P3_PartA_Violet

March 2, 2021

0.1 Project 3 - Part A

0.1.1 1.

Group Name : Violet Members : Yifu Deng, Jiangqiu Shen, Zhiyuan Lu

0.1.2 2-a

Here we suppose that the root node is {N} and the nodes in two way splits are {N1-1, N1-2} and {N2-1, N2-2, N2-3} repectively.

[12]:
```
print('GINI Index for each node')
print('GINI(N) = 1 - (100/210)^2 - (50/210)^2 - (60/210)^2')
gini_n = 1 - (100/210)**2 - (50/210)**2 - (60/210)**2
print('=',gini_n)
print('GINI(N1-1) = 1 - (56/68)^2 - (12/68)^2')
gini_n1_1 = 1 - (56/68)**2 - (12/68)**2
print('=',gini_n1_1)
print('GINI(N1-2) = 1 - (44/142)^2 - (38/142)^2 - (60/142)^2')
gini_n1_2 = 1 - (44/142)**2 - (38/142)**2 - (60/142)**2
print('=',gini_n1_2)
print('GINI(N2-1) = 1 - (62/80)^2 - (18/80)^2')
gini_n2_1 = 1 - (62/80)**2 - (18/80)**2 - 0
print('=', gini_n2_1)
print('GINI(N2-2) = 1 - (28/63)^2 - (11/63)^2 - (24/63)^2')
gini_n2_2 = 1 - (28/63)**2 - (11/63)**2 - (24/63)**2
print('=',gini_n2_2)
print('GINI(N2-3) = 1 - (10/67)^2 - (21/67)^2 - (36/67)^2')
gini_n2_3 = 1 - (10/67)**2 - (21/67)**2 - (36/67)**2
print('=',gini_n2_3)
print('\n\n')
gain_1 = gini_n - (2*gini_n1_1+3*gini_n1_2)/(2+3)
print('The gain in GINI Index for the first splits',gain_1)
gain_2 = gini_n - (2*gini_n2_1+3*gini_n2_2+3*gini_n2_3)/(2+3+3)
print('The gain in GINI Index for the second splits',gain_2)
```

```
GINI Index for each node
GINI(N) = 1 - (100/210)^2 - (50/210)^2 - (60/210)^2
= 0.634920634920635
GINI(N1-1) = 1 - (56/68)^2 - (12/68)^2
```

```
= 0.2906574394463668
GINI(N1-2) = 1 - (44/142)^2 - (38/142)^2 - (60/142)^2
= 0.6538385241023607
GINI(N2-1) = 1 - (62/80)^2 - (18/80)^2
= 0.34874999999999999
GINI(N2-2) = 1 - (28/63)^2 - (11/63)^2 - (24/63)^2
= 0.6268581506676745
GINI(N2-3) = 1 - (10/67)^2 - (21/67)^2 - (36/67)^2
= 0.5907774560035643



The gain in GINI Index for the first splits 0.12635454468067187
The gain in GINI Index for the second splits 0.09111978241892038
```

0.1.3 2-b

Based on selecting node with largest gain in GINI index, N1-1 and N1-2 are preferred.

0.1.4 2-C

[42]:
```python
import math
print('H(N) = - {P(Y=A)logP(Y=A) + P(Y=B)logP(Y=B) +P(Y=C)logP(Y=C)}')
h_n = -( (100/210)*math.log2(100/210) + (50/210)*math.log2(50/210) + (60/
 ↪210)*math.log2(60/210) )
print('Entropy before split is H(N)=', h_n)

print('\nFor the first split 1,')
p1 = 68/210
p2 = 142/210
print('P(X = N1-1) is ', p1,'  P(X=N1-2) is ', p2)
pA_1 = 56/68
pB_1 = 12/68
pC_1 = 0/68
print('P(Y=A|N1-1) is ',pA_1,'  P(Y=B|N1-1) is ',pB_1,'  P(Y=C|N1-1) is ',pC_1)
pA_2 = 44/142
pB_2 = 38/142
pC_2 = 60/142
print('P(Y=A|N1-2) is ',pA_2,'  P(Y=B|N1-2) is ',pB_2,'  P(Y=C|N1-2) is ',pC_2)
h_s1 = - ( p1*( pA_1*math.log2(pA_1)+pB_1*math.log2(pB_1)) + p2*( pA_2*math.
 ↪log2(pA_2)+pB_2*math.log2(pB_2)+pC_2*math.log2(pC_2) ) )
print('Entropy after split is H(N|S1) =',h_s1, '.  Information gain is␣
 ↪H(N)-H(N|S1) =', h_n-h_s1)

print('\nFor the split 2,')
_p1 = 80/210
_p2 = 63/210
_p3 = 67/210
```

```
print('P(X = N2-1) is ', _p1,'  P(X=N2-2) is ', _p2, '  P(X=N2-3) is ', _p3)
_pA_1 = 62/80
_pB_1 = 18/80
_pC_1 = 0/80
print('P(Y=A|N2-1) is ',_pA_1,'  P(Y=B|N2-1) is ',_pB_1,'  P(Y=C|N2-1) is␣
↪',_pC_1)
_pA_2 = 28/63
_pB_2 = 11/63
_pC_2 = 24/63
print('P(Y=A|N2-2) is ',_pA_2,'  P(Y=B|N2-2) is ',_pB_2,'  P(Y=C|N2-2) is␣
↪',_pC_2)
_pA_3 = 10/67
_pB_3 = 21/67
_pC_3 = 36/67
print('P(Y=A|N2-3) is ',_pA_3,'  P(Y=B|N2-3) is ',_pB_3,'  P(Y=C|N2-3) is␣
↪',_pC_3)
h_s2 =0
h_s2 += _p1*(_pA_1 * math.log2(_pA_1) + _pB_1*math.log2(_pB_1) )
h_s2 += _p2*( _pA_2*math.log2(_pA_2) + _pB_2*math.log2(_pB_2) + _pC_2*math.
↪log2(_pC_2) )
h_s2 += _p3*( _pA_3*math.log2(_pA_3) + _pB_3*math.log2(_pB_3) + _pC_3*math.
↪log2(_pC_3) )
h_s2 = -h_s2
print('Entropy after split is H(N|S2) =',h_s2, '.  Information gain is␣
↪H(N)-H(N|S2) =', h_n - h_s2)
```

```
H(N) = - {P(Y=A)logP(Y=A) + P(Y=B)logP(Y=B) +P(Y=C)logP(Y=C)}
Entropy before split is H(N)= 1.5190461643198376

For the first split 1,
P(X = N1-1) is  0.3238095238095238   P(X=N1-2) is  0.6761904761904762
P(Y=A|N1-1) is  0.8235294117647058   P(Y=B|N1-1) is  0.17647058823529413
P(Y=C|N1-1) is  0.0
P(Y=A|N1-2) is  0.30985915492957744   P(Y=B|N1-2) is  0.2676056338028169
P(Y=C|N1-2) is  0.4225352112676056
Entropy after split is H(N|S1) = 1.2710974574447609 .  Information gain is
H(N)-H(N|S1) = 0.247948706875076755

For the split 2,
P(X = N2-1) is  0.38095238095238093   P(X=N2-2) is  0.3   P(X=N2-3) is
0.319047619047619
P(Y=A|N2-1) is  0.775   P(Y=B|N2-1) is  0.225   P(Y=C|N2-1) is  0.0
P(Y=A|N2-2) is  0.4444444444444444   P(Y=B|N2-2) is  0.1746031746031746
P(Y=C|N2-2) is  0.38095238095238093
P(Y=A|N2-3) is  0.14925373134328357   P(Y=B|N2-3) is  0.31343283582089554
P(Y=C|N2-3) is  0.5373134328358209
1.1917047844910704
```

```
Entropy after split is H(N|S2) = 1.1917047844910704 .  Information gain is
H(N)-H(N|S2) = 0.3273413798287672
```

0.1.5 2-d

Based on selecting variable to maxmize information gain, the second way to split is preferred.

