Выполнил: Федюкин Д.А.

Группа: ИУ5-22М

Задание: Необходимо решить задачу классификации текстов, сформировав два варианта векторизации признаков - на основе CountVectorizer и на основе TfidfVectorizer. В качестве классификаторов необходимо использовать два классификатора:

- Random Forest Classifier
- Complement Naive Bayes

```
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import ComplementNB
```

```
df = pd.read_csv('train.csv', usecols=['Description', 'Class Index'],
nrows=10000)
df
```

```
      Class Index                                        Description
0               3  Reuters - Short-sellers, Wall Street's dwindli...
1               3  Reuters - Private investment firm Carlyle Grou...
2               3  Reuters - Soaring crude prices plus worries\ab...
3               3  Reuters - Authorities have halted oil export\f...
4               3  AFP - Tearaway world oil prices, toppling reco...
...           ...                                                ...
9995            4  Users of the music player should watch out for...
9996            4  BMC Software has released a new version of Pat...
9997            3  The chief of Beijing-backed China Aviation Oil...
9998            3  BRUSSELS The European Commission has opened an...
9999            4  Operation Digital Gridlock targets peer-to-pee...

[10000 rows x 2 columns]
```

Feature preparation

```
tfidfv = TfidfVectorizer()
tfidf_ngram_features = tfidfv.fit_transform(df['Description'])
tfidf_ngram_features
```

```
<10000x20257 sparse matrix of type '<class 'numpy.float64'>'
     with 283180 stored elements in Compressed Sparse Row format>
```

```
countvec = CountVectorizer()
countvec_ngram_features = countvec.fit_transform(df['Description'])
countvec_ngram_features
```

```
<10000x20257 sparse matrix of type '<class 'numpy.int64'>'
     with 283180 stored elements in Compressed Sparse Row format>
```

Random Forest Classificator

```
# TFIDF + RFC
X_train, X_test, y_train, y_test =
train_test_split(tfidf_ngram_features, df['Class Index'],
test_size=0.3, random_state=1)
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred, digits=4,
target_names=list(map(str, list(y_test.unique())))))
```

```
              precision    recall  f1-score   support

           1     0.8767    0.7773    0.8240       741
           4     0.8752    0.9127    0.8936       699
           3     0.8640    0.7719    0.8154       741
           2     0.7489    0.8706    0.8052       819

    accuracy                         0.8330      3000
   macro avg     0.8412    0.8331    0.8345      3000
weighted avg     0.8383    0.8330    0.8330      3000
```

```
# CountVec + RFC
X_train, X_test, y_train, y_test =
train_test_split(countvec_ngram_features, df['Class Index'],
test_size=0.3, random_state=1)
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred, digits=4,
target_names=list(map(str, list(y_test.unique())))))
```

```
              precision    recall  f1-score   support

           1     0.8972    0.7773    0.8330       741
           4     0.8517    0.9285    0.8884       699
           3     0.8676    0.7868    0.8252       741
           2     0.7662    0.8645    0.8124       819

    accuracy                         0.8387      3000
   macro avg     0.8457    0.8393    0.8397      3000
weighted avg     0.8435    0.8387    0.8384      3000
```

Complement Naive Bayes

```
# TFIDF + CNB
X_train, X_test, y_train, y_test =
train_test_split(tfidf_ngram_features, df['Class Index'],
test_size=0.3, random_state=1)
model = ComplementNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred, digits=4,
target_names=list(map(str, list(y_test.unique())))))
```

```
              precision    recall  f1-score   support

           1     0.8897    0.8273    0.8573       741
           4     0.8779    0.9671    0.9204       699
           3     0.8386    0.8623    0.8503       741
           2     0.8870    0.8437    0.8648       819

    accuracy                         0.8730      3000
   macro avg     0.8733    0.8751    0.8732      3000
weighted avg     0.8736    0.8730    0.8723      3000
```

```
# CountVec + CNB
X_train, X_test, y_train, y_test =
train_test_split(countvec_ngram_features, df['Class Index'],
test_size=0.3, random_state=1)
model = ComplementNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred, digits=4,
target_names=list(map(str, list(y_test.unique())))))
```

```
              precision    recall  f1-score   support

           1     0.8903    0.8327    0.8605       741
           4     0.8740    0.9728    0.9208       699
           3     0.8593    0.8408    0.8499       741
           2     0.8756    0.8596    0.8675       819

    accuracy                         0.8747      3000
   macro avg     0.8748    0.8765    0.8747      3000
weighted avg     0.8749    0.8747    0.8739      3000
```

Выводы:

1. CountVectorizer показал лучший результат в обоих моделях

2. Complement Naive Bayes показал лучший результат по сравнению с Random Forest