Question 1 :

① Shannon entropy of target variable (edible) is calculated as follows:

$$H(\text{edible}) = -\left(\frac{16}{24} \times \log_2\left(\frac{16}{24}\right)\right) - \left(\frac{8}{24} \times \log_2\left(\frac{8}{24}\right)\right) = 0.92 \text{ bits}$$

② Each feature white, tall, frilly's entropies & information gains are calculated as follows:

- a) White :

white \ edible	True	False
0	9	5
1	7	3

$H(\text{white}, \text{edible}) = P(\text{white}=0) \times E(9,5) + P(\text{white}=1) \times E(7,3)$

(entropy of white=0 ← $E(9,5)$; entropy of white=1 ← $E(7,3)$)

$$P(\text{white}=0) = \frac{14}{24}, \quad E(9,5) = -\left(\frac{9}{14} \times \log_2\left(\frac{9}{14}\right)\right) - \left(\frac{5}{14} \times \log_2\frac{5}{14}\right) = 0.94$$

$$P(\text{white}=1) = \frac{10}{24}, \quad E(7,3) = -\left(\frac{7}{10} \times \log_2\left(\frac{7}{10}\right)\right) - \left(\frac{3}{10} \times \log_2\left(\frac{3}{10}\right)\right) = 0,88$$

$$H(\text{white}, \text{edible}) = \frac{14}{24} \times 0,94 + \frac{10}{24} \times 0,88 = 0.915$$

$$\text{Information gain} = H(\text{edible}) - H(\text{white}, \text{edible}) = 0,92 - 0,915 = 0,005$$

b) Tall :

tall \ edible	True	false
0	6	4
1	10	14

H(tall, edible) =

P(tall=0). E(6,4) + P(tall=1) × E(10,4)

tall=0

$E(\underbrace{6,4}_{\text{tall=0}}) = -\left(\frac{6}{10} \times \log_2 \frac{6}{10}\right) - \left(\frac{4}{10} \times \log_2\left(\frac{4}{10}\right)\right) = 0,97$

$E(\underbrace{10,4}_{\text{tall=1}}) = -\left(\frac{10}{14} \times \log_2 \frac{10}{14}\right) - \left(\frac{4}{14} \times \log_2\left(\frac{4}{14}\right)\right) = 0.861$

P(tall=0) = 10/24

P(tall=1) = 14/24

H(tall, edible) = 0,906

Info gain = H(edible) − H(tall, edible) = 0,92 − 0,906 = 0,014

c) Frilly :

Frilly \ edible	True	False
0	13	3
1	3	5

frilly=0

$E(\underbrace{13,3}_{\text{frilly=0}}) = -\left(\frac{13}{16} \cdot \log_2\left(\frac{13}{16}\right)\right) - \left(\frac{3}{16} \cdot \log_2\left(\frac{3}{16}\right)\right)$

= 0.6962

E(3,5) = 0,9544

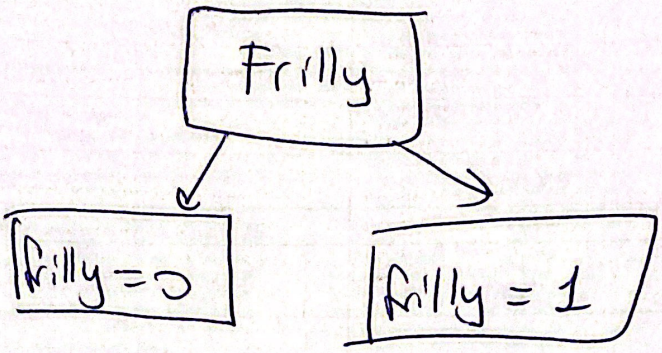
P(frilly=0) = 16/24

P(frilly=1) = 8/24

H(frilly, edible) = 0,78228 → Info gain = 0,92 − 0,7822

= 0.1377

(3) Comparing information gains feature 'Frilly' should be next node with max info gain.



⇒ Frilly=0, consider attribute white's entropy.

Entropy(frilly=0) = 0.696205 as calculated before.

when frilly=0 & white=0, there are 6 True's & 3 Fales.

$$\text{entropy(white=0)} = -\frac{6}{9}\log_2\left(\frac{6}{9}\right) - \frac{3}{9}\log_2\left(\frac{3}{9}\right) = 0.92$$

entropy(white=1) = 0 since all target values are True's.

$$\text{Information gain (frilly=0, white)} = 0.696205 - \underbrace{\frac{9}{16}}_{P(\text{white}=0)} \times \underbrace{0.92}_{\text{entropy white}=0}$$

$$\text{Information gain} = 0.696205 - 0.5175 = 0.178$$

$$- \underbrace{\frac{7}{16}}_{P(\text{white}=1)} \times 0$$

Information gain (frilly=0, white) = 0.178

⇒ Frilly = 0 , consider attribute tall's entropy.

Entropy(frilly=0) = 0.696205

when frilly=0 & tall=0 there are → 4 True
→ 3 False

when frilly=0 & tall=1 there are → 9 True
→ 0 False

entropy(tall=0) = $-\frac{4}{7}\left(\log_2\left(\frac{4}{7}\right)\right) - \frac{3}{7}\left(\log_2\frac{3}{7}\right) = 0.981$

(−0.81 and −1.21)

entropy(tall=1) = 0

Information gain(frilly=0, tall) = E(tall)

= 0.696205 − (P(tall=0).E(tall=0)) − 0

= 0.2672

($\frac{7}{16}$ = 0.429, 0.981)

(4) Therefore, Gain(while) = 0.178
Gain(tall) = 0.2672
) for frilly = 0

(5) Now consider the branch frilly = 1

(frilly = 1)

white	tall	edible
0	0	T
0	0	T
0	1	F
0	1	F
1	0	F
1	1	F
1	1	F
0	1	F

entropy (frilly = 1) = 0,9544

a) Consider white :

(white = 0) & frilly = 1 there are → 2T, 3F

entropy (white = 0) = $-\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right)$ = −0,4 × −1,32 −0,6 × −0,74 = 0,97

entropy (white = 1) = 0

Info gain (frilly = 1, white = 0) = 0,9544 − (P(white = 0), E(white = 0)) − 0

P(white = 0) = 5/8, E(white = 0) = 0.97

= 0.35

b) Consider tall :

tall = 0 & frilly = 1 there are → 2T, 1F

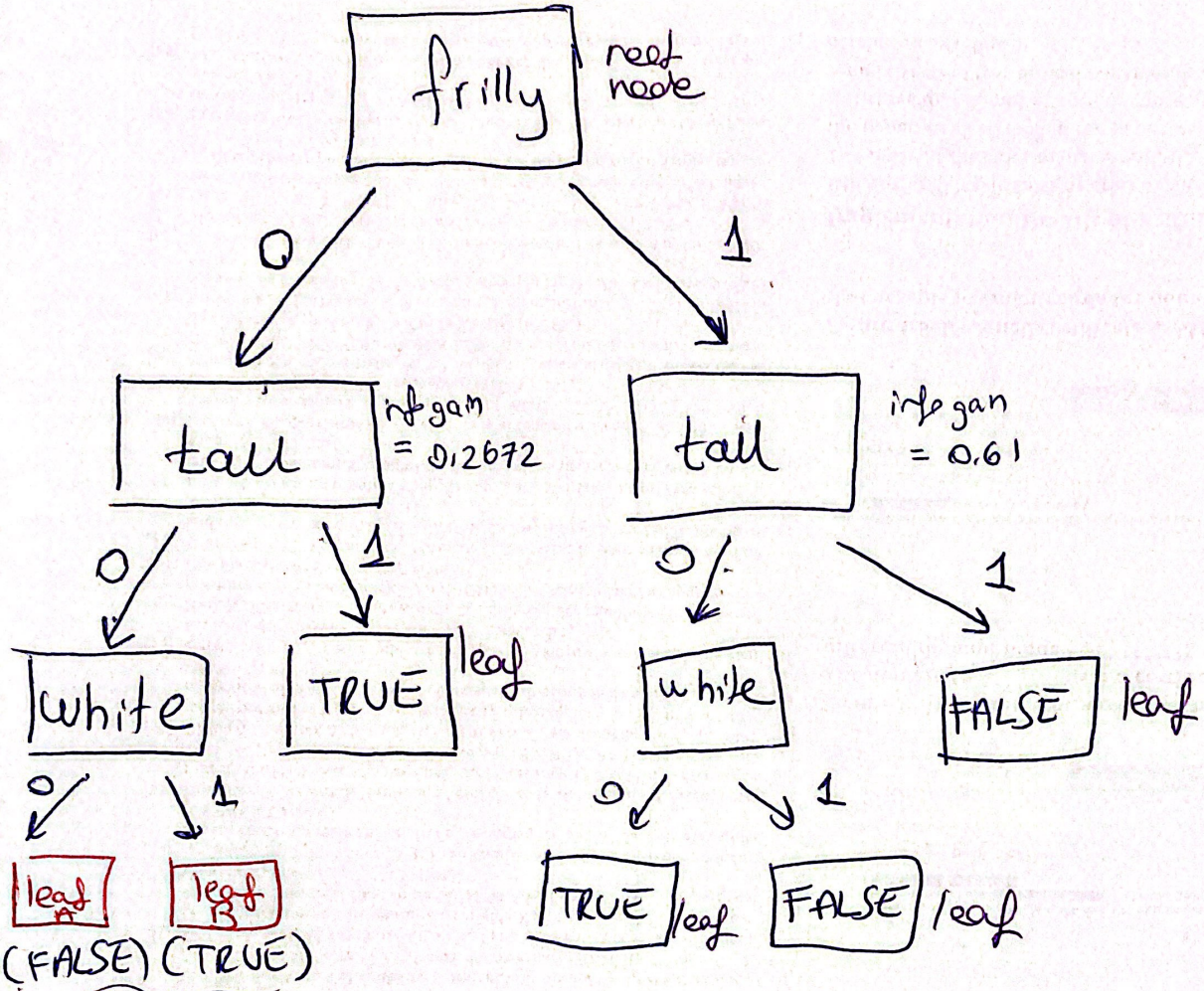
tall = 1 & frilly = 1 there are → 5F, 0T

entropy (tall = 0) = $-\frac{2}{3}\log_2\left(\frac{2}{3}\right) - \frac{1}{3}\log_2\left(\frac{1}{3}\right)$ = 0,9182 ≈ 0,344

entropy (tall = 1) = 0

info gain (frilly = 1, tall) = 0,9544 − 0,9182 × $\frac{3}{5}$ = 0,61

Therefore the tree structure is :



↓ NOTE on this.

① When frilly=0, tall=0, white=0 there are → 1T
                                                ↘ 3F

② " " " , white=1 " " → 3T
                        ↘ 0F

In the raw data table case ① some entries point to different classifications. In this case my opinion is classify frilly=0, tall=0, white=0 (leaf A) as false and leaf B as TRUE. So we are ignoring the false entries in case ①.

Question 2:

The tree looks same with Question 1.

