Выполнил: Богданов Д.А. 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 и Complement Naive Bayes.

```
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

import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('twitter_validation.csv')
df.head()
```

```
     ID     Entity   Sentiment  \
0  3364   Facebook  Irrelevant
1   352     Amazon     Neutral
2  8312  Microsoft    Negative
3  4371      CS-GO    Negative
4  4433     Google     Neutral

                                             Content
0  I mentioned on Facebook that I was struggling ...
1  BBC News - Amazon boss Jeff Bezos rejects clai...
2  @Microsoft Why do I pay for WORD when it funct...
3  CSGO matchmaking is so full of closet hacking,...
4  Now the President is slapping Americans in the...
```

Feature preparation

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

```
<1000x5440 sparse matrix of type '<class 'numpy.float64'>'
     with 19225 stored elements in Compressed Sparse Row format>
```

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

```
<1000x5440 sparse matrix of type '<class 'numpy.int64'>'
     with 19225 stored elements in Compressed Sparse Row format>
```

KNeighboursClassifier

```
# TFIDF + KNC
X_train, X_test, y_train, y_test =
train_test_split(tfidf_ngram_features, df['Sentiment'], 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_test.unique())))))
```

```
              precision    recall  f1-score   support

    Negative     0.2449    0.2553    0.2500        47
     Neutral     0.3874    0.6143    0.4751        70
    Positive     0.5405    0.4255    0.4762        94
  Irrelevant     0.5152    0.3820    0.4387        89

    accuracy                         0.4300       300
   macro avg     0.4220    0.4193    0.4100       300
weighted avg     0.4510    0.4300    0.4294       300
```

```
# CountVec + KNC
X_train, X_test, y_train, y_test =
train_test_split(countvec_ngram_features, df['Sentiment'],
                                                    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_test.unique())))))
```

```
              precision    recall  f1-score   support

    Negative     0.2963    0.1702    0.2162        47
     Neutral     0.2113    0.4286    0.2830        70
    Positive     0.4595    0.1809    0.2595        94
  Irrelevant     0.3085    0.3258    0.3169        89

    accuracy                         0.2800       300
   macro avg     0.3189    0.2764    0.2689       300
```

```
weighted avg     0.3312    0.2800    0.2753       300
```

Complement Naive Bayes

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

    Negative     0.4545    0.3191    0.3750        47
     Neutral     0.4906    0.7429    0.5909        70
    Positive     0.5616    0.4362    0.4910        94
  Irrelevant     0.5000    0.4944    0.4972        89

    accuracy                         0.5067       300
   macro avg     0.5017    0.4981    0.4885       300
weighted avg     0.5100    0.5067    0.4980       300
```

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

    Negative     0.3455    0.4043    0.3725        47
     Neutral     0.4731    0.6286    0.5399        70
    Positive     0.5263    0.4255    0.4706        94
  Irrelevant     0.5526    0.4719    0.5091        89

    accuracy                         0.4833       300
   macro avg     0.4744    0.4826    0.4730       300
weighted avg     0.4934    0.4833    0.4828       300
```

Complement Naive Bayes показал лучший результат