In [1]:
```
from __future__ import division

import numpy as np
import matplotlib.pyplot as plt
import mltools as ml
from sklearn.ensemble import RandomForestClassifier


np.random.seed(0)
%matplotlib inline

X = np.genfromtxt("data/X_train.txt",delimiter=None)
Y = np.genfromtxt("data/Y_train.txt",delimiter=None)
Xte = np.genfromtxt('data/X_test.txt',delimiter=None)
Xtr,Xva,Ytr,Yva = ml.splitData(X,Y,0.80)
```

```
C:\Users\seoda\Anaconda2\lib\site-packages\sklearn\ensemble\weight_boosting.p
y:29: DeprecationWarning: numpy.core.umath_tests is an internal NumPy module
and should not be imported. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
```

In [24]:
```
clf = RandomForestClassifier(n_jobs=2, random_state=0, n_estimators = 50)
clf.fit(Xtr, Ytr)

print clf.score(X, Y)
print clf.score(Xva,Yva)
print clf.score(Xtr, Ytr)
```

```
0.91525
0.7116
0.9661625
```

In [3]:
```
Yfinal = clf.predict_proba(Xva)


Y_sub = np.vstack([np.arange(Xva.shape[0]), Yfinal[:, 1]]).T

# We specify the header (ID, Prob1) and also specify the comments as '' so the
header won't be commented out with
# the # sign.
np.savetxt('data/skRFva.txt', Y_sub, '%d, %.5f', header='ID,Prob1', comments=
'', delimiter=',')
```

In [4]:
```
Yfinal = clf.predict_proba(Xte)


Y_sub = np.vstack([np.arange(Xte.shape[0]), Yfinal[:, 1]]).T

# We specify the header (ID, Prob1) and also specify the comments as '' so the
header won't be commented out with
# the # sign.
np.savetxt('data/skRFte.txt', Y_sub, '%d, %.5f', header='ID,Prob1', comments=
'', delimiter=',')
```

In [5]:
```
#Xs, Ys = Xtr[:4000], Ytr[:4000]
#nn = ml.nnet.nnetClassify()
forest = []
Xtr, Ytr = ml.shuffleData(Xtr, Ytr)
for i in range(50):
    Xb, Yb = ml.bootstrapData(Xtr, Ytr, 20000)
    learner = ml.dtree.treeClassify(Xb, Yb, minLeaf=1, nFeatures=5)
    forest.append(learner)
    #print("{0:>15}: {1:.4f}".format('Train AUC',learner.auc(Xb, Yb)))
    #print("{0:>15}: {1:.4f}".format('Validation AUC', learner.auc(Xva, Yva)))
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
```

In [7]:
```
learnerPredictions = list()
learnerSoftPredictions = list()
for learner in forest:
    learnerPredictions.append(learner.predict(Xva))
    learnerSoftPredictions.append(learner.predictSoft(Xva))
learnerPredictions = np.array(learnerPredictions)
learnerSoftPredictions = np.array(learnerSoftPredictions)
```

In [22]:
```
predictionsVa = []
softPredictionsA = []
softPredictionsB = []
for i in range(20000):
    prob = 0
    for k in range(50):
        prob += learnerSoftPredictions[k][i][1]
    softPredictionsB.append([prob / 50])

for i in range(20000):
    vote = 0
    for k in range(50):
        if learnerPredictions[k][i] == 1.0:
            vote += 1
    if vote > 25:
        predictionsVa.append(1.0)
    else:
        predictionsVa.append(0.0)
    softPredictionsA.append([vote / 50])
```

In [19]:
```
learnerPredictionsTr = list()
for learner in forest:
    learnerPredictionsTr.append(learner.predict(Xtr))
learnerPredictionsTr = np.array(learnerPredictionsTr)
predictionsTr = list()
for i in range(80000):
    vote = 0
    for k in range(50):
        if learnerPredictionsTr[k][i] == 1.0:
            vote += 1
    if vote > 25:
        predictionsTr.append(1.0)
    else:
        predictionsTr.append(0.0)
```

In [23]:
```
predictionsVa = np.array(predictionsVa)
predictionsTr = np.array(predictionsTr)
softPredictionsA = np.array(softPredictionsA)
softPredictionsB = np.array(softPredictionsB)
print(softPredictionsA.shape)
print(softPredictionsB.shape)

def err(Yhat, Yva):
    sum = 0
    for a,b in zip(Yhat, Yva):
        if a != b:
            sum += 1
    return sum / Yhat.shape[0]

print(err(predictionsVa, Yva))
print(err(predictionsTr, Ytr))
print(softPredictionsA)
print(softPredictionsB)
```

```
(20000L, 1L)
(20000L, 1L)
0.27915
0.1679375
[[0.18]
 [0.3 ]
 [0.74]
 ...
 [0.8 ]
 [0.5 ]
 [0.32]]
[[0.32833333]
 [0.40416667]
 [0.62333333]
 ...
 [0.66416667]
 [0.51116667]
 [0.4335    ]]
```

In [10]:
```
Y_sub = np.vstack([np.arange(Xva.shape[0]), softPredictionsA[:, 0]]).T

# We specify the header (ID, Prob1) and also specify the comments as '' so the
header won't be commented out with
# the # sign.
np.savetxt('data/myRF1va.txt', Y_sub, '%d, %.5f', header='ID,Prob1', comments=
'', delimiter=',')

Y_sub = np.vstack([np.arange(Xva.shape[0]), softPredictionsB[:, 0]]).T

# We specify the header (ID, Prob1) and also specify the comments as '' so the
header won't be commented out with
# the # sign.
np.savetxt('data/myRF2va.txt', Y_sub, '%d, %.5f', header='ID,Prob1', comments=
'', delimiter=',')
```

In [11]:
```
ySub = list()
yProb = list()
for learner in forest:
    ySub.append(learner.predict(Xte))
    yProb.append(learner.predictSoft(Xte))
ySub = np.array(ySub)
yProb = np.array(yProb)

YfinalA = []
YfinalB = []
for i in range(Xte.shape[0]):
    vote = 0
    prob = 0
    for k in range(50):
        if ySub[k][i] == 1.0:
            vote += 1
        prob += yProb[k][i][1]
    YfinalA.append([vote / 50])
    YfinalB.append([prob / 50])

YfinalA = np.array(YfinalA)
YfinalB = np.array(YfinalB)
print(YfinalA.shape)
print(YfinalB.shape)
```

```
(100000L, 1L)
(100000L, 1L)
```

In [12]:
```
len(YfinalA[:,0])
len(YfinalB[:,0])
```

Out[12]:
```
100000
```

In [13]:
```
Y_sub = np.vstack([np.arange(Xte.shape[0]), YfinalA[:, 0]]).T

# We specify the header (ID, Prob1) and also specify the comments as '' so the
header won't be commented out with
# the # sign.
np.savetxt('data/myRF1te.txt', Y_sub, '%d, %.5f', header='ID,Prob1', comments=
'', delimiter=',')

Y_sub = np.vstack([np.arange(Xte.shape[0]), YfinalB[:, 0]]).T

# We specify the header (ID, Prob1) and also specify the comments as '' so the
header won't be commented out with
# the # sign.
np.savetxt('data/myRF2te.txt', Y_sub, '%d, %.5f', header='ID,Prob1', comments=
'', delimiter=',')
```

This is the best I have so far. Hope that my team has more than 2 more models for the final report. We gotta figure out a way to combine later.

This is pretty good. I think I can make it even better.

Things I can change: number of trees: best was 20 number of data per bootstrap: best was 20,000 the way to calculate the probability: so far, I just calculate the probability of trees that got it correct.

In [14]:
```
Pv0 = np.genfromtxt('data/skRFva.txt',delimiter=',',skip_header=1)[:,1:2]
Pv1 = np.genfromtxt('data/myRF1va.txt',delimiter=',',skip_header=1)[:,1:2]
Pv2 = np.genfromtxt('data/myRF2va.txt',delimiter=',',skip_header=1)[:,1:2]

Pe0 = np.genfromtxt('data/skRFte.txt',delimiter=',',skip_header=1)[:,1:2]
Pe1 = np.genfromtxt('data/myRF1te.txt',delimiter=',',skip_header=1)[:,1:2]
Pe2 = np.genfromtxt('data/myRF2te.txt',delimiter=',',skip_header=1)[:,1:2]
```

In [15]:
```
Sv = np.hstack((Pv0,Pv1,Pv2))
stack = ml.linearC.linearClassify(Sv,Yva,reg=1e-3)
print "** Stacked AUC: ",stack.auc(Sv,Yva)

Se = np.hstack((Pe0,Pe1,Pe2))
PeS = stack.predictSoft(Se)
```

```
** Stacked AUC:  0.7444583512851285
```

In [16]:
```
Y_sub = np.vstack([np.arange(Xte.shape[0]), PeS[:, 1]]).T

# We specify the header (ID, Prob1) and also specify the comments as '' so the
header won't be commented out with
# the # sign.
np.savetxt('data/myCombinationSub.txt', Y_sub, '%d, %.5f', header='ID,Prob1',
comments='', delimiter=',')
```