Рубежный контроль №2

Андрианов А.А.

ИУ5-23М

Задание

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:

1. Классификатор №1 - LinearSVC
2. Классификатор №2 - Multinomial Naive Bayes - MNB

Для каждого метода необходимо оценить качество классификации. Сделайте вывод о том, какой вариант векторизации признаков в паре с каким классификатором показал лучшее качество.

```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from scipy import stats
from sklearn.datasets import load_iris, load_boston
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
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.svm import LinearSVC
from sklearn.naive_bayes import MultinomialNB
import matplotlib.pyplot as plt
```

```
# Загрузка данных
# Набор твитов в пандемию с указанием эмоциональной окраски
df = pd.read_csv("data.csv", encoding='latin-1')
df = df[1:5000]
df
```

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
1	3800	48752	UK	16-03-2020	advice Talk to your neighbours family to excha...	Positive
2	3801	48753	Vagabonds	16-03-2020	Coronavirus Australia: Woolworths to give elde...	Positive
3	3802	48754	NaN	16-03-2020	My food stock is not the only one which is emp...	Positive
4	3803	48755	NaN	16-03-2020	Me, ready to go at supermarket during the #COV...	Extremely Negative
5	3804	48756	Ã T: 36.319708,-82.363649	16-03-2020	As news of the regionÂ s first confirmed COVID...	Positive
...	...	...	...	...	...	...
4995	8794	53746	Nagpur, India	18-03-2020	Why is Government not transmitting benefits of...	Positive
4996	8795	53747	London	18-03-2020	"As long as we're not seeing markets I would c...	Extremely Positive
4997	8796	53748	Victoria, London	18-03-2020	Will school fees be refunded if the #coronavir...	Neutral
4998	8797	53749	United States	18-03-2020	#USD continues its dominance, markets rebounde...	Negative
4999	8798	53750	DC, dreaming of Norway	18-03-2020	I guess the new normal is hitting the grocery ...	Neutral

4999 rows × 6 columns

```
# Удаление лишних столбцов
df = df.drop('UserName', axis = 1)
df = df.drop('ScreenName', axis = 1)
df = df.drop('TweetAt', axis = 1)
df = df.drop('Location', axis = 1)
df.head()
```

	OriginalTweet	Sentiment
1	advice Talk to your neighbours family to excha...	Positive
2	Coronavirus Australia: Woolworths to give elde...	Positive
3	My food stock is not the only one which is emp...	Positive
4	Me, ready to go at supermarket during the #COV...	Extremely Negative
5	As news of the regionÂ s first confirmed COVID...	Positive

```
# Кодирование эмоциональной окраски
df['Sentiment'] = df['Sentiment'].astype('category')
df['Sentiment'] = df['Sentiment'].cat.codes
df.head()
```

	OriginalTweet	Sentiment
1	advice Talk to your neighbours family to excha...	4
2	Coronavirus Australia: Woolworths to give elde...	4
3	My food stock is not the only one which is emp...	4
4	Me, ready to go at supermarket during the #COV...	0
5	As news of the regionÂ s first confirmed COVID...	4

```
df.shape
```

```
(4999, 2)
```

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

```
['Coronavirus Australia: Woolworths to give elderly, disabled dedicated shopping hours amid COVID-19 outbreak https://t.co/bInC
 "My food stock is not the only one which is empty...\r\r\n\r\r\nPLEASE, don't panic, THERE WILL BE ENOUGH FOOD FOR EVERYONE if
 "Me, ready to go at supermarket during the #COVID19 outbreak.\r\r\n\r\r\nNot because I'm paranoid, but because my food stock i
 'As news of the regionÂ\x92s first confirmed COVID-19 case came out of Sullivan County last week, people flocked to area store
 'Cashier at grocery store was sharing his insights on #Covid_19 To prove his credibility he commented "I\'m in Civics class sc
 "Was at the supermarket today. Didn't buy toilet paper. #Rebel\r\r\n\r\r\n#toiletpapercrisis #covid_19 https://t.co/eVXkQLIdAZ
 'Due to COVID-19 our retail store and classroom in Atlanta will not be open for walk-in business or classes for the next two w
 "For corona prevention,we should stop to buy things with the cash and should use online payment methods because corona can spr
 "All month there hasn't been crowding in the supermarkets or restaurants, however reducing all the hours and closing the malls
```

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

```
Количество сформированных признаков - 17445
```

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

```
infinitely=8041
alias=1276
irregular=8266
smarter=14158
smartmonkey=14161
urqbcvg3of=16149
tk3g3zzqqx=15461
relentless=12857
unnoticed=16058
closely=3443
ofâ=10971
8mzp8tqxp8=781
brexitdryrun=2578
nlwejz1xef=10652
pockets=11815
dspdospsb0=5153
qjinliq8rg=12400
```

```
    qjini1q8rg=12466
    campaignspot=2855
    generating=6725
    irp76nqu8e=8264
    gcx7mz3m21=6697
    shii=13896
    covid19nigeria=4091
    roadwarriors=13192
    adidas=1060
    reebok=12771
    twenty=15839
    4aau45ytcu=467
    storeclosures=14666
    rocking=13217
    cb2tebhj5x=3037
    initial=8076
    removing=12890
    overage=11212
    capped=2914
    applies=1555
    jmuhammadtv=8492
    bowlinggreen=2488
    yamapn77ew=17179
    nau5rnxssy=10450
    excerpt=5742
    triggering=15713
    hollywood=7503
    ink3gvurqk=8084
    salisbury=13431
    6fz2lbqsuz=646
    tzotf1pbze=15870
    byronhttps=2776
    sp=14333
    toyewvovt3=15597
    2xinzxci3b=332
    macys=9541
    httqhcuaif=7632
    lr9zvesn4w=9441
    vmn5tcrs9s=16441
    rpb8s8bzmw=13278
    cloversoftsg=3460
    hlacpwlhvv=7466
    customersâ=4342
```

```
tfidfv = TfidfVectorizer(ngram_range=(1,3))
tfidf_ngram_features = tfidfv.fit_transform(vocab_list)
tfidf_ngram_features
```

```
    <4999x230500 sparse matrix of type '<class 'numpy.float64'>'
            with 452561 stored elements in Compressed Sparse Row format>
```

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

```
    230500
```

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

```

```
    [0.08401773824904957,
     0.10413616574450421,
     0.10413616574450421,
     0.09595605317581173,
     0.10413616574450421,
     0.10413616574450421,
     0.0703581658379043,
     0.10413616574450421,
     0.10413616574450421,
     0.03909376171035052,
     0.08777594060711925,
     0.10413616574450421,
     0.09935110664014131,
     0.10413616574450421,
     0.10413616574450421,
     0.06739230177449762,
     0.10413616574450421,
     0.10413616574450421,
     0.07638662735743722,
     0.10413616574450421,
     0.10413616574450421,
     0.09117099407144882,
     0.10413616574450421,
     0.10413616574450421,
    0.09117099407144882,
    0.10413616574450421,
    0.10413616574450421,
    0.06105325896634548,
    0.09117099407144882,
    0.10413616574450421,
    0.0933226451206084,
    0.10413616574450421,
    0.10413616574450421,
    0.03948152251673741,
    0.10413616574450421,
    0.10413616574450421,
    0.06524065072533805,
    0.10413616574450421,
    0.10413616574450421,
    0.08935179903049259,
    0.10413616574450421,
    0.10413616574450421,
    0.17870359806098518,
    0.10413616574450421,
    0.10413616574450421,
    0.10413616574450421,
    0.10413616574450421,
    0.037162704146206434,
    0.10413616574450421,
    0.10413616574450421,
    0.17028506510383185,
    0.10413616574450421,
    0.10413616574450421,
    0.0933226451206084,
    0.10413616574450421,
    0.04355584283679552,
    0.10413616574450421,
    0.10413616574450421,
```

```python
# Оценка качества работы обоих способов векторизации на обоих методах классификации:
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['OriginalTweet'], df['Sentiment'], scoring='accuracy', cv=3).mean()
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('==========================')
```

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

```
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '0000009375': 2, '0000hrs': 3,
                            '008': 4, '00am': 5, '00pdsup4wb': 6, '00pm': 7,
                            '01': 8, '0101': 9, '01625': 10, '0203': 11,
                            '029': 12, '02ddkwsnxo': 13, '04': 14, '0508': 15,
                            '06': 16, '0600': 17, '0618': 18, '0645': 19,
                            '0712128888': 20, '08': 21, '0800': 22, '0808': 23,
                            '086ohsc4ox': 24, '08smp18fiq': 25,
                            '09093052802': 26, '0aaj71zczs': 27,
                            '0acif25540': 28, '0blnzayudb': 29, ...})
Модель для классификации - LinearSVC()
Accuracy = 0.4474893736738847
==========================
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '0000009375': 2, '0000hrs': 3,
                            '008': 4, '00am': 5, '00pdsup4wb': 6, '00pm': 7,
                            '01': 8, '0101': 9, '01625': 10, '0203': 11,
                            '029': 12, '02ddkwsnxo': 13, '04': 14, '0508': 15,
                            '06': 16, '0600': 17, '0618': 18, '0645': 19,
                            '0712128888': 20, '08': 21, '0800': 22, '0808': 23,
                            '086ohsc4ox': 24, '08smp18fiq': 25,
                            '09093052802': 26, '0aaj71zczs': 27,
                            '0acif25540': 28, '0blnzayudb': 29, ...})
Модель для классификации - MultinomialNB()
Accuracy = 0.3270627987247689
==========================
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000009375': 2, '0000hrs': 3,
                            '008': 4, '00am': 5, '00pdsup4wb': 6, '00pm': 7,
                            '01': 8, '0101': 9, '01625': 10, '0203': 11,
                            '029': 12, '02ddkwsnxo': 13, '04': 14, '0508': 15,
                            '06': 16, '0600': 17, '0618': 18, '0645': 19,
                            '0712128888': 20, '08': 21, '0800': 22, '0808': 23,
                            '086ohsc4ox': 24, '08smp18fiq': 25,
                            '09093052802': 26, '0aaj71zczs': 27,
                            '0acif25540': 28, '0blnzayudb': 29, ...})
Модель для классификации - LinearSVC()
Accuracy = 0.43448693214538364
```

```
===========================
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000009375': 2, '0000hrs': 3,
                            '008': 4, '00am': 5, '00pdsup4wb': 6, '00pm': 7,
                            '01': 8, '0101': 9, '01625': 10, '0203': 11,
                            '029': 12, '02ddkwsnxo': 13, '04': 14, '0508': 15,
                            '06': 16, '0600': 17, '0618': 18, '0645': 19,
                            '0712128888': 20, '08': 21, '0800': 22, '0808': 23,
                            '086ohsc4ox': 24, '08smp18fiq': 25,
                            '09093052802': 26, '0aaj71zczs': 27,
                            '0acif25540': 28, '0blnzayudb': 29, ...})
Модель для классификации - MultinomialNB()
Accuracy = 0.381073965278973
===========================
```

Вывод

Лучшее качество показал вариант векторизации TfidfVectorizer в паре с классификатором LinearSVC. Метрика accuracy составила 0.4.

✓ 7s completed at 1:19 AM ● ✕