LECTURER: Nghia Duong-Trung

MACHINE LEARNING

MACHINE LEARNING
TOPIC OUTLINE

UNIT 5

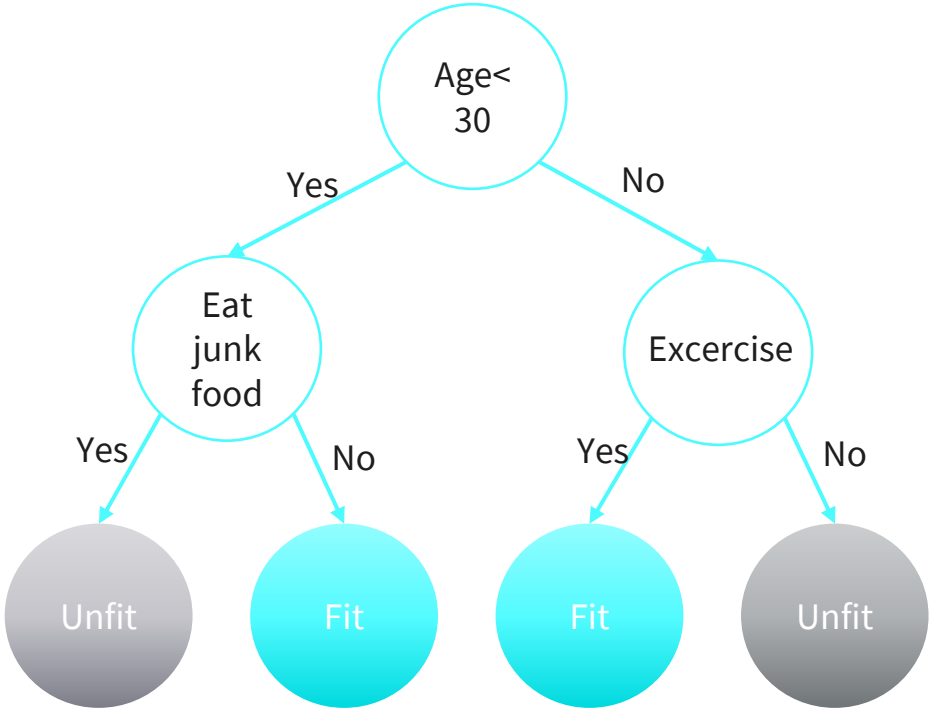
DECISION TREES

STUDY GOALS

- Know the definitions and terms used for decision trees
- Comprehend common applications of decision trees
- Understand different methods of decision trees
- Understand the process of decision tree construction & pruning
- Implement decision tree methods in Python

Decision Trees:

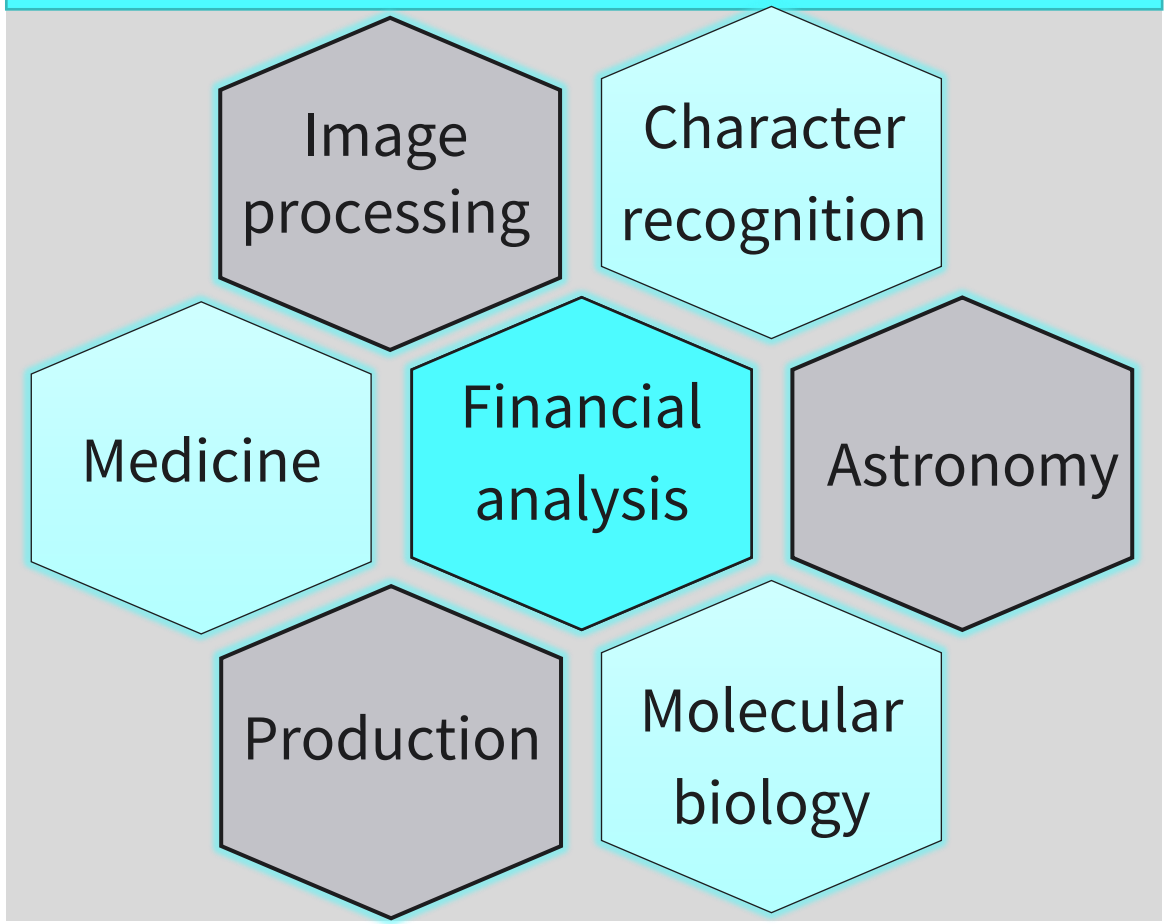
- A **supervised** learning method
- Based on the construction of a **tree like** structure
- A **set of decision rules** to categorize the input objects



Example: Decision tree for predicting a person is fit or unfit

Source of the text: Quinlan, 1986; Zöller, 2022.
Source of image: File: Normdist_regression.png (2023).

Applications of Decision Trees

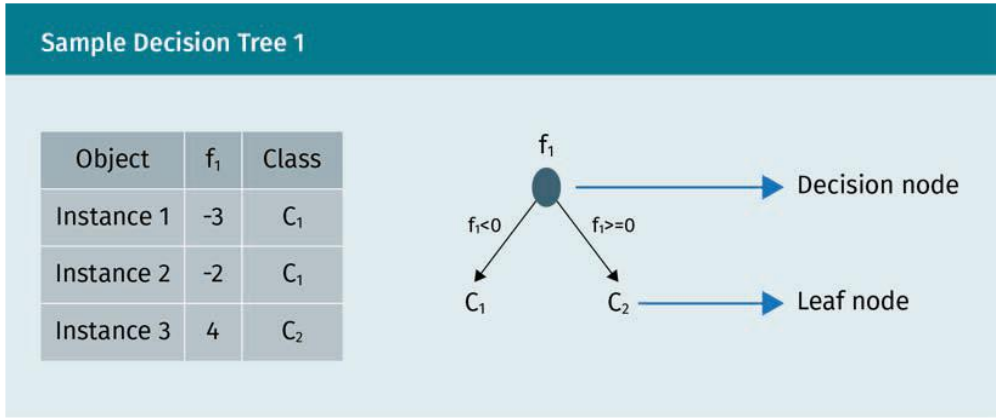


Methods of Decision Trees

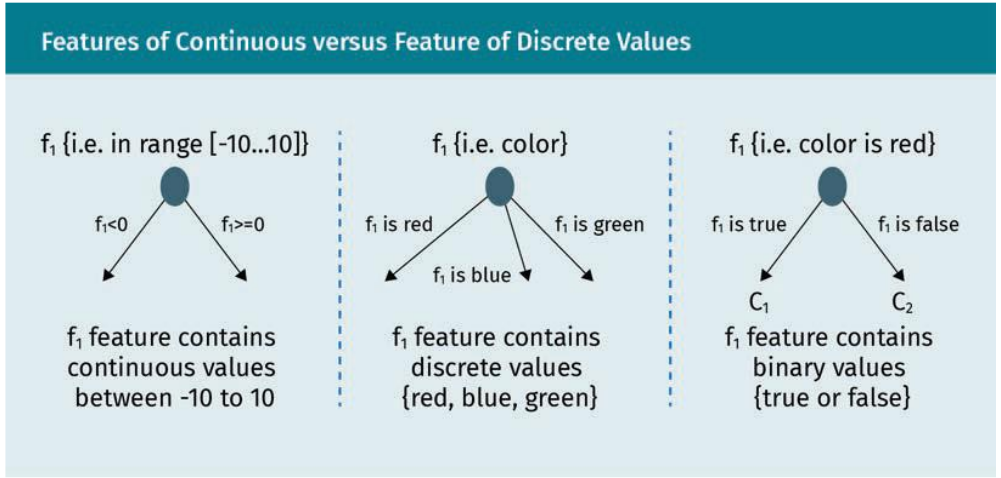
- ID3 & INFORMATION GAIN
- C4.5 & GAIN RATIO
- CART & GINI INDEX
- ID3 & STANDARD DEVIATION

Source of the text: Brian, 2022; Kumari, 2020; Zöller, 2022.

- Two steps: **leaning** & **decision**
- Structure:
  - **Decision nodes** : decision based on features
  - **Branches**: conditional statements (**if**)
  - **Leaf nodes**: classes
- Decision trees work with both **continuous** & **discrete** features



Example of a simple decision tree



Decision trees for continuous and discrete values

Source of the text: Zöller, 2022.
Source of image: Zöller (2022, p.109).

ID3 and Information Gain

- Iterative Dichotomiser (**ID3**) uses information gain (**IG**) to select the best splitting features
- ID3 measures the degree of **homogeneity** of the classes induced by a decision node
- IG is based on the **entropy**, i.e., the degree of disorganization and randomness:

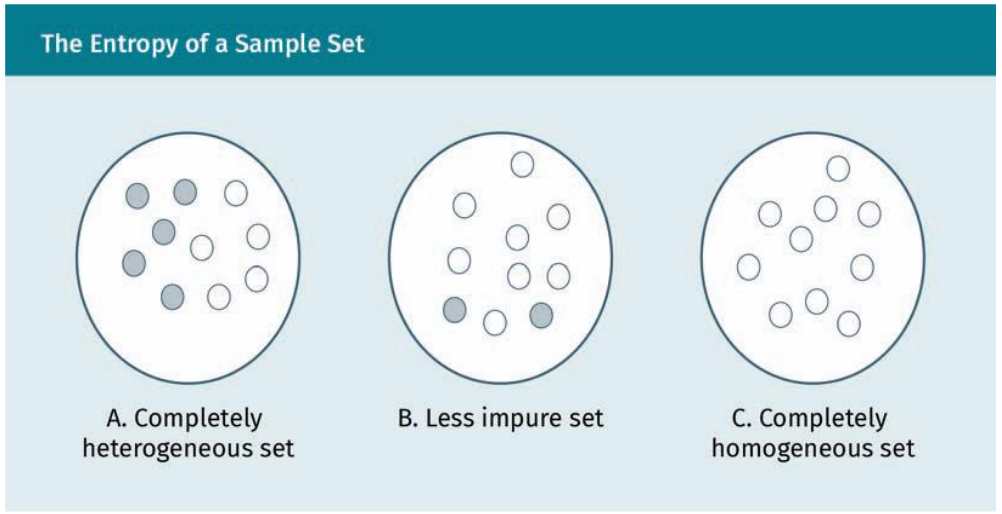
$$Gain(f) = Info(D) - Info_f(D)$$

Where:

$Info(D) = \sum_{k=1}^{m} -P_k \cdot \log(P_k)$: entropy of dataset D

$Info_f(D) = \sum_{i=1}^{v} \frac{|d_i|}{|D|} \cdot Info(d_i)$: entropy of dataset D with respect to feature f which splits D into v subsets d_i

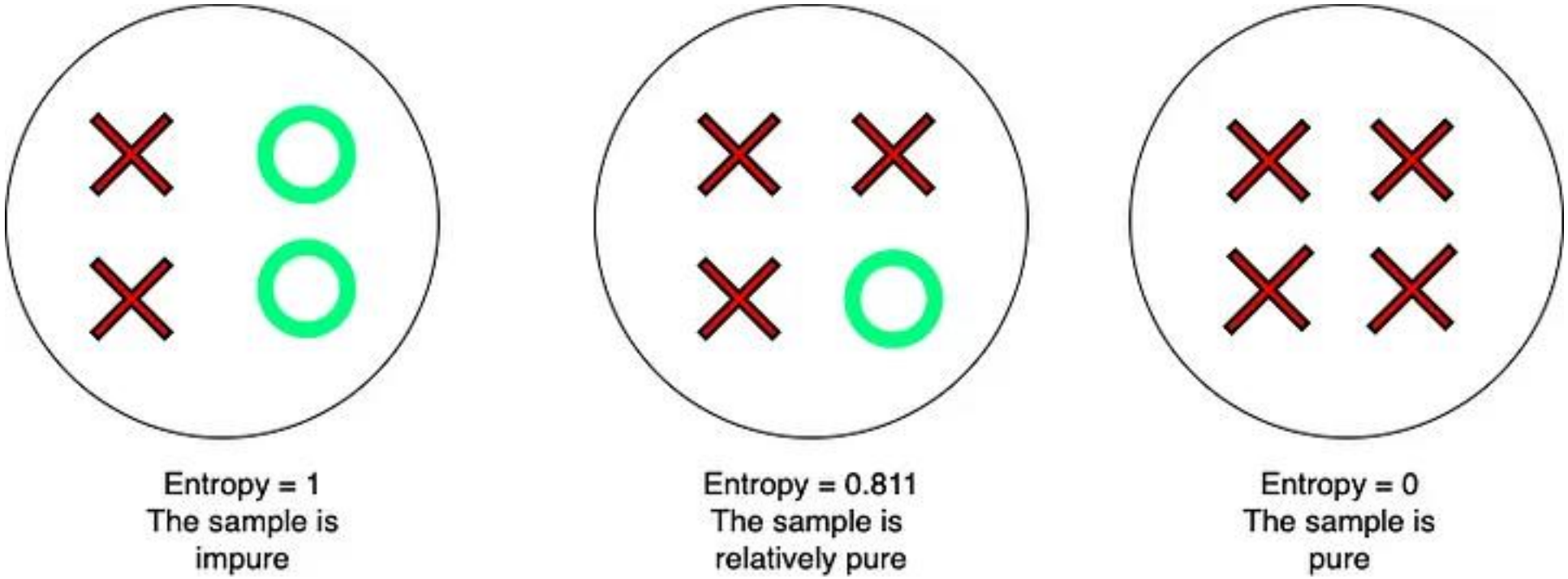
P_k is the probability that any instance in the dataset D belongs to class k



An example of dataset homogeneity

Source of the text: Navlani, 2018; Zöller, 2022.
Source of the image: Zöller (2022, p.112).

In statistics, entropy measures the *level of impurity(heterogeneity) in a dataset*. A fully homogenous dataset has an entropy of 0 whereas a skewed dataset has an entropy closer to 1 as shown in the figure below:



ENTROPY CALCULATIONS

IF we have a set with k different values in it, we can calculate the entropy as follows:

$$\text{Entropy(set)} = \text{I(set)} = -\sum_{i=1}^{k} P(value_i) \cdot log_2(P(value_i))$$

Where $P(value_i)$ is the probability of getting the i^{th} value when randomly selecting one from the set.

Example: a set R = {a,a,a,b,b,b,b,b}

$$\text{Entropy(R)} = \text{I(R)} = -\left[\frac{3}{8} log_2(\frac{3}{8}) + \frac{5}{8} log_2(\frac{5}{8})\right]$$

ENTROPY CALCULATIONS

16 instances: 9 positive, 7 negative

$$\text{I(data)} = -\left[\frac{9}{16} log_2\left(\frac{9}{16}\right) + \frac{7}{16} log_2\left(\frac{7}{16}\right)\right] = 0.9836$$

This makes sense. It's almost a 50/50 split; so, the entropy should be close to 1.

ID	Color	Size	Shape	Target/Class
1	Yellow	Small	Round	1
2	Yellow	Small	Round	0
3	Green	Small	Irregular	1
4	Green	Large	Irregular	0
5	Yellow	Large	Round	1
6	Yellow	Small	Round	1
7	Yellow	Small	Round	1
8	Yellow	Small	Round	1
9	Green	Small	Round	0
10	Yellow	Large	Round	0
11	Yellow	Large	Round	1
12	Yellow	Large	Round	0
13	Yellow	Large	Round	0
14	Yellow	Large	Round	0
15	Yellow	Small	Irregular	1
16	Yellow	Large	Irregular	1

ENTROPY CALCULATIONS

Size

Small

ID	Color	Size	Shape	Target/Class
1	Yellow	Small	Round	1
2	Yellow	Small	Round	0
3	Green	Small	Irregular	1
4	Yellow	Small	Round	1
5	Yellow	Small	Round	1
6	Yellow	Small	Round	1
7	Green	Small	Round	0
8	Yellow	Small	Irregular	1

Large

ID	Color	Size	Shape	Target/Class
1	Green	Large	Irregular	0
2	Yellow	Large	Round	1
3	Yellow	Large	Round	0
4	Yellow	Large	Round	1
5	Yellow	Large	Round	0
6	Yellow	Large	Round	0
7	Yellow	Large	Round	0
8	Yellow	Large	Irregular	1

From ID3 to C4.5

- What is information entropy?
- How Information entropy used to calculate information gain?
- How ID3 decision tree uses information gain?
- Why C4.5 decision tree uses gain ratio over information gain?

C4.5 and Gain Ratio

- C4.5 handles the problem of **generalization** when applying IG for the datasets with **high homogeneity**
- It can deal with both continuous and discrete features
- Method: normalizing the information gain:

$$gainRatio(f) = \frac{gain(f)}{SplitInfo(f)}$$

Where:

$SplitInfo_f(D) = \sum_{i=1}^{v} \frac{|d_i|}{|D|} \cdot log\left(\frac{|d_i|}{|D|}\right)$ - normalizing factor

Source of the text: Zöller, 2022.

C4.5 OBJECTIVE

Decision rules will be found based on entropy and information gain ratio pair of each feature. In each level of decision tree, the feature having the maximum gain ratio will be the decision rule.

WEATHER EXAMPLE

Firstly, we need to calculate global entropy. There are 14 examples; 9 instances refer to yes decision, and 5 instances refer to no decision.

Entropy(Decision) = $\sum$ – p(I) . $\log_2$p(I)
= – p(Yes) . $\log_2$p(Yes) – p(No) . $\log_2$p(No)
= – (9/14) . $\log_2$(9/14) – (5/14) . $\log_2$(5/14) = 0.940

Here, we need to calculate gain ratios instead of gains.

$$gainRatio(f) = \frac{gain(f)}{SplitInfo(f)}$$

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

WIND ATTRIBUTE

Wind is a nominal attribute. Its possible values are weak and strong.

Gain(Decision, Wind) = Entropy(Decision) – ∑ (p(Decision|Wind) . Entropy(Decision|Wind))

Gain(Decision, Wind) = Entropy(Decision) – [
p(Decision|Wind=Weak) * Entropy(Decision|Wind=Weak)]
+ [p(Decision|Wind=Strong) * Entropy(Decision|Wind=Strong)
]

WIND ATTRIBUTE

There are 8 weak wind instances. 2 of them are concluded as no, 6 of them are concluded as yes.
Entropy(Decision|Wind=Weak) = – p(No) * $\log_2$p(No) – p(Yes) * $\log_2$p(Yes)
= – (2/8) * $\log_2$(2/8) – (6/8) * $\log_2$(6/8) = 0.811

Entropy(Decision|Wind=Strong) = – (3/6) * $\log_2$(3/6) – (3/6) * $\log_2$(3/6) = 1

