Методы машинного обучения

ИУ5-22М Ким Р.И.

Рубежный контроль №2

Задание:

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, 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: RandomForestClassifier, Complement Naive Bayes

In [3]:

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer, TfidfV
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 [4]:

```
df = pd.read_csv('https://github.com/OlegusOfficial/ML/blob/main/SP
```

In [5]:

```
df.head()
```

Out[5]:

	Category	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...

Реализуем CountVectorizer, TfidVectorizer

```
In [8]: cv = CountVectorizer()
        df_cv = cv.fit_transform(df['Message'])
        df_cv
```

```
Out[8]: <5572x8709 sparse matrix of type '<class 'numpy.int64'>'
                with 74098 stored elements in Compressed Sparse Row format
        >
```

```
In [9]: tfid = TfidfVectorizer()
        df_tfid = tfid.fit_transform(df['Message'])
        df_tfid
```

```
Out[9]: <5572x8709 sparse matrix of type '<class 'numpy.float64'>'
                with 74098 stored elements in Compressed Sparse Row format
        >
```

Реализуем классификаторы

```
In [24]: # CV + RandomForest
         X_train, X_test, y_train, y_test = train_test_split(df_cv, df['Cate
```

```
In [25]: model = RandomForestClassifier()
```

```
In [26]: model.fit(X_train, y_train)
```

```
Out[26]: RandomForestClassifier()
```

```
In [27]: y_pred = model.predict(X_test)
```

```
In [28]: print(classification_report(y_test, y_pred, digits=4))
```

```
              precision    recall  f1-score   support

         ham     0.9647    1.0000    0.9820      3384
        spam     1.0000    0.7602    0.8637       517

    accuracy                         0.9682      3901
   macro avg     0.9823    0.8801    0.9229      3901
weighted avg     0.9693    0.9682    0.9663      3901
```

```
In [29]: # Tfid + RandomForest
         X_train, X_test, y_train, y_test = train_test_split(df_tfid, df['Ca
```

```
In [30]: model = RandomForestClassifier()
```

In [31]:
```
model.fit(X_train, y_train)
```

Out[31]:
```
RandomForestClassifier()
```

In [32]:
```
y_pred = model.predict(X_test)
```

In [36]:
```
y_pred
```

Out[36]:
```
array(['ham', 'ham', 'ham', ..., 'ham', 'ham', 'ham'], dtype=objec
t)
```

In [33]:
```
print(classification_report(y_test, y_pred, digits=4))
```

```
              precision    recall  f1-score   support

         ham     0.9644    1.0000    0.9819      3384
        spam     1.0000    0.7582    0.8625       517

    accuracy                         0.9680      3901
   macro avg     0.9822    0.8791    0.9222      3901
weighted avg     0.9691    0.9680    0.9660      3901
```

In [34]:
```
# CV + NaiveBaies
X_train, X_test, y_train, y_test = train_test_split(df_cv, df['Cate
```

In [35]:
```
model = ComplementNB()
```

In [37]:
```
model.fit(X_train, y_train)
```

Out[37]:
```
ComplementNB()
```

In [38]:
```
y_pred = model.predict(X_test)
```

In [39]:
```
print(classification_report(y_test, y_pred, digits=4))
```

```
              precision    recall  f1-score   support

         ham     0.9880    0.9752    0.9816      3384
        spam     0.8503    0.9226    0.8850       517

    accuracy                         0.9682      3901
   macro avg     0.9191    0.9489    0.9333      3901
weighted avg     0.9698    0.9682    0.9688      3901
```

In [45]:

```
# Tfid + NaiveBaies
X_train, X_test, y_train, y_test = train_test_split(df_tfid, df['Ca
```

In [46]:

```
model = ComplementNB()
```

In [47]:

```
model.fit(X_train, y_train)
```

Out[47]:

```
ComplementNB()
```

In [48]:

```
y_pred = model.predict(X_test)
```

In [49]:

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

```
              precision    recall  f1-score   support

         ham     0.9695    0.9852    0.9773      3384
        spam     0.8918    0.7969    0.8417       517

    accuracy                         0.9603      3901
   macro avg     0.9306    0.8911    0.9095      3901
weighted avg     0.9592    0.9603    0.9593      3901
```

Вывод:

1. CountVectorizer + RFC/Naive показали наилучший accuracy - 0.9682