ИУ5-21М Гришин С. В.

Рубежный контроль №2

Необходимо решить задачу классификации текстов на основе любого датасета (кроме примера, который рассматривался в лекции). Классификация может быть бинарной или многоклассовой. Целевой признак из 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 группы: KNeighborsClassifieri LogisticRegression.

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

Ход выполнения работы

Импорт библиотек и загрузка набора данных

```
[11] 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.metrics import accuracy_score
     import pandas as pd
     import time
```

```
[2]  from google.colab import drive
     drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[3]  data = pd.read_csv('/content/drive/MyDrive/MMO/cefr_leveled_texts.csv')
```

```
[4]  data.head()
```

	text	label
0	Hi!\nI've been meaning to write for ages and f...	B2
1	It was not so much how hard people found the ...	B2
2	Keith recently came back from a trip to Chicag...	B2
3	The Griffith Observatory is a planetarium, and...	B2
4	-LRB- The Hollywood Reporter -RRB- It's offici...	B2

Разделение выборки на обучающую и тестовую

```
[5] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1494 entries, 0 to 1493
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   text    1494 non-null   object
 1   label   1494 non-null   object
dtypes: object(2)
memory usage: 23.5+ KB
```

```
[6] prop_mask = data.isna()
    props = prop_mask.sum()
    props
```

```
text     0
label    0
dtype: int64
```

```
[7] X, Y = data['text'], data['label']
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)

    time_arr = []
```

```
[8] count_vect = CountVectorizer()
    X_train_counts = count_vect.fit_transform(X_train)
    X_test_counts = count_vect.transform(X_test)
```

```
[9] tfidf_vect = TfidfVectorizer()
    X_train_tfidf = tfidf_vect.fit_transform(X_train)
    X_test_tfidf = tfidf_vect.transform(X_test)
```

Обучение для CountVectorizer

```
[13] gbc = KNeighborsClassifier()
     start_time = time.time()
     gbc.fit(X_train_counts, y_train)
     train_time = time.time() - start_time
     time_arr.append(train_time)
     pred_gbc_counts = gbc.predict(X_test_counts)
     print("Точность CountVectorizer и KNeighborsClassifier: ", accuracy_score(y_test, pred_gbc_counts))
```

```
Точность CountVectorizer и KNeighborsClassifier:  0.47491638795986624
```

```
lr = LogisticRegression(max_iter = 1000)
start_time = time.time()
lr.fit(X_train_counts, y_train)
train_time = time.time() - start_time
time_arr.append(train_time)
pred_lr_counts = lr.predict(X_test_counts)
print("Точность CountVectorizer и Logistic Regression: ", accuracy_score(y_test, pred_lr_counts))
```

```
Точность CountVectorizer и Logistic Regression:  0.6053511705685619
```

```
[15] gbc = KNeighborsClassifier()
     start_time = time.time()
     gbc.fit(X_train_tfidf, y_train)
     train_time = time.time() - start_time
     time_arr.append(train_time)
     pred_gbc_tfidf = gbc.predict(X_test_tfidf)
     print("Точность TfidfVectorizer и KNeighborsClassifier: ", accuracy_score(y_test, pred_gbc_tfidf))
```

```
Точность TfidfVectorizer и KNeighborsClassifier:  0.4882943143812709
```

```
[16] lr = LogisticRegression(max_iter = 1000)
     start_time = time.time()
     lr.fit(X_train_tfidf, y_train)
     train_time = time.time() - start_time
     time_arr.append(train_time)
     pred_lr_tfidf = lr.predict(X_test_tfidf)
     print("Точность TfidfVectorizer и Logistic Regression: ", accuracy_score(y_test, pred_lr_tfidf))
```

```
Точность TfidfVectorizer и Logistic Regression:  0.5986622073578596
```

```python
from tabulate import tabulate

data = [
    ["(CountVectorizer и LogisticRegression)", accuracy_score(y_test, pred_lr_counts), time_arr[0]],
    ["(CountVectorizer и KNeighborsClassifier)", accuracy_score(y_test, pred_gbc_counts), time_arr[1]],
    ["(TfidfVectorizer и LogisticRegression)", accuracy_score(y_test, pred_lr_tfidf), time_arr[2]],
    ["(TfidfVectorizer и KNeighborsClassifier)", accuracy_score(y_test, pred_gbc_tfidf), time_arr[3]]
]

sorted_data = sorted(data, key=lambda x: x[1], reverse=True)

print(tabulate(sorted_data,['Комбинация','Точность','Время обучения'], tablefmt="grid"))
```

```
+-----------------------------------------+------------+------------------+
| Комбинация                              |   Точность |   Время обучения |
+=========================================+============+==================+
| (CountVectorizer и LogisticRegression)  |   0.605351 |       0.00766253 |
+-----------------------------------------+------------+------------------+
| (TfidfVectorizer и LogisticRegression)  |   0.598662 |      60.9049     |
+-----------------------------------------+------------+------------------+
| (TfidfVectorizer и KNeighborsClassifier) |   0.488294 |       0.014632   |
+-----------------------------------------+------------+------------------+
| (CountVectorizer и KNeighborsClassifier) |   0.474916 |       0.00962234 |
+-----------------------------------------+------------+------------------+
```