Рубежный контроль №2

Тема: Методы обработки текстов

Решение задачи классификации текстов.

Необходимо решить задачу классификации текстов на основе любого выбранного Вами датасета (кроме примера, который рассматривался в лекции). Классификация может быть бинарной или многоклассовой. Целевой признак из выбранного Вами датасета может иметь любой физический смысл, примером является задача анализа тональности текста.

Необходимо сформировать два варианта векторизации признаков - на основе CountVectorizer и на основе TfidfVectorizer.

В качестве классификаторов необходимо использовать два классификатора по варианту для Вашей группы:

ИУ5И-22М

Классификатор №1:RandomForestClassifier

Классификатор №2: Complement Naive Bayes - CNB

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 [3]:
```
df=pd.read_csv('youtube.csv')
df
```

Out[3]:

	link	title	description	category
0	JLZICZ0	Ep 1| Travelling through North East India | Of...	Tanya Khanijow\n671K subscribers\nSUBSCRIBE\nT...	travel
1	i9E_Blai8vk	Welcome to Bali | Travel Vlog | Priscilla Lee	Priscilla Lee\n45.6K subscribers\nSUBSCRIBE\n*...	travel
2	r284c-q8oY	My Solo Trip to ALASKA | Cruising From Vancouv...	Allison Anderson\n588K subscribers\nSUBSCRIBE\...	travel
3	Qmi-Xwq-ME	Traveling to the Happiest Country in the World!!	Yes Theory\n6.65M subscribers\nSUBSCRIBE\n*BLA...	travel
4	_IcOX55Ef70	Solo in Paro Bhutan | Tiger's Nest visit | Bhu...	Tanya Khanijow\n671K subscribers\nSUBSCRIBE\nH...	travel
...	...	...	...	...
3594	#NAME?	21st Century Challenges: Crash Course European...	CrashCourse\n12.4M subscribers\nSUBSCRIBE\nThe...	history
3595	d-2Trw8bCa0	EU DataViz webinar - Barnaby Skinner - How to ...	Publications Office of the European Union\n3.2...	history
3596	RCKWarkUL	Stone Age Scandinavia: First People In the Nor...	History Time\n619K subscribers\nSUBSCRIBE\n- W...	history
3597	MF6F3BxJIY	AP European History - Interwar Period: Paris P...	Mr. Raymond's Civics and Social Studies Academ...	history
3598	IByKodp_UK	World War 2 Allied Conferences: AP European Hi...	Paul Sargent\n25.3K subscribers\nSUBSCRIBE\nIn...	history

3599 rows × 4 columns

Предобработка признаков

TFIDF

In [4]:
```
tfidf=TfidfVectorizer()
tfidf_ngram_features=tfidf.fit_transform(df['description'])
print('看一下tfidf_ngram_features的值：\n{}'.format(tfidf_ngram_features))
```

```
看一下tfidf_ngram_features的值：
  (0, 12865)	0.026498899028004747
  (0, 17321)	0.026535936513009763
  (0, 20222)	0.056672786612321085
  (0, 13443)	0.10617413840216165
  (0, 3349)	0.12252855255981054
  (0, 8237)	0.08179721643009606
  (0, 13898)	0.0995092588291622
  (0, 20734)	0.07917522101745697
  (0, 1836)	0.12468252161298352
  (0, 3846)	0.20298293694881073
  (0, 12184)	0.14919309938125191
  (0, 7423)	0.13749638394983638
  (0, 17661)	0.07651065694772624
  (0, 8905)	0.07422251008358562
  (0, 18941)	0.07178710961660202
  (0, 13773)	0.044986203510332834
  (0, 8349)	0.1564895455085993
  (0, 19154)	0.1564895455085993
  (0, 7291)	0.13081629728384944
  (0, 4045)	0.06763707519445934
  (0, 18012)	0.12655743425241778
  (0, 8241)	0.13908983274384046
  (0, 10527)	0.10437382358177655
  (0, 2487)	0.06908816000418752
  (0, 5028)	0.20298293694881073
  :	:
  (3598, 7766)	0.07061592002524433
  (3598, 928)	0.10576974746171705
  (3598, 5388)	0.1116464597082773
  (3598, 2134)	0.07765312328577796
  (3598, 9607)	0.10402784484713663
  (3598, 7812)	0.08812866178590896
  (3598, 9311)	0.07265062864871628
  (3598, 3168)	0.06734810442320449
  (3598, 19027)	0.043524558473489114
  (3598, 13880)	0.04840345897219326
```

```
(3598, 20873)	0.09841590144437862
(3598, 4024)	0.07377923074840587
(3598, 9635)	0.03851122907342391
(3598, 13613)	0.07823235623580176
(3598, 12865)	0.023813631633524963
(3598, 17321)	0.02384691592294429
(3598, 20222)	0.05092984665532338
(3598, 13898)	0.08942548259753819
(3598, 17661)	0.0687574453061848
(3598, 18941)	0.12902511780874973
(3598, 4045)	0.060783068449284224
(3598, 10527)	0.09379709641265413
(3598, 18944)	0.2119791529494695
(3598, 18247)	0.021471690804779037
(3598, 18253)	0.024968265048099708
```

CountVectorizer

In [5]:
```
countv=CountVectorizer()
countv_ngram_features=countv.fit_transform(df['description'])
print('看一下countv_ngram_features的值：\n{}'.format(countv_ngram_features))
```

```
看一下countv_ngram_features的值：
  (0, 18683)	1
  (0, 10848)	1
  (0, 1257)	1
  (0, 18253)	1
  (0, 18247)	1
  (0, 18944)	3
  (0, 10460)	2
  (0, 19187)	4
  (0, 2581)	2
  (0, 13596)	1
  (0, 6676)	1
  (0, 9691)	1
  (0, 3228)	1
  (0, 20776)	1
  (0, 19356)	1
  (0, 7951)	4
  (0, 8668)	1
  (0, 13037)	1
  (0, 9020)	2
  (0, 20580)	2
  (0, 8931)	1
  (0, 14334)	1
  (0, 2265)	2
  (0, 13094)	1
  (0, 19399)	1
  :	:
  (3598, 17314)	1
  (3598, 6610)	1
  (3598, 19005)	3
  (3598, 20646)	1
  (3598, 16784)	1
  (3598, 7757)	1
  (3598, 5452)	2
  (3598, 11466)	1
  (3598, 581)	1
  (3598, 4911)	1
```

```
  (3598, 8992)	1
  (3598, 15084)	1
  (3598, 20506)	3
  (3598, 7864)	1
  (3598, 14426)	1
  (3598, 14390)	1
  (3598, 17144)	1
  (3598, 7886)	1
  (3598, 7862)	1
  (3598, 5113)	2
  (3598, 16710)	1
  (3598, 20531)	1
  (3598, 2098)	1
  (3598, 14938)	1
  (3598, 5734)	1
```

Random Forest Classifier

In [6]:

```
# 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

      travel     0.9423    0.9142    0.9280       268
   art_music     0.9364    0.8371    0.8840       264
        food     0.9664    0.8136    0.8834       177
     history     0.8046    0.9434    0.8685       371

    accuracy                         0.8889      1080
   macro avg     0.9124    0.8771    0.8910      1080
weighted avg     0.8975    0.8889    0.8895      1080
```

In [7]:
```
# CountVec + RFC
X_train, X_test, y_train, y_test = train_test_split(countv_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

      travel     0.9504    0.9291    0.9396       268
   art_music     0.9573    0.8485    0.8996       264
        food     0.9603    0.8192    0.8841       177
     history     0.8199    0.9569    0.8831       371

    accuracy                         0.9009      1080
   macro avg     0.9219    0.8884    0.9016      1080
weighted avg     0.9088    0.9009    0.9013      1080
```

Complement Naive Bayes

In [13]:
```
# 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

      travel     0.8905    0.9403    0.9147       268
   art_music     0.9587    0.8788    0.9170       264
        food     0.9023    0.8870    0.8946       177
     history     0.9081    0.9326    0.9202       371

    accuracy                         0.9139      1080
   macro avg     0.9149    0.9097    0.9116      1080
weighted avg     0.9151    0.9139    0.9139      1080
```

In [14]:
```
# CountVec + CNB
X_train, X_test, y_train, y_test = train_test_split(countv_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

      travel     0.9029    0.9366    0.9194       268
   art_music     0.9283    0.8826    0.9049       264
        food     0.8595    0.8983    0.8785       177
     history     0.9235    0.9111    0.9172       371

    accuracy                         0.9083      1080
   macro avg     0.9035    0.9071    0.9050      1080
weighted avg     0.9091    0.9083    0.9084      1080
```

Выводы:

In [ ]: