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

```python
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_error, r2_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from collections import Counter
from sklearn.datasets import fetch_20newsgroups
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
categories = ["sci.crypt", "sci.electronics", "talk.religion.misc"]
newsgroups = fetch_20newsgroups(subset='train', categories=categories)
data = newsgroups['data']
```

```
def accuracy_score_for_classes(y_true, y_pred):
    df = pd.DataFrame(data={'t': y_true, 'p': y_pred})
    res = dict()
    for c in np.unique(y_true):
        temp_data_flt = df[df['t'] == c]
        temp_acc = accuracy_score(
            temp_data_flt['t'].values,
            temp_data_flt['p'].values
        )
        res[c] = temp_acc
    return res

def print_accuracy_score_for_classes(y_true, y_pred):
    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]))
```

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

```
Feature count - 28701
```

```
first_el = 0
last_el = 9

for word in list(corpusVocab)[first_el:last_el]:
    print(f'{word:10}: {corpusVocab[word]}')
```

```
from        : 12431
mhald       : 17640
lynx        : 16837
dac         : 8923
northeastern: 18926
edu         : 10573
mark        : 17150
hald        : 13352
subject     : 24936
```

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

```
(1563, 28701)
```

```
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],
        [1, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]])
```

```
# Cross-validation classification
def VectorizeAndClassify(vectorizers_list, classifiers_list):
    max_acc = 0

    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=3
            ).mean()
            if score > max_acc:
                max_acc = score
                max_v = v
                max_c = c
            print(f'Векторизация:\t {v}\nКлассификатор:\t {c}\nAccuracy:\t {score}')
            print('='*80)

    print(f'\nЛучший результат: {max_acc}, {type(max_v).__name__}, {type(max_c).__name__}')
```

```
vectorizers_list = (
    CountVectorizer(vocabulary = corpusVocab),
    TfidfVectorizer(vocabulary = corpusVocab),
)
classifiers_list = (
    LogisticRegression(),
    MultinomialNB(),
)

VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
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={'00': 0, '000': 1, '00000000': 2, '00000000b...
                            '00000001': 4, '00000001b': 5, '00000010': 6,
                            '00000010b': 7, '00000011': 8, '00000011b': 9,
                            '00000100': 10, '00000100b': 11, '00000101': 12,
                            '00000101b': 13, '00000110': 14, '00000110b': 15,
                            '00000111': 16, '00000111b': 17, '00001000': 18,
                            '00001000b': 19, '00001001': 20, '00001001b': 21,
                            '00001010': 22, '00001010b': 23, '00001011': 24,
                            '00001011b': 25, '00001100': 26, '00001100b': 27,
                            '00001101': 28, '00001101b': 29, ...})
Классификатор:   LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, 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:  0.9609748411307085
================================================================================
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={'00': 0, '000': 1, '00000000': 2, '00000000b...
                            '00000001': 4, '00000001b': 5, '00000010': 6,
                            '00000010b': 7, '00000011': 8, '00000011b': 9,
                            '00000100': 10, '00000100b': 11, '00000101': 12,
                            '00000101b': 13, '00000110': 14, '00000110b': 15,
                            '00000111': 16, '00000111b': 17, '00001000': 18,
                            '00001000b': 19, '00001001': 20, '00001001b': 21,
                            '00001010': 22, '00001010b': 23, '00001011': 24,
                            '00001011b': 25, '00001100': 26, '00001100b': 27,
                            '00001101': 28, '00001101b': 29, ...})
Классификатор:   MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
Accuracy:  0.984007541400486
================================================================================
```

```
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\\w+\\b',
                tokenizer=None, use...
```

```
                            '00000001': 4, '00000001b': 5, '00000010': 6,
                            '00000010b': 7, '00000011': 8, '00000011b': 9,
                            '00000100': 10, '00000100b': 11, '00000101': 12,
                            '00000101b': 13, '00000110': 14, '00000110b': 15,
                            '00000111': 16, '00000111b': 17, '00001000': 18,
                            '00001000b': 19, '00001001': 20, '00001001b': 21,
                            '00001010': 22, '00001010b': 23, '00001011': 24,
                            '00001011b': 25, '00001100': 26, '00001100b': 27,
                            '00001101': 28, '00001101b': 29, ...})
Классификатор:   LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, 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:  0.9366625786143953
================================================================================
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\\w+\\b',
                tokenizer=None, use...
                            '00000001': 4, '00000001b': 5, '00000010': 6,
                            '00000010b': 7, '00000011': 8, '00000011b': 9,
                            '00000100': 10, '00000100b': 11, '00000101': 12,
                            '00000101b': 13, '00000110': 14, '00000110b': 15,
                            '00000111': 16, '00000111b': 17, '00001000': 18,
                            '00001000b': 19, '00001001': 20, '00001001b': 21,
                            '00001010': 22, '00001010b': 23, '00001011': 24,
                            '00001011b': 25, '00001100': 26, '00001100b': 27,
                            '00001101': 28, '00001101b': 29, ...})
Классификатор:   MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
Accuracy:  0.8675669621264962
================================================================================

Лучший результат: 0.984007541400486, CountVectorizer, MultinomialNB
```