

РК2

Нурлыева Дана Джалилевна И5-21М

Номер по списку группы - 6

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 - LogisticRegression, Multinomial Naive Bayes (MNB)

```
In [10]: 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.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
plt.style.use("ticks")
```

```
In [12]: categories = ["rec.motorcycles", "rec.sport.baseball", "sci.electronics"]
newsgroups = fetch_20newsgroups(subset='train', categories=categories)
data = newsgroups['data']
```

```
In [13]: 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_dataflt = df[df['t']==c]
        # расчет accuracy 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 [14]: vocabVect = CountVectorizer()
vocabVect.fit(data)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(corpusVocab)))

Количество сформированных признаков - 33448
```

```
In [15]: 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 [16]: test_features = vocabVect.transform(data)
test_features
```

```
Out[16]: <2380x33448 sparse matrix of type '<class 'numpy.int64'>'
         with 335176 stored elements in Compressed Sparse Row format>
```

```
In [17]: # Размер нулевой строки
len(test_features.todense()[0].getA1())
```

```
Out[17]: 33448
```

```
In [18]: vocabVect.get_feature_names()[100:120]
```

```
Out[18]: ['01810',
          '01830',
          '018801285',
          '019',
          '02',
          '020',
          '0200',
          '020347',
          '0205',
          '020533',
          '020555',
          '020646',
          '02086551',
          '02115',
          '02118',
          '02138',
          '02139',
          '02142',
          '02154',
          '0216']
```

```
In [19]: 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'], cv=5)
                  print('Векторизация - {}'.format(v))
                  print('Модель для классификации - {}'.format(c))
                  print('Accuracy = {}'.format(score))
                  print('=====')
```

```
In [20]: vectorizers_list = [CountVectorizer(vocabulary = corpusVocab), TfidfVectorizer()]
          classifiers_list = [LogisticRegression(), MultinomialNB()]
          VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/logistic
.py:433: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
```

```
FutureWarning)
```

```
/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/logistic
.py:460: FutureWarning: Default multi_class will be changed to 'auto'
in 0.22. Specify the multi_class option to silence this warning.
"this warning.", FutureWarning)
```

```
Векторизация - CountVectorizer(analyzer='word', binary=False, decode
_error='strict',
dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
lowercase=True, max_df=1.0, max_features=None, min_df=1,
ngram_range=(1, 1), preprocessor=None, stop_words=None,
```

```

strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
tokenizer=None,
vocabulary={'from': 14399, 'nrmendel': 22213, 'unix': 31462,
'amherst': 5287, 'edu': 12444, 'nathaniel': 21624, 'mendell': 20477,
'subject': 29220, 're': 25369, 'bike': 6898, 'advice': 4864, 'organi
zation': 22734, 'college': 9353, 'newsreader': 21886, 'tin': 30443,
'version': 31994, 'pl7': 23928, 'l...105', 'intents': 17321, 'sypmtom
s': 29717, 'focussed': 14098, 'emotionally': 12715, 'advising': 4869
})
Модель для классификации - LogisticRegression(C=1.0, class_weight=No
ne, dual=False, fit_intercept=True,
intercept_scaling=1, max_iter=100, multi_class='warn',
n_jobs=None, penalty='l2', random_state=None, solver='warn
',
tol=0.0001, verbose=0, warm_start=False)

```

Accuracy = 0.9474775613232068

=====

Векторизация - CountVectorizer(analyzer='word', binary=False, decode_error='strict',

dtype=<class 'numpy.int64'>, encoding='utf-8', input='conten

t',
lowercase=True, max_df=1.0, max_features=None, min_df=1,
ngram_range=(1, 1), preprocessor=None, stop_words=None,
strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
tokenizer=None,

vocabulary={'from': 14399, 'nrmendel': 22213, 'unix': 31462, 'amherst': 5287, 'edu': 12444, 'nathaniel': 21624, 'mendell': 20477, 'subject': 29220, 're': 25369, 'bike': 6898, 'advice': 4864, 'organization': 22734, 'college': 9353, 'newsreader': 21886, 'tin': 30443, 'version': 31994, 'pl7': 23928, 'l...105', 'intents': 17321, 'sypmtoms': 29717, 'focussed': 14098, 'emotionally': 12715, 'advising': 4869})

Модель для классификации - MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)

Accuracy = 0.9747904364702481

=====

/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:433: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.

FutureWarning)

/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:460: FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to silence this warning.

"this warning.", FutureWarning)

Векторизация - TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',

dtype=<class 'numpy.float64'>, encoding='utf-8', input='cont

ent',
lowercase=True, max_df=1.0, max_features=None, min_df=1,
ngram_range=(1, 1), norm='l2', preprocessor=None, smooth_idf

=True,
stop_words=None, strip_accents=None, sublinear_tf=False,

```

token_pattern= (?u)\b\w+\b , tokenizer=None, use_idf=True,
vocabulary={'from': 14399, 'nrmendel': 22213, 'unix': 31462,
'amherst': 5287, 'edu': 12444, 'nathaniel': 21624, 'mendell': 20477,
'subject': 29220, 're': 25369, 'bike': 6898, 'advice': 4864, 'organization': 22734, 'college': 9353, 'newsreader': 21886, 'tin': 30443,
'version': 31994, 'pl7': 23928, 'l...105, 'intents': 17321, 'syptoms': 29717, 'focussed': 14098, 'emotionally': 12715, 'advising': 4869
})
Модель для классификации - LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,

intercept_scaling=1, max_iter=100, multi_class='warn',
n_jobs=None, penalty='l2', random_state=None, solver='warn'
,
tol=0.0001, verbose=0, warm_start=False)
Accuracy = 0.9567272619467359
=====
Векторизация - TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
dtype=<class 'numpy.float64'>, encoding='utf-8', input='content',
lowercase=True, max_df=1.0, max_features=None, min_df=1,
ngram_range=(1, 1), norm='l2', preprocessor=None, smooth_idf=True,
stop_words=None, strip_accents=None, sublinear_tf=False,
token_pattern='(?u)\b\w+\b', tokenizer=None, use_idf=True,
vocabulary={'from': 14399, 'nrmendel': 22213, 'unix': 31462,
'amherst': 5287, 'edu': 12444, 'nathaniel': 21624, 'mendell': 20477,
'subject': 29220, 're': 25369, 'bike': 6898, 'advice': 4864, 'organization': 22734, 'college': 9353, 'newsreader': 21886, 'tin': 30443,
'version': 31994, 'pl7': 23928, 'l...105, 'intents': 17321, 'syptoms': 29717, 'focussed': 14098, 'emotionally': 12715, 'advising': 4869
})
Модель для классификации - MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
Accuracy = 0.9722710153812272
=====

```

Лучшую точность показал CountVectorizer и MultinomialNB (Точность составила 97,4%)

In []: