Question 2:

The tree looks same with Question 1.

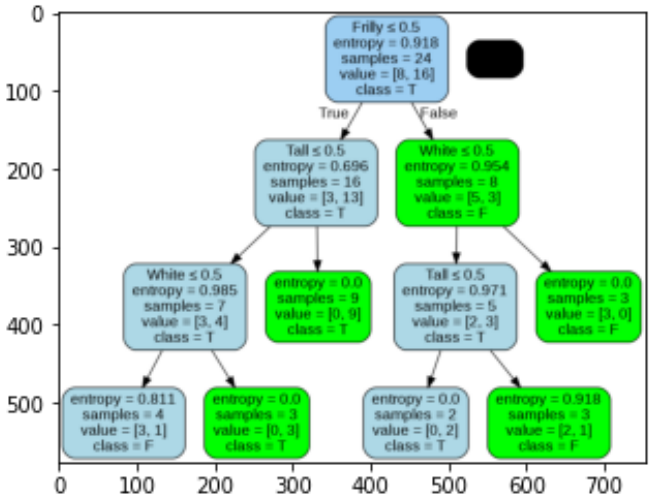

Figure 1: Decision tree

Question 3:

Tree built with entropy criteria:

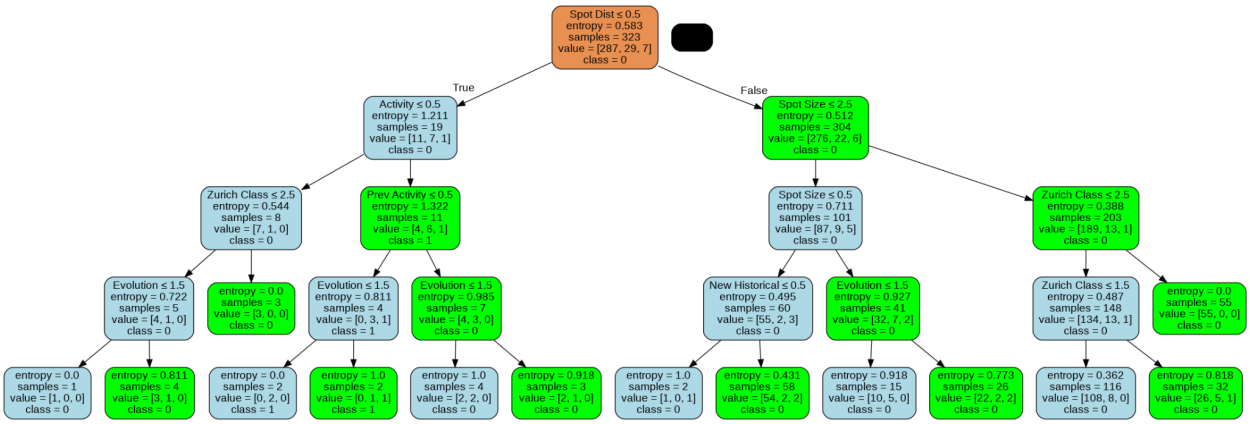

Figure 2: Entropy based built tree

Tree built with gini index:

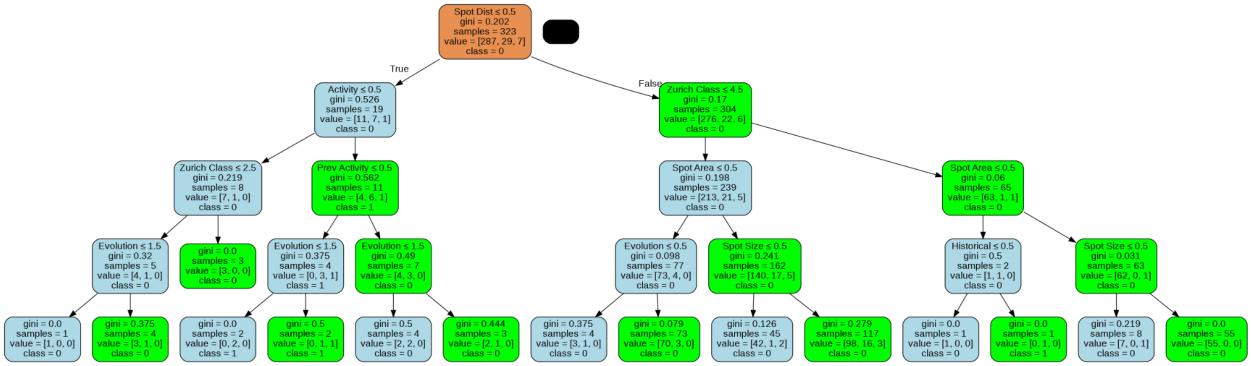

Figure 3: Gini Index based tree

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)
```