LEIC-T 2023/2024
Aprendizagem - Machine Learning

Homework I - Group 007

Miguel Teixeira - 103449

Rodrigo Alves - 103299

I

a) $x_1 = (1, 3, 4, 6)$

$x_2 = (-30, -10, 0, 20)$

Pearson:

$$\mathrm{CORR}(x_1, x_2) = \frac{\mathrm{Cov}(x_1, x_2)}{\sigma(x_1)\,\sigma(x_2)} \qquad \bar{x}_1 = 3{,}5 \quad \bar{x}_2 = -5$$

$$\mathrm{Cov}(x_1, x_2) = \frac{\sum_{i=1}^{4} (a_{i1} - \bar{x}_1)(a_{i2} - \bar{x}_2)}{4} = \frac{(4-3{,}5)(-30-(-5)) + (3-3{,}5)(-10-(-5))}{4}$$

$$\frac{+ (4-3{,}5)(0-(-5)) + (6-3{,}5)(20-(-5))}{} = \frac{62{,}5 + 2{,}5 + 2{,}5 + 62{,}5}{4} = 32{,}5$$

$$\sigma(x_1) = \sqrt{\frac{1}{4}\sum_{k}^{4} (y_1^k - \bar{y}_1)^2} = \frac{\sqrt{\sum_{k}^{4} (y_1^k - \bar{y}_1)^2}}{2} = \frac{\sqrt{(1-3{,}5)^2 + (3-3{,}5)^2 + (4-3{,}5)^2 + (6-3{,}5)^2}}{2} =$$

$$= \frac{\sqrt{6{,}25 + 0{,}25 + 0{,}25 + 6{,}25}}{2} = \frac{\sqrt{13}}{2}$$

$$\sigma(x_2) = \frac{\sqrt{(-30-(-5))^2 + (-10-(-5))^2 + (0-(-5))^2 + (20-(-5))^2}}{2} = \frac{\sqrt{625 + 25 + 25 + 625}}{2}$$

$$= \frac{10\sqrt{13}}{2}$$

$$\mathrm{Corr}(x_1, x_2) = \frac{32{,}5 \times 4}{10 \times 13} = \frac{130}{130} = 1$$

Spearman:

$R(x_1) = (1,2,3,4)$ $R(\bar{x}_1) = 2,5$ $R(\bar{x}_2) = 2,5$

$R(x_2) = (1,2,3,4)$

$$\text{Spearman} = \frac{Cov(R(x_1), R(x_2))}{S(R(x_1))\, S(R(x_2))}$$

$$Cov(R(x_1),R(x_2)) = \frac{(1-2,5)(1-2,5) + (2-2,5)(2-2,5) + (3-2,5)(3-2,5) + (4-2,5)(4-2,5)}{4} =$$

$$= \frac{2,25 + 0,25 + 0,25 + 2,25}{4} = \frac{5}{4}$$

$$S(R(x_1)) = \frac{\sqrt{(1-2,5)^2 + (2-2,5)^2 + (3-2,5)^2 + (4-2,5)^2}}{2} = \frac{\sqrt{5}}{2} = S(R(x_2))$$

In both cases, the correlation coefficients are 1, suggesting that the two variables, x1 and x2, are perfectly positively correlated.

$$\text{Spearman} = \frac{\frac{5}{4}}{\frac{\sqrt{5}}{2} \times \frac{\sqrt{5}}{2}} = 1$$

R.: Spearman and the Pearson Correlation have the same value, That is 1, so x_1 and x_2 are the same

6)

Pearson correlation: $\bar{x}_1 = 3,5$ $\bar{x}_2 = 13,875$

$$Corr(x_1,x_2) = \frac{Cov(x_1,x_2)}{S(x_1)\, S(x_2)}$$

$$Cov(x_1,x_2) = \frac{(1-3,5)(-3-13,875) + (3-3,5)(-0,5-13,875) + (4-3,5)(29-13,875) + (6-3,5)(\ldots}{4}$$

$$= \frac{42,1875 + 7,1875 + 7,5625 + 40,3125}{4} = 24,3125$$

$$S(x_1) = \frac{\sqrt{(1-3,5)^2 + (3-3,5)^2 + (4-3,5)^2 + (6-3,5)^2}}{2} = \frac{\sqrt{13}}{2}$$

$$S(x_2) = \frac{\sqrt{(-3-13,875)^2 + (-0,5-13,875)^2 + (29-13,875)^2 + (30-13,875)^2}}{2} = \frac{31,308}{2}$$

$$Corr(x_1,x_2) = \frac{24,3125}{\frac{\sqrt{13}}{2} \times \frac{31,308}{2}} = 0,862$$

Spearman Correlation:

$R(x_1) = (1,2,3,4)$

$R(x_2) = (1,2,3,4)$

A partir do exercício anterior [1 a)], em que os ranks são os mesmos, concluímos que:

$Cov(R(x_1), R(x_2)) = \frac{5}{4}$ $\qquad R(\bar{x}_1) = 2{,}5 \quad R(\bar{x}_2) = 2{,}5$

$\sigma(R(x_1)) = \sigma(R(x_2)) = \frac{\sqrt{5}}{2}$

$$corr(R(x_1), R(x_2)) = \frac{Cov(R(x_1), R(x_2))}{\sigma(R(x_1))\,\sigma(R(x_2))} = 1$$

R.: Spearman and the Pearson Correlation have different values, so x_1 and x_2 are different

2

a)

F_1	F_2	F_3	F_4	Output
c	a	b	x	n
a	a	c	a	t
a	b	b	a	t
c	b	c	x	m
a	b	b	c	f

$P(n) = P(m) = P(f) = \frac{1}{5}$

$P(t) = \frac{2}{5}$

$I(table) = -\frac{3}{5}\log\frac{1}{5} - \frac{2}{5}\log\frac{2}{5} = 1{,}922$ bit

F_1: $C_c = \{n, m\}$ $\qquad C_a = \{t, f, t\}$

$P(n) = \frac{1}{2}$ $\qquad P(t) = \frac{2}{3}$

$P(m) = \frac{1}{2}$ $\qquad P(f) = \frac{1}{3}$

$I = -\frac{2}{2}\log\frac{1}{2} = 1$ $\qquad I = -\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3} = 0{,}9183$

$I(F_1) = \frac{2}{5} \times 1 + \frac{3}{5} \times 0{,}9183 = 0{,}95098$

$gain(F_1) = 1{,}922 - 0{,}95098 = 0{,}97102$

F_2: $C_a = \{n, t\}$

$P(n) = \frac{1}{2}$

$P(t) = \frac{1}{2}$

$I = -\frac{2}{2} \log \frac{1}{2} = 1$

$C_b = \{t, m, f\}$

$P(t) = \frac{1}{3}$

$P(m) = \frac{1}{3}$

$P(f) = \frac{1}{3}$

$I = -\frac{3}{3} \log \frac{1}{3} = 1{,}585$

$I(F_2) = \frac{2}{5} \times 1 + \frac{3}{5} \times 1{,}585 = 1{,}351$

$gain(F_2) = 1{,}922 - 1{,}351 = 0{,}571$

F_3: $C_b = \{n, t, f\}$

$P(n) = \frac{1}{3}$

$P(t) = \frac{1}{3}$

$P(f) = \frac{1}{3}$

$I = -\log \frac{1}{3} = 1{,}585$

$C_c = \{t, m\}$

$P(t) = \frac{1}{2}$

$P(m) = \frac{1}{2}$

$I = -\frac{2}{2} \log \frac{1}{2} = 1$

$I(F_3) = \frac{2}{5} \times 1 + \frac{3}{5} \times 1{,}585 = 1{,}351$

$gain(F_3) = 1{,}922 - 1{,}351 = 0{,}571$

F_4: $C_x\{n, m\}$ $C_a\{t, t\}$ $C_c\{f\}$

$p(n) = \frac{1}{2}$

$p(m) = \frac{1}{2}$

$I = -\frac{2}{2} \log_2 \frac{1}{2} = 1$

$P(t) = 1$

$I = \log_2 1 = 0$

$p(f) = 1$

$I = \log_2 1 = 0$

$I(F_4) = \frac{2}{5} \times 1 + \frac{2}{5} \times 0 + \frac{1}{5} \times 0 = \frac{2}{5}$

$gain(F_4) = 1{,}922 - 0{,}4 = 1{,}522$

F_4 is the root because it provides the best gain

(b) F_4 is the first node in the tree

$\boxed{F_4 = x}$

F_1	F_2	F_3	Output
c	a	b	m
c	b	c	m

$\boxed{F_4 = a}$

F_1	F_2	F_3	Output
a	a	c	t
a	b	b	t

Done

$\boxed{F_4 = c}$

F_1	F_2	F_3	Output
a	b	b	f

Done

still has uncertainty, we have to decide between testing F_1, F_2 or F_3

$C_c\{m, m\}$

$E(F_1(C_c)) = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = \boxed{1}$

$E(F_1) = \boxed{1}$

$C_a\{m\}$

$C_b\{m\}$

$E(F_2) = E_{F_2}(C_a) + E_{F_2}(C_b) = \boxed{0}$

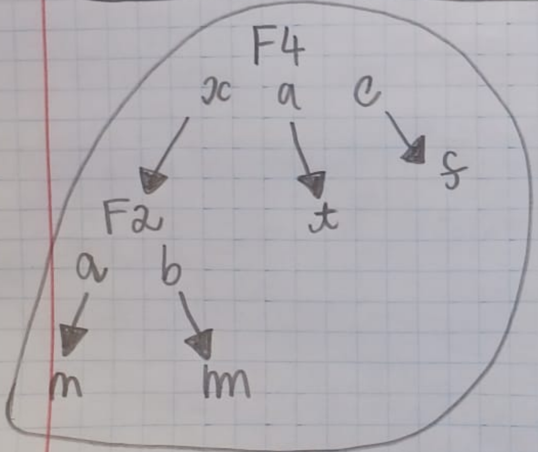
$C_b\{m\}$ $E_{F_3}(C_b) = 0$

$C_c\{m\}$ $E_{F_3}(C_c) = 0$

$E(F_3) = \boxed{0}$

$E(F_2) = E(F3) = 0$,

So we can choose either one, We chose $\boxed{F_2}$



$F_4 = x$

$F_2 = a$

F1	F3	Output
c	b	m

$F_2 = b$

F1	F3	output
c	c	m

(e)

	m	m	t	s
m	1	0	0	0
m	0	1	0	0
t	0	0	2	0
s	0	0	0	1

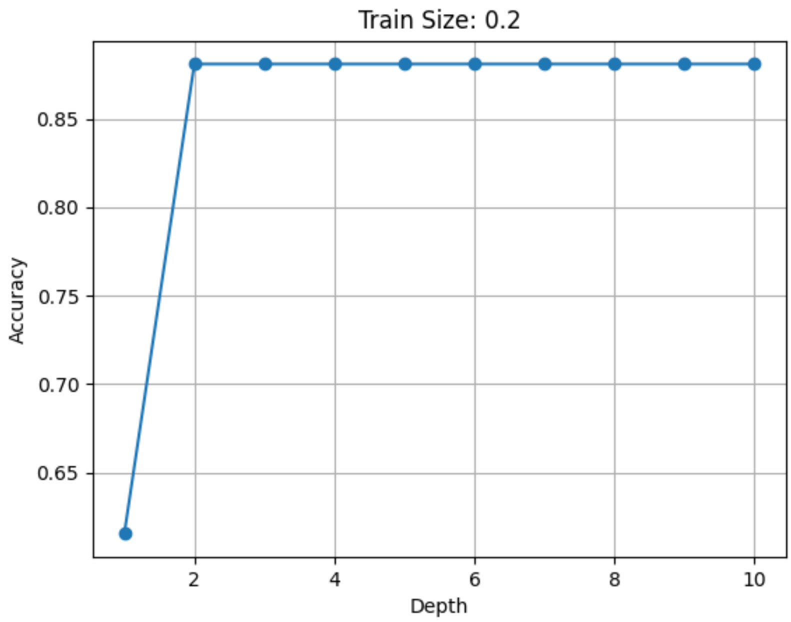
There are no predictions or false data

III Software Experiments

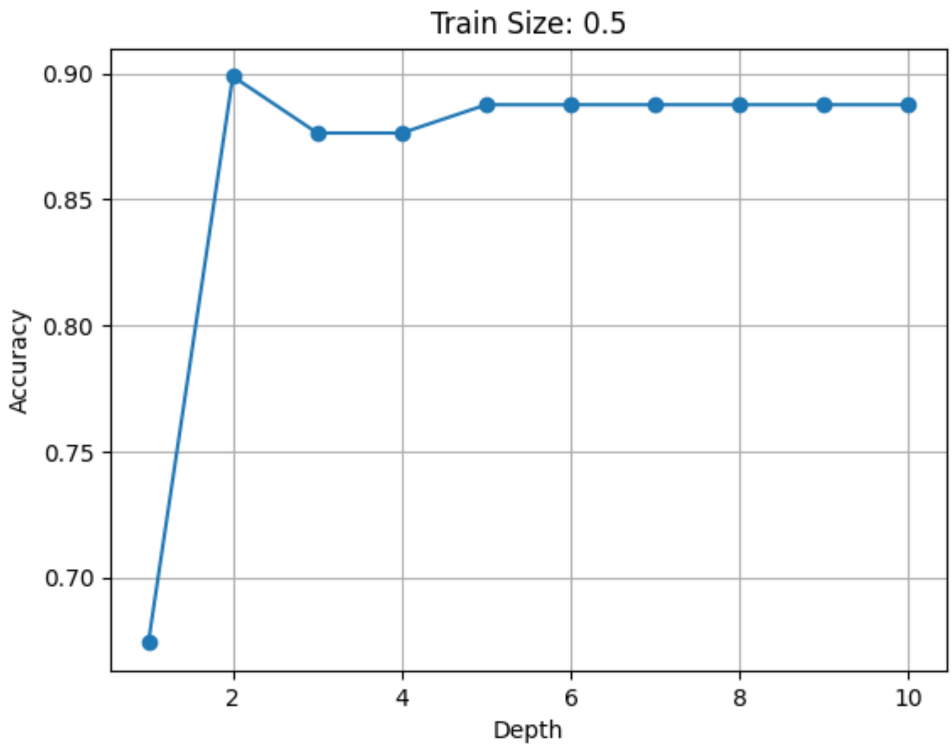
```python
import matplotlib.pyplot as plt
from sklearn import metrics, datasets, tree
from sklearn.model_selection import train_test_split

# Load the wine dataset
wine = datasets.load_wine()
X, y = wine.data, wine.target

# Define the different train sizes
train_sizes = [0.2, 0.5, 0.7]
group_number = 7

# Loop through different train sizes
for train_size in train_sizes:
    # Split the data
    X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=train_size, stratify=y, random_state=int(group_number))

    # Create a decision tree classifier and evaluate its accuracy for different depths
    depths = range(1, 11)  # Test depths from 1 to 10
    accuracies = []

    for depth in depths:
        # Create a decision tree classifier with the current depth
        clf = tree.DecisionTreeClassifier(max_depth=depth, random_state=int(group_number))
        clf.fit(X_train, y_train)

        # Predict on the test set
        y_pred = clf.predict(X_test)

        # Calculate accuracy
        accuracy = metrics.accuracy_score(y_test, y_pred)
        accuracies.append(accuracy)

        # Print depth and accuracy
        print(f"Train Size: {train_size}, Depth: {depth}, Accuracy: {accuracy:.4f}")

    # Plot accuracy vs. depth for the current train size
    plt.figure()
    plt.plot(depths, accuracies, marker='o', linestyle='-')
    plt.title(f"Train Size: {train_size}")
    plt.xlabel("Depth")
    plt.ylabel("Accuracy")
    plt.grid(True)
    plt.show()
```

1. (a)

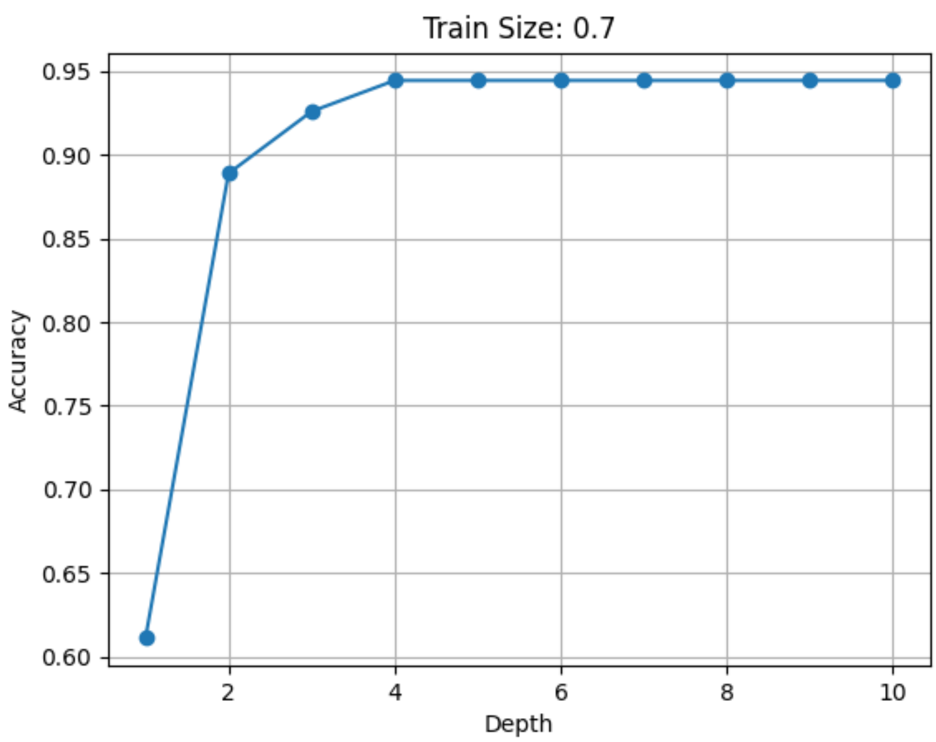
Train Size: 0.2, Depth: 1, Accuracy: 0.6154
Train Size: 0.2, Depth: 2, Accuracy: 0.8811
Train Size: 0.2, Depth: 3, Accuracy: 0.8811
Train Size: 0.2, Depth: 4, Accuracy: 0.8811
Train Size: 0.2, Depth: 5, Accuracy: 0.8811
Train Size: 0.2, Depth: 6, Accuracy: 0.8811
Train Size: 0.2, Depth: 7, Accuracy: 0.8811
Train Size: 0.2, Depth: 8, Accuracy: 0.8811
Train Size: 0.2, Depth: 9, Accuracy: 0.8811
Train Size: 0.2, Depth: 10, Accuracy: 0.8811



