

##ММО Гринин Олег ИУ5-22М

###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

```
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
```

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

```
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, TfidfVectorizer

```
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>
```

```
tfidf = TfidfVectorizer()
df_tfidf = tfidf.fit_transform(df['Message'])
df_tfidf
```

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

Реализуем классификаторы

```
# CV + RandomForest
```

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

```
model = RandomForestClassifier()
```

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

```
RandomForestClassifier()
```

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

```
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

```
# Tfidf + RandomForest
```

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

```
model = RandomForestClassifier()
```

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

```
RandomForestClassifier()
```

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

```
y_pred
```

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

```
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

CV + NaiveBaies

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

```
model = ComplementNB()
```

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

```
ComplementNB()
```

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

```
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

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

```
model = ComplementNB()
```

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

```
ComplementNB()
```

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

```
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