

Рубежный контроль №2

Вариант №4

Студент: Кучеренко М.А.

Группа: ИУ5-21М

Классификатор №1

LogisticRegression

Классификатор №2

Multinomial Naive Bayes (MNB)

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`.

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

Набор данных - [20 newsgroups text dataset](#)

Классы: 20

Выборка: 18846

```
1 import numpy as np
2 import pandas as pd
3 from typing import Dict, Tuple
4 from sklearn.feature_extraction.text import CountVectorizer,
  TfidfVectorizer
5 from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
6 from sklearn.metrics import accuracy_score, balanced_accuracy_score
7 from sklearn.metrics import precision_score, recall_score, f1_score,
  classification_report
8 from sklearn.metrics import confusion_matrix
9 from sklearn.model_selection import cross_val_score
10 from sklearn.pipeline import Pipeline
11 from sklearn.metrics import mean_absolute_error, mean_squared_error,
  mean_squared_log_error, median_absolute_error, r2_score
12 from sklearn.metrics import roc_curve, roc_auc_score
13 from sklearn.naive_bayes import MultinomialNB
14 from sklearn.linear_model import LogisticRegression
```

```
15 from collections import Counter
16 from sklearn.datasets import fetch_20newsgroups
17 import matplotlib
18 import matplotlib.pyplot as plt
19 import seaborn as sns
20 %matplotlib inline
```

```
1 categories = ["sci.crypt", "sci.electronics", "talk.religion.misc",
2 "rec.sport.baseball"]
3 newsgroups = fetch_20newsgroups(subset='train', categories=categories)
4 data = newsgroups['data']
```

```
1 Downloading 20news dataset. This may take a few minutes.
2 Downloading dataset from https://ndownloader.figshare.com/files/5975967 (14
  MB)
```

```
1 def accuracy_score_for_classes(y_true, y_pred):
2     df = pd.DataFrame(data={'t': y_true, 'p': y_pred})
3     res = dict()
4     for c in np.unique(y_true):
5         temp_data_flt = df[df['t'] == c]
6         temp_acc = accuracy_score(
7             temp_data_flt['t'].values,
8             temp_data_flt['p'].values
9         )
10        res[c] = temp_acc
11    return res
12
13 def print_accuracy_score_for_classes(y_true, y_pred):
14     accs = accuracy_score_for_classes(y_true, y_pred)
15     if len(accs) > 0:
16         print('Метка \t Accuracy')
17         for i in accs:
18             print('{} \t {}'.format(i, accs[i]))
```

```
1 # С помощью CountVectorizer преобразуем коллекцию текстовых данных в
  матрицу счётчиков токенов
2 vocabVect = CountVectorizer()
3 vocabVect.fit(data)
4 corpusVocab = vocabVect.vocabulary_
5 print(f'Feature count - {len(corpusVocab)}')
```

```
1 Feature count - 33282
```

```

1 first_el = 0
2 last_el = 9
3
4 for word in list(corpusVocab)[first_el:last_el]:
5     print(f'{word:10}: {corpusVocab[word]}')
```

```

1 from      : 14539
2 philly    : 23632
3 ravel     : 25268
4 udel      : 30929
5 edu       : 12456
6 robert    : 26356
7 hite      : 16186
8 subject   : 29047
9 re        : 25308
```

```

1 test_features = vocabVect.transform(data)
2 test_features.todense().shape
```

```

1 (2160, 33282)
```

```

1 test_features.todense()
```

```

1 matrix([[0, 3, 0, ..., 0, 0, 0],
2         [0, 0, 0, ..., 0, 0, 0],
3         [0, 0, 0, ..., 0, 0, 0],
4         ...,
5         [0, 0, 0, ..., 0, 0, 0],
6         [0, 0, 0, ..., 0, 0, 0],
7         [0, 0, 0, ..., 0, 0, 0]])
```

```

1 # Cross-validation classification
2 def VectorizeAndClassify(vectorizers_list, classifiers_list):
3     max_acc = 0
4
5     for v in vectorizers_list:
6         for c in classifiers_list:
7             pipeline1 = Pipeline([('vectorizer', v), ('classifier', c)])
```

```

8         score = cross_val_score(
9             pipeline1,
10            newsgroups['data'],
11            newsgroups['target'],
12            scoring='accuracy',
13            cv=3
14        ).mean()
15        if score > max_acc:
16            max_acc = score
17            max_v = v
18            max_c = c
19        print(f'Векторизация:\t {v}\nКлассификатор:\t {c}\nAccuracy:\t
{score}')
20        print('='*80)
21
22        print(f'\nЛучший результат: {max_acc}, {type(max_v).__name__},
{type(max_c).__name__}')

```

```

1  vectorizers_list = (
2      CountVectorizer(vocabulary = corpusVocab),
3      TfidfVectorizer(vocabulary = corpusVocab),
4  )
5  classifiers_list = (
6      LogisticRegression(),
7      MultinomialNB(),
8  )
9
10 VectorizeAndClassify(vectorizers_list, classifiers_list)

```

```

1  Векторизация: CountVectorizer(analyzer='word', binary=False,
decode_error='strict',
2      dtype=<class 'numpy.int64'>, encoding='utf-8',
input='content',
3      lowercase=True, max_df=1.0, max_features=None, min_df=1,
4      ngram_range=(1, 1), preprocessor=None, stop_words=None,
5      strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
6      tokenizer=None,
7      vocabulary={'00': 0, '000': 1, '00000000': 2,
'00000000b...
8          '00000001': 4, '00000001b': 5, '00000010': 6,
9          '00000010b': 7, '00000011': 8, '00000011b': 9,
10         '00000100': 10, '00000100b': 11, '00000101':
12,
11         '00000101b': 13, '00000110': 14, '00000110b':
15,
12         '00000111': 16, '00000111b': 17, '00001000':
18,

```

```

13         '00001000b': 19, '00001001': 20, '00001001b':
14         21,
15         '00001010': 22, '00001010b': 23, '00001011':
16         24,
17         '00001011b': 25, '00001100': 26, '00001100b':
18         27,
19         '00001101': 28, '00001101b': 29, ...})
20
21 Классификатор: LogisticRegression(C=1.0, class_weight=None, dual=False,
22 fit_intercept=True,
23         intercept_scaling=1, l1_ratio=None, max_iter=100,
24         multi_class='warn', n_jobs=None, penalty='l2',
25         random_state=None, solver='warn', tol=0.0001,
26         verbose=0,
27         warm_start=False)
28
29 Accuracy: 0.9625025202237687
30
31 =====
32 =====
33
34 Векторизация: CountVectorizer(analyzer='word', binary=False,
35 decode_error='strict',
36         dtype=<class 'numpy.int64'>, encoding='utf-8',
37         input='content',
38         lowercase=True, max_df=1.0, max_features=None, min_df=1,
39         ngram_range=(1, 1), preprocessor=None, stop_words=None,
40         strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
41         tokenizer=None,
42         vocabulary={'00': 0, '000': 1, '00000000': 2,
43         '00000000b...
44         '00000001': 4, '00000001b': 5, '00000010': 6,
45         '00000010b': 7, '00000011': 8, '00000011b': 9,
46         '00000100': 10, '00000100b': 11, '00000101':
47         12,
48         '00000101b': 13, '00000110': 14, '00000110b':
49         15,
50         '00000111': 16, '00000111b': 17, '00001000':
51         18,
52         '00001000b': 19, '00001001': 20, '00001001b':
53         21,
54         '00001010': 22, '00001010b': 23, '00001011':
55         24,
56         '00001011b': 25, '00001100': 26, '00001100b':
57         27,
58         '00001101': 28, '00001101b': 29, ...})
59
60 Классификатор: MultinomialNB(alpha=1.0, class_prior=None,
61 fit_prior=True)
62
63 Accuracy: 0.980561313132677
64
65 =====
66 =====

```

```

1 Векторизация: TfidfVectorizer(analyzer='word', binary=False,
decode_error='strict',
2         dtype=<class 'numpy.float64'>, encoding='utf-8',
3         input='content', lowercase=True, max_df=1.0,
max_features=None,
4         min_df=1, ngram_range=(1, 1), norm='l2',
preprocessor=None,
5         smooth_idf=True, stop_words=None, strip_accents=None,
6         sublinear_tf=False, token_pattern='(?u)\\b\\w\\w+\\b',
7         tokenizer=None, use...
8         '00000001': 4, '00000001b': 5, '00000010': 6,
9         '00000010b': 7, '00000011': 8, '00000011b': 9,
10        '00000100': 10, '00000100b': 11, '00000101':
12,
11        '00000101b': 13, '00000110': 14, '00000110b':
15,
12        '00000111': 16, '00000111b': 17, '00001000':
18,
13        '00001000b': 19, '00001001': 20, '00001001b':
21,
14        '00001010': 22, '00001010b': 23, '00001011':
24,
15        '00001011b': 25, '00001100': 26, '00001100b':
27,
16        '00001101': 28, '00001101b': 29, ...})
17 Классификатор: LogisticRegression(C=1.0, class_weight=None, dual=False,
fit_intercept=True,
18         intercept_scaling=1, l1_ratio=None, max_iter=100,
19         multi_class='warn', n_jobs=None, penalty='l2',
20         random_state=None, solver='warn', tol=0.0001,
verbose=0,
21         warm_start=False)
22 Accuracy: 0.9499986461165014
23 =====
24 Векторизация: TfidfVectorizer(analyzer='word', binary=False,
decode_error='strict',
25         dtype=<class 'numpy.float64'>, encoding='utf-8',
26         input='content', lowercase=True, max_df=1.0,
max_features=None,
27         min_df=1, ngram_range=(1, 1), norm='l2',
preprocessor=None,
28         smooth_idf=True, stop_words=None, strip_accents=None,
29         sublinear_tf=False, token_pattern='(?u)\\b\\w\\w+\\b',
30         tokenizer=None, use...
31         '00000001': 4, '00000001b': 5, '00000010': 6,
32         '00000010b': 7, '00000011': 8, '00000011b': 9,
33        '00000100': 10, '00000100b': 11, '00000101':
12,

```

```

34         '00000101b': 13, '00000110': 14, '00000110b':
15,
35         '00000111': 16, '00000111b': 17, '00001000':
18,
36         '00001000b': 19, '00001001': 20, '00001001b':
21,
37         '00001010': 22, '00001010b': 23, '00001011':
24,
38         '00001011b': 25, '00001100': 26, '00001100b':
27,
39         '00001101': 28, '00001101b': 29, ...})
40 Классификатор: MultinomialNB(alpha=1.0, class_prior=None,
fit_prior=True)
41 Accuracy: 0.8898076033311392
42 =====
43
44 Лучший результат: 0.980561313132677, CountVectorizer, MultinomialNB

```