МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Кафедра «Систем обработки информации и управления»

ОТЧЕТ

Рубежный контроль №-2

Выполнил:

студент группы ИУ5И-24М

Аунг Пьио Нанда

Москва - 2024

Тема: Методы обработки текстов.

Решение задачи классификации текстов.

Необходимо решить задачу классификации текстов на основе любого выбранного Вами датасета . 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 - GradientBoostingClassifier

Классификатор №2 - LogisticRegression

Решение

Загрузка и предобработка данных

[1]:
```
from sklearn.datasets import fetch_20newsgroups
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
```

[2]:
```
# Загрузка датасета "20 Newsgroups"
categories = ['alt.atheism', 'soc.religion.christian', 'comp.graphics', 'sci.med']
data = fetch_20newsgroups(subset='all', categories=categories, shuffle=True, random_state=42)
```

[3]:
```
# Разделение данных на признаки и целевую переменную
X = data.data
y = data.target
```

[4]:
```
# Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Теперь у нас есть обучающая и тестовая выборки. Далее применим два метода векторизации признаков: CountVectorizer и TfidfVectorizer, и обучим два классификатора: GradientBoostingClassifier и LogisticRegression.

CountVectorizer и GradientBoostingClassifier-

[5]:
```
# Векторизация признаков на основе CountVectorizer
count_vectorizer = CountVectorizer()
X_train_count = count_vectorizer.fit_transform(X_train)
X_test_count = count_vectorizer.transform(X_test)
```

[6]:
```
# Обучение GradientBoostingClassifier
gb_classifier_count = GradientBoostingClassifier()
gb_classifier_count.fit(X_train_count, y_train)
```

[6]:
```
GradientBoostingClassifier
GradientBoostingClassifier()
```

[7]:
```
# Оценка качества модели на тестовой выборке
y_pred_gb_count = gb_classifier_count.predict(X_test_count)
accuracy_gb_count = accuracy_score(y_test, y_pred_gb_count)
```

[8]:
```
print("Accuracy GradientBoostingClassifier с CountVectorizer:", accuracy_gb_count)
print("Отчет о классификации:")
print(classification_report(y_test, y_pred_gb_count))
```

```
Accuracy GradientBoostingClassifier с CountVectorizer: 0.9242021276595744
Отчет о классификации:
              precision    recall  f1-score   support

           0       0.98      0.87      0.92       175
           1       0.91      0.95      0.93       200
           2       0.89      0.93      0.91       200
           3       0.93      0.94      0.94       177

    accuracy                           0.92       752
   macro avg       0.93      0.92      0.92       752
weighted avg       0.93      0.92      0.92       752
```

LogisticRegression с использованием CountVectorizer

[9]:
```
# Обучение LogisticRegression с CountVectorizer
log_reg_classifier_count = LogisticRegression(max_iter=1000)
log_reg_classifier_count.fit(X_train_count, y_train)
```

[9]:
```
LogisticRegression(max_iter=1000)
```

[10]:
```
# Оценка качества модели на тестовой выборке
y_pred_lr_count = log_reg_classifier_count.predict(X_test_count)
accuracy_lr_count = accuracy_score(y_test, y_pred_lr_count)
```

[11]:
```
print("Accuracy LogisticRegression с CountVectorizer:", accuracy_lr_count)
print("Отчет о классификации:")
print(classification_report(y_test, y_pred_lr_count))
```

```
Accuracy LogisticRegression с CountVectorizer: 0.9481382978723404
Отчет о классификации:
              precision    recall  f1-score   support

           0       0.98      0.90      0.94       175
           1       0.94      0.97      0.96       200
           2       0.96      0.94      0.95       200
           3       0.92      0.97      0.95       177

    accuracy                           0.95       752
   macro avg       0.95      0.95      0.95       752
weighted avg       0.95      0.95      0.95       752
```

Векторизации признаков с использованием TfidfVectorizer и обучению моделей GradientBoostingClassifier и LogisticRegression.

[12]:
```
# Векторизация признаков на основе TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

[13]:
```
# Обучение GradientBoostingClassifier с TfidfVectorizer
gb_classifier_tfidf = GradientBoostingClassifier()
gb_classifier_tfidf.fit(X_train_tfidf, y_train)
```

[13]:
```
GradientBoostingClassifier
GradientBoostingClassifier()
```

[14]:
```
# Оценка качества модели на тестовой выборке
y_pred_gb_tfidf = gb_classifier_tfidf.predict(X_test_tfidf)
accuracy_gb_tfidf = accuracy_score(y_test, y_pred_gb_tfidf)
```

[15]:
```
print("Accuracy GradientBoostingClassifier с TfidfVectorizer:", accuracy_gb_tfidf)
print("Отчет о классификации:")
print(classification_report(y_test, y_pred_gb_tfidf))
```

```
Accuracy GradientBoostingClassifier с TfidfVectorizer: 0.9321808510638298
Отчет о классификации:
              precision    recall  f1-score   support

           0       0.98      0.88      0.93       175
           1       0.91      0.95      0.93       200
           2       0.91      0.93      0.92       200
           3       0.93      0.96      0.95       177

    accuracy                           0.93       752
   macro avg       0.94      0.93      0.93       752
weighted avg       0.93      0.93      0.93       752
```

[16]:
```
# Обучение LogisticRegression с TfidfVectorizer
log_reg_classifier_tfidf = LogisticRegression(max_iter=1000)
log_reg_classifier_tfidf.fit(X_train_tfidf, y_train)
```

[16]:
```
LogisticRegression
LogisticRegression(max_iter=1000)
```

[17]:
```
# Оценка качества модели на тестовой выборке
y_pred_lr_tfidf = log_reg_classifier_tfidf.predict(X_test_tfidf)
accuracy_lr_tfidf = accuracy_score(y_test, y_pred_lr_tfidf)
```

[18]:
```
print("Accuracy LogisticRegression с TfidfVectorizer:", accuracy_lr_tfidf)
print("Отчет о классификации:")
print(classification_report(y_test, y_pred_lr_tfidf))
```

```
Accuracy LogisticRegression с TfidfVectorizer: 0.9521276595744681
Отчет о классификации:
              precision    recall  f1-score   support

           0       0.99      0.92      0.95       175
           1       0.92      0.99      0.95       200
           2       0.97      0.95      0.96       200
           3       0.94      0.94      0.94       177

    accuracy                           0.95       752
   macro avg       0.95      0.95      0.95       752
weighted avg       0.95      0.95      0.95       752
```

Вывод

Этот код позволяет провести классификацию текстов на основе выбранного датасета "20 Newsgroups" с использованием двух методов векторизации признаков (CountVectorizer и TfidfVectorizer) и двух классификаторов (GradientBoostingClassifier и LogisticRegression). Результаты оценки качества моделей выводятся на экран.