In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [19]:

```
from sklearn import datasets, naive_bayes, model_selection
```

In [5]:

```
digits = datasets.load_digits()
```

In [16]:

```
digits.data[:5]
```

Out[16]:

```
array([[  0.,   0.,   5.,  13.,   9.,   1.,   0.,   0.,   0.,   0.,
13.,
         15.,  10.,  15.,   5.,   0.,   0.,   3.,  15.,   2.,   0.,
11.,
          8.,   0.,   0.,   4.,  12.,   0.,   0.,   8.,   8.,   0.,
  0.,
          5.,   8.,   0.,   0.,   9.,   8.,   0.,   0.,   4.,  11.,
  0.,
          1.,  12.,   7.,   0.,   0.,   2.,  14.,   5.,  10.,  12.,
  0.,
          0.,   0.,   0.,   6.,  13.,  10.,   0.,   0.,   0.],
       [  0.,   0.,   0.,  12.,  13.,   5.,   0.,   0.,   0.,   0.,
  0.,
         11.,  16.,   9.,   0.,   0.,   0.,   0.,   3.,  15.,  16.,
  6.,
          0.,   0.,   0.,   7.,  15.,  16.,  16.,   2.,   0.,   0.,
  0.,
          0.,   1.,  16.,  16.,   3.,   0.,   0.,   0.,   0.,   1.,
16.,
         16.,   6.,   0.,   0.,   0.,   0.,   1.,  16.,  16.,   6.,
  0.,
          0.,   0.,   0.,   0.,  11.,  16.,  10.,   0.,   0.],
       [  0.,   0.,   0.,   4.,  15.,  12.,   0.,   0.,   0.,   0.,
  3.,
         16.,  15.,  14.,   0.,   0.,   0.,   0.,   8.,  13.,   8.,
16.,
          0.,   0.,   0.,   0.,   1.,   6.,  15.,  11.,   0.,   0.,
  0.,
          1.,   8.,  13.,  15.,   1.,   0.,   0.,   0.,   9.,  16.,
16.,
          5.,   0.,   0.,   0.,   0.,   3.,  13.,  16.,  16.,  11.,
  5.,
          0.,   0.,   0.,   0.,   3.,  11.,  16.,   9.,   0.],
       [  0.,   0.,   7.,  15.,  13.,   1.,   0.,   0.,   0.,   8.,
13.,
          6.,  15.,   4.,   0.,   0.,   0.,   2.,   1.,  13.,  13.,
  0.,
          0.,   0.,   0.,   0.,   2.,  15.,  11.,   1.,   0.,   0.,
  0.,
          0.,   0.,   1.,  12.,  12.,   1.,   0.,   0.,   0.,   0.,
  0.,
          1.,  10.,   8.,   0.,   0.,   0.,   8.,   4.,   5.,  14.,
  9.,
          0.,   0.,   0.,   7.,  13.,  13.,   9.,   0.,   0.],
       [  0.,   0.,   0.,   1.,  11.,   0.,   0.,   0.,   0.,   0.,
  0.,
          7.,   8.,   0.,   0.,   0.,   0.,   0.,   1.,  13.,   6.,
  2.,
          2.,   0.,   0.,   0.,   7.,  15.,   0.,   9.,   8.,   0.,
  0.,
          5.,  16.,  10.,   0.,  16.,   6.,   0.,   0.,   4.,  15.,
16.,
         13.,  16.,   1.,   0.,   0.,   0.,   0.,   3.,  15.,  10.,
  0.,
          0.,   0.,   0.,   0.,   2.,  16.,   4.,   0.,   0.]])
```

Признаки целые числа.

In [42]:

```
model_selection.cross_val_score(naive_bayes.BernoulliNB(), digits.data, digits.target).mean()
```

Out[42]:

```
0.82582365077805819
```

In [43]:

```
model_selection.cross_val_score(naive_bayes.MultinomialNB(), digits.data, digits.target).mean()
```

Out[43]:

```
0.87087714897350532
```

In [44]:

```
model_selection.cross_val_score(naive_bayes.GaussianNB(), digits.data, digits.target).mean()
```

Out[44]:

```
0.81860038035501381
```

Так как признаки целые числа, но не только {0,1}, то неудивительно то, что наивный байес с мультиномиальным распределением показал наилучший результат. При этом на этом датасете предположение о независимости признаков оказывается не самым лучшим.

In [12]:

```
cancer = datasets.load_breast_cancer()
```

In [17]:

```
cancer.data[:5]
```

Out[17]:

```
array([[  1.79900000e+01,   1.03800000e+01,   1.22800000e+02,
          1.00100000e+03,   1.18400000e-01,   2.77600000e-01,
          3.00100000e-01,   1.47100000e-01,   2.41900000e-01,
          7.87100000e-02,   1.09500000e+00,   9.05300000e-01,
          8.58900000e+00,   1.53400000e+02,   6.39900000e-03,
          4.90400000e-02,   5.37300000e-02,   1.58700000e-02,
          3.00300000e-02,   6.19300000e-03,   2.53800000e+01,
          1.73300000e+01,   1.84600000e+02,   2.01900000e+03,
          1.62200000e-01,   6.65600000e-01,   7.11900000e-01,
          2.65400000e-01,   4.60100000e-01,   1.18900000e-01],
       [  2.05700000e+01,   1.77700000e+01,   1.32900000e+02,
          1.32600000e+03,   8.47400000e-02,   7.86400000e-02,
          8.69000000e-02,   7.01700000e-02,   1.81200000e-01,
          5.66700000e-02,   5.43500000e-01,   7.33900000e-01,
          3.39800000e+00,   7.40800000e+01,   5.22500000e-03,
          1.30800000e-02,   1.86000000e-02,   1.34000000e-02,
          1.38900000e-02,   3.53200000e-03,   2.49900000e+01,
          2.34100000e+01,   1.58800000e+02,   1.95600000e+03,
          1.23800000e-01,   1.86600000e-01,   2.41600000e-01,
          1.86000000e-01,   2.75000000e-01,   8.90200000e-02],
       [  1.96900000e+01,   2.12500000e+01,   1.30000000e+02,
          1.20300000e+03,   1.09600000e-01,   1.59900000e-01,
          1.97400000e-01,   1.27900000e-01,   2.06900000e-01,
          5.99900000e-02,   7.45600000e-01,   7.86900000e-01,
          4.58500000e+00,   9.40300000e+01,   6.15000000e-03,
          4.00600000e-02,   3.83200000e-02,   2.05800000e-02,
          2.25000000e-02,   4.57100000e-03,   2.35700000e+01,
          2.55300000e+01,   1.52500000e+02,   1.70900000e+03,
          1.44400000e-01,   4.24500000e-01,   4.50400000e-01,
          2.43000000e-01,   3.61300000e-01,   8.75800000e-02],
       [  1.14200000e+01,   2.03800000e+01,   7.75800000e+01,
          3.86100000e+02,   1.42500000e-01,   2.83900000e-01,
          2.41400000e-01,   1.05200000e-01,   2.59700000e-01,
          9.74400000e-02,   4.95600000e-01,   1.15600000e+00,
          3.44500000e+00,   2.72300000e+01,   9.11000000e-03,
          7.45800000e-02,   5.66100000e-02,   1.86700000e-02,
          5.96300000e-02,   9.20800000e-03,   1.49100000e+01,
          2.65000000e+01,   9.88700000e+01,   5.67700000e+02,
          2.09800000e-01,   8.66300000e-01,   6.86900000e-01,
          2.57500000e-01,   6.63800000e-01,   1.73000000e-01],
       [  2.02900000e+01,   1.43400000e+01,   1.35100000e+02,
          1.29700000e+03,   1.00300000e-01,   1.32800000e-01,
          1.98000000e-01,   1.04300000e-01,   1.80900000e-01,
          5.88300000e-02,   7.57200000e-01,   7.81300000e-01,
          5.43800000e+00,   9.44400000e+01,   1.14900000e-02,
          2.46100000e-02,   5.68800000e-02,   1.88500000e-02,
          1.75600000e-02,   5.11500000e-03,   2.25400000e+01,
          1.66700000e+01,   1.52200000e+02,   1.57500000e+03,
          1.37400000e-01,   2.05000000e-01,   4.00000000e-01,
          1.62500000e-01,   2.36400000e-01,   7.67800000e-02]])
```

Признаки вещественные числа

In [45]:

```
model_selection.cross_val_score(naive_bayes.BernoulliNB(), cancer.data, cancer.target).mean()
```

Out[45]:

```
0.62742040285899936
```

In [46]:

```
model_selection.cross_val_score(naive_bayes.MultinomialNB(), cancer.data, cancer.target).mean()
```

Out[46]:

```
0.89457904019307521
```

In [47]:

```
model_selection.cross_val_score(naive_bayes.GaussianNB(), cancer.data, cancer.target).mean()
```

Out[47]:

```
0.9367492806089297
```

Так как признаки вещественные, то гауссовское распределние оказалось наилучшим.

Вопросы:

1) Максимальное качество получилось на breast_cancer 0.94 с использованием наивного байеса с гауссовским распределением.

2) Максимальное качество получилось на digits 0.87 с использованием наивного байеса с мультиномиальным распределением.

3) Верны: (c), (d): мультиномиальное распределние лучше всего на целых неотрицательных признаках. На вещественных признаках лучше всего Гауссовское распределение.

Рассмотрим также датасет ирисы Фишера

In [32]:

```
iris = datasets.load_iris()
```

In [33]:

```
iris.data[:5]
```

Out[33]:

```
array([[ 5.1,  3.5,  1.4,  0.2],
       [ 4.9,  3. ,  1.4,  0.2],
       [ 4.7,  3.2,  1.3,  0.2],
       [ 4.6,  3.1,  1.5,  0.2],
       [ 5. ,  3.6,  1.4,  0.2]])
```

In [39]:

```
model_selection.cross_val_score(naive_bayes.BernoulliNB(), iris.data, iris.target).mean()
```

Out[39]:

```
0.33333333333333331
```

In [40]:

```
model_selection.cross_val_score(naive_bayes.MultinomialNB(), iris.data, iris.target).mean()
```

Out[40]:

```
0.96078431372549022
```

In [41]:

```
model_selection.cross_val_score(naive_bayes.GaussianNB(), iris.data,
iris.target).mean()
```

Out[41]:

```
0.93423202614379086
```

Хотя и числа здесь вещественные, но все они представлены с точностью 1 знак после запятой, этим можно объяснить то, что наивный байес с мультиномиальным распределением оказался лучше, чем с гауссовским.