In [1]:
```
%matplotlib inline
import numpy as np
from sklearn import datasets
from sklearn.datasets import fetch_mldata
from sklearn.metrics import hamming_loss
import matplotlib.pyplot as plt




mnist = fetch_mldata('MNIST original')

train, test = mnist.data[0:60000, :], mnist.data[60000:, :]

X_trn, y_trn = mnist.data[0:60000, :], mnist.target[0:60000]
X_tst, y_tst = mnist.data[60000:, :], mnist.target[60000:]

error = np.zeros(10)
tmp=0;

from sklearn.naive_bayes import MultinomialNB

alphas = [0.0000001, 0.0001, 1, 1000, 1000000, 1000000000, 10000000000
, 1000000000000, 100000000000000, 10000000000000000]
for alpha in alphas:
    clf = MultinomialNB(alpha, class_prior=None, fit_prior=True)
    clf.fit(X_trn, y_trn)
    y_pred = clf.predict(X_tst)
    error[tmp]=hamming_loss(y_tst, y_pred)
    del clf
    print(error[tmp])
    tmp=tmp+1

plt.plot(error)
plt.title("Error rate wrt beta")
```

```
0.163
0.1631
0.1635
0.1644
0.2426
0.2616
0.2698
0.8865
0.8865
0.8865
```

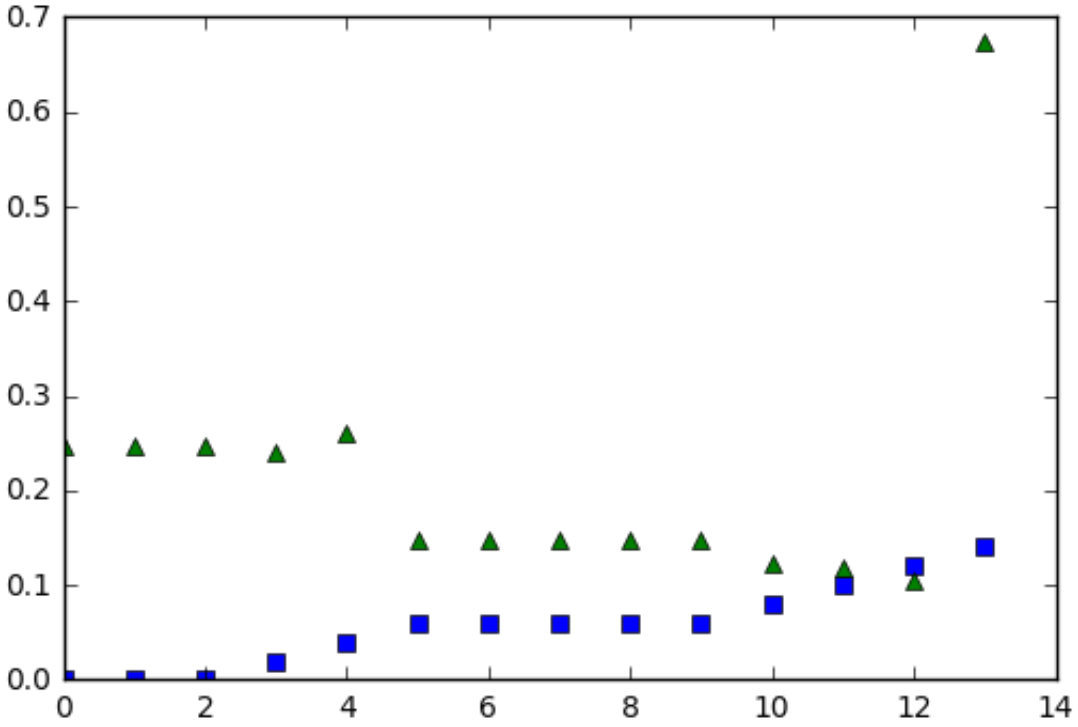
```
Out[1]: <matplotlib.text.Text at 0x105890f28>
```

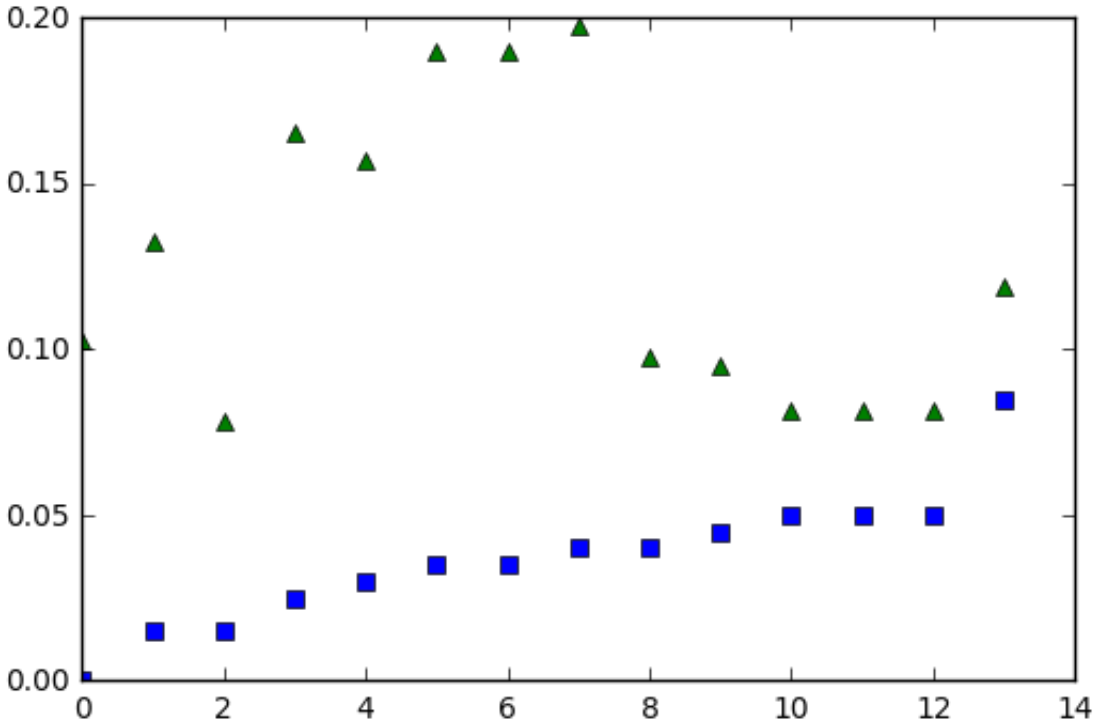
In [2]:
```
%matplotlib inline
import numpy as np
from sklearn.metrics import hamming_loss
from sklearn.naive_bayes import GaussianNB



data = np.genfromtxt('wdbc.txt', delimiter=',')

size = [50, 200, 400]
for sz in size:
    train, test = data[0:sz, :], data[sz:, :]
    X_trn, y_trn = train[:, 2:], train[:,1]
    X_tst, y_tst = test[:, 2:], test[:,1]
    gnb = GaussianNB()
    gnb.fit(X_trn, y_trn)
    y_pred = gnb.predict(X_tst)
    error = hamming_loss(y_tst, y_pred)
    print(error)
    del gnb
```

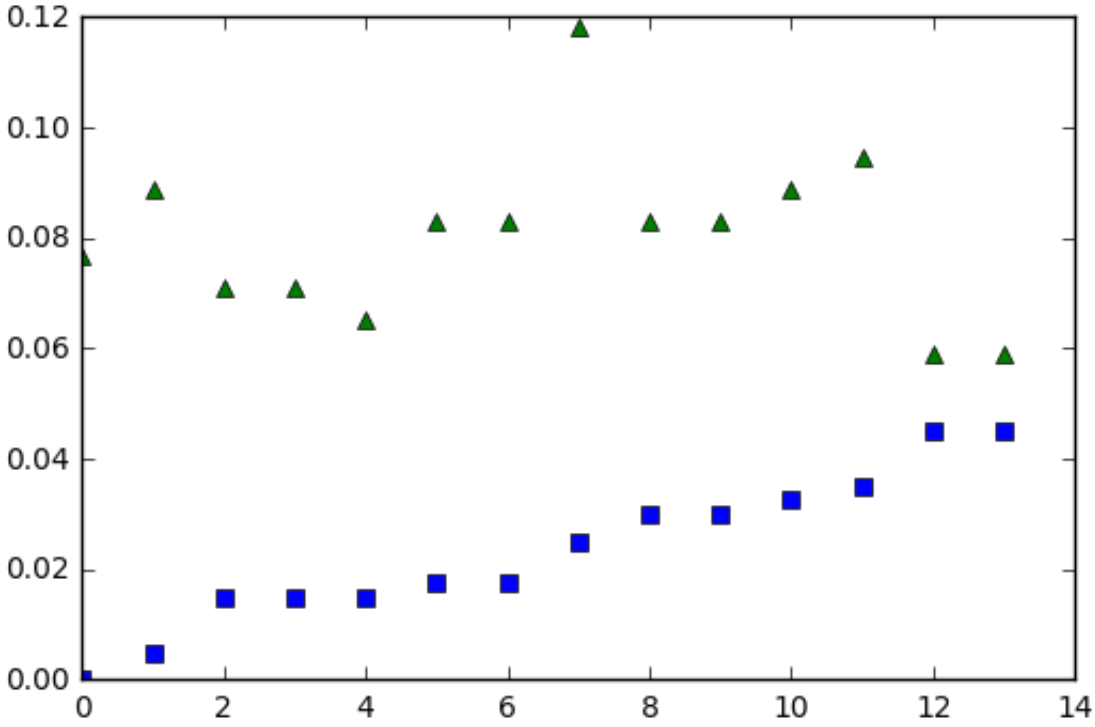
```
0.109826589595
0.0352303523035
0.0355029585799
```

In [3]:

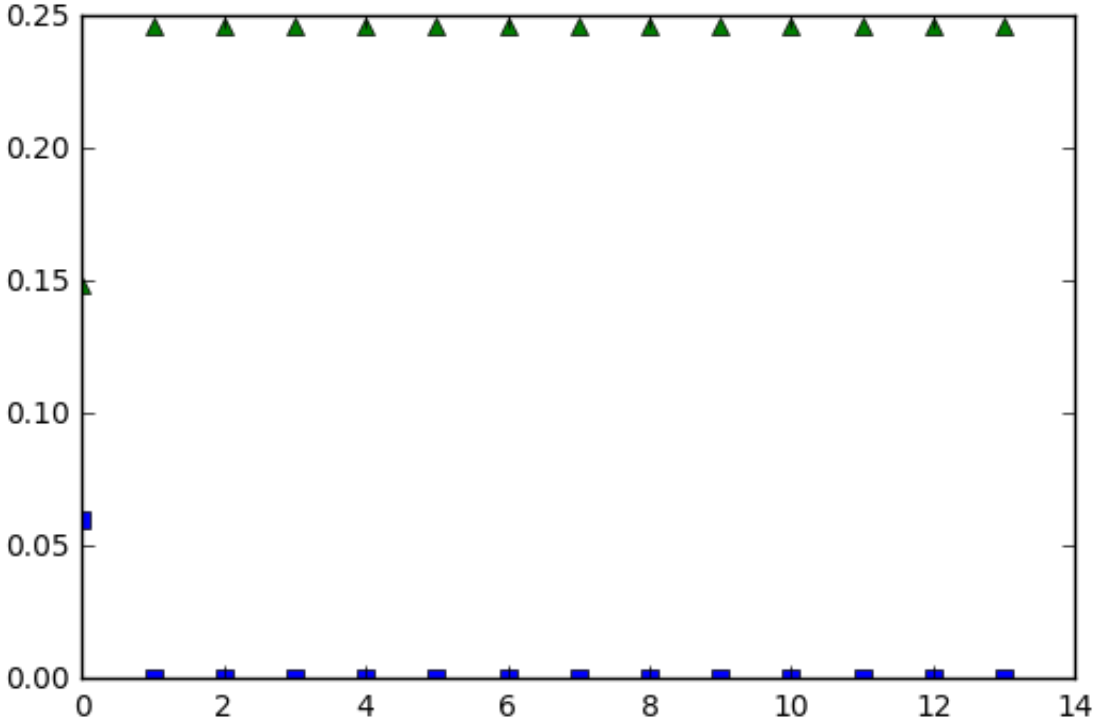
```
%matplotlib inline
import numpy as np
from sklearn.metrics import hamming_loss
from sklearn.naive_bayes import GaussianNB

from sklearn import tree

from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt



data = np.genfromtxt('wdbc.txt', delimiter=',')
size = [50, 200, 400]

for sz in size:
    train, test = data[0:sz, :], data[sz:, :]
    X_trn, y_trn = train[:, 2:], train[:,1]
    X_tst, y_tst = test[:, 2:], test[:,1]
    errtrn = np.zeros(14)
    errtst = np.zeros(14)
    tmp = 0
    for i in range(1, 15):
        clf = tree.DecisionTreeClassifier(criterion='entropy', min_sam
ples_leaf=i, random_state=6)
        clf = clf.fit(X_trn, y_trn)
        y_pred = clf.predict(X_tst)
        y_predtrn = clf.predict(X_trn)
        errtst[tmp] = hamming_loss(y_tst, y_pred)
        errtrn[tmp] = hamming_loss(y_trn, y_predtrn)
        tmp = tmp+1
        del clf
    plt.plot(errtrn, 'bs', errtst, 'g^')
    plt.show()
```

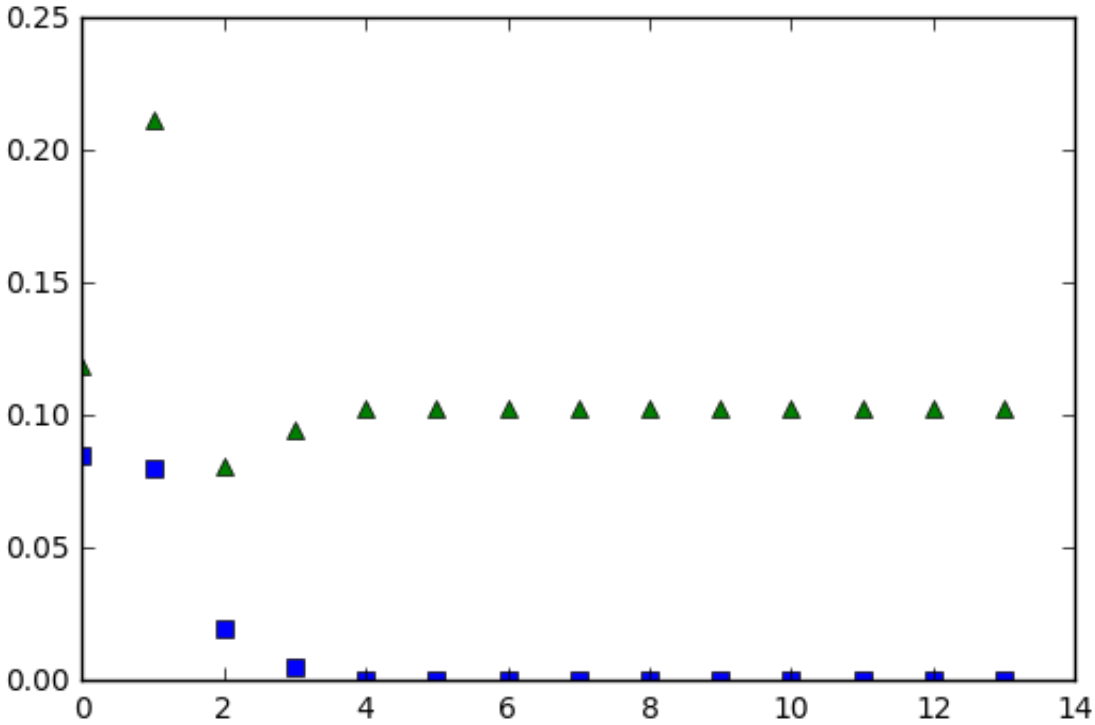

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
2
4
6
8
10
12
14

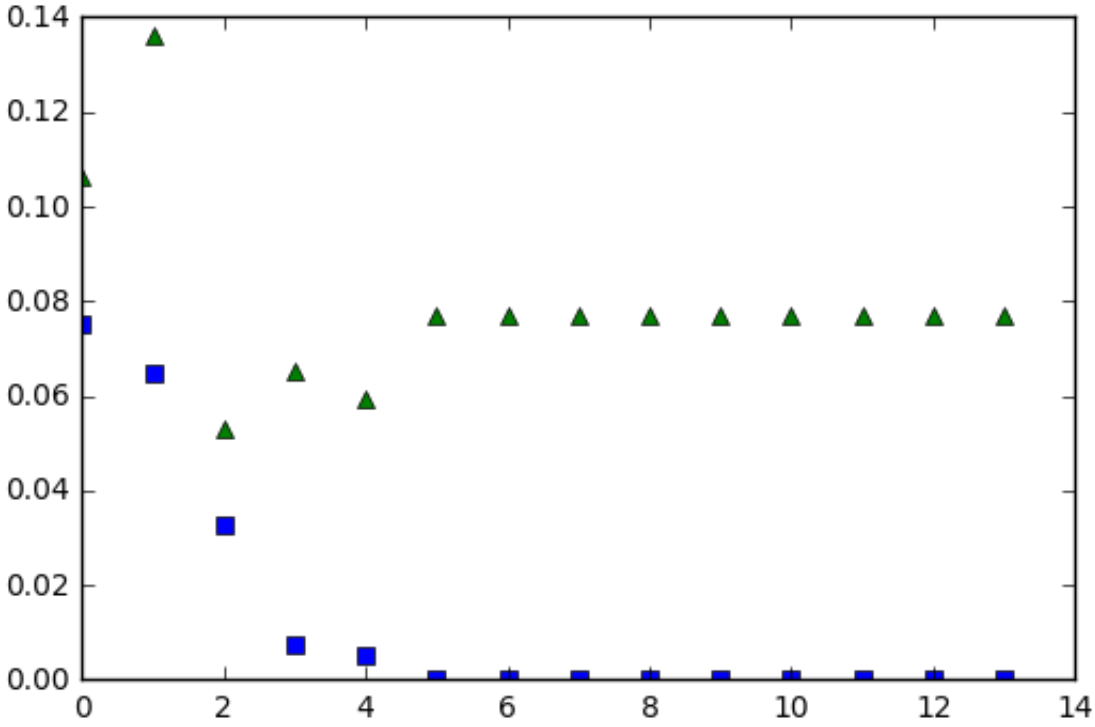

0.20
0.15
0.10
0.05
0.00
0
2
4
6
8
10
12
14


0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8
10
12
14

In [5]:
```
%matplotlib inline
import numpy as np
from sklearn.metrics import hamming_loss
from sklearn.naive_bayes import GaussianNB

from sklearn import tree

from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt



data = np.genfromtxt('wdbc.txt', delimiter=',')
size = [50, 200, 400]

for sz in size:
    train, test = data[0:sz, :], data[sz:, :]
    X_trn, y_trn = train[:, 2:], train[:,1]
    X_tst, y_tst = test[:, 2:], test[:,1]
    errtrn = np.zeros(14)
    errtst = np.zeros(14)
    tmp = 0
    for i in range(1,15):
        clf = tree.DecisionTreeClassifier(criterion='entropy', max_dep
th=i, random_state=6)
        clf = clf.fit(X_trn, y_trn)
        y_pred = clf.predict(X_tst)
        y_predtrn = clf.predict(X_trn)
        errtst[tmp] = hamming_loss(y_tst, y_pred)
        errtrn[tmp] = hamming_loss(y_trn, y_predtrn)
        tmp = tmp+1
        del clf
    plt.plot(errtrn, 'bs', errtst, 'g^')
    plt.show()
```


0.25
0.20
0.15
0.10
0.05
0.00
0
2
4
6
8
10
12
14


0.25
0.20
0.15
0.10
0.05
0.00
0
2
4
6
8
10
12
14



In [ ]: