LAB No 3

Decision Tree Classifier

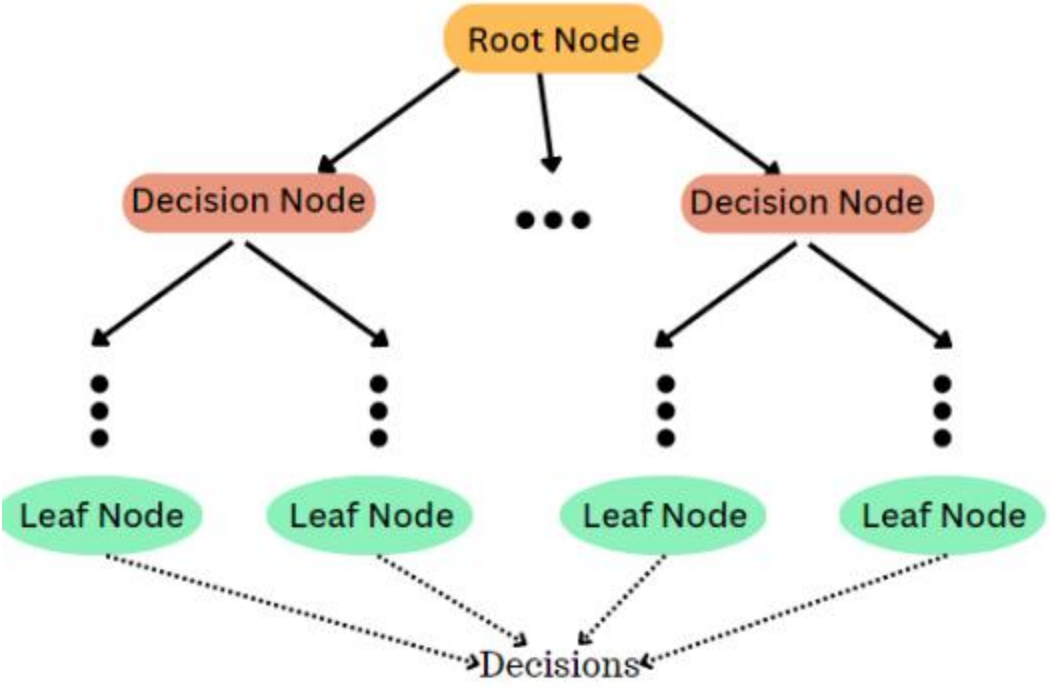
In this lab, students will study and implement the Decision Tree Classifier, a supervised machine learning algorithm used for classification tasks. The lab begins with a manual calculation of entropy and information gain to understand how decision trees choose splitting attributes. Students will then implement a decision tree on a small categorical dataset, followed by applying the algorithm to a real-world dataset (Iris) using Python and Scikit-learn. Through this lab, students will gain both theoretical clarity and practical experience in building, training, and visualizing decision tree models.

Introduction & Theory

A **Decision Tree Classifier** is a tree-structured model used to make decisions based on feature values. Each internal node represents a test on an attribute, each branch represents an outcome, and each leaf node represents a class label.

The construction of a decision tree is based on:

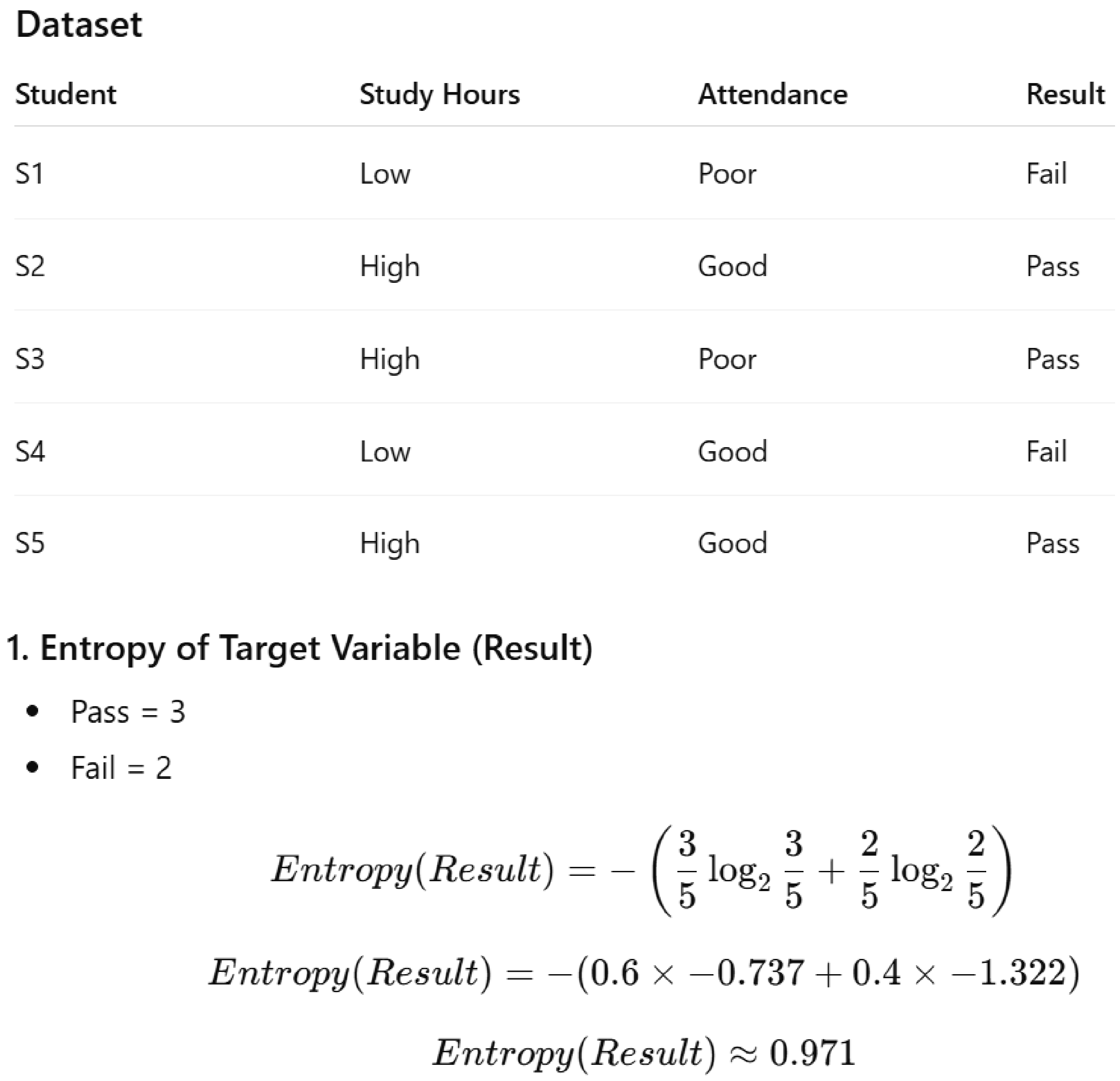
- **Entropy**: A measure of uncertainty or impurity in a dataset
- **Information Gain**: Reduction in entropy after splitting on an attribute



