Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления

Рубежный контроль 2

Вариант №2

Выполнил:

студент группы ИУ5-25М

Ветошкин Артём

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.

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import pandas as pd
import time
```

```
# Загрузка данных
df = pd.read_csv('train_40k.csv')
```

```
df.head(10)
```

	productId	Title	userId	Helpfulness	Score	Time	Text	Ca
0	B000E46LYG	Golden Valley Natural Buffalo Jerky	A3MQDNGHDJU4MK	0/0	3.0	-1	The description and photo on this product need...	groc gourr fc
1	B000GRA6N8	Westing Game	unknown	0/0	5.0	860630400	This was a great book!!!! It is well thought t...	t gar
2	B000GRA6N8	Westing Game	unknown	0/0	5.0	883008000	I am a first year teacher, teaching 5th grade....	t gar
3	B000GRA6N8	Westing Game	unknown	0/0	5.0	897696000	I got the book at my bookfair at school lookin...	t gar

Далее: Посмотреть рекомендованные графики

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40000 entries, 0 to 39999
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   productId    40000 non-null  object
 1   Title        39984 non-null  object
 2   userId       40000 non-null  object
 3   Helpfulness  40000 non-null  object
 4   Score        40000 non-null  float64
 5   Time         40000 non-null  int64
 6   Text         40000 non-null  object
```

```
 7   Cat1         40000 non-null  object
 8   Cat2         40000 non-null  object
 9   Cat3         40000 non-null  object
dtypes: float64(1), int64(1), object(8)
memory usage: 3.1+ MB
```

```
# проверим пропуски в данных и устраним их
na_mask = df.isna()
na_counts = na_mask.sum()
na_counts
```

```
productId       0
Title          16
userId          0
Helpfulness     0
Score           0
Time            0
Text            0
Cat1            0
Cat2            0
Cat3            0
dtype: int64
```

```
df.dropna(inplace=True)
na_mask = df.isna()
na_counts = na_mask.sum()
na_counts
```

```
productId      0
Title          0
userId         0
Helpfulness    0
Score          0
Time           0
Text           0
Cat1           0
Cat2           0
Cat3           0
dtype: int64
```

```
# Разделим набор данных на обучающую и тестувую выборки
X, Y = df['Text'], df['Cat2']
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

time_arr = []
```

```
# векторизация признаков с помощью CountVectorizer
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
X_test_counts = count_vect.transform(X_test)
```

```
# векторизация признаков с помощью TfidfVectorizer
tfidf_vect = TfidfVectorizer()
X_train_tfidf = tfidf_vect.fit_transform(X_train)
X_test_tfidf = tfidf_vect.transform(X_test)
```

```
# Произведем обучения вдух классификаторов (по варианту) для CountVectorizer

# SVC
gbc = SVC()
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 + SVC):", accuracy_score(y_test, pred_gbc_counts))

# Logistic Regression
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 + LogisticRegression):", accuracy_score(y_test, pred_lr_counts))
```

```
Точность (CountVectorizer + SVC): 0.4666750031261723
Точность (CountVectorizer + LogisticRegression): 0.5997248968363136
```

```
# Произведем обучения вдух классификаторов (по варианту) для TfidfVectorizer

# SVC
gbc = SVC()
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 + LinearSVC):", accuracy_score(y_test, pred_gbc_tfidf))

# Logistic Regression
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 + LogisticRegression):", accuracy_score(y_test, pred_lr_tfidf))
```

```
Точность (TfidfVectorizer + LinearSVC): 0.6087282731024134
Точность (TfidfVectorizer + LogisticRegression): 0.6222333375015631
```

```
from tabulate import tabulate

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

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

# Вывод отсортированных данных в виде таблицы
print(tabulate(sorted_data, ['Модели','Точность валидации', 'Время обучения'], tablefmt="grid"))
```

```
+---------------------------------------+----------------------+------------------+
| Связка                                |   Точность валидации |   Время обучения |
+=======================================+======================+==================+
| (TfidfVectorizer + LogisticRegression) |             0.622233 |         1299.28  |
+---------------------------------------+----------------------+------------------+
| (TfidfVectorizer + LinearSVC)          |             0.608728 |          159.232 |
+---------------------------------------+----------------------+------------------+
| (CountVectorizer + LogisticRegression) |             0.599725 |         1005.25  |
+---------------------------------------+----------------------+------------------+
| (CountVectorizer + LinearSVC)          |             0.466675 |          510.494 |
+---------------------------------------+----------------------+------------------+
```

Лучше всего показл себя TFIDF векторайзер в паре с логистической регрессиой