Рубежный контроль №2

Студент: Пинская Ника Марковна
Группа: ИУ5-21М
Вариант (номер по списку группы): 9
Тема: Методы обработки текстов

Решение задачи классификации текстов

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 группы ИУ5-21М:

Классификатор №1: LogisticRegression
Классификатор №2: Multinomial Naive Bayes - MNB

```
#Импорт библиотек

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, classificat
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_square
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
import seaborn as sns
from collections import Counter
from sklearn.model_selection import train_test_split

#from sklearn.datasets import fetch_20newsgroups
import matplotlib.pyplot as plt

%matplotlib inline
```

✓ 11s completed at 9:13 AM

```
# Подключение к gogle диску
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, ca
```

```
# Вывод содержимого папки на диске
import os
data_root = '/content/drive/MyDrive/MMO'
print(os.listdir(data_root))
```

```
['pulitzer-circulation-data.csv', 'mmsa-icu-beds2.csv', 'movies.csv.zip', '
```

```
#Распаковка архива с датасетом
!unzip /content/drive/MyDrive/MMO/Tweets.csv.zip
```

```
Archive:  /content/drive/MyDrive/MMO/Tweets.csv.zip
replace Tweets.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: Tweets.csv
```

```
# Загрузка данных
data = pd.read_csv("Tweets.csv")
data.head()
```

	tweet_id	airline_sentiment	airline_sentiment_confidence	ne
0	570306133677760513	neutral	1.0000	
1	570301130888122368	positive	0.3486	
2	570301083672813571	neutral	0.6837	
3	570301031407624196	negative	1.0000	
4	570300817074462722	negative	1.0000	

```
data.shape
```

```
(14640, 15)
```

```
# data = data.overview.fillna(' ')
```

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

```
['jnardino',
 'yvonnalynn',
 'jnardino',
 'jnardino',
 'jnardino',
 'cjmcginnis',
 'pilot',
 'dhepburn',
 'YupitsTate']
```

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

```
Количество сформированных признаков - 7704
```

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

```
jnardino=3479
yvonnalynn=7668
cjmcginnis=1391
pilot=5660
dhepburn=1873
yupitstate=7667
idk_but_youtube=2980
hypercamilax=2941
mollanderson=5011
```

```
# categories = ["rec.motorcycles", "rec.sport.baseball", "sci.electronics","sci.
# newsgroups = fetch_20newsgroups(subset='train', categories=categories)
# data = newsgroups['data']
```

```
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]))


# x = data.overview.fillna(' ')
# vocabVect = CountVectorizer()
# vocabVect.fit(data[x])
# corpusVocab = vocabVect.vocabulary_
# print('Количество сформированных признаков - {}'.format(len(corpusVocab)))


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

Использование класса *CountVectorizer*

Подсчитывает количество слов словаря, входящих в данный текст.

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

```
<15x7704 sparse matrix of type '<class 'numpy.int64'>'
        with 0 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())
```

```
7704
```

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

```
[]
```

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

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: Fut
  warnings.warn(msg, category=FutureWarning)
['_mulazoe',
 '_paulinha9',
 '_raulll',
 '_robprice',
 '_rtuck',
 '_samanthaakira',
 '_saranguyen',
 '_stephanieejayy',
 '_therobynshow',
 '_troyjohnson',
 '_yohoka_',
 'a_a_ron_cbus',
 'a_for_adnauseam',
 'a_geechi',
 'a_lichtenfels',
 'a_panfalone',
 'aakopel',
 'aaron_lawlor',
 'aaronaebie',
 'aarongirson']
```

Использование класса *TfidfVectorizer*

Вычисляет специфичность текста в корпусе текстов на основе метрики *TF-IDF*.

```
tfidfv = TfidfVectorizer(ngram_range=(1,3))
```

```
tfidfv = TfidfVectorizer(ngram_range=(1,3))
tfidf_ngram_features = tfidfv.fit_transform(vocab_list)
# tfidf_ngram_features = tfidfv.fit_transform(data['overview'])
tfidf_ngram_features
```

```
<14640x7707 sparse matrix of type '<class 'numpy.float64'>'
        with 14654 stored elements in Compressed Sparse Row format>
```

```
tfidf_ngram_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(tfidf_ngram_features.todense()[0].getA1())
```

```
7707
```

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

```
[1.0]
```

Решение задачи анализа тональности текста

С использованием кросс-валидации попробуем применить к корпусу текстов различные варианты векторизации и классификации.

```
# Загрузка данных
data2 = pd.read_csv("/content/drive/MyDrive/MMO/pulitzer-circulation-data.csv")
data2.head()
```

	Newspaper	Daily Circulation, 2004	Daily Circulation, 2013	Change in Daily Circulation, 2004-2013	Pulitzer Prize Winners and Finalists, 1990-2003	Pulit Pr Winn Finalis 2004-2
0	USA Today	2,192,098	1,674,306	-24%	1	
1	Wall Street Journal	2,101,017	2,378,827	+13%	30	
2	New York Times	1,119,027	1,865,318	+67%	55	
3	Los Angeles	983,727	653,868	-34%	44	

```
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, data['name'], data['airline_senti
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('===========================')


vectorizers_list = [CountVectorizer(vocabulary = corpusVocab), TfidfVectorizer(v
classifiers_list = [LogisticRegression(C=3.0), MultinomialNB()]
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:81
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:81
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:81
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
Векторизация - CountVectorizer(vocabulary={'0504traveller': 0, '09202010':
                            '0veranalyser': 2, '0xjared': 3, '10eshaa': 4,
                            '1234567890_': 5, '140justinc': 6, '18handicap'
                            '1_7_8_0': 8, '1jensaba': 9, '1kingmeech': 10,
                            '1lovept': 11, '1malindac': 12,
                            '1misterhandsome': 13, '1stcrown': 14,
                            '201chef': 15, '215strongbul': 16, '219jondn':
                            '21stcenturymom': 18, '2533107724paul': 19,
                            '27_powers': 20, '29mc29': 21, '2avsagas': 22,
                            '2cjustice4all': 23, '2emmyz': 24, '2hats1mike'
                            '2littlebirds': 26, '2lnr': 27, '2tsierole': 28
                            '2v': 29, ...})
Модель для классификации - LogisticRegression(C=3.0)
Accuracy = 0.6157786885245902
===========================
Векторизация - CountVectorizer(vocabulary={'0504traveller': 0, '09202010':
                            '0veranalyser': 2, '0xjared': 3, '10eshaa': 4,
                            '1234567890_': 5, '140justinc': 6, '18handicap'
```

```
                                    '1_7_8_0': 8, '1jensaba': 9, '1kingmeech': 10,
                                    '1lovept': 11, '1malindac': 12,
                                    '1misterhandsome': 13, '1stcrown': 14,
                                    '201chef': 15, '215strongbul': 16, '219jondn':
                                    '21stcenturymom': 18, '2533107724paul': 19,
                                    '27_powers': 20, '29mc29': 21, '2avsagas': 22,
                                    '2cjustice4all': 23, '2emmyz': 24, '2hats1mike'
                                    '2littlebirds': 26, '2lnr': 27, '2tsierole': 28
                                    '2v': 29, ...})
    Модель для классификации - MultinomialNB()
    Accuracy = 0.6280054644808742
    ============================
    /usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:81
    STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

    Increase the number of iterations (max_iter) or scale the data as shown in:
```

Разделим выборку на обучающую и тестовую

```
X_train, X_test, y_train, y_test = train_test_split(data['name'], data['airline_
```

```
def sentiment(v, c):
    model = Pipeline(
        [("vectorizer", v),
         ("classifier", c)])
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print_accuracy_score_for_classes(y_test, y_pred)
```

```
sentiment(TfidfVectorizer(), LogisticRegression(C=5.0))
```

```
    Метка    Accuracy
    negative         0.9344760251681493
    neutral          0.20038659793814434
    positive         0.09577221742881795
    /usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:81
    STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

    Increase the number of iterations (max_iter) or scale the data as shown in:
        https://scikit-learn.org/stable/modules/preprocessing.html
    Please also refer to the documentation for alternative solver options:
        https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
      extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

```
sentiment(TfidfVectorizer(ngram_range=(1,3)), LogisticRegression(C=5.0))
```

```
    Метка    Accuracy
    negative         0.9344760251681493
    neutral          0.20038659793814434
    positive         0.09577221742881795
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:81
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

```
sentiment(TfidfVectorizer(ngram_range=(2,3)), LogisticRegression(C=5.0))
```

```
Метка    Accuracy
negative         1.0
neutral          0.0
positive         0.0
```

```
sentiment(TfidfVectorizer(ngram_range=(1,4)), LogisticRegression(C=5.0))
```

```
Метка    Accuracy
negative         0.9344760251681493
neutral          0.20038659793814434
positive         0.09577221742881795
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:81
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

```
sentiment(TfidfVectorizer(ngram_range=(2,4)), LogisticRegression(C=5.0))
```

```
Метка    Accuracy
negative         1.0
neutral          0.0
positive         0.0
```

ψ

```
**Вывод:**
Как видно из результатов, лучшую точност
*MultinomialNB*.
Точность составила 62,8%.
```

Вывод:

Как видно из результатов, лучшую точность показал *CountVectorizer* и *MultinomialNB*.

Точность составила 62,8%.