МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Систем обработки информации и управления»

ОТЧЕТ

**Рубежный контроль №2**

по дисциплине«Методы машинного обучения»

Тема: «Методы обработки текстов»  
Вариант 9

ИСПОЛНИТЕЛЬ: \_\_\_Очеретная С.В.\_\_

ФИО

группа ИУ5-25М \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

"\_\_"\_\_\_\_\_\_\_\_\_2024 г.

ПРЕПОДАВАТЕЛЬ: \_\_\_\_\_Гапанюк Ю.Е.\_\_\_\_

ФИО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

"\_\_"\_\_\_\_\_\_\_\_\_2024 г.

Москва - 2024

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Задание

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 группы (ИУ5-25М): **SVC, LogisticRegression**.

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

# Ход работы

## Импорт данных и библиотек

Импорт библиотек

import numpy as np  
import pandas as pd  
import matplotlib as mpl  
import matplotlib.pyplot as plt  
%matplotlib inline  
import seaborn as sns  
sns.set(style="ticks")  
  
from sklearn.feature\_extraction.text import CountVectorizer, TfidfVectorizer  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LogisticRegression  
from sklearn.svm import SVC  
from typing import Dict, Tuple  
from sklearn.model\_selection import cross\_val\_predict  
from sklearn.metrics import classification\_report  
from sklearn.pipeline import Pipeline

Подключение к диску

from google.colab import drive  
drive.mount('/content/drive/')

Импорт датасета

data = pd.read\_csv("/content/drive/My Drive/Учеба/магистратура/2 сем/ММО/Airline Occurences.csv")

Размер набора данных пришлось сократить, т.к. один запуск длился несколько часов (а нам нужно 4 запуска, а это нескольких дней из-за лимитов коллаба)

data = data[0:1000]

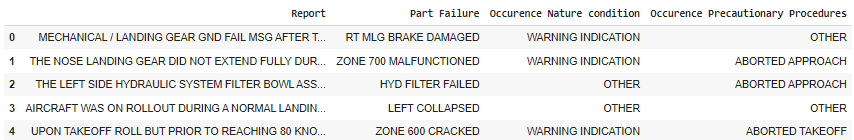
data.shape

(1000, 4)

## Исследование и предобработка набора

Импортируемый набор данных содержит отчеты об инцидентах, представленные пилотами, авиадиспетчерами и другими авиационными специалистами. Он включает в себя данные о неисправностях оборудования и предпринятых корректирующих действиях.

data.head()



Размер набора данных:

data.shape

(1000, 4)

Проверим набор данных на null значения:

print(data.isnull().sum())

Report 0  
Part Failure 0  
Occurence Nature condition 0  
Occurence Precautionary Procedures 0  
dtype: int64

Null значений нет, значит можем продолжить работу с набором.

Теперь проверим уникальные значения столбцов:

print(data.nunique())

Report 997Part Failure 716Occurence Nature condition 18Occurence Precautionary Procedures 11dtype: int64

Для столбцов "Occurence Precautionary Procedures", "Occurence Nature condition" выведем список уникальных значений:

data["Occurence Precautionary Procedures"].unique()

array(['OTHER ', 'ABORTED APPROACH ',  
 'ABORTED TAKEOFF ', 'EMER. DESCENT ',  
 'O2 MASK DEPLOYED ', 'UNSCHED LANDING ',  
 'NONE ', 'RETURN TO BLOCK ',  
 'DEACTIVATE SYST/CIRCUITS', 'ENGINE SHUTDOWN ',  
 'DUMP FUEL '], dtype=object)

data["Occurence Nature condition"].unique()

array(['WARNING INDICATION ', 'OTHER ',  
 'ENGINE FLAMEOUT ', 'ELECT. POWER LOSS-50 PC ',  
 'SMOKE/FUMES/ODORS/SPARKS ', 'FLUID LOSS ',  
 'FALSE WARNING ', 'NO TEST ',  
 'PARTIAL RPM/PWR LOSS ', 'VIBRATION/BUFFET ',  
 'AFFECT SYSTEMS ', 'FLT CONT AFFECTED ',  
 'MULTIPLE FAILURE ', 'SIGNIFICANT FAILURE REPORT',  
 'INADEQUATE Q C ', 'NO WARNING INDICATION ',  
 'OVER TEMP ', 'FLT. ATTITUDE INST. '],  
 dtype=object)

