Рубежный контроль

Тема: Методы обработки текстов.

Выполнил: Хуан Цзэсян

Группа: ИУ5И-24М

Классификатор1:GradientBoostingClassifier

Классификатор2:LogisticRegression

```
import nltk
from nltk.corpus import movie_reviews
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

# Загрузить набор данных отзывов о фильмах
nltk.download('movie_reviews')

# Загрузка данных
documents = [(list(movie_reviews.words(fileid)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]

# Преобразование формата данных
reviews = [" ".join(doc) for doc, category in documents]
labels = [category for doc, category in documents]

# Разделить данные на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(reviews,
labels, test_size=0.2, random_state=42)

# Преобразование меток чувств к бинарной форме: positive -> 1,
negative -> 0
y_train = y_train.map({'positive': 1, 'negative': 0})
```

```
y_test = y_test.map({'positive': 1, 'negative': 0})

# Метод 1: Использование CountVectorizer для векторизации
признаков и GradientBoostingClassifier для классификации
count_vectorizer = CountVectorizer()
X_train_count = count_vectorizer.fit_transform(X_train)
X_test_count = count_vectorizer.transform(X_test)

gb_clf_count = GradientBoostingClassifier()
gb_clf_count.fit(X_train_count, y_train)
y_pred_gb_count = gb_clf_count.predict(X_test_count)
accuracy_gb_count = accuracy_score(y_test, y_pred_gb_count)
print("GradientBoostingClassifier + CountVectorizer Точность:",
accuracy_gb_count)

# Метод 2: Использование TfidfVectorizer для векторизации
признаков и GradientBoostingClassifier для классификации
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

gb_clf_tfidf = GradientBoostingClassifier()
gb_clf_tfidf.fit(X_train_tfidf, y_train)
y_pred_gb_tfidf = gb_clf_tfidf.predict(X_test_tfidf)
accuracy_gb_tfidf = accuracy_score(y_test, y_pred_gb_tfidf)
print("GradientBoostingClassifier + TfidfVectorizer Точность:",
accuracy_gb_tfidf)

# Метод 3: Использование CountVectorizer для векторизации
признаков и LogisticRegression для классификации
lr_clf_count = LogisticRegression(max_iter=1000)
lr_clf_count.fit(X_train_count, y_train)
y_pred_lr_count = lr_clf_count.predict(X_test_count)
accuracy_lr_count = accuracy_score(y_test, y_pred_lr_count)
print("LogisticRegression + CountVectorizer Точность:",
accuracy_lr_count)

# Метод 4: Использование TfidfVectorizer для векторизации
признаков и LogisticRegression для классификации
lr_clf_tfidf = LogisticRegression(max_iter=1000)
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 + TfidfVectorizer Точность:",
accuracy_lr_tfidf)
```

```
GradientBoostingClassifier + CountVectorizer Accuracy: 0.8225
GradientBoostingClassifier + TfidfVectorizer Accuracy: 0.8225
LogisticRegression + CountVectorizer Accuracy: 0.82
LogisticRegression + TfidfVectorizer Accuracy: 0.8075
```

Выводы:

Точность после классификации набора данных обзора фильмов с использованием различных методов векторизации признаков (CountVectorizer и TfidfVectorizer) и разных классификаторов (GradientBoostingClassifier и LogisticRегрессия). Конкретные результаты заключаются в следующем:

GradientBoostingClassifier + CountVectorizer Точность: 0,8225

Используйте CountVectorizer для преобразования текстовых данных в матрицу частот слов и используйте GradientBoostingClassifier для классификации. Полученная точность составляет 82,25%.

GradientBoostingClassifier + TfidfVectorizer Точность: 0,8225

Используя TfidfVectorizer для преобразования текстовых данных в функции TF-IDF и используя GradientBoostingClassifier для классификации, полученная точность также составляет 82,25%.

Логистическая регрессия + CountVectorizer Точность: 0,82

Используйте CountVectorizer для преобразования текстовых данных в матрицу частот слов и используйте LogisticRegrade для классификации. Полученная точность составляет 82%.

Логистическая регрессия + TfidfVectorizer Точность: 0,8075

Используйте TfidfVectorizer для преобразования текстовых данных в функции TF-IDF и используйте LogisticRegrade для классификации. Полученная точность составляет 80,75%.

GradientBoostingClassifier очень хорошо работает с обоими методами векторизации объектов (CountVectorizer и TfidfVectorizer) с одинаковой точностью 82,25%.

Точность LogisticRегрессии при использовании CountVectorizer (82%) немного выше, чем при использовании TfidfVectorizer (80,75%).

В совокупности комбинация GradientBoostingClassifier и двух методов векторизации объектов работает лучше, а LogisticRегрессия работает лучше при использовании CountVectorizer.

Таким образом, GradientBoostingClassifier лучше всего работает в сочетании с этими двумя методами векторизации объектов, особенно в этом эксперименте, он работает лучше, чем LogisticRегрессия.