



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления и искусственный интеллект
КАФЕДРА Системы обработки информации и управления

Домашняя работа №1

По курсу

«Методы машинного обучения»

На тему:

«Классификация текста»

Подготовил:

Студент группы

ИУ5-25М Клюкин Н. А.

27.03.2024

Проверил:

Гапанюк Ю.Е.

2024 г.

- **Цель домашней работы:**

Задание

- Для произвольного набора данных, предназначенного для классификации текстов, решите задачу классификации текста двумя способами:
 - Способ 1. На основе CountVectorizer или TfidfVectorizer.
 - Способ 2. На основе моделей word2vec или Glove или fastText.

- Сравните качество полученных моделей.

Импорт библиотек

```
In [1]: import numpy as np
import pandas as pd
from typing import Dict, Tuple
from scipy import stats
from IPython.display import Image
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
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, classification
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_l
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

Подгрузка датасета и подготовительные действия

- Используем набор Depressive/Non-Depressive Tweets Data - Депрессивные/недепрессивные твиты в период с декабря 2019 года по декабрь 2020 года

```
In [2]: # Загрузка данных
df = pd.read_csv("datasets/clean_tweet.csv")

# Удаление первой колонки
df = df.drop(df.columns[0], axis=1)
df = df.rename(columns={'sentiment': 'value'})

df.head()
```

```
Out[2]:
```

	text	value
0		0
1	rising cases of covid does not alarm me rising...	1
2	please vote for chicagoindiareolution marking...	0
3	wishing all of you eidaladha hazrat ibrahim as...	1
4	daily coronavirus cases in india top for first...	1

```
In [3]: df.shape
```

```
Out[3]: (134348, 2)
```

Датасет слишком большой, а также содержит пропуски. Возьмём часть и проведем очистку данных

```
In [4]: df = df.sample(frac=0.05)
df = df.dropna()
```

```
In [5]: X_train, X_test, y_train, y_test = train_test_split(df['text'], df['value'], test_s
```

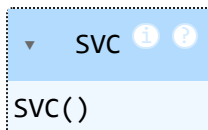
1. Решение задачи классификации текстов с использованием классификатора SVC на основе TfidfVectorizer

```
In [7]: tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
```

```
In [8]: svc_classifier_tfidf = SVC()
```

```
In [9]: svc_classifier_tfidf.fit(X_train_tfidf, y_train)
```

```
Out[9]:
```



```
In [10]: X_test_tfidf = tfidf_vectorizer.transform(X_test)
svc_accuracy = svc_classifier_tfidf.score(X_test_tfidf, y_test)
print(f'Точность для метода векторизации TfidfVectorizer через классификатор SVC =
```

Точность для метода векторизации TfidfVectorizer через классификатор SVC = 0.8325892857142857

2. На основе модели word2vec

```
In [15]: from gensim.models import word2vec
import re
import pandas as pd
import numpy as np
from typing import Dict, Tuple
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from nltk import WordPunctTokenizer
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\NKliukin\AppData\Roaming\nltk_data...
[nltk_data] Unzipping corpora\stopwords.zip.
```

Out[15]: True

```
In [16]: # Подготовим корпус
corpus = []
stop_words = stopwords.words('english')
tok = WordPunctTokenizer()
for line in df['text'].values:
    line1 = line.strip().lower()
    line1 = re.sub("[^a-zA-Z]", " ", line1)
    text_tok = tok.tokenize(line1)
    text_tok1 = [w for w in text_tok if not w in stop_words]
    corpus.append(text_tok1)
```

```
In [17]: corpus[:5]
```

```
Out[17]: [['tbt',
           'male',
           'coworker',
           'previous',
           'job',
           'gifted',
           'self',
           'help',
           'book',
           'staying',
           'single',
           'called',
           'single',
           'superpower',
           'told',
           'ck',
           'front',
           'people',
           'asked',
           'take',
           'back',
           'men',
           'seriously',
           'audacity'],
          ['stay', 'side'],
          ['really', 'looks', 'like', 'daehyun'],
          ['literally',
           'never',
           'understand',
           'people',
           'dislike',
           'sun',
           'burned',
           'badly',
           'many',
           'times',
           'life',
           'get',
           'heatstroke',
           'insanely',
           'easily',
           'could',
           'sit',
           'sun',
           'warmth',
           'hour',
           'every',
           'day',
           'pretty',
           'sure',
           'would',
           'even',
           'depressed'],
          ['rt',
           'provided',
           'irrefutable',
```

```
'evidence',  
'india',  
'state',  
'sponsored',  
'terrorism',  
'inside',  
'pak',  
'details',  
'financial',  
'material',  
'support',  
'indian',  
'state',  
'direct',  
'involvement',  
'terrorism',  
'given',  
'world']]
```

```
In [18]: # количество текстов в корпусе не изменилось и соответствует целевому признаку  
assert df.shape[0]==len(corpus)
```

```
In [19]: %time model = word2vec.Word2Vec(corpus, workers=4, min_count=10, window=10, sample=
```

CPU times: total: 547 ms

Wall time: 230 ms

```
In [24]: # Проверим, что модель обучилась  
print(model.wv.most_similar(positive=['inside'], topn=5))
```

```
[('suicide', 0.9993067383766174), ('responsible', 0.9992725253105164), ('coming', 0.  
999215841293335), ('said', 0.9991757869720459), ('team', 0.9991641044616699)]
```

Подведение результатов

Таким образом, можно заключить, что наилучшие результаты были получены с использованием модели word2vec, что связано с тем, что эта модель использует нейронные сети, которые позволяют более гибко проводить классификацию.