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()
```

	Report	Part Failure	Occurence Nature condition	Occurence Precautionary Procedures
0	MECHANICAL / LANDING GEAR GND FAIL MSG AFTER T...	RT MLG BRAKE DAMAGED	WARNING INDICATION	OTHER
1	THE NOSE LANDING GEAR DID NOT EXTEND FULLY DUR...	ZONE 700 MALFUNCTIONED	WARNING INDICATION	ABORTED APPROACH
2	THE LEFT SIDE HYDRAULIC SYSTEM FILTER BOWL ASS...	HYD FILTER FAILED	OTHER	ABORTED APPROACH
3	AIRCRAFT WAS ON ROLLOUT DURING A NORMAL LANDIN...	LEFT COLLAPSED	OTHER	OTHER
4	UPON TAKEOFF ROLL BUT PRIOR TO REACHING 80 KNO...	ZONE 600 CRACKED	WARNING INDICATION	ABORTED TAKEOFF

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

	precision	recall	f1-score	support
affect systems	1.000000	0.000000	0.000000	2
elect. power loss-50 pc	1.000000	0.000000	0.000000	1
engine flameout	1.000000	0.000000	0.000000	1
false warning	1.000000	0.000000	0.000000	10
flt cont affected	1.000000	0.000000	0.000000	2
fluid loss	1.000000	0.000000	0.000000	9
multiple failure	1.000000	0.000000	0.000000	3
no test	1.000000	0.272727	0.428571	11
other	0.840580	0.910995	0.874372	191
over temp	1.000000	0.000000	0.000000	1
partial rpm/pwr loss	1.000000	0.000000	0.000000	1
smoke/fumes/odors/sparks	1.000000	0.000000	0.000000	5
vibration/buffet	1.000000	0.000000	0.000000	3
warning indication	0.577778	0.866667	0.693333	60
accuracy			0.763333	300
macro avg	0.958454	0.146456	0.142591	300
weighted avg	0.814058	0.763333	0.711064	300

```
test_model(TfidfVectorizer(), SVC())
```

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

	precision	recall	f1-score	support
affect systems	1.000000	0.000000	0.000000	2
elect. power loss-50 pc	1.000000	0.000000	0.000000	1
engine flameout	1.000000	0.000000	0.000000	1
false warning	1.000000	0.000000	0.000000	10
flt cont affected	1.000000	0.000000	0.000000	2
fluid loss	1.000000	0.000000	0.000000	9
multiple failure	1.000000	0.000000	0.000000	3
no test	1.000000	0.363636	0.533333	11
other	0.852941	0.910995	0.881013	191
over temp	1.000000	0.000000	0.000000	1
partial rpm/pwr loss	1.000000	0.000000	0.000000	1
smoke/fumes/odors/sparks	1.000000	0.000000	0.000000	5
vibration/buffet	1.000000	0.000000	0.000000	3
warning indication	0.554348	0.850000	0.671053	60
accuracy			0.763333	300
macro avg	0.957664	0.151759	0.148957	300
weighted avg	0.817242	0.763333	0.714677	300

```
test_model(CountVectorizer(), LogisticRegression(C=3.0))
```

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

Оценка точности:

	precision	recall	f1-score	suppor
affect systems	0.000000	0.000000	0.000000	2
elect. power loss-50 pc	1.000000	0.000000	0.000000	1
engine flameout	1.000000	0.000000	0.000000	1
false warning	0.312500	0.500000	0.384615	10
flt cont affected	1.000000	0.000000	0.000000	2
fluid loss	0.666667	0.222222	0.333333	9
multiple failure	1.000000	0.000000	0.000000	3
no test	0.888889	0.727273	0.800000	11
other	0.887755	0.910995	0.899225	191
over temp	1.000000	0.000000	0.000000	1
partial rpm/pwr loss	0.000000	0.000000	0.000000	1
smoke/fumes/odors/sparks	1.000000	0.400000	0.571429	5
vibration/buffet	1.000000	0.000000	0.000000	3
warning indication	0.600000	0.700000	0.646154	60
accuracy			0.776667	300
macro avg	0.739701	0.247178	0.259625	300
weighted avg	0.801547	0.776667	0.763415	300

```
test_model(TfidfVectorizer(), LogisticRegression(C=3.0))
```

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

	precision	recall	f1-score	suppo
affect systems	1.000000	0.000000	0.000000	
elect. power loss-50 pc	1.000000	0.000000	0.000000	
engine flameout	1.000000	0.000000	0.000000	
false warning	0.333333	0.100000	0.153846	1
flt cont affected	1.000000	0.000000	0.000000	
fluid loss	1.000000	0.111111	0.200000	
multiple failure	1.000000	0.000000	0.000000	
no test	1.000000	0.454545	0.625000	1
other	0.868293	0.931937	0.898990	19
over temp	1.000000	0.000000	0.000000	
partial rpm/pwr loss	1.000000	0.000000	0.000000	
smoke/fumes/odors/sparks	1.000000	0.200000	0.333333	
vibration/buffet	1.000000	0.000000	0.000000	
warning indication	0.576471	0.816667	0.675862	6
accuracy			0.783333	30
macro avg	0.912721	0.186733	0.206217	30
weighted avg	0.809218	0.783333	0.747130	30

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:

	precision	recall	f1-score	suppor
affect systems	1.00	0.00	0.00	2
elect. power loss-50 pc	1.00	0.00	0.00	1
engine flameout	1.00	0.00	0.00	1
false warning	0.60	0.30	0.40	10
flt cont affected	1.00	0.00	0.00	2
fluid loss	1.00	0.22	0.36	9
multiple failure	1.00	0.00	0.00	3
no test	1.00	0.45	0.62	11
other	0.88	0.94	0.91	191
over temp	1.00	0.00	0.00	1
partial rpm/pwr loss	1.00	0.00	0.00	1
smoke/fumes/odors/sparks	1.00	0.20	0.33	5
vibration/buffet	1.00	0.00	0.00	3
warning indication	0.58	0.82	0.68	60
accuracy			0.80	300
macro avg	0.93	0.21	0.24	300
weighted avg	0.83	0.80	0.77	300

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).