Рубежный контроль №2 по курсу "Методы машинного обучения"

Выполнил: Арбузов А.П. группа ИУ5-24М

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 по варианту для Вашей группы: Классификатор №1: KNeighborsClassifier Классификатор №2: Complement Naive Bayes (CNB)

Для каждого метода необходимо оценить качество классификации. Сделайте вывод о том, какой вариант векторизации признаков в паре с каким классификатором показал лучшее качество.

В [2]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import ComplementNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

В [3]:

```
data = pd.read_csv('SPAM text message 20170820 - Data.csv', sep = ',')
data.head()
```

Out[3]:

	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...

В [4]:

```
msgContent = data['Message']
msgClass = data['Category']
```

В [5]:

```
TrainX,TestX,TrainY,TestY = train_test_split(msgContent, msgClass, test_size=0.3, randc
report = []
```

B [6]:

```
def ModelPredictReport(vectoriser, classifier, modelName, vectName):
    model = Pipeline(
        [("vectorizer", vectoriser),
         ("classifier", classifier)])
    model.fit(TrainX,TrainY)
    prediction = model.predict(TestX)
    report = [modelName, vectName]
    report.append(accuracy_score(TestY,prediction))
    report.append(recall_score(TestY,prediction,pos_label = "ham"))
    report.append(f1_score(TestY,prediction,pos_label = "ham"))
    return report
```

B [7]:

```
report.append(ModelPredictReport(CountVectorizer(),KNeighborsClassifier(),'KNN','Count\
report.append(ModelPredictReport(TfidfVectorizer(),KNeighborsClassifier(),'KNN','Tfidf\
```

B [8]:

```
report.append(ModelPredictReport(CountVectorizer(),ComplementNB(),'CNB','CountVectorize
report.append(ModelPredictReport(TfidfVectorizer(),ComplementNB(),'CNB','TfidfVectorize
```

B [9]:

```
pd.DataFrame(report, columns = ['Model','Vectorizer','Accuracy','Recall','F1'])
```

Out[9]:

	Model	Vectorizer	Accuracy	Recall	F1
0	KNN	CountVectorizer	0.907297	1.000000	0.948996
1	KNN	TfidfVectorizer	0.900120	1.000000	0.945264
2	CNB	CountVectorizer	0.979665	0.985437	0.988178
3	CNB	TfidfVectorizer	0.982656	0.995146	0.989997

Наилучшим оказался CNB с векторизатором TfidfVectorizer