In [1]:
```
%matplotlib inline
```

In [2]:
```
import matplotlib.pyplot as plt
```

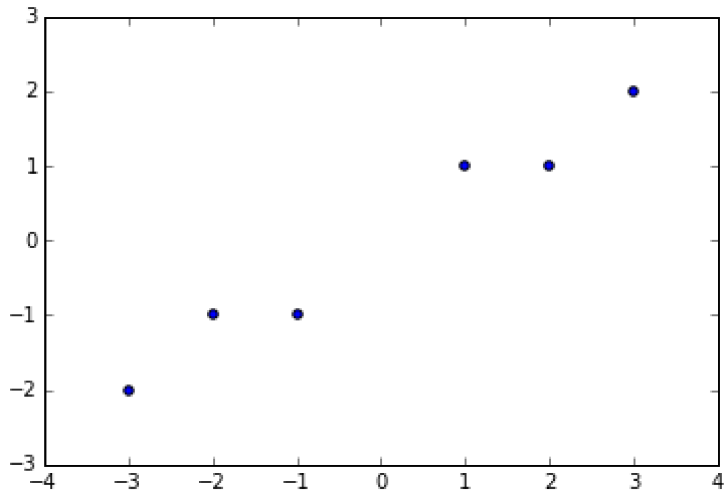
In [3]:
```
import numpy as np
```

In [4]:
```
# Generisanje slucajnih brojeva - random
# potrebno za kreiranje "toy example" programa
```

In [5]:
```
# toy example

import numpy as np
from sklearn.lda import LDA
X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
y = np.array([1, 1, 1, 2, 2, 2])
plt.scatter(X[:,0],X[:,1])
clf = LDA()
clf.fit(X, y)
```

Out[5]:
```
LDA(n_components=None, priors=None, shrinkage=None, solver='svd',
  store_covariance=False, tol=0.0001)
```



In [6]:
```
# KLASA 1
mean = [0, 0]
cov  = [ [1,-0.5] , [-0.5,1 ]  ]
x1, y1 = np.random.multivariate_normal(mean, cov, 500).T

x1_test, y1_test = np.random.multivariate_normal(mean, cov, 100).T
```

In [7]:
```
np.random.random_integers(1,10,2)
```

Out[7]:
```
array([3, 2])
```

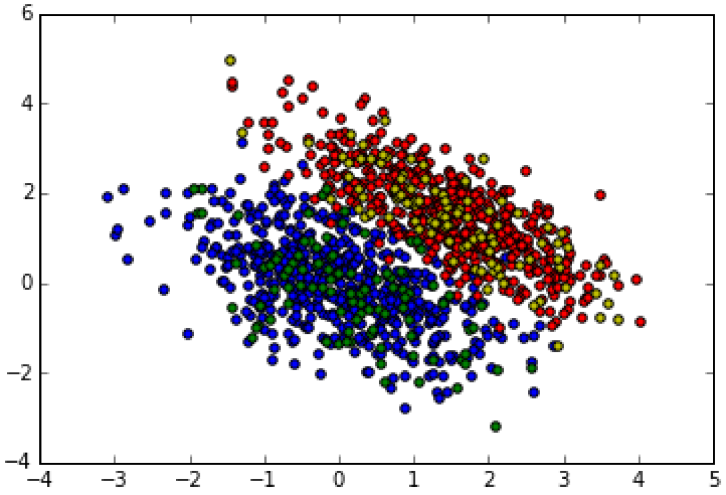
In [8]:
```
np.random.randn(10)
```

Out[8]:
```
array([ 1.11007092, -0.24341141, -0.46680127,  0.34626651, -1.17135972,
       -1.03091606,  0.06517114,  1.80389311,  0.11320554, -0.1013184 ])
```

In [9]:
```
# KLASA 2
mean = [1.5,1.5]
cov =  [ [1,-0.8],[-0.8,1 ]  ]
x2, y2 = np.random.multivariate_normal(mean, cov, 500).T
x2_test, y2_test = np.random.multivariate_normal(mean, cov, 100).T
```

In [10]:
```
plt.scatter(x1,y1)
plt.scatter(x2,y2,c=u'r')
plt.scatter(x1_test, y1_test, c=u'g')
plt.scatter(x2_test, y2_test, c=u'y')
```

Out[10]:
```
<matplotlib.collections.PathCollection at 0x1357b128>
```



In [11]:
```
# ova naredba se menjala u zavisnosti od verzija
from sklearn.lda import LDA
```

VEZBA 1. KREIRATI 2D GAUSOVSKU RASPODELU ZA RAZLICITE VREDNOSTI 'mean' i 'cov' # VEZBA 2. ISPLOTOVATI OVE RASPODELE. NACRTATI TRENING I TEST PRIMERE ...

In [12]:
```
X1 = np.hstack([x1,x2])
Y2 =np.hstack([y1,y2])
XX = np.vstack([X1,Y2])
XX=XX.T
print(XX)
print(XX[0])
print(x1[0], y1[0])
```

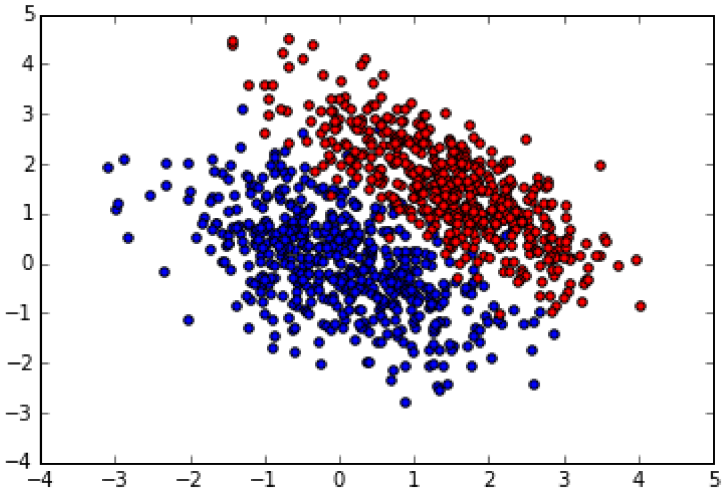
```
[[-0.52766766  0.75148864]
 [-0.46238619  0.54170927]
 [ 0.42970282 -0.12035764]
 ...,
 [ 1.93757408  0.83046094]
 [ 2.29273359  2.04233062]
 [ 2.67061142  0.2445634 ]]
[-0.52766766  0.75148864]
(-0.52766766055656333, 0.75148863687908918)
```

In [13]:
```
X1_test = np.hstack([x1_test,x2_test])
X1_test.shape
Y2_test =np.hstack([y1_test,y2_test])
XX_test = np.vstack([X1_test,Y2_test])
XX_test=XX_test.T
XX_test.shape
```

Out[13]:
```
(200L, 2L)
```

In [14]:
```
plt.scatter(XX[:500,0], XX[:500,1])
plt.scatter(XX[500:,0], XX[500:,1], c=u'r')
```

Out[14]:
```
<matplotlib.collections.PathCollection at 0x1362dbe0>
```



In [15]:
```
# hstack, vstack funkcije se takodje mogu koristiti
# concatenate

# helping stackers c_ i r_  :: columns and rows
```

In [16]:
```
target = np.ones(1000)
```

In [17]:
```
target[500:] = 2
```

In [18]:
```
# Kreiranje LDA klasifikatora. u NOVIJOJ VERZIJI OVA KOMANDA JE "LinearDiscrim
inantAnalysis"
clf=  LDA()
```

