РК2

Мартынова П.В. ИУ5-21М

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.

Классификатор №1	Классификатор №2
KNeighborsClassifier	LogisticRegression

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

data = pd.read_csv('spam.csv', encoding='latin-1')
data = data[['v1', 'v2']]
data = data.rename(columns={'v1': 'label', 'v2': 'text'})
```

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

CountVectorizer

```
vectorizer = CountVectorizer()
X_train_count = vectorizer.fit_transform(X_train)
X_test_count = vectorizer.transform(X_test)
```

KNeighborsClassifier

```
knn = KNeighborsClassifier()
knn.fit(X_train_count, y_train)
y_pred_knn_count = knn.predict(X_test_count)
print('Accuracy KNN CountVectorizer:', accuracy_score(y_test, y_pred_knn_count))
print(classification_report(y_test, y_pred_knn_count))
```

```
Accuracy KNN CountVectorizer: 0.9192825112107623
              precision    recall  f1-score   support

         ham       0.91      1.00      0.96       965
        spam       1.00      0.40      0.57       150

    accuracy                           0.92      1115
   macro avg       0.96      0.70      0.76      1115
weighted avg       0.93      0.92      0.90      1115
```

LogisticRegression

```
lr = LogisticRegression()
lr.fit(X_train_count, y_train)
y_pred_lr_count = lr.predict(X_test_count)
print('Accuracy LR CountVectorizer:', accuracy_score(y_test, y_pred_lr_count))
print(classification_report(y_test, y_pred_lr_count))
```

```
Accuracy LR CountVectorizer: 0.97847533632287
              precision    recall  f1-score   support

         ham       0.98      1.00      0.99       965
        spam       1.00      0.84      0.91       150

    accuracy                           0.98      1115
   macro avg       0.99      0.92      0.95      1115
weighted avg       0.98      0.98      0.98      1115
```

TfidfVectorizer

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

KNeighborsClassifier

```
knn = KNeighborsClassifier()
knn.fit(X_train_tfidf, y_train)
y_pred_knn_tfidf = knn.predict(X_test_tfidf)
print('Accuracy KNN TfidfVectorizer:', accuracy_score(y_test, y_pred_knn_tfidf))
print(classification_report(y_test, y_pred_knn_tfidf))
```

```
Accuracy KNN TfidfVectorizer: 0.915695067264574
              precision    recall  f1-score   support

         ham       0.91      1.00      0.95       965
        spam       1.00      0.37      0.54       150

    accuracy                           0.92      1115
   macro avg       0.96      0.69      0.75      1115
weighted avg       0.92      0.92      0.90      1115
```

LogisticRegression

```
lr = LogisticRegression()
lr.fit(X_train_tfidf, y_train)
y_pred_lr_tfidf = lr.predict(X_test_tfidf)
print('Accuracy LR TfidfVectorizer:', accuracy_score(y_test, y_pred_lr_tfidf))
print(classification_report(y_test, y_pred_lr_tfidf))
```

```
Accuracy LR TfidfVectorizer: 0.9659192825112107
              precision    recall  f1-score   support

         ham       0.96      1.00      0.98       965
        spam       0.99      0.75      0.86       150

    accuracy                           0.97      1115
   macro avg       0.98      0.88      0.92      1115
weighted avg       0.97      0.97      0.96      1115
```