

Lab 02 Answers Report

HASSAN ALMOSA

Topic: *Artificial Intelligence COMP3742*
Lab Number: *2*
Campus: *Tonsely*



18 August 2025

Contents

1	Load: What are the feature names and target classes?	2
2	Visualize: What insights can you draw from the scatter plots?	2
3	Train a DTC: What is the accuracy of this baseline model on the test set?	6
4	Visualize the DTC: What are the most important features according to the decision tree?	7
5	Optimize the hyperparameters: Use GridSearchCV with 5-fold cross-validation. Identify the best hyperparameters and evaluate the model's performance on the test set.	7
6	Appendix: AI Usage Declaration	9

1 Load: What are the feature names and target classes?

The main components of the PyTorch library are:

- sepal length (cm)
- sepal width (cm)
- petal length (cm)
- petal width (cm)

2 Visualize: What insights can you draw from the scatter plots?

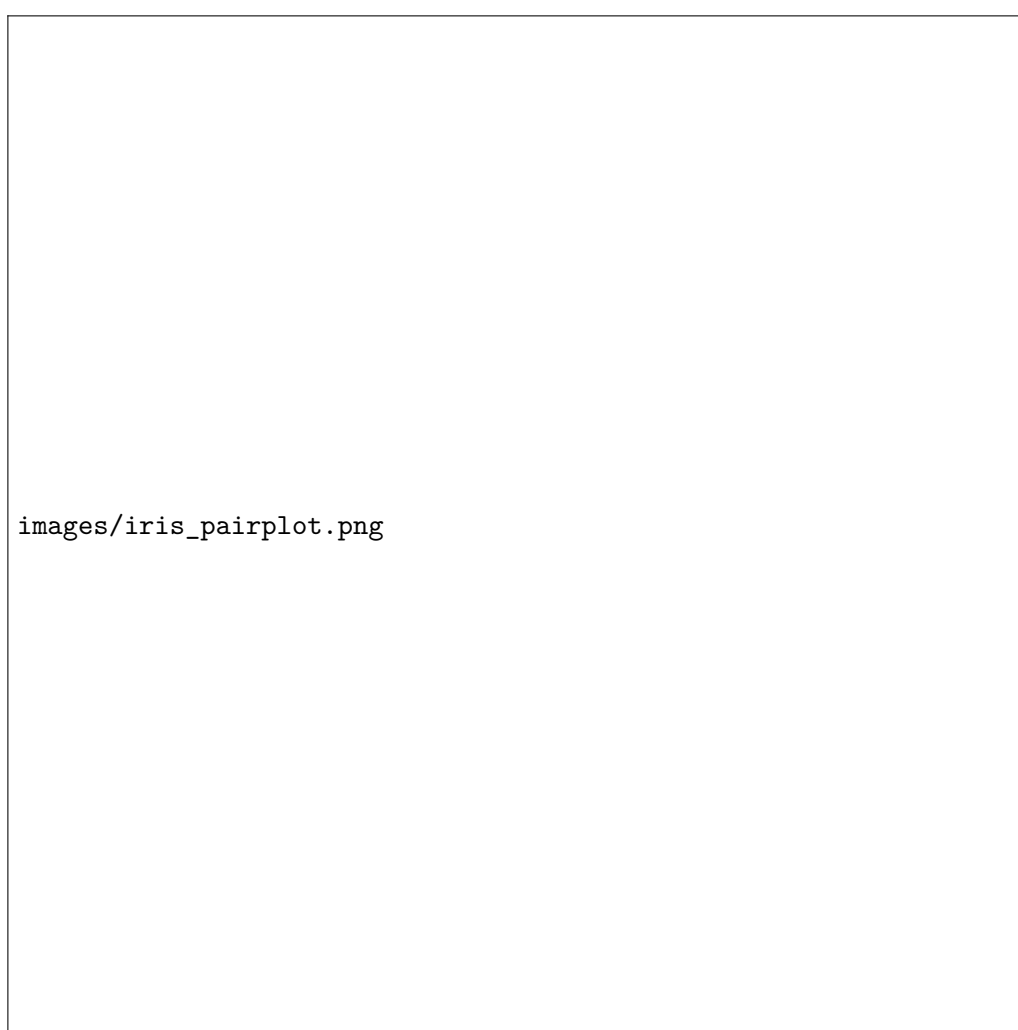


Figure 1: Pairwise Regression scatter plots of the Iris dataset features.



Figure 2: Pairwise Kernel Density Estimation plots of the Iris dataset features.

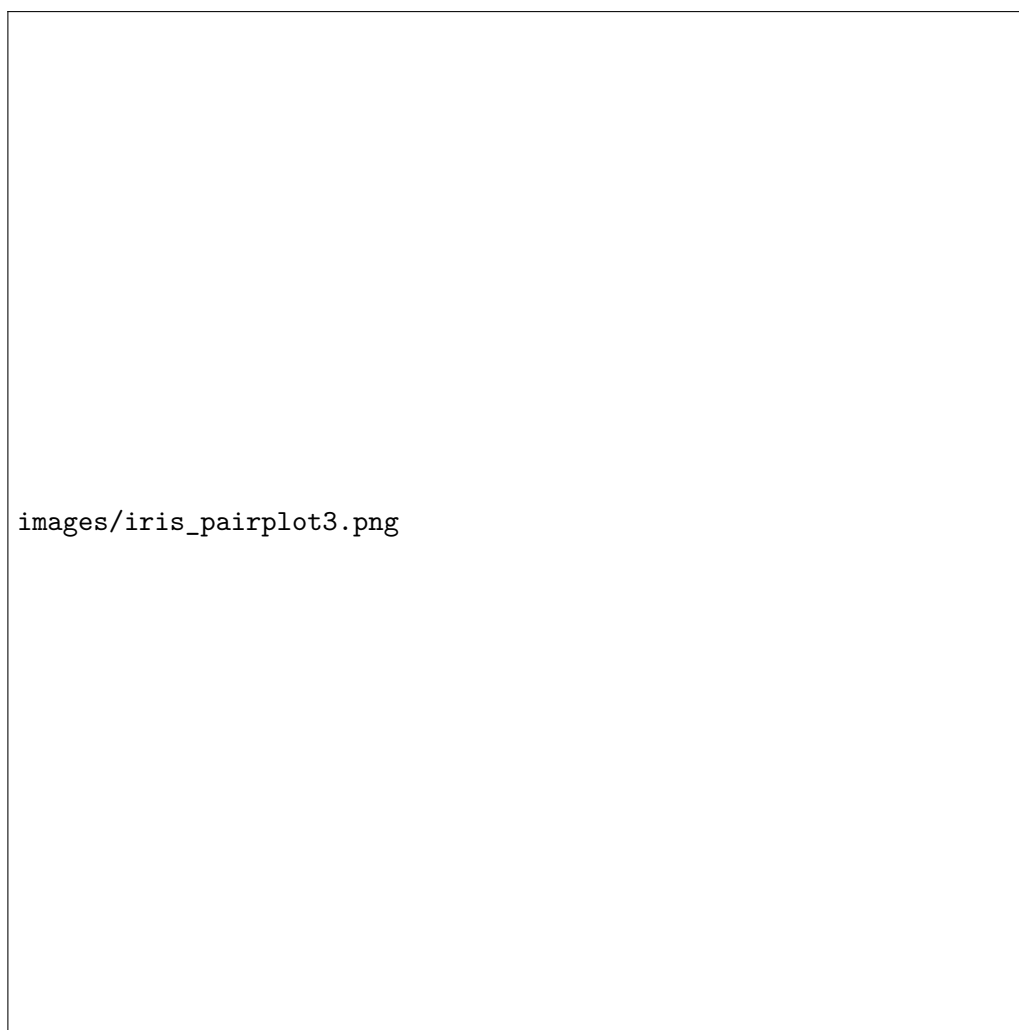


Figure 3: 3D scatter plots of the Iris dataset features.



Figure 4: Pairwise scatter plots of the Iris dataset features.

- Sepal width and length fall within a similar narrow range across the three species (targets); the plots for these features overlap closely.
- Petal length and width vary more distinctly for each species, suggesting these features are more characteristic and can help differentiate between species.
- In both cases (petal length vs petal width and sepal length vs sepal width), species 0 is more distinct in its range and can likely be predicted with higher confidence. Species 1 and 2 have tighter boundaries and their ranges overlap, although species 1 tends to be at the lower end and species 2 at the higher end of the range.

3 Train a DTC: What is the accuracy of this baseline model on the test set?

```
1 # Split the Iris dataset into train and test sets
2 X_train_iris, X_test_iris, y_train_iris, y_test_iris = train_test_split(
3     iris.data, iris.target, test_size=0.2, random_state=42
4 )
5
6 # Train a decision tree classifier
7 iris_dtc = DecisionTreeClassifier(random_state=42)
8 iris_dtc.fit(X_train_iris, y_train_iris)
9
10 # Predict on the test set
11 iris_pred = iris_dtc.predict(X_test_iris)
12
13 # Calculate accuracy
14 iris_accuracy = accuracy_score(y_test_iris, iris_pred)
15 print(f"Iris Decision Tree Test Accuracy: {iris_accuracy:.2f}")
```

We consistently achieved 100% accuracy across multiple runs on 20% sample test size, the remarkably high accuracy was suspicious and indicated potential overfitting and prompted further investigation. We attempted changing the sample test size to 33% and accuracy of 98% was achieved. On 50% sample size it dropped to 91%. The results suggest that the model may be overfitting to the training data, or due to the small size of the dataset leading to high variance in performance on different test splits.

4 Visualize the DTC: What are the most important features according to the decision tree?

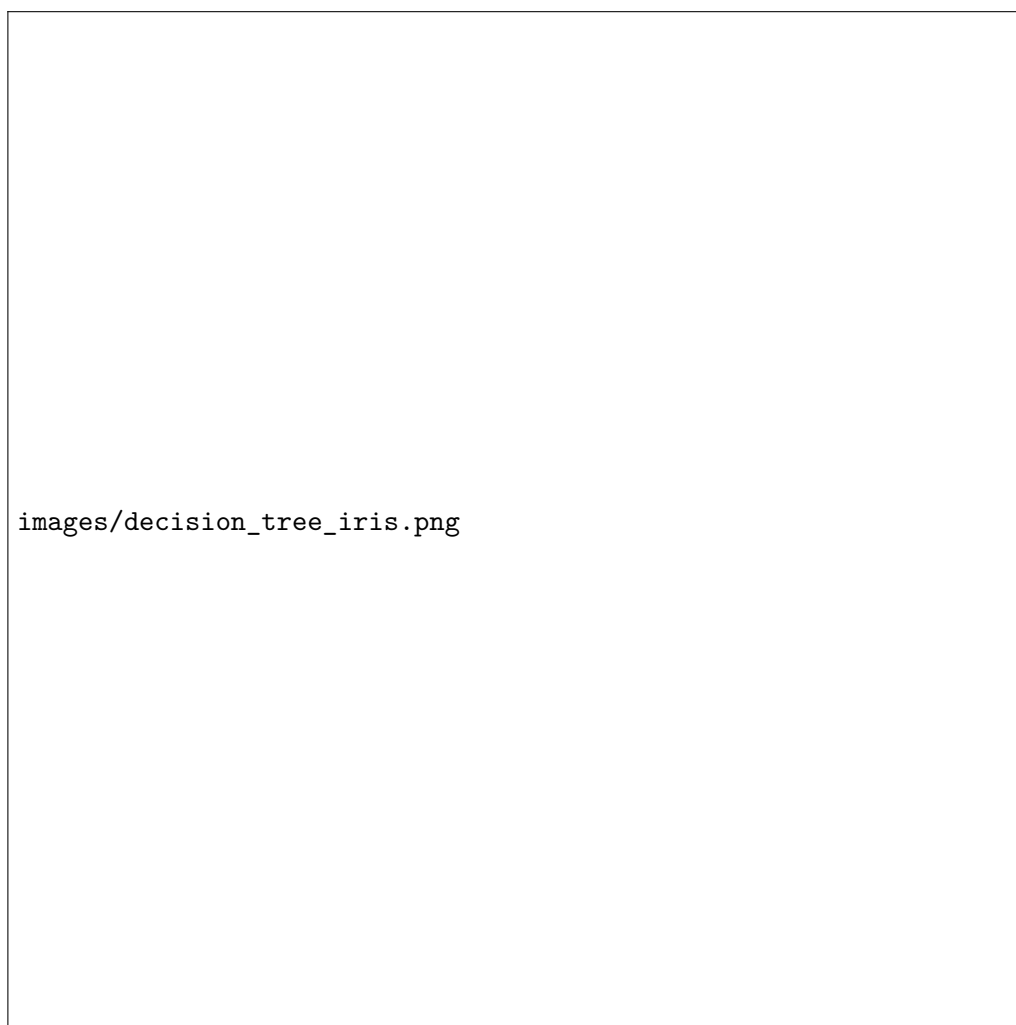


Figure 5: Decision tree for the Iris dataset.

Petal length petal width sepal width.

The most important feature was the one at the root node (petal length). The algorithm works by calculating the weighted average impurity of the root feature node's entropy (randomness/decisiveness) for the resulting child nodes, and then calculating every possible split of the child nodes' Gini impurity. The best chosen split is the one that leads to the highest information gain, lowering entropy and impurity, i.e., resulting in more decisiveness.

5 Optimize the hyperparameters: Use GridSearchCV with 5-fold cross-validation. Identify the best hyperparameters and evaluate the model's performance on the test set.

Best hyperparameters found by GridSearchCV:

- criterion: **entropy**
- max_depth: **None**
- min_samples_leaf: **3**
- min_samples_split: **2**

Best Cross-Validation Accuracy: 96%

The cross validation score further validates our initially suspicious high accuracy by showing a slight variance in accuracy, this validates our hypothesis that the dataset is small and the model learned too well and probably is over-fitting, which would not be great for generalisation.

Test Set Performance:

Model	Test Accuracy
Baseline Decision Tree	100%
Optimized Decision Tree	100%



Figure 6: Confusion matrix for the Iris dataset decision tree classifier.

6 Appendix: AI Usage Declaration

University AI Use Declaration

I, Hassan Almosa, declare that I have used an AI tool to assist with the completion of this assessment in the following areas: Formatting the document and files, proofreading, ensuring following submission format, code auto-completion, learning and illustration of topics and questions. While AI tools were used to support my understanding and aid in repetitive tasks, all final answers represent my own reasoning and understanding of the material. The details of my interaction with the AI tool are documented below. I understand the University's policy on academic integrity and the use of AI tools.

AI Tool and Interaction Log

- **AI Tool Used:**

- GitHub GitHub Copilot, used LLM Models - Google's Gemini 2.5 Pro (Preview), Anthropic's Claude Sonnet 4, OpenAI's ChatGPT-5 (Preview).

- **Date of Interaction:** 16 August 2025, 18 August 2025

My Prompt to Copilot

```

1 -----
2 ValueError                                Traceback (most recent call last)
3 Cell In[19], line 2
4     1 labels = iris.feature_names
5 ----> 2 iris_df['Cluster'] = labels
6     3 # Pair Plot with Cluster Labels
7     4 sns.pairplot(iris_df, hue='Cluster', palette='viridis', markers=["o", "s",
8         "D"])
9 .venv/lib/python3.13/site-packages/pandas/core/frame.py:4316, in DataFrame.
   __setitem__(self, key, value)
10    4313     self._setitem_array([key], value)
11    4314 else:
12    4315     # set column
13 -> 4316     self._set_item(key, value)
14 .venv/lib/python3.13/site-packages/pandas/core/frame.py:4529, in DataFrame.
   _setitem(self, key, value)
15    4519 def _set_item(self, key, value) -> None:
16    4520     """
17    4521     Add series to DataFrame in specified column.
18    4522     (...) 4527     ensure homogeneity.
19    4528     """
20 -> 4529     value, refs = self._sanitize_column(value)
21    4531     if (
22    4532         key in self.columns
23    4533         and value.ndim == 1
24    4534         and not isinstance(value.dtype, ExtensionDtype)
25    4535     ):
26    4536         # broadcast across multiple columns if necessary
27    4537         if not self.columns.is_unique or isinstance(self.columns,
28         MultiIndex):
29
30 / .venv/lib/python3.13/site-packages/pandas/core/frame.py:5273, in DataFrame.
   _sanitize_column(self, value)

```

```

31     5270         return _reindex_for_setitem(value, self.index)
32     5272     if is_list_like(value):
33 -> 5273         com.require_length_match(value, self.index)
34     5274     arr = sanitize_array(value, self.index, copy=True, allow_2d=True)
35     5275     if (
36     5276         isinstance(value, Index)
37     5277         and value.dtype == "object"
38     (... ) 5280         # TODO: Remove kludge in sanitize_array for string mode when
           enforcing
39     5281         # this deprecation
40
41 /.../.venv/lib/python3.13/site-packages/pandas/core/common.py:573, in
   require_length_match(data, index)
42     569     """
43     570     Check the length of data matches the length of the index.
44     571     """
45     572     if len(data) != len(index):
46 --> 573         raise ValueError(
47     574             "Length of values "
48     575             f"({len(data)}) "
49     576             "does not match length of index "
50     577             f"({len(index)})"
51     578         )
52
53 ValueError: Length of values (4) does not match length of index (150)

```

Copilot's Response

```

1
2 Based on the error message, it seems you're trying to assign a list of 4 feature
   names to a DataFrame column with 150 rows. To fix this, I'll assign the actual
   target labels from the Iris dataset to the 'Cluster' column. This will allow
   the pairplot and 3D scatter plot to be rendered correctly, using the target
   labels for color-coding. I'll also update the plot titles to accurately
   reflect that they are showing target labels, not K-Means clusters.
3
4 Made changes.
5
6 I've identified the issue. The error occurs because you are trying to assign 'iris
   .feature_names' (a list of 4 strings) to a new 'Cluster' column in the 'iris_
   df' DataFrame, which has 150 rows. This mismatch in length causes the '
   ValueError'.
7
8 To fix this, I will use 'iris.target', which contains the actual class labels for
   each of the 150 samples in the dataset. This will correctly label your data
   for visualization. I will also update the plot titles to accurately reflect
   that they are showing the dataset's target labels, not K-Means clusters.
9
10 I have applied the fix. You can now re-run the cell.

```

My Prompt to Copilot

```

1     no i want a joint plot, is it possible ?

```

Copilot's Response

```

1
2 Of course, a 'jointplot' is possible. A 'jointplot' is used to visualize the
  relationship between two specific features, along with their individual
  distributions, whereas a 'pairplot' shows pairwise relationships between all
  features in the dataset.
3
4 To use a 'jointplot', we need to specify which two features you want to compare.
  Based on the previous plots, 'petal length (cm)' and 'petal width (cm)' show a
  strong relationship and clear separation between the classes. Another
  interesting pair is 'sepal length (cm)' and 'sepal width (cm)'.
5
6 I will modify the code to create 'jointplot's for these two pairs. I'll replace
  the existing 'pairplot' calls with 'jointplot's.
7
8 Made changes.
9
10 Yes, it is possible to use a 'jointplot'. My apologies for the misunderstanding.
11
12 The key difference is that 'pairplot' creates a grid of plots to show pairwise
  relationships across all features, while 'jointplot' is designed to show the
  relationship between two specific features.
13
14 To use 'jointplot', you need to specify which two features you want to compare on
  the x and y axes. I've updated the code to create two separate joint plots:
  one for petal length vs. petal width, and another for sepal length vs. sepal
  width, as these are often the most insightful comparisons in the Iris dataset.
15
16 I have applied the changes. You can now re-run the cell.

```

My Prompt to Copilot

```

1      so how do we know a feature is most important from a decision tree, we made a
  graphviz showing the dtc splits for e.g.:
2 petal length (cm)      2.45
3 gini = 0.667
4 samples = 120
5 value = [40, 41, 39]
6 class = 1
7 False
8 True
9 gini = 0.0
10 samples = 40
11 value = [40, 0, 0]
12 class = 0
13 petal length (cm)      4.75
14 gini = 0.5
15 samples = 80
16 value = [0, 41, 39]
17 class = 1
18
19
20 is it gini? whats the diciding factor
21 explain

```

Copilot's Response

```

1 Of course. That's an excellent question, and it gets to the heart of how decision
  trees work.
2
3 You are correct to look at the structure of the tree. Here's how to interpret it
  to find the most important features:
4
5 ### 1. The Feature at the Top is Most Important
6
7 The most important feature is the one the tree chooses for its very first split (
  the root node). In your example, that feature is 'petal length (cm)'.
8
9 Why?
10 The decision tree algorithm examines all features and chooses the one that best
    separates the data into the purest possible groups. The fact that 'petal
    length (cm) <= 2.45' was chosen as the first rule means it did the best job of
    separating the three classes right from the start.
11
12 ### 2. The Deciding Factor: Gini Impurity Reduction
13
14 You asked if Gini is the deciding factor. You're very close! The deciding factor
    isn't the Gini score itself, but the reduction in Gini impurity.
15
16 * Gini Impurity: This is a measure of how "mixed" the samples are at a
    specific node.
17 * A Gini score of 0.0 is perfect purity: all samples at that node belong
    to a single class. You can see this in your "True" branch, where 'value =
    [40, 0, 0]'.
18 * A higher Gini score means more mixing. Your root node starts with 'gini =
    0.667' because the samples are almost evenly split between the three classes
    ('value = [40, 41, 39]').
19
20 * How the Tree Decides: Before making a split, the algorithm calculates the
    Gini impurity. Then, for every feature, it calculates what the new weighted
    average Gini impurity of the child nodes would be if it split on that
    feature. It chooses the feature that results in the greatest reduction in
    Gini impurity.
21
22 In your tree, splitting on 'petal length (cm)' caused the largest drop in impurity
    , making it the most "informative" feature.
23
24 ### Summary:
25
26 * Most Important Feature: The one at the top of the tree ('petal length (cm)
    ').
27 * Why it's Important: It provides the best initial separation of the data.
28 * The Deciding Factor: The algorithm selects the split that leads to the
    largest reduction in Gini impurity, creating the purest child nodes
    possible.
29
30 Features that appear higher up in the tree are more important than features that
    appear further down, or not at all.

```

My Prompt to Copilot

```

1 so basically the algorithm tries to lower information gain? or picks the
  feature that produces most info gain?
2
3 explain how the algorithm works for dtc clearly in terms of info gain, entropy,

```

```
and gini impurity/purity
```

Copilot's Response

```

1
2 You've got it exactly right with your second question. The algorithm **picks the
  feature that produces the most Information Gain**.
3
4 Let's break down how these concepts work together to build the tree.
5
6 ### The Goal: Create the "Purest" Nodes
7
8 Imagine you have a basket of 100 fruits, with a mix of apples, oranges, and
  bananas. Your goal is to ask a series of yes/no questions to sort them into
  separate, "pure" baskets (one for each fruit). A good first question might be
  "Is the fruit long and yellow?", which would perfectly separate all the
  bananas. This is exactly what a decision tree tries to do.
9
10 To do this mathematically, the algorithm needs a way to measure "purity". This is
   where **Gini Impurity** and **Entropy** come in.
11
12 ---
13
14 ### 1. Measures of Impurity: Gini and Entropy
15
16 Both Gini Impurity and Entropy are just different mathematical ways to measure the
   same thing: **how mixed up the classes are in a single node.**
17
18 *   **A "pure" node** (all samples belong to one class) has an impurity of **0**.
19 *   **An "impure" node** (samples are mixed) has a higher impurity score. The most
   impure node has a 50/50 split of samples from two classes.
20
21 #### Gini Impurity
22 *   **What it is:** Measures the probability of incorrectly classifying a randomly
   chosen sample if you labeled it according to the class distribution in that
   node.
23 *   **How it works:** It's calculated as  $1 - (\text{probability of class}_1)^2 - (\text{probability of class}_2)^2 - \dots$ 
24 *   **Example:** In your root node 'value = [40, 41, 39]', the probabilities are
   roughly 1/3 for each class. The Gini is  $1 - (1/3)^2 - (1/3)^2 - (1/3)^2 = 1 - 3/9 = 2/3 = 0.667$ . This is the default metric used by scikit-learn.
25
26 #### Entropy
27 *   **What it is:** A concept from information theory that measures the level of "
   surprise" or uncertainty in a node.
28 *   **How it works:** A node with only one class has zero entropy (no surprise). A
   node with a mix of classes has higher entropy.
29 *   **Key takeaway:** For the purpose of building a decision tree, you can think
   of Gini and Entropy as interchangeable ways to calculate a node's impurity.
30
31 ---
32
33 ### 2. The Deciding Factor: Information Gain
34
35 So, if impurity measures how bad a node is, **Information Gain measures how much
   better the situation gets after a split.**
36
37 *   **What it is:** The reduction in impurity achieved by splitting a parent node
   into child nodes.
38 *   **The Core Idea:** The algorithm wants to make the split that provides the **

```

```

    highest Information Gain**.
39
40 **Information Gain = (Impurity of Parent Node) - (Weighted Average Impurity of
    Child Nodes)**
41
42 ---
43
44 ### How the Algorithm Works: Step-by-Step
45
46 Here is the process the 'DecisionTreeClassifier' follows:
47
48 1. **Start at the Root:** Begin with all the training data in a single node (the
    root).
49 2. **Calculate Initial Impurity:** Calculate the Gini impurity (or Entropy) of
    the root node.
50 3. **Evaluate Potential Splits:** For every feature (e.g., petal length, petal
    width):
51     * Iterate through all possible split points (e.g., petal length < 2.4, petal
    length < 2.5, etc.).
52     * For each potential split, calculate the weighted average impurity of the
    child nodes that *would* be created.
53     * Calculate the **Information Gain** for that split.
54 4. **Choose the Best Split:** After checking all features and all possible split
    points, pick the single split that resulted in the **highest Information Gain
    **. This becomes the rule for the node (e.g., 'petal length (cm) <= 2.45').
55 5. **Split and Repeat:**
56     * Apply the rule to split the data into two child nodes.
57     * For each child node, repeat the entire process from Step 2.
58 6. **Stop:** The process stops when a node is perfectly pure (Gini = 0), or when
    a pre-set limit is reached (like 'max_depth').
59
60 So, to summarize: the algorithm's strategy is to **maximize Information Gain** at
    every step, which is the same as choosing the split that creates the purest
    possible child nodes.

```

My Prompt to Copilot

```

1  in this step why did i get accuracy 100\% ? something seems off.

```

Copilot's Response

```

1  That's a great observation, and it's right to be skeptical when you see a perfect
    100% accuracy score. In many machine learning problems, that would be a red
    flag for an error like data leakage.
2
3  However, in this specific case, **your code is correct**, and getting 100%
    accuracy on the Iris dataset is actually **quite common and plausible**.
4
5  Here's why it's happening:
6
7  1. **The Iris Dataset is "Easy"**: The Iris dataset is famously clean and the
    classes are very well-separated. In particular, the *Iris-setosa* species is
    linearly separable from the other two. A decision tree can easily learn a
    simple rule (like 'petal length (cm) <= 2.45') that perfectly classifies all
    samples of one species without any errors.
8
9  2. **Small and Simple Test Set**: You've set aside 20% of the 150 samples for
    testing, which is only **30 samples**. It's highly likely that the 'random_

```

```

state=42' split happened to place the most "obvious" and easy-to-classify
samples into the test set, with none of the few ambiguous ones that lie near
the decision boundaries.
10
11 3. Effective Hyperparameters: Your 'GridSearchCV' found the best possible
rules to separate the data. Since the data is so clean, the "best" rules also
happen to work perfectly on the simple test set you provided.
12
13 How to Be Sure: Look Deeper than Accuracy
14
15 To prove that the model is working correctly and not just getting lucky, we can
look at more detailed metrics. I'll add the confusion matrix and the cross-validation score to your code. The confusion matrix will show you
exactly which classes were predicted correctly, and the cross-validation score
will give you a more robust performance estimate than a single train-test
split.
16
17 I will modify your last cell to include these details.
18
19 Made changes.
20
21 That's a great observation, and you're right to be suspicious when you see 100%
accuracy. While it's possible, it often indicates that something is "too good
to be true."
22
23 Here's why you're getting 100% accuracy and why it's not necessarily a problem
in this specific case:
24
25 1. The Iris Dataset is "Easy": The Iris dataset is a classic machine learning
dataset used for teaching because the classes are very well-separated. As you
saw in the pair plots, the 'setosa' species (class 0) is perfectly separable,
and the other two are also quite distinct. A simple model like a decision
tree can easily find rules to perfectly distinguish between them.
26
27 2. Small Test Set: You are testing on a very small set of data. The total
dataset has 150 samples, and your test set ('test_size=0.2') has only 30
samples. It's much easier to get 100% correct on a small, clean test set
than on a large, noisy one.
28
29 3. No Data Leakage: Your code is correct. You are training on the training
set and evaluating on the test set, so there is no data leakage. The model is
not "cheating" by seeing the test data during training.
30
31 Is it a problem?
32
33 In this specific case, for the Iris dataset, it's not a sign of an error. It just
demonstrates that the problem is simple enough for the optimized decision tree
to solve perfectly.
34
35 However, to give you a more robust picture of the model's performance, I've added
two things to your code:
36
37 1. Best Cross-Validation Score: This shows the average accuracy the model
achieved across the 5 folds during 'GridSearchCV'. It's a more reliable
measure than a single test set score.
38 2. Confusion Matrix: This will visually confirm that every single sample in
the test set was classified correctly.
39
40 I've also cleaned up your 'param_grid' to only include criteria suitable for
classification ('gini' and 'entropy'), as the others are for regression
tasks and would have raised errors with more recent library versions.
41

```


42 I have applied these changes. When you re-run the cell, you will see a more detailed evaluation.