CS 422 Homework 2

Abdullah Moizuddin

Recitation Exercises

Question 2:

1.
$Gini = \sum_{i=1}^{n}(p_i)^2$

A.
$= 1 - [(\frac{10}{20})^2 + (\frac{10}{20})^2]$
$= 1 - [\frac{10000}{40000} + \frac{10000}{40000}]$
$= 1 - [\frac{1}{4} + \frac{1}{4}]$
$= \frac{1}{2}$

B.
$= 1 - [(\frac{0}{20})^2 + (\frac{20}{20})^2]$
$= 1 - [\frac{0}{1} + \frac{1}{1}]$
$= 0$

C.
Male:
$= 1 - [(\frac{6}{10})^2 + (\frac{4}{10})^2]$
$= 1 - [\frac{36}{100} + \frac{16}{100}]$
$= 1 - \frac{52}{100}$
$= .48$
Its the same thing for female cause there is 6 females and 4 males in the other class. Then we can take the Average GINI (weighted).
$Avg. Gini = \frac{.48+.48}{2} = .48$

D.
Family Car:
$= 1 - [(\frac{1}{4})^2 + (\frac{3}{4})^2]$
$= 1 - [\frac{1}{16} + \frac{9}{16}]$
$= 1 - \frac{10}{16}$
$= .375$

Sports Car:
$= 1 - [(\frac{8}{8})^2 + (\frac{0}{8})^2]$
$= 1 - [\frac{1}{1} + 0]$
$= 0$

Luxury Car:
$= 1 - [(\frac{1}{8})^2 + (\frac{7}{8})^2]$
$= 1 - [\frac{1}{64} + \frac{49}{64}]$
$= 1 - \frac{50}{64}$
$= .21875$

avg. Gini (weighted).
$[\frac{4}{20} * (.375)] + [\frac{8}{20} * (.21875)] + [0]$
$= 0.1625$

E.
Small Shirt:
$= 1 - [(\frac{3}{5})^2 + (\frac{2}{5})^2]$
$= 1 - [.52]$
$= .48$

Medium Shirt:
$= 1 - [(\frac{3}{7})^2 + (\frac{4}{7})^2]$
$= 1 - [.51]$
$= .49$

Large Shirt:
$= 1 - [(\frac{2}{4})^2 + (\frac{2}{4})^2]$
$= 1 - [.5]$
$= .5$

Extra Large Shirt:
$= 1 - [(\frac{2}{4})^2 + (\frac{2}{4})^2]$
$= 1 - [.5]$
$= .5$

Average GINI (weighted).
$[\frac{5}{20} * (.48)] + [\frac{7}{20} * (.49)] + [\frac{4}{20} * (.5)] + [\frac{4}{20} * (.5)]$
$= 0.4915$

F.
The car type has the Lowest GINI.

G.
Customer ID cant be a good predictor because every new customer gets assigned a new ID and so there wont ever be any shared IDs

Question 3:

1. $Entropy = \sum_{i=1}^{n}(p_i)log_2(p_i)$
   A.
   $= -(\frac{4}{9})log_2(\frac{4}{9}) - (\frac{5}{9})log_2(\frac{5}{9})$
   $= 0.9911$
   B.
   Entropy for a_1:
   $\frac{4}{9}[-(\frac{3}{4})log_2(\frac{3}{4}) - (\frac{1}{4})log_2(\frac{1}{4})] + \frac{5}{9}[-(\frac{1}{5})log_2(\frac{1}{5}) - (\frac{4}{5})log_2(\frac{4}{5})]$
   $= .7616$
   Entropy for a_2:
   $\frac{5}{9}[-(\frac{2}{5})log_2(\frac{2}{5}) - (\frac{3}{5})log_2(\frac{3}{5})] + \frac{4}{9}[-(\frac{2}{4})log_2(\frac{2}{4}) - (\frac{2}{4})log_2(\frac{2}{4})]$
   $= .9839$
   Gain from a_1:
   $0.9911 - .7616 = 0.2294$ Gain from a_2:
   $0.9911 - .9893 = 0.0072$
   C.
   sorted mapping of counts of a_3
   a3 = {

```
1.0 : 1,
3.0 : 1,
4.0 : 1,
5.0 : 2,
6.0 : 1,
7.0 : 2,
8.0 : 1}
```

