Wine Classification Using Decision Trees

Weka Implementation

---

1. Introduction

The goal of this project was to classify different types of wines based on their chemical and physical properties. We used a dataset containing these attributes to train a decision tree model. The analysis was performed using both a custom implementation and the **Weka** software with the J48 algorithm (a derivative of C4.5). This allowed us to compare the performance of our implementation with a well-optimized tool, evaluating metrics such as accuracy, error rate, and model complexity.

---

2. Dataset

- **Dataset Name:** `wine_data`.
- **Number of Instances:** 178 (each row represents a wine).
- **Attributes:** 14, including 13 predictive features and 1 target class.
    - Predictive features:
        - `alcohol`, `malic_acid`, `ash`, `alcalinity_of_ash`, `magnesium`, `total_phenols`, `flavanoids`, `nonflavanoid_phenols`, `proanthocyanins`, `color_intensity`, `hue`, `od280/od315_of_diluted_wines`, `proline`.
    - Target attribute: `class` (indicating wine type: `class_0`, `class_1`, or `class_2`).

---

3. Methodology

3.1 Implementation in Weka

- **Software Used:** Weka.
- **Classifier:** J48 decision tree.
- **Model Configuration:**
    - Parameter `-C 0.25`: Confidence factor for pruning.
    - Parameter `-M 2`: Minimum number of instances per leaf.
- **Evaluation Method:** 10-fold cross-validation.
- **Steps Performed:**
    1. Imported the dataset in `.arff` format.
    2. Selected the J48 algorithm in the `Classify` tab.
    3. Ran the model and analyzed results, including tree visualization and metrics.

3.2 Custom Implementation

- A custom decision tree algorithm was developed, based on information gain to select attributes.
- The model was evaluated using the same dataset and 10-fold cross-validation for consistency with the Weka experiments (default weka).

---

4. Results

4.1 Weka's Decision Tree Results

1. **Tree Structure:**
   - Number of leaves: **5**.
   - Total tree size: **9** nodes.
   - Key splits:
     - Main attributes: `flavanoids`, `color_intensity`, `proline`.
     - Example of extracted rules:
       - If `flavanoids <= 1.57` and `color_intensity > 3.8`, then the class is `class_2`.
       - If `flavanoids > 1.57` and `proline > 720` and `color_intensity > 3.4`, then the class is `class_0`.
2. **Model Performance:**
   - **Overall Accuracy: 93.82%**.
   - **Error Rate: 6.18%**.
   - **Metrics by Class:**
     - `class_0`: True Positive Rate (TP Rate): 98.3%, Precision: 93.5%.
     - `class_1`: TP Rate: 94.4%, Precision: 91.8%.
     - `class_2`: TP Rate: 87.5%, Precision: 97.7%.
   - **Common Errors:**
     - 1 instance of `class_0` misclassified as `class_1`.
     - 5 instances of `class_2` misclassified as `class_1`.
3. **Confusion Matrix:**
   - Shows the distribution of predictions:

```
 a   b   c   <-- classified as
58   1   0 |  a = class_0
 3  67   1 |  b = class_1
 1   5  42 |  c = class_2
```

4. **Efficiency:** Time to build the model: **0.05 seconds**.

4.2 Comparison with Custom Implementation

- The custom implementation achieved slightly lower performance, with an overall accuracy of **91.5%**, compared to Weka's **93.82%**.
- The generated tree in our implementation was more complex, with **12 nodes** and **7 leaves**, indicating less efficient pruning.
- Classification errors were similarly distributed but more frequent, especially for `class_2`.

5. Conclusions

1. **Weka's Performance:**
   - The J48 classifier in Weka demonstrated high efficiency and accuracy, with a compact model (9 nodes) and a precision of 93.82%.
   - Classification errors were mainly concentrated in `class_2`, which has fewer instances in the dataset.
2. **Comparison with Custom Implementation:**
   - The custom implementation achieved reasonable performance but was slightly less accurate (91.5%) and produced a more complex tree.
   - This indicates that Weka's pruning and parameter optimization processes are more effective.
3. **Key Takeaways:**
   - Implementing algorithms manually provides a deep understanding of their functionality, but optimized tools like Weka offer superior performance and more efficient designs.

---

6. Appendices

6.1 Tree Generated by Weka