0.1.6 3-a

Fruit Data

Columns: Type, Weight, Height, Width

Type: 1 - apples 2 - oranges 3 - lemons

Weight (grams) 0 - if wt <= 179.42 1 - otherwise

Height (cm) 2 - if ht > 8.5 1 - if ht <= 8.5 && > 7.3 0 - otherwise

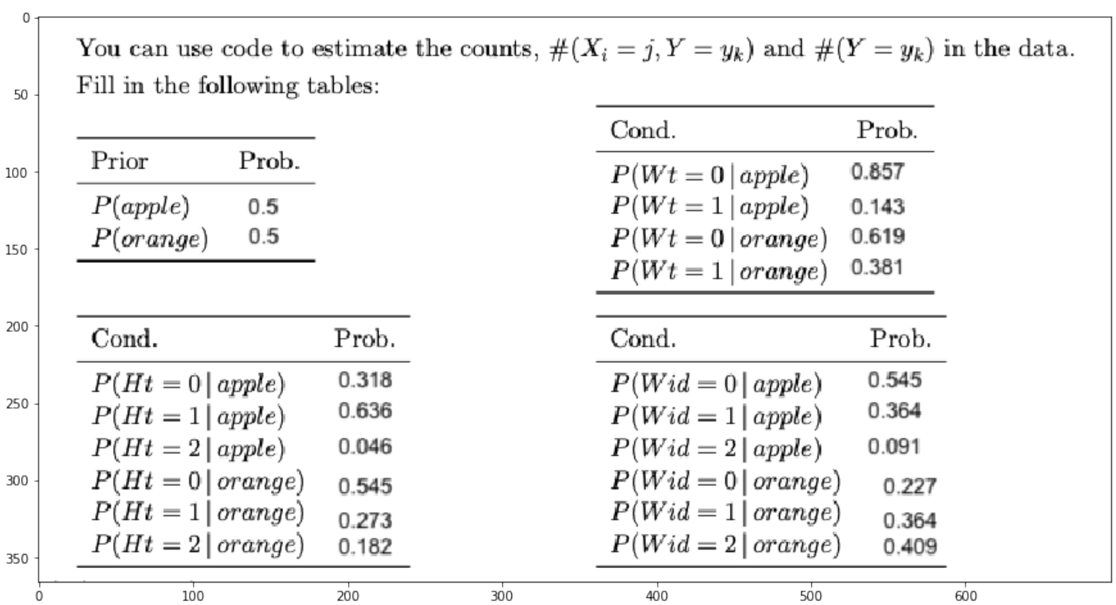
Width (cm) 2 - if width > 7.8 1 - if width <= 7.8 && >7.3 0 - otherwise

[23]:
```
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv("./data/fruit.txt",names=['T','Wt','Ht','Wid'])
data = data.loc[data['T']<3]
#print(data)
appl = data.loc[data['T']==1]
orag = data.loc[data['T']==2]

num_appl = len(appl)
num_orag = len(orag)
print('num_appl',num_appl,'num_orag',num_orag)

#Prior

p_appl = len(appl)/len(data)
p_orag = len(orag)/len(data)
print('p_appl',p_appl,'p_orag',p_orag)

# Wt
num_Wt0_appl = len(appl.loc[appl['Wt']==0])
num_Wt1_appl = len(appl.loc[appl['Wt']==1])
print('\nnum_Wt0_appl',num_Wt0_appl,'num_Wt1_appl',num_Wt1_appl)
num_Wt0_orag = len(orag.loc[orag['Wt']==0])
num_Wt1_orag = len(orag.loc[orag['Wt']==1])
print('\nnum_Wt0_orag',num_Wt0_orag,'num_Wt1_orag',num_Wt1_orag)

dom_Wt = 2

p_wt0_a  = (num_Wt0_appl + 1)/(num_appl + dom_Wt)
p_wt1_a  = (num_Wt1_appl + 1)/(num_appl + dom_Wt)
```

```python
p_wt0_o  = (num_Wt0_orag + 1)/(num_orag + dom_Wt)
p_wt1_o  = (num_Wt1_orag + 1)/(num_orag + dom_Wt)
print('\np_wt0_a', p_wt0_a ,'p_wt1_a',p_wt1_a,'p_wt0_o',p_wt0_o,'p_wt1_o',p_wt1_o)

# Ht
num_Ht0_appl = len(appl.loc[appl['Ht']==0])
num_Ht1_appl = len(appl.loc[appl['Ht']==1])
num_Ht2_appl = len(appl.loc[appl['Ht']==2])
num_Ht0_orag = len(orag.loc[orag['Ht']==0])
num_Ht1_orag = len(orag.loc[orag['Ht']==1])
num_Ht2_orag = len(orag.loc[orag['Ht']==2])
print('\nnum_Ht0_appl','num_Ht1_appl','num_Ht2_appl','num_Ht0_orag','num_Ht1_orag','num_Ht2_or
print(num_Ht0_appl,num_Ht1_appl,num_Ht2_appl,num_Ht0_orag,num_Ht1_orag,num_Ht2_orag)

dom_Ht = 3

p_h0_a  = (num_Ht0_appl+1)/(num_appl+dom_Ht)
p_h1_a  = (num_Ht1_appl+1)/(num_appl+dom_Ht)
p_h2_a  = (num_Ht2_appl+1)/(num_appl+dom_Ht)
p_h0_o  = (num_Ht0_orag+1)/(num_orag+dom_Ht)
p_h1_o  = (num_Ht1_orag+1)/(num_orag+dom_Ht)
p_h2_o  = (num_Ht2_orag+1)/(num_orag+dom_Ht)
print('\np_h0_a','p_h1_a','p_h2_a' ,'p_h0_o' ,'p_h1_o' ,'p_h2_o')
print(p_h0_a,p_h1_a,p_h2_a ,p_h0_o ,p_h1_o ,p_h2_o)

# Wid
num_Wid0_appl = len(appl.loc[appl['Wid']==0])
num_Wid1_appl = len(appl.loc[appl['Wid']==1])
num_Wid2_appl = len(appl.loc[appl['Wid']==2])
num_Wid0_orag = len(orag.loc[orag['Wid']==0])
num_Wid1_orag = len(orag.loc[orag['Wid']==1])
num_Wid2_orag = len(orag.loc[orag['Wid']==2])
print('\nnum_Wid0_appl','num_Wid1_appl','num_Wid2_appl','num_Wid0_orag','num_Wid1_orag','num_W
print(num_Wid0_appl,num_Wid1_appl,num_Wid2_appl,num_Wid0_orag,num_Wid1_orag,num_Wid2_orag)

dom_Wid = 3

p_Wid0_a  = (num_Wid0_appl+1)/(num_appl+dom_Wid)
p_Wid1_a  = (num_Wid1_appl+1)/(num_appl+dom_Wid)
p_Wid2_a  = (num_Wid2_appl+1)/(num_appl+dom_Wid)
p_Wid0_o  = (num_Wid0_orag+1)/(num_orag+dom_Wid)
p_Wid1_o  = (num_Wid1_orag+1)/(num_orag+dom_Wid)
p_Wid2_o  = (num_Wid2_orag+1)/(num_orag+dom_Wid)
print('\np_Wid0_a','p_Wid1_a','p_Wid2_a' ,'p_Wid0_o' ,'p_Wid1_o' ,'p_Wid2_o')
print(p_Wid0_a,p_Wid1_a,p_Wid2_a ,p_Wid0_o ,p_Wid1_o ,p_Wid2_o)
im = plt.imread('Q3.png')
```

```
plt.figure(figsize=(15,9))
plt.imshow(im)
plt.show()
```

```
num_appl 19 num_orag 19
p_appl 0.5 p_orag 0.5

num_Wt0_appl 17 num_Wt1_appl 2

num_Wt0_orag 12 num_Wt1_orag 7

p_wt0_a 0.8571428571428571 p_wt1_a 0.14285714285714285 p_wt0_o
0.6190476190476191 p_wt1_o 0.38095238095238093

num_Ht0_appl num_Ht1_appl num_Ht2_appl num_Ht0_orag num_Ht1_orag num_Ht2_orag
6 13 0 11 5 3

p_h0_a p_h1_a p_h2_a p_h0_o p_h1_o p_h2_o
0.3181818181818182 0.6363636363636364 0.045454545454545456 0.5454545454545454
0.2727272727272727 0.18181818181818182

num_Wid0_appl num_Wid1_appl num_Wid2_appl num_Wid0_orag num_Wid1_orag
num_Wid2_orag
11 7 1 4 7 8

p_Wid0_a p_Wid1_a p_Wid2_a p_Wid0_o p_Wid1_o p_Wid2_o
0.5454545454545454 0.36363636363636365 0.09090909090909091 0.22727272727272727
0.36363636363636365 0.4090909090909091
```



You can use code to estimate the counts, $\#(X_i = j, Y = y_k)$ and $\#(Y = y_k)$ in the data. Fill in the following tables:

Prior	Prob.
$P(apple)$	0.5
$P(orange)$	0.5

Cond.	Prob.
$P(Wt = 0 \mid apple)$	0.857
$P(Wt = 1 \mid apple)$	0.143
$P(Wt = 0 \mid orange)$	0.619
$P(Wt = 1 \mid orange)$	0.381

Cond.	Prob.
$P(Ht = 0 \mid apple)$	0.318
$P(Ht = 1 \mid apple)$	0.636
$P(Ht = 2 \mid apple)$	0.046
$P(Ht = 0 \mid orange)$	0.545
$P(Ht = 1 \mid orange)$	0.273
$P(Ht = 2 \mid orange)$	0.182

Cond.	Prob.
$P(Wid = 0 \mid apple)$	0.545
$P(Wid = 1 \mid apple)$	0.364
$P(Wid = 2 \mid apple)$	0.091
$P(Wid = 0 \mid orange)$	0.227
$P(Wid = 1 \mid orange)$	0.364
$P(Wid = 2 \mid orange)$	0.409

0.1.7 3-b

[10]:
```
# Sample 1
p_appl_Wt1_Ht1_Wid0 = p_wt1_a * p_h1_a * p_Wid0_a * p_appl
p_orag_Wt1_Ht1_Wid0 = p_wt1_o * p_h1_o * p_Wid0_o * p_orag
print('p_appl_Wt1_Ht1_Wid0',p_appl_Wt1_Ht1_Wid0,'p_orag_Wt1_Ht1_Wid0',␣
 ↪p_orag_Wt1_Ht1_Wid0)

# Sample 2
p_appl_Wt0_Ht0_Wid1 = p_wt0_a * p_h0_a * p_Wid1_a * p_appl
p_orag_Wt0_Ht0_Wid1 = p_wt0_o * p_h0_o * p_Wid1_o * p_orag
print('p_appl_Wt0_Ht0_Wid1',p_appl_Wt0_Ht0_Wid1,␣
 ↪'p_orag_Wt0_Ht0_Wid1',p_orag_Wt0_Ht0_Wid1)

# Sample 3
p_appl_Wt0_Ht0_Wid1 = p_wt0_a * p_h0_a * p_Wid1_a * p_appl
p_orag_Wt0_Ht0_Wid1 = p_wt0_o * p_h0_o * p_Wid1_o * p_orag
print('p_appl_Wt0_Ht0_Wid1',p_appl_Wt0_Ht0_Wid1,␣
 ↪'p_orag_Wt0_Ht0_Wid1',p_orag_Wt0_Ht0_Wid1)

# Sample 4
p_appl_Wt1_Ht0_Wid0 = p_wt1_a * p_h0_a * p_Wid0_a * p_appl
p_orag_Wt1_Ht0_Wid0 = p_wt1_o * p_h0_o * p_Wid0_o * p_orag
print('p_appl_Wt1_Ht0_Wid0',p_appl_Wt1_Ht0_Wid0,␣
 ↪'p_orag_Wt1_Ht0_Wid0',p_orag_Wt1_Ht0_Wid0)
```

```
p_appl_Wt1_Ht1_Wid0 0.024793388429752066 p_orag_Wt1_Ht1_Wid0 0.011806375442739077
p_appl_Wt0_Ht0_Wid1 0.04958677685950413 p_orag_Wt0_Ht0_Wid1 0.061393152302243216
p_appl_Wt0_Ht0_Wid1 0.04958677685950413 p_orag_Wt0_Ht0_Wid1 0.061393152302243216
p_appl_Wt1_Ht0_Wid0 0.012396694214876033 p_orag_Wt1_Ht0_Wid0 0.023612750885478154
```

Thus
Sample-1 is predicted as apple
Sample-2 is predicted as orange
Sample-3 is predicted as orange
Sample-4 is predicted as orange

0.1.8 3-c

Sample1 - TP
Sample2 - FN
Sample3 - TP
Sample4 - TP