Щипицина К.В. ИУ5-22М

Рубежный контроль №2

Решение задачи классификации текстов

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 TfidfVectorizer, CountVectorizer
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import precision_score, recall_score, plot_confusion_matrix, classification_report, accura
from sklearn import metrics
from typing import Dict, Tuple
from sklearn.naive_bayes import ComplementNB
from sklearn.ensemble import RandomForestClassifier
```

In [2]:
```
df=pd.read_csv("../spam.csv", encoding="latin-1")
```

In [3]:
```
# Оставим только необходимые признаки
to_drop = ["Unnamed: 2","Unnamed: 3","Unnamed: 4"]
df = df.drop(df[to_drop], axis=1)
df.rename(columns = {"v1":"target", "v2":"message"}, inplace = True)
df.head()
```

Out[3]:

	target	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...

In [4]:
```
df.target.value_counts()
```

Out[4]:
```
ham     4825
spam     747
Name: target, dtype: int64
```

In [5]:
```
# Сформируем общий словарь для обучения моделей из обучающей и тестовой выборки
vocab_list = df['message'].tolist()
vocab_list[1:5]
```

Out[5]:
```
['Ok lar... Joking wif u oni...',
 "Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry questio
n(std txt rate)T&C's apply 08452810075over18's",
 'U dun say so early hor... U c already then say...',
 "Nah I don't think he goes to usf, he lives around here though"]
```

In [6]:
```
# посмотрим на количество признаков
vocabVect = CountVectorizer()
vocabVect.fit(vocab_list)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(corpusVocab)))
```

```
Количество сформированных признаков - 8672
```

In [7]:
```
def VectorizeAndClassify(vectorizers_list, classifiers_list):
    for v in vectorizers_list:
        for c in classifiers_list:
            pipeline1 = Pipeline([("vectorizer", v), ("classifier", c)])
            score = cross_val_score(pipeline1, df['message'], df['target'], scoring='accuracy', cv=3).mean()
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('============================')
```

In [8]:
```
vectorizers_list = [CountVectorizer(), TfidfVectorizer()]
classifiers_list = [RandomForestClassifier(), ComplementNB()]
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
Векторизация - CountVectorizer()
Модель для классификации - RandomForestClassifier()
Accuracy = 0.9739768008982391
============================
Векторизация - CountVectorizer()
Модель для классификации - ComplementNB()
Accuracy = 0.9782846313727922
============================
Векторизация - TfidfVectorizer()
Модель для классификации - RandomForestClassifier()
Accuracy = 0.9746944184081064
============================
Векторизация - TfidfVectorizer()
Модель для классификации - ComplementNB()
Accuracy = 0.9784628764327956
============================
```

Использование N-грамм

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(df['message'], df['target'], test_size=0.3, random_state=1)
```

In [10]:
```
def sentiment(v, c):
    model = Pipeline(
        [("vectorizer", v),
         ("classifier", c)])
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(classification_report(y_test, y_pred, digits=4))
```

In [11]:
```
sentiment(TfidfVectorizer(), ComplementNB())
```

```
              precision    recall  f1-score   support

         ham     0.9803    0.9917    0.9860      1454
        spam     0.9403    0.8670    0.9021       218

    accuracy                         0.9755      1672
   macro avg     0.9603    0.9294    0.9441      1672
weighted avg     0.9751    0.9755    0.9751      1672
```

In [12]:
```
sentiment(TfidfVectorizer(ngram_range=(1,3)), ComplementNB())
```

```
              precision    recall  f1-score   support

         ham     0.9719    0.9993    0.9854      1454
        spam     0.9944    0.8073    0.8911       218

    accuracy                         0.9743      1672
   macro avg     0.9831    0.9033    0.9383      1672
weighted avg     0.9748    0.9743    0.9731      1672
```

In [13]:
```
sentiment(TfidfVectorizer(ngram_range=(2,3)), ComplementNB())
```

```
              precision    recall  f1-score   support

         ham     0.9847    0.9759    0.9803      1454
        spam     0.8485    0.8991    0.8731       218

    accuracy                         0.9659      1672
   macro avg     0.9166    0.9375    0.9267      1672
weighted avg     0.9670    0.9659    0.9663      1672
```

Вывод

Если оценивать качество, используя метрику accuracy, то лучшим оказался вариант TfidfVectorizer + ComplementNB.

Однако, в зависимости от целей посторения модели, необходимо обратить внимание на метрики precision, recall и f-меру.

При использовании N-грамм точность уменьшилась.