In [3]:
```
import math
def entropy(pc1,pc2):
    return -1*(pc1)*math.log2(pc1) -(pc2)*math.log2(pc2)
def gain(e_original,e_attribute):
    return e_original-e_attribute
def weighted_avg(weight1,e1,weight2,e2):
    return (weight1*e1)+(weight2*e2)
def gini(pc1,pc2):
    return 1-((pc1**2)+(pc2**2))

e0 = entropy(4/9,5/9)
```

Split 0.5:
vals <= split: 0
vals > split: 4/9 are of '+' 5/9 are of '-'

In [4]:
```
e1 = 0
e2 = entropy(4/9,5/9)
avg_e = weighted_avg(0,e1,9/9,e2)
print("entropy of vals <= split:", e1)
print("entropy of vals > split:", e2)
print("gain:", gain(e0,avg_e))
```

```
entropy of vals <= split: 0
entropy of vals > split: 0.9910760598382222
gain: 0.0
```

Split 2:

vals <= split: 1

1/1 are of '+' 0/1 are of '-'

vals > split: 8

3/8 are of '+' 5/8 are of '-'

In [5]:
```
e1 = 0
e2 = entropy(3/8,5/8)
avg_e = weighted_avg(1/9,e1,8/9,e2)
print("entropy of vals <= split:", e1)
print("entropy of vals > split:", e2)
print("weighted avg:", avg_e)
print("gain:", gain(e0,avg_e))
```

```
entropy of vals <= split: 0
entropy of vals > split: 0.9544340029249649
weighted avg: 0.8483857803777466
gain: 0.14269027946047563
```

Split 3.5:

vals <= split: 2

1/2 are of '+' 1/2 are of '-'

vals > split: 7

3/7 are of '+' 4/7 are of '-'

In [6]:
```
e1 = entropy(1/2,1/2)
e2 = entropy(3/7,4/7)
avg_e = weighted_avg(2/9,e1,7/9,e2)
print("entropy of vals <= split:", e1)
print("entropy of vals > split:", e2)
print("weighted avg:", avg_e)
print("gain:", gain(e0,avg_e))
```

```
entropy of vals <= split: 1.0
entropy of vals > split: 0.9852281360342515
weighted avg: 0.9885107724710845
gain: 0.002565287367137681
```

Split 4.5:

vals <= split: 3

3/2 are of '+' 1/3 are of '-'

vals > split: 6

2/6 are of '+' 4/6 are of '-'

In [7]:
```
e1 = entropy(2/3,1/3)
e2 = entropy(2/6,4/6)
avg_e = weighted_avg(3/9,e1,6/9,e2)
print("entropy of vals <= split:", e1)
print("entropy of vals > split:", e2)
print("weighted avg:", avg_e)
print("gain:", gain(e0,avg_e))
```

```
entropy of vals <= split: 0.9182958340544896
entropy of vals > split: 0.9182958340544896
weighted avg: 0.9182958340544896
gain: 0.07278022578373267
```

Split 5.5:

vals <= split: 4

2/4 are of '+' 2/4 are of '-'

vals > split: 5

2/5 are of '+' 2/5 are of '-'

In [8]:
```
e1 = entropy(2/4,2/4)
e2 = entropy(2/5,3/5)
avg_e = weighted_avg(4/9,e1,5/9,e2)
print("entropy of vals <= split:", e1)
print("entropy of vals > split:", e2)
print("weighted avg:", avg_e)
print("gain:", gain(e0,avg_e))
```

```
entropy of vals <= split: 1.0
entropy of vals > split: 0.9709505944546686
weighted avg: 0.9838614413637048
gain: 0.007214618474517431
```

Split 6.5:

vals <= split: 6

3/6 are of '+' 3/6 are of '-'

1/3 are of '+' 2/3 are of '-'

In [9]:
```
e1 = entropy(3/6,3/6)
e2 = entropy(1/3,2/3)
avg_e = weighted_avg(6/9,e1,3/9,e2)
print("entropy of vals <= split:", e1)
print("entropy of vals > split:", e2)
print("weighted avg:", avg_e)
print("gain:", gain(e0,avg_e))
```

```
entropy of vals <= split: 1.0
entropy of vals > split: 0.9182958340544896
weighted avg: 0.9727652780181631
gain: 0.018310781820059074
```

Split 7.5:

vals <= split: 8

4/8 are of '+' 4/8 are of '-'

vals > split: 1

0/1 are of '+' 1/1 are of '-'

In [10]:
```
e1 = entropy(4/8,4/8)
e2 = 0
avg_e = weighted_avg(8/9,e1,1/9,e2)
print("entropy of vals <= split:", e1)
print("entropy of vals > split:", e2)
print("weighted avg:", avg_e)
print("gain:", gain(e0,avg_e))
```

```
entropy of vals <= split: 1.0
entropy of vals > split: 0
weighted avg: 0.8888888888888888
gain: 0.10218717094933338
```

Split 8.5:

vals <= split: 9

4/9 are of '+' 5/9 are of '-'

vals > split: 0

In [11]:
```
e1 = entropy(4/9,5/9)
e2 = 0
avg_e = weighted_avg(9/9,e1,0,e2)
print("entropy of vals <= split:", e1)
print("entropy of vals > split:", e2)
print("gain:", gain(e0,avg_e))
```

```
entropy of vals <= split: 0.9910760598382222
entropy of vals > split: 0
gain: 0.0
```

D.
The best split is a_1
E.
Classification error $= 1 - max(p_i)$
$C(a_1) = 1 - \frac{7}{9}$
$= \frac{2}{9}$
$C(a_2) = 1 - \frac{5}{9}$
$= \frac{4}{9}$
a_1 is better because it has a lower classification error.
F.

In [12]:
```
g1 = gini(3/4,1/4)
g2 = gini(1/5,4/5)
g_avg = weighted_avg(4/9,g1,5/9,g2)
print("gini of a1:", g_avg)
```

```
gini of a1: 0.34444444444444433
```

In [13]:
```
g1 = gini(2/5,3/5)
g2 = gini(2/4,2/4)
g_avg = weighted_avg(5/9,g1,4/9,g2)
print("gini of a2:", g_avg)
```

```
gini of a2: 0.4888888888888889
```

a_1 has the better gini. It is the better split.

Question 5

1.
   when A = T --> '+' : 4 and '-' : 3
   when A = F --> '+' : 0 and '-' : 3
   when B = T --> '+' : 3 and '-' : 1
   when A = F --> '+' : 1 and '-' : 5

In [22]:
```
E_origin = entropy(4/10,6/10)
E_AT = entropy(4/7,3/7)
E_AF = 0
w_avg = weighted_avg(7/10,E_AT,3/10,E_AF)
print("gain with split on A: ", gain(E_origin,w_avg))

E_BT = entropy(3/4,1/4)
E_BF = entropy(1/6,5/6)
w_avg = weighted_avg(4/10,E_BT,6/10,E_BF)
print("gain with split on B: ", gain(E_origin,w_avg))
```

```
gain with split on A:  0.2812908992306925
gain with split on B:  0.256425891682003
```

A has more gain, Thus A is the better split.

B.

In [23]:
```
G_origin = gini(4/10,6/10)
G_AT = gini(4/7,3/7)
G_AF = 0
w_avg = weighted_avg(7/10,G_AT,3/10,G_AF)
print("gain with split on A: ", gain(G_origin,w_avg))

G_BT = gini(3/4,1/4)
G_BF = gini(1/6,5/6)
w_avg = weighted_avg(4/10,G_BT,6/10,G_BF)
print("gain with split on B: ", gain(G_origin,w_avg))
```

```
gain with split on A:  0.13714285714285707
gain with split on B:  0.16333333333333333
```

B has more gain. Thus, B is the better split.

C.
yes because the gain as a result from GINI isnt the same as the gain as a result from entropy.

Question 6

Class 0: $P = 7$, $C_1 = 3$, $C_2 = 4$
Class 1: $P = 3$, $C_1 = 0$, $C_2 = 3$

In [29]:
```
G_P = gini(3/10,7/10)
G_c1 = gini(3/7,4/7)
G_c2 = 0
W_g=weighted_avg(7/10,G_split0,3/10,G_split1)
print("")
print("Gini of parent:", G_P)
print("misclassification error Parent:", 3/10)
print("Gini of C1:", G_c1)
print("misclassification error C1:", 3/7)
print("Gini of C2:", G_c2)
print("misclassification error C2:", 0)
print("weighted gini:", W_g)
print("weighted missclassification:", .3)
```

```
Gini of parent: 0.42000000000000004
misclassification error Parent: 0.3
Gini of C1: 0.48979591836734704
misclassification error C1: 0.42857142857142855
Gini of C2: 0
misclassification error C2: 0
weighted gini: 0.3428571428571429
weighted missclassification: 0.3
```

no. at child 1 the GINI is near .5 and the miss classification is near .5 also so its essentially flipping a coin.

Recitation Exercises

Problem 2.1

In [32]:
```
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.preprocessing import Imputer
from sklearn import decomposition
import matplotlib.pyplot as plt
%matplotlib inline
```

In [33]:
```
from scipy import stats

from sklearn.datasets import load_iris

from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_graphviz

from sklearn import model_selection
from sklearn import metrics
```

In [39]:
```
iris = load_iris()
x = iris.data
y = iris.target
X_train, X_test, y_train, y_test = model_selection.train_test_split(x
,y,test_size=0.2,random_state=10)

for x in range(5):
    classifier = DecisionTreeClassifier(max_depth=x+1,min_samples_spl
it=5,
                                        min_impurity_decrease=0.095,
random_state=6)
    classifier = classifier.fit(X_train,
                  y_train)
    expected = y_test
    predicted = classifier.predict(X_test)
    print('max_depth =', x+1, '\n', metrics.classification_report(exp
ected,predicted))
    print()
```

```
max_depth = 1
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       0.00      0.00      0.00        13
           2       0.35      1.00      0.52         7

    accuracy                           0.57        30
   macro avg       0.45      0.67      0.51        30
weighted avg       0.41      0.57      0.45        30


max_depth = 2
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.85      0.92        13
           2       0.78      1.00      0.88         7

    accuracy                           0.93        30
   macro avg       0.93      0.95      0.93        30
weighted avg       0.95      0.93      0.93        30


max_depth = 3
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.85      0.92        13
           2       0.78      1.00      0.88         7

    accuracy                           0.93        30
   macro avg       0.93      0.95      0.93        30
weighted avg       0.95      0.93      0.93        30


max_depth = 4
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.85      0.92        13
           2       0.78      1.00      0.88         7

    accuracy                           0.93        30
   macro avg       0.93      0.95      0.93        30
weighted avg       0.95      0.93      0.93        30


max_depth = 5
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.85      0.92        13
           2       0.78      1.00      0.88         7

    accuracy                           0.93        30
   macro avg       0.93      0.95      0.93        30
```

```
weighted avg       0.95      0.93      0.93        30
```

```
/home/abdullah/anaconda3/lib/python3.7/site-packages/sklearn/metrics/
classification.py:1437: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted samp
les.
  'precision', 'predicted', average, warn_for)
```

In [41]:

```
print(metrics.confusion_matrix(expected,predicted))
```

```
[[10  0  0]
 [ 0 11  2]
 [ 0  0  7]]
```

Problem 2.2

In [ ]:

```

```