

RK2

Костян Алина Алексеевна И5-23М

Номер по списку группы - 10

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 Вашей группы:

ИУ5-21М - LinearSVC, Multinomial Naive Bayes (MNB)

```
In [12]: 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
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
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 [13]: categories = ["rec.motorcycles", "rec.sport.baseball", "sci.electronics", "sci.space"]
newsgroups = fetch_20newsgroups(subset='train', categories=categories)
data = newsgroups['data']
```

```
In [3]: 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]))

```

```

In [4]: vocabVect = CountVectorizer()
vocabVect.fit(data)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(corpusVocab)))

```

Количество сформированных признаков - 33448

```

In [5]: 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 [6]: test_features = vocabVect.transform(data)
test_features

```

```

Out[6]: <2380x33448 sparse matrix of type '<class 'numpy.int64'>'
        with 335176 stored elements in Compressed Sparse Row format>

```

```
In [7]: # Размер нулевой строки
len(test_features.todense()[0].getA1())
```

```
Out[7]: 33448
```

```
In [8]: vocabVect.get_feature_names()[100:120]
```

```
Out[8]: ['01810',
'01830',
'018801285',
'019',
'02',
'020',
'0200',
'020347',
'0205',
'020533',
'020555',
'020646',
'02086551',
'02115',
'02118',
'02138',
'02139',
'02142',
'02154',
'0216']
```

```
In [9]: 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
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('=====')
```

```
In [14]: vectorizers_list = [CountVectorizer(vocabulary = corpusVocab), TfidfVectorizer]
classifiers_list = [LinearSVC(), MultinomialNB()]
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '000000
0004': 3,
                                '00000000005': 4, '00000000667': 5, '00000001200': 6,
                                '0001': 7, '00014': 8, '0002': 9, '0003': 10,
                                '0005111312': 11, '0005111312nalem': 12,
                                '00072': 13, '000851': 14, '000rpm': 15,
                                '000th': 16, '001': 17, '0010': 18, '001004': 19,
                                '0011': 20, '001211': 21, '0013': 22, '001642': 2
3,
                                '001813': 24, '002': 25, '002222': 26,
                                '002251w': 27, '0023': 28, '002937': 29, ...})
Модель для классификации - LinearSVC()
Accuracy = 0.9453742497059174
=====
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '000000
0004': 3,
                                '00000000005': 4, '00000000667': 5, '00000001200': 6,
                                '0001': 7, '00014': 8, '0002': 9, '0003': 10,
                                '0005111312': 11, '0005111312nalem': 12,
                                '00072': 13, '000851': 14, '000rpm': 15,
```

```

'000th': 16, '001': 17, '0010': 18, '001004': 19,
'0011': 20, '001211': 21, '0013': 22, '001642': 2
3,
'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, '000000
0004': 3,
'0000000005': 4, '0000000667': 5, '0000001200': 6,
'0001': 7, '00014': 8, '0002': 9, '0003': 10,
'0005111312': 11, '0005111312nalem': 12,
'00072': 13, '000851': 14, '000rpm': 15,
'000th': 16, '001': 17, '0010': 18, '001004': 19,
'0011': 20, '001211': 21, '0013': 22, '001642': 2
3,
'001813': 24, '002': 25, '002222': 26,
'002251w': 27, '0023': 28, '002937': 29, ...})
Модель для классификации - LinearSVC()
Accuracy = 0.9731101165424162
=====
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '000000
0004': 3,
'0000000005': 4, '0000000667': 5, '0000001200': 6,
'0001': 7, '00014': 8, '0002': 9, '0003': 10,
'0005111312': 11, '0005111312nalem': 12,
'00072': 13, '000851': 14, '000rpm': 15,
'000th': 16, '001': 17, '0010': 18, '001004': 19,
'0011': 20, '001211': 21, '0013': 22, '001642': 2
3,
'001813': 24, '002': 25, '002222': 26,
'002251w': 27, '0023': 28, '002937': 29, ...})
Модель для классификации - MultinomialNB()
Accuracy = 0.9722710153812272
=====
Лучшую точность показал LinearSVC и MultinomialNB (Точность составила 97,5%)

```

In []: