**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 Косенков В.Д.

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 значеним max_features = 10000

b. Вывести и проанализировать первые 20 наиболее частотных слов всей выборки и каждого класса по-отдельности. c. Применить процедуру отсечения 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 9

| | |
|---|---|
| 9 | 6, 17, 19 |

Рисунок 1 - Вариант для выполнения

| | |
|---|---|
| 6 | 'comp.windows.x' |
| 17 | 'talk.politics.guns' |
| 19 | 'talk.politics.misc' |

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.windows.x', 'talk.politics.guns', 'talk.politics.misc']
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.

```
print(twenty_train_full.data[0])
print(twenty_train_full.data[2])
```

```
That's open for debate.  Certainly, an excessive number of people are
murdered every year but people also do save innocent lives with firearms.
The media just don't tell us when it happens...


I think there are more of us than there are federal marshalls...


Crap.  It's simplistic thinking on the part of feather-headed dolts.


Nuts.
Hi, looking for any advice or suggestions about a problem I'm
having with MIT X11R5's editres, in particular under twm variants.

For a start, 9 times out of 10 (but NOT always) editres won't
grab a widget tree when running on our NCD (Decwindows) Xterms,
which I'm told will be fixed when the R5 (not R4) XDm is installed.
OK, so I tried running it on a Sun, running real R5, on the same
network - I get a widget tree, but it's ALWAYS for 'TWM Icon Manager'

Anybody know of any patches for (a) twm or (b) editres that I should
look at?
```

Рисунок 4 - Вывод данных

Далее необходимо применить стэмминг, это показано на рисунке 5

```
import nltk
from nltk import word_tokenize
from nltk.stem import *

nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
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

stem_train = stemming(twenty_train_full.data)
stem_test = stemming(twenty_test_full.data)
```

```
print(stem_train[0])
```

```
 that 's open for debat . certainli , an excess number of peopl are murder everi year but peopl also do save innoc liv
```

Рисунок 5 - Данные после стеммнга

Векторизация представлена на рисунке 6.

```
# Провести векторизацию выборки:
# a. Векторизовать обучающую и тестовую выборки простым подсчетом слов (CountVectorizer) и значеним max_features = 10000
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': 11036},
 {'to': 5842},
 {'of': 4550},
 {'and': 4361},
 {'that': 3429},
 {'in': 3060},
 {'is': 2713},
 {'it': 2384},
 {'you': 2335},
 {'for': 1897},
 {'on': 1601},
 {'have': 1545},
 {'this': 1527},
 {'be': 1376},
 {'with': 1336},
 {'not': 1325},
 {'are': 1249},
 {'they': 1214},
 {'or': 1203},
 {'was': 1129}]
```

Рисунок 6 - Векторизация простым подсчетом слов

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

```python
#c. Применить процедуру отсечения стоп-слов и повторить пункт b.
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': 568},
 {'dos': 519},
 {'don': 493},
 {'just': 441},
 {'use': 420},
 {'think': 412},
 {'like': 401},
 {'know': 384},
 {'windows': 323},
 {'time': 311},
 {'did': 310},
 {'right': 298},
 {'fbi': 283},
 {'president': 283},
 {'ms': 282},
 {'make': 253},
 {'does': 251},
 {'government': 250},
 {'way': 240},
 {'window': 235}]
```

Рисунок 7 - Векторизация с отсечением стоп-слов

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

```python
#d. Провести пункты a – c для обучающей и тестовой выборки, для которой проведена процедура стемминга.
vect_stem_without_stop = CountVectorizer(max_features=10000)
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)
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
```

```
[{'the': 11034},
 {'to': 5846},
 {'of': 4553},
 {'and': 4361},
 {'that': 3437},
 {'in': 3060},
 {'is': 2799},
 {'it': 2525},
 {'you': 2334},
 {'for': 1897},
 {'have': 1683},
 {'on': 1611},
 {'be': 1563},
 {'thi': 1525},
 {'not': 1378},
 {'with': 1336},
 {'do': 1331},
 {'are': 1279},
 {'they': 1214},
 {'or': 1203}]
```

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 a – c для обучающей и тестовой выборки, для которой проведена процедура стемминга. Применена процедура отсечения 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': 1525},
 {'wa': 1172},
 {'use': 849},
 {'ani': 592},
 {'peopl': 565},
 {'ha': 491},
 {'like': 458},
 {'window': 446},
 {'just': 441},
 {'think': 441},
 {'did': 423},
 {'know': 415},
 {'right': 398},
 {'doe': 381},
 {'make': 357},
 {'time': 349},
 {'onli': 340},
 {'gun': 337},
 {'say': 337},
 {'hi': 333}]
```

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

```
#Векторизовать выборки с помощью TfidfTransformer (с использованием TF и TF-IDF взвешиваний) и повторить пункты b-d.
#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': 148.07631709130683},
 {'to': 82.70848721798517},
 {'of': 70.48220550839375},
 {'and': 62.40203390038185},
 {'that': 54.7764440210131},
 {'in': 50.998032235663516},
 {'you': 48.79497272110181},
 {'is': 48.45276044652336},
 {'it': 47.26375191577538},
 {'for': 35.22744608190951},
 {'have': 33.37079986318576},
 {'they': 33.110364509733145},
 {'on': 30.8959806334108},
 {'not': 30.52023179802792},
 {'this': 29.74057211015482},
 {'was': 29.029245252121708},
 {'be': 28.81839868790516},
 {'with': 28.197377242777097},
 {'are': 27.128129436958407},
 {'if': 25.884267669683894}]
```

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
```

```
[{'people': 21.082433558261677},
 {'just': 19.465792714334412},
 {'fbi': 19.232888123652298},
 {'like': 17.61993064491381},
 {'don': 17.14700932489806},
 {'batf': 16.46455321268367},
 {'use': 14.909986985590248},
 {'koresh': 14.664094230583716},
 {'know': 14.601175000406249},
 {'did': 13.685460903563849},
 {'does': 13.411156177328131},
 {'time': 12.561706106286989},
 {'government': 12.429574561297667},
 {'ve': 12.098103949366822},
 {'right': 12.025011473347078},
 {'think': 11.88366349515811},
 {'way': 11.114111833009403},
 {'window': 11.061996678352118},
 {'children': 10.814574468054},
 {'thanks': 10.751234221907023}]
```

Рисунок 11 - Со стоп-словами, без стэмминга

```python
#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': 147.32608070216364},
 {'to': 82.73856117374731},
 {'of': 70.27288730542155},
 {'and': 62.20946463398296},
 {'that': 54.630659205752615},
 {'in': 50.93562121540392},
 {'is': 49.64690244725028},
 {'it': 49.435002391303875},
 {'you': 48.51163328044606},
 {'for': 35.494676702574374},
 {'have': 35.16746976491118},
 {'they': 33.06231121233108},
 {'not': 31.27708259587158},
 {'be': 31.23476761837217},
 {'on': 31.03124356922075},
 {'thi': 29.65721649482315},
 {'wa': 29.425896497136236},
 {'do': 28.290251088825766},
 {'with': 28.16768993310699},
 {'are': 27.610026608728543}]
```

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_tf, 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_tf, 20)]
top_terms_stem_stop_tfidf_test
```

```
[{'pension': 40.45022264099219},
 {'knee': 37.57367902098388},
 {'longjmp': 34.36356412662803},
 {'document': 33.904900378786536},
 {'lauderdal': 27.46517225549892},
 {'w4600': 27.109263298922723},
 {'divert': 22.845058965845674},
 {'femal': 22.540728977965426},
 {'ugli': 21.700659297867233},
 {'desire': 21.416946847554655},
 {'tweak': 20.43465337166575},
 {'salmon': 19.096893896431272},
 {'went': 18.88917125560885},
 {'gorman': 18.7792199367161},
 {'wwii': 18.35488189567649},
 {'ardent': 17.69659395191857},
 {'matrix': 16.723642956579642},
 {'tt': 15.816477455062518},
 {'xd': 15.559002011693183},
 {'wait': 15.47609091054902}]
```

Рисунок 13 - Со стоп-словами и стэммингом

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

```
#По результатам пункта 6 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': 21030} | {'people': 964} | {'the': 554.8486882073121} | {'people': 59.954810056238806} | {'the': 212.43590464752484} | {'people': 30.6842945150969} |
| 1 | {'to': 11046} | {'file': 886} | {'to': 320.6295458643513} | {'don': 52.730680729634905} | {'to': 128.45957050671944} | {'don': 26.600320939693077} |
| 2 | {'of': 8892} | {'don': 744} | {'of': 227.01295130027935} | {'like': 48.644735710709014} | {'of': 101.40352339219568} | {'know': 24.877827496037572} |
| 3 | {'and': 7733} | {'use': 723} | {'and': 203.74577183479929} | {'just': 46.612076767195205} | {'and': 89.58591707610924} | {'like': 24.816508195950927} |
| 4 | {'in': 5967} | {'like': 656} | {'that': 167.11700800428042} | {'know': 46.31646804907305} | {'that': 80.29494493702127} | {'window': 24.22002401579132} |
| 5 | {'that': 5745} | {'know': 628} | {'in': 166.49189038535724} | {'use': 40.4208368247289} | {'is': 77.95576761817283} | {'just': 23.828546673415435} |
| 6 | {'is': 5640} | {'gun': 621} | {'is': 161.46825933663712} | {'think': 35.64703709817017} | {'in': 75.7166871470798} | {'use': 21.945655574215436} |
| 7 | {'it': 4248} | {'think': 618} | {'it': 137.75030887096042} | {'window': 34.42661997251195} | {'you': 69.86064206644143} | {'gun': 21.168438506443707} |
| 8 | {'for': 3630} | {'just': 608} | {'you': 126.8038927413327} | {'does': 32.754331362948605} | {'it': 67.41391344496809} | {'think': 20.49895681314723} |
| 9 | {'you': 3614} | {'window': 575} | {'for': 109.0379847293877} | {'gun': 31.36664245047985} | {'for': 54.62324514583435} | {'does': 18.643404575102984} |
| 10 | {'this': 3047} | {'time': 544} | {'this': 100.1614734581991} | {'time': 28.96690400166267} | {'this': 51.71255809911795} | {'government': 18.436302594536873} |
| 11 | {'on': 2910} | {'mr': 529} | {'have': 82.80632496342685} | {'government': 28.018425347358047} | {'have': 45.75492102396245} | {'thanks': 17.136615956390376} |
| 12 | {'be': 2610} | {'program': 529} | {'on': 82.43063908732124} | {'make': 27.03180438299153} | {'on': 45.41746696978206} | {'time': 16.62842521655699} |
| 13 | {'are': 2540} | {'make': 493} | {'be': 75.00069756092036} | {'right': 26.451686939941894} | {'are': 43.98716531332075} | {'right': 16.3923721963806} |
| 14 | {'have': 2415} | {'edu': 484} | {'are': 72.05937975933722} | {'want': 25.33552524802435} | {'be': 43.28092066536194} | {'server': 16.037290681248695} |
| 15 | {'not': 2353} | {'president': 468} | {'not': 71.42901495874035} | {'thanks': 25.25438852527886} | {'not': 42.21359480091894} | {'ve': 15.754844733146905} |
| 16 | {'with': 2243} | {'does': 465} | {'with': 69.08350319338527} | {'ve': 25.01318781350344} | {'as': 40.407586088249836} | {'make': 15.740554707625035} |
| 17 | {'as': 2143} | {'government': 456} | {'if': 64.12971112293212} | {'did': 24.695374506397958} | {'with': 39.13631314385873} | {'want': 15.701202983786704} |
| 18 | {'or': 2055} | {'new': 444} | {'as': 63.67591081632106} | {'way': 24.256210914247443} | {'they': 38.69301414688328} | {'did': 15.516022196706396} |
| 19 | {'if': 1918} | {'right': 414} | {'or': 57.09807727880445} | {'using': 23.016362738923053} | {'if': 37.865469263042854} | {'motif': 15.025153071976474} |

Рисунок 14 - Результат векторизации для обучающего множества без стэмминга

```
df2 = pd.DataFrame(columns=columns)
df2['Count', 'С стоп-словами'] = top_terms_stop
df2['TF', 'С стоп-словами'] = top_terms_stop_tf
df2['TF-IDF', 'С стоп-словами'] = top_terms_stop_tfidf
df2
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': 11036} | {'people': 568} | {'the': 377.09973253951495} | {'people': 40.45022264099219} | {'the': 148.07631709130683} | {'people': 21.082433558261677} |
| 1 | {'to': 5842} | {'dos': 519} | {'to': 200.0280686519804} | {'just': 37.57367902098388} | {'to': 82.70848721798517} | {'just': 19.465792714334412} |
| 2 | {'of': 4550} | {'don': 493} | {'of': 156.31003721479456} | {'like': 34.36356412662803} | {'of': 70.48220550839375} | {'fbi': 19.232888123652298} |
| 3 | {'and': 4361} | {'just': 441} | {'and': 138.19940302326603} | {'don': 33.904900378786536} | {'and': 62.40203390038185} | {'like': 17.61993064491381} |
| 4 | {'that': 3429} | {'use': 420} | {'that': 110.09981347634826} | {'know': 27.46517225549892} | {'that': 54.7764440210131} | {'don': 17.14700932489806} |
| 5 | {'in': 3060} | {'think': 412} | {'in': 108.85332695743035} | {'use': 27.109263298922723} | {'in': 50.998032235663516} | {'batf': 16.46455321268367} |
| 6 | {'is': 2713} | {'like': 401} | {'is': 99.21932095496899} | {'does': 22.845058965845674} | {'you': 48.79497272110181} | {'use': 14.909986985590248} |
| 7 | {'it': 2384} | {'know': 384} | {'it': 95.63620519187951} | {'fbi': 22.540728977965426} | {'is': 48.45276044652336} | {'koresh': 14.664094230583716} |
| 8 | {'you': 2335} | {'windows': 323} | {'you': 84.0372118833688} | {'time': 21.700659297867233} | {'it': 47.26375191577538} | {'know': 14.601175000406249} |
| 9 | {'for': 1897} | {'time': 311} | {'for': 69.79735051531073} | {'did': 21.416946847554655} | {'for': 35.22744608190951} | {'did': 13.685460903563849} |
| 10 | {'on': 1601} | {'did': 310} | {'have': 58.09560331590626} | {'think': 20.43465337166575} | {'have': 33.37079986318576} | {'does': 13.411156177328131} |
| 11 | {'have': 1545} | {'right': 298} | {'this': 55.73417829243327} | {'right': 19.096893896431272} | {'they': 33.110364509733145} | {'time': 12.561706106286989} |
| 12 | {'this': 1527} | {'fbi': 283} | {'on': 54.85358390317564} | {'ve': 18.88917125560885} | {'on': 30.8959806334108} | {'government': 12.429574561297667} |
| 13 | {'be': 1376} | {'president': 283} | {'not': 50.47592153986833} | {'government': 18.7792199367161} | {'not': 30.52023179802792} | {'ve': 12.098103949366822} |
| 14 | {'with': 1336} | {'ms': 282} | {'be': 48.63535056136824} | {'way': 18.35488189567649} | {'this': 29.74057211015482} | {'right': 12.025011473347078} |
| 15 | {'not': 1325} | {'make': 253} | {'with': 47.903401201604986} | {'batf': 17.69659395191857} | {'was': 29.029245252121708} | {'think': 11.88366349515811} |
| 16 | {'are': 1249} | {'does': 251} | {'they': 45.79934276771499} | {'make': 16.723642956579642} | {'be': 28.81839868790516} | {'way': 11.114111833009403} |
| 17 | {'they': 1214} | {'government': 250} | {'are': 43.21743530714192} | {'thanks': 15.816477455062518} | {'with': 28.197377242777097} | {'window': 11.061996678352118} |
| 18 | {'or': 1203} | {'way': 240} | {'if': 42.796996090000526} | {'window': 15.559002011693183} | {'are': 27.128129436958407} | {'children': 10.814574468054} |
| 19 | {'was': 1129} | {'window': 235} | {'or': 39.11863440523408} | {'using': 15.47609091054902} | {'if': 25.884267669683894} | {'thanks': 10.751234221907023} |

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': 21029} | {'thi': 3049} | {'the': 543.4391576124485} | {'pension': 59.954810056238806} | {'the': 212.44434755812273} | {'pension': 59.954810056238806} |
| 1 | {'to': 11046} | {'wa': 1655} | {'to': 313.9775917503421} | {'document': 52.730680729634905} | {'to': 128.91889779912216} | {'document': 52.730680729634905} |
| 2 | {'of': 8893} | {'use': 1557} | {'of': 222.29449330038193} | {'longjmp': 48.644735710709014} | {'of': 101.40014571739448} | {'longjmp': 48.644735710709014} |
| 3 | {'and': 7733} | {'file': 1117} | {'and': 199.4210767460697} | {'knee': 46.612076767195205} | {'and': 89.70977919139284} | {'knee': 46.612076767195205} |
| 4 | {'in': 5968} | {'ha': 1072} | {'that': 164.0862355077531} | {'lauderdal': 46.31646804907305} | {'that': 80.57436419379303} | {'lauderdal': 46.31646804907305} |
| 5 | {'is': 5770} | {'peopl': 965} | {'is': 163.17509947069564} | {'w4600': 40.4208368247289} | {'is': 79.82800532872957} | {'w4600': 40.4208368247289} |
| 6 | {'that': 5752} | {'ani': 958} | {'in': 163.007196640662} | {'tweak': 35.64703709817017} | {'in': 75.73900608406692} | {'tweak': 35.64703709817017} |
| 7 | {'it': 4563} | {'gun': 915} | {'it': 142.4691091418549} | {'xd': 34.42661997251195} | {'it': 70.6382044986766} | {'xd': 34.42661997251195} |
| 8 | {'for': 3630} | {'like': 783} | {'you': 124.23431385799317} | {'divert': 32.754331362948605} | {'you': 70.0362143497004} | {'divert': 32.754331362948605} |
| 9 | {'you': 3614} | {'program': 776} | {'for': 106.7839186917083} | {'growth': 31.36664245047985} | {'for': 54.788536904724005} | {'growth': 31.36664245047985} |
| 10 | {'thi': 3049} | {'window': 730} | {'thi': 98.20886997750844} | {'ugli': 28.96690400166267} | {'thi': 52.05098221131972} | {'ugli': 28.96690400166267} |
| 11 | {'be': 2952} | {'make': 698} | {'have': 90.15590969764725} | {'gorman': 28.018425347358047} | {'have': 49.243705984344274} | {'gorman': 28.018425347358047} |
| 12 | {'on': 2914} | {'work': 689} | {'be': 84.78052592768354} | {'matrix': 27.03180438299153} | {'be': 48.076159712311764} | {'matrix': 27.03180438299153} |
| 13 | {'have': 2649} | {'doe': 687} | {'on': 80.72356621473094} | {'salmon': 26.451686939941894} | {'on': 45.513621198504794} | {'salmon': 26.451686939941894} |
| 14 | {'are': 2585} | {'know': 677} | {'do': 74.44876881509532} | {'workspac': 25.33552524802435} | {'are': 44.773164550672526} | {'workspac': 25.33552524802435} |
| 15 | {'not': 2465} | {'think': 666} | {'not': 73.4057864614913} | {'tt': 25.25438852527886} | {'do': 44.07691134403636} | {'tt': 25.25438852527886} |
| 16 | {'with': 2243} | {'time': 663} | {'are': 72.28665508614623} | {'went': 25.01318781350344} | {'not': 43.62023522370044} | {'went': 25.01318781350344} |
| 17 | {'as': 2143} | {'state': 650} | {'with': 67.55610424812832} | {'desire': 24.695374506397958} | {'as': 40.46771391082354} | {'desire': 24.695374506397958} |
| 18 | {'do': 2112} | {'just': 608} | {'if': 62.78011153289489} | {'wwii': 24.256210914247443} | {'with': 39.1237129865674} | {'wwii': 24.256210914247443} |
| 19 | {'or': 2055} | {'right': 596} | {'as': 62.29332997846665} | {'wait': 23.016362738923053} | {'they': 39.0112618405604} | {'wait': 23.016362738923053} |

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': 11034} | {'thi': 1525} | {'the': 368.8710276776722} | {'pension': 40.45022264099219} | {'the': 147.32608070216364} | {'pension': 40.45022264099219} |
| 1 | {'to': 5846} | {'wa': 1172} | {'to': 195.67444595801882} | {'knee': 37.57367902098388} | {'to': 82.73856117374731} | {'knee': 37.57367902098388} |
| 2 | {'of': 4553} | {'use': 849} | {'of': 153.03809722795222} | {'longjmp': 34.36356412662803} | {'of': 70.27288730542155} | {'longjmp': 34.36356412662803} |
| 3 | {'and': 4361} | {'ani': 592} | {'and': 135.16833760921565} | {'document': 33.904900378786536} | {'and': 62.20946463398296} | {'document': 33.904900378786536} |
| 4 | {'that': 3437} | {'peopl': 565} | {'that': 107.76309375323476} | {'lauderdal': 27.46517225549892} | {'that': 54.630659205752615} | {'lauderdal': 27.46517225549892} |
| 5 | {'in': 3060} | {'ha': 491} | {'in': 106.45015708487696} | {'w4600': 27.109263298922723} | {'in': 50.93562121540392} | {'w4600': 27.109263298922723} |
| 6 | {'is': 2799} | {'like': 458} | {'is': 100.18834034160336} | {'divert': 22.845058965845674} | {'is': 49.64690244725028} | {'divert': 22.845058965845674} |
| 7 | {'it': 2525} | {'window': 446} | {'it': 98.89929676688507} | {'femal': 22.540728977965426} | {'it': 49.435002391303875} | {'femal': 22.540728977965426} |
| 8 | {'you': 2334} | {'just': 441} | {'you': 82.08308886067267} | {'ugli': 21.700659297867233} | {'you': 48.51163328044606} | {'ugli': 21.700659297867233} |
| 9 | {'for': 1897} | {'think': 441} | {'for': 68.41689425305424} | {'desire': 21.416946847554655} | {'for': 35.494676702574374} | {'desire': 21.416946847554655} |
| 10 | {'have': 1683} | {'did': 423} | {'have': 61.99383823109575} | {'tweak': 20.43465337166575} | {'have': 35.16746976491118} | {'tweak': 20.43465337166575} |
| 11 | {'on': 1611} | {'know': 415} | {'thi': 54.40486912258514} | {'salmon': 19.096893896431272} | {'they': 33.06231121233108} | {'salmon': 19.096893896431272} |
| 12 | {'be': 1563} | {'right': 398} | {'on': 53.87111601989148} | {'went': 18.88917125560885} | {'not': 31.27708259587158} | {'went': 18.88917125560885} |
| 13 | {'thi': 1525} | {'doe': 381} | {'be': 53.66740959075927} | {'gorman': 18.7792199367161} | {'be': 31.23476761837217} | {'gorman': 18.7792199367161} |
| 14 | {'not': 1378} | {'make': 357} | {'not': 51.56147583842764} | {'wwii': 18.35488189567649} | {'on': 31.03124356922075} | {'wwii': 18.35488189567649} |
| 15 | {'with': 1336} | {'time': 349} | {'with': 46.85014370574915} | {'ardent': 17.69659395191857} | {'thi': 29.65721649482315} | {'ardent': 17.69659395191857} |
| 16 | {'do': 1331} | {'onli': 340} | {'do': 45.68521636706058} | {'matrix': 16.723642956579642} | {'wa': 29.425896497136236} | {'matrix': 16.723642956579642} |
| 17 | {'are': 1279} | {'gun': 337} | {'they': 44.86864445476717} | {'tt': 15.816477455062518} | {'do': 28.290251088825766} | {'tt': 15.816477455062518} |
| 18 | {'they': 1214} | {'say': 337} | {'are': 43.45312116545678} | {'xd': 15.559002011693183} | {'with': 28.16768993310699} | {'xd': 15.559002011693183} |
| 19 | {'or': 1203} | {'hi': 333} | {'if': 41.788209579526914} | {'wait': 15.47609091054902} | {'are': 27.610026608728543} | {'wait': 15.47609091054902} |

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.

```
[32] from sklearn.pipeline import Pipeline

     parameters = {
         'vect__max_features': max_features_values,
         'vect__stop_words': stop_words,
         'tfidf__use_idf': use_idf
     }

     text_clf = Pipeline([('vect', CountVectorizer()),
                          ('tfidf', TfidfTransformer()),
                          ('clf', MultinomialNB())])
     from sklearn.model_selection import GridSearchCV

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

```
GridSearchCV
GridSearchCV(estimator=Pipeline(steps=[('vect', CountVectorizer()),
                                       ('tfidf', TfidfTransformer()),
                                       ('clf', MultinomialNB())]),
             param_grid={'tfidf__use_idf': [True, False],
                         'vect__max_features': [100, 500, 1000, 2000, 3000,
                                                4000, 5000],
                         'vect__stop_words': [None, 'english']})
estimator: Pipeline
Pipeline(steps=[('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                ('clf', MultinomialNB())])
CountVectorizer
CountVectorizer()
TfidfTransformer
TfidfTransformer()
MultinomialNB
MultinomialNB()
```

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

```
              precision    recall  f1-score   support

           0       0.97      0.91      0.94       420
           1       0.88      0.69      0.78       466
           2       0.50      0.85      0.63       183

    accuracy                           0.80      1069
   macro avg       0.78      0.82      0.78      1069
weighted avg       0.85      0.80      0.82      1069
```

```
[34] gscv.best_params_
```

```
{'tfidf__use_idf': True,
 'vect__max_features': 4000,
 '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.