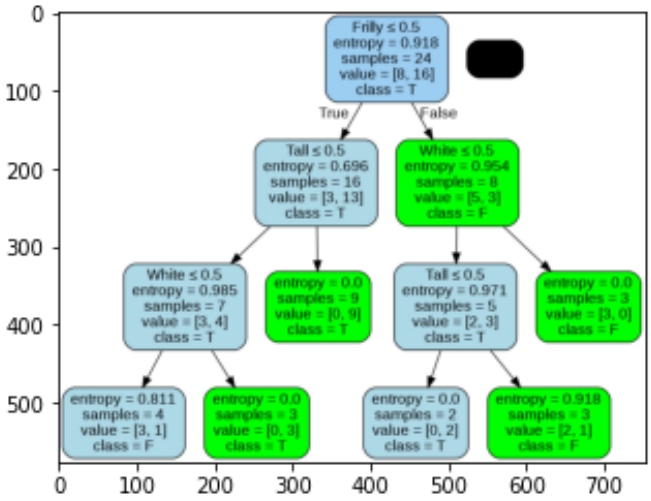


Figure 1: Decision tree

Question 3:

Tree built with entropy criteria:

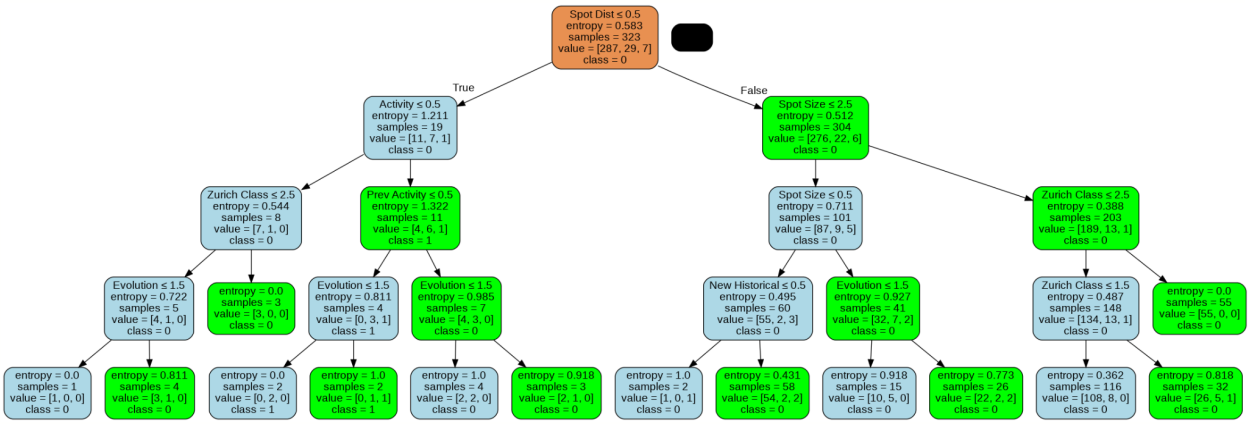


Figure 2: Entropy based built tree

Tree built with gini index:

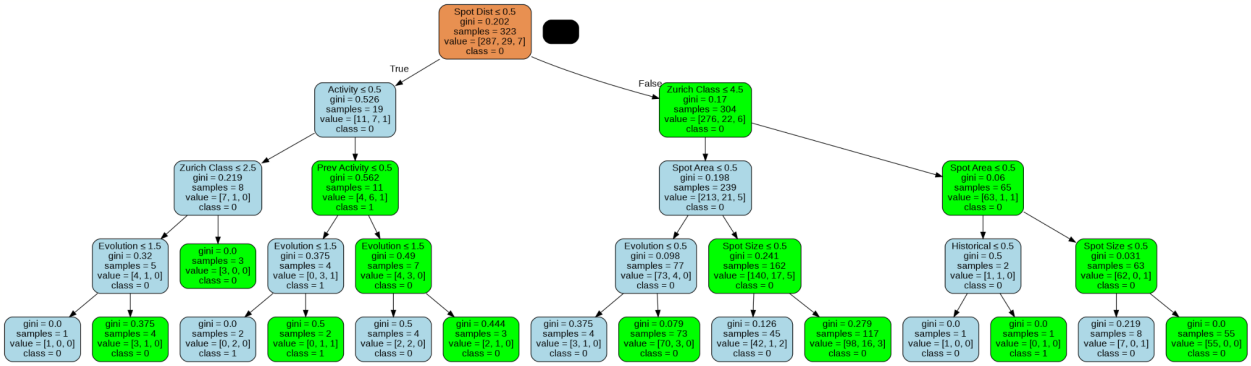


Figure 3: Gini Index based tree

Gini index seems to work better.

