Выполнил: Кудрявцев С.Д. ИУ5-22М

Задание: Решить задачу классификации текстов, сформировав два варианта векторизации признаков - CountVectorizer и TfidfVectorizer. В качестве классификаторов необходимо использовать:

- 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('SPAM.csv')
df
```

```
     Category                                            Message
0         ham  Go until jurong point, crazy.. Available only ...
1         ham                      Ok lar... Joking wif u oni...
2        spam  Free entry in 2 a wkly comp to win FA Cup fina...
3         ham  U dun say so early hor... U c already then say...
4         ham  Nah I don't think he goes to usf, he lives aro...
...       ...                                                ...
5567     spam  This is the 2nd time we have tried 2 contact u...
5568      ham              Will ü b going to esplanade fr home?
5569      ham  Pity, * was in mood for that. So...any other s...
5570      ham  The guy did some bitching but I acted like i'd...
5571      ham                        Rofl. Its true to its name

[5572 rows x 2 columns]
```

Предобработка признаков

###TFIDF

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

```
<5572x8709 sparse matrix of type '<class 'numpy.float64'>'
     with 74098 stored elements in Compressed Sparse Row format>
```

###CountVectoriser

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

```
<5572x8709 sparse matrix of type '<class 'numpy.int64'>'
     with 74098 stored elements in Compressed Sparse Row format>
```

Random Forest Classifier

```
# TFIDF + RFC
X_train, X_test, y_train, y_test =
train_test_split(tfidf_ngram_features, df['Category'], 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

         ham     0.9743    0.9986    0.9863      1442
        spam     0.9897    0.8348    0.9057       230

    accuracy                         0.9761      1672
   macro avg     0.9820    0.9167    0.9460      1672
weighted avg     0.9764    0.9761    0.9752      1672
```

```
# CountVec + RFC
X_train, X_test, y_train, y_test =
train_test_split(countvec_ngram_features, df['Category'],
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

         ham     0.9749    0.9986    0.9866      1442
        spam     0.9897    0.8391    0.9082       230

    accuracy                         0.9767      1672
   macro avg     0.9823    0.9189    0.9474      1672
weighted avg     0.9770    0.9767    0.9759      1672
```

Complement Naive Bayes

```
# TFIDF + CNB
X_train, X_test, y_train, y_test =
train_test_split(tfidf_ngram_features, df['Category'], 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

         ham     0.9874    0.9771    0.9822      1442
        spam     0.8653    0.9217    0.8926       230

    accuracy                         0.9695      1672
   macro avg     0.9263    0.9494    0.9374      1672
weighted avg     0.9706    0.9695    0.9699      1672
```

```
# CountVec + CNB
X_train, X_test, y_train, y_test =
train_test_split(countvec_ngram_features, df['Category'],
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

         ham     0.9929    0.9716    0.9821      1442
        spam     0.8429    0.9565    0.8961       230

    accuracy                         0.9695      1672
   macro avg     0.9179    0.9640    0.9391      1672
weighted avg     0.9723    0.9695    0.9703      1672
```

Выводы:

1. CountVectorizer с Random Forest Classifier показал лучшие результаты, чем TFIDF, а с Complement Naive Bayes оба векторизатора показали одинаковые результаты
2. Random Forest Classifier показал лучшие, чем Complement Naive Bayes результаты