19.4 : Exercises :

Q1. (a) False.
(b) True.

Q2. $\pi = 100\%$
$D =$

	Age	Car	Risk
x_1	25	Sports	L
x_2	20	Vintage	H
x_3	25	Sports	L
x_4	45	SUV	H
x_5	20	Sports	H
x_6	25	SUV	H

For Numeric Attribute : Car :

→ Sort on Age attribute : (20, 20, 25, 25, 25, 45)
→ Get the set of midpoints : (22.5, 35)

- $\hat{P}(C_L) = \frac{2}{6}$
- $\hat{P}(C_H) = \frac{4}{6}$

$\therefore H(D) = -(\frac{2}{6}\log\frac{2}{6}) - \frac{4}{6}\log\frac{4}{6}$
$= 0.276$

First Consider split point (22.5)

- $N_{VL} = 0$, $N_{VH} = 2$
- $\hat{P}(C_L \mid D_Y) = \frac{N_{VL}}{N_{VL}+N_{VH}} = \frac{0}{0+2} = 0$
- $\hat{P}(C_H \mid D_Y) = \frac{N_{VH}}{N_{VL}+N_{VH}} = \frac{2}{0+2} = 1$
- $\hat{P}(C_L \mid D_N) = \frac{n_L - N_{VL}}{(n_L - N_{VL})+(n_H - N_{VH})} = \frac{(2-0)}{(2-0)+(4-2)} = \frac{1}{2}$
- $\hat{P}(C_H \mid D_N) = \frac{n_H - N_{VH}}{(n_H - N_{VH})+(n_L - N_{VL})} = \frac{(4-2)}{(4-2)+(2-0)} = \frac{1}{2}$

Page :
Date :

$H(D_Y) = -(0 \log 0 + 1 \log 1) = 0$

$H(D_N) = -(\frac{1}{2} \log \frac{1}{2} + \frac{1}{2} \log \frac{1}{2}) = 0.301$

Split Point Entropy $H(D_Y, D_N) = \frac{n_{D_Y}}{n_D} H(D_Y) + \frac{n_{D_N}}{n_D} H(D_N) = 0.201$

Information Gain $Gain = H(D) - H(D_Y, D_N) = 0.276 - 0.201 = 0.075$

Consider Split Point (Age ≤ 35):

- $N_{VL} = 2$, $N_{VH} = 3$
- $\hat{P}(C_L | D_Y) = \frac{N_{VL}}{N_{VL} + N_{VH}} = \frac{2}{2+3} = \frac{2}{5}$
- $\hat{P}(C_H | D_Y) = \frac{N_{VH}}{N_{VH} + N_{VH}} = \frac{3}{5}$
- $\hat{P}(C_L | D_N) = \frac{n_L - N_{VL}}{(n_L - N_{VL}) + (n_H - N_{VH})} = \frac{(2-2)}{(2-2)+(4-3)} = 0$
- $\hat{P}(C_H | D_N) = \frac{n_H - N_{VH}}{(n_L - N_{VL}) + (n_H - N_{VH})} = \frac{(4-3)}{(2-2)+(4-3)} = 1$
- $H(D_Y) = -(\frac{2}{5} \log(\frac{2}{5}) + \frac{3}{5} \log(\frac{3}{5})) = 0.292$
- $H(D_N) = 0$
- $H(D_Y, D_N) = (\frac{5}{6})(0.292) = 0.243$

$Gain = H(D) - H(D_Y, D_N) = 0.276 - 0.243 = 0.033$

* For Categorical Attribute "Car": dom(Car) = [Sports, Vintage, SUV]
⇒ Possible splitpoints :

① $X \in \{Sports\}$ ② $X \in \{Vintage\}$ ③ $X \in \{SUV\}$

For ①,

- $\hat{P}(C_L \mid D_Y) = \frac{n_{sports\,L}}{n_{sports\,L} + n_{sports\,H}} = \frac{2}{2+1} = \frac{2}{3}$
- $\hat{P}(C_H \mid D_Y) = 1 - \frac{2}{3} = \frac{1}{3}$
- $\hat{P}(C_L \mid D_N) = \frac{n_{SUV\,L} + n_{vintage\,L}}{n_{SUV\,L} + n_{vintage\,L} + n_{SUV\,H} + n_{vintage\,H}}$

$= \frac{0+0}{0+0+2+3} = 0$

- $\hat{P}(C_H \mid D_N) = \frac{n_{SUV\,H} + n_{vintage\,H}}{\text{\\ \\}} = \frac{2+3}{0+0+2+3} = 1$

- $H(D_Y) = -\frac{1}{3}\log_2(\frac{1}{3}) - \frac{2}{3}\log_2(\frac{2}{3}) = 0.276$
- $H(D_N) = 0$
- $H(D_Y, D_N) = \frac{n_Y}{n} H(D_Y) + \frac{n_N}{n} H(D_N) = 0.138$ ($\frac{3}{6}$, 0.276, 0)
- Gain = $H(D) - H(D_Y, D_N) = 0.276 - 0.138 = 0.138$

For (2),

- $\hat{P}(C_L \mid D_Y) = \frac{0}{0+1} = 0$
- $\hat{P}(C_H \mid D_Y) = \frac{1}{0+1} = 1$
- $\hat{P}(C_L \mid D_N) = \frac{2+0}{(2+0)+(1+2)} = \frac{2}{5}$
- $\hat{P}(C_H \mid D_N) = \frac{1+2}{(2+0)+(1+2)} = \frac{3}{5}$
- $H(D_Y) = 0$
- $H(D_N) = -\log(2/5) \times 2/5 - \log(3/5) \times 3/5 = 0.292$
- $H(D_Y, D_N) = \frac{1}{6} \times 0 + \frac{5}{6} \times 0.292 = 0.244$
- Gain $= 0.276 - 0.244 = 0.032$

For (6),

- $\hat{P}(C_L \mid D_Y) = \frac{0}{0+2} = 0$
- $\hat{P}(C_H \mid D_Y) = \frac{2}{0+2} = 1$
- $\hat{P}(C_L \mid D_N) = \frac{(2+0)}{(2+0)+(1+1)} = \frac{2}{4} = \frac{1}{2}$
- $\hat{P}(C_H \mid D_N) = 1/2$
- $H(D_Y) = 0$, $H(D_N) = 0.301$
- $H(D_Y, D_N) = \frac{2}{6} \times 0 + \frac{4}{6} \times 0.301 = 0.201$
- Gain $= 0.075$

Sheet 6_Decision_Trees

April 15, 2019

1 Sheet#6 Decision Trees

Question 2: Coding the Decision Trees

In [1]:
```
#imports cell
from os import listdir
from PIL import Image as PImage
import matplotlib.pyplot as plt
import numpy as np
import math
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

a. Use the Face-data set we used in Assignment 1. Use the 50-50 Split for train-test split.

In [2]:
```
#Utility Method to loadImages into list from a given path parameter.
def loadImages(path):
    # return array of images
    foldersList = listdir(path)
    loadedImages = []
    for folder in foldersList :
        imagesList = listdir(path+folder)
        for image in imagesList:
            img = PImage.open(path +folder+'/'+ image)
            loadedImages.append(img)
    return loadedImages
```

In [3]:
```
#Loading our Faces dataset.
path = "../Projects/Face-Recoginition/orl_faces/"
imgs = loadImages(path)
#converting the images list into data matrix.
dataMatrix = np.arange(4121600).reshape(400,10304)
j=0
label = []
for i in range(0,400) :
    if(i%10 == 0):
        j = j+1
    dataMatrix[i] = np.array(imgs[i]).flatten()     #flatten() method is used to unrol
    label.append(j)
```

```
In [4]: # (50 - 50) split is used here.
        #Even instances for test set.
        #Odd instances for train set.
        trainSet=np.arange(200*10304).reshape(200,10304)
        testSet=np.arange(200*10304).reshape(200,10304)
        trainLabel=[]
        testLabel=[]
        j,k=0,0
        for i in range(0,400):
            if(i%2==0):
                testSet[j]=dataMatrix[i]
                testLabel.append(label[i])
                j+=1
            else:
                trainSet[k]=dataMatrix[i]
                trainLabel.append(label[i])
                k+=1
```

b. Implement your Decision Trees Classifier. Use Information Gain as a splitting measure.

```
In [7]: class TreeNode(object):
            def __init__(self):
                self.left = None
                self.right = None
                self.data = None
                self.index = None
        root = TreeNode()
```

```
In [ ]: def eval_numeric_split(dataset,k,label,attr):
            attr = np.sort(attr)
            M = {}
            ni = np.zeros(k)
            for j in range(attr.shape[0]-1):
                ni[label[j]-1] +=1
```

```
In [6]: def DecisionTree(dataset,label,node,k,criterion='gain', purity_threshold= None, max_lea
            data_size = dataset.shape[0]                #n
            attr_size = dataset.shape[1]                #d
            data_purity = 0
            max_purity = 0
            max_purity_class = 1
            best_split_point = [0,0]
            best_score = 0
            num_classes = np.unique(label).shape[0]
            class_size = np.zeros(num_classes)          #n_i vector initalize.
            data_yes = {}
            label_yes = {}
            data_no = {}
```

```
        label_no = {}

        for i in range(data_size):
            class_size[label[i]-1] += 1;       #Here the -1 is for that classes start from
        for i in range(num_classes):
            data_purity = class_size[i]/data_size
            if(data_purity > max_purity):
                max_purity = data_purity
                max_purity_class = i+1

        #stopping condition.
        if(data_size <= max_leaf_nodes or data_purity >= purity_threshold):
            #create leaf node with max_purity_class as label.
                node.data = dataset
                node.index = max_purity_class
            return

        #Do every Recursive step.
        for i in range(attr_size):
            split_point, score = eval_numeric_split(dataset,k,label,dataset[:,i])
            if (score > best_score):
                best_split_point[0] = i
                best_split_point[1] = split_point
                best_score = score

        #partition D into D Y and D N using split point.
        for i in range(data_size):
            if(dataset[i,best_split_point[0]] <= best_split_point[1]):
                data_yes.append(dataset[i])
                label_yes.append(label[i])
            else:
                data_no.append(dataset[i])
                label_no.append(label[i])

        #create internal node
        node.data = best_split_point[1]
        node.index = best_split_point[0]
        node.left = TreeNode()
        node.right = TreeNode()
        DecisionTree(np.array(data_yes), np.array(label_yes),node.left)
        DecisionTree(np.array(data_no), np.array(label_no),node.right)
```

In [ ]:

In [ ]:

In [ ]: