Jose Luis Vargas
57291630

Problem 1

a) $P(y=-1) = \frac{6}{10} = \frac{3}{5} \rightarrow H(y) = H(3/5) = \frac{3}{5}\log(5/3) + \frac{2}{5}\log(\frac{5}{2})$

$= 0.44 + 0.52$

$\boxed{H(y) = 0.97}$

b) Calculate the information gain for each feature x_i. Which feature should I split on first.

For X_1

When $X_1 = 0$ When $X_1 = 1$

$P(y=1) = \frac{1}{4}$ $P(y=1) = \frac{3}{6}$ $P(X_1=0) = \frac{4}{10}$ $P(X_1=1) = \frac{6}{10}$

$IG(X_1) = H(y) - H(x_1) = H(y) - (H(x_1=0) + H(x_1=1)) = H(y) - (\frac{2}{5}H(\frac{1}{4}) + \frac{3}{5}H(\frac{3}{6}))$

$0.97 - \left[\left(\frac{2}{5}\left(\frac{1}{4}\log(4) + \frac{3}{4}\log(4/3)\right)\right) + \left(\frac{3}{5}\right)\left(\frac{3}{6}\log\frac{6}{3} + \frac{3}{6}\log 2\right)\right]$

$\frac{2}{5}(.5 + .31) + \frac{3}{5}(.5 + .5)$

$0.97 - \left[\frac{2}{5}(.81) + \frac{3}{5}\right]$

$0.97 - 0.92 = \boxed{0.0416}$

For X_2

When $X_2 = 0$ When $X_2 = 1$

$P(y=1) = 4/5$ $P(y=1) = 0/5$ $P(X_2=0) = \frac{1}{2}$ $P(X_2=1) = \frac{1}{2}$

$IG(X_2) = H(y) - H(X_2) = H(y) - (H(X_2=0) + H(X_2=1))$

$= 0.97 - \left[\frac{1}{2}\left(\frac{4}{5}\log(5/4) + \frac{1}{5}\log(5)\right) + 0\right]$

$= 0.97 - 0.36$

$\boxed{0.61}$

For X_3

where $X_3 = 0$ $\quad X_3 = 1$

$P(y=1) = \frac{1}{3}$ $\quad p(y=1) = \frac{3}{7}$ $\quad p(x_3=0) = \frac{3}{10}$ $\quad p(x_3=1) = \frac{7}{10}$

$$IG(X_3) = H(y) - H(x_3) = H(y) - \left(\frac{3}{10} H\left(\frac{1}{3}\right) + \frac{7}{10} H\left(\frac{3}{7}\right)\right.$$

$$= 0.97 - \left[\frac{3}{10}\left(\frac{1}{3}\log(3) + \frac{2}{3}\log(3/2)\right) + \frac{7}{10}\left(\frac{3}{7}\log(7/3) + \frac{4}{7}\log(7/4)\right)\right]$$

$$0.97 - \left[\frac{3}{10}(.52 + .389) + \frac{7}{10}(.52 + .46)\right]$$

$$= \boxed{.0058}$$

For X_4

where $X_4 = 0$ $\quad X_4 = 1$

$p(y=1) = \frac{2}{3}$ $\quad p(y=1) = 2/7$ $\quad p(x=0) = 3/10$ $\quad p(x=1) = 7/10$

$$IG(X_4) = Hy - H(X_4) = H(y) - \left(\frac{3}{10} H\left(\frac{2}{3}\right) + \frac{7}{10} H\left(\frac{2}{7}\right)\right)$$

$$= .97 - \left[\frac{3}{10}\left(\frac{2}{3}\log\left(\frac{3}{2}\right) + \frac{1}{3}\log(3)\right) + \frac{7}{10}\left(\frac{2}{7}\log(7/2) + \frac{5}{7}\log(7/5)\right)\right]$$

$$\frac{3}{10}(0.3899 + .528) + \frac{7}{10}(.516 + .346)$$

$$.97 - .8787 = \underline{0.0913}$$

For X_5

where $X_5 = 0$ $\quad X_5 = 1$

$p(y=1) = 3/7$ $\quad p(y=1) = \frac{1}{3}$ $\quad p(x=0) = \frac{7}{10}$ $\quad p(x=1) = \frac{3}{10}$

$$IG(X_5) = H(y) - H(X_5) = H(y) - \left(\frac{7}{10}(H(3/7)) + \frac{3}{10}(H(\tfrac{1}{3}))\right.$$

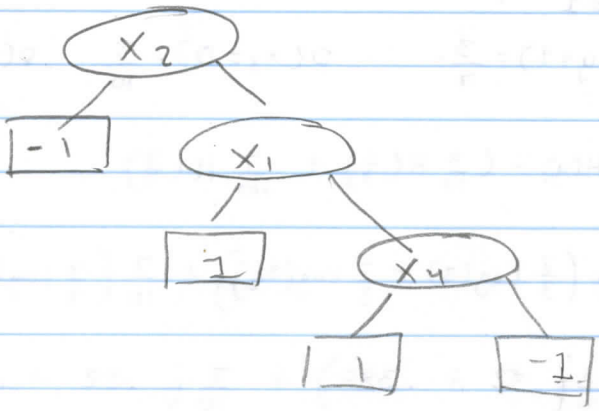
$$= .97 - \left[\frac{7}{10}\left(\frac{3}{7}\log(7/3) + \frac{4}{7}\log(7/4)\right) + \frac{3}{10}\left(\frac{1}{3}\log(3) + \frac{2}{3}\log(3/2)\right)\right]$$

$$= .97 - \left[\frac{7}{10}(.5238 + .4613) + \frac{3}{10}(.5283 + .3899)\right]$$

$$.96503$$

Split should occur on X_2 $\quad \underline{.00497}$

c)



First we split Feature 1

The right part is

X_1	X_2	X_3	X_4	X_5	y
0	0	1	1	0	-1
1	0	1	1	1	1
0	0	1	0	0	1
1	0	0	0	0	1
1	0	1	1	0	1

left part will allway be -1

Then we split Feature 1

where the left data will be

X_1	X_2	X_3	X_4	X_5	y
1	0	1	1	1	1
1	0	0	0	0	1
1	0	1	1	0	1

where right part is

X_1	X_2	X_3	X_4	X_5	y
0	0	1	1	0	-1
0	0	1	0	0	1

```
import numpy as np
import matplotlib.pyplot as plt
import mltools as ml
import mltools.dtree as dt
np.random.seed(0)
```

Problem 2A

```
X_train = np.genfromtxt("Data/X_train.txt",delimiter=None)
Y_train = np.genfromtxt("Data/Y_train.txt",delimiter=None)
#Xt,Xv,Yt,Yv = ml.splitData(X_train,Y_train,0.80)
Xt = X_train[:10000,:]
Yt = Y_train[:10000]
Xv = X_train[10001:20001,:]
Yv = Y_train[10001:20001]
```

Problem 2B

```
learner = dt.treeClassify(Xt,Yt, maxDepth=50)
print("Error Rate for Train Data: {}".format(learner.err(Xt,Yt)))
print("Error Rate for Validation Data: {}".format(learner.err(Xv,Yv)))
```

```
Error Rate for Train Data: 0.0047
Error Rate for Validation Data: 0.3816
```

Problem 2C

```
##d=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
#e=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
#r=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
for i in range(16):
    learner = dt.treeClassify(Xt,Yt, maxDepth=i)

    print("Depth = {}".format(i))
    print("Error Rate for Train Data: {}".format(learner.err(Xt,Yt) ))
    print("Error Rate for Validation Data: {}".format(learner.err(Xv,Yv)))
    print("")
```

```
Depth = 0
Error Rate for Train Data: 0.3418
Error Rate for Validation Data: 0.342

Depth = 1
Error Rate for Train Data: 0.3418
Error Rate for Validation Data: 0.342

Depth = 2
Error Rate for Train Data: 0.3223
Error Rate for Validation Data: 0.3191

Depth = 3
Error Rate for Train Data: 0.3133
Error Rate for Validation Data: 0.3126

Depth = 4
Error Rate for Train Data: 0.3105
Error Rate for Validation Data: 0.3152

Depth = 5
Error Rate for Train Data: 0.3008
Error Rate for Validation Data: 0.3102

Depth = 6
Error Rate for Train Data: 0.2949
Error Rate for Validation Data: 0.3103

Depth = 7
Error Rate for Train Data: 0.2872
Error Rate for Validation Data: 0.3118

Depth = 8
Error Rate for Train Data: 0.277
Error Rate for Validation Data: 0.313

Depth = 9
Error Rate for Train Data: 0.2632
Error Rate for Validation Data: 0.3186

Depth = 10
Error Rate for Train Data: 0.2455
Error Rate for Validation Data: 0.3243

Depth = 11
Error Rate for Train Data: 0.2309
Error Rate for Validation Data: 0.3258

Depth = 12
Error Rate for Train Data: 0.2088
Error Rate for Validation Data: 0.3344

Depth = 13
Error Rate for Train Data: 0.1923
Error Rate for Validation Data: 0.3431

Depth = 14
Error Rate for Train Data: 0.1648
Error Rate for Validation Data: 0.3439

Depth = 15
Error Rate for Train Data: 0.1454
Error Rate for Validation Data: 0.3575
```

Complexity starts to increase towards Depth = 15. Also, we start to see that overfitting begins to occur when depth = 5. By depth = 15 the difference between the validation and training error is greater than the previous comparissons.

Problem 2D

```
for j in range(2,13):
    learner = dt.treeClassify(Xt,Yt, maxDepth=50, minLeaf = 2**j)
    #minLeaf is used to control complexity.
    print("minleaf = 2^{}".format(j))
    print("Error Rate for Train Data: {}".format(learner.err(Xt,Yt)))
    print("Error Rate for Validation Data: {}".format(learner.err(Xv,Yv)))
    print("")
```

```
minleaf = 2^2
Error Rate for Train Data: 0.0964
Error Rate for Validation Data: 0.3794

minleaf = 2^3
Error Rate for Train Data: 0.1692
Error Rate for Validation Data: 0.3755

minleaf = 2^4
Error Rate for Train Data: 0.2256
Error Rate for Validation Data: 0.3546

minleaf = 2^5
Error Rate for Train Data: 0.2637
Error Rate for Validation Data: 0.3335

minleaf = 2^6
Error Rate for Train Data: 0.2899
Error Rate for Validation Data: 0.3276

minleaf = 2^7
Error Rate for Train Data: 0.3012
Error Rate for Validation Data: 0.3119

minleaf = 2^8
Error Rate for Train Data: 0.3085
Error Rate for Validation Data: 0.3172

minleaf = 2^9
Error Rate for Train Data: 0.3135
Error Rate for Validation Data: 0.3127

minleaf = 2^10
Error Rate for Train Data: 0.3223
Error Rate for Validation Data: 0.3191
```

```
minleaf = 2^11
Error Rate for Train Data: 0.3418
Error Rate for Validation Data: 0.342

minleaf = 2^12
Error Rate for Train Data: 0.3418
Error Rate for Validation Data: 0.342
```

As minLeaf grows, the complexity starts to decrease. Also when the

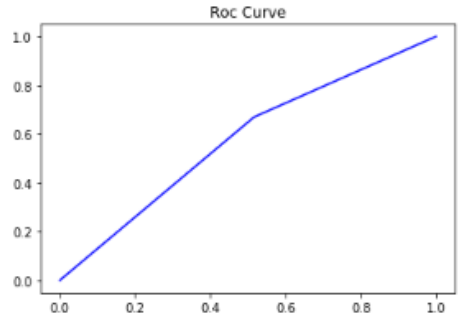
minleaf is 4 and below the model seems to be overfitting. I would select minleaf 2^7 as the complexity control value. This is because, it does,not overfit, compared to the other values and it doesnt underfit.

Problem 2F

```
#c = learner.roc(Xv, Yv)
fpr,tpr,tnr = learner.roc(Xt, Yt)

#the roc function returns three array
#1st array is the false positive rate
#2nd array is the true positive rate
#3rd array is the true negative rate
#for this implementation we would need

plt.title("Roc Curve")
plt.plot(fpr,tpr, 'b-')
plt.show()
```




```
print("Area under the curve = {}".format(learner.auc(Xv, Yv)))
```

```
Area under the curve = 0.599570735349
```

Problem 2G

```
lr = dt.treeClassify(Xt, Yt,maxDepth=50,minLeaf = 128)
Xte = np.genfromtxt("Data/X_test.txt")
Ypred = learner.predictSoft( Xte )
# Now output a file with two columns, a row ID and a confidence in class 1:
np.savetxt('Yhat_dtree.txt',
np.vstack( (np.arange(len(Ypred)) , Ypred[:,1]) ).T,
'%d, %.2f',header='ID,Prob1',comments='',delimiter=',');
print("The AUC of the validation data: {}".format(learner.auc(Xv, Yv)))
print("The AUC of my model: {}".format(lr.auc(Xv,Yv)))
```

```
The AUC of the validation data: 0.599570735349
The AUC of my model: 0.647156454967
```

Problem 3A ¶

```
X_train = np.genfromtxt("Data/X_train.txt",delimiter=None)
Y_train = np.genfromtxt("Data/Y_train.txt",delimiter=None)
Xtest = np.genfromtxt("Data/X_test.txt",delimiter=None)
#Xt,Xv,Yt,Yv = ml.splitData(X_train,Y_train,0.80)
Xt = X_train[:10000,:]
Yt = Y_train[:10000]
Xv = X_train[10001:20001,:] #This is the validation split.
Yv = Y_train[10001:20001]
```

```
nBag = 25
m,n = Xt.shape
d = [1,5,10,25]
mTest = Xv.shape[0]


ensemble = [None]*25
#the first for loop to build the assembler
for i in range(nBag):
    #this is for training data
    Xi,Yi = ml.bootstrapData(Xt,Yt, m)
    ensemble[i] = dt.treeClassify(Xi,Yi, maxDepth=50, minLeaf = 4)


#for loop to compute the prediction
Tpredict = np.zeros((m, nBag))
Vpredict = np.zeros((mTest, nBag))
for j in range(nBag):
    Tpredict[:,j] = ensemble[j].predict(Xt)
    Vpredict[:,j] = ensemble[j].predict(Xv)
#Tpredict = np.mean(Tpredict, axis=1)

trainErr =  [0,0,0,0]
valErr = [0,0,0,0]
for j,i in enumerate(d):
    trainErr[j] = (Tpredict[:,0:i].sum(axis =1)).mean()
    valErr[j] = (Vpredict[:,0:i].sum(axis =1)).mean()

plt.title("Validation Error vs Training Error")
plt.semilogy(d, trainErr,'r',d, valErr, 'b')
plt.show()
```