Рубежный контроль №2

Группа: ИУ5И-22М

Выполнила: Чжан Аньци

Задание:Решение задачи классификации текстов.

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

In [1]:

```
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
```

In [2]:

```
df = pd.read_csv('news_articles.csv', usecols=['text', 'hasImage'],nrows=1000)
df
```

Out[2]:

	text	hasImage
0	print they should pay all the back all the mon...	1
1	why did attorney general loretta lynch plead t...	1
2	red state \nfox news sunday reported this mor...	1
3	email kayla mueller was a prisoner and torture...	1
4	email healthcare reform to make america great ...	1
...	...	...
995	freitag november gaagnagna zum babywort des ...	1
996	freitag november junge in brunnen gefallen ...	1
997	samstag november bremen ersetzt als erstes b...	1
998	morgen in pams steinmeier mit absoluter mehrh...	1
999	sonntag november stiftung warentest benotet ...	1

1000 rows × 2 columns

Предобработка признаков

In [3]:

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

Out[3]:

```
<1000x29611 sparse matrix of type '<class 'numpy.float64'>'
	with 197165 stored elements in Compressed Sparse Row format>
```

In [4]:

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

Out[4]:

```
<1000x29611 sparse matrix of type '<class 'numpy.int64'>'
	with 197165 stored elements in Compressed Sparse Row format>
```

Random Forest Classificator

In [5]:

```
# TFIDF + RFC
X_train, X_test, y_train, y_test = train_test_split(tfidf_ngram_features, df['hasImage'], test_size=
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     0.9444    0.4857    0.6415        70
           0     0.8636    0.9913    0.9231       230

    accuracy                         0.8733       300
   macro avg     0.9040    0.7385    0.7823       300
weighted avg     0.8825    0.8733    0.8574       300
```

In [6]:

```
# CountVec + RFC
X_train, X_test, y_train, y_test = train_test_split(countvec_ngram_features, df['hasImage'], test_si
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     0.8750    0.6000    0.7119        70
           0     0.8889    0.9739    0.9295       230

    accuracy                         0.8867       300
   macro avg     0.8819    0.7870    0.8207       300
weighted avg     0.8856    0.8867    0.8787       300
```

Complement Naive Bayes

In [7]:

```
# TFIDF + CNB
X_train, X_test, y_train, y_test = train_test_split(tfidf_ngram_features, df['hasImage'], test_size=
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     1.0000    0.0286    0.0556        70
           0     0.7718    1.0000    0.8712       230

    accuracy                         0.7733       300
   macro avg     0.8859    0.5143    0.4634       300
weighted avg     0.8251    0.7733    0.6809       300
```

In [8]:

```
# CountVec + CNB
X_train, X_test, y_train, y_test = train_test_split(countvec_ngram_features, df['hasImage'], test_si
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     0.8846    0.3286    0.4792        70
           0     0.8285    0.9870    0.9008       230

    accuracy                         0.8333       300
   macro avg     0.8565    0.6578    0.6900       300
weighted avg     0.8416    0.8333    0.8024       300
```

Выводы:

1. CountVectorizer показал лучший результат в обоих моделях
2. Random Forest Classifier показал лучшие, чем Complement Naive Bayes результаты