

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

ОТЧЕТ

Рубежный контроль № 2
по дисциплине «Методы машинного обучения»

Тема: «Методы обработки данных»

ИСПОЛНИТЕЛЬ:

группа ИУ5-24М

Кравцов А.Н.

ФИО

подпись

"26" апреля 2024 г.

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

ФИО

подпись

"__" _____ 2024 г.

Москва – 2024

Задание

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: GradientBoostingClassifier, LogisticRegression

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

```
##Выполнение

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from typing import Dict, Tuple
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score

df=pd.read_csv('https://storage.googleapis.com/kagglestdsdata/datasets/2050/3494/SPAM%20text%20message%2020170820%20-%20Data.csv?X-Goog-Algorithm=GD

df.head()
```

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
#Переименуем целевой столбец на spam, а значения ham и spam на 0 и 1, соответственно
df.rename(columns={'Category': 'spam', 'Message': 'message'}, inplace=True)
df['spam'] = df['spam'].replace({'ham': 0, 'spam': 1})
```

```
def accuracy_score_for_classes(y_true: np.ndarray, y_pred: np.ndarray) -> Dict[int, float]:
    """
    Вычисление метрики ассурасу для каждого класса
    y_true - истинные значения классов
    y_pred - предсказанные значения классов
    Возвращает словарь: ключ - метка класса,
    значение - Ассурасу 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 ассурасу 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 ассурасу 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]))
```

```
# Сформируем общий словарь для обучения моделей из обучающей и тестовой выборки
vocab_list = df['message'].tolist()
vocab_list[1:10]
```

```
['Ok lar... Joking wif u oni...',
 "Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's",
 'U dun say so early hor... U c already then say...',
 "Nah I don't think he goes to usf, he lives around here though",
 "FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, £1.50 to rcv",
 'Even my brother is not like to speak with me. They treat me like aids patent.',
 "As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune",
 'WINNER!! As a valued network customer you have been selected to receivea £900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.',
 'Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030']
```

```
vocabVect = CountVectorizer()
vocabVect.fit(vocab_list)
corpusVocab = vocabVect.vocabulary_
```

```
print('Количество сформированных признаков - {}'.format(len(corpusVocab)))
```

Количество сформированных признаков - 8709

```
for i in list(corpusVocab)[1:10]:
    print('{}={}'.format(i, corpusVocab[i]))
```

until=8080
jurong=4370
point=5954
crazy=2334
available=1313
only=5567
in=4110
bugis=1763
great=3651

```
test_features = vocabVect.transform(vocab_list)
```

```
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, df['message'], df['spam'], scoring='accuracy', cv=3).mean()
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('=====')
```

```
vectorizers_list = [CountVectorizer(vocabulary = corpusVocab), TfidfVectorizer(vocabulary = corpusVocab)]
classifiers_list = [GradientBoostingClassifier(), LogisticRegression(C=3.0)]
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '000pes': 2, '008704050406': 3,
'0089': 4, '0121': 5, '01223585236': 6,
'01223585334': 7, '0125698789': 8, '02': 9,
'0207': 10, '02072069400': 11, '02073162414': 12,
'02085076972': 13, '021': 14, '03': 15, '04': 16,
'0430': 17, '05': 18, '050703': 19, '0578': 20,
'06': 21, '07': 22, '07008009200': 23,
'07046744435': 24, '07090201529': 25,
'07090298926': 26, '07099833605': 27,
'07123456789': 28, '0721072': 29, ...})
```

Модель для классификации - GradientBoostingClassifier()

Accuracy = 0.9702085360931272

=====

```
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '000pes': 2, '008704050406': 3,
'0089': 4, '0121': 5, '01223585236': 6,
'01223585334': 7, '0125698789': 8, '02': 9,
'0207': 10, '02072069400': 11, '02073162414': 12,
'02085076972': 13, '021': 14, '03': 15, '04': 16,
'0430': 17, '05': 18, '050703': 19, '0578': 20,
'06': 21, '07': 22, '07008009200': 23,
'07046744435': 24, '07090201529': 25,
'07090298926': 26, '07099833605': 27,
'07123456789': 28, '0721072': 29, ...})
```

Модель для классификации - LogisticRegression(C=3.0)

Accuracy = 0.982591785578825

...

```
'07123456789': 28, '0721072': 29, ...})
```

Модель для классификации - LogisticRegression(C=3.0)

Accuracy = 0.9712851555775054

=====

Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings](#)...