

МГТУ им. Н. Э. Баумана, кафедра ИУ5
курс “Методы машинного обучения”

Рубежный контроль №2

«Методы обработки текстов»

ВЫПОЛНИЛ:

Акушко А.С.

Группа: ИУ5-21М

Вариант: 1

ПРОВЕРИЛ:

Гапанюк Ю.Е.

Москва 2022

РК2

Студент: Акушко Антон Сергеевич

Группа: ИУ5-21М

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)

```
In [ ]: 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_err
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.datasets import fetch_20newsgroups
import matplotlib.pyplot as plt

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

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

In []:

```
def accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray) -> Dict[int, float]:
    """
    Вычисление метрики ассурасу для каждого класса
    y_true - истинные значения классов
    y_pred - предсказанные значения классов
    Возвращает словарь: ключ - метка класса,
    значение - Ассурасу 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_dataflt = df[df['t']==c]
        # расчет ассурасу для заданной метки класса
        temp_acc = accuracy_score(
            temp_dataflt['t'].values,
            temp_dataflt['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 [ ]: vocabVect = CountVectorizer()
vocabVect.fit(data)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(corpusVocab)))
```

Количество сформированных признаков - 33448

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

```
nrmendel=22213
unix=31462
amherst=5287
edu=12444
nathaniel=21624
mendell=20477
subject=29220
re=25369
bike=6898
```

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

```
Out[ ]: <2380x33448 sparse matrix of type '<class 'numpy.int64'>'
        with 335176 stored elements in Compressed Sparse Row format>
```

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

```
Out[ ]: 33448
```

```
In [ ]: 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)
```

```
Out[ ]: ['01810',
'01830',
'018801285',
'019',
'02',
'020',
'0200',
'020347',
'0205',
'020533',
'020555',
'020646',
'02086551',
'02115',
'02118',
'02138',
'02139',
'02142',
'02154',
'0216']
```

```
In [ ]: 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=
                print('Векторизация - {}'.format(v))
                print('Модель для классификации - {}'.format(c))
                print('Accuracy = {}'.format(score))
                print('=====')
```

```
In [ ]: vectorizers_list = [CountVectorizer(vocabulary = corpusVocab), TfidfVectorizer(vocabulary = corpusVocab)]
        classifiers_list = [LogisticRegression(), MultinomialNB()]
        VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to
converge (status=1):
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-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to
converge (status=1):
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-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to
converge (status=1):
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-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

```
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000000004': 3,
      '0000000005': 4, '0000000667': 5, '0000001200': 6,
      '0001': 7, '00014': 8, '0002': 9, '0003': 10,
      '0005111312': 11, '0005111312na1em': 12,
      '00072': 13, '000851': 14, '000rpm': 15,
      '000th': 16, '001': 17, '0010': 18, '001004': 19,
      '0011': 20, '001211': 21, '0013': 22, '001642': 23,
      '001813': 24, '002': 25, '002222': 26,
      '002251w': 27, '0023': 28, '002937': 29, ...})
```

Модель для классификации - LogisticRegression()

Accuracy = 0.9382336841146768

=====

```
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000000004': 3,
      '0000000005': 4, '0000000667': 5, '0000001200': 6,
      '0001': 7, '00014': 8, '0002': 9, '0003': 10,
      '0005111312': 11, '0005111312na1em': 12,
      '00072': 13, '000851': 14, '000rpm': 15,
      '000th': 16, '001': 17, '0010': 18, '001004': 19,
      '0011': 20, '001211': 21, '0013': 22, '001642': 23,
      '001813': 24, '002': 25, '002222': 26,
      '002251w': 27, '0023': 28, '002937': 29, ...})
```

Модель для классификации - MultinomialNB()

Accuracy = 0.9747904364702481

=====

```
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000000004': 3,
      '0000000005': 4, '0000000667': 5, '0000001200': 6,
      '0001': 7, '00014': 8, '0002': 9, '0003': 10,
      '0005111312': 11, '0005111312na1em': 12,
      '00072': 13, '000851': 14, '000rpm': 15,
      '000th': 16, '001': 17, '0010': 18, '001004': 19,
      '0011': 20, '001211': 21, '0013': 22, '001642': 23,
      '001813': 24, '002': 25, '002222': 26,
      '002251w': 27, '0023': 28, '002937': 29, ...})
```

Модель для классификации - LogisticRegression()

Accuracy = 0.9584091700786584

=====

```
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000000004': 3,
      '0000000005': 4, '0000000667': 5, '0000001200': 6,
      '0001': 7, '00014': 8, '0002': 9, '0003': 10,
      '0005111312': 11, '0005111312na1em': 12,
      '00072': 13, '000851': 14, '000rpm': 15,
```

```
'000th': 16, '001': 17, '0010': 18, '001004': 19,
'0011': 20, '001211': 21, '0013': 22, '001642': 23,
'001813': 24, '002': 25, '002222': 26,
'002251w': 27, '0023': 28, '002937': 29, ...})
Модель для классификации - MultinomialNB()
Accuracy = 0.9722710153812272
=====
```

Как видно из результатов, лучшую точность показал **CountVectorizer** и **MultinomialNB** (Точность составила 97,4%)

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js