Gain(Decision, Wind) = 0.940 – (8/14)*(0.811) – (6/14)*(1) = 0.940 – 0.463 – 0.428 = 0.049

There are 8 decisions for weak wind, and 6 decisions for strong wind.
SplitInfo(Decision, Wind) = -(8/14)*$\log_2$(8/14) – (6/14)*$\log_2$(6/14) = 0.461 + 0.524 = 0.985

GainRatio(Decision, Wind) = Gain(Decision, Wind) / SplitInfo(Decision, Wind) = 0.049 / 0.985 = 0.049

OUTLOOK ATTRIBUTE

Outlook is a nominal attribute, too. Its possible values are sunny, overcast and rain.

$$Gain(Decision, Outlook) = Entropy(Decision) - \sum (p(Decision|Outlook) * Entropy(Decision|Outlook))$$

$$Gain(Decision, Outlook) = Entropy(Decision) - p(Decision|Outlook=Sunny) * Entropy(Decision|Outlook=Sunny) - p(Decision|Outlook=Overcast) * Entropy(Decision|Outlook=Overcast) - p(Decision|Outlook=Rain) * Entropy(Decision|Outlook=Rain)$$

There are 5 sunny instances. 3 of them are concluded as no, 2 of them are concluded as yes.

Entropy(Decision|Outlook=Sunny) = – p(No) * $\log_2$p(No) – p(Yes) * $\log_2$p(Yes) = -(3/5)*$\log_2$(3/5) – (2/5)*$\log_2$(2/5) = 0.441 + 0.528 = 0.970

Entropy(Decision|Outlook=Overcast) = – p(No) * $\log_2$p(No) – p(Yes) * $\log_2$p(Yes) = -(0/4).$\log_2$(0/4) – (4/4).$\log_2$(4/4) = 0

Entropy(Decision|Outlook=Rain) = – p(No) . $\log_2$p(No) – p(Yes) . $\log_2$p(Yes) = -(2/5).$\log_2$(2/5) – (3/5).$\log_2$(3/5) = 0.528 + 0.441 = 0.970

Gain(Decision, Outlook) = 0.940 – (5/14).(0.970) – (4/14).(0) – (5/14).(0.970) – (5/14).(0.970) = 0.246

There are 5 instances for sunny, 4 instances for overcast and 5 instances for rain

SplitInfo(Decision, Outlook) = -(5/14).$\log_2$(5/14) -(4/14).$\log_2$(4/14) -(5/14).$\log_2$(5/14) = 1.577

GainRatio(Decision, Outlook) = Gain(Decision, Outlook)/SplitInfo(Decision, Outlook) = 0.246/1.577 = 0.155

HUMIDITY ATTRIBUTE

As an exception, humidity is a continuous attribute. We need to convert continuous values to nominal ones. C4.5 proposes to perform binary split based on a threshold value. Threshold should be a value which offers maximum gain for that attribute. Let's focus on humidity attribute. Firstly, we need to sort humidity values smallest to largest.

Now, we need to iterate on all humidity values and seperate dataset into two parts as instances less than or equal to current value, and instances greater than the current value. We would calculate the gain or gain ratio for every step. The value which maximizes the gain would be the threshold.

Day	Humidity	Decision
7	65	Yes
6	70	No
9	70	Yes
11	70	Yes
13	75	Yes
3	78	Yes
5	80	Yes
10	80	Yes
14	80	No
1	85	No
2	90	No
12	90	Yes
8	95	No
4	96	Yes

HUMIDITY ATTRIBUTE

Check 65 as a threshold for humidity

Entropy(Decision|Humidity<=65) = – p(No) * $\log_2$p(No) – p(Yes) * $\log_2$p(Yes) = -(0/1)*$\log_2$(0/1) – (1/1)*$\log_2$(1/1) = 0

Entropy(Decision|Humidity>65) = -(5/13)*$\log_2$(5/13) – (8/13)*$\log_2$(8/13) =0.530 + 0.431 = 0.961

Gain(Decision, Humidity<> 65) = 0.940 – (1/14)*0 – (13/14)*(0.961) = 0.048

SplitInfo(Decision, Humidity<> 65) = -(1/14)*log2(1/14) -(13/14)*log2(13/14) = 0.371

GainRatio(Decision, Humidity<> 65) = 0.126

HUMIDITY ATTRIBUTE

Check 70 as a threshold for humidity

Entropy(Decision|Humidity<=70) = – (1/4)*$\log_2$(1/4) – (3/4)*$\log_2$(3/4) = 0.811

Entropy(Decision|Humidity>70) = – (4/10)*$\log_2$(4/10) – (6/10)*$\log_2$(6/10) = 0.970

Gain(Decision, Humidity<> 70) = 0.940 – (4/14)*(0.811) – (10/14)*(0.970) = 0.940 – 0.231 – 0.692 = 0.014

SplitInfo(Decision, Humidity<> 70) = -(4/14)*$\log_2$(4/14) -(10/14)*$\log_2$(10/14) = 0.863

GainRatio(Decision, Humidity<> 70) = 0.016

HUMIDITY ATTRIBUTE

Check 75 as a threshold for humidity

Entropy(Decision|Humidity<=75) = – (1/5).$\log_2$(1/5) – (4/5).$\log_2$(4/5) = 0.721

Entropy(Decision|Humidity>75) = – (4/9).$\log_2$(4/9) – (5/9).$\log_2$(5/9) = 0.991

Gain(Decision, Humidity<> 75) = 0.940 – (5/14).(0.721) – (9/14).(0.991) = 0.940 – 0.2575 – 0.637 = 0.045

SplitInfo(Decision, Humidity<> 75) = -(5/14).$\log_2$(4/14) -(9/14).$\log_2$(10/14) = 0.940

GainRatio(Decision, Humidity<> 75) = 0.047

HUMIDITY ATTRIBUTE

Gain(Decision, Humidity <> 78) =0.090, GainRatio(Decision, Humidity <> 78) =0.090

Gain(Decision, Humidity <> 80) = 0.101, GainRatio(Decision, Humidity <> 80) = 0.107

Gain(Decision, Humidity <> 85) = 0.024, GainRatio(Decision, Humidity <> 85) = 0.027

Gain(Decision, Humidity <> 90) = 0.010, GainRatio(Decision, Humidity <> 90) = 0.016

Gain(Decision, Humidity <> 95) = 0.048, GainRatio(Decision, Humidity <> 95) = 0.128

Here, I ignore the value 96 as threshold because humidity cannot be greater than this value.
As seen, gain maximizes when threshold is equal to 80 for humidity. This means that we need to compare other nominal attributes and comparison of humidity to 80 to create a branch in our tree.

TEMPERATURE ATTRIBUTE

Temperature feature is continuous as well. When I apply binary split to temperature for all possible split points, the following decision rule maximizes for both gain and gain ratio.
Gain(Decision, Temperature <> 83) = 0.113, GainRatio(Decision, Temperature<> 83) = 0.305

WEATHER EXAMPLE

Let’s summarize calculated gain and gain ratios. Outlook attribute comes with both maximized gain and gain ratio. This means that we need to put outlook decision in root of decision tree.

Attribute	Gain	GainRatio
Wind	0.049	0.049
Outlook	0.246	0.155
Humidity <> 80	0.101	0.107
Temperature <> 83	0.113	0.305

If we will use gain metric, then outlook will be the root node because it has the highest gain value. On the other hand, if we use gain ratio metric, then temperature will be the root node because it has the highest gain ratio value.

CART and Gini Index

- CART uses Gini Index $Gini(D)$ to measure the **impurity** in a dataset D:

$$Gini(D) = 1 - \sum_{k=1}^{m} P_k^2$$

where: P_k - probability that an instance in D belongs to class k

- The feature that maximizes the **impurity reduction** $\Delta Gini(f)$ is selected as an important feature:

$$\Delta Gini(f) = Gini(D) - Gini_f(D)$$

where: $Gini_f(D) = \frac{d_1}{D} Gini(d_1) + \frac{d_2}{D} Gini(d_2)$ - impurity index with respect to the feature f which splits D into d_1 and d_2

Source of the text: Zöller, 2022.

ID3 & Standard deviation

- ID3 can be applied for **regression** problems by using standard deviation reduction instead of IG
- **Standard deviation** S of target c: $S_c = \sqrt{\frac{\sum(c-\bar{c})^2}{n}}$
- Standard deviation based on the feature vector x: $S(c,x) = \sum_{v \in x} P(v) \cdot S_c(v)$
- **Reduction** in standard deviation: $SDR(c,x) = S(c) - S(c,x)$
- **Coefficient** of variation: $CV = \frac{S}{\bar{x}} \cdot 100\%$

Source of the text: Zöller, 2022.

Top-down process to construct a decision tree:

1. Select the highest ranked feature → create the decision node
2. From this node, create the branches with distinct value (range)

– If all instances of this feature value (range) are of the same class:

→ the child node from this branch is a leaf node

– else:

→ repeat step 1 and 2

Source of the text: Zöller, 2022.

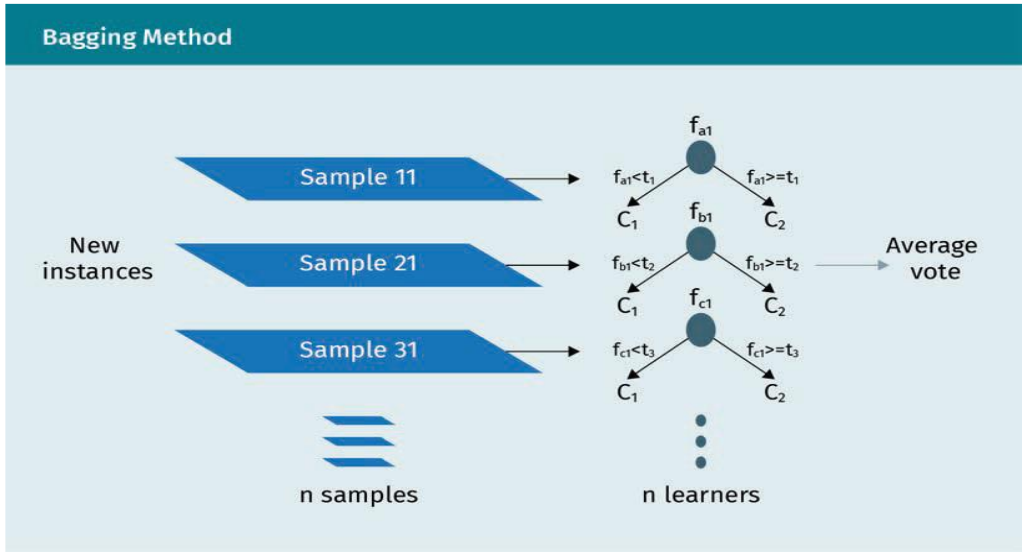
Decision tree **pruning:**

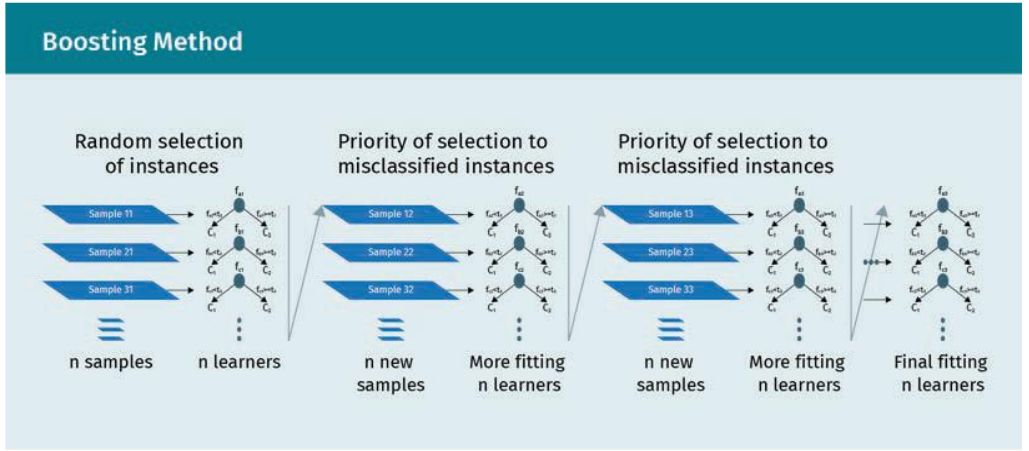
- **handles overfitting** by **decreasing the size** of the tree to make it less complex
- Method: **Removing sub-trees** in the decision tree that have low classification power
- Two types:
  - **Pre-pruning:** avoids building up the low-discriminating sub-trees while the decision tree is being constructed, and replaces with leaf nodes
  - **Post-pruning:** removes spurious sub-trees from the fully constructed decision tree, and replaces with leaf nodes

Source of the text: Zöller, 2022.

Ensemble methods:

- **combine** a series of machine learning algorithms to improve the learning performance
- **train many** weak base-classifiers that are good at different parts of the input space.
- Types:
  - **Homogenous: bagging** (or bootstrapping) & **boosting**
  - **Heterogeneous**





Source of the text: Zöller, 2022.
Source of the image: Zöller (2022, p. 125-126).

5.7. IMPLEMENTATION OF DECISION TREE IN PYTHON

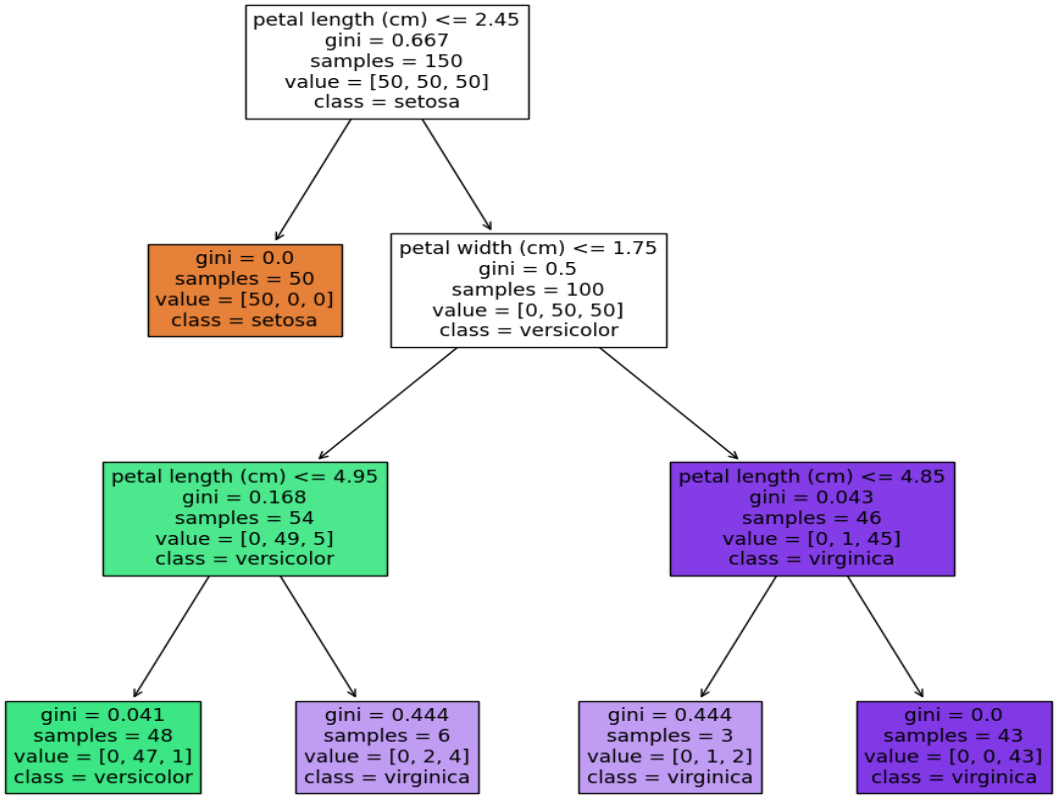
```
# Decision tree classifier
# Import libraries
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

# Load the iris dataset
iris = load_iris()

# Define the decision tree classifier model
clf = DecisionTreeClassifier(max_depth=3)

# Train the model on the iris dataset
clf.fit(iris.data, iris.target)

# Plot the decision tree
fig, ax = plt.subplots(figsize=(12, 12))
plot_tree(clf, filled=True, feature_names=iris.feature_names,
class_names=iris.target_names, ax=ax)
plt.show()
```



