Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика, искусственный интеллект и системы управления

КАФЕДРА Системы обработки информации и управления

Отчет по рубежному контролю №2 по курсу «Методы машинного обучения»

Рубежный контроль №2

«Методы обработки текста»

Выполнил:
Тураев Г.В.
ИУ5-25М

Проверил:

Гапанюк Ю.Е

Москва, 2023

Условие.

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

Выполнение.

```
[39] import pandas as pd
     import numpy as np
     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
```

```
[41] df = pd.read_csv('https://github.com/OlegusOfficial/ML/blob/main/SPAM.csv?raw=True')
```

```
[42] df.head()
```

	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:

```
[45] cv = CountVectorizer()
     df_cv = cv.fit_transform(df['Message'])
     df_cv
```

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

```
[46] tfid = TfidfVectorizer()
     df_tfid = tfid.fit_transform(df['Message'])
     df_tfid
```

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

Теперь реализуем классификаторы:

CV + RandomForest

```
[53] X_train, X_test, y_train, y_test = train_test_split(df_cv, df['Category'], train_size=0.3, random_state=32)
```

```
[54] model = RandomForestClassifier()
```

```
[56] model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
[58] y_pred = model.predict(X_test)
```

```
[61] print(classification_report(y_test, y_pred, digits=4))
```

```
                  precision    recall  f1-score   support

             ham     0.9649    0.9997    0.9820      3384
            spam     0.9975    0.7621    0.8640       517

        accuracy                         0.9682      3901
       macro avg     0.9812    0.8809    0.9230      3901
    weighted avg     0.9692    0.9682    0.9664      3901
```

Tfid + RandomForest

```
[62] X_train, X_test, y_train, y_test = train_test_split(df_tfid, df['Category'], train_size=0.3, random_state=32)
```

```
[65] model = RandomForestClassifier()
```

```
[67] model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
[68] y_pred = model.predict(X_test)
```

```
[69] y_pred
```

```
array(['ham', 'ham', 'ham', ..., 'ham', 'ham', 'ham'], dtype=object)
```

```
[70] print(classification_report(y_test, y_pred, digits=4))
```

```
                  precision    recall  f1-score   support

             ham     0.9668    0.9994    0.9829      3384
            spam     0.9950    0.7756    0.8717       517

        accuracy                         0.9698      3901
       macro avg     0.9809    0.8875    0.9273      3901
    weighted avg     0.9706    0.9698    0.9681      3901
```

CV + NaiveBaies

```
[74] X_train, X_test, y_train, y_test = train_test_split(df_cv, df['Category'], train_size=0.3, random_state=32)
```

```
[75] model = ComplementNB()
```

```
[76] model.fit(X_train, y_train)
```

```
▾ ComplementNB
ComplementNB()
```

```
[77] y_pred = model.predict(X_test)
```

```
[79] 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
```

Tfid + NaiveBaies

```
[81] X_train, X_test, y_train, y_test = train_test_split(df_tfid, df['Category'], train_size=0.3, random_state=32)
```

```
[82] model = ComplementNB()
```

```
[83] model.fit(X_train, y_train)
```

```
▾ ComplementNB
ComplementNB()
```

```
[84] y_pred = model.predict(X_test)
```

```
[87] 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
```

Таким образом, мы видим из результатов, что лучшую точность показал Tfid + RandomForest, где accuracy составил 0.9698.