LAB Assignment No 3

Topic: Decision Tree Classifier

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
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?

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.

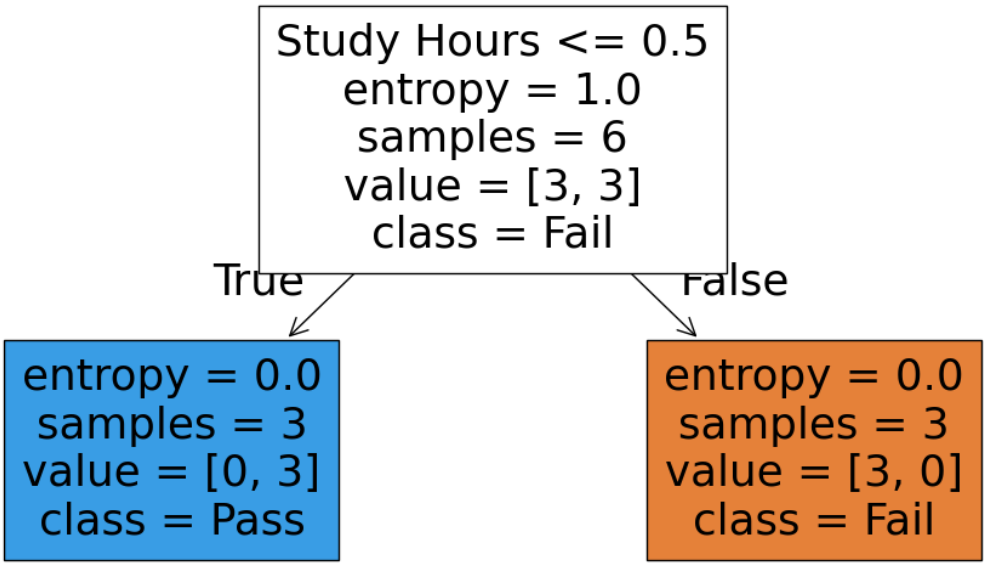
5. Predict whether a student with Study Hours=Low and Attendance=Good will pass or fail.

Code:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
data = {
   'Study Hours': ['Low', 'Low', 'High', 'High', 'Low', 'High'],
   'Attendance': ['Good', 'Poor', 'Good', 'Poor', 'Good', 'Poor'],
   'Result': ['Fail', 'Fail', 'Pass', 'Pass', 'Fail', 'Pass']
}
df = pd.DataFrame(data)
le = LabelEncoder()
df['Study Hours'] = le.fit_transform(df['Study Hours'])
df['Attendance'] = le.fit_transform(df['Attendance'])
df['Result'] = le.fit_transform(df['Result'])

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

model = DecisionTreeClassifier(criterion='entropy')
model.fit(X, y)
plt.figure(figsize=(10, 6))
plot_tree(model, feature_names=['Study Hours', 'Attendance'], class_names=['Fail',
'Pass'], filled=True)
plt.show()

prediction = model.predict([[0, 0]])
result = 'Pass' if prediction[0] == 1 else 'Fail'
print("Prediction for Study Hours=Low and Attendance=Good:", result)
```

output:

Prediction for Study Hours=Low and Attendance=Good: Pass



Question 3

Decision Tree Classifier on Iris Dataset

Objective: Apply decision trees to a real dataset.

Question:

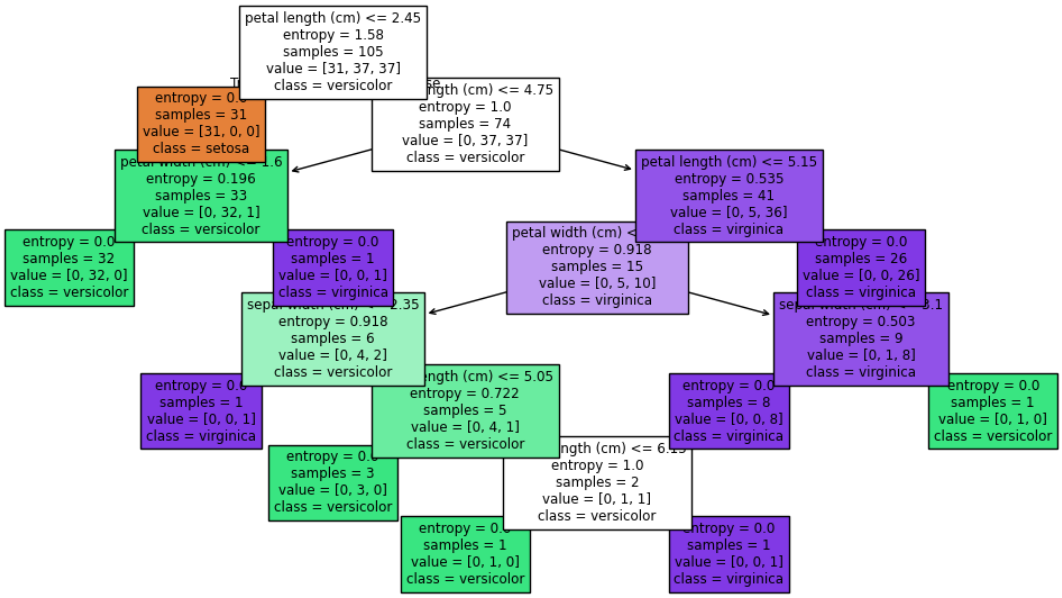
1. Load the **Iris dataset** using sklearn.datasets.load_iris.
2. Split it into training (70%) and testing (30%) sets.
3. Train a decision tree using **criterion='entropy'**.
4. Print the accuracy on the test set.
5. Visualize the tree and explain which feature provides the most information gain at the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

Code:

```
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
iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
model = DecisionTreeClassifier(criterion='entropy', random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("✅ Accuracy on Test Set:", accuracy_score(y_test, y_pred))
plot_tree(
   model,
   feature_names=iris.feature_names,
   class_names=iris.target_names,
   filled=True)
plt.show()
sample = [[5.1, 3.5, 1.4, 0.2]]  # example flower measurements
predicted_class = model.predict(sample)
print(f"Prediction for sample {sample}: {iris.target_names[predicted_class][0]}")
```

Output:

✅ Accuracy on Test Set: 0.9777777777777777
Prediction for sample [[5.1, 3.5, 1.4, 0.2]]: setosa



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

Code:

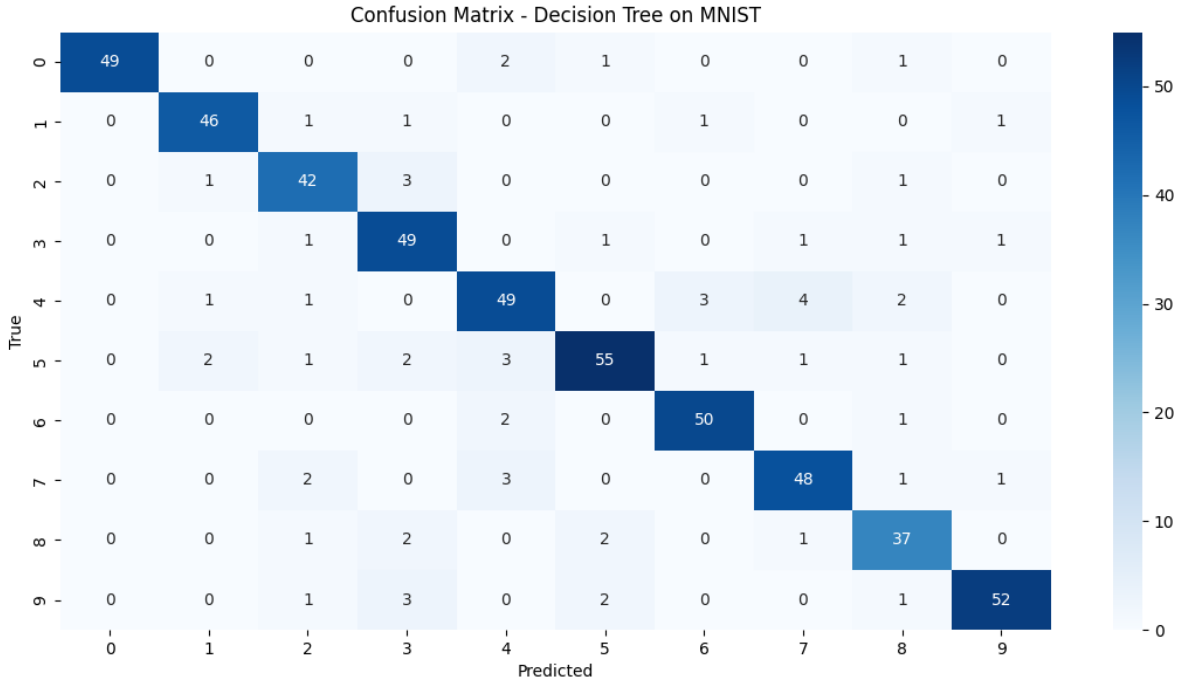
```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt
digits = load_digits()
X = digits.data    # Flattened 8x8 images (64 features per sample)
y = digits.target  # Labels 0–9
X_train, X_test, y_train, y_test = train_test_split(  X, y, test_size=0.3, random_state=42)
model = DecisionTreeClassifier(criterion='entropy', random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print("✅ Accuracy on Test Set:", acc)
print("\n📊 Classification Report:\n", classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix - Decision Tree on MNIST")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
plt.figure(figsize=(10,2))
for index, (image, label) in enumerate(zip(digits.images[:5], digits.target[:5])):
   plt.subplot(1, 5, index + 1)
   plt.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
   plt.title(f'True: {label}')
  plt.show()
```

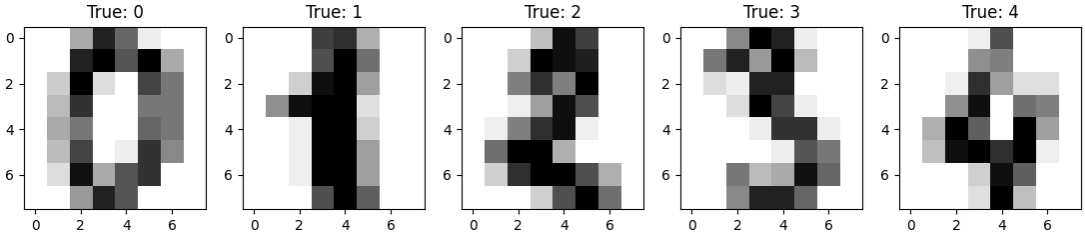
Output:

✅ **Accuracy on Test Set: 0.8833333333333333**

📊 **Classification Report:**

	precision	recall	f1-score	support
0	1.00	0.92	0.96	53
1	0.92	0.92	0.92	50
2	0.84	0.89	0.87	47
3	0.82	0.91	0.86	54
4	0.83	0.82	0.82	60
5	0.90	0.83	0.87	66
6	0.91	0.94	0.93	53
7	0.87	0.87	0.87	55
8	0.80	0.86	0.83	43
9	0.95	0.88	0.91	59
accuracy			0.88	540
macro avg	0.88	0.89	0.88	540
weighted avg	0.89	0.88	0.88	540




True: 0
True: 1
True: 2
True: 3
True: 4
0
2
4
6