Московский Государственный Технический Университет
имениН.Э.Баумана
Факультет Информатика и системы управления
Кафедра ИУ-5
«Системы обработки информации и управления»

Рубежный контроль по дисциплине

«Методы машинного обучения»

Тема: Методы обработки текстов

Студент: Лу Жуньда
Группа:ИУ5И-22М

Москва 2024г

Варианты заданий

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.

Классификатор №1: RandomForestClassifier

Классификатор №2: LogisticRegression

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

Импорт библиотек и подготовка данных

```
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
# Загрузка данных
newsgroups = fetch_20newsgroups(subset='all', categories=None,
shuffle=True, random_state=42, data_home='/path/to/your/data_home')
X, y = newsgroups.data, newsgroups.target
# Разделение на обучающую и тестовую выборки
  X_train, X_test, y_train, y_test = train_test_split(X, y,
  test_size=0.3, random_state=42)
```

Векторизация признаков

CountVectorizer

```
count_vectorizer = CountVectorizer()
X_train_counts = count_vectorizer.fit_transform(X_train)
X_test_counts = count_vectorizer.transform(X_test)
```

TfidfVectorizer

```
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

Обучение и оценка классификаторов

RandomForestClassifier с CountVectorizer

```
rf_clf_count = RandomForestClassifier(random_state=42)
rf_clf_count.fit(X_train_counts, y_train)
y_pred_rf_count = rf_clf_count.predict(X_test_counts)

accuracy_rf_count = accuracy_score(y_test, y_pred_rf_count)
print("RandomForestClassifier with CountVectorizer Accuracy:",
accuracy_rf_count)
print(classification_report(y_test, y_pred_rf_count))
```

```
RandomForestClassifier with CountVectorizer Accuracy: 0.8381676689069685
              precision    recall  f1-score   support

           0       0.87      0.81      0.84       236
           1       0.67      0.80      0.73       287
           2       0.76      0.85      0.80       290
           3       0.69      0.67      0.68       285
           4       0.83      0.82      0.82       312
           5       0.89      0.81      0.85       308
           6       0.70      0.87      0.78       276
           7       0.84      0.86      0.85       304
           8       0.95      0.93      0.94       279
           9       0.90      0.90      0.90       308
          10       0.90      0.97      0.93       309
          11       0.92      0.94      0.93       290
          12       0.81      0.62      0.70       304
          13       0.90      0.84      0.87       300
          14       0.89      0.94      0.91       297
          15       0.77      0.98      0.86       292
          16       0.83      0.90      0.86       270
          17       0.98      0.96      0.97       272
          18       0.96      0.69      0.80       239
          19       0.86      0.47      0.61       196

    accuracy                           0.84      5654
   macro avg       0.85      0.83      0.83      5654
weighted avg       0.84      0.84      0.84      5654
```

RandomForestClassifier с TfidfVectorizer

```
rf_clf_tfidf = RandomForestClassifier(random_state=42)
rf_clf_tfidf.fit(X_train_tfidf, y_train)
y_pred_rf_tfidf = rf_clf_tfidf.predict(X_test_tfidf)

accuracy_rf_tfidf = accuracy_score(y_test, y_pred_rf_tfidf)
print("RandomForestClassifier with TfidfVectorizer Accuracy:",
accuracy_rf_tfidf)
print(classification_report(y_test, y_pred_rf_tfidf))
```

RandomForestClassifier with TfidfVectorizer Accuracy: 0.8349840820657941

	precision	recall	f1-score	support
0	0.86	0.82	0.84	236
1	0.65	0.78	0.71	287
2	0.75	0.86	0.80	290
3	0.68	0.67	0.67	285
4	0.85	0.79	0.82	312
5	0.86	0.82	0.84	308
6	0.71	0.87	0.78	276
7	0.85	0.87	0.86	304
8	0.94	0.94	0.94	279
9	0.92	0.93	0.92	308
10	0.91	0.96	0.93	309
11	0.92	0.93	0.92	290
12	0.81	0.63	0.71	304
13	0.88	0.84	0.86	300
14	0.89	0.93	0.91	297
15	0.75	0.95	0.84	292
16	0.83	0.91	0.87	270
17	0.96	0.94	0.95	272
18	0.96	0.69	0.80	239
19	0.87	0.40	0.55	196
accuracy			0.83	5654
macro avg	0.84	0.83	0.83	5654
weighted avg	0.84	0.83	0.83	5654

LogisticRegression c CountVectorizer

```
lr_clf_count = LogisticRegression(max_iter=300, random_state=42)
lr_clf_count.fit(X_train_counts, y_train)
y_pred_lr_count = lr_clf_count.predict(X_test_counts)


accuracy_lr_count = accuracy_score(y_test, y_pred_lr_count)
print("LogisticRegression with CountVectorizer Accuracy:",
accuracy_lr_count)
print(classification_report(y_test, y_pred_lr_count))
```

LogisticRegression with CountVectorizer Accuracy: 0.8806154934559604

	precision	recall	f1-score	support
0	0.87	0.89	0.88	236
1	0.74	0.80	0.77	287
2	0.83	0.82	0.82	290
3	0.73	0.74	0.73	285
4	0.86	0.84	0.85	312
5	0.87	0.83	0.85	308
6	0.81	0.84	0.82	276
7	0.88	0.90	0.89	304
8	0.98	0.94	0.96	279
9	0.93	0.95	0.94	308
10	0.96	0.95	0.96	309
11	0.96	0.93	0.95	290
12	0.82	0.80	0.81	304
13	0.90	0.94	0.92	300
14	0.92	0.93	0.92	297
15	0.90	0.96	0.93	292
16	0.91	0.93	0.92	270
17	0.98	0.97	0.97	272
18	0.92	0.86	0.89	239
19	0.85	0.76	0.80	196
accuracy			0.88	5654
macro avg	0.88	0.88	0.88	5654
weighted avg	0.88	0.88	0.88	5654

LogisticRegression с TfidfVectorizer

```
lr_clf_tfidf = LogisticRegression(max_iter=300, random_state=42)
lr_clf_tfidf.fit(X_train_tfidf, y_train)
y_pred_lr_tfidf = lr_clf_tfidf.predict(X_test_tfidf)

accuracy_lr_tfidf = accuracy_score(y_test, y_pred_lr_tfidf)
print("LogisticRegression with TfidfVectorizer Accuracy:",
accuracy_lr_tfidf)
print(classification_report(y_test, y_pred_lr_tfidf))
```

```
LogisticRegression with TfidfVectorizer Accuracy: 0.8938804386275203
              precision    recall  f1-score   support

           0       0.89      0.89      0.89       236
           1       0.77      0.86      0.81       287
           2       0.83      0.86      0.84       290
           3       0.75      0.76      0.75       285
           4       0.89      0.85      0.87       312
           5       0.91      0.86      0.88       308
           6       0.75      0.84      0.79       276
           7       0.94      0.92      0.93       304
           8       0.97      0.94      0.96       279
           9       0.96      0.97      0.96       308
          10       0.96      0.96      0.96       309
          11       0.99      0.95      0.97       290
          12       0.84      0.84      0.84       304
          13       0.94      0.93      0.93       300
          14       0.93      0.96      0.95       297
          15       0.88      0.97      0.92       292
          16       0.92      0.93      0.92       270
          17       0.99      0.98      0.98       272
          18       0.93      0.87      0.90       239
          19       0.89      0.64      0.74       196

    accuracy                           0.89      5654
   macro avg       0.90      0.89      0.89      5654
weighted avg       0.90      0.89      0.89      5654
```

Выводы

LogisticRegression с TfidfVectorizer показал наилучшее качество классификации, демонстрируя наивысшую точность среди всех комбинаций.

RandomForestClassifier также показал неплохие результаты, но уступил LogisticRegression в обеих векторизациях.

Таким образом, для данной задачи классификации текстов наилучший результат достигнут при использовании LogisticRegression в сочетании с TfidfVectorizer.