Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Рубежный контроль №2
«Методы обработки текстов»
по дисциплине «Методы машинного обучения»
Вариант 2

ИСПОЛНИТЕЛЬ:

Болотин Андрей Сергеевич
Группа ИУ5-23М

____

"__"___________2021 г.

Москва 2021

Необходимо решить задачу классификации текстов на основе любого выбранного Вами датасета (кроме примера, который рассматривался в лекции). Классификация может быть бинарной или многоклассовой. Целевой признак из выбранного Вами датасета может иметь любой физический смысл, примером является задача анализа тональности текста. Необходимо сформировать два варианта векторизации признаков - на основе CountVectorizer и на основе TfidfVectorizer. В качестве классификаторов необходимо использовать два классификатора по варианту для Вашей группы: LinearSVC, MNB

Болотин Андрей ИУ5-23М

В [22]:
```
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_report
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_error, median_absolute_error, r2_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
import seaborn as sns
from collections import Counter
from sklearn.datasets import fetch_20newsgroups
import matplotlib.pyplot as plt
```

```
categories = ['talk.politics.guns',
 'rec.autos',
 'rec.motorcycles',
 'rec.sport.baseball',
 'rec.sport.hockey'
              ]
newsgroups = fetch_20newsgroups(subset='train', categories=categories)
data = newsgroups['data']
```

```
data
```

```
['From: kirsch@staff.tc.umn.edu (Dave \'Almost Cursed the Jays\' Kirsch)\nSubject: Going to a Cubbies game ..  \nKeywords: ti
ckets?, parking?, parka?\nNntp-Posting-Host: staff.tc.umn.edu\nOrganization: Li\'l Carlos and the Hormones\nDistribution: usa
\nLines: 30\n\n  Well, after suffering from an intense fit of Minnesota-induced cabin fever,\nI\'ve decided to road trip to M
ilwaukee and take in a couple of games this\nweekend. A couple games at County stadium will be great to relieve tension, \nbu
t I thought "Why not go to Wrigley for a game too?"  \n\n  I see the Cubs are playing the Phillies on Sat (2:05 start, I beli
eve\nthat\'s Eastern time listed). I figured it would be fun to bounce down to\nWrigley for the day game and live it up a lit
tle. I\'m wondering if anyone\n(esp. Cubbie fans) have some advice on: \n\n  1) If I\'m taking 41 (Skokie Hwy) south until it
runs into 94, what\'s the \n     best way to get to Wrigley? I\'m planning on getting there an hour or  \n     two early and
paying through the nose for parking to keep things easy. \n\n  2) Is it probable that I\'ll be able to walk up and get bleach
er seats (2 or\n     3) on game day? I figure since it\'s early in the year, Ryno\'s out and \n     the weather isn\'t great
I should be able to get tickets. If not, what\'s \n     the best way to get advance tickets; can I call the Cubs\' ticket off
ice\n     directly and pick up tickets at the will call window?  \n\n  3) Any advice on where to eat before or after the gam
e? \n\n  4) Do they allow inflatable I-luv-ewe dolls (present from Lundy) into the \n     bleachers? :-) \n \n-- \nDave Hung
Like a Jim Acker Slider Kirsch      Blue Jays - Do it again in \'93 \nkirsch@staff.tc.umn.edu                             New .. q
uotes out of context!\n"Not to beat a dead horse, but it\'s been a couple o\' weeks .. this \n disappoints me..punishments..d
ischarges..jackhammering.." - Stephen Lawrence \n',
 'From: dholle15@ursa.calvin.edu (David Hollebeek)\nSubject: Phillies Mailing List?\nNntp-Posting-Host: ursa\nOrganization: C
alvin College\nLines: 7\n\nAnyone know of a phillies mailing list out there? .... they don\'t get much\ncoverage up here in G
rand Rapids, MI *sob*\n\n--\n------------------------------------------------------------------------------\n"Elaborate .sig
```

В [11]:
```
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]))
```

В [12]:
```
vocabVect = CountVectorizer()
vocabVect.fit(data)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(corpusVocab)))
```

```
Количество сформированных признаков - 39162
```

В [13]:
```
for i in list(corpusVocab)[1:10]:
    print('{}={}'.format(i, corpusVocab[i]))
```

```
kirsch=21713
staff=33536
tc=34826
umn=36293
edu=14819
dave=12985
almost=6726
cursed=12684
the=35084
```

В [14]:
```
test_features = vocabVect.transform(data)
test_features
```

Out[14]:
```
<2935x39162 sparse matrix of type '<class 'numpy.int64'>'
	with 448025 stored elements in Compressed Sparse Row format>
```

В [15]:
```
len(test_features.todense()[0].getA1())
```

Out[15]:
```
39162
```

В [16]:
```
vocabVect.get_feature_names()[100:120]
```

Out[16]:
```
['013939',
 '014',
 '014237',
 '014638',
 '015',
 '015043',
 '015209',
 '015225',
 '015415',
 '015442',
 '015908',
 '015936',
 '016',
 '0161',
 '01730',
 '018',
 '01810',
 '01854',
 '018801285',
 '019']
```

```
В [17]: 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'], scoring='accuracy', cv=3).mean()
                    print('Векторизация - {}'.format(v))
                    print('Модель для классификации - {}'.format(c))
                    print('Accuracy = {}'.format(score))
                    print('===========================')
```

```
В [23]: vectorizers_list = [CountVectorizer(vocabulary = corpusVocab), TfidfVectorizer(vocabulary = corpusVocab)]
        classifiers_list = [LinearSVC(), MultinomialNB()]
        VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000000004': 3,
                            '0000000005': 4, '0000000667': 5, '0000001200': 6,
                            '0001': 7, '000152': 8, '0002': 9, '000256': 10,
                            '0003': 11, '0005111312': 12, '0005111312na1em': 13,
                            '0005111312na3em': 14, '000601': 15, '000710': 16,
                            '00072': 17, '000851': 18, '000mi': 19,
                            '000miles': 20, '000rpm': 21, '000s': 22,
                            '000th': 23, '001': 24, '0010': 25, '0011': 26,
                            '001211': 27, '0013': 28, '001319': 29, ...})
Модель для классификации - LinearSVC()
Accuracy = 0.937646611562652
===========================
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000000004': 3,
                            '0000000005': 4, '0000000667': 5, '0000001200': 6,
                            '0001': 7, '000152': 8, '0002': 9, '000256': 10,
                            '0003': 11, '0005111312': 12, '0005111312na1em': 13,
                            '0005111312na3em': 14, '000601': 15, '000710': 16,
                            '00072': 17, '000851': 18, '000mi': 19,
                            '000miles': 20, '000rpm': 21, '000s': 22,
                            '000th': 23, '001': 24, '0010': 25, '0011': 26,
                            '001211': 27, '0013': 28, '001319': 29, ...})
Модель для классификации - MultinomialNB()
Accuracy = 0.9649054827589328
===========================
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000000004': 3,
                            '0000000005': 4, '0000000667': 5, '0000001200': 6,
                            '0001': 7, '000152': 8, '0002': 9, '000256': 10,
                            '0003': 11, '0005111312': 12, '0005111312na1em': 13,
                            '0005111312na3em': 14, '000601': 15, '000710': 16,
                            '00072': 17, '000851': 18, '000mi': 19,
                            '000miles': 20, '000rpm': 21, '000s': 22,
                            '000th': 23, '001': 24, '0010': 25, '0011': 26,
                            '001211': 27, '0013': 28, '001319': 29, ...})
Модель для классификации - LinearSVC()
Accuracy = 0.9649051346163086
===========================
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000000004': 3,
                            '0000000005': 4, '0000000667': 5, '0000001200': 6,
                            '0001': 7, '000152': 8, '0002': 9, '000256': 10,
                            '0003': 11, '0005111312': 12, '0005111312na1em': 13,
                            '0005111312na3em': 14, '000601': 15, '000710': 16,
                            '00072': 17, '000851': 18, '000mi': 19,
                            '000miles': 20, '000rpm': 21, '000s': 22,
                            '000th': 23, '001': 24, '0010': 25, '0011': 26,
                            '001211': 27, '0013': 28, '001319': 29, ...})
Модель для классификации - MultinomialNB()
Accuracy = 0.9597957934622993
===========================
```


```
Модель для классификации - LinearSVC()
Accuracy = 0.9649051346163086
===========================
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000000004': 3,
                            '0000000005': 4, '0000000667': 5, '0000001200': 6,
                            '0001': 7, '000152': 8, '0002': 9, '000256': 10,
                            '0003': 11, '0005111312': 12, '0005111312na1em': 13,
                            '0005111312na3em': 14, '000601': 15, '000710': 16,
                            '00072': 17, '000851': 18, '000mi': 19,
                            '000miles': 20, '000rpm': 21, '000s': 22,
                            '000th': 23, '001': 24, '0010': 25, '0011': 26,
                            '001211': 27, '0013': 28, '001319': 29, ...})
Модель для классификации - MultinomialNB()
Accuracy = 0.9597957934622993
===========================
```


Лучшая точность у CountVectorizer с MultinominalNB