Decision tree classifier

Source of the text: Scikit-learn, n.d.

5.7. IMPLEMENTATION OF DECISION TREE IN PYTHON

```
# Decision tree regressor
# Import libraries
import numpy as np
from sklearn.tree import DecisionTreeRegressor, plot_tree
import matplotlib.pyplot as plt
# Create dataset
rng = np.random.RandomState(1)
X = np.sort(5 * rng.rand(100, 1), axis=0)
y = np.sin(X).ravel(); y[::5] += 3 * (0.5 - rng.rand(20))
# Defin regressor
regr = DecisionTreeRegressor(max_depth=2)
# Train & test the model
regr.fit(X, y)
X_test = np.arange(0.0, 5.0, 0.01)[:, np.newaxis];
y_1 = regr.predict(X_test)
# Plot the results
plt.figure()
plt.scatter(X, y, s=20, edgecolor="black", c="yellow", label="data")
plt.plot(X_test, y_1, color="red", label="Regressor", linewidth=2)
plt.xlabel("X") ; plt.ylabel("y") ; plt.title("Decision Tree Regression")
plt.legend(); plt.show()
```

Decision tree regressor

Source of the text: Scikit-learn, n.d.

REVIEW STUDY GOALS

- Know the definitions and terms used for decision trees
- Comprehend common applications of decision trees
- Understand different methods of decision trees
- Understand the process of decision tree construction & pruning
- Implement decision tree methods in Python

SESSION 5

DECISION TREES

TRANSFER TASKS

1. Implement decision tree classifier for Iris dataset with the following configuration:

- Classification criteria: information gain (entropy)
- Maximum depth of the decision tree: 5

2. Create a dataset using the following code:

```
>>> import numpy as np
>>> rng = np.random.RandomState(1)
>>> X = np.sort(10 * rng.rand(80, 1), axis=0)
>>> y = np.sin(X).ravel(); y[::5] += 3 * (0.5 - rng.rand(16))
```

Implement decision tree regressor with the following configuration:

- Regression criteria: absolute error
- Maximum depth of the decision tree: 3

TRANSFER TASK
PRESENTATION OF THE RESULTS

Please present your results.

The results will be discussed in plenary.

1. _____ area decision support tool that use a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.

a) Graphs
b) Trees
c) Decision trees
d) Neural networks

2. The ID3, C4.5, and CART approaches use _____, _____, and _____ techniques for feature ranking, respectively.

a) Gini index, information gain and gain ratio
b) Information gain, gain ratio and Gini index
c) Gain ratio, Gini index and information gain
d) Gain ratio, information gain and Gini index

3. The _____ technique provides a higher ranking value to the features having more distinct values.

a) Information gain
b) Gain ratio and Gini index
c) Gain ratio
d) Gini index

4. The _____ technique provides a higher ranking value to the features that maximize the reduction in the impurity of the partitioned instance sets.

a) Gain ratio
b) Information gain
c) Gain ratio and Gini index
d) Gini index

5. Which of the following is/are true about boosting trees?

(1): In boosting trees, individual weak learners are independent of each other.

(2): It is the method for improving the performance by aggregating the results of weak learners.

a) Option 2 is correct
b) Neither option is correct
c) Option 1 and 2 are correct
d) Option 1 is correct

LIST OF SOURCES

Text:

Zöller, T. (2022). Course Book – Machine Learning. *IU International University of Applied Science.*

Navlani, A. (2018). Decision tree classification in Python. https://www.datacamp.com/community/tutorials/decision-tree-classification-python

Quinlan, J.R.(1986). Introduction of decision trees. Machine Learning, 1(1), 81-106. https://doi.org/10.1023/A:1022643204877

Scikit-Learn. (n.d). Machine Learning in Python. Open-source: https://scikit-learn.org/stable/modules/tree.html#decision-trees

Images:

Zöller (2022).

Zöller (2022, p. 109).

Zöller (2022, p. 112).

Zöller (2022, p. 125-126).