Train Size: 0.5, Depth: 1, Accuracy: 0.6742
Train Size: 0.5, Depth: 2, Accuracy: 0.8989
Train Size: 0.5, Depth: 3, Accuracy: 0.8764
Train Size: 0.5, Depth: 4, Accuracy: 0.8764
Train Size: 0.5, Depth: 5, Accuracy: 0.8876
Train Size: 0.5, Depth: 6, Accuracy: 0.8876
Train Size: 0.5, Depth: 7, Accuracy: 0.8876
Train Size: 0.5, Depth: 8, Accuracy: 0.8876
Train Size: 0.5, Depth: 9, Accuracy: 0.8876
Train Size: 0.5, Depth: 10, Accuracy: 0.8876

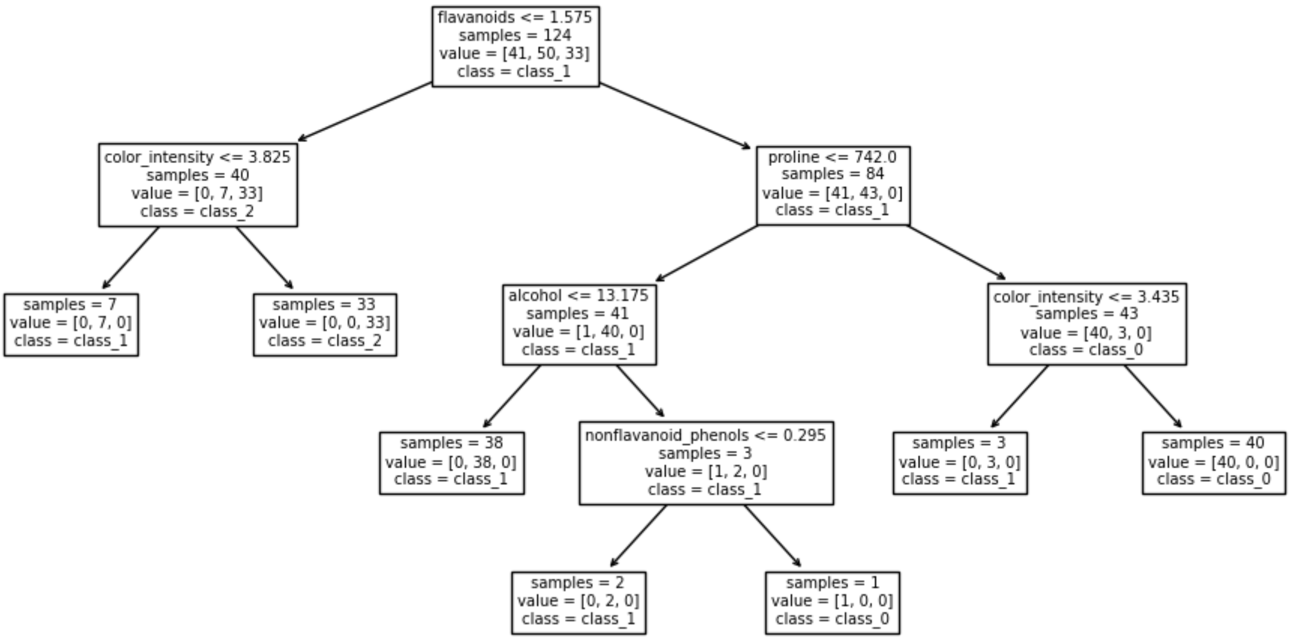


Train Size: 0.7, Depth: 1, Accuracy: 0.6111
Train Size: 0.7, Depth: 2, Accuracy: 0.8889
Train Size: 0.7, Depth: 3, Accuracy: 0.9259
Train Size: 0.7, Depth: 4, Accuracy: 0.9444
Train Size: 0.7, Depth: 5, Accuracy: 0.9444
Train Size: 0.7, Depth: 6, Accuracy: 0.9444
Train Size: 0.7, Depth: 7, Accuracy: 0.9444
Train Size: 0.7, Depth: 8, Accuracy: 0.9444
Train Size: 0.7, Depth: 9, Accuracy: 0.9444
Train Size: 0.7, Depth: 10, Accuracy: 0.9444



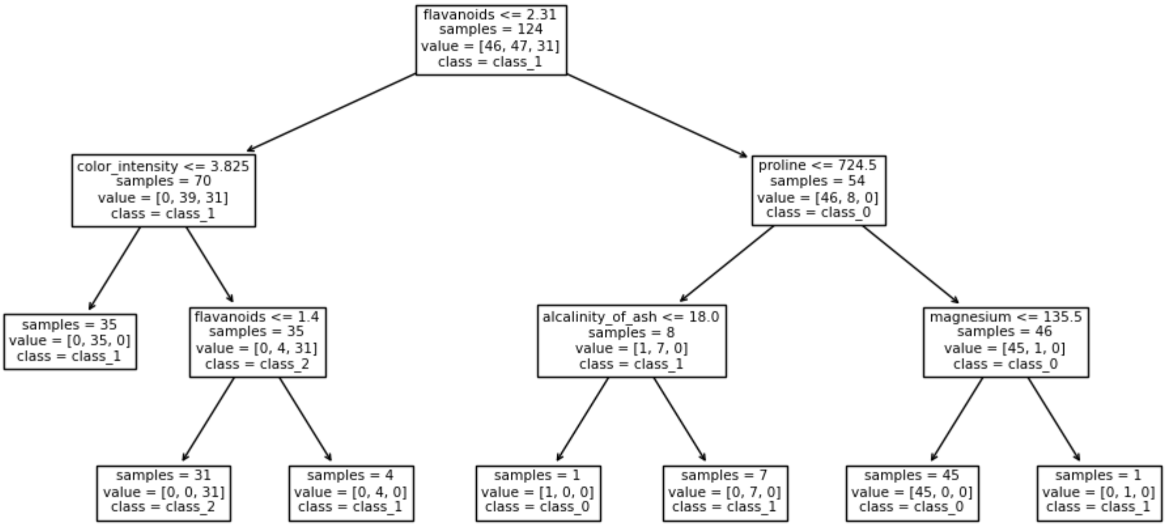
(b)

```python
import matplotlib.pyplot as plt
from sklearn import datasets, tree
from sklearn.model_selection import train_test_split

# Load the Wine dataset
wine = datasets.load_wine()
X, y = wine.data, wine.target

# Define the classifier (Decision Tree with entropy)
predictor = tree.DecisionTreeClassifier(criterion='entropy')

# Split the dat
train_size = 0.7
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=train_size, stratify=y, random_state=7)

# Train the classifier on the training data
predictor.fit(X_train, y_train)

# Plot the decision tree
figure = plt.figure(figsize=(12, 6))
tree.plot_tree(predictor, feature_names=wine.feature_names, class_names=wine.target_names, impurity=False)

# Save the decision tree as an image (e.g., PNG)
plt.savefig('decision_tree.png')

# Display the plot (optional)
plt.show()

```



(c)

```python
import matplotlib.pyplot as plt
from sklearn import datasets, tree
from sklearn.model_selection import train_test_split

# Load the Wine dataset
wine = datasets.load_wine()
X, y = wine.data, wine.target

# Define the classifier (Decision Tree with entropy)
predictor = tree.DecisionTreeClassifier(criterion='entropy')

# Split the data
train_size = 0.7

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=train_size, random_state=7)

# Train the classifier on the training data
predictor.fit(X_train, y_train)

# Plot the decision tree
figure = plt.figure(figsize=(12, 6))
tree.plot_tree(predictor, feature_names=wine.feature_names, class_names=wine.target_names, impurity=False)
plt.show()

```



The results may be different when not using stratify=y because, without stratification, the class distribution in the training and testing sets may not be representative of the overall dataset, potentially leading to biased model evaluations, while stratify=y ensures a stratified fashion, preserving the same proportion of target classes in both subsets.