Лабораторное занятие 5. Задачи для реализации метода ближайших соседей. Работа с библиотекой sklearn.neighbors import KNeighborsClassifier

5-зертханалық сабақ. Жақын көршілер әдісіне есептер.

```
In [ ]: %matplotlib inline
        import matplotlib.pyplot as plt

        import pandas
        data = pandas.read_csv("http://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.data", header=None)
```

```
In [ ]: data.head()
```

```
In [5]: names = []
        for line in text.split("\r\n"):
            if len(line) > 0:
                if (line[0] != "|" and line[0] != "1"):
                    names.append(line.split(":")[0])
        names.append("spam/not")
```

```
In [6]: print(names)
```

```
['word_freq_make', 'word_freq_address', 'word_freq_all', 'word_freq_3d', 'word_freq_our', 'word_freq_over', 'word_freq_remo
ve', 'word_freq_internet', 'word_freq_order', 'word_freq_mail', 'word_freq_receive', 'word_freq_will', 'word_freq_people',
'word_freq_report', 'word_freq_addresses', 'word_freq_free', 'word_freq_business', 'word_freq_email', 'word_freq_you', 'wor
d_freq_credit', 'word_freq_your', 'word_freq_font', 'word_freq_000', 'word_freq_money', 'word_freq_hp', 'word_freq_hpl', 'w
ord_freq_george', 'word_freq_650', 'word_freq_lab', 'word_freq_labs', 'word_freq_telnet', 'word_freq_857', 'word_freq_data
', 'word_freq_415', 'word_freq_85', 'word_freq_technology', 'word_freq_1999', 'word_freq_parts', 'word_freq_pm', 'word_freq
_direct', 'word_freq_cs', 'word_freq_meeting', 'word_freq_original', 'word_freq_project', 'word_freq_re', 'word_freq_edu',
'word_freq_table', 'word_freq_conference', 'char_freq_;', 'char_freq_(', 'char_freq_[', 'char_freq_!', 'char_freq_$', 'char
_freq_#', 'capital_run_length_average', 'capital_run_length_longest', 'capital_run_length_total', 'spam/not']
```

```
In [7]: data.columns = names
```

```
In [8]: data.head()
```

Out[8]:

	word_freq_make	word_freq_address	word_freq_all	word_freq_3d	word_freq_our	word_freq_over	word_freq_remove	word_freq_internet	word_freq_order
0	0.00	0.64	0.64	0.0	0.32	0.00	0.00	0.00	0.00
1	0.21	0.28	0.50	0.0	0.14	0.28	0.21	0.07	0.00
2	0.06	0.00	0.71	0.0	1.23	0.19	0.19	0.12	0.64
3	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31
4	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31

6 rows × 58 columns

```
In [9]: features = data.iloc[:, : -2]
        response = data.iloc[:, -1 ]
```

```
In [10]: import numpy as np
         X = features.values
         Y = response.values
```

```
In [11]: print(X)
         print(Y)
```

```
[[  0.00000000e+00   6.40000000e-01   6.40000000e-01 ...,   0.00000000e+00
    3.75600000e+00   6.10000000e+01]
 [  2.10000000e-01   2.80000000e-01   5.00000000e-01 ...,   4.80000000e-02
    5.11400000e+00   1.01000000e+02]
 [  6.00000000e-02   0.00000000e+00   7.10000000e-01 ...,   1.00000000e-02
    9.82100000e+00   4.85000000e+02]
 ...,
 [  3.00000000e-01   0.00000000e+00   3.00000000e-01 ...,   0.00000000e+00
    1.40400000e+00   6.00000000e+00]
 [  9.60000000e-01   0.00000000e+00   0.00000000e+00 ...,   0.00000000e+00
    1.14700000e+00   5.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   6.50000000e-01 ...,   0.00000000e+00
    1.25000000e+00   5.00000000e+00]]
[1 1 1 ..., 0 0 0]
```

```
In [12]: Количество писем в датасете
         data.shape[0]
```

```
Out[12]: 4601
```

```
In [13]: Доля спама
         (data.loc[data["spam/not"] == 1].shape[0]) / data.shape[0]
```

```
Out[13]: 0.39404477207546186
```

```
In [14]: for i in data.columns:
             print(i)
```

```
word_freq_make
word_freq_address
word_freq_all
word_freq_3d
word_freq_our
```

In [15]:
```
#группы признаков:
#    word_freq_WORD
#    char_freq_I
#    capital_run_length_(average, longest, total)

#кол-во определенного слова
#кол-во определенных символов
#Заглавные буквы (средняя продолжительность, самая длинная последовательность, всего)
```

In [16]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size= X.shape[0] - 3000, random_state=42)
```

In [17]:
```
from sklearn.preprocessing import scale
X_test_norm = scale(X_test)
X_train_norm = scale(X_train)
```

In [18]:
```
from sklearn.neighbors import KNeighborsClassifier

clfKNN = KNeighborsClassifier()
```

In [19]:
```
clfKNN.fit(X_train_norm, Y_train)
Y_pred = clfKNN.predict(X_test_norm)
```

In [20]:
```
from sklearn import metrics
```

In [21]:
```
print("recall = " + str(metrics.recall_score(Y_test, Y_pred)))
print("precision = " + str(metrics.precision_score(Y_test, Y_pred)))
print("f1 = " + str(metrics.f1_score(Y_test, Y_pred)))
print("accuracy = " + str(metrics.accuracy_score(Y_test, Y_pred)))
```

```
recall = 0.839150227618
precision = 0.900651465798
f1 = 0.868813825609
accuracy = 0.895690193429
```

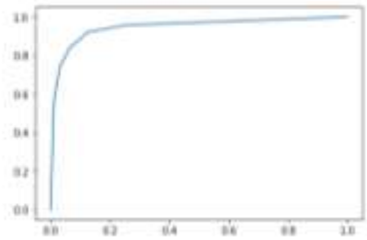
In [22]:
```
Y_pred_proba = clfKNN.predict_proba(X_test_norm)[:,1]
```

In [23]:
```
fprKNN, tprKNN, thresholdsKNN = metrics.roc_curve(Y_test, Y_pred_proba)
roc_aucKNN  = metrics.auc(fprKNN, tprKNN)
print(roc_aucKNN)
```

```
0.944983462487
```

In [24]:
```
plt.plot(fprKNN, tprKNN)
```

Out[24]:
```
[<matplotlib.lines.Line2D at 0x2c3a64bc50>]
```



In [27]:
```
from sklearn.model_selection import GridSearchCV

KNN = KNeighborsClassifier()
tuned_parametersKNN = [{"n_neighbors": np.arange(1, 5),
                        "p": [1, 2],
                        "weights" : ["uniform", "distance"]}]
gsKNN = GridSearchCV(KNN, tuned_parametersKNN)
```

In [28]:
```
gsKNN.fit(X_train, Y_train)
```

Out[28]:
```
GridSearchCV(cv=None, error_score='raise',
       estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform'),
       fit_params={}, iid=True, n_jobs=1,
       param_grid=[{'n_neighbors': array([1, 2, 3, 4]), 'p': [1, 2], 'weights': ['uniform', 'distance']}],
       pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
       scoring=None, verbose=0)
```

In [29]:
```
gsKNN.best_params_
```

Out[29]:
```
{'n_neighbors': 4, 'p': 1, 'weights': 'distance'}
```