Московский Государственный Технический Университет имени

Н.Э.Баумана

Факультет Информатика и системы управления

Кафедра ИУ-5

«Методы машинного обучения»

Рубежный контроль №2

По дисциплине

Выполнили студенты группы ИУ-5 24М

Линь Юаньси

Москва 2022г

Тема: Методы обработки текстов

Решение задачи классификации текстов

Необходимо сформировать два варианта векторизации признаков - на основе CountVectorizer и на основе TfidfVectorizer.

Группа	Классификатор №1	Классификатор №2
ИУ5И-24М	KNeighborsClassifier	Complement Naive Bayes - CNB

```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from scipy import stats
import torch
from IPython.display import Image
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split

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

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.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

```python
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
```

```python
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]))
```

```python
# Загрузка данных
df = pd.read_csv("Test.csv")
df.head()
```

	text	label
0	I always wrote this series off as being a comp...	0
1	1st watched 12/7/2002 - 3 out of 10(Dir-Steve ...	0
2	This movie was so poorly written and directed ...	0
3	The most interesting thing about Miryang (Secr...	1
4	when i first read about "berlin am meer" i did...	0

```
# Сформируем общий словарь для обучения моделей из обучающей и тестовой выборки
vocab_list = df['text'].tolist()
vocab_list[1:10]
```

```
["1st watched 12/7/2002 - 3 out of 10(Dir-Steve Purcell): Typical Mary Kate & Ashley fare with a few more kisses. It looks to me like the girls are getting pretty tir
ed of this stuff and it will be interesting what happens to them if they ever decide to split up and go there own ways. In this episode of their adventures they are i
nterns in Rome for a `fashion' designer who puts them right into the mailroom to learn what working hard is all about(I guess..). Besides the typical flirtations with
boys there is nothing much else except the Rome scenario until about ¾ way into the movie when it's finally revealed why they are getting fired, then re-hired, then f
ired again, then re-hired again. This is definetly made by people who don't understand the corporate world and it shows in their interpretation of it. Maybe the real
world will be their next adventure(if there is one.). Even my kids didn't seem to care for this boring `adventure' in the make-believe. Let's see they probably only h
ave a couple of years till their legal adults. We'll see what happens then.",
 'This movie was so poorly written and directed I fell asleep 30 minutes through the movie. The jokes in the movie are corny and even though the plot is interesting a
t some angles, it is too far fetched and at some points- ridiculous. If you are 11 or older you will overlook the writing in the movie and be disappointed, but if you
are 10 or younger this is a film that will capture your attention and be amazed with all the stunts (which I might add are poorly done) and wish you were some warrior
to. The casting in this movie wasn\'t very good, and the music was very disappointing because it was like they were trying to build up the tension but it didn\'t fit
at all. On a scale of 1-10 (10 being excellent, 1 being horrible) the acting in this movie is a 4. Brenda Song is talented in comedy, but with this kind of movie, in
some of the more serious scenes, her acting was laughable. When she made some of her "fighting" poses, I started laughing out loud. I think the worst thing about this
movie is definitely the directing, for example, the part where her enemy turns out to be the person the evil villain is possesing, how her voice turns dark and evil,
I think that was incredibly stupid, and how Wendy\'s (Brenda Song)teachers were all her teachers at school being possessed by monks, that was pretty ridiculous to. So
to sumamrize it all, a disappointing movie, but okay if you\'re 10 or under.',
```

```
df.shape
```

```
(5000, 2)
```

```
# Сформируем общий словарь для обучения моделей из обучающей и тестовой выборки
vocab_list = df['text'].tolist()
vocab_list[1:10]
```

```
["1st watched 12/7/2002 - 3 out of 10(Dir-Steve Purcell): Typical Mary Kate & Ashley fare with a few more kisses. It looks to me like the girls are getting pretty tir
ed of this stuff and it will be interesting what happens to them if they ever decide to split up and go there own ways. In this episode of their adventures they are i
nterns in Rome for a `fashion' designer who puts them right into the mailroom to learn what working hard is all about(I guess..). Besides the typical flirtations with
boys there is nothing much else except the Rome scenario until about ¾ way into the movie when it's finally revealed why they are getting fired, then re-hired, then f
ired again, then re-hired again. This is definetly made by people who don't understand the corporate world and it shows in their interpretation of it. Maybe the real
world will be their next adventure(if there is one.). Even my kids didn't seem to care for this boring `adventure' in the make-believe. Let's see they probably only h
```

```
vocabVect = CountVectorizer()
vocabVect.fit(vocab_list)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(corpusVocab)))
```

```
Количество сформированных признаков - 39126
```

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

```
wrote=38700
this=34932
series=30826
off=24255
as=2274
being=3467
complete=7225
stink=33134
fest=12941
```

```
test_features = vocabVect.transform(vocab_list)
```

```
test_features
```

```
<5000x39126 sparse matrix of type '<class 'numpy.int64'>'
	with 685304 stored elements in Compressed Sparse Row format>
```

```
test_features.todense()
```

```
matrix([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]])
```

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

```
39126
```

```
# Непустые значения нулевой строки
[i for i in test_features.todense()[0].getA1() if i>0]
```

```
[1,
 1,
 2,
 1,
 1,
 2,
 2,
 1,
 1,
 1,
 2,
 9,
 1,
 1,
 1,
```

```
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 will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
['165',
 '168',
 '16a',
 '16b',
 '16ieme',
 '16mm',
 '16s',
 '16th',
 '16ème',
 '17',
 '170',
 '1701',
 '1710',
 '175',
 '1790',
 '1794',
 '17th',
 '18',
 '180',
 '1800']
```

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

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

```
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '00015': 2, '001': 3,
                                           '003830': 4, '0069': 5, '007': 6, '0079': 7,
                                           '0083': 8, '00am': 9, '00s': 10, '01': 11, '02': 12,
                                           '0230': 13, '04': 14, '041': 15, '05': 16, '06': 17,
                                           '07': 18, '07b': 19, '09': 20, '0s': 21, '10': 22,
                                           '100': 23, '1000': 24, '1001': 25, '100b': 26,
                                           '100k': 27, '101': 28, '102': 29, ...})
Модель для классификации - ComplementNB()
Accuracy = 0.813400105093027
```

```
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '00015': 2, '001': 3,
                                           '003830': 4, '0069': 5, '007': 6, '0079': 7,
                                           '0083': 8, '00am': 9, '00s': 10, '01': 11, '02': 12,
                                           '0230': 13, '04': 14, '041': 15, '05': 16, '06': 17,
                                           '07': 18, '07b': 19, '09': 20, '0s': 21, '10': 22,
                                           '100': 23, '1000': 24, '1001': 25, '100b': 26,
                                           '100k': 27, '101': 28, '102': 29, ...})
Модель для классификации - KNeighborsClassifier()
Accuracy = 0.6017986078654617
```

```
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '00015': 2, '001': 3,
                            '003830': 4, '0069': 5, '007': 6, '0079': 7,
                            '0083': 8, '00am': 9, '00s': 10, '01': 11, '02': 12,
                            '0230': 13, '04': 14, '041': 15, '05': 16, '06': 17,
                            '07': 18, '07b': 19, '09': 20, '0s': 21, '10': 22,
                            '100': 23, '1000': 24, '1001': 25, '100b': 26,
                            '100k': 27, '101': 28, '102': 29, ...})
Модель для классификации - ComplementNB()
Accuracy = 0.8214019861093808
```

```
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '00015': 2, '001': 3,
                            '003830': 4, '0069': 5, '007': 6, '0079': 7,
                            '0083': 8, '00am': 9, '00s': 10, '01': 11, '02': 12,
                            '0230': 13, '04': 14, '041': 15, '05': 16, '06': 17,
                            '07': 18, '07b': 19, '09': 20, '0s': 21, '10': 22,
                            '100': 23, '1000': 24, '1001': 25, '100b': 26,
                            '100k': 27, '101': 28, '102': 29, ...})
Модель для классификации - KNeighborsClassifier()
Accuracy = 0.6558028130268304
```

Вывод: вариант векторизации признаков TfidfVectorizer в паре с классификатором Complement Naive Bayes показал наилучшее качество.