

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №2

**по дисциплине «Прикладные интеллектуальные системы и экспертные
системы»**

«Предварительная обработка текстовых данных»

Студент

Цыганов Н.А.

Группы М-ИАП-23

Руководитель

Кургасов В.В.

Доцент

Липецк 2023 г

Цель работы

Получить практические навыки решения задачи бинарной классификации данных в среде Jupiter Notebook. Научиться загружать данные, обучать классификаторы и проводить классификацию. Научиться оценивать точность полученных моделей.

Задание кафедры

Задание:

- 1) В среде Jupiter Notebook создать новый ноутбук (Notebook)
- 2) Импортировать необходимые для работы библиотеки и модули
- 3) Загрузить обучающую и экзаменационную выборку в соответствие с вариантом
- 4) Вывести на экран по одному-два документа каждого класса.
- 5) Применить стемминг, записав обработанные выборки (тестовую и обучающую) в новые переменные.
- 6) Провести векторизацию выборки:
 - a. Векторизовать обучающую и тестовую выборки простым подсчетом слов (CountVectorizer) и значением `max_features = 10000`
 - b. Вывести и проанализировать первые 20 наиболее частотных слов всей выборки и каждого класса по-отдельности.
 - c. Применить процедуру отсека стоп-слов и повторить пункт b.
 - d. Провести пункты a – c для обучающей и тестовой выборки, для которой проведена процедура стемминга.
 - e. Векторизовать выборки с помощью TfidfTransformer (с использованием TF и TF-IDF взвешиваний) и повторить пункты b-d.
- 7) По результатам пункта 6 заполнить таблицы наиболее частотными терминами обучающей выборки и каждого класса по отдельности. Всего должно получиться по 4 таблицы для выборки, к которой применялась операция стемминга и 4 таблицы для выборки, к которой операция стемминга не применялась
- 8) Используя конвейер (Pipeline) реализовать модель Наивного Байесовского классификатора и выявить на основе показателей качества (значения полноты, точности, f1-меры и аккуратности), какая предварительная обработка данных обеспечит наилучшие результаты классификации. Должны быть исследованы следующие характеристики:

- Наличие - отсутствие стемминга
- Отсечение – не отсечение стоп-слов
- Количество информативных терминов (max_features)
- Взвешивание: Count, TF, TF-IDF

9) По каждому пункту работы занести в отчет программный код и результат вывода.

10) По результатам классификации занести в отчет выводы о наиболее подходящей предварительной обработке данных (наличие стемминга, взвешивание терминов, стоп-слова, количество информативных терминов).

Ход работы

Вариант по журналу 17, вариантов 12, следовательно: вариант 5 представлен на рисунке 1.

5	4, 14, 18
4	'comp.sys.ibm.pc.hardware'
14	'sci.med'
18	'talk.politics.mideast'

Рисунок 1 - Вариант для выполнения

На рисунке 2 изображен импорт библиотек для загрузки данных.

```
: import warnings
import nltk
from sklearn.datasets import fetch_20newsgroups
warnings.simplefilter(action='ignore', category=FutureWarning)
```

Рисунок 2 - Импорт необходимых библиотек

```
: categories = ['comp.sys.ibm.pc.hardware', 'sci.med', 'talk.politics.mideast']
remove = ('headers', 'footers', 'quotes')

twenty_train_full = fetch_20newsgroups(subset='train', categories=categories, shuffle=True, random_state=42, remove=remove)
twenty_test_full = fetch_20newsgroups(subset='test', categories=categories, shuffle=True, random_state=42, remove=remove)
```

Рисунок 3 - Загрузка данных

Пример загруженных данных на рисунке 4.

```
[3]: print(twenty_train_full.data[2])
```

Let's face it, if the words don't get into your noggin in the first place, there's no hope. Now tell us, 'SDPA.ORG', a mouthpiece of the fascist x-Soviet Armenian Government: what was your role in the murder of Orhan Gunduz and Kemal Arikan? How many more Muslims will be slaughtered by 'SDPA.ORG' as publicly declared and filed with the legal authorities?

"...that more people have to die..."

SDPA <91@urartu.UUCP>

"Yes, I stated this and stand by it."

SDPA <255@urartu.UUCP>

January 28, 1982 - Los Angeles
Kemal Arikan is slaughtered by two Armenians while driving to work.

March 22 1982 - Cambridge Massachusetts

Рисунок 4 - Вывод данных

Далее необходимо применить стэмминг, это показано на рисунке 5

```
In [6]: def stemming(data):  
        porter_stemmer = PorterStemmer()  
        stem = []  
        for text in data:  
            nltk_tokens = word_tokenize(text)  
            line = ''.join([' ' + porter_stemmer.stem(word) for word in nltk_tokens])  
            stem.append(line)  
        return stem
```

```
In [7]: stem_train = stemming(twenty_train_full.data)  
        stem_test = stemming(twenty_test_full.data)
```

```
In [8]: print(stem_train[0])
```

ay > in mani recent advertis i have seen both `` 486dx-50 '' and `` 486dx ay > base system . doe the first realli ex
ist and doe it impli that all ay > circuitri on the motherboard with it work at that speed , as opposit ay > latter ,
where onli the intern of the cpu are work at 50mhz ? ay > ay > mani thanx in advanc ! ay > ay > andrew . andrew , ye
there is a dx and dx2 version of the 50mhz 486 . if you are consid buy one or the other , definit go for the dx with
a nice size extern cach ! the perform is far greater . the dx2 onli ha the intern 8k cach to work with at 50mhz , whi
le the dx ha a potenti much larger cach to work at 50mhz with . neither system could actual run a program out of main
memori , sinc dram is still too slow for that high of bu speed (60n = 16.66mhz < 50mhz) . -rdd -- - . winqwk 2.0b #
0 . unregist evalu copi * kmail 2.95d w-net hq , hal9k.ann-arbor.mi.u , +1 313 663 4173 or 3959

```
In [9]: print(stem_test[0])
```

whi ? there is no need to go into thi espec i thi rivet piec of inform . as i rememb , someon did ask if uv h
ad a speach code . but , realli , there is no need for thi brief survey cours . how wonder for you .

Рисунок 5 - Данные после стемминга

Векторизация представлена на рисунке 6.

```
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
```

```
# Векторизация обучающей и тестовой выборки простым подсчетом слов (CountVectorizer) и значением max_features = 10.000
vect_without_stop = CountVectorizer(max_features=10000)
```

```
train_data = vect_without_stop.fit_transform(twenty_train_full.data)
test_data = vect_without_stop.transform(twenty_test_full.data)
```

```
def sort_by_tf(input_str):
    return input_str[1]

def top_terms(vector, data, count):
    x = list(zip(vector.get_feature_names_out(), np.ravel(data.sum(axis=0))))
    x.sort(key=sort_by_tf, reverse=True)
    return x[:count]
```

b. Вывести и проанализировать первые 20 наиболее частотных слов всей выборки и каждого класса по-отдельности.

```
top_terms_without_stop = [{term[0]: term[1]} for term in top_terms(vect_without_stop, train_data, 20)]
top_terms_without_stop

top_terms_without_stop_test = [{term[0]: term[1]} for term in top_terms(vect_without_stop, test_data, 20)]
top_terms_without_stop_test
```

```
[{'the': 14650},
 {'of': 7249},
 {'to': 6579},
 {'and': 6307},
 {'in': 4663},
 {'that': 3192},
 {'is': 3189},
 {'it': 2491},
 {'for': 2343},
 {'on': 1807},
 {'you': 1799},
 {'with': 1691},
 {'this': 1654},
 {'have': 1545},
 {'as': 1533},
 {'was': 1485},
 {'not': 1481},
 {'are': 1478},
 {'be': 1343},
 {'by': 1340}]
```

Рисунок 6 - Векторизация простым подсчетом слов

Отсечем стоп-слова и повторим предыдущее действие, это показано на рисунке 7.

с. Применить процедуру отсеечения стоп-слов и повторить пункт б.

```
: vect_stop = CountVectorizer(max_features=10000, stop_words='english')

: train_data_stop = vect_stop.fit_transform(twenty_train_full.data)
  test_data_stop = vect_stop.transform(twenty_test_full.data)

: top_terms_stop = [{term[0]: term[1]} for term in top_terms(vect_stop, train_data_stop, 20)]
  top_terms_stop

top_terms_stop_test = [{term[0]: term[1]} for term in top_terms(vect_stop, test_data_stop, 20)]
top_terms_stop_test

: [{'people': 558},
   {'like': 439},
   {'just': 407},
   {'know': 399},
   {'armenian': 392},
   {'don': 374},
   {'said': 366},
   {'time': 317},
   {'armenians': 314},
   {'new': 309},
   {'does': 307},
   {'israel': 304},
   {'use': 294},
   {'ve': 271},
   {'think': 258},
   {'jews': 256},
   {'did': 247},
   {'jewish': 246},
   {'good': 227},
   {'92': 217}]
```

Рисунок 7 - Векторизация с отсечением стоп-слов

Теперь проведем векторизацию после стэмминга без использования стоп слов, это показано на рисунке 8.

d. Провести пункты а – с для обучающей и тестовой выборки, для которой проведена процедура стэмминга.

```
In [18]: vect_stem_without_stop = CountVectorizer(max_features=10000)

In [19]: train_data_without_stop_stem = vect_stem_without_stop.fit_transform(stem_train)
         test_data_without_stop_stem = vect_stem_without_stop.transform(stem_test)

In [20]: top_terms_stem = [{term[0]: term[1]} for term in top_terms(vect_stem_without_stop, train_data_without_stop_stem, 20)]
         top_terms_stem

top_terms_stem_test = [{term[0]: term[1]} for term in top_terms(vect_stem_without_stop, test_data_without_stop_stem, 20)]
top_terms_stem_test

Out[20]: [{'the': 14650},
          {'of': 7249},
          {'to': 6580},
          {'and': 6307},
          {'in': 4664},
          {'is': 3244},
          {'that': 3201},
          {'it': 2798},
          {'for': 2343},
          {'on': 1809},
          {'you': 1798},
          {'with': 1691},
          {'have': 1668},
          {'thi': 1654},
          {'be': 1532},
          {'not': 1532},
          {'as': 1531},
          {'are': 1511},
          {'wa': 1508},
          {'by': 1338}]
```

Рисунок 8 - Векторизация после стэмминга без использования стоп слов

Проведем векторизацию после стэмминга с использованием стоп-слов, это представлено на рисунке 9

```
d. Провести пункты а - с для обучающей и тестовой выборки, для которой проведена процедура стемминга. Применена процедура отсеечения стоп слов
```

```
]: vect_stem = CountVectorizer(max_features=10000, stop_words='english')

]: train_data_stop_stem = vect_stem.fit_transform(stem_train)
   test_data_stop_stem = vect_stem.transform(stem_test)

]: top_terms_stop_stem = [{term[0]: term[1]} for term in top_terms(vect_stem, train_data_stop_stem, 20)]
   top_terms_stop_stem

top_terms_stop_stem_test = [{term[0]: term[1]} for term in top_terms(vect_stem, test_data_stop_stem, 20)]
top_terms_stop_stem_test

]: [{'thi': 1654},
    {'wa': 1508},
    {'ha': 748},
    {'armenian': 706},
    {'use': 687},
    {'peopl': 577},
    {'ani': 551},
    {'like': 476},
    {'know': 452},
    {'hi': 418},
    {'time': 408},
    {'just': 407},
    {'doe': 390},
    {'muslim': 376},
    {'onli': 371},
    {'did': 369},
    {'said': 366},
    {'year': 360},
    {'work': 342},
    {'new': 330}]
```

Рисунок 9 - Векторизация после стэмминга с использованием стоп слов

Далее необходимо векторизовать выборки с помощью TfidfTransformer (с использованием TF и TF-IDF взвешиваний) и повторить пункты b-d, это

показано на рисунках 10-13

```
Без использования стоп-слов

: from sklearn.feature_extraction.text import TfidfTransformer

: tf = TfidfTransformer(use_idf=False)
: tfidf = TfidfTransformer(use_idf=True)

: train_data_tf = tf.fit_transform(train_data)
: test_data_tf = tf.transform(test_data)

: train_data_tfidf = tfidf.fit_transform(train_data)
: test_data_tfidf = tfidf.transform(test_data)

: top_terms_tf = [{term[0]: term[1]} for term in top_terms(vect_without_stop, train_data_tf, 20)]
: top_terms_tf

: top_terms_tf_test = [{term[0]: term[1]} for term in top_terms(vect_without_stop, test_data_tf, 20)]
: top_terms_tf_test

: top_terms_tfidf = [{term[0]: term[1]} for term in top_terms(vect_without_stop, train_data_tfidf, 20)]
: top_terms_tfidf

: top_terms_tfidf_test = [{term[0]: term[1]} for term in top_terms(vect_without_stop, test_data_tfidf, 20)]
: top_terms_tfidf_test

: [{ 'the': 159.2169690232235},
:   { 'to': 89.02190713477806},
:   { 'of': 83.45129921232822},
:   { 'and': 73.7766936838158},
:   { 'in': 60.65224299548882},
:   { 'is': 58.103868781563605},
:   { 'that': 53.177739920859196},
:   { 'it': 48.24061768176479},
:   { 'you': 45.649903936602},
:   { 'for': 42.72281944930034},
:   { 'this': 37.901483549050184},
:   { 'have': 35.800297109784495},
:   { 'on': 33.416342525930226},
:   { 'are': 32.4915980194941},
:   { 'not': 32.124698387113376},
:   { 'with': 31.14359170735619},
:   { 'as': 30.193569888878972},
:   { 'be': 29.125311312829243},
:   { 'or': 27.659896044595506},
:   { 'was': 26.0177820514447}]
```

Рисунок 10 - Без стоп слов, без стэмминга

С использованием стоп-слов

```
: tf = TfidfTransformer(use_idf=False)
tfidf = TfidfTransformer(use_idf=True)

: train_data_stop_tf = tf.fit_transform(train_data_stop)
test_data_stop_tf = tf.transform(test_data_stop)

train_data_stop_tfidf = tfidf.fit_transform(train_data_stop)
test_data_stop_tfidf = tfidf.transform(test_data_stop)

: top_terms_stop_tf = [{term[0]: term[1]} for term in top_terms(vect_stop, train_data_stop_tf, 20)]
top_terms_stop_tf

top_terms_stop_tf_test = [{term[0]: term[1]} for term in top_terms(vect_stop, test_data_stop_tf, 20)]
top_terms_stop_tf_test

top_terms_stop_tfidf = [{term[0]: term[1]} for term in top_terms(vect_stop, train_data_stop_tfidf, 20)]
top_terms_stop_tfidf

top_terms_stop_tfidf_test = [{term[0]: term[1]} for term in top_terms(vect_stop, test_data_stop_tfidf, 20)]
top_terms_stop_tfidf_test

: [{'know': 19.78579921737784},
 {'like': 17.72640179203392},
 {'people': 17.7089654853161},
 {'just': 17.631226432525903},
 {'don': 17.583760771958584},
 {'does': 17.22809995409527},
 {'drive': 16.39874301043245},
 {'thanks': 15.453835857073658},
 {'ve': 14.080995472361815},
 {'use': 14.06988552250666},
 {'think': 13.975194482131897},
 {'israel': 13.496273726006486},
 {'good': 12.577620578291661},
 {'card': 12.276145758372772},
 {'did': 11.839884300272598},
 {'need': 11.59412077354018},
 {'jews': 11.412098006263552},
 {'problem': 11.396809292843278},
 {'muslims': 11.244332076286735},
 {'information': 10.72185387994127}]
```

Рисунок 11 - Со стоп-словами, без стэмминга

Со стеммингом без стоп-слов

```
tf = TfidfTransformer(use_idf=False)
tfidf = TfidfTransformer(use_idf=True)

train_data_stem_tf = tf.fit_transform(train_data_without_stop_stem)
test_data_stem_tf = tf.transform(test_data_without_stop_stem)

train_data_stem_tfidf = tfidf.fit_transform(train_data_without_stop_stem)
test_data_stem_tfidf = tfidf.transform(test_data_without_stop_stem)

top_terms_stem_tf = [{term[0]: term[1]} for term in top_terms(vect_stem_without_stop, train_data_stem_tf, 20)]
top_terms_stem_tf

top_terms_stem_tf_test = [{term[0]: term[1]} for term in top_terms(vect_stem_without_stop, test_data_stem_tf, 20)]
top_terms_stem_tf_test

top_terms_stem_tfidf = [{term[0]: term[1]} for term in top_terms(vect_stem_without_stop, train_data_stem_tfidf, 20)]
top_terms_stem_tfidf

top_terms_stem_tfidf_test = [{term[0]: term[1]} for term in top_terms(vect_stem_without_stop, test_data_stem_tfidf, 20)]
top_terms_stem_tfidf_test

[{'the': 157.2388941502928},
 {'to': 88.25882757708708},
 {'of': 82.15071361679951},
 {'and': 72.81946837666995},
 {'in': 59.67514134362165},
 {'is': 58.74700675437686},
 {'that': 53.114529463275495},
 {'it': 50.782768659461645},
 {'you': 45.24027565870208},
 {'for': 42.32351737189909},
 {'have': 37.89706132683007},
 {'this': 37.640352013093114},
 {'on': 32.969598013545124},
 {'are': 32.733953715289694},
 {'not': 32.53959169338456},
 {'do': 32.37287087935601},
 {'be': 31.33777132043142},
 {'with': 30.746733916646296},
 {'as': 29.68626987221284},
 {'or': 27.490714607187865}]
```

Рисунок 12 - Без стоп слов, со стэммингом

Со стеммингом с использованием стоп-слов

```
tf = TfidfTransformer(use_idf=False)
tfidf = TfidfTransformer(use_idf=True)
```

```
train_data_stem_stop_tf = tf.fit_transform(train_data_stop_stem)
test_data_stem_stop_tf = tf.transform(test_data_stop_stem)
```

```
train_data_stem_stop_tfidf = tfidf.fit_transform(train_data_stop_stem)
test_data_stem_stop_tfidf = tfidf.transform(test_data_stop_stem)
```

```
top_terms_stem_stop_tf = [{term[0]: term[1]} for term in top_terms(vect_stem, train_data_stop_tf, 20)]
top_terms_stem_stop_tf
```

```
top_terms_stem_stop_tf_test = [{term[0]: term[1]} for term in top_terms(vect_stem, test_data_stop_tf, 20)]
top_terms_stem_stop_tf_test
```

```
top_terms_stem_stop_tfidf = [{term[0]: term[1]} for term in top_terms(vect_stem, train_data_stop_tfidf, 20)]
top_terms_stem_stop_tfidf
```

```
top_terms_stem_stop_tfidf_test = [{term[0]: term[1]} for term in top_terms(vect_stem, test_data_stop_tfidf, 20)]
top_terms_stem_stop_tfidf_test
```

```
[{'kuvvetli': 38.58159096500337},
 {'liu': 36.12468490625079},
 {'keysystem': 35.07900385428372},
 {'papa': 33.899805109818885},
 {'drain': 33.22247165159427},
 {'download': 31.356326600450455},
 {'tire': 24.756486685752243},
 {'va': 23.68612643837865},
 {'tile': 23.60052634582734},
 {'dunya': 21.770502304466586},
 {'vertic': 21.18233857137946},
 {'hada': 19.81800160459357},
 {'tour': 19.631632693113296},
 {'joke': 19.001475698647734},
 {'dip': 18.620097671734133},
 {'powel': 18.270319216644406},
 {'nan': 17.341536187667092},
 {'broccoli': 16.80690032882596},
 {'honeywell': 16.734896985500633},
 {'weyrich': 16.175907790854883}]
```

Рисунок 13 - Со стоп-словами и стэммингом

Теперь необходимо заполнить таблицы наиболее частотными терминами обучающей выборки и каждого класса по отдельности.

```
import pandas as pd
```

```
columns = pd.MultiIndex.from_product(['Count', 'TF', 'TF-IDF'], ['Без стоп-слов', 'С стоп-словами'])
```

Без стемминга

```
df1 = pd.DataFrame(columns=columns)

df1['Count', 'Без стоп-слов'] = top_terms_without_stop
df1['TF', 'Без стоп-слов'] = top_terms_tf
df1['TF-IDF', 'Без стоп-слов'] = top_terms_tfidf

df1['Count', 'С стоп-словами'] = top_terms_stop
df1['TF', 'С стоп-словами'] = top_terms_stop_tf
df1['TF-IDF', 'С стоп-словами'] = top_terms_stop_tfidf

df1
```

	Count		TF		TF-IDF	
	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
0	{'the': 20927}	{'people': 1049}	{'the': 593.3256841596507}	{'know': 54.741560260006736}	{'the': 227.9558306920963}	{'know': 27.728806445196387}
1	{'of': 10253}	{'know': 727}	{'to': 324.35056386604447}	{'just': 52.77569413890487}	{'to': 128.8488479541235}	{'just': 26.560026322737713}
2	{'to': 9895}	{'like': 705}	{'of': 284.32768892589365}	{'like': 52.615200558633546}	{'of': 122.79902042561119}	{'people': 25.89595434402102}
3	{'and': 9303}	{'don': 690}	{'and': 246.055767993875}	{'people': 50.36062593469473}	{'and': 104.65751352906601}	{'like': 25.473994640289288}
4	{'in': 6686}	{'just': 678}	{'is': 211.8971731527514}	{'don': 48.3356858391453}	{'is': 90.84365943836319}	{'don': 25.138891964059667}
5	{'that': 5243}	{'said': 617}	{'it': 185.64787315807015}	{'does': 43.35324301900809}	{'in': 86.92944107213987}	{'israel': 24.77736759387494}
6	{'is': 5012}	{'armenian': 611}	{'in': 185.35842967934371}	{'time': 37.27135075539442}	{'it': 84.30246063746429}	{'drive': 24.12998277979711}
7	{'it': 4456}	{'edu': 587}	{'that': 168.34202229537448}	{'think': 36.292810556154315}	{'that': 79.44737741122822}	{'does': 23.83766430970143}
8	{'you': 3309}	{'time': 571}	{'you': 131.24534400434962}	{'israel': 35.5649078550382}	{'you': 74.58316802538543}	{'edu': 20.714882299147984}
9	{'for': 3249}	{'drive': 523}	{'for': 116.12627141885021}	{'drive': 32.30302778343974}	{'for': 58.15598723309614}	{'time': 20.349971601319805}
10	{'on': 2649}	{'armenians': 511}	{'this': 94.73104189498784}	{'edu': 32.04577092336103}	{'this': 50.54309921413386}	{'think': 20.319287163702608}
11	{'with': 2582}	{'use': 509}	{'have': 90.34966254353837}	{'use': 31.273268013974864}	{'not': 49.02490900988524}	{'scsi': 19.654863630271507}
12	{'they': 2458}	{'israel': 502}	{'with': 88.53034271562834}	{'thanks': 29.87072552182338}	{'are': 48.95164833232471}	{'thanks': 19.2117559423666}
13	{'this': 2421}	{'turkish': 474}	{'on': 87.38918274513544}	{'good': 26.74248581100969}	{'have': 48.24550427656501}	{'card': 19.165641352111816}
14	{'are': 2413}	{'scsi': 466}	{'not': 86.91825929049}	{'card': 26.492309800567398}	{'on': 47.8903986738175}	{'use': 18.351130157961457}
15	{'not': 2409}	{'think': 452}	{'are': 84.46973508912207}	{'problem': 26.022392993607035}	{'with': 47.12072738733223}	{'good': 16.541818286397422}
16	{'was': 2407}	{'does': 449}	{'be': 79.94429654337597}	{'help': 25.264808465903076}	{'be': 45.22377483664475}	{'help': 16.153930034104715}
17	{'have': 2243}	{'jews': 424}	{'as': 67.25737298825979}	{'did': 24.7113547918968}	{'as': 42.44215634594219}	{'problem': 16.099688236177283}
18	{'be': 2098}	{'did': 401}	{'or': 66.304294374055}	{'new': 24.524137168129766}	{'my': 39.23353188963498}	{'israeli': 15.900155757721613}
19	{'as': 2062}	{'new': 381}	{'if': 60.82774382644909}	{'way': 23.039159762462827}	{'or': 38.69725612115456}	{'did': 15.810382536473046}

Рисунок 14 - Результат векторизации для обучающего множества без
СТЭММИНГА

```
df2 = pd.DataFrame(columns=columns)

df2['Count', 'Без стоп-слов'] = top_terms_without_stop_test
df2['TF', 'Без стоп-слов'] = top_terms_tf_test
df2['TF-IDF', 'Без стоп-слов'] = top_terms_tfidf_test

df2['Count', 'С стоп-словами'] = top_terms_stop_test
df2['TF', 'С стоп-словами'] = top_terms_stop_tf_test
df2['TF-IDF', 'С стоп-словами'] = top_terms_stop_tfidf_test

df2
```

	Count		TF		TF-IDF	
	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
0	{'the': 14650}	{'people': 558}	{'the': 403.27852840515715}	{'know': 38.58159096500337}	{'the': 159.2169690232235}	{'know': 19.78579921737784}
1	{'of': 7249}	{'like': 439}	{'to': 217.90584846740072}	{'like': 36.12468490625079}	{'to': 89.02190713477806}	{'like': 17.72640179203392}
2	{'to': 6579}	{'just': 407}	{'of': 188.1102669338932}	{'just': 35.07900385428372}	{'of': 83.45129921232822}	{'people': 17.7089654853161}
3	{'and': 6307}	{'know': 399}	{'and': 169.6239850732778}	{'people': 33.899805109818885}	{'and': 73.7766936838158}	{'just': 17.631226432525903}
4	{'in': 4663}	{'armenian': 392}	{'is': 133.31865743350403}	{'don': 33.22247165159427}	{'in': 60.65224299548882}	{'don': 17.583760771958584}
5	{'that': 3192}	{'don': 374}	{'in': 125.38323764466679}	{'does': 31.356326600450455}	{'is': 58.103868781563605}	{'does': 17.22809995409527}
6	{'is': 3189}	{'said': 366}	{'that': 109.29535702682418}	{'think': 24.756486685752243}	{'that': 53.177739920859196}	{'drive': 16.39874301043245}
7	{'it': 2491}	{'time': 317}	{'it': 104.13977257385334}	{'use': 23.68612643837865}	{'it': 48.24061768176479}	{'thanks': 15.453835857073658}
8	{'for': 2343}	{'armenians': 314}	{'for': 83.38528167692749}	{'thanks': 23.60052634582734}	{'you': 45.649903936602}	{'ve': 14.080995472361815}
9	{'on': 1807}	{'new': 309}	{'you': 77.5189215357162}	{'drive': 21.770502304466586}	{'for': 42.72281944930034}	{'use': 14.06988552250666}
10	{'you': 1799}	{'does': 307}	{'this': 69.59671509746121}	{'ve': 21.18233857137946}	{'this': 37.901483549050184}	{'think': 13.975194482131897}
11	{'with': 1691}	{'israel': 304}	{'have': 64.8416836220305}	{'good': 19.81800160459357}	{'have': 35.800297109784495}	{'israel': 13.496273726006486}
12	{'this': 1654}	{'use': 294}	{'on': 59.378895325157465}	{'time': 19.631632693113296}	{'on': 33.416342525930226}	{'good': 12.577620578291661}
13	{'have': 1545}	{'ve': 271}	{'with': 56.453491676804056}	{'israel': 19.001475698647734}	{'are': 32.4915980194941}	{'card': 12.276145758372772}
14	{'as': 1533}	{'think': 258}	{'not': 55.012670530019044}	{'did': 18.620097671734133}	{'not': 32.124698387113376}	{'did': 11.839884300272598}
15	{'was': 1485}	{'jews': 256}	{'are': 54.230257103284806}	{'problem': 18.270319216644406}	{'with': 31.14359170735619}	{'need': 11.59412077354018}
16	{'not': 1481}	{'did': 247}	{'be': 49.588587994454585}	{'need': 17.341536187667092}	{'as': 30.19356988878972}	{'jews': 11.412098006263552}
17	{'are': 1478}	{'jewish': 246}	{'as': 46.076040061871375}	{'card': 16.80690032882596}	{'be': 29.125311312829243}	{'problem': 11.396809292843278}
18	{'be': 1343}	{'good': 227}	{'or': 45.885330296526945}	{'help': 16.734896985500633}	{'or': 27.659896044595506}	{'muslims': 11.244332076286735}
19	{'by': 1340}	{'92': 217}	{'but': 41.98162538060753}	{'way': 16.175907790854883}	{'was': 26.0177820514447}	{'information': 10.72185387994127}

Рисунок 15 - Результат векторизации для тестового набора без стэмминга

Со стеммингом

```
df3 = pd.DataFrame(columns=columns)

df3['Count', 'Без стоп-слов'] = top_terms_stem
df3['TF', 'Без стоп-слов'] = top_terms_stem_tf
df3['TF-IDF', 'Без стоп-слов'] = top_terms_stem_tfidf

df3['Count', 'С стоп-словами'] = top_terms_stop_stem
df3['TF', 'С стоп-словами'] = top_terms_stem_stop_tf
df3['TF-IDF', 'С стоп-словами'] = top_terms_stem_stop_tfidf

df3
```

	Count		TF		TF-IDF	
	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
0	{'the': 20919}	{'wa': 2470}	{'the': 580.9195455198186}	{'kuvvetli': 54.741560260006736}	{'the': 226.73601702282315}	{'kuvvetli': 54.741560260006736}
1	{'of': 10253}	{'thi': 2422}	{'to': 317.3474366692875}	{'keysystem': 52.77569413890487}	{'to': 128.4318213943837}	{'keysystem': 52.77569413890487}
2	{'to': 9895}	{'armenian': 1123}	{'of': 278.6135135551277}	{'liu': 52.615200558633546}	{'of': 122.29885435480524}	{'liu': 52.615200558633546}
3	{'and': 9305}	{'use': 1113}	{'and': 240.78718078291644}	{'papa': 50.36062593469473}	{'and': 104.03434074458796}	{'papa': 50.36062593469473}
4	{'in': 6686}	{'peopl': 1062}	{'is': 210.61193877763262}	{'drain': 48.3356858391453}	{'is': 91.72437877615828}	{'drain': 48.3356858391453}
5	{'that': 5249}	{'ha': 1033}	{'it': 193.046367308862}	{'download': 43.35324301900809}	{'it': 88.41412066279622}	{'download': 43.35324301900809}
6	{'is': 5082}	{'ani': 923}	{'in': 181.44162075274158}	{'tour': 37.27135075539442}	{'in': 86.40389015137937}	{'tour': 37.27135075539442}
7	{'it': 4834}	{'know': 813}	{'that': 164.96081627832928}	{'tire': 36.292810556154315}	{'that': 79.38729165022512}	{'tire': 36.292810556154315}
8	{'you': 3309}	{'drive': 778}	{'you': 128.50829611456712}	{'joke': 35.5649078550382}	{'you': 74.64096646413044}	{'joke': 35.5649078550382}
9	{'for': 3249}	{'like': 763}	{'for': 113.48764322258968}	{'dunya': 32.30302778343974}	{'for': 57.82953546770416}	{'dunya': 32.30302778343974}
10	{'on': 2653}	{'time': 752}	{'have': 96.35224116826025}	{'eigen': 32.04577092336103}	{'have': 51.02324512463676}	{'eigen': 32.04577092336103}
11	{'with': 2582}	{'hi': 732}	{'thi': 92.78319885420794}	{'va': 31.273268013974864}	{'thi': 50.409120246347435}	{'va': 31.273268013974864}
12	{'not': 2505}	{'did': 702}	{'not': 88.07689480155399}	{'tile': 29.87072552182338}	{'not': 49.86390651191747}	{'tile': 29.87072552182338}
13	{'are': 2473}	{'onli': 684}	{'be': 86.95367114840924}	{'hada': 26.74248581100969}	{'are': 49.66775892441848}	{'hada': 26.74248581100969}
14	{'wa': 2470}	{'just': 678}	{'with': 86.61804035263049}	{'broccoli': 26.492309800567398}	{'be': 48.57961627372931}	{'broccoli': 26.492309800567398}
15	{'they': 2458}	{'say': 666}	{'on': 85.5882724969143}	{'powel': 26.022392993607035}	{'on': 47.70476608122967}	{'powel': 26.022392993607035}
16	{'have': 2439}	{'said': 617}	{'are': 84.31941150127194}	{'honeywell': 25.264808465903076}	{'with': 47.00092217462357}	{'honeywell': 25.264808465903076}
17	{'thi': 2422}	{'edu': 587}	{'do': 73.39506975724343}	{'dip': 24.7113547918968}	{'do': 44.72365073513757}	{'dip': 24.7113547918968}
18	{'be': 2385}	{'doe': 583}	{'as': 65.75951900628384}	{'ndw': 24.524137168129766}	{'as': 42.21231611823069}	{'ndw': 24.524137168129766}
19	{'as': 2060}	{'becaus': 568}	{'or': 64.80237842415754}	{'weyrich': 23.039159762462827}	{'my': 39.14384832245557}	{'weyrich': 23.039159762462827}

Рисунок 16 - Результат векторизации со стеммингом для обучающего множества

```

|: df4 = pd.DataFrame(columns=columns)

df4['Count', 'Без стоп-слов'] = top_terms_stem_test
df4['TF', 'Без стоп-слов'] = top_terms_stem_tf_test
df4['TF-IDF', 'Без стоп-слов'] = top_terms_stem_tfidf_test

df4['Count', 'С стоп-словами'] = top_terms_stop_stem_test
df4['TF', 'С стоп-словами'] = top_terms_stem_stop_tf_test
df4['TF-IDF', 'С стоп-словами'] = top_terms_stem_stop_tfidf_test

df4

