Московский государственный технический университет им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

Задания для РК №2.

по дисциплине «Методы машинного обучения»

Выполнила:

студентка группы ИУ5И-23М

Цзян Юхуэй

Москва — 2024 г.

Задание

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.

Мои классификатора: LinearSVC и LogisticRegression.

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

Основной раздел кода

В этом задании мы используем набор данных "Набор данных кинорецензий IMDB".

Данные о нагрузке:

```
import pandas as pd

from sklearn.datasets import load_files

# 下载并加载 IMDB 电影评论数据集

!wget -O aclImdb_v1.tar.gz https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz

!tar -xf aclImdb_v1.tar.gz

# 加载训练数据

train_data = load_files('aclImdb/train', categories=['pos', 'neg'], shuffle=True, random_state=42)

test_data = load_files('aclImdb/test', categories=['pos', 'neg'], shuffle=True, random_state=42)

# 分离数据和标签

X_train, y_train = train_data.data, train_data.target

X_test, y_test = test_data.data, test_data.target
```

Составьте портфолио:

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

from sklearn.svm import LinearSVC

from sklearn.linear_model import LogisticRegression

from sklearn.pipeline import make_pipeline

from sklearn.metrics import classification_report

# 创建不同的管道

pipelines = {

    'CountVectorizer + LinearSVC': make_pipeline(CountVectorizer(), LinearSVC(random_state=42)),

    'CountVectorizer + LogisticRegression': make_pipeline(CountVectorizer(), LogisticRegression(random_state=42)),

    'TfidfVectorizer + LinearSVC': make_pipeline(TfidfVectorizer(), LinearSVC(random_state=42)),

    'TfidfVectorizer + LogisticRegression': make_pipeline(TfidfVectorizer(), LogisticRegression(random_state=42))

}

# 训练和评估每个管道
```

```
for name, pipeline in pipelines.items():
    print('='*80)
    print(f'Training and evaluating pipeline: {name}')
    print('='*80)
    pipeline.fit(X_train, y_train)
    y_pred = pipeline.predict(X_test)
    print(classification_report(y_test, y_pred))
    print('-' * 80)
```

Результат

```
================================================================================
Training and evaluating pipeline: CountVectorizer + LinearSVC
================================================================================
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: ConvergenceWarn
  warnings.warn(
              precision    recall  f1-score   support

           0       0.84      0.86      0.85     12500
           1       0.85      0.84      0.84     12500

    accuracy                           0.85     25000
   macro avg       0.85      0.85      0.85     25000
weighted avg       0.85      0.85      0.85     25000

--------------------------------------------------------------------------------
================================================================================
Training and evaluating pipeline: CountVectorizer + LogisticRegression
================================================================================
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Con
STOP:  TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
              precision    recall  f1-score   support

           0       0.86      0.87      0.87     12500
           1       0.87      0.86      0.86     12500

    accuracy                           0.86     25000
   macro avg       0.86      0.86      0.86     25000
weighted avg       0.86      0.86      0.86     25000

--------------------------------------------------------------------------------
================================================================================
Training and evaluating pipeline: TfidfVectorizer + LinearSVC
================================================================================
              precision    recall  f1-score   support

           0       0.87      0.89      0.88     12500
           1       0.89      0.87      0.88     12500

    accuracy                           0.88     25000
   macro avg       0.88      0.88      0.88     25000
weighted avg       0.88      0.88      0.88     25000

 ================================================================================
 Training and evaluating pipeline: TfidfVectorizer + LogisticRegression
 ================================================================================
               precision    recall  f1-score   support

            0       0.88      0.88      0.88     12500
            1       0.88      0.88      0.88     12500

     accuracy                           0.88     25000
    macro avg       0.88      0.88      0.88     25000
 weighted avg       0.88      0.88      0.88     25000

 --------------------------------------------------------------------------------
```

Рис 1. Результаты различных комбинаций.

Вывод

На основании приведенных выше результатов мы можем сделать следующие выводы:

1. TfidfVectorizer обычно лучше передает семантическую информацию текста, чем CountVectorizer, и, следовательно, лучше справляется с этой задачей.

2. LinearSVC обычно лучше справляется с высокоразмерными данными, чем LogisticRegression, и поэтому лучше справляется с этой задачей.

Таким образом, комбинация TfidfVectorizer + LinearSVC лучше всего справляется с задачей классификации текста.