Decision trees are easy to interpret and visualize but may suffer from **overfitting**, especially on large or complex datasets.

Solved Examples:

Example 1: Entropy and Information Gain



Dataset

Student	Study Hours	Attendance	Result
S1	Low	Poor	Fail
S2	High	Good	Pass
S3	High	Poor	Pass
S4	Low	Good	Fail
S5	High	Good	Pass

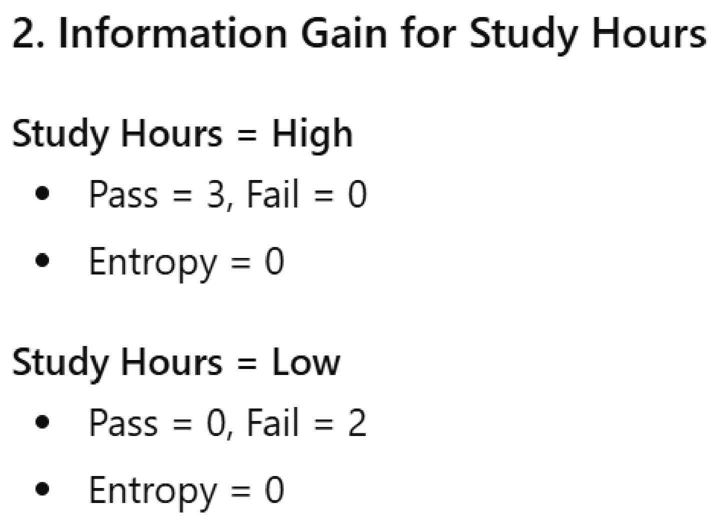
1. Entropy of Target Variable (Result)

- Pass = 3
- Fail = 2

$$Entropy(Result) = -\left(\frac{3}{5}\log_2\frac{3}{5} + \frac{2}{5}\log_2\frac{2}{5}\right)$$

$$Entropy(Result) = -(0.6 \times -0.737 + 0.4 \times -1.322)$$

$$Entropy(Result) \approx 0.971$$



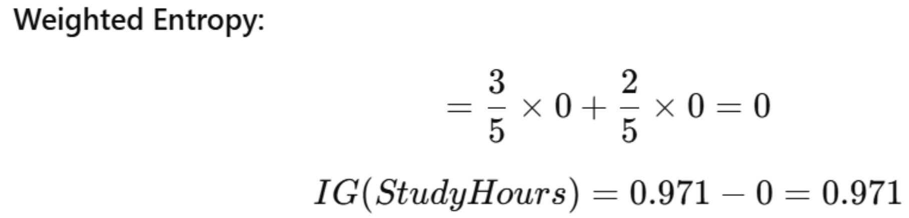
2. Information Gain for Study Hours

Study Hours = High

- Pass = 3, Fail = 0
- Entropy = 0

Study Hours = Low

- Pass = 0, Fail = 2
- Entropy = 0



Weighted Entropy:

$$= \frac{3}{5} \times 0 + \frac{2}{5} \times 0 = 0$$

$$IG(StudyHours) = 0.971 - 0 = 0.971$$

3. Root Node Selection

Since **Study Hours** provides the **maximum information gain**, it should be selected as the **root node**.

Example 2: Decision Tree on Small Dataset (Python Implementation)

Question

Build and visualize a decision tree using entropy.

Solution:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

# Create dataset
data = {
    'StudyHours': ['Low', 'High', 'High', 'Low', 'High'],
    'Attendance': ['Poor', 'Good', 'Poor', 'Good', 'Good'],
    'Result': ['Fail', 'Pass', 'Pass', 'Fail', 'Pass']
}

df = pd.DataFrame(data)

# Encode categorical values
encoder = LabelEncoder()
for col in df.columns:
    df[col] = encoder.fit_transform(df[col])

# Features and target
X = df[['StudyHours', 'Attendance']]
y = df['Result']

# Train decision tree
model = DecisionTreeClassifier(criterion='entropy')
model.fit(X, y)

# Visualize tree
plt.figure(figsize=(10,6))
plot_tree(model, feature_names=X.columns, class_names=['Fail', 'Pass'], filled=True)
plt.show()
```

Example 3: Decision Tree Classifier on Iris Dataset

Objective

Apply decision trees to a real dataset and evaluate accuracy.

Solution

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

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

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Train model
model = DecisionTreeClassifier(criterion='entropy')
model.fit(X_train, y_train)


# Predict and evaluate
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)

# Visualize tree
plt.figure(figsize=(14,8))
plot_tree(model, feature_names=iris.feature_names,
          class_names=iris.target_names, filled=True)
plt.show()
```

LAB Assignment No. 3

Question 1

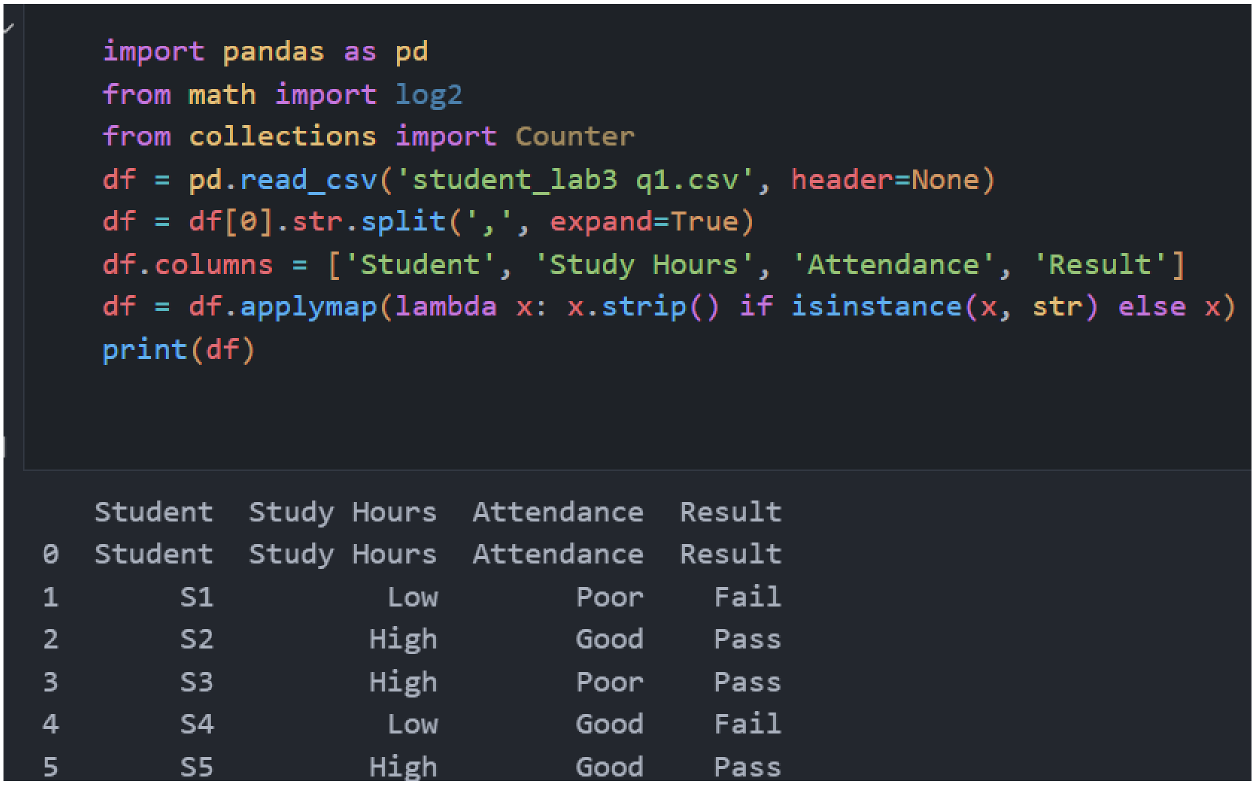
Entropy and Information Gain (Manual Calculation)

Given the dataset below about whether students pass an exam based on study time and attendance:

Student	**Study Hours**	**Attendance**	**Result**
S1	Low	Poor	Fail
S2	High	Good	Pass
S3	High	Poor	Pass
S4	Low	Good	Fail

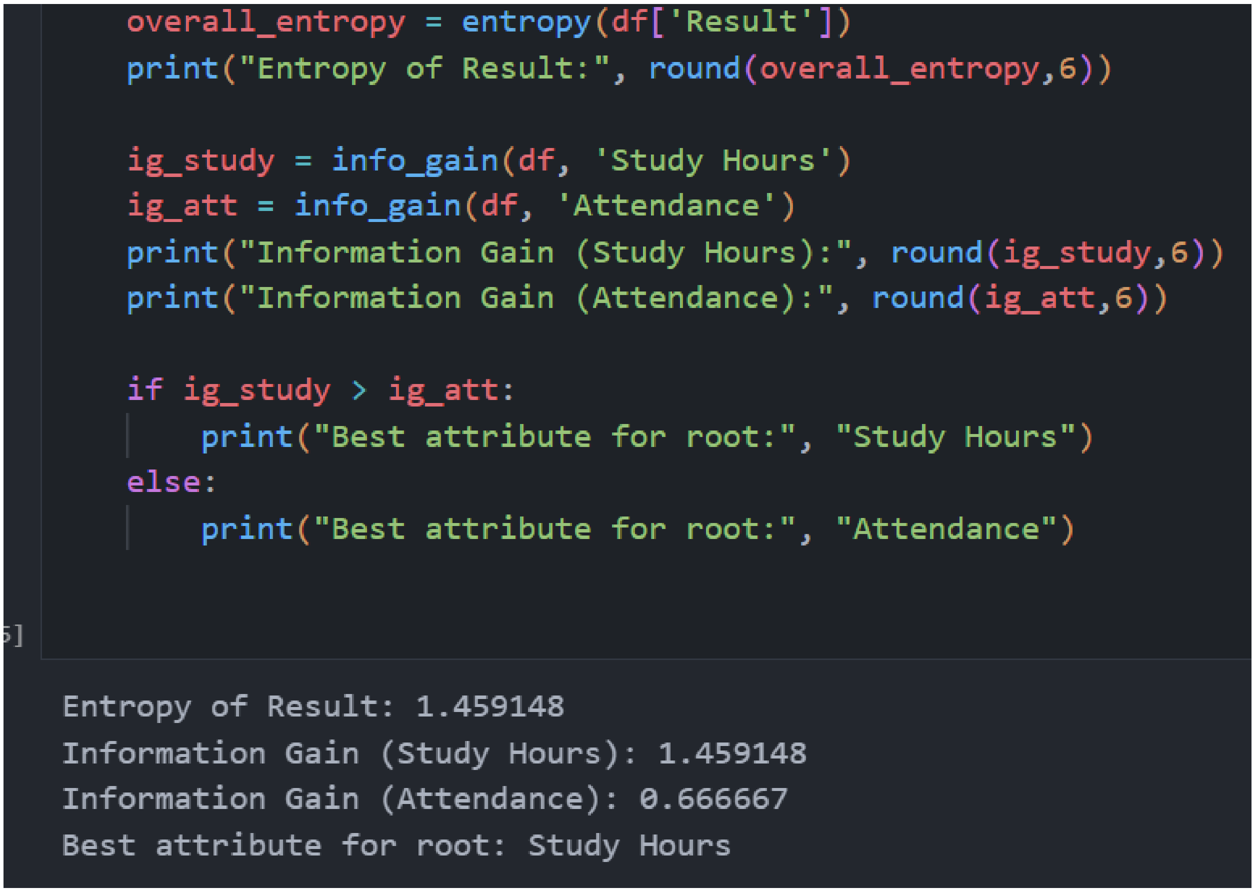
Student	Study Hours	Attendance	Result
S5	High	Good	Pass

1. Calculate the **entropy** of the target variable (Result).
2. Compute the **information gain** for the attribute Study Hours.
3. Which attribute should be selected for the root node based on maximum information gain?



```
import pandas as pd
from math import log2
from collections import Counter
df = pd.read_csv('student_lab3 q1.csv', header=None)
df = df[0].str.split(',', expand=True)
df.columns = ['Student', 'Study Hours', 'Attendance', 'Result']
df = df.applymap(lambda x: x.strip() if isinstance(x, str) else x)
print(df)
```

```
   Student  Study Hours  Attendance  Result
0  Student  Study Hours  Attendance  Result
1       S1          Low        Poor    Fail
2       S2         High        Good    Pass
3       S3         High        Poor    Pass
4       S4          Low        Good    Fail
5       S5         High        Good    Pass
```



```
overall_entropy = entropy(df['Result'])
print("Entropy of Result:", round(overall_entropy,6))

ig_study = info_gain(df, 'Study Hours')
ig_att = info_gain(df, 'Attendance')
print("Information Gain (Study Hours):", round(ig_study,6))
print("Information Gain (Attendance):", round(ig_att,6))

if ig_study > ig_att:
    print("Best attribute for root:", "Study Hours")
else:
    print("Best attribute for root:", "Attendance")
```

```
Entropy of Result: 1.459148
Information Gain (Study Hours): 1.459148
Information Gain (Attendance): 0.666667
Best attribute for root: Study Hours
```

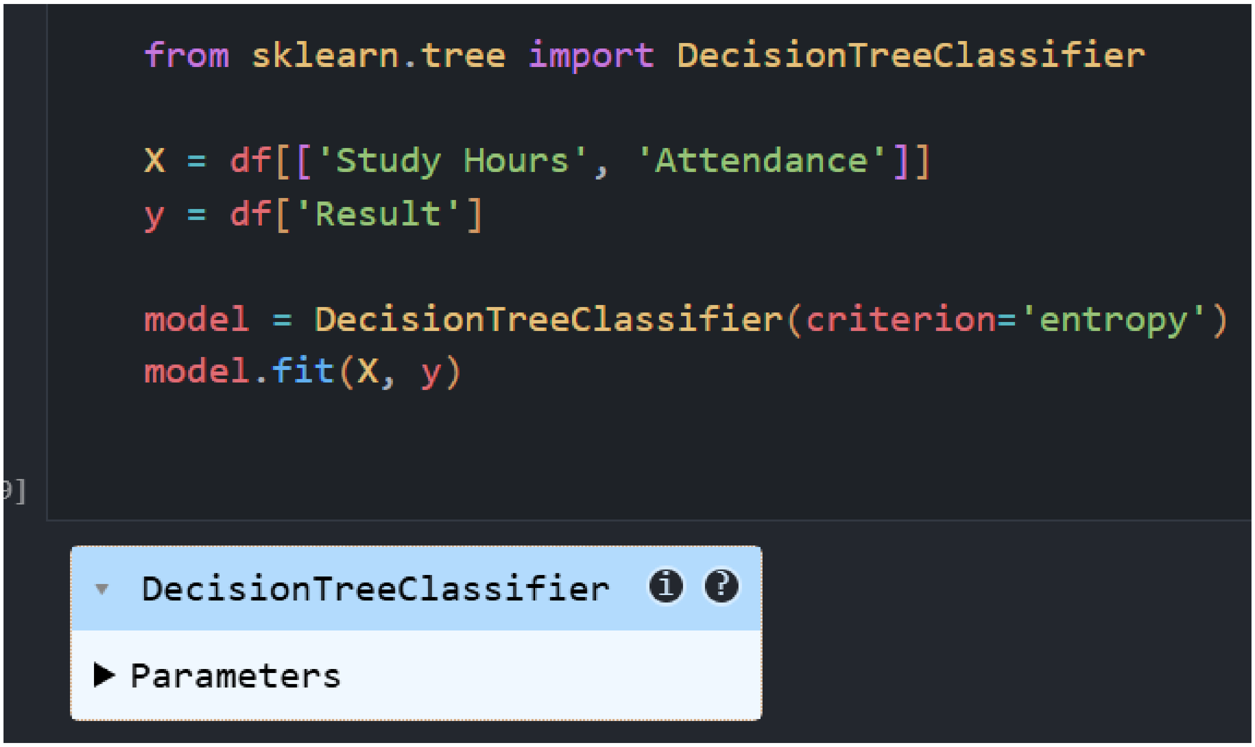
Question No. 2

Implement Decision Tree Classifier on a Small Dataset

Build and visualize a simple decision tree.

Question:
Using the same dataset as above:

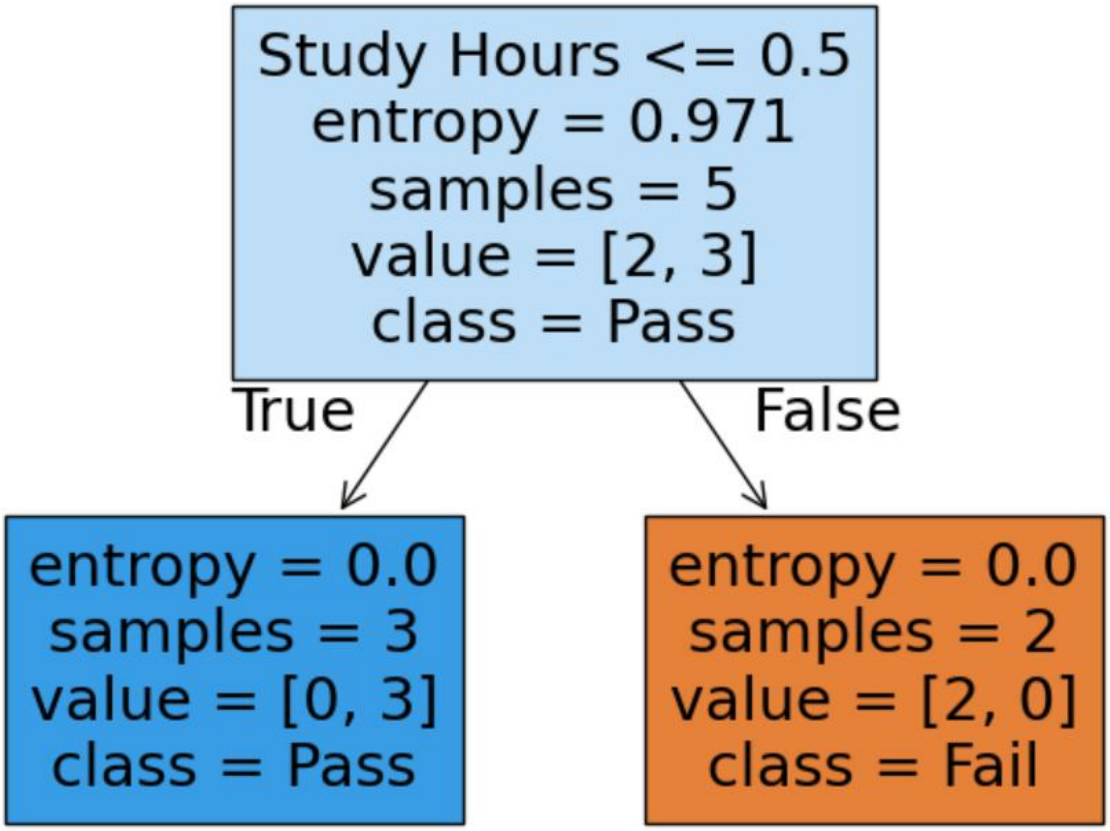
1. Use pandas to create a DataFrame.
2. Convert categorical values into numerical using LabelEncoder.
3. Train a **DecisionTreeClassifier** using **criterion='entropy'**.
4. Visualize the decision tree using plot_tree() from sklearn.tree.



```
from sklearn.tree import DecisionTreeClassifier

X = df[['Study Hours', 'Attendance']]
y = df['Result']

model = DecisionTreeClassifier(criterion='entropy')
model.fit(X, y)
```



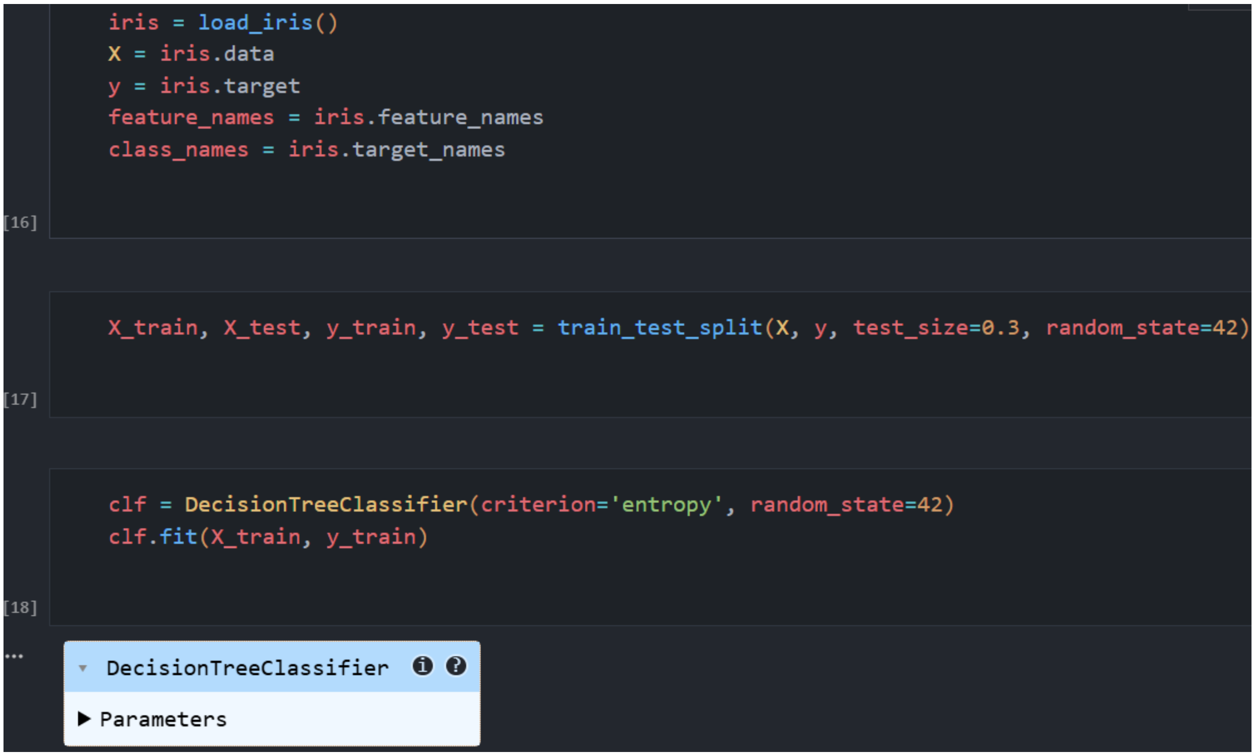
Question 3

Decision Tree Classifier on Iris Dataset

Objective: Apply decision trees to a real dataset.

Question:

1. Load the **Iris dataset** using sklearn.datasets.load_iris.
2. Split it into training (70%) and testing (30%) sets.
3. Train a decision tree using **criterion='entropy'**.
4. Print the accuracy on the test set.
5. Visualize the tree and explain which feature provides the most information gain at the root.



```
iris = load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names
class_names = iris.target_names
```

[16]

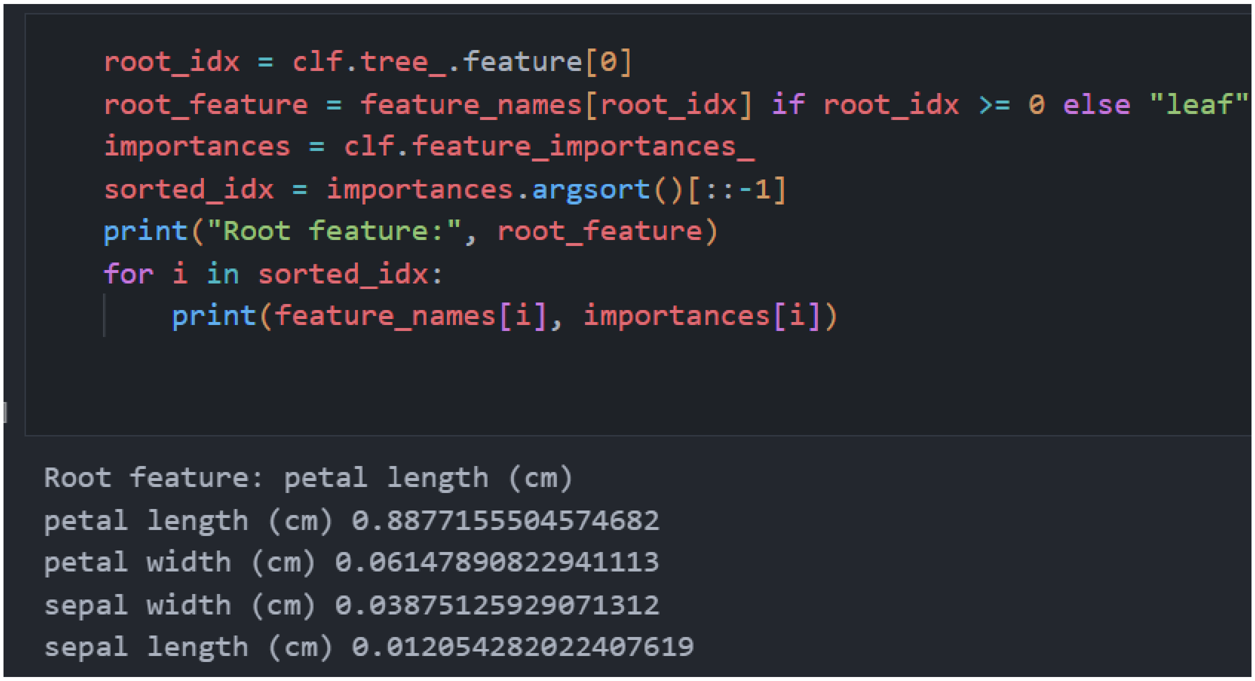
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

[17]

```
clf = DecisionTreeClassifier(criterion='entropy', random_state=42)
clf.fit(X_train, y_train)
```

[18]

```
...
▾ DecisionTreeClassifier
▶ Parameters
```



```
root_idx = clf.tree_.feature[0]
root_feature = feature_names[root_idx] if root_idx >= 0 else "leaf"
importances = clf.feature_importances_
sorted_idx = importances.argsort()[::-1]
print("Root feature:", root_feature)
for i in sorted_idx:
    print(feature_names[i], importances[i])
```

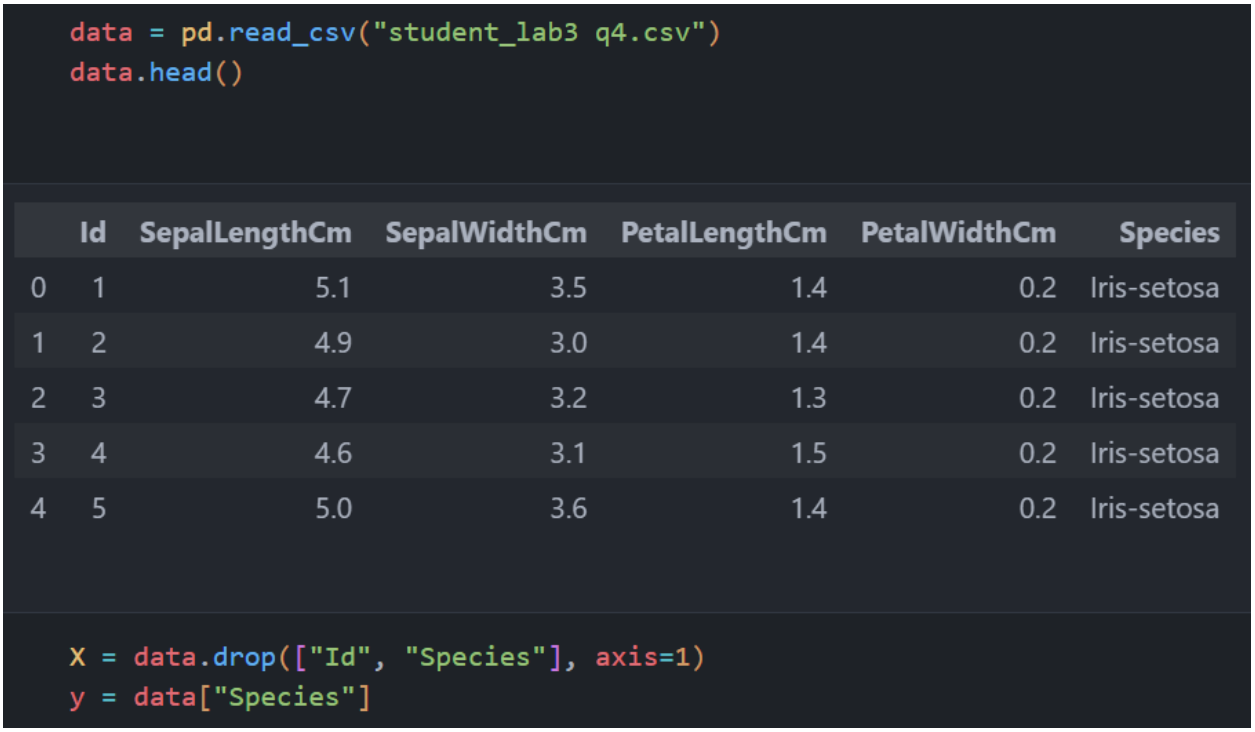
```
Root feature: petal length (cm)
petal length (cm) 0.8877155504574682
petal width (cm) 0.06147890822941113
sepal width (cm) 0.03875125929071312
sepal length (cm) 0.012054282022407619
```

