Рубежный контроль №2 по курсу "Методы машинного обучения"

Чжан Чжибо ИУ5И-21М

Вариант №1: CountVectorizer, TfidfVectorizer, LogisticRegression, Multinomial Naive Bayes (MNB)

```
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
sns.set(style="ticks")


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 значения
            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]))
```

Загрузка данных:

```
%cd /content/drive/MyDrive/dataset/spam/
```

```
/content/drive/MyDrive/dataset/spam
```

```
dataset = pd.read_csv("enron_spam_data.csv")
dataset.head()
```

	Unnamed: 0	Subject	Message	Spam/Ham	Date
0	0	christmas tree farm pictures	NaN	ham	1999-12-10
1	1	vastar resources , inc .	gary , production from the high island larger ...	ham	1999-12-13
2	2	calpine daily gas nomination	- calpine daily gas nomination 1 . doc	ham	1999-12-14

```
dataset=dataset.drop(['Unnamed: 0','Subject','Date'],axis=1)
dataset.head()
```

	Message	Spam/Ham

```
dataset['Spam/Ham']=dataset['Spam/Ham'].replace(['ham','spam'],[0,1])
dataset.head()
```

	Message	Spam/Ham
0	NaN	0
1	gary , production from the high island larger ...	0
2	- calpine daily gas nomination 1 . doc	0
3	fyi - see note below - already done .\nstella\...	0
4	fyi .\n- - - - - - - - - - - - - - - - - - - -...	0

```
dataset=dataset.dropna()
dataset.head()
```

	Message	Spam/Ham
1	gary , production from the high island larger ...	0
2	- calpine daily gas nomination 1 . doc	0
3	fyi - see note below - already done .\nstella\...	0
4	fyi .\n- - - - - - - - - - - - - - - - - - - -...	0
5	jackie ,\nsince the inlet to 3 river plant is ...	0

Сформируем общий словарь для обучения моделей из обучающей и тестовой выборки

```
vocab_list = dataset['Message'].tolist()
vocab_list[1:10]
```

```
['- calpine daily gas nomination 1 . doc',
 'fyi - see note below - already done .\nstella\n- - - - - - - - - - - - - - - - - - - - - - - - -
 'fyi .\n- - - - - - - - - - - - - - - - - - - - - - - - forwarded by lauri a allen / hou / ect c
 'jackie ,\nsince the inlet to 3 river plant is shut in on 10 / 19 / 99 ( the last day of\nfl
 'george ,\ni need the following done :\njan 13\nzero out 012 - 27049 - 02 - 001 receipt pack
 "fyi\n- - - - - - - - - - - - - - - - - - - - - - - - forwarded by gary l payne / hou / ect on 1
 'there are two fields of gas that i am having difficulty with in the unify\nsystem .\n1 . ca
 'thanks so much for the memo . i would like to reiterate my support on two key\nissues :\n1
 'the purpose of the email is to recap the kickoff meeting held on yesterday\nwith members fr
```

```
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, dataset['Message'], dataset['Spam/Ha
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy - {}'.format(score))
```

```
                print('Accuracy = {}'.format(score))
                print('============================')


vocabVect = CountVectorizer()
vocabVect.fit(vocab_list)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(co
```

```
Количество сформированных признаков - 60049
```

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

```
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None,
                vocabulary={'00': 0, '000': 1, '0000': 2, '000000': 3,
                            '00...
                            '000000000005412': 12, '000000000005413': 13,
                            '000000000005820': 14, '000000000006238': 15,
                            '000000000006452': 16, '000000000007399': 17,
                            '000000000007494': 18, '000000000007498': 19,
                            '000000000007568': 20, '000000000007588': 21,
                            '000000000007589': 22, '000000000007590': 23,
                            '000000000007591': 24, '000000000007592': 25,
                            '000000000007593': 26, '000000000007666': 27,
                            '000000000007874': 28, '000000000007876': 29, ...})
Модель для классификации - MultinomialNB(alpha=1.0, class_prior=None
Accuracy = 0.7107923073877237
============================
Векторизация - TfidfVectorizer(analyzer='word', binary=False, decode_error='st
                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...
                            '000000000005412': 12, '000000000005413': 13,
                            '000000000005820': 14, '000000000006238': 15,
                            '000000000006452': 16, '000000000007399': 17,
                            '000000000007494': 18, '000000000007498': 19,
                            '000000000007568': 20, '000000000007588': 21,
                            '000000000007589': 22, '000000000007590': 23,
                            '000000000007591': 24, '000000000007592': 25,
                            '000000000007593': 26, '000000000007666': 27,
                            '000000000007874': 28, '000000000007876': 29, ...})
Модель для классификации - LogisticRegression(C=1.0, class_weight=No
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
Accuracy = 0.7313786840625275
============================
Векторизация - TfidfVectorizer(analyzer='word', binary=False, decode_error='st
                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...
```

```
                                '000000000005412': 12, '000000000005413': 13,
                                '000000000005820': 14, '000000000006238': 15,
                                '000000000006452': 16, '000000000007399': 17,
                                '000000000007494': 18, '000000000007498': 19,
                                '000000000007568': 20, '000000000007588': 21,
                                '000000000007589': 22, '000000000007590': 23,
                                '000000000007591': 24, '000000000007592': 25,
                                '000000000007593': 26, '000000000007666': 27,
                                '000000000007874': 28, '000000000007876': 29, ...})
Модель для классификации - MultinomialNB(alpha=1.0, class_prior=None
Accuracy = 0.7113864156174105
============================
```

Лучшую точность показал TfidfVectorizer и LogisticRegression (73,1%)