Выполнил: Гудилин Д.С.

Группа: ИУ5-24М

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:

- KNeighborsClassifier
- Complement Naive Bayes

Ввод [7]:

```
import os
import gzip
import shutil

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.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import ComplementNB
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings('ignore')
```

Ввод [8]:

```
df_perf_all = pd.read_csv('SPAM.csv', sep=",", encoding = 'ansi')
df_perf_all.head()
```

Out[8]:

	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...

Ввод [9]:

```
le=LabelEncoder()
le.fit(df_perf_all.Category)
df_perf_all['cat']=le.transform(df_perf_all.Category)
df=df_perf_all
df
```

Out[9]:

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

5572 rows × 3 columns

Feature preparation

Ввод [10]:

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

Out[10]:

```
<5572x8789 sparse matrix of type '<class 'numpy.float64'>'
        with 74369 stored elements in Compressed Sparse Row format>
```

Ввод [11]:

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

Out[11]:

```
<5572x8789 sparse matrix of type '<class 'numpy.int64'>'
        with 74369 stored elements in Compressed Sparse Row format>
```

KNeighboursClassifier

Ввод [12]:

```
# TFIDF + KNC
X_train, X_test, y_train, y_test = train_test_split(tfidf_ngram_features, df['cat'], test_s
model = KNeighborsClassifier()
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_tes
```

```
              precision    recall  f1-score   support

           0     0.8929    1.0000    0.9434      1442
           1     1.0000    0.2478    0.3972       230

    accuracy                         0.8965      1672
   macro avg     0.9464    0.6239    0.6703      1672
weighted avg     0.9076    0.8965    0.8683      1672
```

Ввод [13]:

```
# CountVec + KNC
X_train, X_test, y_train, y_test = train_test_split(countvec_ngram_features, df['cat'],
                                                    test_size=0.3, random_state=1)
model = KNeighborsClassifier()
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_tes
```

```
              precision    recall  f1-score   support

           0     0.9007    1.0000    0.9477      1442
           1     1.0000    0.3087    0.4718       230

    accuracy                         0.9049      1672
   macro avg     0.9503    0.6543    0.7098      1672
weighted avg     0.9143    0.9049    0.8823      1672
```

Complement Naive Bayes

Ввод [14]:

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

```
              precision    recall  f1-score   support

           0     0.9874    0.9785    0.9829      1442
           1     0.8724    0.9217    0.8964       230

    accuracy                         0.9707      1672
   macro avg     0.9299    0.9501    0.9397      1672
weighted avg     0.9716    0.9707    0.9710      1672
```

Ввод [15]:

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

```
              precision    recall  f1-score   support

           0     0.9922    0.9716    0.9818      1442
           1     0.8423    0.9522    0.8939       230

    accuracy                         0.9689      1672
   macro avg     0.9173    0.9619    0.9378      1672
weighted avg     0.9716    0.9689    0.9697      1672
```

Выводы:

1. TfidfVectorizer показал лучший результат в обоих моделях
2. Complement Naive Bayes показал лучший результат по сравнению с KNeighboursClassifier

Ввод [ ]: