Сорокин А.Э, 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.

Решение

```
# 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 = []
```

```
# 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)
```

```
# 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))
```

```python
# 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))
```

Python

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

Результаты:

```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"))
```

Python

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