Как можно заметить выше, в значениях столбцов у нас есть пробелы в конце. Удалим их и также приведем к нижнему регистру.

data['Report'] = data['Report'].str.strip().str.lower()  
data['Part Failure'] = data['Part Failure'].str.strip().str.lower()  
data['Occurence Nature condition'] = data['Occurence Nature condition'].str.strip().str.lower()  
data['Occurence Precautionary Procedures'] = data['Occurence Precautionary Procedures'].str.strip().str.lower()

Теперь пробелов больше нет и значения в нижнем регистре:

data["Occurence Precautionary Procedures"].unique()

array(['other', 'aborted approach', 'aborted takeoff', 'emer. descent',  
 'o2 mask deployed', 'unsched landing', 'none', 'return to block',  
 'deactivate syst/circuits', 'engine shutdown', 'dump fuel'],  
 dtype=object)

Далее разделим выборку на тестовую и обучающую. В качестве целевого признака возьмем столбец "Occurence Nature condition", описывающий причину возникновения аварии. Столбец "Occurence Precautionary Procedures" рассматривать не будем.

X = data[['Report', 'Part Failure']]  
y = data['Occurence Nature condition']  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

## Векторизация признаков

По заданию необходимо сформировать два варианта векторизации признаков - на основе CountVectorizer и на основе TfidfVectorizer.

Сначала сформируем общий словарь для обучения моделей из обучающей и тестовой выборки. В словаре будет текст из 2х столбцов

vocab\_list = data[['Report', 'Part Failure']].apply(lambda x: ' '.join(x), axis=1)  
vocab\_list[1:10]

1 the nose landing gear did not extend fully dur...  
2 the left side hydraulic system filter bowl ass...  
3 aircraft was on rollout during a normal landin...  
4 upon takeoff roll but prior to reaching 80 kno...  
5 failure of the #1 engine hp fuel pump drive co...  
6 75 amp emergency battery circuit breaker on co...  
7 crew smelled an odor, took actions to isolate ...  
8 per pilot report: during climb(thru fl360) le...  
9 enroute from lsgg-rjaa with 4 crew on board an...  
dtype: object

### CountVectorizer:

count\_vectorizer = CountVectorizer()  
count\_vectorizer.fit(vocab\_list)  
count\_vectorizer\_vocab = count\_vectorizer.vocabulary\_  
print('Количество сформированных признаков - {}'.format(len(count\_vectorizer\_vocab)))

Количество сформированных признаков - 4817

Просмотрим некоторые из слов сформированного с помощью CountVectorizer словаря:

for i in list(count\_vectorizer\_vocab)[1:10]:  
 print('{}={}'.format(i, count\_vectorizer\_vocab[i]))

landing=3072  
gear=2703  
gnd=2712  
fail=2418  
msg=3344  
after=1395  
takeoff=4416  
emergency=2314  
declared=2078

### TfidfVectorizer:

tfidf\_vectorizer = TfidfVectorizer()  
tfidf\_vectorizer.fit(vocab\_list)  
tfidf\_vectorizer\_vocab = tfidf\_vectorizer.vocabulary\_  
print('Количество сформированных признаков - {}'.format(len(tfidf\_vectorizer\_vocab)))

Количество сформированных признаков - 4817

for i in list(count\_vectorizer\_vocab)[1:10]:  
 print('{}={}'.format(i, tfidf\_vectorizer\_vocab[i]))

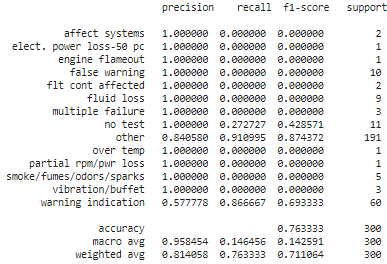
landing=3072  
gear=2703  
gnd=2712  
fail=2418  
msg=3344  
after=1395  
takeoff=4416  
emergency=2314  
declared=2078

## Классификация

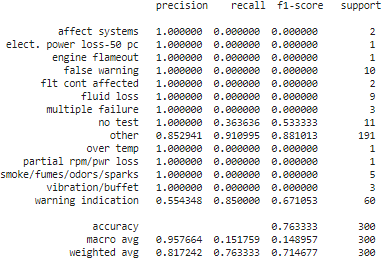
По моему варианту необходимо использовать методы SVC, LogisticRegression. Проверим данные методы совместно с рассмотренными выше вариантами векторизации.

def test\_model(v, c):  
 model = c  
 X\_train\_vec = v.fit\_transform(X\_train.apply(lambda x: ' '.join(x), axis=1))  
 X\_test\_vec = v.transform(X\_test.apply(lambda x: ' '.join(x), axis=1))  
  
 model.fit(X\_train\_vec, y\_train)  
 y\_pred = model.predict(X\_test\_vec)  
  
 print('Метод векторизации: {}'.format(v))  
 print('Метод классификации: {}'.format(c))  
 print('Оценка точности:\n', classification\_report(y\_test, y\_pred, zero\_division=True, digits=6))

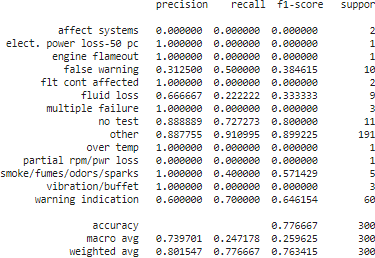
test\_model(CountVectorizer(), SVC())

Метод векторизации: CountVectorizer()  
Метод классификации: SVC()  
Оценка точности:  


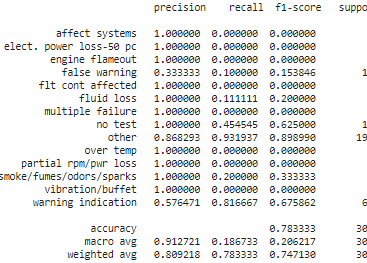
test\_model(TfidfVectorizer(), SVC())

Метод векторизации: TfidfVectorizer()  
Метод классификации: SVC()  
Оценка точности:  


test\_model(CountVectorizer(), LogisticRegression(C=3.0))

Метод векторизации: CountVectorizer()  
Метод классификации: LogisticRegression(C=3.0)  
Оценка точности:  


test\_model(TfidfVectorizer(), LogisticRegression(C=3.0))

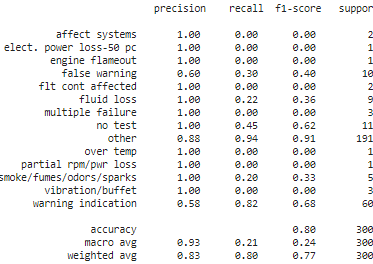
Метод векторизации: TfidfVectorizer()  
Метод классификации: LogisticRegression(C=3.0)  
Оценка точности:  


На основе 4х запусков можно сделать следующие выводы:

* точность методов векторизации оказалась одинаковой для классификатора SVC (где-то точность 1го класса была лучше, где-то другого, но в целом точность одинаковая). Но с классификатором LogisticRegression метод векторизации TfidfVectorizer показал результат лучше.
* метод логистической регрессии оказался лучше, чем SVC

Также можно попробовать запустить логистическую регрессию с другими параметрами:

test\_model(TfidfVectorizer(), LogisticRegression(C=5.0))

Метод векторизации: TfidfVectorizer()  
Метод классификации: LogisticRegression(C=5.0)  
Оценка точности:  


Таким образом, результат оказался еще лучше. Если дальше увеличивать C, то результат уже не меняется. Значит итоговая точность - 80%. Наилучшую точность получили с методом векторизации TfidfVectorizer и классификатором LogisticRegression(C=5.0).