Лабораторная работа №7. Метод опорных векторов и логистическая регрессия в решении задач машинного обучения.

Часть 1. Метод опорных векторов.

Используется текстовый (https://habr.com/ru/post/264339/) набор данных The 20 newsgroups (https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html).

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import classification_report, roc_curve, roc_auc_score
from sklearn.datasets import fetch_20newsgroups
import os
import requests

%matplotlib inline

pd.options.display.max_columns = None
```

In [2]:

```
data = fetch_20newsgroups()
data.target_names
```

Out[2]:

```
['alt.atheism',
 'comp.graphics',
 'comp.os.ms-windows.misc',
 'comp.sys.ibm.pc.hardware',
 'comp.sys.mac.hardware',
 'comp.windows.x',
 'misc.forsale',
 'rec.autos',
 'rec.motorcycles',
 'rec.sport.baseball',
 'rec.sport.hockey',
 'sci.crypt',
 'sci.electronics',
 'sci.med',
 'sci.space',
 'soc.religion.christian',
 'talk.politics.guns',
 'talk.politics.mideast',
 'talk.politics.misc',
 'talk.religion.misc']
```

Вычислим TF-IDF-признаки для всех текстов

In [3]:

```
categories = ["comp.sys.mac.hardware", "rec.sport.baseball", "sci.space", "talk.religio
n.misc"]
data_train = fetch_20newsgroups(subset="train", remove=("headers", "footers", "quotes"
), categories=categories)

vectorizer = TfidfVectorizer(stop_words="english")
vectors_train = vectorizer.fit_transform(data_train.data)
```

Подберем минимальный лучший параметр C для метода опорных векторов с линейным ядром при помощи кроссвалидации по k-блокам.

In [4]:

```
cv = KFold(n_splits=10)
params = {"kernel": ["linear"], "C": np.power(10.0, np.arange(-5, 6))}
gscv = GridSearchCV(SVC(), params, scoring="accuracy", cv=cv)
gscv.fit(vectors_train, data_train.target)
gscv.best_params_
```

Out[4]:

```
{'C': 1.0, 'kernel': 'linear'}
```

Обучим модель с помощью метода опорных векторов по всей выборке с лучшим параметром C

In [5]:

```
data_test = fetch_20newsgroups(subset="test", remove=("headers", "footers", "quotes"),
categories=categories)
vectors_test = vectorizer.transform(data_test.data)

svc_best = gscv.best_estimator_
y_pred = svc_best.predict(vectors_test)
```

Оценим получившуюся классификацию.

In [6]:

```
print(classification_report(data_test.target, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.86      0.90       385
           1       0.90      0.89      0.89       397
           2       0.76      0.92      0.83       394
           3       0.93      0.76      0.84       251

    accuracy                           0.87      1427
   macro avg       0.88      0.86      0.87      1427
weighted avg       0.88      0.87      0.87      1427
```

Найдем 10 слов с наибольшим по модулю весом.

In [7]:

```
all_idx = np.argsort(np.absolute(svc_best.coef_.toarray()))
idx = np.array(all_idx[0,-10:])
print(np.array(vectorizer.get_feature_names())[idx].tolist())
```

```
['scsi', 'monitor', 'mouse', 'team', 'problem', 'se', 'drive', 'apple', 'b
aseball', 'mac']
```