In [1]:

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

#1.1 It can only solve for graph a,b but not for c d. because it can only seperate the feature by fixed gradient straight line.In graph c, if two point on one side of speration line with +1 -1 then it would fail.Same for d.

#1.2 It can only solve for graph a,b,c but not for d.Because the line can change gradient to ajust the shuffle,in the graph d, if two points on one side of speration line with +1 -1 then it would fail.

#1.3 It can only solve for graph a,b,c but not for d. It has a circle to seperate variables, in the graph d, if inner circle has two points with +1 -1 then it would fail.

In [97]:

```
#2.1

p=0.4

y=-(p*np.log2(p) + (1-p)*np.log2(1-p))

print("entropy is ",y)
```

```
entropy is  0.9709505944546686
```

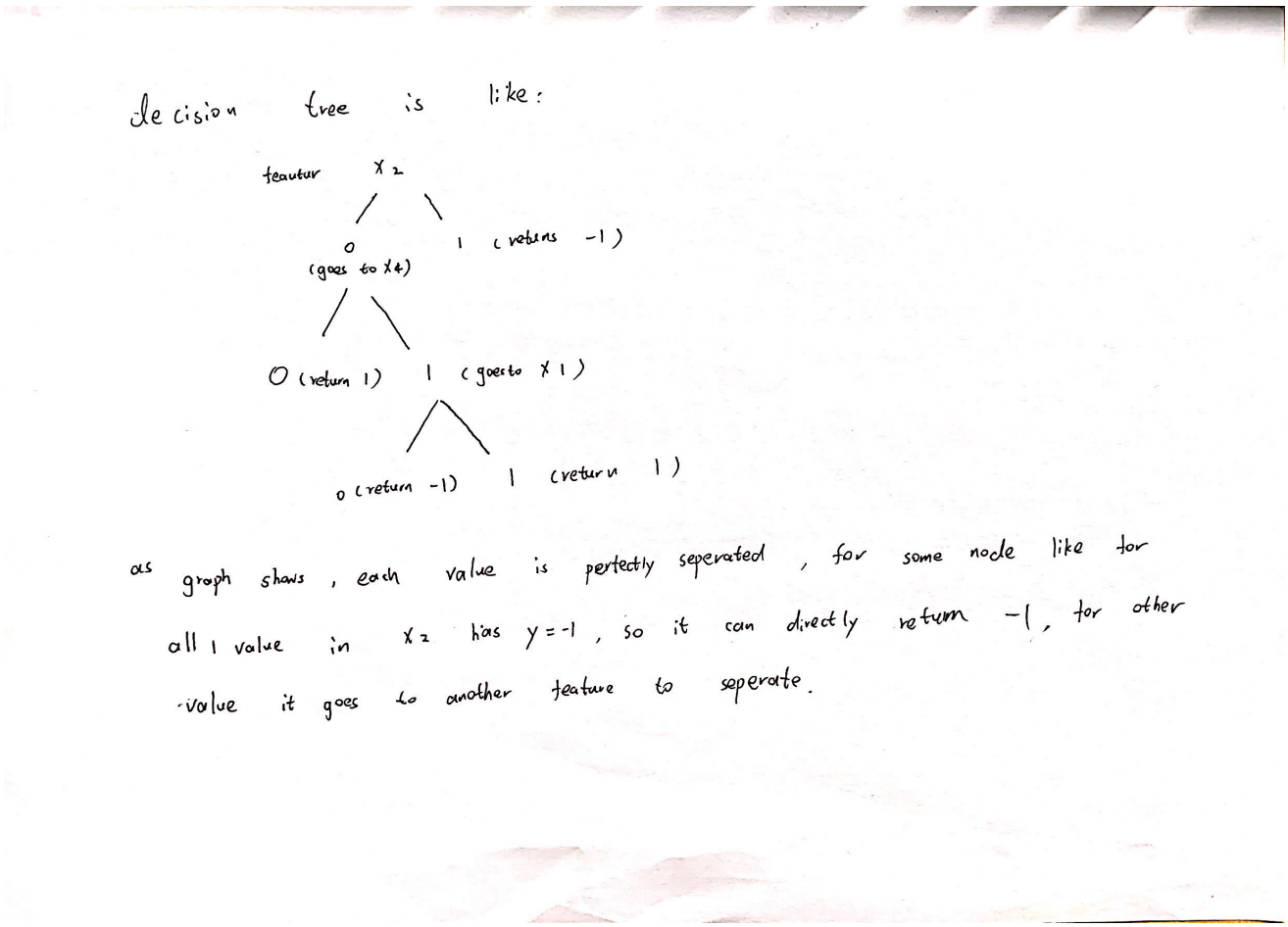
In [96]:

```
#2.2

e1 = 3/4*np.log2(4/3) + (1/4)*np.log2(4)
e2 = 0.5*np.log2(2) + 0.5*np.log2(2)
print("x1 ",y - 0.4*e1 - 0.6*e2)

e1 = 1/5*np.log2(5) + 4/5*np.log2(5/4)
e2 = 0
print("x2 ",y - 1/2*e1 - 0.5*e2)

e1 = 2/3*np.log2(3/2) + 1/3*np.log2(3)
e2 = 4/7*np.log2(7/4) + 3/7*np.log2(7/3)
print("x3 ",y - 3/10*e1 - 7/10*e2)

e1 = 1/3*np.log2(3) + 2/3*np.log2(3/2)
e2 = 5/7*np.log2(7/5) + 2/7*np.log2(7/2)
print("x4 ",y - 3/10*e1 - 7/10*e2)


e1 = 4/7*np.log2(7/4) + 3/7*np.log2(7/3)
e2 = 2/3*np.log2(3/2) + 1/3*np.log2(3)
print("x5 ",y - 7/10*e1 - 3/10*e2)


#from the result, we can see x2 with 0.6099 has highest gain, we should choose x2 feature.
```

```
x1  0.046439344671015514
x2  0.6099865470109874
x3  0.0058021490143458365
x4  0.09127744624168022
x5  0.0058021490143458365
```

In [100]:

#2.3


decision tree is like:
feautur X2
0
(goes to X1)
1 (returns -1)
0 (return 1)
1 (goes to X1)
0 (return -1)
1 (return 1)
as graph shows, each value is pertectly seperated, for some nocle like for all 1 value in X2 bias y=-1, so it can directly return -1, for other value it goes to another feature to seperate.

In [112]:

```
#3.1
X = np.genfromtxt('X_train.txt', delimiter=None)
Y = np.genfromtxt('Y_train.txt', delimiter=None)
X,Y = ml.shuffleData(X,Y)


for i in range(14):
    print(" min ",i+1,np.min(X[:,i]),'\n'," max ",i+1,np.max(X[:,i]),'\n'," mean ",i+1,np.mean(X[:,i
    print()
```

```
 min  1 0.87589
  max  1 19.167
  mean  1 6.498652902749999
 var  1 6.40504819135735

 min  2 0.0
  max  2 9238.0
  mean  2 138.09383
 var  2 443951.7464459311

 min  3 0.0
  max  3 13630.0
  mean  3 928.25902
 var  3 3081761.8169486397

 min  4 0.0
  max  4 13.23
  mean  4 2.09713912048
 var  4 4.36344047061341

 min  5 152.5
  max  5 252.5
  mean  5 232.82676815000002
 var  5 97.62573174864559

 min  6 0.99049
  max  6 975.04
  mean  6 10.2715904759
 var  6 404.6462450411813

 min  7 10.0
  max  7 31048.0
  mean  7 3089.923365
 var  7 15651513.756432077

 min  8 0.0
  max  8 73.902
  mean  8 2.6917184521500004
 var  8 2.198778474358265

 min  9 214.25
  max  9 252.5
  mean  9 241.55415049999993
 var  9 35.28633980334975

 min  10 0.0
  max  10 66.761
```

```
 mean  10 4.21766040935
var  10 4.086371884226909

min  11 193.5
 max  11 253.0
 mean  11 241.60110369999998
var  11 83.4991711498463

min  12 0.0
 max  12 125.17
 mean  12 3.2485793302999997
var  12 8.219485024912494

min  13 -999.9
 max  13 797.2
 mean  13 5.781480499999999
var  13 3406.520550978119

min  14 152.5
 max  14 249.0
 mean  14 227.37657130000002
var  14 92.62559312501632
```

In [87]:

```
#part3.2
Xtr=  X[0:10000]
Ytr =Y[0:10000]
Xva= X[10000:20000]
Yva= Y[10000:20000]
```

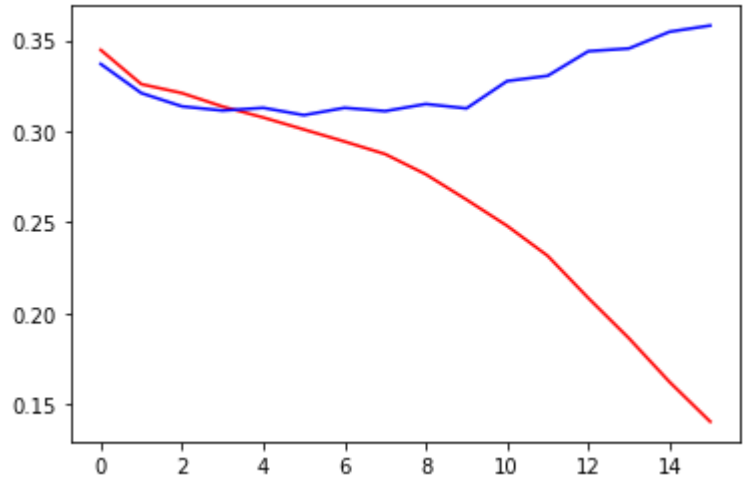
In [88]:

```
#part3.2
learner = ml.dtree.treeClassify(Xtr, Ytr, maxDepth=50)
print("train error:", learner.err(Xtr, Ytr))
print("validation error:", learner.err(Xva, Yva))
```

```
Training error: 0.0064
Validation error: 0.3932
```

In [106]:

```
#part3.3
list1=[]
terr = []
verr = []
for i in range(16):
    list1.append(i)
    learner = ml.dtree.treeClassify(Xtr, Ytr, maxDepth=i)
    terr.append(learner.err(Xtr, Ytr))
    verr.append(learner.err(Xva, Yva))


# blue is validation, red is train
plt.plot(list1, terr,"red")
plt.plot(list1, verr,"blue")
plt.show()

print(np.argmin(verr))
#higher maxdepth 5 has more complexity, I choose maxdepth with 5.
```

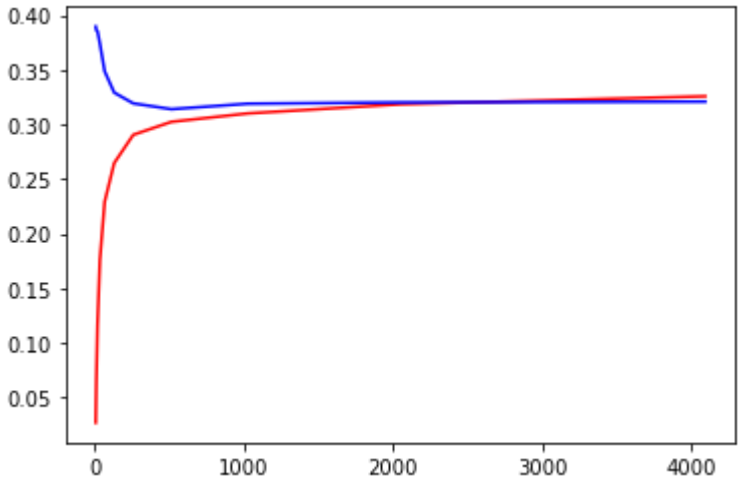


5

In [105]:

```
#part 3.4

list2=[]
terr = []
verr = []
for i in range(2,13):
    list2.append(2**i)
    learner = ml.dtree.treeClassify(Xtr, Ytr, minParent= 2**i)
    terr.append(learner.err(Xtr, Ytr))
    verr.append(learner.err(Xva, Yva))

# blue is validation, red is train
print(np.argmin(verr))
plt.plot(list2, terr,"red")
plt.plot(list2, verr,"blue")
plt.show()

#higher minparent has lower complexity, minparent 512 with lowest error,so choose 512
```

7



In [69]:

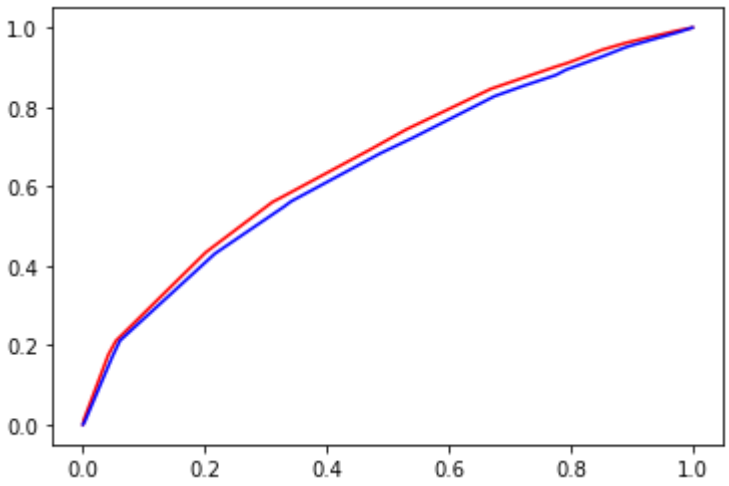
```
#3.4
print(list2[7])
```

512

In [111]:

```
#3.5
learner = ml.dtree.treeClassify(Xtr, Ytr, maxDepth=5,minParent= 512)
t1, t2,t3 = learner.roc(Xtr, Ytr)
v1, v2,v3 = learner.roc(Xva, Yva)
plt.plot(t1, t2, "red")
plt.plot(v1, v2, "blue")
# blue is validation, red is train
print('train:', learner.auc(Xtr, Ytr))
print('validation:', learner.auc(Xva, Yva))
```

```
train: 0.6714431746191424
validation: 0.6494224958380408
```



In [104]:

```
#3.6
learner = ml.dtree.treeClassify(X, Y, maxDepth=5,minParent= 512)
Xte = np.genfromtxt('X_test.txt', delimiter=None)
Yte = np.vstack((np.arange(Xte.shape[0]), learner.predictSoft(Xte)[:,1])).T
np.savetxt('Y_submit.txt',Yte,'%d, %.2f',header='ID,Prob1',comments='',delimiter=',')

#Kaggle username :Minzhe Chen
# score in kaggle: 0.66400
```

#4 I have done this homework myself.