UNC GREENSBORO

CS 405/605 Data Science

Dr. Qianqian Tong

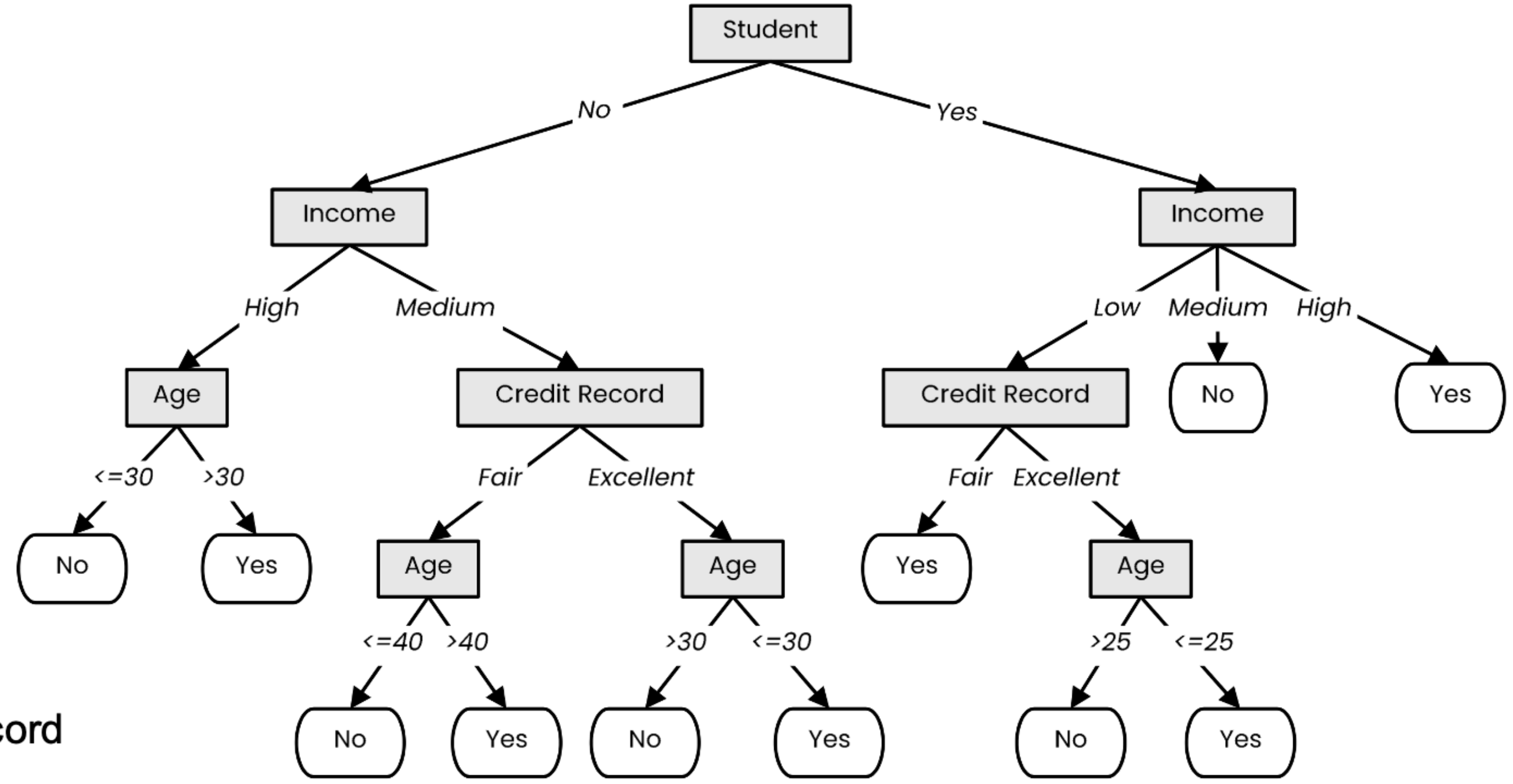
Decision Tree

- A tree-like model that illustrates series of events leading to certain decisions
- Each node represents a test on an attribute and each branch is an outcome of that test

Who to loan?

- Not a student
- 45 years old
- Medium income
- Fair credit record

- Student
- 27 years old
- Low income
- Excellent credit record



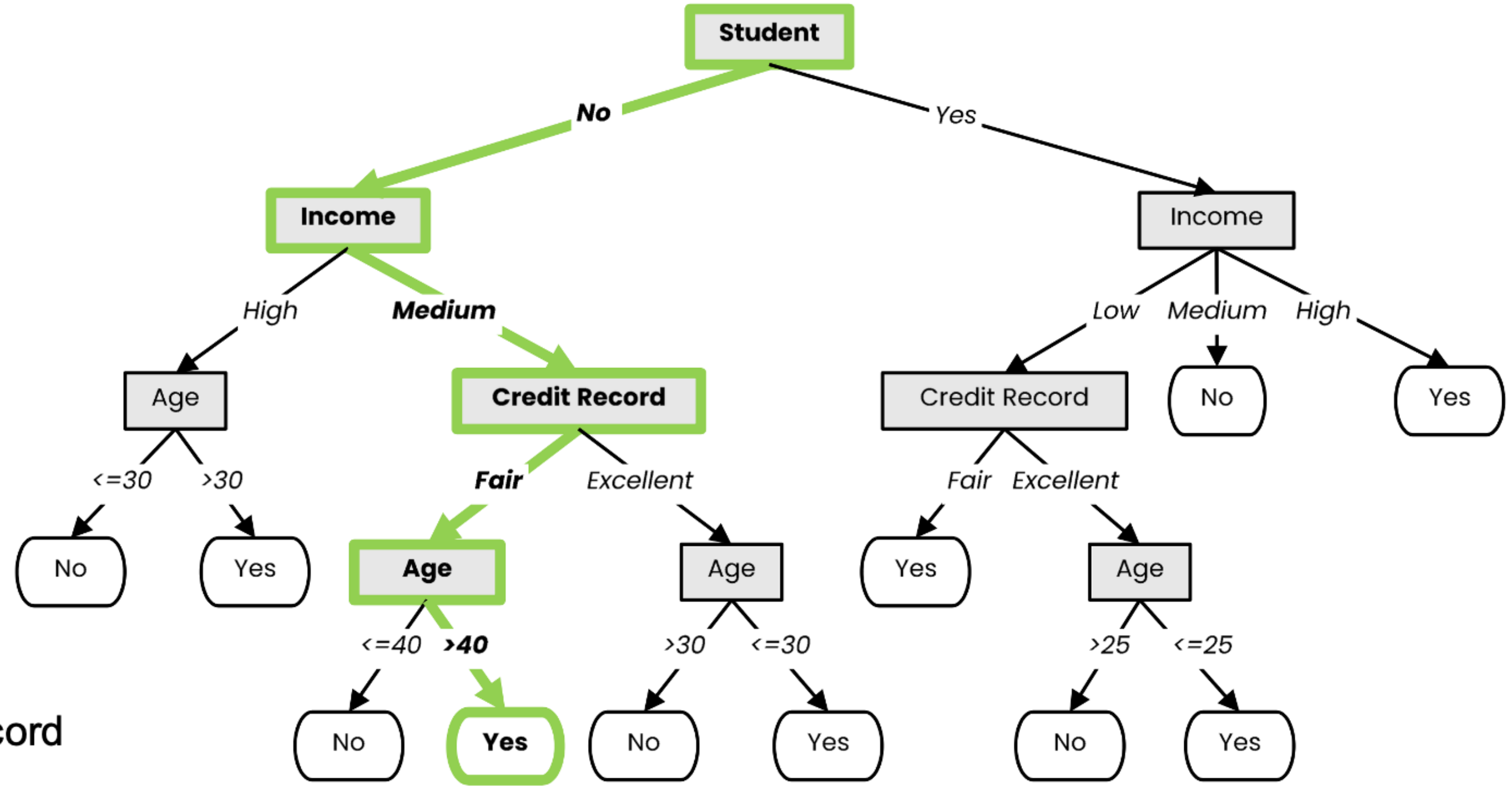
Decision Tree

- A tree-like model that illustrates series of events leading to certain decisions
- Each node represents a test on an attribute and each branch is an outcome of that test

Who to loan?

- Not a student
- 45 years old
- Medium income
- Fair credit record
- ➢ Yes

- Student
- 27 years old
- Low income
- Excellent credit record



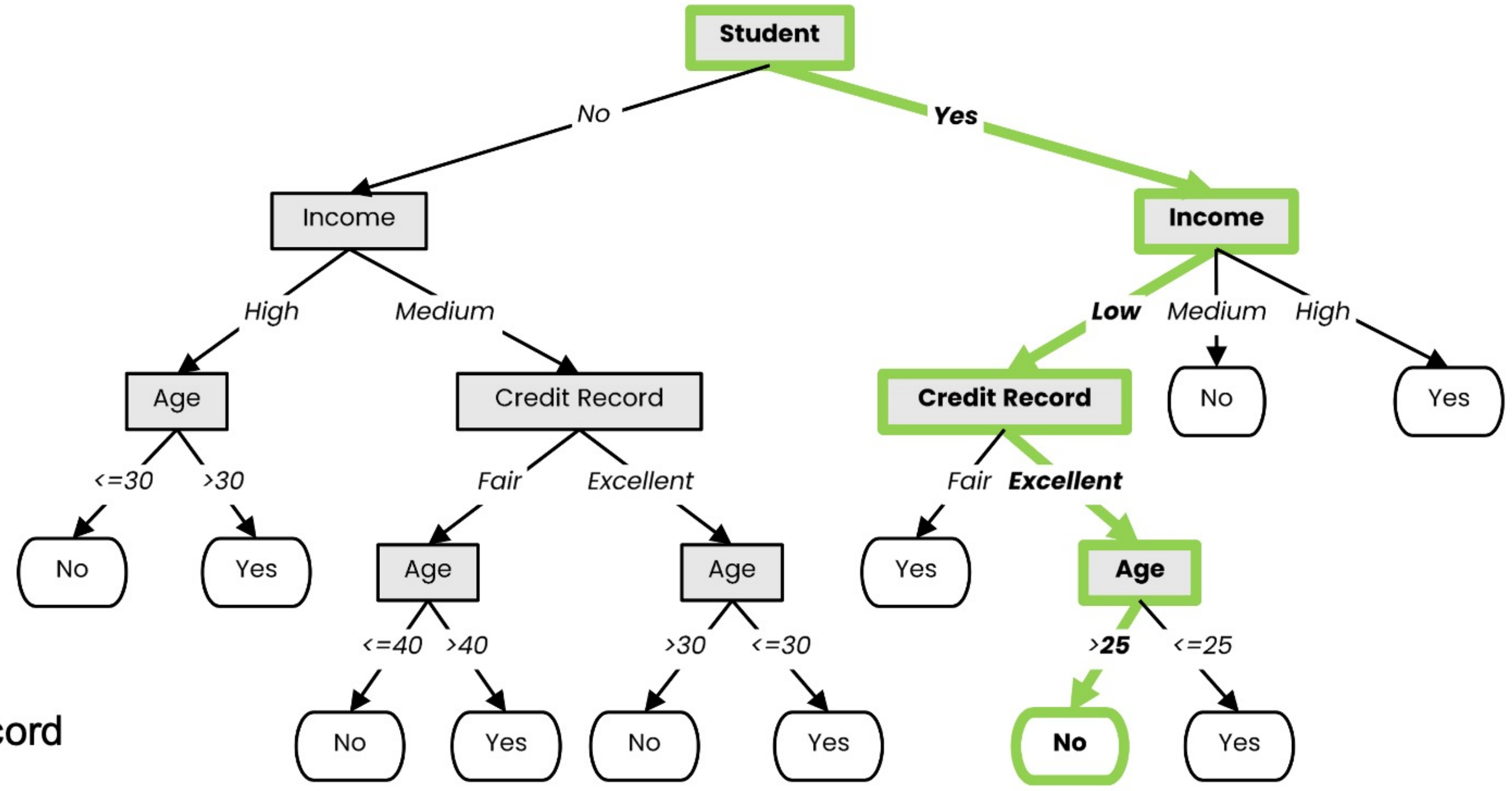
Decision Tree

- A tree-like model that illustrates series of events leading to certain decisions
- Each node represents a test on an attribute and each branch is an outcome of that test

Who to loan?

- Not a student
- 45 years old
- Medium income
- Fair credit record
- ➤ Yes

- Student
- 27 years old
- Low income
- Excellent credit record
- ➤ No



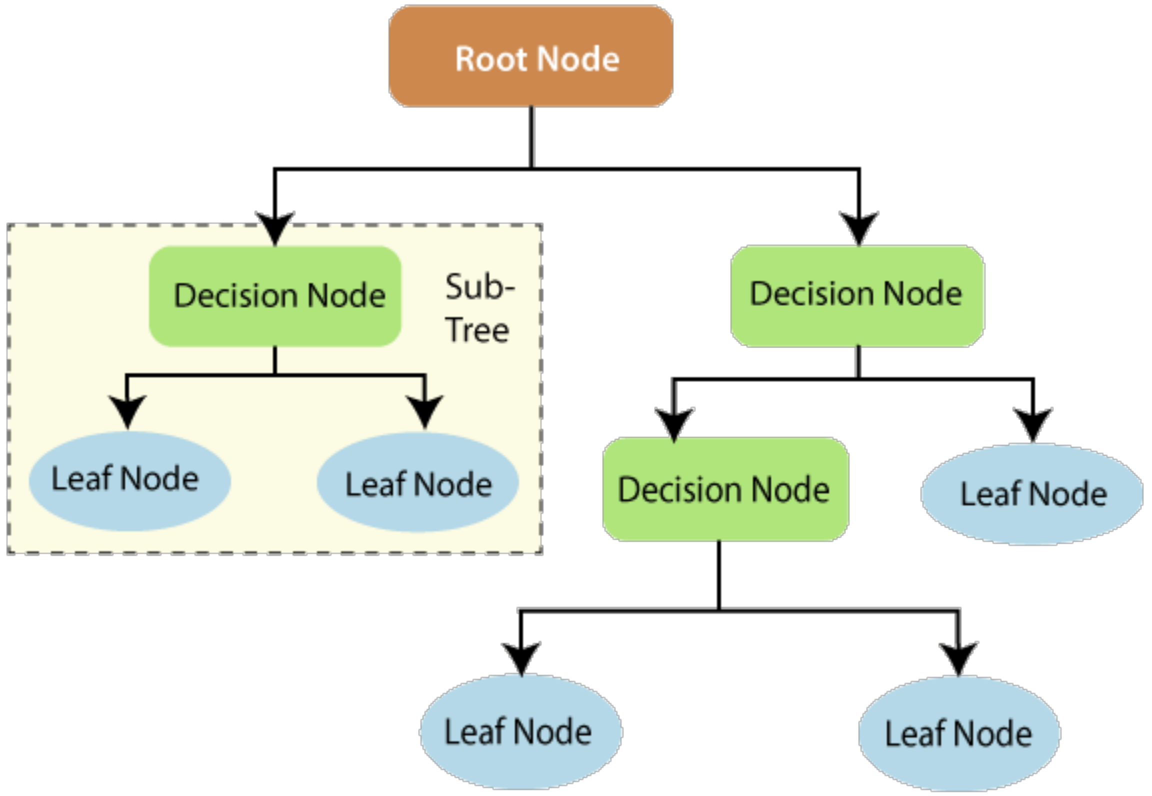
Decision Tree

What is a Decision Tree?

A decision tree is a supervised machine learning algorithm used for both classification and regression tasks. It works by partitioning the dataset into subsets based on the attribute values. This process is performed recursively, resulting in a tree-like model of decisions.

Definitions:

1. **Root Node**: It represents the entire dataset and further gets divided into two or more homogeneous sets.
2. **Decision Node**: When a sub-node splits into further sub-nodes, then it is called the decision node.
3. **Leaf/ Terminal Node**: Nodes that do not split are called Leaf or Terminal nodes.
4. **Parent and Child Node**: A node that gets divided into sub-nodes is called a parent node, whereas the sub-nodes are the child nodes of the parent node.



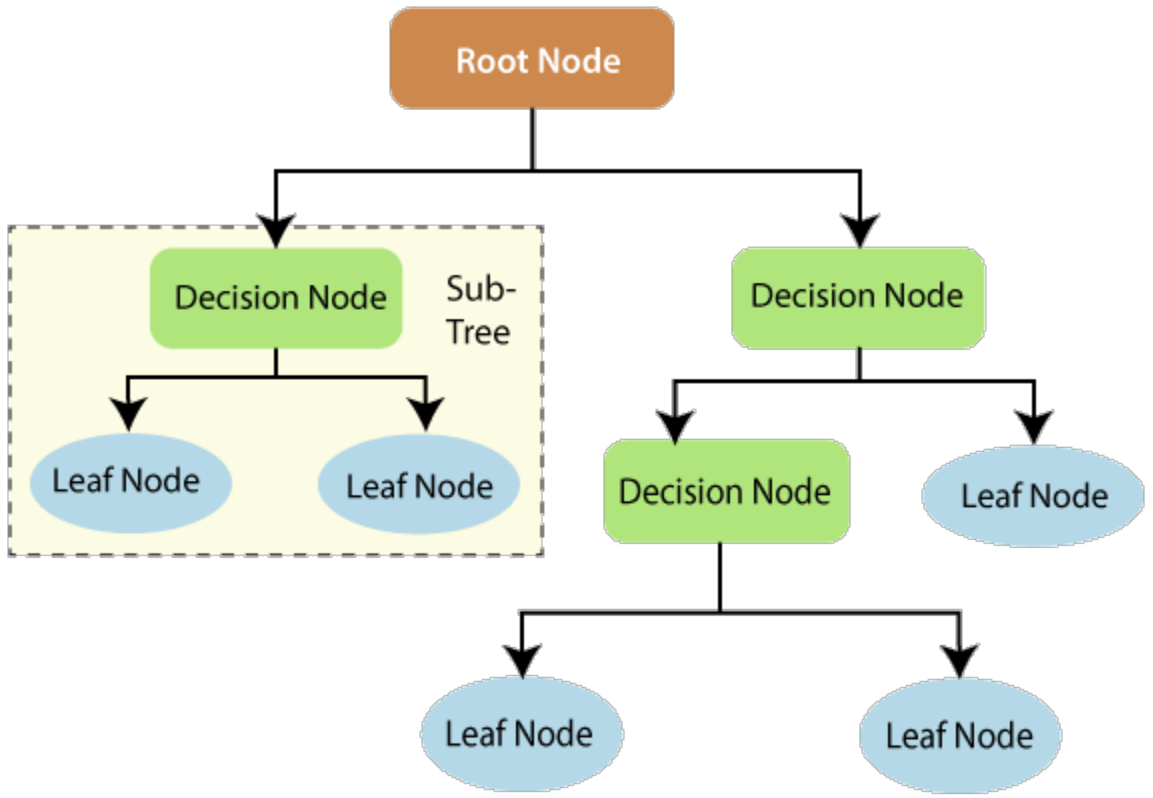
Decision Tree

What is a Decision Tree?

A decision tree is a supervised machine learning algorithm used for both classification and regression tasks. It works by partitioning the dataset into subsets based on the attribute values. This process is performed recursively, resulting in a tree-like model of decisions.

Definitions:

5. **Pruning**: Removing branches that have little importance.
6. **Branch / Sub-Tree:** A subsection of the decision tree.



Decision Tree

- We use labeled data to obtain a suitable decision tree for future predictions
  - We want a decision tree that works well on unseen data, while asking as few questions as possible

- Why we use decision trees?
  - Easy to understand and interpret.
  - Requries little data preparation.
  - Can handle both numerical and categorical data.
  - Performs well with large datasets.

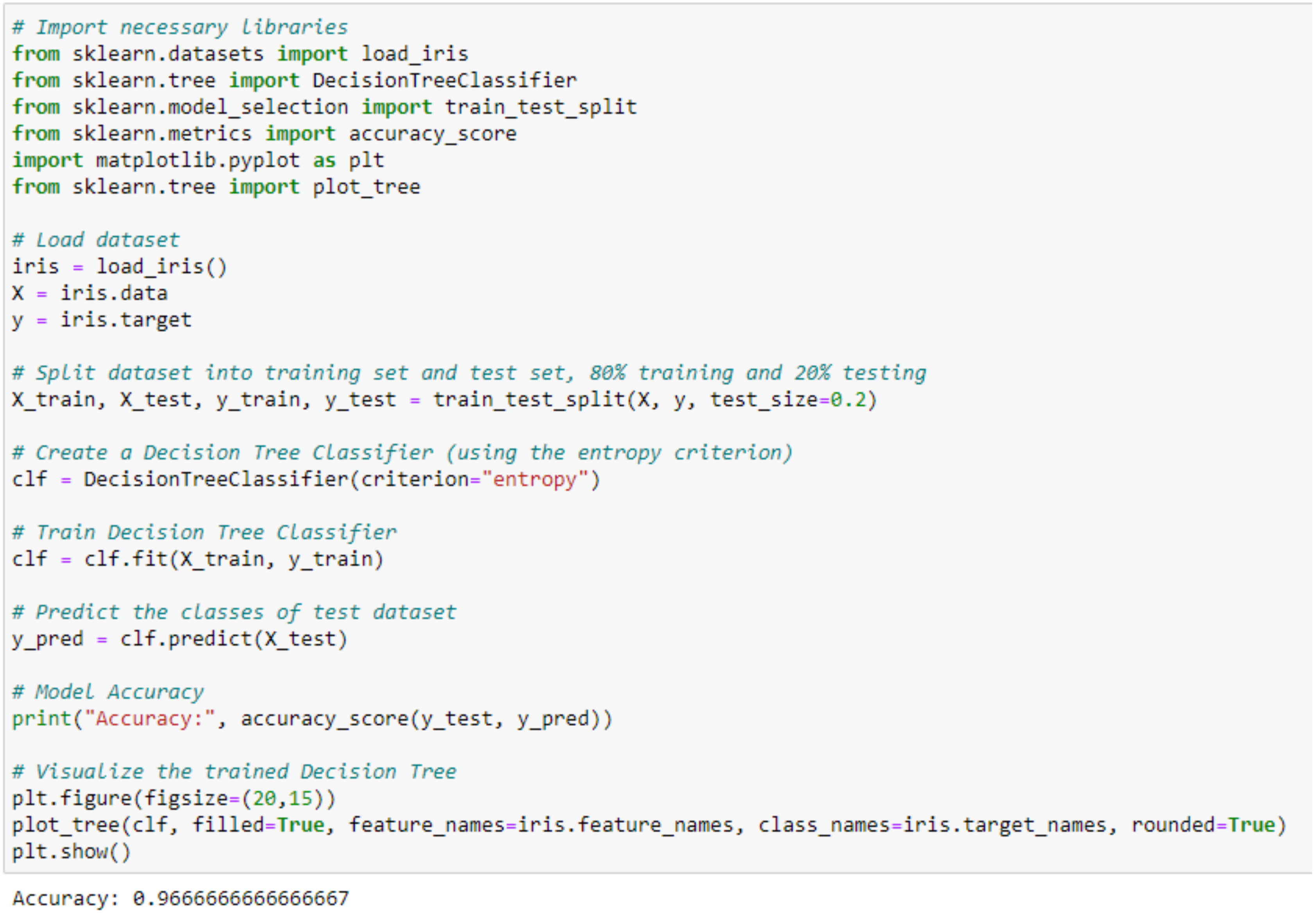
- Basic step:
  1. Choose an attribute from the dataset.
  2. Calculate the significance of the attribute in splitting the data.
  3. Split the data based on the value of the best attribute.
  4. Go to each branch and repeat until the data is segmented into homogeneous groups.

Decision Tree - python

Decision Tree Learning (Python)

```
def make_tree(X):
    node = TreeNode(X)
    if should_be_leaf_node(X):
        node.label = majority_label(X)
    else:
        a = select_best_splitting_attribute(X)
        for v in values(a):
            $X_v = \{x \in X | x[a] == v\}$
            node.children.append(make_tree($X_v$))
    return node
```

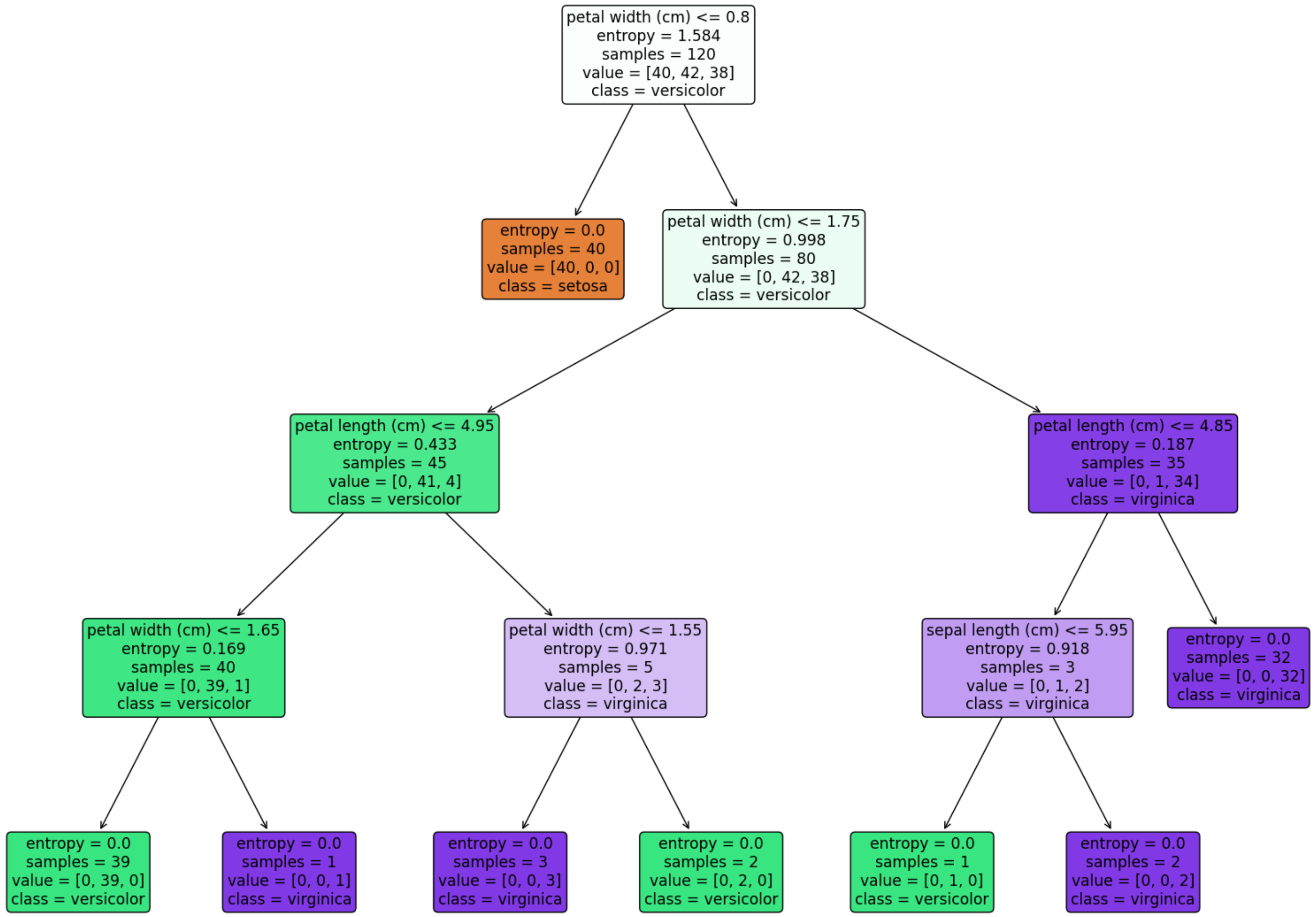
Decision Tree – scikit-learn



```python
# Import necessary libraries
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split dataset into training set and test set, 80% training and 20% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Create a Decision Tree Classifier (using the entropy criterion)
clf = DecisionTreeClassifier(criterion="entropy")

# Train Decision Tree Classifier
clf = clf.fit(X_train, y_train)

# Predict the classes of test dataset
y_pred = clf.predict(X_test)

# Model Accuracy
print("Accuracy:", accuracy_score(y_test, y_pred))

# Visualize the trained Decision Tree
plt.figure(figsize=(20,15))
plot_tree(clf, filled=True, feature_names=iris.feature_names, class_names=iris.target_names, rounded=True)
plt.show()
```

```
Accuracy: 0.9666666666666667
```

Decision Tree – scikit-learn



Decision Tree

Metrics for Splitting

1. Gini Impurity
   - Measures the disorder of a set.
   - A Gini Impurity of 0 indicates that the set is perfectly classified (i.e., all items are of a single class).
   - The maximum value of Gini Impurity (for a binary classification) is 0.5, indicating that the set is equally split between the two classes.

2. Information Gain
   - Based on the concept of entropy from information theory.
   - Measures the amount of information in one random variable gained from observing another random variable.
   - The attribute with the highest Information Gain is preferred for splitting.

3. Chi-Square
4. Gain Ratio

Decision Tree

Metrics for Splitting

1. Gini Impurity

- Measures the disorder of a set.
- A Gini Impurity of 0 indicates that the set is perfectly classified (i.e., all items are of a single class).
- The maximum value of Gini Impurity (for a binary classification) is 0.5, indicating that the set is equally split between the two classes.

Why is it Important in Decision Trees?

When constructing a decision tree, at each node, we evaluate possible splits of the data based on different feature thresholds. The Gini Impurity helps in determining the "best" split by quantifying how mixed the labels are in the two resulting subsets. The goal is to minimize the Gini Impurity, leading to purer nodes with more homogenous data.

Decision Tree

Metrics for Splitting - 1. Gini Impurity

Example:
Let's consider a dataset where you have the following colors of balls: Red, Red, Blue, Blue, and Blue.
The Gini Impurity for this set would be calculated as:

$$P(Red) = \frac{2}{5}$$
$$P(Blue) = \frac{3}{5}$$

$$Gini(D) = 1 - (\frac{2}{5})^2 - (\frac{3}{5})^2$$
$$Gini(D) = 1 - 0.16 - 0.36 = 0.48$$

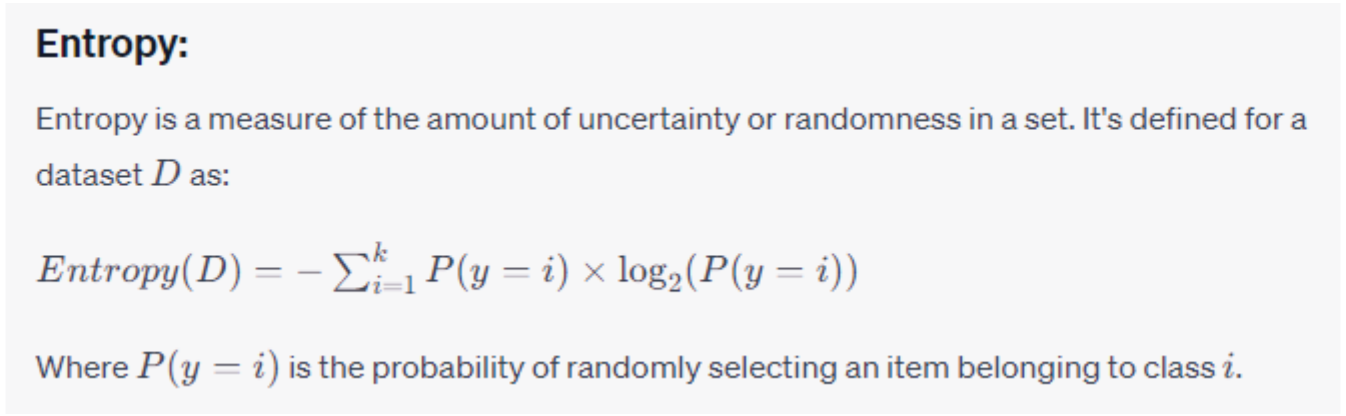
This value indicates some level of impurity in the set, which is expected since we have a mix of Red and Blue balls.

Decision Tree

Metrics for Splitting - 2. Information Gain

- Based on the concept of entropy from information theory.
- Measures the amount of information in one random variable gained from observing another random variable.
- The attribute with the highest Information Gain is preferred for splitting.

Entropy is a measure of uncertainty, randomness, or impurity in a set of data. The higher the entropy, the more random or disordered the data is; conversely, the lower the entropy, the more ordered the data is.



Entropy:

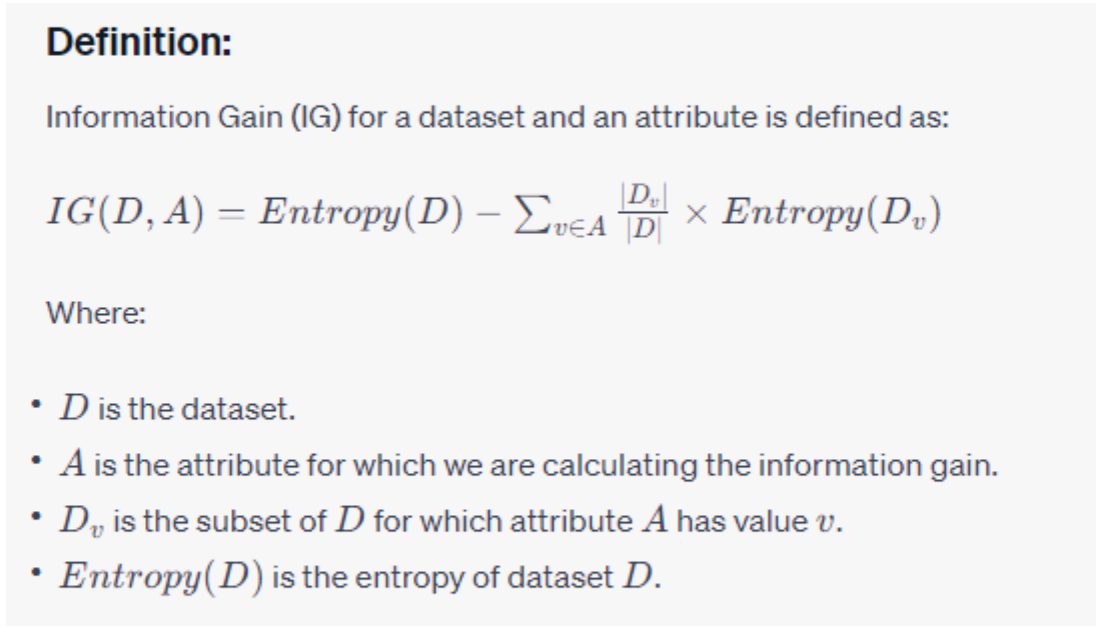
Entropy is a measure of the amount of uncertainty or randomness in a set. It's defined for a dataset D as:

$$Entropy(D) = -\sum_{i=1}^{k} P(y=i) \times \log_2(P(y=i))$$

Where $P(y=i)$ is the probability of randomly selecting an item belonging to class i.

Decision Tree

Metrics for Splitting - 2. Information Gain



Definition:

Information Gain (IG) for a dataset and an attribute is defined as:

$$IG(D, A) = Entropy(D) - \sum_{v \in A} \frac{|D_v|}{|D|} \times Entropy(D_v)$$

Where:

- D is the dataset.
- A is the attribute for which we are calculating the information gain.
- D_v is the subset of D for which attribute A has value v.
- $Entropy(D)$ is the entropy of dataset D.

Interpretation:

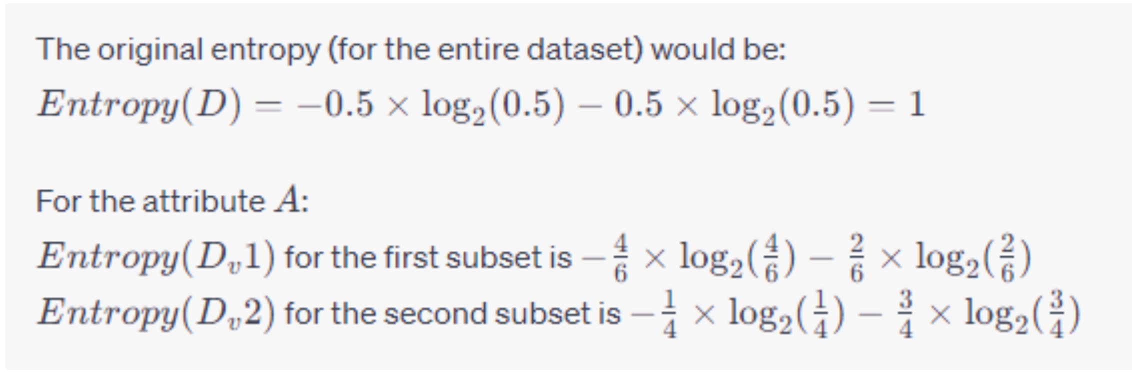
•If the Information Gain is high, it means the attribute does a good job in partitioning the data into classes.

•If the Information Gain is low, it means the attribute doesn't tell us much about the target class.

Decision Tree

Metrics for Splitting - 2. Information Gain

Example:
Consider a simple dataset with 10 instances, where 5 belong to class Positive (+) and 5 belong to class Negative (-). Let's consider a binary attribute *A* that splits the data into two subsets: 4+ and 2- in one subset, and 1+ and 3- in the other.



The original entropy (for the entire dataset) would be:

$Entropy(D) = -0.5 \times \log_2(0.5) - 0.5 \times \log_2(0.5) = 1$

For the attribute A:

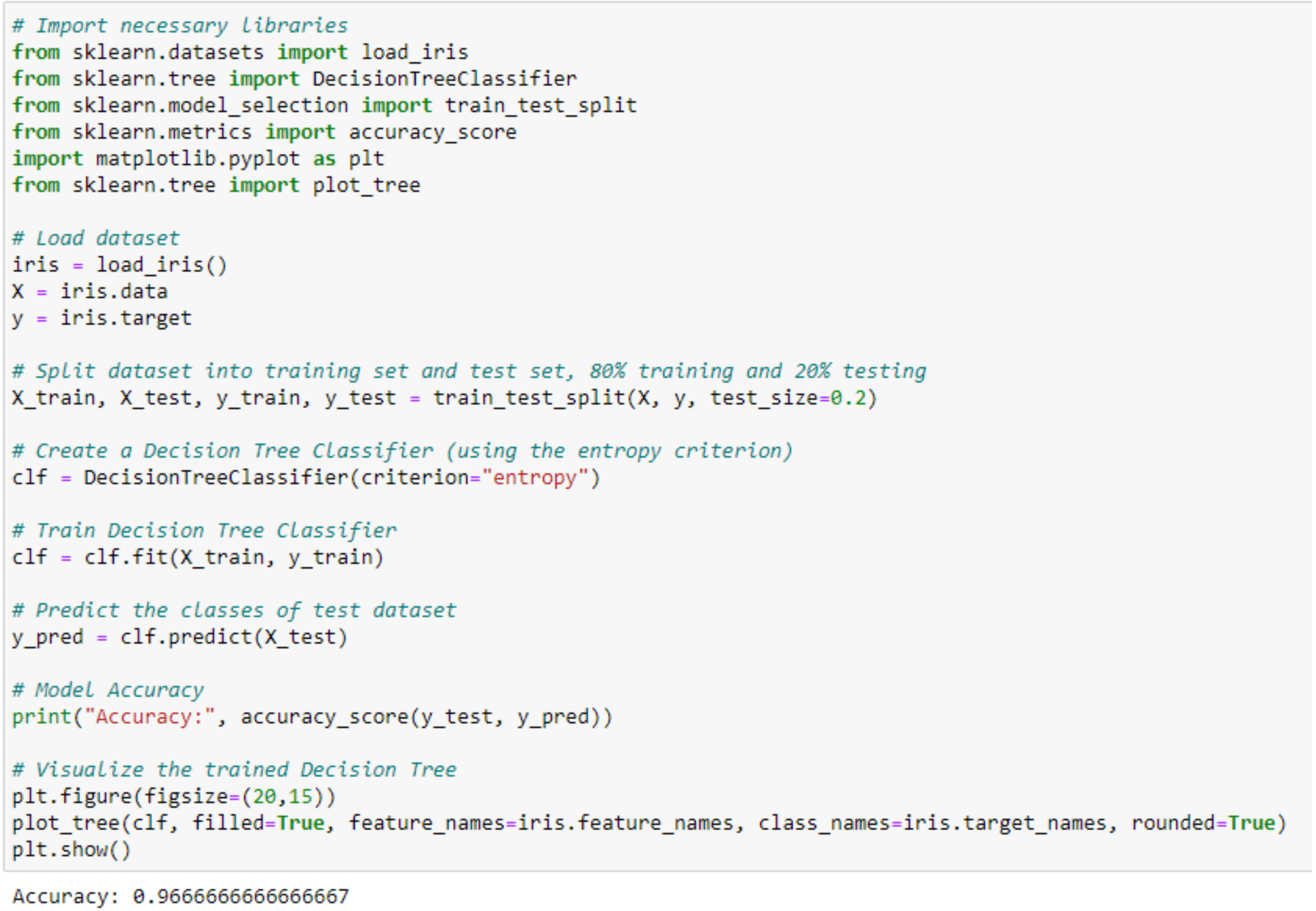
$Entropy(D_v1)$ for the first subset is $-\frac{4}{6} \times \log_2(\frac{4}{6}) - \frac{2}{6} \times \log_2(\frac{2}{6})$

$Entropy(D_v2)$ for the second subset is $-\frac{1}{4} \times \log_2(\frac{1}{4}) - \frac{3}{4} \times \log_2(\frac{3}{4})$

The weighted average of these entropies, subtracted from the original entropy, gives the Information Gain for attribute *A*.
 If this is the highest IG among all attributes, *A* would be chosen as the splitting attribute for the current node in the decision tree.

UNC GREENSBORO

Decision Tree – scikit-learn



```python
# Import necessary libraries
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split dataset into training set and test set, 80% training and 20% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Create a Decision Tree Classifier (using the entropy criterion)
clf = DecisionTreeClassifier(criterion="entropy")

# Train Decision Tree Classifier
clf = clf.fit(X_train, y_train)

# Predict the classes of test dataset
y_pred = clf.predict(X_test)

# Model Accuracy
print("Accuracy:", accuracy_score(y_test, y_pred))

# Visualize the trained Decision Tree
plt.figure(figsize=(20,15))
plot_tree(clf, filled=True, feature_names=iris.feature_names, class_names=iris.target_names, rounded=True)
plt.show()
```

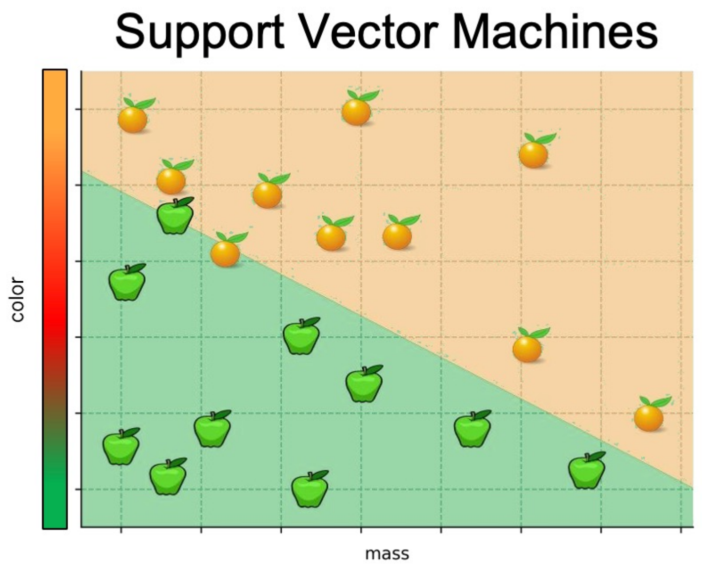
```
Accuracy: 0.9666666666666667
```

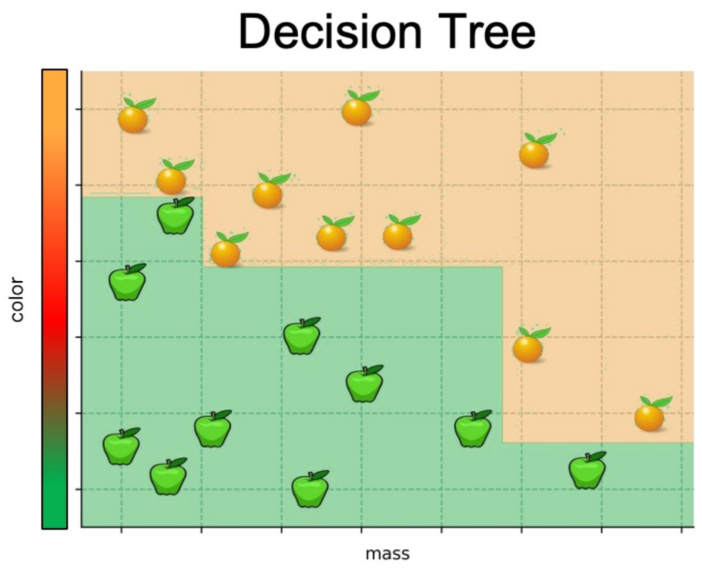
UNC GREENSBORO

Decision Tree

Decision Boundaries

- Decision trees produce non-linear decision boundaries





Decision Tree

Overfitting in Decision Trees

- A common problem in decision trees.
- Occurs when the tree is too complex and captures noise.
- Solution:
  - Stop growing when data split is not statistically significant
  - Get more data
  - Remove non-relevant attributes – Feature Tuning
  - Pruning – Grow full tree and then remove branches with low Information Gain

Decision Tree

Advantages:

- Simple to understand.

- Requires little data prep.

- Fast.

Disadvantages:

- Prone to overfitting.

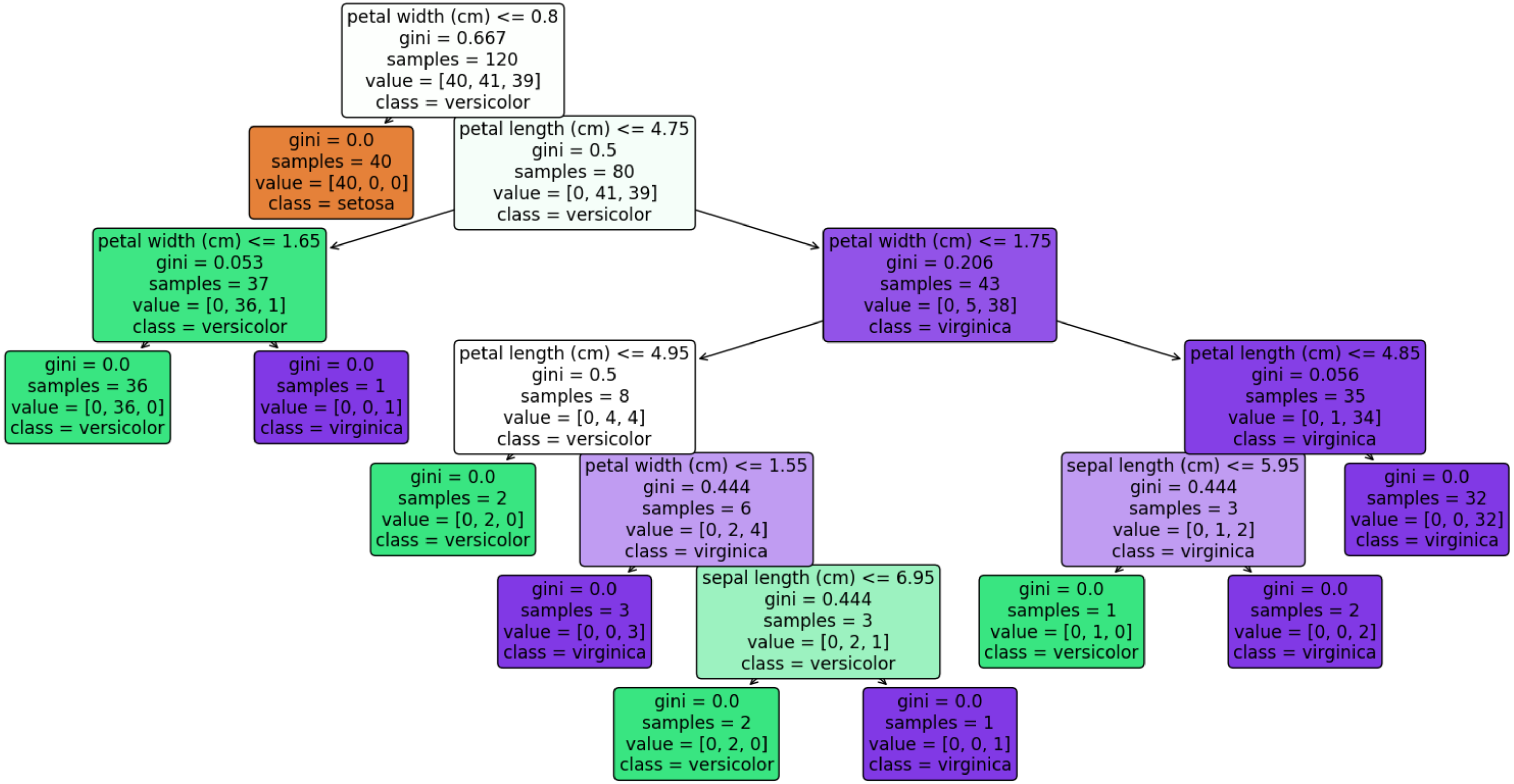
- Biased to dominant classes.

- Can be unstable (small changes in data can lead to a different tree).

Exercise

Objective: Predict the species of iris flowers based on the length and width of their sepals and petals.

1. Data Loading and Exploration (Hint: df['species'] = iris.target)
2. Split Data into Training and Testing Sets
3. Build and Train a Decision Tree Model
4. Evaluate the Model (Hint: from sklearn.metrics import accuracy_score)
5. Visualize the Decision Tree (Hint: plt.figure(); plot_tree())


petal width (cm) <= 0.8
gini = 0.667
samples = 120
value = [40, 41, 39]
class = versicolor
gini = 0.0
samples = 40
value = [40, 0, 0]
class = setosa
petal length (cm) <= 4.75
gini = 0.5
samples = 80
value = [0, 41, 39]
class = versicolor
petal width (cm) <= 1.65
gini = 0.053
samples = 37
value = [0, 36, 1]
class = versicolor
petal width (cm) <= 1.75
gini = 0.206
samples = 43
value = [0, 5, 38]
class = virginica
gini = 0.0
samples = 36
value = [0, 36, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
petal length (cm) <= 4.95
gini = 0.5
samples = 8
value = [0, 4, 4]
class = versicolor
petal length (cm) <= 4.85
gini = 0.056
samples = 35
value = [0, 1, 34]
class = virginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
petal width (cm) <= 1.55
gini = 0.444
samples = 6
value = [0, 2, 4]
class = virginica
sepal length (cm) <= 5.95
gini = 0.444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 32
value = [0, 0, 32]
class = virginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = virginica
sepal length (cm) <= 6.95
gini = 0.444
samples = 3
value = [0, 2, 1]
class = versicolor
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica

GREENSBORO