```

	Count		TF		TF-IDF	
	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
0	{'the': 14650}	{'thi': 1654}	{'the': 393.8825021805069}	{'kuvvetli': 38.58159096500337}	{'the': 157.2388941502928}	{'kuvvetli': 38.58159096500337}
1	{'of': 7249}	{'wa': 1508}	{'to': 212.70394281586968}	{'liu': 36.12468490625079}	{'to': 88.25882757708708}	{'liu': 36.12468490625079}
2	{'to': 6580}	{'ha': 748}	{'of': 183.61602638204053}	{'keyssystem': 35.07900385428372}	{'of': 82.15071361679951}	{'keyssystem': 35.07900385428372}
3	{'and': 6307}	{'armenian': 706}	{'and': 165.45839194498524}	{'papa': 33.899805109818885}	{'and': 72.81946837666995}	{'papa': 33.899805109818885}
4	{'in': 4664}	{'use': 687}	{'is': 133.29227227576882}	{'drain': 33.22247165159427}	{'in': 59.67514134362165}	{'drain': 33.22247165159427}
5	{'is': 3244}	{'peopl': 577}	{'in': 122.3052103845949}	{'download': 31.356326600450455}	{'is': 58.74700675437686}	{'download': 31.356326600450455}
6	{'that': 3201}	{'ani': 551}	{'it': 109.05515745423625}	{'tire': 24.756486685752243}	{'that': 53.114529463275495}	{'tire': 24.756486685752243}
7	{'it': 2798}	{'like': 476}	{'that': 107.48290520963673}	{'va': 23.68612643837865}	{'it': 50.782768659461645}	{'va': 23.68612643837865}
8	{'for': 2343}	{'know': 452}	{'for': 81.34252982190382}	{'tile': 23.60052634582734}	{'you': 45.24027565870208}	{'tile': 23.60052634582734}
9	{'on': 1809}	{'hi': 418}	{'you': 75.6859615361589}	{'dunya': 21.770502304466586}	{'for': 42.32351737189909}	{'dunya': 21.770502304466586}
10	{'you': 1798}	{'time': 408}	{'have': 69.20996614485878}	{'vertic': 21.18233857137946}	{'have': 37.89706132683007}	{'vertic': 21.18233857137946}
11	{'with': 1691}	{'just': 407}	{'thi': 67.92376484399286}	{'hada': 19.81800160459357}	{'thi': 37.640352013093114}	{'hada': 19.81800160459357}
12	{'have': 1668}	{'doe': 390}	{'on': 57.93501770971535}	{'tour': 19.631632693113296}	{'on': 32.969598013545124}	{'tour': 19.631632693113296}
13	{'thi': 1654}	{'muslim': 376}	{'not': 55.61804308266621}	{'joke': 19.001475698647734}	{'are': 32.733953715289694}	{'joke': 19.001475698647734}
14	{'be': 1532}	{'onli': 371}	{'with': 54.98452705060883}	{'dip': 18.620097671734133}	{'not': 32.53959169338456}	{'dip': 18.620097671734133}
15	{'not': 1532}	{'did': 369}	{'are': 54.48411532873619}	{'powel': 18.270319216644406}	{'do': 32.37287087935601}	{'powel': 18.270319216644406}
16	{'as': 1531}	{'said': 366}	{'be': 54.28869092725317}	{'nan': 17.341536187667092}	{'be': 31.33777132043142}	{'nan': 17.341536187667092}
17	{'are': 1511}	{'year': 360}	{'do': 51.98500680932219}	{'broccoli': 16.80690032882596}	{'with': 30.746733916646296}	{'broccoli': 16.80690032882596}
18	{'wa': 1508}	{'work': 342}	{'as': 44.83158523890473}	{'honeywell': 16.734896985500633}	{'as': 29.68626987221284}	{'honeywell': 16.734896985500633}
19	{'by': 1338}	{'new': 330}	{'or': 44.795997941256026}	{'weyrich': 16.175907790854883}	{'or': 27.490714607187865}	{'weyrich': 16.175907790854883}

Рисунок 17 - Результат векторизации со стеммингом для тестового множества

Используя конвейер (Pipeline) реализовать модель Наивного Байесовского классификатора и выявить на основе показателей качества (значения полноты, точности, f1-меры и аккуратности), какая предварительная обработка данных обеспечит наилучшие результаты классификации. Это показано на рисунке 18.


```

from sklearn.model_selection import GridSearchCV

gscv = GridSearchCV(text_clf, param_grid=parameters)
gscv.fit(twenty_train_full.data, twenty_train_full.target)

```



```

print(classification_report(gscv.predict(twenty_test_full.data), twenty_test_full.target))

```

	precision	recall	f1-score	support
0	0.97	0.94	0.96	404
1	0.92	0.92	0.92	393
2	0.93	0.95	0.94	367
accuracy			0.94	1164
macro avg	0.94	0.94	0.94	1164
weighted avg	0.94	0.94	0.94	1164

```

gscv.best_params_

```

```

{'tfidf__use_idf': True,
 'vect__max_features': 5000,
 'vect__stop_words': 'english'}

```

Рисунок 18 - Результат классификации

Вывод: В результате выполнения работы получены практические навыки обработки текстовых данных в среде Jupiter Notebook. Проведена предварительная обработка текстовых данных и выявлены параметры обработки, позволяющие добиться наилучшей точности классификации.