Рубежный контроль №2

Васильев А.Р. ИУ5-24М, 2021г.

Тема: Методы обработки текстов

Решение задачи классификации текстов.

Необходимо решить задачу классификации текстов на основе любого выбранного Вами датасета:

(Классификация может быть бинарной или многоклассовой. Целевой признак из выбранного Вами датасета может иметь любой физический смысл)

- 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 ### Группа: ИУ5-24М ### Классификатор 1: KNeighborsClassifier ### Классификатор 2: Complement Naive Bayes (CNB)
- Для каждого метода необходимо оценить качество классификации
- Сделать вывод о том, какой вариант векторизации признаков в паре с каким классификатором показал лучшее качество.

Выбранный Датасет

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
!pip install category_encoders
```

```
Collecting category_encoders
  Downloading https://files.pythonhosted.org/packages/44/57/fcef41c248701ee62e832502
6b90c432adea35555cbc870aff9cfba23727/category_encoders-2.2.2-py2.py3-none-any.whl (8
0kB)
     |████████████████████████████████| 81kB 3.7MB/s
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.7/dist-package
s (from category_encoders) (0.5.1)
Requirement already satisfied: pandas>=0.21.1 in /usr/local/lib/python3.7/dist-packa
ges (from category_encoders) (1.1.5)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.7/dist
-packages (from category_encoders) (0.22.2.post1)
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.7/dist-packag
es (from category_encoders) (1.19.5)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.7/dist-p
ackages (from category_encoders) (0.10.2)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-package
s (from category_encoders) (1.4.1)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from p
atsy>=0.5.1->category_encoders) (1.15.0)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-package
```

```
s (from pandas>=0.21.1->category_encoders) (2018.9)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/di
st-packages (from pandas>=0.21.1->category_encoders) (2.8.1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-package
s (from scikit-learn>=0.20.0->category_encoders) (1.0.1)
Installing collected packages: category-encoders
Successfully installed category-encoders-2.2.2
```

In [5]:

```
!pip install kaggle
import os
os.environ['KAGGLE_USERNAME'] = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
os.environ['KAGGLE_KEY'] = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
!kaggle datasets download clmentbisaillon/fake-and-real-news-dataset
!unzip fake-and-real-news-dataset.zip
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages (1.
5.12)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (fr
om kaggle) (2020.12.5)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-packa
ges (from kaggle) (5.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (f
rom kaggle) (2.23.0)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packages
(from kaggle) (1.15.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from
kaggle) (4.41.1)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-pack
ages (from kaggle) (2.8.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (fr
om kaggle) (1.24.3)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/dist-
packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-pa
ckages (from requests->kaggle) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-package
s (from requests->kaggle) (2.10)
fake-and-real-news-dataset.zip: Skipping, found more recently modified local copy (u
se --force to force download)
Archive:  fake-and-real-news-dataset.zip
replace Fake.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename:
```

Анализируем датасет и готовим категориальный признак

In [46]:

```
SUBSAMPLE_SIZE = 10000

df_fake = pd.read_csv('Fake.csv', encoding='utf-8')[:SUBSAMPLE_SIZE]
df_fake['target'] = np.zeros(df_fake.shape[0], dtype=np.int8)
df_true = pd.read_csv('True.csv', encoding='utf-8')[:SUBSAMPLE_SIZE]
df_true['target'] = np.ones(df_true.shape[0], dtype=np.int8)
df = pd.concat((df_fake, df_true), axis=0)
df.sample(frac=1).reset_index(drop=True)
df
```

Out[46]:

	title	text	subject	date	target
0	Donald Trump Sends Out Embarrassing New Year’...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017	0
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017	0

	title	text	subject	date	target
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017	0
3	Trump Is So Obsessed He Even Has Obama’s Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017	0
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017	0
...	...	...	...	...	...
9995	Obama says Clinton never jeopardized national ...	WASHINGTON (Reuters) - U.S. President Barack O...	politicsNews	April 10, 2016	1
9996	U.S. plans to curb tax 'inversions' could hit ...	LONDON (Reuters) - Planned changes that Presid...	politicsNews	April 11, 2016	1
9997	U.S. Democrat Clinton downplays chance of cont...	WASHINGTON (Reuters) - Democratic front-runner...	politicsNews	April 10, 2016	1
9998	Boston Globe denounces Trump candidacy in 'fro...	(Reuters) - Headlines screaming “Deportations ...	politicsNews	April 10, 2016	1
9999	Lawyers evasive about ex-U.S. House speaker's ...	(Reuters) - Former U.S. House Speaker Dennis H...	politicsNews	April 9, 2016	1

20000 rows × 5 columns

In [47]:

```
df.target.value_counts()
```

Out[47]:

```
1    10000
0    10000
Name: target, dtype: int64
```

In [48]:

```
df.subject.value_counts()
```

Out[48]:

```
politicsNews    10000
News             9050
politics          950
Name: subject, dtype: int64
```

In [49]:

```
from category_encoders import TargetEncoder
from sklearn.preprocessing import StandardScaler, LabelEncoder

encoder = LabelEncoder()
scaler = StandardScaler()
df['subject'] = encoder.fit_transform(df.subject)
```

In [50]:

```
df.subject.unique()
```

Out[50]:

```
array([0, 1, 2])
```

In [51]:

```
(df.subject - df.target).sum()
```

Out[51]:

```
10950
```

In [52]:

```
df.date.value_counts()
```

Out[52]:

```
November 9, 2016     115
April 7, 2017         73
February 1, 2017      51
February 2, 2017      49
January 23, 2017      47
                    ...
May 22, 2016           1
May 28, 2016           1
December 22, 2017      1
November 19, 2017      1
December 4, 2017       1
Name: date, Length: 1480, dtype: int64
```

In [53]:

```
df.drop(columns=['date'], inplace=True)
```

Сразу делим данные на две выборки train и test

In [54]:

```
from sklearn.model_selection import train_test_split

X = df[[i for i in df.columns if i !='target']]
y = df.target

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y,

x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

Out[54]:

```
((16000, 3), (16000,), (4000, 3), (4000,))
```

Предобрабатываем текстовые данные

In [55]:

```
for title in df.title:
  print(title, type(title))
  break
```

```
 Donald Trump Sends Out Embarrassing New Year’s Eve Message; This is Disturbing <cla
ss 'str'>
```

In [56]:

```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
import re

def preproc_func(title):
  # make more preprocessing if it well be needed!
  if isinstance(title, str):
      title = re.sub('[^a-zA-Z0-9]', ' ', title)
      return title.lower().strip()
  else:
    return ''

def get_tfidf_matrix(df, column, preproc_function, vectorizer=None):
  """
    returns matrix, trained vectorizer
  """
  processed_col = df[column].apply(preproc_function)
  if vectorizer is None:
    vectorizer = TfidfVectorizer()
    vectorizer.fit(processed_col)

  matrix = vectorizer.transform(processed_col)
  return matrix, vectorizer

def get_count_matrix(df, column, preproc_function, vectorizer=None):
  """
    returns matrix, trained vectorizer
  """
  processed_col = df[column].apply(preproc_function)
  if vectorizer is None:
    vectorizer = CountVectorizer()
    vectorizer.fit(processed_col)

  matrix = vectorizer.transform(processed_col)
  return matrix, vectorizer


train_title_matrix_tfidf, tfidf_vectorizer = get_tfidf_matrix(x_train, 'title', prep
test_title_matrix_tfidf, tfidf_vectorizer = get_tfidf_matrix(x_test, 'title',
                                                        preproc_func, vectorizer=tfid

train_title_matrix_count, count_vectorizer = get_count_matrix(x_train, 'title', prep
test_title_matrix_count, count_vectorizer = get_count_matrix(x_test, 'title',
                                                        preproc_func, vectorizer=coun
```

In [57]:

```
train_title_matrix_tfidf.shape, test_title_matrix_tfidf.shape, train_title_matrix_co
```

Out[57]:

```
((16000, 13240), (4000, 13240), (16000, 13240), (4000, 13240))
```

In [58]:

```
train_text_matrix_tfidf, tfidf_text_vectorizer = get_tfidf_matrix(x_train, 'text', p
test_text_matrix_tfidf, tfidf_text_vectorizer = get_tfidf_matrix(x_test, 'text',
                                                        preproc_func, vectorizer=tfid

train_text_matrix_count, count_text_vectorizer = get_tfidf_matrix(x_train, 'text', p
test_text_matrix_count, count_text_vectorizer = get_tfidf_matrix(x_test, 'text',
                                                        preproc_func, vectorizer=coun
```

In [59]:

```
train_text_matrix_tfidf.shape, test_text_matrix_tfidf.shape
```

Out[59]:

```
((16000, 73574), (4000, 73574))
```

In [60]:

```
train_text_matrix_count.shape, test_text_matrix_count.shape
```

Out[60]:

```
((16000, 73574), (4000, 73574))
```

In [61]:

```
from scipy import sparse

subject_train_sparse = sparse.csr_matrix(np.array(x_train.subject).reshape(-1, 1))
subject_test_sparse = sparse.csr_matrix(np.array(x_test.subject).reshape(-1, 1))

# subject_train_sparse.shape
texts_tfidf_train_matrix = sparse.hstack((train_text_matrix_tfidf,train_title_matrix
texts_count_train_matrix = sparse.hstack((train_text_matrix_count,train_title_matrix
texts_tfidf_test_matrix = sparse.hstack((test_text_matrix_tfidf,test_title_matrix_tf
texts_count_test_matrix = sparse.hstack((test_text_matrix_count,test_title_matrix_co
```

In [62]:

```
texts_tfidf_train_matrix.shape
```

```
Out[62]: (16000, 86815)
```

KNN with CountVectorizer

```
In [63]:
import sklearn
sklearn.metrics.SCORERS.keys()
```

```
Out[63]: dict_keys(['explained_variance', 'r2', 'max_error', 'neg_median_absolute_error', 'ne
g_mean_absolute_error', 'neg_mean_squared_error', 'neg_mean_squared_log_error', 'neg
_root_mean_squared_error', 'neg_mean_poisson_deviance', 'neg_mean_gamma_deviance',
'accuracy', 'roc_auc', 'roc_auc_ovr', 'roc_auc_ovo', 'roc_auc_ovr_weighted', 'roc_au
c_ovo_weighted', 'balanced_accuracy', 'average_precision', 'neg_log_loss', 'neg_brie
r_score', 'adjusted_rand_score', 'homogeneity_score', 'completeness_score', 'v_measu
re_score', 'mutual_info_score', 'adjusted_mutual_info_score', 'normalized_mutual_inf
o_score', 'fowlkes_mallows_score', 'precision', 'precision_macro', 'precision_micr
o', 'precision_samples', 'precision_weighted', 'recall', 'recall_macro', 'recall_mic
ro', 'recall_samples', 'recall_weighted', 'f1', 'f1_macro', 'f1_micro', 'f1_sample
s', 'f1_weighted', 'jaccard', 'jaccard_macro', 'jaccard_micro', 'jaccard_samples',
'jaccard_weighted'])
```

```
In [68]:
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV


parameters = {'n_neighbors': [2, 3, 5, 7, 9]}


knn_clf = KNeighborsClassifier()

knn_grid_count_clf = GridSearchCV(knn_clf, parameters, verbose=4, scoring='f1_macro'

knn_grid_count_clf.fit(texts_count_train_matrix, y_train)

pd.DataFrame(knn_grid_count_clf.cv_results_)
```

```
Fitting 3 folds for each of 5 candidates, totalling 15 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
/usr/local/lib/python3.7/dist-packages/joblib/externals/loky/process_executor.py:69
1: UserWarning: A worker stopped while some jobs were given to the executor. This ca
n be caused by a too short worker timeout or by a memory leak.
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  15 out of  15 | elapsed:  2.3min finished
```

Out[68]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_n_neighbors	params
0	0.043154	0.006815	17.902901	0.059754	2	{'n_neighbors': 2}
1	0.038842	0.003350	17.913085	0.087726	3	{'n_neighbors': 3}
2	0.032646	0.001017	18.183517	0.161322	5	{'n_neighbors': 5}
3	0.037480	0.005340	18.015832	0.229499	7	{'n_neighbors': 7}
4	0.033504	0.002923	15.156522	3.367733	9	{'n_neighbors': 9}

```
In [88]:
best_knn_count_clf = KNeighborsClassifier(n_neighbors=2)
```

```
best_knn_count_clf.fit(texts_count_train_matrix, y_train)

pred = best_knn_count_clf.predict(X=texts_count_test_matrix)

best_knn_count = classification_report(y_test, pred, digits=4, output_dict=True)

print(classification_report(y_test, pred, digits=4) )
```

```
              precision    recall  f1-score   support

           0     0.9979    0.9645    0.9809      2000
           1     0.9657    0.9980    0.9816      2000

    accuracy                         0.9812      4000
   macro avg     0.9818    0.9812    0.9812      4000
weighted avg     0.9818    0.9812    0.9812      4000
```

KNN with TfidfVectorizer

In [70]:

```
parameters = {'n_neighbors': [2, 3, 5, 7, 9]}


knn_clf = KNeighborsClassifier()

knn_grid_tfidf_clf = GridSearchCV(knn_clf, parameters, verbose=4, scoring='f1_macro'

knn_grid_tfidf_clf.fit(texts_tfidf_train_matrix, y_train)

pd.DataFrame(knn_grid_tfidf_clf.cv_results_)
```

```
Fitting 3 folds for each of 5 candidates, totalling 15 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  15 out of  15 | elapsed:  2.2min finished
```

Out[70]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_n_neighbors	params
0	0.040579	0.008529	17.776371	0.101339	2	{'n_neighbors': 2}
1	0.033366	0.002534	17.876721	0.032259	3	{'n_neighbors': 3}
2	0.032483	0.000412	17.735867	0.167531	5	{'n_neighbors': 5}
3	0.032828	0.000958	17.826399	0.103103	7	{'n_neighbors': 7}
4	0.033005	0.001057	15.127525	3.763987	9	{'n_neighbors': 9}

In [87]:

```
best_knn_clf_tfidf = KNeighborsClassifier(n_neighbors=9)

best_knn_clf_tfidf.fit(texts_tfidf_train_matrix, y_train)

pred_tfidf = best_knn_clf_tfidf.predict(X=texts_tfidf_test_matrix)

best_knn_tfidf = classification_report(y_test, pred_tfidf, digits=4, output_dict=Tru

print(classification_report(y_test, pred_tfidf, digits=4))
```

```
              precision    recall  f1-score   support

           0     1.0000    0.9995    0.9997      2000
           1     0.9995    1.0000    0.9998      2000

    accuracy                         0.9998      4000
   macro avg     0.9998    0.9998    0.9997      4000
weighted avg     0.9998    0.9998    0.9997      4000
```

Complement Bayes with CountVectorizer

In [76]:

```
from sklearn.naive_bayes import ComplementNB

parameters = {'alpha': [0, 0.5, 1, 2, 4], 'norm': [True, False]}

comp_clf = ComplementNB()

comp_grid_count_clf = GridSearchCV(comp_clf, parameters, verbose=4, scoring='f1_macr

comp_grid_count_clf.fit(texts_count_train_matrix, y_train)

pd.DataFrame(comp_grid_count_clf.cv_results_)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  21 tasks      | elapsed:    2.5s
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:    3.0s finished
```

Out[76]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	param_norm	params
0	0.082283	0.006767	0.022128	0.001450	0	True	{'alpha': 0, 'norm': True}
1	0.071416	0.003475	0.025980	0.003800	0	False	{'alpha': 0, 'norm': False}
2	0.071194	0.004742	0.024543	0.004015	0.5	True	{'alpha': 0.5, 'norm': True}
3	0.074546	0.007701	0.025762	0.003408	0.5	False	{'alpha': 0.5, 'norm': False}
4	0.081686	0.010859	0.023026	0.002556	1	True	{'alpha': 1, 'norm': True}
5	0.074433	0.000955	0.024953	0.004297	1	False	{'alpha': 1, 'norm': False}
6	0.076517	0.001306	0.021385	0.000913	2	True	{'alpha': 2, 'norm': True}

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	param_norm	params
7	0.071744	0.003374	0.026582	0.003688	2	False	{'alpha': 2, 'norm': False}
8	0.075935	0.005429	0.025764	0.003974	4	True	{'alpha': 4, 'norm': True}
9	0.088978	0.009061	0.020555	0.001498	4	False	{'alpha': 4, 'norm': False}

In [84]:

```
comp_clf = ComplementNB(alpha=4, norm=True)

comp_clf.fit(texts_count_train_matrix, y_train)
bayes_pred = comp_clf.predict(X=texts_count_test_matrix)

best_comp_count = classification_report(y_test, bayes_pred, digits=4, output_dict=Tr

print(classification_report(y_test, bayes_pred, digits=4))
```

```
              precision    recall  f1-score   support

           0     0.9899    0.9810    0.9854      2000
           1     0.9812    0.9900    0.9856      2000

    accuracy                         0.9855      4000
   macro avg     0.9855    0.9855    0.9855      4000
weighted avg     0.9855    0.9855    0.9855      4000
```

Complement Bayes with TfidfVectorizer

In [78]:

```
parameters = {'alpha': [0, 0.5, 1, 2, 4], 'norm': [True, False]}

comp_clf = ComplementNB()

comp_grid_tfidf_clf = GridSearchCV(comp_clf, parameters, verbose=4, scoring='f1_macr

comp_grid_tfidf_clf.fit(texts_tfidf_train_matrix, y_train)

pd.DataFrame(comp_grid_tfidf_clf.cv_results_)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:    1.7s finished
```

Out[78]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	param_norm	params
0	0.083233	0.002153	0.024081	0.002496	0	True	{'alpha': 0, 'norm': True}

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	param_norm	params
1	0.087036	0.004704	0.029833	0.002978	0	False	{'alpha': 0, 'norm': False}
2	0.077481	0.006106	0.025016	0.003635	0.5	True	{'alpha': 0.5, 'norm': True}
3	0.077322	0.002169	0.023997	0.002036	0.5	False	{'alpha': 0.5, 'norm': False}
4	0.083339	0.006933	0.023331	0.001769	1	True	{'alpha': 1, 'norm': True}
5	0.074882	0.002866	0.025482	0.004060	1	False	{'alpha': 1, 'norm': False}
6	0.072398	0.003656	0.026683	0.002331	2	True	{'alpha': 2, 'norm': True}
7	0.073242	0.003548	0.030089	0.005934	2	False	{'alpha': 2, 'norm': False}
8	0.076986	0.004502	0.026368	0.002535	4	True	{'alpha': 4, 'norm': True}
9	0.075503	0.006863	0.024912	0.006045	4	False	{'alpha': 4, 'norm': False}

In [86]:

```
comp_clf_tfidf = ComplementNB(alpha=0.5, norm=False)

comp_clf_tfidf.fit(texts_tfidf_train_matrix, y_train)
bayes_pred_tfidf = comp_clf_tfidf.predict(X=texts_tfidf_test_matrix)

best_comp_tfidf = classification_report(y_test, bayes_pred_tfidf, digits=4, output_d

print(classification_report(y_test, bayes_pred_tfidf, digits=4))
```

```
              precision    recall  f1-score   support

           0     0.9974    0.9735    0.9853      2000
           1     0.9741    0.9975    0.9857      2000

    accuracy                         0.9855      4000
   macro avg     0.9858    0.9855    0.9855      4000
```

```
weighted avg     0.9858     0.9855     0.9855      4000
```

Итоговое способов

In [110…

```python
best_comp_count['macro avg']

models = ['ComplementNB CountVectorizer', 'ComplementNB TfidfVectorizer', 'KNeighbor
f1 = []
precision = []
recall = []

best_comp_count['macro avg']['f1-score']

for enum, i in enumerate([best_comp_count, best_comp_tfidf, best_knn_count, best_knn
  # print(enum)
  f1.append(i['macro avg']['f1-score'])
  precision.append(i['macro avg']['precision'])
  recall.append(i['macro avg']['recall'])

pd.DataFrame({'labels':models, 'f1-score macro avg': f1, 'precision macro': precisio
```

Out[110…

	labels	f1-score macro avg	precision macro	recall_macro
0	KNeighborsClassifier TfidfVectorizer	0.999750	0.999750	0.99975
1	ComplementNB CountVectorizer	0.985500	0.985539	0.98550
2	ComplementNB TfidfVectorizer	0.985498	0.985780	0.98550
3	KNeighborsClassifier CountVectorizer	0.981245	0.981791	0.98125