In [23]:

```
#James Wilson 93265297

# Import all required libraries
from __future__ import division # For python 2.*

import numpy as np
import matplotlib.pyplot as plt
import mltools as ml

np.random.seed(0)
%matplotlib inline

# Data Loading
X = np.genfromtxt('data/X_train.txt', delimiter=None)
Y = np.genfromtxt('data/Y_train.txt', delimiter=None)
X,Y = ml.shuffleData(X,Y)
[Xtr, Xva, Ytr, Yva] = ml.splitData(X,Y,0.80)

#Test Data
Xte = np.genfromtxt('data/X_test.txt', delimiter=None)
Xt, Yt = Xtr[:], Ytr[:]
```

Computes 1 decision tree's training and validation error rates.

In [38]:

```
Xt, Yt = Xtr[:], Ytr[:]
Xv, Yv = Xtr[10000: ], Ytr[10000: ]

XtrainSet, parameters = ml.rescale(Xt) #normalizing features of training data set
XvalidSet, foo = ml.rescale(Xva, parameters) #normalize features of Validation Set

#decision tree classifier with minLeaf of 25 and maxDepth of 50
learner = ml.dtree.treeClassify(Xt, Yt, minLeaf=25,  maxDepth=50)

#Prediction
probs = learner.predictSoft(Xv)

#Area Under Curve
print("{0:>15}: {1:.4f}".format("Train AUC", learner.auc(Xt, Yt)))
print("{0:>15}: {1:.4f}".format("Validation AUC", learner.auc(Xv, Yv)))
```

```
      Train AUC: 0.8231
 Validation AUC: 0.8237
```

Random Forest Method

In [24]:

```
np.random.seed(0)
n_bags = 10
bags = []
for l in range(n_bags):
    Xi, Yi = ml.bootstrapData(Xt, Yt, Xt.shape[0]) #boost data to size of original c
    # Train the model on draw
    tree = ml.dtree.treeClassify(Xi, Yi, minParent=2**6, maxDepth=25, nFeatures=6)
    bags.append(tree)
```

In [36]:

```
print("X shape: ", X.shape)
print("Y shape: ", Y.shape)

for feature in range(n_bags):
    print("\nTree",feature)
    print("{0:>15}: {1:.4f}".format("Train AUC", bags[feature].auc(Xt,Yt)))
    print("{0:>15}: {1:.4f}".format("Validation AUC", bags[feature].auc(Xva, Yva)))
```

```
X shape:  (100000, 14)
Y shape:  (100000,)

Tree 0
      Train AUC: 0.7713
 Validation AUC: 0.6593

Tree 1
      Train AUC: 0.7767
 Validation AUC: 0.6646

Tree 2
      Train AUC: 0.7712
 Validation AUC: 0.6552

Tree 3
      Train AUC: 0.7764
 Validation AUC: 0.6537

Tree 4
      Train AUC: 0.7835
 Validation AUC: 0.6725

Tree 5
      Train AUC: 0.7790
 Validation AUC: 0.6516
```

```
Tree 6
      Train AUC: 0.7784
 Validation AUC: 0.6571

Tree 7
      Train AUC: 0.7806
 Validation AUC: 0.6532

Tree 8
      Train AUC: 0.7731
 Validation AUC: 0.6579

Tree 9
      Train AUC: 0.7782
 Validation AUC: 0.6621
```

In [28]:

```
class BaggedTree(ml.base.classifier):
    def __init__(self, learners):
        """Construct BaggedTree with leaners"""
        self.learners = learners

    def predictSoft(self, X):
        """Predicts probability of eached bagged leaner and averages"""
        n_bags = len(self.learners)
        predictions = [self.learners[l].predictSoft(X) for l in range(n_bags)]
        return np.mean(predictions, axis=0)
```

In [34]:

```
bt = BaggedTree(bags)
bt.classes = np.unique(Y)

print("Averaged Score of bagged Trees")
print("{0:>15}: {1:.4f}".format("Train AUC", bt.auc(Xt,Yt)))
print("{0:>15}: {1:.4f}".format("Validation AUC", bt.auc(Xva, Yva)))
```

```
Averaged Score of bagged Trees
      Train AUC: 0.8932
 Validation AUC: 0.7341
```

Interpretation

The Single Decision Tree has a Validation AUC of about 82% and Train AUC of 82%, which is better than the Bagged Tree Method above, or other methods I tried like Boosting, or Max Depth methods. Unlike the notes or sample notebook methods, I allowed Xt, Yt to train on the entire dataset, which improved accuracy.

In [141]:

```
# kaggle username: wilsonhj
# James W.
```

In [130]:

```
```

```
[[227.      220.      240.03    ...   1.7651   2.7532  26.     ]
 [245.5     231.      243.91    ...   3.1359  24.383    0.     ]
 [253.      226.      239.53    ...   1.8709   3.2658   0.     ]
 ...
 [247.      239.      246.3     ...   2.1083  20.       0.     ]
 [247.      242.      250.38    ...   1.6083  20.       0.     ]
 [250.      238.      247.78    ...   2.89    20.       0.     ]] X traini
ng
[0. 0. 0. ... 0. 0. 0.] Y training
```

3.1 Problem 3: Decision Trees on Kaggle (50 points)

In [131]:

```
X shape:  (100000, 14)
Y shape:  (100000,)
                Feature 1 |
Minimum:  193.0
Maximum:  253.0
Mean:  241.58774299999996
Variance:  83.950806887951
                Feature 2 |
Minimum:  152.5
Maximum:  248.0
Mean:  227.3859329
Variance:  92.29796657769761
                Feature 3 |
Minimum:  214.25
Maximum:  252.38
Mean:  241.56281370000002
Variance:  35.300050500092304
                Feature 4 |
Minimum:  152.5
```

3.2 Compute and report your decision tree's training and validation error rates.

In [135]:

```
      Train AUC: 0.8033
 Validation AUC: 0.6267
```

In [136]:

In [137]:

```
      Train AUC: 0.5950
 Validation AUC: 0.5951
maxDepth=1

      Train AUC: 0.6261
 Validation AUC: 0.6275
maxDepth=2

      Train AUC: 0.6770
 Validation AUC: 0.6470
maxDepth=5

      Train AUC: 0.7884
 Validation AUC: 0.6280
maxDepth=10

      Train AUC: 0.9908
 Validation AUC: 0.5920
maxDepth=25
```

3.3 Interpretation

Based on results above from a Decision tree with a fixed minLeaf value of 25, Training and Validation accuracy improve as MaxDepth increases from 1 to 5. However, no gains are seen for maxDepth values >= 25. The highest value of Validation AUC is maxDepth = 5.

In [142]:

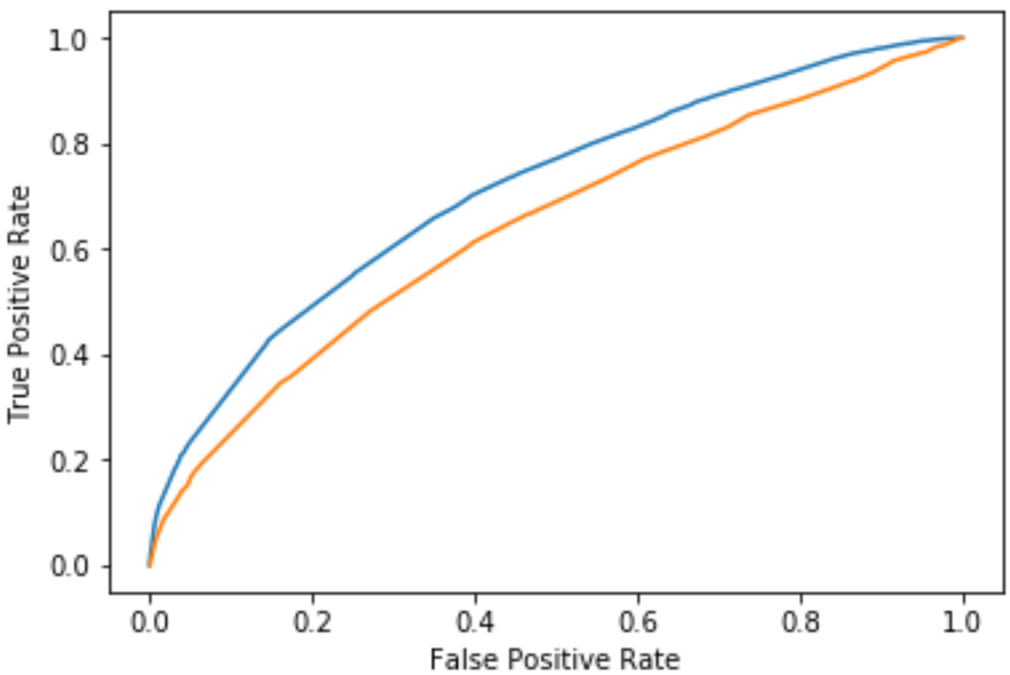
```
      Train AUC: 1.0000
 Validation AUC: 0.5770
minParent =  2
      Train AUC: 0.9984
 Validation AUC: 0.5739
minParent =  4
      Train AUC: 0.9879
 Validation AUC: 0.5826
minParent =  8
      Train AUC: 0.9532
 Validation AUC: 0.6009
minParent =  16
      Train AUC: 0.9058
 Validation AUC: 0.6084
minParent =  32
      Train AUC: 0.8510
 Validation AUC: 0.6142
minParent =  64
      Train AUC: 0.7883
```

3.4 interpretation

Modls with higher MinParent have higher Complexity. Min parent values greater than 16 yield diminishing returns with regard to Validation AUC. Validation AUC improves marginally for values >= 16, while train AUC diminishes significantly. The best compromise between train AUC and Validation AUC above may be minParent = 16. That said, the highest validation AUC in this set is .6491, when minParent = 1024. However, train AUC is .672, which is much lower than .9537 when minParent = 16.

In [139]:

```
[0. 0. 0. ... 0. 0. 0.]
      Train AUC: 0.7137
 Validation AUC: 0.6421
```



In [140]:

In [141]:

Problem 4: Statement of Collaboration (5 points)

I consulted with the T.A's in this course, and referenced course materials, notably discussions and notebooks on Sameer Sing's ML course website.

In [ ]: