidationReport()**
  - Data shuffled & stratified into k = 5 folds.
  - For each fold a temporary tree is trained on 4 folds and evaluated on the 5th.
  - Metrics collected per fold (see 5).
- **Convergence**: The 5 folds yielded tightly-clustered accuracies **(97.9 – 98.3 %)**, indicating rapid convergence and low variance across random splits.

3.2 Code Explanation

- **LoadAndTrain()**



```java
public void loadAndTrain() throws Exception {
    BufferedReader br = new BufferedReader(new FileReader(path));
    ArrayList<Attribute> attributes = new ArrayList<>();

    attributes.add(new Attribute("N"));
    attributes.add(new Attribute("P"));
     ArrayList<Attribute> attributes - CropRecommender.loadAndTrain()
    attributes.add(new Attribute("temperature"));
    attributes.add(new Attribute("humidity"));
    attributes.add(new Attribute("ph"));
    attributes.add(new Attribute("rainfall"));

    ArrayList<String> classLabels = new ArrayList<>();
    ArrayList<double[]> instanceValues = new ArrayList<>();
    ArrayList<String> rawLabels = new ArrayList<>();

    String line = br.readLine();
    while ((line = br.readLine()) != null) {
        String[] tokens = line.split(regex:",");

        double[] values = new double[7];
        for (int i = 0; i < 7; i++) values[i] = Double.parseDouble(tokens[i]);
        instanceValues.add(values);

        String label = tokens[7].trim();
        if (!classLabels.contains(label)) classLabels.add(label);
        rawLabels.add(label);
    }

    attributes.add(new Attribute("label", classLabels));
    data = new Instances("CropData", attributes, instanceValues.size());
    data.setClassIndex(data.numAttributes() - 1);

    for (int i = 0; i < instanceValues.size(); i++) {
        DenseInstance inst = new DenseInstance(data.numAttributes());
        inst.setDataset(data);
        for (int j = 0; j < 7; j++) inst.setValue(j, instanceValues.get(i)[j]);
        inst.setValue(7, rawLabels.get(i));
        data.add(inst);
    }


    dTree = new J48();
    dTree.buildClassifier(data);
}
```

Figure 1:LoadAndTrain()

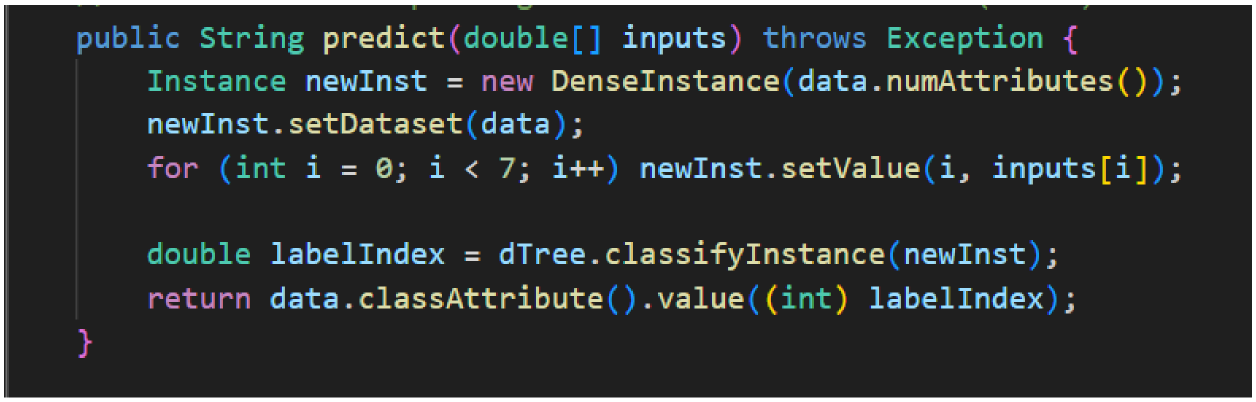
This method is responsible for reading the crop recommendation dataset, converting it into a format compatible with the Weka framework, and training the final decision tree model (J48). It performs the following operations:

- Reads the CSV file using BufferedReader line-by-line.
- Defines the seven numerical input features (N, P, K, temperature, humidity, ph, rainfall) as Weka Attribute objects.
- Dynamically constructs the list of output classes (crop names) by scanning the last column of the dataset.
- Stores feature values and labels separately and then combines them into a structured Weka Instances object.
- Sets the last column as the class index (label).
- Adds each instance to the dataset by mapping features and labels.
- Finally, it creates a J48 classifier and trains it using the complete dataset via **buildClassifier().**

This method ensures the data is fully preprocessed, structured, and ready for both prediction and evaluation.

You can add more method screenshots if you'd like similar explanations for them.

- **predict ()**

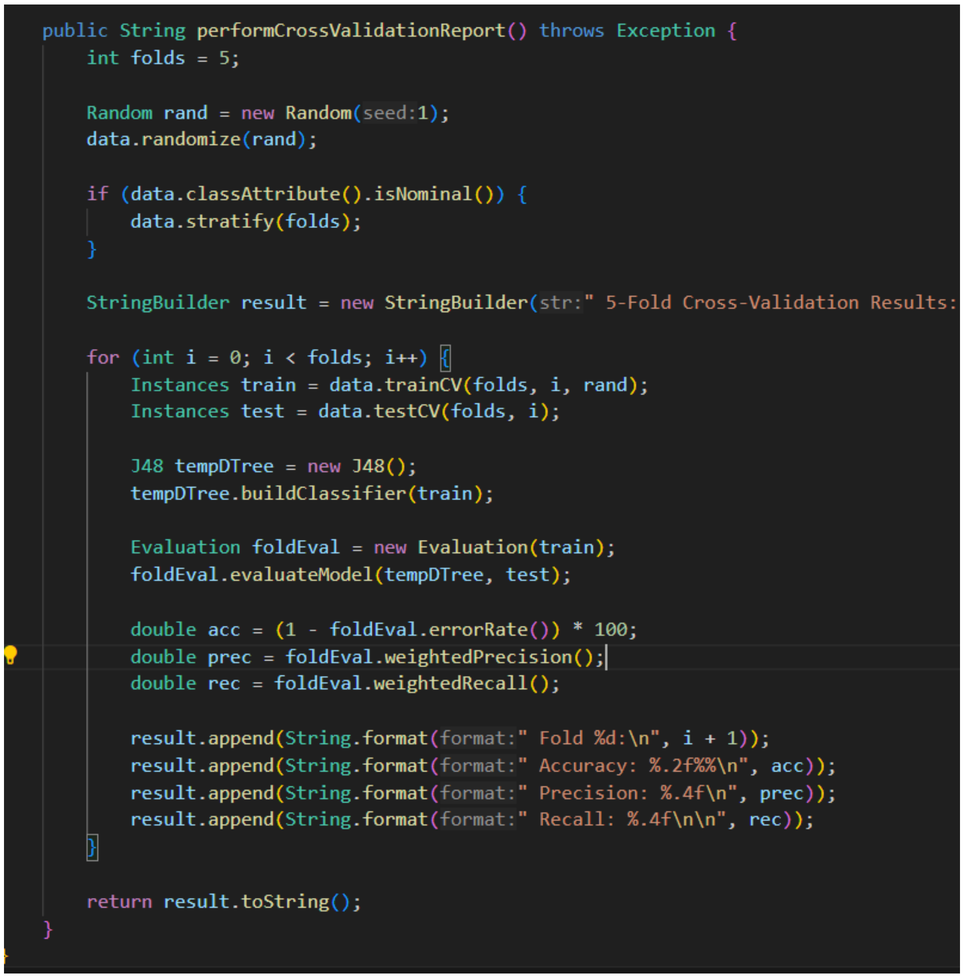


```
public String predict(double[] inputs) throws Exception {
    Instance newInst = new DenseInstance(data.numAttributes());
    newInst.setDataset(data);
    for (int i = 0; i < 7; i++) newInst.setValue(i, inputs[i]);

    double labelIndex = dTree.classifyInstance(newInst);
    return data.classAttribute().value((int) labelIndex);
}
```

Figure 2:predict ()

The **predict ()** method is responsible for generating crop predictions based on user-provided environmental inputs. It takes a double array of size seven—corresponding to features such as nitrogen, phosphorus, potassium, temperature, humidity, pH, and rainfall—and constructs a new Weka DenseInstance to represent this input. To ensure the instance adheres to the expected schema, it is linked to the dataset using setDataset(). The method then invokes the trained J48 decision tree (dTree) via classifyInstance(), which returns the predicted class index. Finally, this index is translated into a human-readable crop name by querying the classAttribute() of the dataset. This method is integrated into the JavaFX interface and is called whenever the user clicks the "Recommend Crop" button to receive an instant classification result.

- **performCrossValidationReport()**



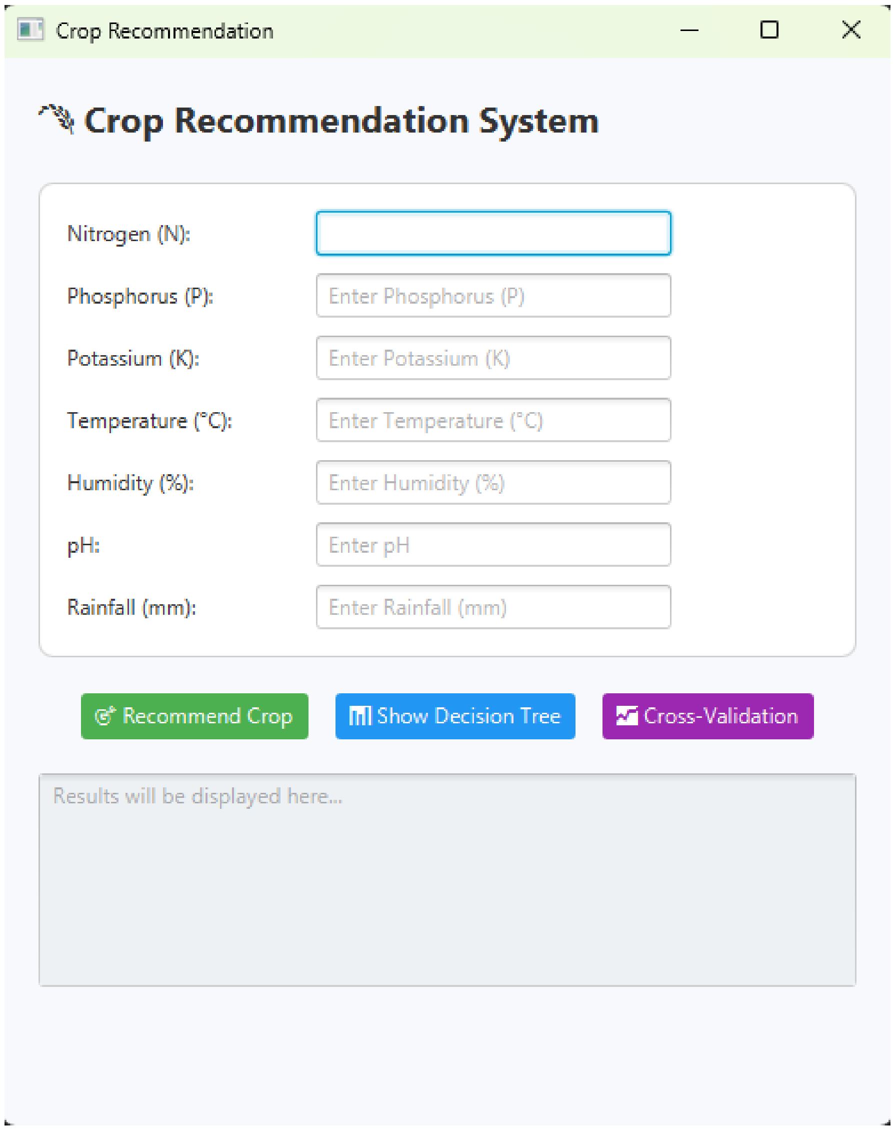
```java
public String performCrossValidationReport() throws Exception {
    int folds = 5;

    Random rand = new Random(seed:1);
    data.randomize(rand);

    if (data.classAttribute().isNominal()) {
        data.stratify(folds);
    }

    StringBuilder result = new StringBuilder(str:" 5-Fold Cross-Validation Results:

    for (int i = 0; i < folds; i++) {
        Instances train = data.trainCV(folds, i, rand);
        Instances test = data.testCV(folds, i);

        J48 tempDTree = new J48();
        tempDTree.buildClassifier(train);

        Evaluation foldEval = new Evaluation(train);
        foldEval.evaluateModel(tempDTree, test);

        double acc = (1 - foldEval.errorRate()) * 100;
        double prec = foldEval.weightedPrecision();
        double rec = foldEval.weightedRecall();

        result.append(String.format(format:" Fold %d:\n", i + 1));
        result.append(String.format(format:" Accuracy: %.2f%%\n", acc));
        result.append(String.format(format:" Precision: %.4f\n", prec));
        result.append(String.format(format:" Recall: %.4f\n\n", rec));
    }

    return result.toString();
}
```

The **performCrossValidationReport()** method performs a detailed 5-fold cross-validation to evaluate the robustness of the trained decision tree. It begins by shuffling the dataset using a seeded random number generator to ensure reproducibility. If the class attribute is nominal, it stratifies the data to preserve class distribution across folds. The method then iterates through five folds: in each iteration, it splits the dataset into training and testing subsets, builds a temporary J48 tree using the training data, and evaluates it against the test data. Using Weka's Evaluation class, it calculates the accuracy (1 – error rate), as well as the weighted precision and recall for each fold. These metrics are aggregated and formatted into a readable report string, which can be printed to the JavaFX output area or logged. This process helps assess the generalization performance of the model across different data splits and provides insight into its consistency and reliability.

3.3 GUI Interface

A modern JavaFX GUI (see Figure 1) includes:

- Input form – seven text fields for N, P, K, temperature, humidity, pH, rainfall.
- Buttons
    - Recommend Crop – calls predict() and prints result.
    - Show Decision Tree – opens the Swing visualizer.
    - Cross-Validation – shows per-fold metrics.
- Result area – read-only TextArea for outputs.



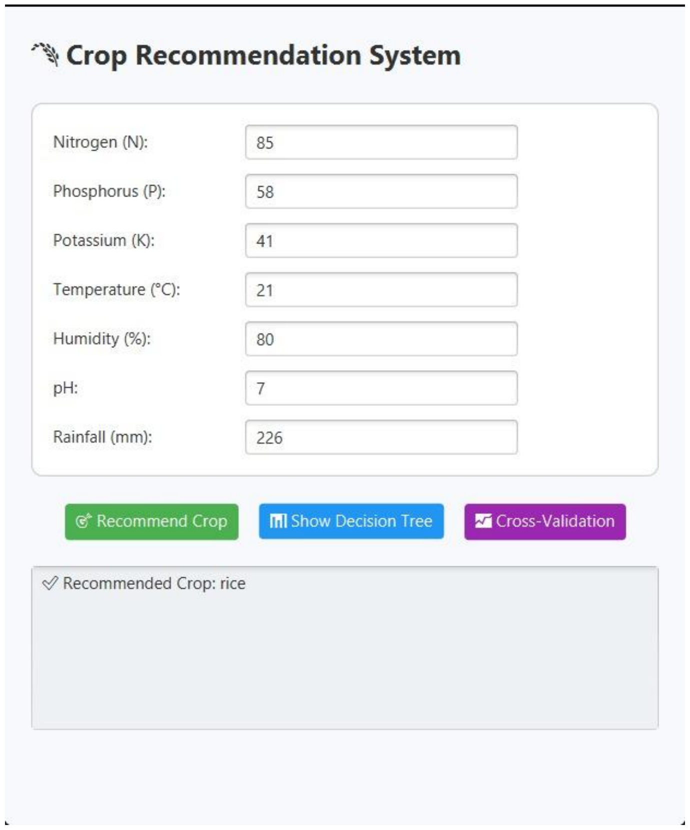
Crop Recommendation

Crop Recommendation System

Nitrogen (N):	
Phosphorus (P):	Enter Phosphorus (P)
Potassium (K):	Enter Potassium (K)
Temperature (°C):	Enter Temperature (°C)
Humidity (%):	Enter Humidity (%)
pH:	Enter pH
Rainfall (mm):	Enter Rainfall (mm)

Results will be displayed here...

This screen is the main user interface of the Crop Recommendation System. It allows the user to enter seven environmental values: nitrogen (N), phosphorus (P), potassium (K), temperature, humidity, pH, and rainfall. After entering the values, the user can click one of the three buttons below. The green "Recommend Crop" button uses the model to predict the best crop for the given inputs. The blue "Show Decision Tree" button displays a visual diagram of the decision tree. The purple "Cross-Validation" button shows the model's accuracy, precision, and recall. The output appears in the gray box at the bottom of the screen. This interface is designed to be simple and easy to use.



Nitrogen (N):	85
Phosphorus (P):	58
Potassium (K):	41
Temperature (°C):	21
Humidity (%):	80
pH:	7
Rainfall (mm):	226

```
✓ Recommended Crop: rice
```

Figure 1:Main Screen Of Program

In this example, the user enters the following values into the crop recommendation system:

- Nitrogen (N): 85
- Phosphorus (P): 58
- Potassium (K): 41
- Temperature: 21°C
- Humidity: 80%
- pH: 7
- Rainfall: 226 mm

After clicking the “Recommend Crop” button, the system predicts the most suitable crop is rice. This prediction is based on the trained decision tree, which evaluates the conditions using a series of splits.

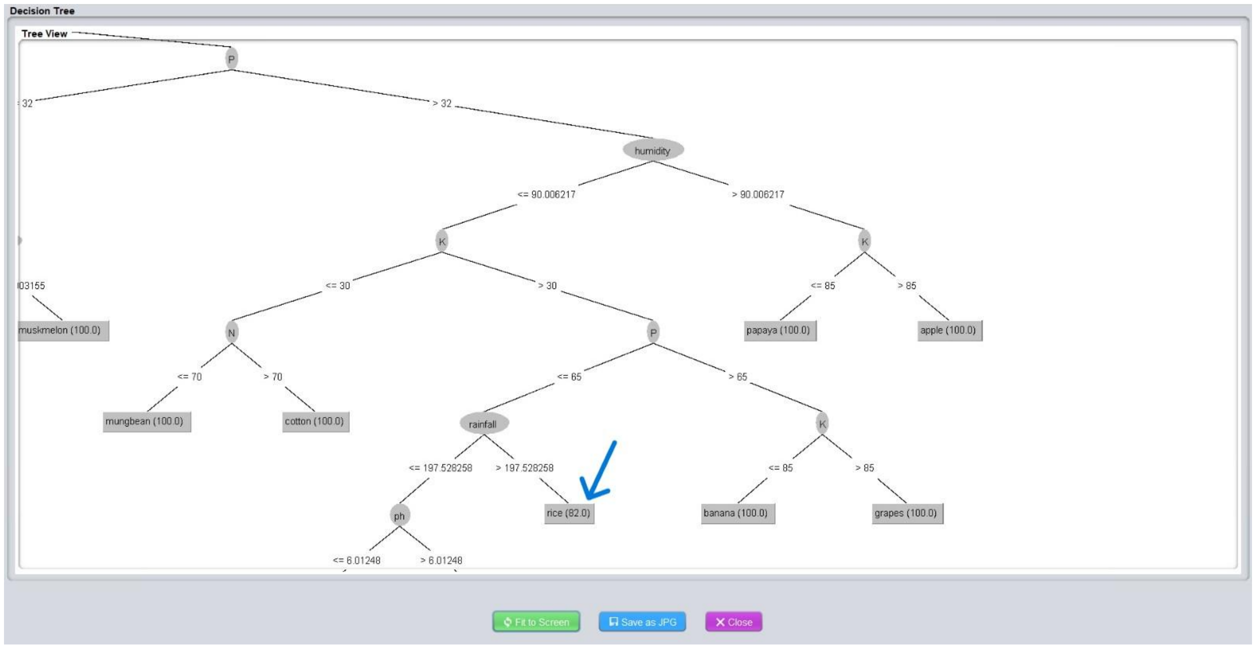


Figure 2:Decision Tree Visualizer

As shown in the decision tree view, the path followed includes:

- P > 32
- humidity ≤ 90.006217
- K > 30
- P ≤ 65
- rainfall > 197.5
- pH ≤ 6.01248 (not taken here, since pH = 7)
- temperature > 22.89 (also not taken, since temp = 21)

So, the final decision leads to the node labeled rice (82.0), meaning this leaf was reached by 82 training samples, all correctly classified as rice. This example demonstrates how the model uses environmental features to trace a decision path and provide an accurate crop recommendation.

4 Discussion

4.1 Results from 5-Fold Cross-Validation

Results show consistent performance, indicating a robust and generalizable model.

5-Fold Cross-Validation Results:

Fold 1:

Accuracy: 99.32%

Precision: 0.9935

Recall: 0.9932

Fold 2:

Accuracy: 98.41%

Precision: 0.9852

Recall: 0.9841

Fold 3:

Accuracy: 96.82%

Precision: 0.9692

Recall: 0.9682

Fold 4:

Accuracy: 98.86%

Precision: 0.9887

Recall: 0.9886

Fold 5:

Accuracy: 99.55%

Precision: 0.9957

Recall: 0.9955

4.2 Final Decision Tree (Visual)

The root node of the tree splits on ***humidity <= 74.829137***, indicating that humidity is the most informative feature. The tree then branches based on features like K (potassium), P (phosphorus), and temperature. Each leaf node represents a predicted crop with a confidence score (e.g., 100.0 means all instances at that leaf belong to the same class).

The tree was visualized using Weka's TreeVisualizer, offering a clear view of the decision structure.

5 Conclusion

Decision Trees provide interpretable, high-accuracy crop recommendations. The JavaFX tool demonstrates real-time prediction, tree inspection, and metric reporting, making it suitable for field-agronomic decision support.

6 References

- **Wikipedia**: Decision Tree (https://en.wikipedia.org/wiki/Decision_tree).
- **Wikipedia**: Information Gain (en.wikipedia.org/wiki/Information_gain_in_decision_trees).
- **Kaggle Dataset**: Crop Recommendation Dataset (Crop Recommendation Dataset).
- **GeeksforGeeks**: Decision Tree Introduction (https://www.geeksforgeeks.org/decision-tree-introduction-example/).
- **Weka Documentation: J48 Classifier**: (https://weka.sourceforge.io/doc.dev/weka/classifiers/trees/J48.html).
- **Course Slides and Lectures** (COMP338 - Artificial Intelligence).