Question 4

MNIST digit dataset (available via Keras / sklearn.datasets) as a baseline

Objectives

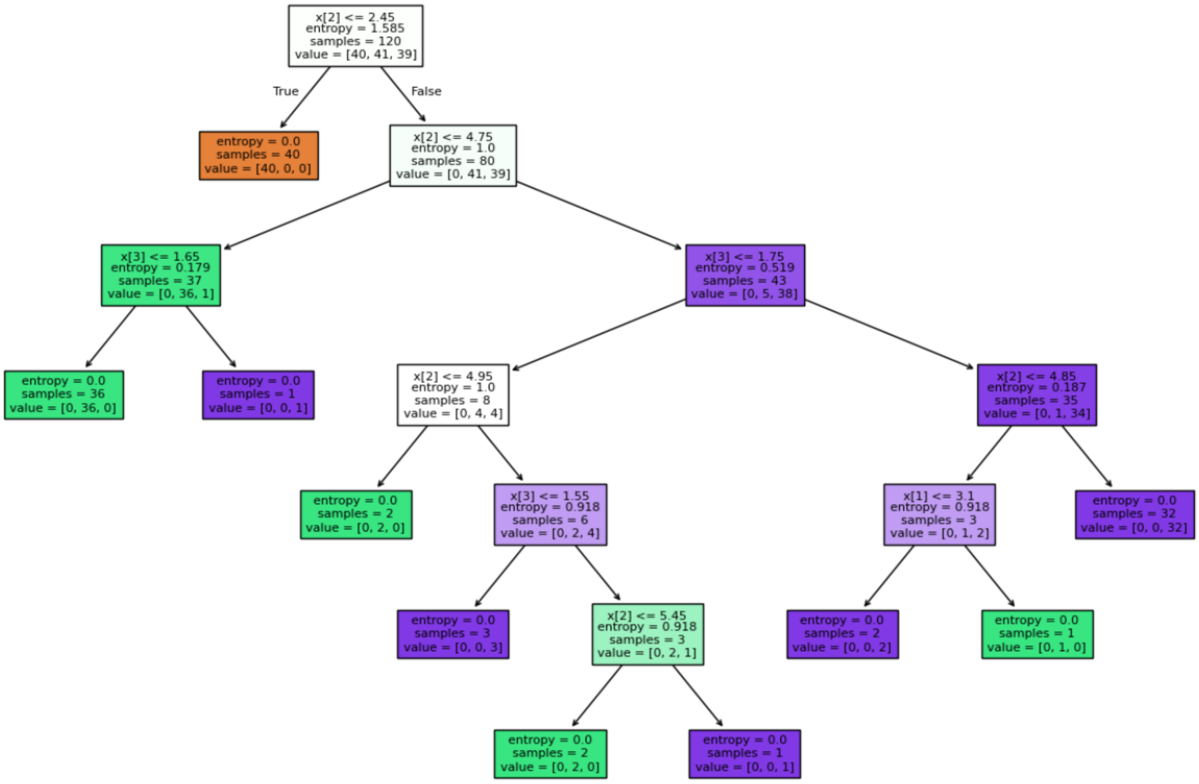
- Preprocess image data for classification
- Train a **Decision Tree Classifier** (or variants)
- Evaluate accuracy, confusion matrix, and discuss limitations



```
data = pd.read_csv("student_lab3 q4.csv")
data.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
X = data.drop(["Id", "Species"], axis=1)
y = data["Species"]
```


x[2] <= 2.45
entropy = 1.585
samples = 120
value = [40, 41, 39]
True
False
entropy = 0.0
samples = 40
value = [40, 0, 0]
x[2] <= 4.75
entropy = 1.0
samples = 80
value = [0, 41, 39]
x[3] <= 1.65
entropy = 0.179
samples = 37
value = [0, 36, 1]
x[3] <= 1.75
entropy = 0.519
samples = 43
value = [0, 5, 38]
entropy = 0.0
samples = 36
value = [0, 36, 0]
entropy = 0.0
samples = 1
value = [0, 0, 1]
x[2] <= 4.95
entropy = 1.0
samples = 8
value = [0, 4, 4]
x[2] <= 4.85
entropy = 0.187
samples = 35
value = [0, 1, 34]
entropy = 0.0
samples = 2
value = [0, 2, 0]
x[3] <= 1.55
entropy = 0.918
samples = 6
value = [0, 2, 4]
x[1] <= 3.1
entropy = 0.918
samples = 3
value = [0, 1, 2]
entropy = 0.0
samples = 32
value = [0, 0, 32]
entropy = 0.0
samples = 3
value = [0, 0, 3]
x[2] <= 5.45
entropy = 0.918
samples = 3
value = [0, 2, 1]
entropy = 0.0
samples = 2
value = [0, 0, 2]
entropy = 0.0
samples = 1
value = [0, 1, 0]
entropy = 0.0
samples = 2
value = [0, 2, 0]
entropy = 0.0
samples = 1
value = [0, 0, 1]