МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

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

ОТЧЕТ

Рубежный контроль №__2__
по дисциплине «Методы машинного обучения»

ИСПОЛНИТЕЛЬ: ____Кожуро Б.Е.____
ФИО
группа _____ИУ5-21М_____
_____
подпись
"__"___ __ ___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 для Вашей группы:

ИУ5-21М - KNeighborsClassifier и LogisticRegression

Выполнение

Рубежный контроль 2.

датасет https://www.kaggle.com/datasets/team-ai/spam-text-message-classification

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:

Группа ИУ5-21М

`KNeighborsClassifier`
`LogisticRegression`

```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from scipy import stats
from IPython.display import Image
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor,
KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
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.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR,
NuSVR, LinearSVR
import seaborn as sns
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 значениях
        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]))

data = pd.read_csv(r'C:\Users\ksarb\Documents\MMO_2024\Datasets\
train.csv', sep=",", encoding = "utf-8")

data.head()
```

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

```
data.shape
```

```
(5574, 2)
```

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

```
['Ok lar... Joking wif u oni...\n',
 "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\n",
 'U dun say so early hor... U c already then say...\n',
 "Nah I don't think he goes to usf, he lives around here though\n",
 "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\n",
 'Even my brother is not like to speak with me. They treat me like
aids patent.\n',
 "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\n",
 '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.\n',
 '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\n']
```

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

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

```
Количество сформированных признаков - 8713
```

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

```
until=8084
jurong=4374
```

```
point=5958
crazy=2338
available=1316
only=5571
in=4114
bugis=1767
great=3655

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, data['sms'],
data['label'], 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 = [KNeighborsClassifier(), 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 - KNeighborsClassifier()
Accuracy = 0.9126300681736633
============================
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 = 0.982956584140653
============================
Векторизация - TfidfVectorizer(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 - KNeighborsClassifier()
Accuracy = 0.9250089702188733
============================
Векторизация - TfidfVectorizer(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 = 0.9714747039827771
============================
```

Лучшая модель - Logistic regression с векторизацией CountVectorizer