Problem 1

1. y = T(a + bx1) is a line. Dataset a and b can be shattered. However, for dataset c, it cannot be shatter if (2,2) and (4,8) are same except (6,4). Dataset d has same situation with dataset c.
2. y = T(a + bx1 + cx2) is a line. Dataset a and b can be shattered.
3. y = T((x1 − a)^2 + (x2 - b)^2 + c) is a circle. Dataset a, b, and c can be shattered.

Problem 2

Problem 2.1

$$H(y) = (4/10) * log(5/2) + (6/10) * log(5/3) = 0.971$$

Problem 2.2

$$IG(x1) = 0.971 - (2/5) * H(1/4) + (3/5) * H(1/2) = 0.0465$$
$$IG(x2) = 0.971 - (1/2) * H(4/5) + (1/2) * H(0) = 0.61$$
$$IG(x3) = 0.971 - (3/10) * H(2/3) + (7/10) * H(3/7) = 0.0058$$
$$IG(x4) = 0.971 - (3/10) * H(2/3) + (7/10) * H(2/7) = 0.0913$$
$$IG(x5) = 0.971 - (3/10) * H(2/3) + (7/10) * H(3/7) = 0.0058$$

Root node should be x2 because it has highest information gain

Problem 2.3

In [1]:
```
# if x2 == 1:
#     y = -1
# else:
#     if x4 == 0:
#         y = 1
#     else:
#         if x1 == 1:
#             y = 1
#         else:
#             y = -1
```

Problem 3

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import mltools as ml

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

Problem 3.1

In [2]:
```
print("Minimum of each of the 14 features:\n", np.min(X,axis=0))
print("Maximum of each of the 14 features:\n", np.max(X,axis=0))
print("Mean of each of the 14 features:\n", np.mean(X,axis=0))
print("Variance of each of the 14 features:\n", np.var(X,axis=0))
```

```
Minimum of each of the 14 features:
 [ 1.9350e+02  1.5250e+02  2.1425e+02  1.5250e+02  1.0000e+01  0.0000e+00
  0.0000e+00  0.0000e+00  8.7589e-01  0.0000e+00  0.0000e+00  0.0000e+00
  9.9049e-01 -9.9990e+02]
Maximum of each of the 14 features:
 [2.5300e+02 2.4900e+02 2.5250e+02 2.5250e+02 3.1048e+04 1.3630e+04
 9.2380e+03 1.2517e+02 1.9167e+01 1.3230e+01 6.6761e+01 7.3902e+01
 9.7504e+02 7.9720e+02]
Mean of each of the 14 features:
 [2.41601104e+02 2.27376571e+02 2.41554150e+02 2.32826768e+02
 3.08992337e+03 9.28259020e+02 1.38093830e+02 3.24857933e+00
 6.49865290e+00 2.09713912e+00 4.21766041e+00 2.69171845e+00
 1.02715905e+01 5.78148050e+00]
Variance of each of the 14 features:
 [8.34991711e+01 9.26255931e+01 3.52863398e+01 9.76257317e+01
 1.56515138e+07 3.08176182e+06 4.43951746e+05 8.21948502e+00
 6.40504819e+00 4.36344047e+00 4.08637188e+00 2.19877847e+00
 4.04646245e+02 3.40652055e+03]
```

Problem 3.2

In [40]:
```
X, Y = ml.shuffleData(X, Y)
Xtr, Ytr = X[:10000], Y[:10000]
Xva, Yva = X[10000:], Y[10000:]
```

In [41]:
```
learner = ml.dtree.treeClassify(Xtr, Ytr, maxDepth=50)
```

In [42]:
```
print("training error rates: ", learner.err(Xtr,Ytr))
print("validation error rates: ", learner.err(Xva,Yva))
```

```
training error rates:  0.0061
validation error rates:  0.3821473684210526
```

Problem 3.3

In [43]:
```
depth = np.array(range(16))
tr_err = np.zeros(16)
va_err = np.zeros(16)
for i in range(16):
    learner = ml.dtree.treeClassify(Xtr, Ytr, maxDepth=i)
    tr_err[i] = learner.err(Xtr,Ytr)
    va_err[i] = learner.err(Xva,Yva)
    print("When depth = ", i, "training error rates = ", learner.err(Xtr,Ytr), "validation

plt.plot(depth, tr_err, color = "green", label = "Training error")
plt.plot(depth, va_err, color = "red", label = "Validation error")
plt.xlabel("maxDepth")
plt.ylabel("err")
plt.legend()
plt.show()
```

```
When depth =  0 training error rates =  0.3501 validation error rates =  0.342515789473
6842
When depth =  1 training error rates =  0.341 validation error rates =  0.3381578947368
421
When depth =  2 training error rates =  0.3286 validation error rates =  0.323147368421
0526
When depth =  3 training error rates =  0.32 validation error rates =  0.31838421052631
577
When depth =  4 training error rates =  0.3133 validation error rates =  0.316605263157
8947
When depth =  5 training error rates =  0.3082 validation error rates =  0.3173
When depth =  6 training error rates =  0.2993 validation error rates =  0.314136842105
26316
When depth =  7 training error rates =  0.2907 validation error rates =  0.318268421052
6316
When depth =  8 training error rates =  0.281 validation error rates =  0.3187684210526
316
When depth =  9 training error rates =  0.2674 validation error rates =  0.323994736842
1053
When depth =  10 training error rates =  0.2472 validation error rates =  0.32626842105
26316
When depth =  11 training error rates =  0.232 validation error rates =  0.327484210526
31577
When depth =  12 training error rates =  0.2093 validation error rates =  0.33804210526
31579
When depth =  13 training error rates =  0.189 validation error rates =  0.336647368421
05264
When depth =  14 training error rates =  0.1712 validation error rates =  0.34498421052
63158
When depth =  15 training error rates =  0.1526 validation error rates =  0.34202631578
94737
```

Higher maxDepth has higher complexity. Depth=6 has lowest validation error which has the best model.

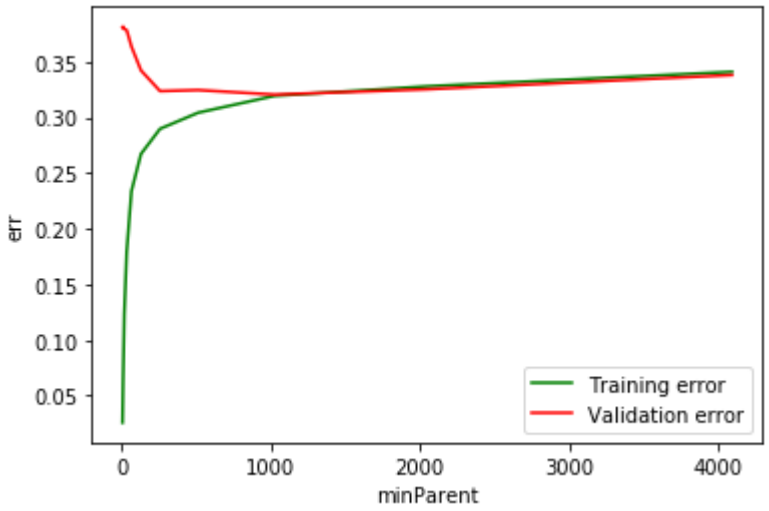
Problem 3.4

In [46]:
```
learner = ml.dtree.treeClassify(Xtr, Ytr, maxDepth=50)
```

In [55]:
```
mP = [2**i for i in range(2,13)]
tr_err = np.zeros(11)
va_err = np.zeros(11)
for i,j in enumerate(mP):
    learner.train(Xtr, Ytr, minParent = j)
    tr_err[i] = learner.err(Xtr,Ytr)
    va_err[i] = learner.err(Xva,Yva)
    print("When minParent = ", j, "training error rates = ", learner.err(Xtr,Ytr), "valida

plt.plot(mP, tr_err, color = "green", label = "Training error")
plt.plot(mP, va_err, color = "red", label = "Validation error")
plt.xlabel("minParent")
plt.ylabel("err")
plt.legend()
plt.show()
```

```
When minParent =  4 training error rates =  0.0257 validation error rates =  0.3810421052631579
When minParent =  8 training error rates =  0.0642 validation error rates =  0.3817210526315789
When minParent =  16 training error rates =  0.1233 validation error rates =  0.3794263157894737
When minParent =  32 training error rates =  0.1789 validation error rates =  0.3786526315789473
When minParent =  64 training error rates =  0.2338 validation error rates =  0.3637842105263157
When minParent =  128 training error rates =  0.2672 validation error rates =  0.34231052631578945
When minParent =  256 training error rates =  0.2898 validation error rates =  0.32371052631578945
When minParent =  512 training error rates =  0.3043 validation error rates =  0.3246263157894737
When minParent =  1024 training error rates =  0.3194 validation error rates =  0.3205736842105263
When minParent =  2048 training error rates =  0.328 validation error rates =  0.32533684210526315
When minParent =  4096 training error rates =  0.341 validation error rates =  0.3381578947368421
```



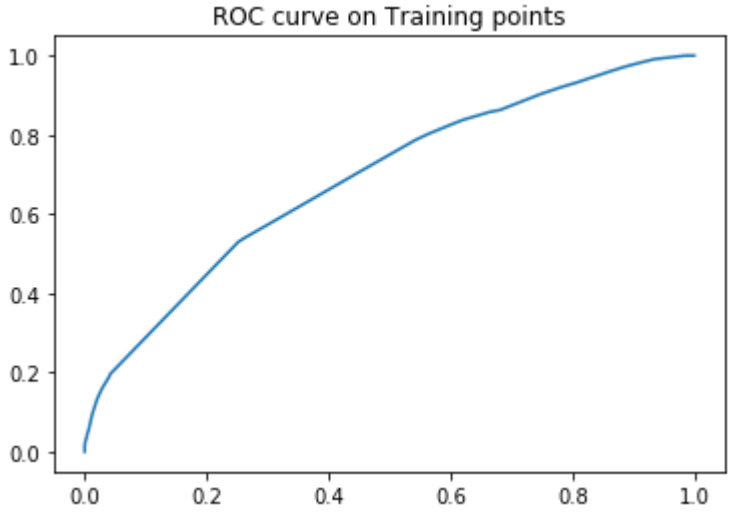
Higher minParent has lower complexity, minParent=1024 has lowest validation error which

has the best model.

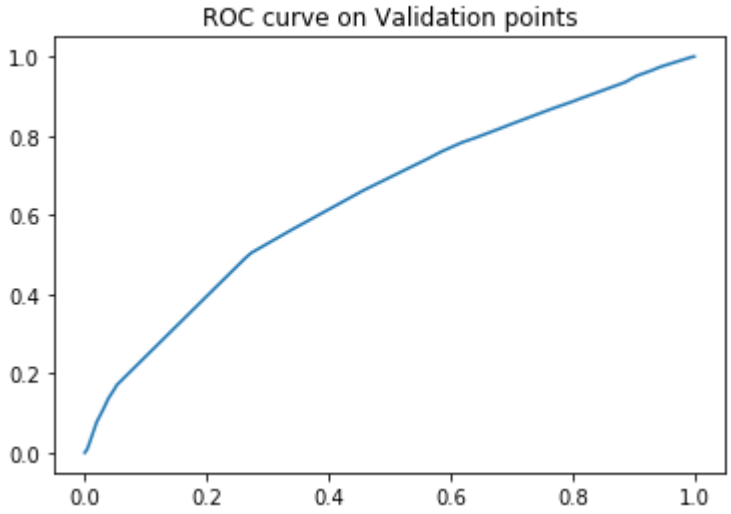
Problem 3.5

In [52]:
```
#Choose only maxDepth = 6 because it has least validation error rate
learner = ml.dtree.treeClassify(Xtr,Ytr,maxDepth=6)
```

In [53]:
```
tr_roc = learner.roc(Xtr,Ytr)
va_roc = learner.roc(Xva,Yva)
plt.plot(tr_roc[0],tr_roc[1])
plt.title("ROC curve on Training points")
plt.show()
```



In [54]:
```
plt.plot(va_roc[0],va_roc[1])
plt.title("ROC curve on Validation points")
plt.show()
```



In [51]:
```
print("AUC for training data is ", learner.auc(Xtr, Ytr))
print("AUC for validation data is ", learner.auc(Xva, Yva))
```

```
AUC for training data is  0.6599761200710289
AUC for validation data is  0.6449153209800155
```

Problem 3.6

In [57]:
```
learner = ml.dtree.treeClassify(X,Y,maxDepth=8, minParent = 1024)
```

In [58]:
```
Xte = np.genfromtxt('data/X_test.txt', delimiter=None)
```

In [59]:
```
Yte = np.vstack((np.arange(Xte.shape[0]), learner.predictSoft(Xte)[:,1])).T
```

In [60]:
```
np.savetxt('Y_submit.txt',Yte,'%d, %.2f',header='ID,Prob1',comments='',delimiter=',')
```

Kaggle username: Bingchen Lu

AUC: 0.6788

Problem 4

I have followed the academic honesty guidelines posted on the course website

In [ ]:
```
```