Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Рубежный контроль №2
по дисциплине
«Методы машинного обучения»
на тему

«Методы обработки текстов»

Выполнил:
студент группы ИУ5-23М
Чжэн Сяохуэй

Москва — 2024 г.

Варианты заданий

Группа	Классификатор №1	Классификатор №2
ИУ5-23М, ИУ5И-23М	LinearSVC	LogisticRegression

I. Цель эксперимента

Цель данного эксперимента – сравнить эффективность различных методов извлечения признаков и классификаторов при решении задачи классификации текстов, чтобы определить наилучшую комбинацию. Конкретная задача включает использование CountVectorizer и TfidfVectorizer для извлечения признаков и классификаторов LinearSVC и LogisticRegression для классификации.

II. Датасет

Для эксперимента был выбран SMS Spam Collection Dataset с Kaggle. Датасет содержит 5572 SMS сообщения, каждое из которых помечено как "ham" (нормальное) или "spam" (спам).

III. Шаги эксперимента

1. Загрузка и импорт датасета

Сначала загружаем датасет с помощью Kaggle API и проводим предобработку данных, включая чтение CSV файла, извлечение текста и меток, кодирование меток и т.д.

```
import pandas as pd

# Импорт датасета
data = pd.read_csv('spam.csv', encoding='latin-1')
data = data[['v1', 'v2']]
data.columns = ['label', 'text']
```

```
# Кодирование меток
data['label'] = data['label'].map({'ham': 0, 'spam': 1})
```

2. Разделение датасета

Разделяем датасет на тренировочную и тестовую выборки в соотношении 80:20.

```
from sklearn.model_selection import train_test_split

X = data['text']
y = data['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

3. Извлечение признаков

Используем CountVectorizer и TfidfVectorizer для извлечения признаков из текста.

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

count_vectorizer = CountVectorizer()
tfidf_vectorizer = TfidfVectorizer()

X_train_count = count_vectorizer.fit_transform(X_train)
X_test_count = count_vectorizer.transform(X_test)

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

4. Обучение и оценка моделей

Обучаем и оцениваем модели LinearSVC и LogisticRegression на данных, полученных с помощью двух методов извлечения признаков.

```
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# LinearSVC
svc_count = LinearSVC()
svc_count.fit(X_train_count, y_train)
y_pred_svc_count = svc_count.predict(X_test_count)

svc_tfidf = LinearSVC()
svc_tfidf.fit(X_train_tfidf, y_train)
y_pred_svc_tfidf = svc_tfidf.predict(X_test_tfidf)

# LogisticRegression
lr_count = LogisticRegression()
lr_count.fit(X_train_count, y_train)
y_pred_lr_count = lr_count.predict(X_test_count)

lr_tfidf = LogisticRegression()
lr_tfidf.fit(X_train_tfidf, y_train)
y_pred_lr_tfidf = lr_tfidf.predict(X_test_tfidf)
```

5. Оценка результатов

Используем отчет по классификации для оценки эффективности моделей, включая точность, полноту, метрику F1 и др.

```
print("LinearSVC with CountVectorizer")
print(classification_report(y_test, y_pred_svc_count))
print("LinearSVC with TfidfVectorizer")
print(classification_report(y_test, y_pred_svc_tfidf))
print("LogisticRegression with CountVectorizer")
print(classification_report(y_test, y_pred_lr_count))
print("LogisticRegression with TfidfVectorizer")
print(classification_report(y_test, y_pred_lr_tfidf))
```

IV. Результаты эксперимента и анализ

На основе отчетов по классификации можно сделать следующие выводы:

1. LinearSVC с CountVectorizer
    - Высокие показатели точности, полноты и метрики F1.
2. LinearSVC с TfidfVectorizer
    - В некоторых метриках лучше, чем CountVectorizer, но в целом разница небольшая.
3. LogisticRegression с CountVectorizer
    - Хорошие показатели, но немного уступают LinearSVC.
4. LogisticRegression с TfidfVectorizer
    - Лучше, чем CountVectorizer, особенно по метрикам точности и F1.

Анализ показал, что комбинация TfidfVectorizer и LinearSVC дает наилучшие результаты, так как достигает высоких показателей по всем ключевым метрикам.

LinearSVC with CountVectorizer

	precision	recall	f1-score	support
0	0.98	1.00	0.99	965
1	0.99	0.88	0.93	150
accuracy			0.98	1115
macro avg	0.98	0.94	0.96	1115
weighted avg	0.98	0.98	0.98	1115

LinearSVC with TfidfVectorizer

	precision	recall	f1-score	support
0	0.98	1.00	0.99	965
1	0.99	0.89	0.93	150

```
   accuracy                           0.98      1115
  macro avg       0.98      0.94      0.96      1115
weighted avg       0.98      0.98      0.98      1115

LogisticRegression with CountVectorizer
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       965
           1       0.99      0.84      0.91       150

   accuracy                           0.98      1115
  macro avg       0.98      0.92      0.95      1115
weighted avg       0.98      0.98      0.98      1115

LogisticRegression with TfidfVectorizer
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       965
           1       0.99      0.77      0.86       150

   accuracy                           0.97      1115
  macro avg       0.98      0.88      0.92      1115
weighted avg       0.97      0.97      0.97      1115
```

V. Заключение

В ходе эксперимента было сравнено несколько методов извлечения признаков (CountVectorizer и TfidfVectorizer) и классификаторов (LinearSVC и LogisticRegression) для задачи классификации текстов. Результаты показали, что комбинация TfidfVectorizer и LinearSVC является наиболее эффективной. Рекомендуется использовать эту комбинацию для подобных задач классификации текстов.

Список литературы

[1] Гапанюк Ю. Е. COURSE_MMO_SPRING_2024// GitHub. — 2024. — Режим доступа: https://github.com/ugapanyuk/courses_current/wiki/COURSE_MMO_SPRING_2024

[2] https://www.kaggle.com/datasets