РК2 ММО Савельев Алексей ИУ5-24М

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:

ИУ5-24М, ИУ5И-24М GradientBoostingClassifier LogisticRegression

Решение

Загрузка набора данных (Набор твитов (с обозначенным настроением для каждого))

```
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
import pandas as pd
import time

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

[46] ✓ 0.3s

Данные датасета:

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

[59] ✓ 0.0s

	Unnamed: 0	text	sentiment
0	0	text	0
1	1	rising cases of covid does not alarm me rising...	1
2	2	please vote for chicagoindiaresolution marking...	0
3	3	wishing all of you eidaladha hazrat ibrahim as...	1
4	4	daily coronavirus cases in india top for first...	1
5	5	sitting here india style watching the raindrop...	0
6	6	who out there believes that if china and india...	1
7	7	very happy with the new education policy imple...	1
8	8	india development partnership does not come wi...	0
9	9	hon prime minister india how the new education...	1

Информация по столбцам набора данных

```
df.info()
```

[48] ✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 134348 entries, 0 to 134347
Data columns (total 3 columns):
 #   Column      Non-Null Count   Dtype 
---  ------      --------------   ----- 
 0   Unnamed: 0  134348 non-null  int64 
 1   text        134330 non-null  object
 2   sentiment   134348 non-null  int64 
dtypes: int64(2), object(1)
memory usage: 3.1+ MB
```

Количество пропусков по столбцам (и удаление их):

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

[49] ✓ 0.0s

```
Unnamed: 0     0
text          18
sentiment      0
dtype: int64
```

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

[50] ✓ 0.0s

```
Unnamed: 0    0
text          0
sentiment     0
dtype: int64
```

Разделение на тренировочную и тестовую выборки

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

time_arr = []
```

[51] ✓ 0.0s

Формирование двух вариантов векторизации признаков:

```
# 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)
```

[52] ✓ 2.6s

```
# 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)
```

[53] ✓ 2.7s

Обучения для CountVectorizer:

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

# Gradient Boosting Classifier
gbc = GradientBoostingClassifier()
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 + GradientBoosting):", 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))
```

[54] ✓ 48.3s Python

```
Точность (CountVectorizer + GradientBoosting): 0.8087917814337825
Точность (CountVectorizer + LogisticRegression): 0.9355691208218566
```

Обучения для TfidfVectorizer:

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

# Gradient Boosting Classifier
gbc = GradientBoostingClassifier()
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 + GradientBoosting):", 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))
```

[55] ✓ 1m 55.3s Python

```
Точность (TfidfVectorizer + GradientBoosting): 0.8083823419935978
Точность (TfidfVectorizer + LogisticRegression): 0.9194148738182089
```

Вывод результатов:

```python
from tabulate import tabulate

data = [
    ["(CountVectorizer + LogisticRegression)", accuracy_score(y_test, pred_lr_counts), time_arr[0]],
    ["(CountVectorizer + GradientBoosting)", accuracy_score(y_test, pred_gbc_counts), time_arr[1]],
    ["(TfidfVectorizer + LogisticRegression)", accuracy_score(y_test, pred_lr_tfidf), time_arr[2]],
    ["(TfidfVectorizer + GradientBoosting)", 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, ['Связка','Точность валидации', 'Время обучения'], tablefmt="grid"))
```

[57] ✓ 0.0s

Сравнение и анализ результатов обучения:

Связка	Точность валидации	Время обучения
(CountVectorizer + LogisticRegression)	0.935569	37.3713
(TfidfVectorizer + LogisticRegression)	0.919415	113.357
(CountVectorizer + GradientBoosting)	0.808792	10.9069
(TfidfVectorizer + GradientBoosting)	0.808382	1.94459

Наибольшую точность показал классификатор LogisticRegression в месте с векторизацией признаков CountVectorizer, а именно 93,5% на валидационной выборке. При этом время выполнения находится в удовлетворимом интервале, в отличие от связки TfidVectorizer с LogisticRegression, где обучение длилось значительно дольше остальных.

В результате сравнения можно сказать, что, независимо от варианта векторизации, качество обучения классификатора Логической Регрессии выше, чем Градиентного Бустинга.