Рубежный контроль №2. Методы машинного обучения

Студент: Седойкин Г.С., ИУ5-22М

Задание.

Необходимо решить задачу классификации текстов на основе любого выбранного Вами датасета (кроме примера, который рассматривался в лекции). 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 (`RandomForestClassifier` , `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.

Выполнение задания.

Задания буду выполнять на датасете из статьи "Stop Clickbait: Detecting and Preventing Clickbaits in Online News Media".
Датасет предполагает бинарную классификацию.

In [4]:
```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from scipy import stats
from sklearn.naive_bayes import ComplementNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classifi
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squ
```

Датасет состоит из 2х файлов (кликбейт и не кликбейт заголовки). Загрузим их, установим значение целевого признак, объеденим датасеты в единый DataFrame и перемешаем строки:

In [5]:
```
data_clickbait = pd.read_csv("../data/clickbait_data.txt", sep = "\n\n", engi
data_clickbait["is_clickbait"] = [1] * data_clickbait.shape[0]
data_non_clickbait = pd.read_csv("../data/non_clickbait_data.txt", sep = "\n\
data_non_clickbait["is_clickbait"] = [0] * data_non_clickbait.shape[0]
dataset = data_clickbait.append(data_non_clickbait)
```

```
dataset = dataset.sample(frac = 1, random_state = 100)
dataset = dataset.reset_index(drop = True)
dataset
```

Out[5]:

	header_text	is_clickbait
0	In Michigan, Bank Lends Little of Its Bailout ...	0
1	Four dead, more than a million in U.S. without...	0
2	In Wyoming, Debate Rages Over Elk Feeding Program	0
3	Bryant and Lakers Return to the N.B.A. Finals	0
4	This Baby's Reaction To Hearing About How She ...	1
...	...	...
31995	When You Binge-Watch "Making A Murderer" And T...	1
31996	Schiphol airliner crash blamed on altimeter fa...	0
31997	Radiohead Release Rejected Bond Theme Song And...	1
31998	Chernobyl Taking a Toll on Invertebrates Too	0
31999	8 Things No One Tells Guys With Body Image Anx...	1

32000 rows × 2 columns

Сформируем общий словарь:

In [10]:
```
vocabulary = CountVectorizer().fit(dataset["header_text"].tolist()).vocabular
print("Размер словаря: {}".format(len(vocabulary)))
```

```
Размер словаря: 22761
```

Обучим модели используя кросс-валидацию и разные векторизации:

In [17]:
```
def VectorizeAndClassify(vectorizers_list, classifiers_list):
    for vectorizer in vectorizers_list:
        for classifier in classifiers_list:
            pipeline = Pipeline([("vectorizer", vectorizer), ("classifier", c
            score = cross_val_score(pipeline, dataset["header_text"], dataset
                                    scoring = 'accuracy', cv = 3, n_jobs = -1
            print('Векторизация - {}'.format(vectorizer))
            print('Модель для классификации - {}'.format(classifier))
            print('Accuracy = {}'.format(score))
            print('============================')
        print("\n")

vectorizers_list = [CountVectorizer(vocabulary = vocabulary), TfidfVectorizer
classifiers_list = [RandomForestClassifier(), ComplementNB()]

VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '000th': 2, '00
s': 3, '01': 4,
                               '04': 5, '05': 6, '08': 7, '08m': 8, '09': 9,
                               '10': 10, '100': 11, '1000': 12, '10000th': 13,
                               '1000blackgirls': 14, '100k': 15, '100m': 16,
                               '100th': 17, '100°f': 18, '101': 19, '101st': 20,
                               '102': 21, '103': 22, '104': 23, '105': 24,
                               '106': 25, '107': 26, '108': 27, '109': 28,
                               '109th': 29, ...})
Модель для классификации - RandomForestClassifier()
Accuracy = 0.9566250135875828
============================
```

```
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '000th': 2, '00
s': 3, '01': 4,
                               '04': 5, '05': 6, '08': 7, '08m': 8, '09': 9,
                               '10': 10, '100': 11, '1000': 12, '10000th': 13,
                               '1000blackgirls': 14, '100k': 15, '100m': 16,
                               '100th': 17, '100°f': 18, '101': 19, '101st': 20,
                               '102': 21, '103': 22, '104': 23, '105': 24,
                               '106': 25, '107': 26, '108': 27, '109': 28,
                               '109th': 29, ...})
Модель для классификации - ComplementNB()
Accuracy = 0.9718125370554792
============================


Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '000th': 2, '00
s': 3, '01': 4,
                               '04': 5, '05': 6, '08': 7, '08m': 8, '09': 9,
                               '10': 10, '100': 11, '1000': 12, '10000th': 13,
                               '1000blackgirls': 14, '100k': 15, '100m': 16,
                               '100th': 17, '100°f': 18, '101': 19, '101st': 20,
                               '102': 21, '103': 22, '104': 23, '105': 24,
                               '106': 25, '107': 26, '108': 27, '109': 28,
                               '109th': 29, ...})
Модель для классификации - RandomForestClassifier()
Accuracy = 0.9575937245257542
============================
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '000th': 2, '00
s': 3, '01': 4,
                               '04': 5, '05': 6, '08': 7, '08m': 8, '09': 9,
                               '10': 10, '100': 11, '1000': 12, '10000th': 13,
                               '1000blackgirls': 14, '100k': 15, '100m': 16,
                               '100th': 17, '100°f': 18, '101': 19, '101st': 20,
                               '102': 21, '103': 22, '104': 23, '105': 24,
                               '106': 25, '107': 26, '108': 27, '109': 28,
                               '109th': 29, ...})
Модель для классификации - ComplementNB()
Accuracy = 0.96787500189044
============================
```

Вывод

Наилучший результат был получен для векторизатора - `CountVectorizer` с классификатром `ComplementNB()` : *0.9718125370554792*.