The codes for both questions are given at the Appendix.

Appendix:
Code:
Question 2:

```
import pandas as pd
df = pd.read_excel('AlienMushrooms.xlsx')
df.head(8)
X = df.drop(['Edible'],axis = 1)
Y = df.Edible
from sklearn import tree
model = tree.DecisionTreeClassifier(criterion='entropy')
model = model.fit(X,Y)
import pydotplus
import collections

# for a two-class tree, call this function like this:
# writegraphtofile(clf, ('F', 'T'), dirname+graphfilename)
def writegraphtofile(classifier, classnames, pathname):
    dot_data = tree.export_graphviz(model, out_file=None,    # merely to
write the tree out
                            feature_names=X.columns,
                            class_names=classnames,
                            filled=True, rounded=True,
                            special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data)
    colors = ('lightblue', 'green')
    edges = collections.defaultdict(list)
    for edge in graph.get_edge_list():
        edges[edge.get_source()].append(int(edge.get_destination()))
    for edge in edges:
        edges[edge].sort()
        for i in range(2):
            dest = graph.get_node(str(edges[edge][i]))[0]
            dest.set_fillcolor(colors[i])
    graph.write_png(pathname)

writegraphtofile(model, ('F', 'T'), 'deno.png')
```

Question 3:

```
df = pd.read_excel('FlareData.xlsx')
df.head()
input = df.drop(['C class','M class','X class'],axis='columns')
input.head()
target = df['C class']
target.head()
from sklearn.preprocessing import LabelEncoder
le_model = LabelEncoder()
input['Zurich Class'] = le_model.fit_transform(input['Zurich Class'])
input['Spot Size'] = le_model.fit_transform(input['Spot Size'])
input['Spot Dist'] = le_model.fit_transform(input['Spot Dist'])
input['Activity'] = le_model.fit_transform(input['Activity'])
input['Evolution'] = le_model.fit_transform(input['Evolution'])
input['Prev Activity'] = le_model.fit_transform(input['Prev Activity'])
input['Historical'] = le_model.fit_transform(input['Historical'])
input['New Historical'] = le_model.fit_transform(input['New Historical'])
input['Area'] = le_model.fit_transform(input['Area'])
input['Spot Area'] = le_model.fit_transform(input['Spot Area'])
input.head(10)
model2 = tree.DecisionTreeClassifier(criterion='entropy',max_depth=4)
model2 = model2.fit(input,target)
featurelabels= ['Zurich Class','Spot Size','Spot
Dist','Activity','Evolution','Prev Activity','Historical','New
Historical','Area', 'Spot Area' ]
def writegraphtofile2(classifier, classnames, pathname):
    dot_data = tree.export_graphviz(model2, out_file=None,    # merely to
write the tree out
                            feature_names=featurelabels,
                            class_names=classnames,
                            filled=True, rounded=True,
                            special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data)
    colors = ('lightblue', 'green')
    edges = collections.defaultdict(list)
    for edge in graph.get_edge_list():
        edges[edge.get_source()].append(int(edge.get_destination()))
    for edge in edges:
        edges[edge].sort()
        for i in range(2):
```

```
            dest = graph.get_node(str(edges[edge][i]))[0]
            dest.set_fillcolor(colors[i])
    graph.write_png(pathname)
writegraphtofile2(model2, ('0', '1', '2'), 'ent.png')
# Read Images
img = mpimg.imread('ent.png')

# Output Images
plt.imshow(img)

model2 = tree.DecisionTreeClassifier(max_depth=4)
model2 = model2.fit(input,target)

writegraphtofile2(model2, ('0', '1', '2'), 'ent2.png')

# Read Images
img = mpimg.imread('ent2.png')

# Output Images
plt.imshow(img)
```