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?

```python
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
C:\Users\hp\AppData\Local\Temp\ipykernel_2352\100899370.py:7: FutureWarning: DataFrame.applymap has been deprecated. Use DataFrame.map instead.
  df = df.applymap(lambda x: x.strip() if isinstance(x, str) else x)
```

```python
def entropy(labels):
    from math import log2
    from collections import Counter
    n = len(labels)
    counts = Counter(labels)
    ent = 0
    for count in counts.values():
        p = count / n
        ent -= p * log2(p)
    return ent

def info_gain(df, attribute, target='Result'):
    total_entropy = entropy(df[target])
    weighted = 0
```

```python
def entropy(labels):
    from math import log2
    from collections import Counter
    n = len(labels)
    counts = Counter(labels)
    ent = 0
    for count in counts.values():
        p = count / n
        ent -= p * log2(p)
    return ent

def info_gain(df, attribute, target='Result'):
    total_entropy = entropy(df[target])
    weighted = 0
    for val, subset in df.groupby(attribute):
        weighted += (len(subset)/len(df)) * entropy(subset[target])
    return total_entropy - weighted

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

```
import pandas as pd

df = pd.read_csv("student_lab3 q2.csv")
print(df)
```

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

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
for col in df.columns:
    df[col] = le.fit_transform(df[col])

print(df)
```

```
   Student  Study Hours  Attendance  Result
0        0            1           1       0
1        1            0           0       1
2        2            0           1       1
3        3            1           0       0
4        4            0           0       1
```

[18]

```
   Student  Study Hours  Attendance  Result
0        0            1           1       0
1        1            0           0       1
2        2            0           1       1
3        3            1           0       0
4        4            0           0       1
```

```python
from sklearn.tree import DecisionTreeClassifier

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

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

[19] Python

```
DecisionTreeClassifier
▶ Parameters
```

```python
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
plot_tree(model, feature_names=['Study Hours', 'Attendance'],
          class_names=['Fail', 'Pass'], filled=True)
plt.show()
```

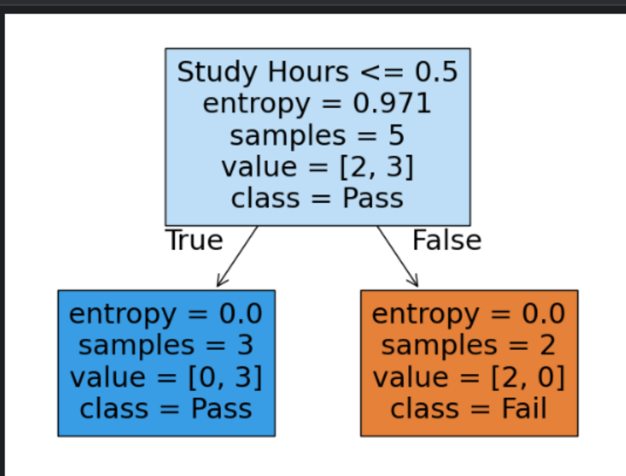
[20] Python

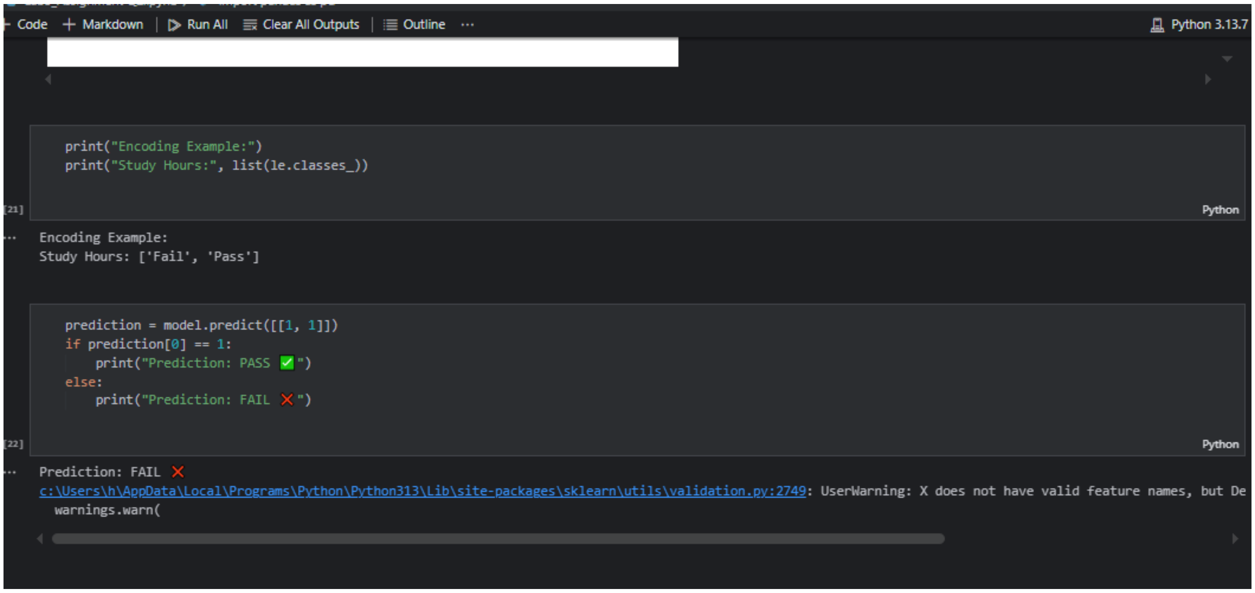
+ Code + Markdown | Run All Clear All Outputs | Outline ··· Python 3.13.7


```python
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
plot_tree(model, feature_names=['Study Hours', 'Attendance'],
          class_names=['Fail', 'Pass'], filled=True)
plt.show()
```


[20] Python





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 + Markdown | Run All Clear All Outputs | Outline ... Python 3.13.7

```python
from sklearn.datasets import load_iris
import pandas as pd
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
feature_names = iris.feature_names
class_names = iris.target_names
```

[16] Python

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

[17] Python

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

[18] Python

```
DecisionTreeClassifier
▶ Parameters
```


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

[18] Python

```
DecisionTreeClassifier
▶ Parameters
```

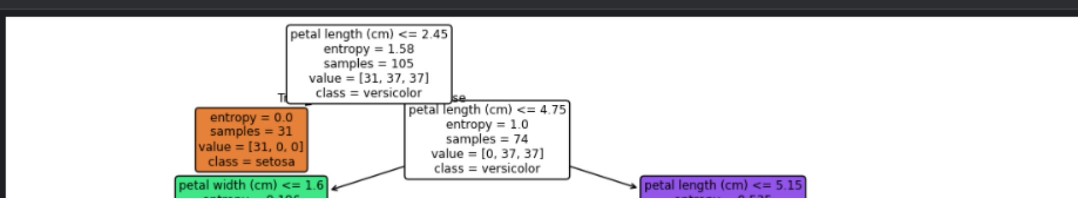

```python
y_pred = clf.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print(acc)
```

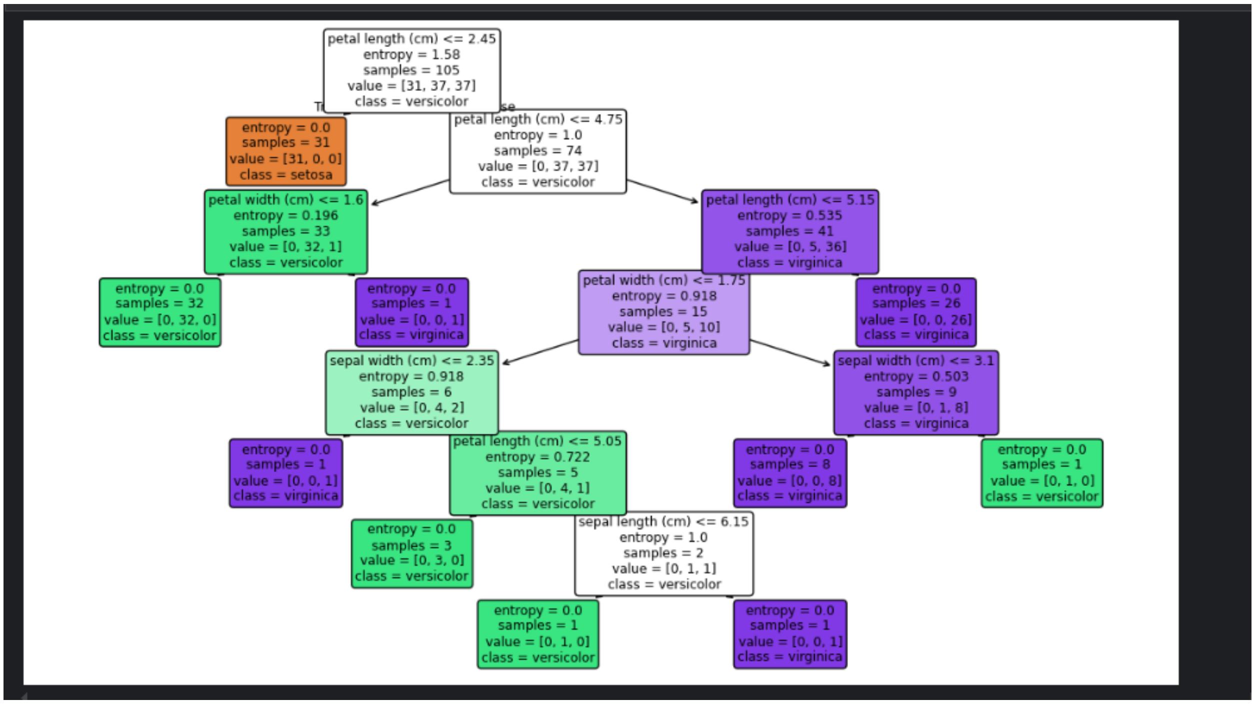
[19] Python

```
0.9777777777777777
```

```python
plt.figure(figsize=(14,8))
plot_tree(clf, feature_names=feature_names, class_names=class_names, filled=True, rounded=True)
plt.show()
```

[20] Python






```python
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])
```

Python

```
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

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Python 3.13.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
```

[107] Python

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

[108] Python

	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

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

[109] Python

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Python 3.13.7

[109] Python

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

[110] Python

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

[111] Python

DecisionTreeClassifier

▶ Parameters

```python
y_pred = clf.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print("Accuracy:", acc)
```

[112] Python

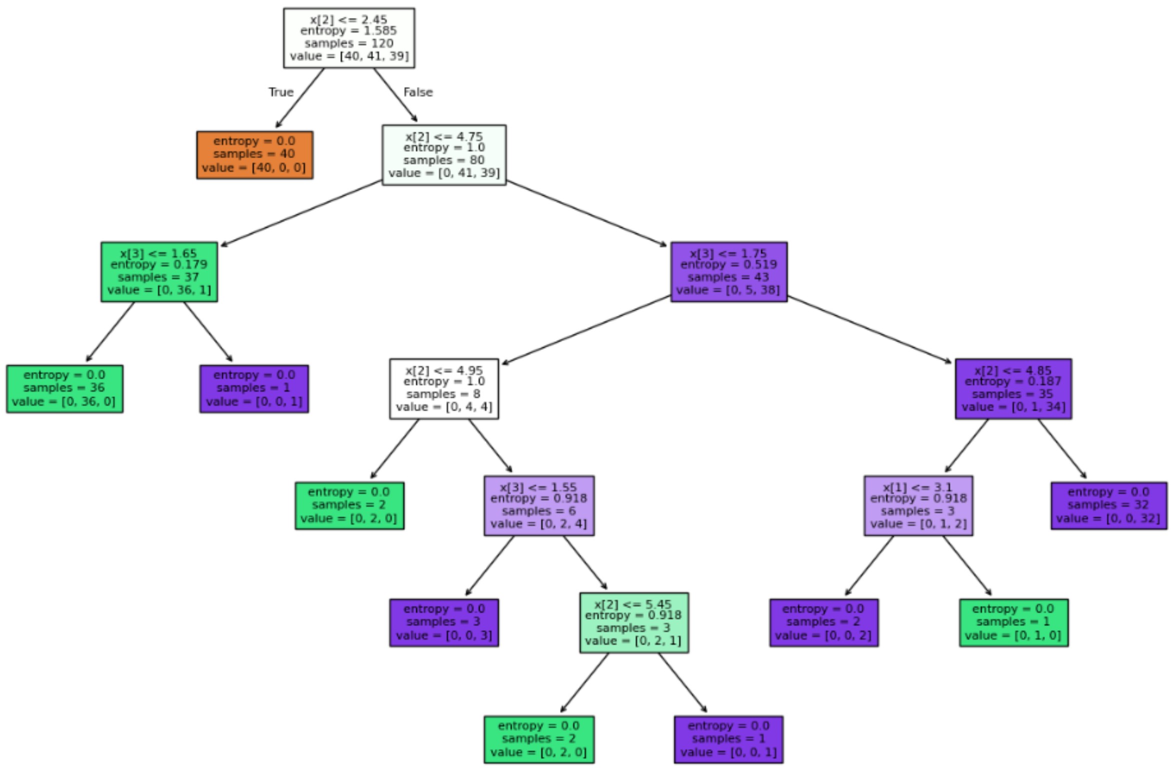
```
Accuracy: 1.0
```

```python
from sklearn import tree
import matplotlib.pyplot as plt

plt.figure(figsize=(15,10))
tree.plot_tree(clf, filled=True, fontsize=8)
plt.show()
```

[113] Python

Spaces: 4 Cell 6 of 8




```python
print("Limitations:")
print("- Decision Trees can overfit easily, especially on small datasets.")
print("- They are sensitive to small changes in data.")
print("- Pruning or ensemble methods (like Random Forest) can improve performance.")
```

14] Python

```
Limitations:
- Decision Trees can overfit easily, especially on small datasets.
- They are sensitive to small changes in data.
- Pruning or ensemble methods (like Random Forest) can improve performance.
```
