Name: Hootan Hosseinzadeganbushehri

[redacted]

UCI Email: hootanh@uci.edu

Problem 1 Part 1:

In [127]:
```python
from __future__ import division
import numpy as np
import mltools as ml
import matplotlib.pyplot as plt
import IPython.display


np.random.seed(0)
dataSet = np.array([[0,0,1,1,0,-1], [1,1,0,1,0,-1], [0,1,1,1,1,-1], [1,1,1,1,0,-1], [0,1,0,0,0,-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], [1,1,1,1,1,-1]])
temp_x = dataSet[:,0:-1]
temp_y = dataSet[:,-1]
temp_mean = np.mean(temp_y > 0)
resultEntropy = -(temp_mean * np.log2(temp_mean) + (1 - temp_mean) * np.log2(1 - temp_mean))
print()
print('\033[1m' + "Entropy H(y) =", resultEntropy)
```

```
Entropy H(y) = 0.9709505944546686
```

Problem 1 Part 2:

In [128]:
```python
a_list = []
for i in range(5):
    idx = temp_x[:,i] > 0
    if ((idx.sum() == 0) or ((1 - idx).sum() == 0)):
        l[i] = 0
        continue
    temp1 = 1e-12 + np.mean( temp_y[idx] > 0 )
    temp2 = 1e-12 + np.mean(temp_y[idx == False] > 0)
    temp3 = 1e-12 + np.mean(idx)
    a_list.append(resultEntropy + temp3 * (temp1 * np.log2(temp1) + (1-temp1) * np.log2(1 - temp1)) + (1 - temp3) * (temp2 * np.
log2(temp2) + (1 - temp2) * np.log2(1 - temp2)))
print('\033[1m' + "\nInformation Gain:\n")
for x in range(5):
    print('\033[1m' + "\tFeature", str(x+1) + ":", a_list[x])
```

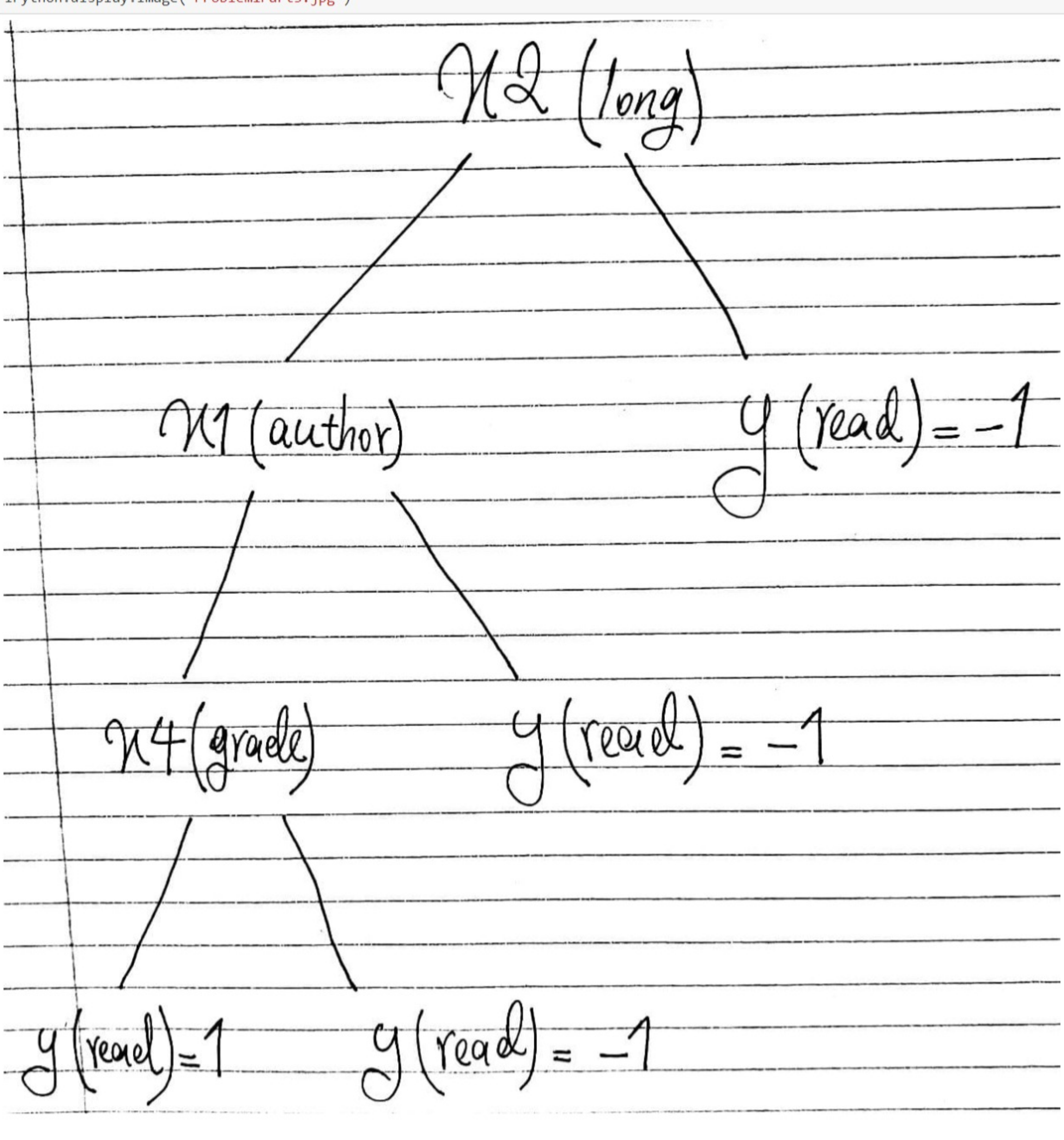
```
Information Gain:

	Feature 1: 0.046439344670192784
	Feature 2: 0.6099865469920565
	Feature 3: 0.005802149013688251
	Feature 4: 0.09127744624110995
	Feature 5: 0.005802149013822033
```

Feature 2 with 0.6099865469920565 information gain should be splitted on for the root node of the decision tree since it has the most information gain compare to other features.

Problem 1 Part 3:

In [129]:
```python
IPython.display.Image("Problem1Part3.jpg")
```

Out[129]:



Problem 2 Part 1:

In [130]:
```python
X = np.genfromtxt('data/X_train.txt', delimiter=',')
Y = np.genfromtxt('data/Y_train.txt', delimiter=',')
X,Y = ml.shuffleData(X,Y)
X = X[:,:41]
for k in range(5):
    print('\033[1m' + "\nFeature", str(k+1) + ":\n",)
    print('\033[1m' + "\tMinimum:", np.min(X[:,k]))
    print('\033[1m' + "\tMaximum:", np.max(X[:,k]))
    print('\033[1m' + "\tMean:", np.mean(X[:,k]))
    print('\033[1m' + "\tVariance:", np.var(X[:,k]))
```

```
Feature 1:

	Minimum: 0.0
	Maximum: 110285.0
	Mean: 1321.1174134446987
	Variance: 6747189.595085322

Feature 2:

	Minimum: 0.0
	Maximum: 35.0
	Mean: 6.5916745251246125
	Variance: 34.70690630279573

Feature 3:

	Minimum: 0.0
	Maximum: 51536.0
	Mean: 1152.273237235619
	Variance: 5376518.288798102

Feature 4:

	Minimum: 0.0
	Maximum: 21768.0
	Mean: 234.8262548834703
	Variance: 260120.83053297663

Feature 5:

	Minimum: 0.0
	Maximum: 27210.0
	Mean: 289.75871211100633
	Variance: 406615.8651128233
```

Problem 2 Part 2:

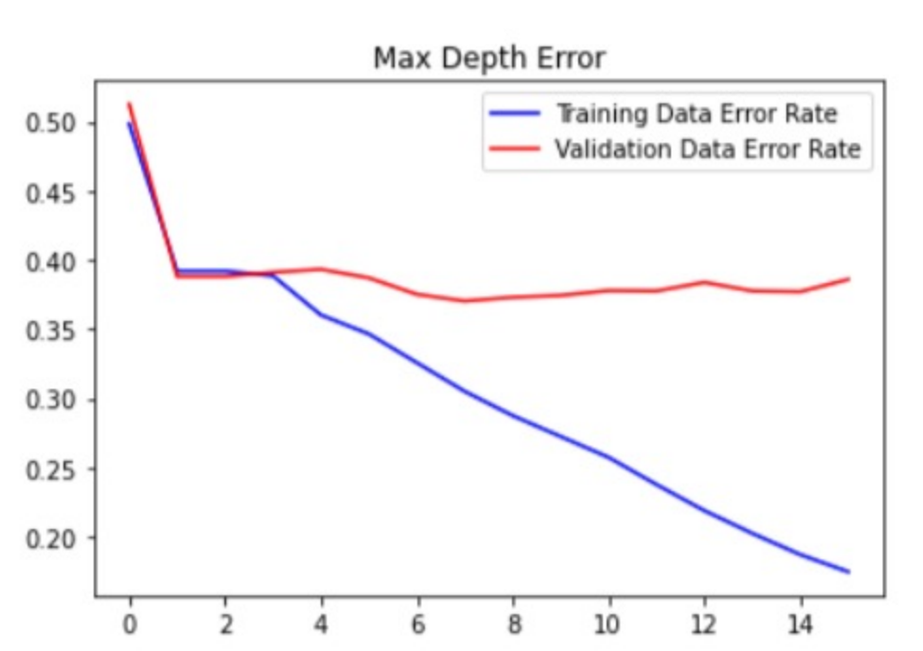
In [131]:
```python
Xtr, Ytr = X[:3712], Y[:3712]
Xva, Yva = X[3712:7423], Y[3712:7423]
learner = ml.dtree.treeClassify(Xtr, Ytr, maxDepth=50)
trainingError = learner.err(Xtr,Ytr)
validationError = learner.err(Xva,Yva)
print ('\033[1m' + "\nTraining Error:", trainingError, "\n")
print ('\033[1m' + "Validation Error:", validationError)
```

```
Training Error: 0.0

Validation Error: 0.40878469415251956
```

Problem 2 Part 3:

In [132]:
```python
training_list1 = []
validation_list1 = []
for depth in range(16):
    learner = ml.dtree.treeClassify(Xtr,Ytr,maxDepth = depth)
    training_list1.append(learner.err(Xtr,Ytr) )
    validation_list1.append(learner.err(Xva,Yva) )
    print('\033[1m' + "\nDepth:" , depth)
    print('\033[1m' + "Training Error Rate:", learner.err(Xtr,Ytr))
    print('\033[1m' + "Validation Error Rate:", learner.err(Xva,Yva))
print()
plt.title("Max Depth Error")
plt.plot(training_list1, c = "Blue",label = "Training Data Error Rate")
plt.plot(validation_list1, c = "Red",label = "Validation Data Error Rate")
plt.legend()
plt.show()
```

```
Depth: 0
Training Error Rate: 0.4983836206896552
Validation Error Rate: 0.5125303152789006

Depth: 1
Training Error Rate: 0.39197198275862066
Validation Error Rate: 0.3880355699272433

Depth: 2
Training Error Rate: 0.39197198275862066
Validation Error Rate: 0.3880355699272433

Depth: 3
Training Error Rate: 0.3884698275862069
Validation Error Rate: 0.39099973053085424

Depth: 4
Training Error Rate: 0.36018318965517243
Validation Error Rate: 0.39315548369711667

Depth: 5
Training Error Rate: 0.3464439655172414
Validation Error Rate: 0.3869576933441121

Depth: 6
Training Error Rate: 0.32570043103448276
Validation Error Rate: 0.37510105092966856

Depth: 7
Training Error Rate: 0.30495689655172414
Validation Error Rate: 0.370250606305578

Depth: 8
Training Error Rate: 0.2874461206896552
Validation Error Rate: 0.37294529776340607

Depth: 9
Training Error Rate: 0.2723599137931034
Validation Error Rate: 0.3742926434923201

Depth: 10
Training Error Rate: 0.2572737068965517
Validation Error Rate: 0.37779574238749664

Depth: 11
Training Error Rate: 0.23760775862068967
Validation Error Rate: 0.37752627324171384

Depth: 12
Training Error Rate: 0.21875
Validation Error Rate: 0.3837240635947184

Depth: 13
Training Error Rate: 0.2023168103448276
Validation Error Rate: 0.37752627324171384

Depth: 14
Training Error Rate: 0.1869612068965517
Validation Error Rate: 0.3769873349501482

Depth: 15
Training Error Rate: 0.17456896551724138
Validation Error Rate: 0.3858798167609809
```



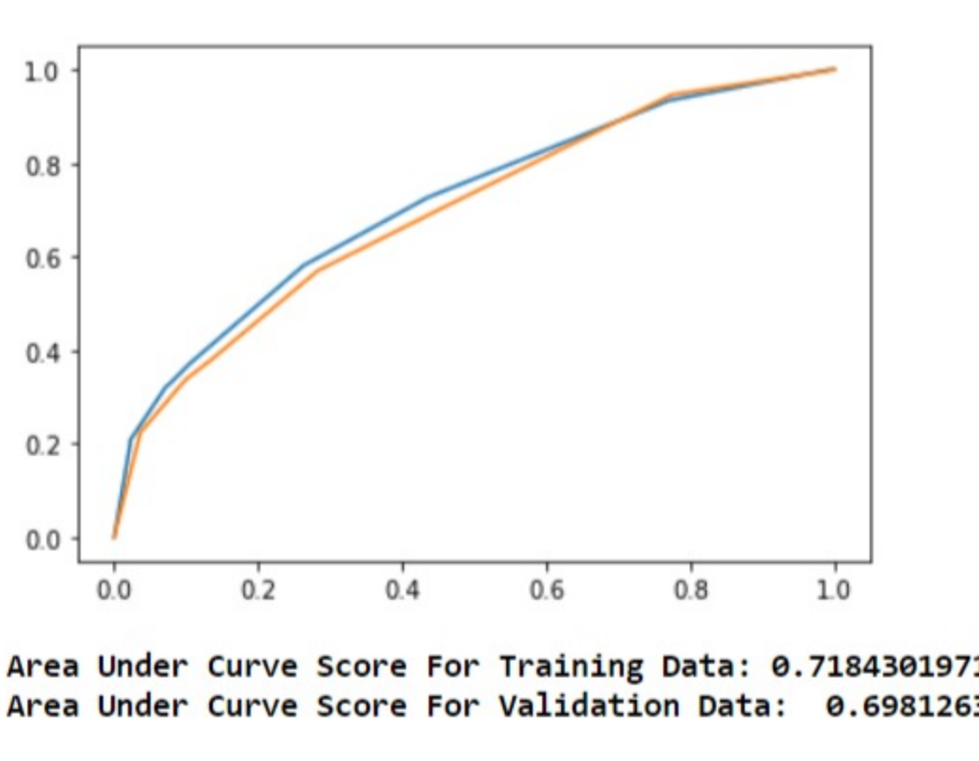
Problem 2 Part 4:

In [133]:
```python
training_list2 = []
validation_list2 = []
for parent in [2**p for p in range(0,14)]:
    learner = ml.dtree.treeClassify(Xtr, Ytr, maxDepth = 50, minParent = parent)
    training_list2.append(learner.err(Xtr,Ytr))
    validation_list2.append(learner.err(Xva,Yva))
    print('\033[1m' + "\nMin Parent:" , parent)
    print('\033[1m' + "Training Error Rate:", learner.err(Xtr,Ytr))
    print('\033[1m' + "Validation Error Rate:", learner.err(Xva,Yva))
print()
plt.plot([x for x in range(0,14)], training_list2, c = "Blue",label = "Training Data Error Rate")
plt.plot([y for y in range(0,14)], validation_list2, c = "red",label = "Validation Data Error Rate")
plt.legend()
plt.show()
```

```
Min Parent: 1
Training Error Rate: 0.0
Validation Error Rate: 0.4125572621934788

Min Parent: 2
Training Error Rate: 0.0
Validation Error Rate: 0.3971975208838588

Min Parent: 4
Training Error Rate: 0.009428879310344827
Validation Error Rate: 0.40905416329830235

Min Parent: 8
Training Error Rate: 0.03879310344827586
Validation Error Rate: 0.4033953112368634

Min Parent: 16
Training Error Rate: 0.09617456896551724
Validation Error Rate: 0.4074373484236055

Min Parent: 32
Training Error Rate: 0.15975215517241378
Validation Error Rate: 0.40662894098625707

Min Parent: 64
Training Error Rate: 0.2200969827586207
Validation Error Rate: 0.39611964430072755

Min Parent: 128
Training Error Rate: 0.26643318965517243
Validation Error Rate: 0.3856103476151981

Min Parent: 256
Training Error Rate: 0.30145474137931033
Validation Error Rate: 0.38884397736459175

Min Parent: 512
Training Error Rate: 0.32112068965517243
Validation Error Rate: 0.36189706278631095

Min Parent: 1024
Training Error Rate: 0.3407866379310345
Validation Error Rate: 0.35408245755860956

Min Parent: 2048
Training Error Rate: 0.36907327586206895
Validation Error Rate: 0.38426300188628404

Min Parent: 4096
Training Error Rate: 0.4983836206896552
Validation Error Rate: 0.5125303152789006

Min Parent: 8192
Training Error Rate: 0.4983836206896552
Validation Error Rate: 0.5125303152789006
```

Problem 2 Part 6:

In [134]:
```python
learner = ml.dtree.treeClassify(Xtr,Ytr,maxDepth = 50,minParent = 1024)
trainingData = learner.roc(Xtr,Ytr)
validationData = learner.roc(Xva,Yva)
plt.plot(trainingData[0], trainingData[1], validationData[0], validationData[1])
print()
plt.show()
trainingAreaUnderCurve = learner.auc(Xtr,Ytr)
validationAreaUnderCurve = learner.auc(Xva,Yva)
print('\033[1m' + "Area Under Curve Score For Training Data:", trainingAreaUnderCurve)
print('\033[1m' + "Area Under Curve Score For Validation Data: ", validationAreaUnderCurve)
```




```
Area Under Curve Score For Training Data: 0.7184301971144076
Area Under Curve Score For Validation Data:  0.69812637943592
```

Problem 2 Part7:

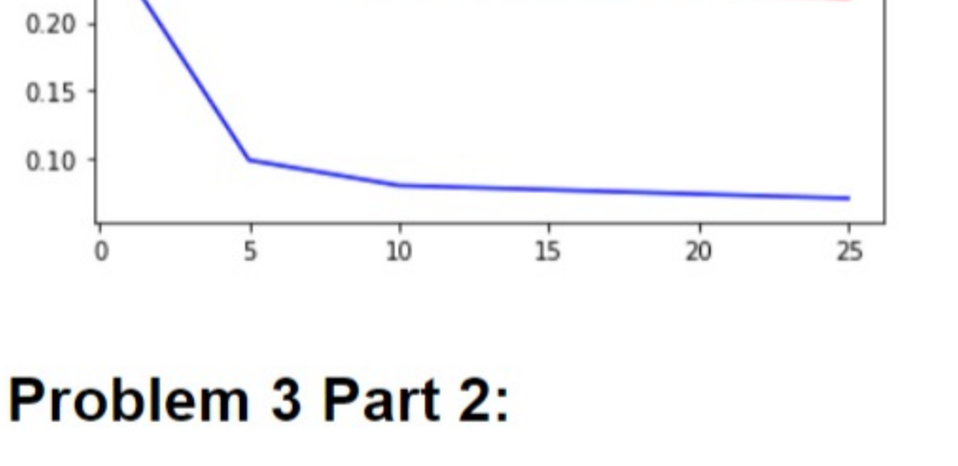
In [ ]:
```python
Xtr = np.genfromtxt('data/X_train.txt', delimiter = ',')
Ytr = np.genfromtxt('data/Y_train.txt', delimiter = ',')
Xtr,Ytr = ml.shuffleData(Xtr,Ytr)
learner = ml.dtree.treeClassify(Xtr,Ytr,maxDepth = 50, minParent = 1024)
Xte = np.genfromtxt('data/X_test.txt', delimiter=',')
Yte = np.vstack((np.arange(Xte.shape[0]), learner.predictSoft(Xte)[:,1])).T
np.savetxt('Y_submit.txt', Yte,'%d, %.2f', header='ID,Predicted', delimiter=',')
```

Problem 3 with option 1 (Random Forests)

Problem 3 Part 1:

In [137]:
```python
ensemble = [i for i in range(25)]
training_list3 = []
testing_list3 = []
Ytr_hat = np.zeros((np.size(Ytr),25))
Yva_hat = np.zeros((np.size(Yva),25))
for j in range(0,25):
    X_bsd,Y_bsd = ml.bootstrapData(Xtr,Ytr)
    ensemble[j] = ml.dtree.treeClassify(X_bsd, Y_bsd, maxDepth= 20, minLeaf = 4, nFeatures = 8)
    Ytr_hat[:,j] = ensemble[j].predict(Xtr)
    Yva_hat[:,j] = ensemble[j].predict(Xva)
for i, l in enumerate([1,5,10,25]):
    training = np.mean((Ytr - np.mean(Ytr_hat[:,0:l],1)) ** 2)
    validation = np.mean((Yva - np.mean(Yva_hat[:,0:l],1)) ** 2)
    training_list3.append(training)
    testing_list3.append(validation)
    print('\033[1m' + "\nLearner Number " + str(l) +":")
    print('\033[1m' + "MSE On Training Data Set:", training_list3[i])
    print('\033[1m' + "MSE On Testing Data Set:", testing_list3[i])
print()
_,axis = plt.subplots()
axis.plot([1,5,10,25],training_list3, c = 'Blue',label = "MSE For Training Data")
axis.plot([1,5,10,25],testing_list3, c = 'Red',label = "MSE For Testing Data")
plt.legend()
plt.show()
```

```
Learner Number 1:
MSE On Training Data Set: 0.2335668103448276
MSE On Testing Data Set: 0.4141740770681757

Learner Number 5:
MSE On Training Data Set: 0.09911637931034485
MSE On Testing Data Set: 0.2544543249797898

Learner Number 10:
MSE On Training Data Set: 0.08052532327586208
MSE On Testing Data Set: 0.2323012665049852

Learner Number 25:
MSE On Training Data Set: 0.07107586206896553
MSE On Testing Data Set: 0.2189538129884128
```



Problem 3 Part 2:

In [167]:
```python
from sklearn.metrics import auc
ens = [i for i in range(25)]
Xva = np.genfromtxt("data/X_test.txt",delimiter= ',')
Y_hat = np.zeros((np.size(Xva,0), 1))
for i in range(0,25):
    X_bsd,Y_bsd = ml.bootstrapData(Xtr,Ytr)
    ens[i] = ml.dtree.treeClassify(X_bsd,Y_bsd, maxDepth = 8, minLeaf = 128, nFeatures = 8,minParent = 256)
    Y_hat = Y_hat + ens[i].predictSoft(Xva)
Y_hat = Y_hat / 25
perf = ens[24].auc(X_bsd,Y_bsd)
print('\033[1m' + "\nPerformance For 25 Learners(Area Under Curve):", perf)
Yte = np.vstack((np.arange(len(Y_hat)) , Y_hat[:,1])).T
np.savetxt('Y_Random_Forest.txt', Yte,header='ID,Predicted',delimiter=',');
```

```
Performance For 25 Learners(Area Under Curve): 0.7339182738303149
```

Kaggle Username: Hootan Hosseinzadegan

Kaggle Team name: H.H

Kaggle Leaderboard Score: 0.72995

Problem4:

I completed HW4 entirely on my own by using the lecture videos and the discussion sessions. Furthermore, I completely followed the academic honesty guidelines which is on our canvas website and I did not discuss my homework with anyone in-person.