**Рубежный контроль №2  
Студент: Попов М.Ю.  
Группа: ИУ5-25М**

## Решение задачи классификации текстов.

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:

|  |  |  |
| --- | --- | --- |
| ИУ5-25М, ИУ5И-25М, ИУ5И-26М | [SVC](https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html) | [LogisticRegression](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html) |

Импортируем библиотеки

from sklearn.feature\_extraction.text import CountVectorizer, TfidfVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

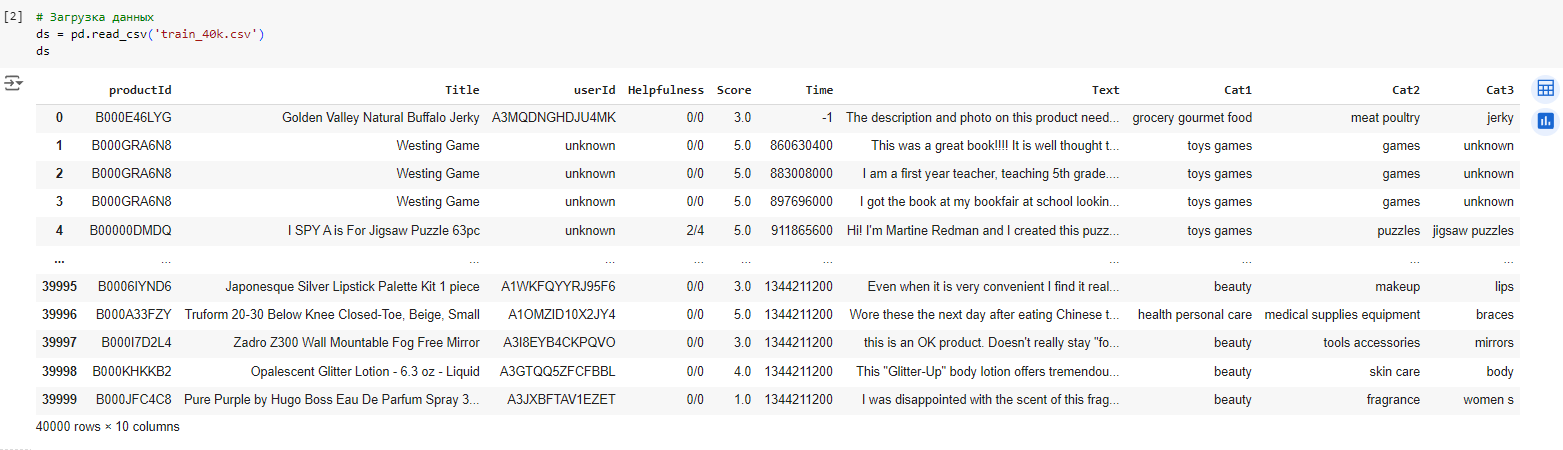
from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

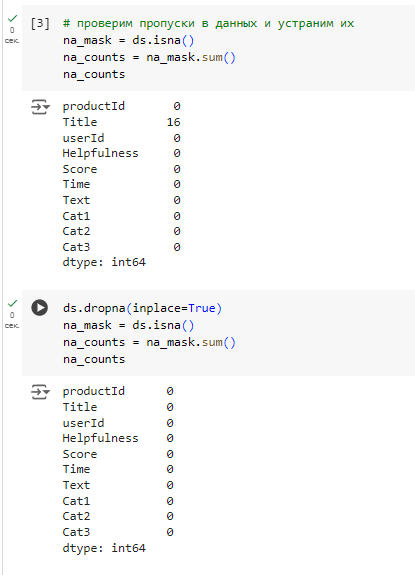
import pandas as pd

import time

Загружаем набор данных



В качестве исходных данных был выбран набор данных 'train\_40k.csv’. В нём некоторые признаки содержат пропуски:



Разделим набор данных на обучающую и тестувую выборки

X, Y = ds['Text'], ds['Cat1']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=42)

time\_arr = []

# векторизация признаков с помощью CountVectorizer

count\_vect = CountVectorizer()

X\_train\_counts = count\_vect.fit\_transform(X\_train)

X\_test\_counts = count\_vect.transform(X\_test)

# векторизация признаков с помощью TfidfVectorizer

tfidf\_vect = TfidfVectorizer()

X\_train\_tfidf = tfidf\_vect.fit\_transform(X\_train)

X\_test\_tfidf = tfidf\_vect.transform(X\_test)

Часть 1

Произведем обучения вдух классификаторов (по варианту) для CountVectorizer

# SVC

gbc = SVC()

start\_time = time.time()

gbc.fit(X\_train\_counts, y\_train)

train\_time = time.time() - start\_time

time\_arr.append(train\_time)

pred\_gbc\_counts = gbc.predict(X\_test\_counts)

print("Точность (CountVectorizer + SVC):", accuracy\_score(y\_test, pred\_gbc\_counts))

# Logistic Regression

lr = LogisticRegression(max\_iter=1000)

start\_time = time.time()

lr.fit(X\_train\_counts, y\_train)

train\_time = time.time() - start\_time

time\_arr.append(train\_time)

pred\_lr\_counts = lr.predict(X\_test\_counts)

print("Точность (CountVectorizer + LogisticRegression):", accuracy\_score(y\_test, pred\_lr\_counts))

Часть 2

Произведем обучения вдух классификаторов (по варианту) для TfidfVectorizer

# SVC

gbc = SVC()

start\_time = time.time()

gbc.fit(X\_train\_tfidf, y\_train)

train\_time = time.time() - start\_time

time\_arr.append(train\_time)

pred\_gbc\_tfidf = gbc.predict(X\_test\_tfidf)

print("Точность (TfidfVectorizer + LinearSVC):", accuracy\_score(y\_test, pred\_gbc\_tfidf))

# Logistic Regression

lr = LogisticRegression(max\_iter=1000)

start\_time = time.time()

lr.fit(X\_train\_tfidf, y\_train)

train\_time = time.time() - start\_time

time\_arr.append(train\_time)

pred\_lr\_tfidf = lr.predict(X\_test\_tfidf)

print("Точность (TfidfVectorizer + LogisticRegression):", accuracy\_score(y\_test, pred\_lr\_tfidf))

Сравним полученные результаты:

from tabulate import tabulate

data = [

    ["(CountVectorizer + LogisticRegression)", accuracy\_score(y\_test, pred\_lr\_counts), time\_arr[0]],

    ["(CountVectorizer + LinearSVC)", accuracy\_score(y\_test, pred\_gbc\_counts), time\_arr[1]],

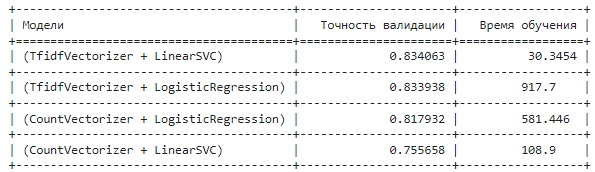
    ["(TfidfVectorizer + LogisticRegression)", accuracy\_score(y\_test, pred\_lr\_tfidf), time\_arr[2]],

    ["(TfidfVectorizer + LinearSVC)", accuracy\_score(y\_test, pred\_gbc\_tfidf), time\_arr[3]]

]

sorted\_data = sorted(data, key=lambda x: x[1], reverse=True)

print(tabulate(sorted\_data, ['Модели','Точность валидации', 'Время обучения'], tablefmt="grid"))



Вывод: Лучше всего показала себя пара TfidfVectorizer + LinearSVC