РК2 Ерохин И.А. ИУ5-24М

Классификатор №1: KNeighborsClassifier

Классификатор №2: Complement Naive Bayes - CNB

In [10]:
```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_re
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_
from sklearn.metrics import roc_curve, roc_auc_score

# from sklearn.naive_bayes import MultinomialNB
# from sklearn.linear_model import LogisticRegression

from sklearn.naive_bayes import ComplementNB
from sklearn.neighbors import KNeighborsClassifier

import seaborn as sns
from collections import Counter
from sklearn.datasets import fetch_20newsgroups
import matplotlib.pyplot as plt

%matplotlib inline
sns.set(style="ticks")
```

In [11]:
```
categories = ["rec.sport.hockey", "sci.electronics", "sci.med"]
newsgroups = fetch_20newsgroups(subset='train', categories=categories)
data = newsgroups['data']
```

In [12]:
```
def accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray) -> Dict[int, float]:
    """
    Вычисление метрики accuracy для каждого класса
    y_true - истинные значения классов
    y_pred - предсказанные значения классов
    Возвращает словарь: ключ - метка класса,
    значение - Accuracy для данного класса
    """
    # Для удобства фильтрации сформируем Pandas DataFrame
    d = {'t': y_true, 'p': y_pred}
    df = pd.DataFrame(data=d)
    # Метки классов
    classes = np.unique(y_true)
    # Результирующий словарь
    res = dict()
    # Перебор меток классов
    for c in classes:
        # отфильтруем данные, которые соответствуют
        # текущей метке класса в истинных значениях
        temp_data_flt = df[df['t']==c]
        # расчет accuracy для заданной метки класса
        temp_acc = accuracy_score(
            temp_data_flt['t'].values,
            temp_data_flt['p'].values)
        # сохранение результата в словарь
        res[c] = temp_acc
    return res

def print_accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray):
    """
    Вывод метрики accuracy для каждого класса
    """
    accs = accuracy_score_for_classes(y_true, y_pred)
    if len(accs)>0:
        print('Метка \t Accuracy')
    for i in accs:
        print('{} \t {}'.format(i, accs[i]))
```

In [13]:
```
vocabVect = CountVectorizer()
vocabVect.fit(data)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(corpusVocab)))
```

```
Количество сформированных признаков - 29970
```

In [14]:
```
for i in list(corpusVocab)[1:10]:
    print('{}={}'.format(i, corpusVocab[i]))
```

```
carl=7229
sol1=25350
gps=13433
caltech=7081
edu=10976
lydick=17538
subject=26157
re=22713
krillean=16582
```

In [15]:
```
test_features = vocabVect.transform(data)
test_features
```

Out[15]:
```
<1785x29970 sparse matrix of type '<class 'numpy.int64'>'
        with 266767 stored elements in Compressed Sparse Row format>
```

In [16]:
```
# Размер нулевой строки
len(test_features.todense()[0].getA1())
```

Out[16]:
```
29970
```

In [17]:
```
vocabVect.get_feature_names()[100:120]
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning:
Function get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and w
ill be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

Out[17]:
```
['03756',
 '038',
 '04',
 '0400',
 '04046',
 '041505',
 '042',
 '042100',
 '043426',
 '0435',
 '043654',
 '044045',
 '044140',
 '044323',
 '044636',
 '045046',
 '0453',
 '0458',
 '047',
 '0483']
```

In [18]:
```
def VectorizeAndClassify(vectorizers_list, classifiers_list):
    for v in vectorizers_list:
        for c in classifiers_list:
            pipeline1 = Pipeline([("vectorizer", v), ("classifier", c)])
            score = cross_val_score(pipeline1, newsgroups['data'], newsgroups['target
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('============================')
```

In [19]:
```
vectorizers_list = [CountVectorizer(vocabulary = corpusVocab), TfidfVectorizer(vocabul
classifiers_list = [KNeighborsClassifier(), ComplementNB()]
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000001200':
3,
                            '00014': 4, '000256': 5, '001': 6, '0010': 7,
                            '001004': 8, '001323': 9, '001642': 10, '00309': 11,
                            '003221': 12, '003258u19250': 13, '0033': 14,
                            '003800': 15, '004021809': 16, '004158': 17,
                            '004418': 18, '004627': 19, '005': 20, '00500': 21,
                            '005148': 22, '005150': 23, '005512': 24,
                            '0059': 25, '007': 26, '0078': 27, '008': 28,
                            '008561': 29, ...})
Модель для классификации - KNeighborsClassifier()
Accuracy = 0.687955182072829
============================
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000001200':
3,
                            '00014': 4, '000256': 5, '001': 6, '0010': 7,
                            '001004': 8, '001323': 9, '001642': 10, '00309': 11,
                            '003221': 12, '003258u19250': 13, '0033': 14,
                            '003800': 15, '004021809': 16, '004158': 17,
                            '004418': 18, '004627': 19, '005': 20, '00500': 21,
                            '005148': 22, '005150': 23, '005512': 24,
                            '0059': 25, '007': 26, '0078': 27, '008': 28,
                            '008561': 29, ...})
Модель для классификации - ComplementNB()
Accuracy = 0.984873949579832
============================
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000001200':
3,
                            '00014': 4, '000256': 5, '001': 6, '0010': 7,
                            '001004': 8, '001323': 9, '001642': 10, '00309': 11,
                            '003221': 12, '003258u19250': 13, '0033': 14,
                            '003800': 15, '004021809': 16, '004158': 17,
                            '004418': 18, '004627': 19, '005': 20, '00500': 21,
                            '005148': 22, '005150': 23, '005512': 24,
                            '0059': 25, '007': 26, '0078': 27, '008': 28,
                            '008561': 29, ...})
Модель для классификации - KNeighborsClassifier()
Accuracy = 0.9198879551820728
============================
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000001200':
3,
                            '00014': 4, '000256': 5, '001': 6, '0010': 7,
                            '001004': 8, '001323': 9, '001642': 10, '00309': 11,
                            '003221': 12, '003258u19250': 13, '0033': 14,
                            '003800': 15, '004021809': 16, '004158': 17,
                            '004418': 18, '004627': 19, '005': 20, '00500': 21,
                            '005148': 22, '005150': 23, '005512': 24,
                            '0059': 25, '007': 26, '0078': 27, '008': 28,
                            '008561': 29, ...})
Модель для классификации - ComplementNB()
Accuracy = 0.9815126050420169
============================
```

Наилучшие результаты показал CountVectorizer с классификатором Complement Naive Bayes - CNB (0.985)