In [19]:
```
target.shape
```

Out[19]:
```
(1000L,)
```

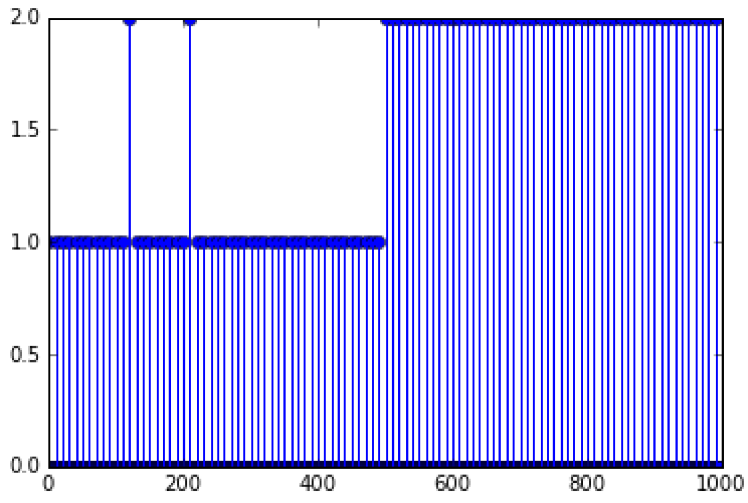
In [20]:
```
clf.fit(XX,target)
```

Out[20]:
```
LDA(n_components=None, priors=None, shrinkage=None, solver='svd',
  store_covariance=False, tol=0.0001)
```

In [21]:
```
predicted = np.zeros(1000)
for i in range(1,len(predicted),10):
    predicted[i] = clf.predict(XX[i]  + 0.05 *
np.array([np.random.randn(),np.random.randn()]) )
```

In [22]:
```
plt.stem(predicted)
```

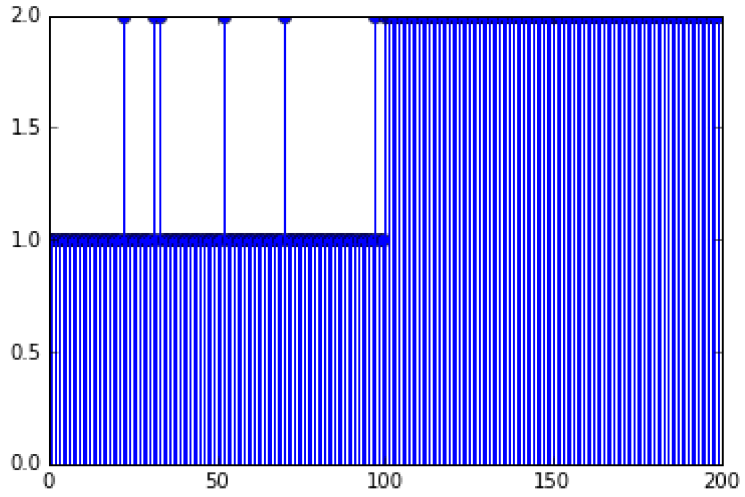
Out[22]:
```
<Container object of 3 artists>
```



In [23]:
```
y_predicted = clf.predict(XX_test)
```

In [24]:
```
plt.stem(y_predicted)
```

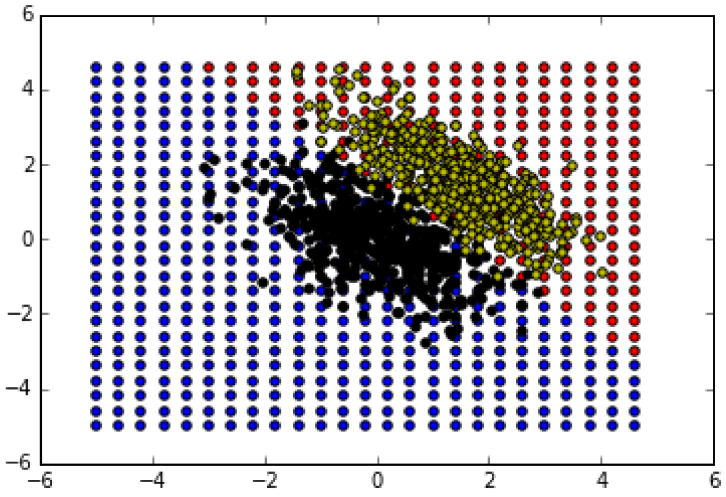
Out[24]:
```
<Container object of 3 artists>
```



In [45]:
```
x = np.arange(-5,5,0.4)
y = np.arange(-5,5,0.4)
for i in range(len(x)):
    for j in range(len(y)):
        temp  = clf.predict([x[i] , y[j]])
        if (temp ==1):
            plt.scatter(x[i], y[j])
        else:
            plt.scatter(x[i], y[j],c=u'r')


plt.scatter(XX[:500,0], XX[:500,1],c=u'k')
plt.scatter(XX[500:,0], XX[500:,1], c=u'y')
```

Out[45]:
```
<matplotlib.collections.PathCollection at 0x2e5107f0>
```



In [ ]:

In [ ]: