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 Коровайцев А.А.

Группа М-ИАП-23-1

Руководитель Кургасов В.В.

Доцент

Липецк 2023 г.

Цель работы

Получить практические навыки обработки текстовых данных в среде Jupiter Notebook. Научиться проводить предварительную обработку текстовых данных и выявлять параметры обработки, позволяющие добиться наилучшей точности классификации.

Задание кафедры

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.

Без стемминга						
	Count		TF		TF-IDF	
№	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
1						
2						
…						
20						

Со стеммингом						
	Count		TF		TF-IDF	
№	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
1						
2						
…						
20						

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).

Вариант №7

Классы 3, 7, 13 ('comp.os.ms-windows.misc', 'misc.forsale', 'sci.electronics'

Ход работы

Загрузим обучающую и экзаменационную выборку в соответствии с вариантом. Код для загрузки данных представлен на рисунке 1.

```
categories = ['comp.os.ms-windows.misc', 'misc.forsale', 'sci.electronics']
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 1 – Код для загрузки данных

После успешной загрузки данных посмотрим на записи. Для этого выведем по одному значению из обучаемой и тестовой выборки, которые представлены на рисунке 2.

```
twenty_train_full.data[0]
```

```
'\nThere\'s a program called "Icon Frightener" included with the book Stupid\nWindows Tricks by Bob LeVitus and Ed Tittel (Addi
son-Wesley, 1992).  It\'s\nfreeware.  If it\'s not on the net anywhere, I\'ll happily email a copy to\nsomeone who\'s willing t
o upload it (I can\'t upload through our Internet\nfirewall).\n'
```

```
twenty_test_full.data[0]
```

```
'After hearing endless debate (READ: name-calling) over which os is better, dos\nand windows or OS/2 and finally having enought
resourses to play with a couple\nof different operating systems, I have decided to put the two products to a\nhead to head tes
t, as so many fellow newsposters have suggested.  I have, \nhowever, no desire whatsoever to use a version of os/2 which wont R
EALLY\ndo what it says (i.e. run windows apps)  OS/2 2.0-2.1 will not run windows\napps in 386 enhansed mode, something that mo
st larger windows apps require, but\nOS/2 2.2, which is supposed to be in beta test, is supposed to.  I have heard\nthat os/2
2.2 beta is available via ftp, and I was wondering if anyone knew\nwhere to obtain a copy.  I would appreciate any information,
as I would like, \nonce and for all, to establish for myself which is the best os for my needs.'
```

Рисунок 2 – Пример загруженных данных

Применим стемминг к исходным данным в соответствии с кодом и посмотрим на обработанные данные, которые представлены на рисунке 3.

```
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)
```

```
stem_train[0]
```

```
" there 's a program call `` icon frighten '' includ with the book stupid window trick by bob levitu and ed tittel ( addison-we
sley , 1992 ) . it' freewar . if it 's not on the net anywher , i 'll happili email a copi to someon who 's will to upload it (
i ca n't upload through our internet firewal ) ."
```

```
stem_test[0]
```

```
' after hear endless debat ( read : name-cal ) over which os is better , do and window or os/2 and final have enought resours t
o play with a coupl of differ oper system , i have decid to put the two product to a head to head test , as so mani fellow news
post have suggest . i have , howev , no desir whatsoev to use a version of os/2 which wont realli do what it say ( i.e . run wi
ndow app ) os/2 2.0-2.1 will not run window app in 386 enhans mode , someth that most larger window app requir , but os/2 2.2 ,
which is suppos to be in beta test , is suppos to . i have heard that os/2 2.2 beta is avail via ftp , and i wa wonder if anyon
knew where to obtain a copi . i would appreci ani inform , as i would like , onc and for all , to establish for myself which is
the best os for my need .'
```

Рисунок 3 – Данные, обработанные стеммингом

Проведем векторизацию выборки. Для этого векторизуем обучающую и тестовую выборку простым подсчетом слов с использованием класса CountVectorizer и значением max_features = 10000, код для выполнения данного способа представлен на рисунке 4. Выведем первые 20 наиболее частотных слов по всей выборки и отобразим на рисунке 5.

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

```
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]
```

Рисунок 4 – Код для векторизации обучающей и тестовой выборки простым подсчетом слов

```
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': 5303},
 {'to': 3088},
 {'and': 2434},
 {'of': 2139},
 {'is': 1796},
 {'for': 1707},
 {'it': 1636},
 {'in': 1527},
 {'that': 1281},
 {'you': 1264},
 {'have': 1009},
 {'with': 921},
 {'this': 869},
 {'on': 860},
 {'or': 818},
 {'if': 783},
 {'are': 766},
 {'be': 719},
 {'but': 618},
 {'not': 616}]
```

Рисунок 5 – Результат векторизации обучающей и тестовой выборки простым подсчетом слов

Применим процедуру отсечения стоп-слов и повторим вывод полученных результатов. Код для обработки данных путем отсечения стоп- 7 слов представлен на рисунке 6. Результат векторизации обучающей и тестовой выборки простым подсчетом слов с отсечением стоп-слов представлен на рисунке 7.

```
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)
```

Рисунок 6 – Код для векторизации обучающей и тестовой выборки простым подсчетом слов с отсечением стоп-слов

```
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': 5301},
 {'to': 3088},
 {'and': 2435},
 {'of': 2139},
 {'is': 1838},
 {'it': 1711},
 {'for': 1707},
 {'in': 1528},
 {'that': 1284},
 {'you': 1264},
 {'have': 1066},
 {'with': 921},
 {'do': 913},
 {'thi': 869},
 {'on': 863},
 {'or': 818},
 {'be': 791},
 {'if': 783},
 {'are': 781},
 {'use': 734}]
```

Рисунок 7 – Результат векторизации обучающей и тестовой выборки простым подсчетом слов с отсечением стоп-слов

Также проведем аналогичный анализ для данных после стемминга. Результат векторизации обучающей и тестовой выборки после стемминга простым подсчетом слов без отсечения стоп-слов представлен на рисунке 8. Результат векторизации обучающей и тестовой выборки после стемминга простым подсчетом слов с отсечением стоп-слов представлен на рисунке 9.

```
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': 5301},
 {'to': 3088},
 {'and': 2435},
 {'of': 2139},
 {'is': 1838},
 {'it': 1711},
 {'for': 1707},
 {'in': 1528},
 {'that': 1284},
 {'you': 1264},
 {'have': 1066},
 {'with': 921},
 {'do': 913},
 {'thi': 869},
 {'on': 863},
 {'or': 818},
 {'be': 791},
 {'if': 783},
 {'are': 781},
 {'use': 734}]
```

Рисунок 8 – Результат векторизации обучающей и тестовой выборки после стемминга простым подсчетом слов без 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': 869},
 {'use': 734},
 {'00': 583},
 {'window': 564},
 {'file': 380},
 {'work': 361},
 {'ani': 336},
 {'ha': 326},
 {'wa': 326},
 {'like': 325},
 {'know': 289},
 {'new': 278},
 {'drive': 267},
 {'just': 261},
 {'10': 256},
 {'run': 254},
 {'doe': 250},
 {'pleas': 236},
 {'program': 236},
 {'make': 235}]
```

Рисунок 9 – Результат векторизации обучающей и тестовой выборки после стемминга простым подсчетом слов с отсечением стоп-слов

Воспользуемся векторизацией выборки с помощью TfidfTransformer (с использованием TF и TF-IDF взвешиваний). Векторизация выборки с использованием TfidfTransformer для набора данных без использования стопслов представлен на рисунке 10, с использованием стоп-слов представлен на рисунке 11.

```
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': 110.54034814082188},
 {'to': 74.53330706423196},
 {'and': 59.49702901688886},
 {'of': 55.00612226795159},
 {'it': 51.437481151335135},
 {'for': 51.171080962654656},
 {'is': 50.718071500871154},
 {'that': 44.362737722486365},
 {'you': 42.734546090195195},
 {'in': 42.50992737276512},
 {'have': 35.42627259647494},
 {'or': 30.117888661918553},
 {'on': 30.0179938849296},
 {'with': 28.318691298767323},
 {'if': 28.21545774406861},
 {'this': 27.8614040902488},
 {'are': 26.408399492923788},
 {'can': 25.72077920823754},
 {'windows': 25.587587452136255},
 {'be': 25.386806878300593}]
```

Рисунок 10 – Результат векторизации набора данных без использования 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
```

```
[{'windows': 29.525766988247245},
 {'know': 19.401071908073977},
 {'dos': 18.328724185534426},
 {'like': 18.22825166865843},
 {'edu': 17.720422089543977},
 {'just': 17.59219450994192},
 {'don': 16.953572988898507},
 {'thanks': 16.89266199027737},
 {'use': 16.745586293513156},
 {'does': 14.87062950349624},
 {'sale': 14.848851201411348},
 {'00': 14.63805388598326},
 {'new': 14.253487477230415},
 {'drive': 14.107584262411871},
 {'os': 14.072038969746567},
 {'program': 14.02943548800833},
 {'mail': 13.843080897295174},
 {'file': 13.606763403975274},
 {'think': 13.491354914405813},
 {'make': 13.380164760732074}]
```

Рисунок 11 – Результат векторизации набора данных с использованием стоп-слов

Проведем аналогичную векторизацию для набора данных после стемминга. Результат векторизации набора данных после стемминга без использования стоп-слов представлен на рисунке 12, с использованием стопслов представлен на рисунке 13.

```
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': 110.1500725968721},
 {'to': 74.6970778664183},
 {'and': 59.452672256420016},
 {'of': 54.694210335382635},
 {'it': 52.94695312834682},
 {'is': 51.959038586981414},
 {'for': 51.36681718395666},
 {'that': 44.33877517975381},
 {'you': 42.705239546726474},
 {'in': 42.419008264604095},
 {'have': 36.75793858768384},
 {'do': 34.455101685751444},
 {'on': 30.18515384300593},
 {'or': 30.111057936497932},
 {'if': 28.29711228464798},
 {'with': 28.247127641386854},
 {'thi': 27.88182171437507},
 {'window': 27.20067077113115},
 {'be': 27.19403159253378},
 {'are': 26.861145726964967}]
```

Рисунок 12 – Результат векторизации набора данных после стемминга без использования 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
```

```
[{'wmbxlt': 52.138731148601934},
 {'k9': 34.439374615801476},
 {'jd0': 33.66646041364366},
 {'v5': 31.87961880209994},
 {'introductori': 30.841892061552258},
 {'cy': 28.71858022009789},
 {'titl': 28.010190251053274},
 {'dk': 27.356607275584427},
 {'cxs': 24.509161821966384},
 {'mower': 24.466160631107165},
 {'d1': 24.422073918787923},
 {'rmw': 24.36581862630775},
 {'v5e': 23.16783154652809},
 {'like': 21.926491062038906},
 {'lite': 20.570809342527713},
 {'tr': 19.925039649403804},
 {'g9v4e': 19.677782012899794},
 {'toler': 19.18157693518253},
 {'morn': 18.717487462169895},
 {'00': 18.247527832473146}]
```

Рисунок 13 – Результат векторизации набора данных после стемминга с использованием стоп-слов

Составим сводную таблицу для отображения результатов векторизации и сохраним её в файл Excel. Составленная таблица для обучающего набора данных без применения стемминга представлена на рисунке 14. Для тестового набора данных без применения стемминга представлена на рисунке 15. Для обучающего набора данных с применением стемминга представлен на рисунке 16. Для тестового набора данных с применением стемминга представлен на рисунке 17.

	Count		TF		TF-IDF	
	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
0	{'ax': 62375}	{'ax': 62375}	{'the': 446.4656940475736}	{'windows': 71.81439255394872}	{'the': 162.60429021536854}	{'windows': 39.7445262196255}
1	{'the': 8252}	{'max': 4490}	{'to': 261.22665446068356}	{'like': 54.5113487304099}	{'to': 102.25127726748715}	{'like': 28.008918118727177}
2	{'max': 4490}	{'g9v': 1166}	{'and': 205.2391454163949}	{'use': 53.13620559333252}	{'and': 81.36832968340721}	{'use': 27.171910022459535}
3	{'to': 4457}	{'b8f': 1111}	{'for': 192.4074870803448}	{'know': 45.23828076117531}	{'for': 76.68641363067908}	{'thanks': 26.03241106725381}
4	{'and': 3552}	{'a86': 916}	{'of': 169.08696371281525}	{'thanks': 43.89006613294745}	{'it': 73.52989253705975}	{'know': 25.37531710215435}
5	{'of': 3065}	{'pl': 826}	{'it': 155.34973140754138}	{'new': 41.10851125187996}	{'of': 73.14049941942646}	{'new': 23.644988255365696}
6	{'is': 2799}	{'145': 756}	{'is': 154.12013563203988}	{'just': 39.226259668278665}	{'is': 69.70977368538801}	{'does': 23.577533895766063}
7	{'for': 2744}	{'windows': 719}	{'in': 146.6094196335047}	{'does': 39.000166212753065}	{'in': 62.568116761046504}	{'just': 21.6903989307871}
8	{'it': 2419}	{'1d9': 672}	{'you': 117.07274987561605}	{'used': 37.871521733587635}	{'you': 61.27974186630007}	{'edu': 21.014223496022545}
9	{'in': 2391}	{'00': 651}	{'that': 98.40429548045161}	{'don': 35.704323145993186}	{'that': 53.68564233993087}	{'used': 20.941176184552322}
10	{'you': 1993}	{'use': 571}	{'have': 97.1906091376847}	{'edu': 34.99341982654917}	{'have': 49.29092638324962}	{'mail': 20.92060454000407}
11	{'that': 1713}	{'34u': 549}	{'with': 93.01300913398077}	{'good': 34.190376652753564}	{'with': 46.76683441413689}	{'00': 20.669340203568055}
12	{'with': 1479}	{'1t': 510}	{'or': 84.39187366427578}	{'sale': 33.78559867655903}	{'or': 44.652404227660746}	{'good': 20.581265300443146}
13	{'have': 1438}	{'0t': 505}	{'this': 79.32304211220192}	{'mail': 33.64708925406388}	{'this': 43.22291314488}	{'sale': 20.443368744942852}
14	{'or': 1363}	{'like': 476}	{'on': 77.3371201250956}	{'file': 28.109132980833586}	{'on': 42.84704031115693}	{'file': 20.299105838793295}
15	{'on': 1288}	{'bhj': 456}	{'if': 70.99502055432943}	{'card': 27.873923641453192}	{'are': 39.76222598104707}	{'don': 20.250698917903524}
16	{'this': 1272}	{'75u': 447}	{'are': 69.62363103209533}	{'need': 27.769691894594608}	{'if': 39.66713272933124}	{'card': 19.750641402136843}
17	{'are': 1182}	{'3t': 441}	{'be': 62.340543997594516}	{'using': 27.023144307210433}	{'be': 37.09993721376812}	{'dos': 19.491632090866304}
18	{'g9v': 1166}	{'new': 436}	{'but': 55.17214853046288}	{'offer': 26.670074361863733}	{'windows': 35.131933480856496}	{'offer': 18.28785076259062}
19	{'be': 1135}	{'giz': 433}	{'can': 55.09397266533334}	{'dos': 26.461288189181644}	{'can': 34.188839774160826}	{'looking': 17.969392799477593}

Рисунок 14 – Таблица результата векторизации для обучающего набора данных без применения стемминга

	Count		TF		TF-IDF	
	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
0	{'the': 5303}	{'00': 583}	{'the': 291.83451302743805}	{'windows': 52.138731148601934}	{'the': 110.54034814082188}	{'windows': 29.525766988247245}
1	{'to': 3088}	{'windows': 545}	{'to': 183.90838083387956}	{'like': 34.439374615801476}	{'to': 74.53330706423196}	{'know': 19.401071908073977}
2	{'and': 2434}	{'use': 312}	{'and': 146.156505085496}	{'know': 33.66646041364366}	{'and': 59.49702901688886}	{'dos': 18.328724185534426}
3	{'of': 2139}	{'like': 308}	{'for': 125.43397981498244}	{'use': 31.87961880209994}	{'of': 55.00612226795159}	{'like': 18.22825166865843}
4	{'is': 1796}	{'dos': 281}	{'of': 122.066568487921}	{'just': 30.841892061552258}	{'it': 51.437481151335135}	{'edu': 17.720422089543977}
5	{'for': 1707}	{'new': 273}	{'is': 107.84097417025771}	{'don': 28.71858022009789}	{'for': 51.171080962654656}	{'just': 17.59219450994192}
6	{'it': 1636}	{'know': 271}	{'it': 105.27518554867675}	{'thanks': 28.010190251053274}	{'is': 50.718071500871154}	{'don': 16.953572988898507}
7	{'in': 1527}	{'don': 266}	{'in': 97.01628770404322}	{'edu': 27.356607275584427}	{'that': 44.362737722486365}	{'thanks': 16.89266199027737}
8	{'that': 1281}	{'just': 261}	{'you': 78.68365428133383}	{'does': 24.509161821966384}	{'you': 42.734546090195195}	{'use': 16.745586293513156}
9	{'you': 1264}	{'file': 259}	{'that': 78.59916577831366}	{'new': 24.466160631107165}	{'in': 42.50992737276512}	{'does': 14.870629503496624}
10	{'have': 1009}	{'10': 256}	{'have': 67.79878229557147}	{'dos': 24.422073918787923}	{'have': 35.42627259647494}	{'sale': 14.848851201411348}
11	{'with': 921}	{'50': 220}	{'or': 55.76257536048489}	{'sale': 24.365818626303775}	{'or': 30.117888661918553}	{'00': 14.63805388598326}
12	{'this': 869}	{'used': 218}	{'with': 55.63974205275723}	{'used': 23.16783154652809}	{'on': 30.0179938849296}	{'new': 14.253487477230415}
13	{'on': 860}	{'drive': 211}	{'on': 52.33115839131635}	{'mail': 21.926491062038906}	{'with': 28.318691298767323}	{'drive': 14.107584262411871}
14	{'or': 818}	{'20': 208}	{'this': 49.35365878734478}	{'make': 20.570809342527713}	{'if': 28.21545774406861}	{'os': 14.072038969746567}
15	{'if': 783}	{'edu': 201}	{'if': 48.68433864711239}	{'time': 19.925039649403804}	{'this': 27.8614040902488}	{'program': 14.02943548800833}
16	{'are': 766}	{'time': 194}	{'are': 44.17551403613028}	{'good': 19.677782012899794}	{'are': 26.408399492923788}	{'mail': 13.843080897295174}
17	{'be': 719}	{'does': 187}	{'be': 41.09584961366332}	{'think': 19.18157693518253}	{'can': 25.72077920823754}	{'file': 13.606763403975274}
18	{'but': 618}	{'software': 184}	{'can': 40.015755826654384}	{'need': 18.717487462169895}	{'windows': 25.587587452136255}	{'think': 13.491354914405813}
19	{'not': 616}	{'make': 178}	{'but': 38.73433767399363}	{'00': 18.247527832473146}	{'be': 25.386806878300593}	{'make': 13.380164760732074}

Рисунок 15 – Таблица результата векторизации для тестового набора данных без применения стемминга

	Count		TF		TF-IDF	
	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
0	{'ax': 62375}	{'ax': 62375}	{'the': 436.84019326795124}	{'wmbxlt': 71.81439255394872}	{'the': 163.18149367238405}	{'wmbxlt': 71.81439255394872}
1	{'the': 8249}	{'max': 4490}	{'to': 255.5572536985919}	{'k9': 54.5113487304099}	{'to': 102.82755373820844}	{'k9': 54.5113487304099}
2	{'max': 4490}	{'use': 1290}	{'and': 200.74287736762383}	{'v5': 53.13620559333252}	{'and': 81.7662926378468}	{'v5': 53.13620559333252}
3	{'to': 4457}	{'thi': 1272}	{'for': 188.5383430099311}	{'jd0': 45.23828076117531}	{'for': 77.33052295360623}	{'jd0': 45.23828076117531}
4	{'and': 3552}	{'g9v': 1166}	{'of': 165.41403810545853}	{'titl': 43.89006613294745}	{'it': 76.46507212637587}	{'titl': 43.89006613294745}
5	{'of': 3065}	{'b8f': 1111}	{'it': 159.27872175977353}	{'mower': 41.10851125187996}	{'of': 73.26001364847929}	{'mower': 41.10851125187996}
6	{'is': 2808}	{'a86': 916}	{'is': 152.59232013654986}	{'introductori': 39.226259668278665}	{'is': 70.62789802262674}	{'introductori': 39.226259668278665}
7	{'for': 2744}	{'pl': 829}	{'in': 143.98872667709514}	{'cxs': 39.000166212753065}	{'in': 63.04433639188749}	{'cxs': 39.000166212753065}
8	{'it': 2531}	{'145': 761}	{'you': 114.60481658877012}	{'v5e': 37.871521733587635}	{'you': 61.57191070046019}	{'v5e': 37.871521733587635}
9	{'in': 2394}	{'window': 759}	{'have': 100.55424907563342}	{'cy': 35.704323145993186}	{'that': 54.56537437025866}	{'cy': 35.704323145993186}
10	{'you': 1993}	{'1d9': 672}	{'that': 97.25280082907429}	{'dk': 34.99341982654917}	{'have': 51.78266241945223}	{'dk': 34.99341982654917}
11	{'that': 1720}	{'00': 651}	{'with': 91.19315593199107}	{'g9v4e': 34.190376652753564}	{'with': 47.09624775472306}	{'g9v4e': 34.190376652753564}
12	{'have': 1537}	{'ani': 559}	{'or': 82.7442737385287}	{'rmw': 33.78559867655903}	{'or': 44.89778756660415}	{'rmw': 33.78559867655903}
13	{'with': 1480}	{'34u': 549}	{'thi': 77.76300923413989}	{'like': 33.64708925406388}	{'thi': 43.55554563655819}	{'like': 33.64708925406388}
14	{'or': 1367}	{'wa': 528}	{'use': 76.43777522668313}	{'explod': 28.109132980833586}	{'use': 43.373275014753865}	{'explod': 28.109132980833586}
15	{'on': 1293}	{'1t': 510}	{'on': 76.17828196867185}	{'blast': 27.873923641453192}	{'on': 43.242823778192445}	{'blast': 27.873923641453192}
16	{'use': 1290}	{'ha': 509}	{'if': 69.55210760739459}	{'morn': 27.769691894594608}	{'do': 40.564703597441934}	{'morn': 27.769691894594608}
17	{'thi': 1272}	{'like': 509}	{'are': 68.58927691827455}	{'v6t': 27.023144307210433}	{'are': 40.13792028514502}	{'v6t': 27.023144307210433}
18	{'do': 1232}	{'0t': 505}	{'do': 66.01461842570892}	{'nahf': 26.670074361863733}	{'if': 39.909567124307145}	{'nahf': 26.670074361863733}
19	{'be': 1206}	{'file': 464}	{'be': 64.38224533839065}	{'d1': 26.461288189181644}	{'be': 38.931443147381515}	{'d1': 26.461288189181644}

Рисунок 16 – Таблица результата векторизации для обучающего набора данных с применением стемминга

	Count		TF		TF-IDF	
	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
0	{'the': 5301}	{'thi': 869}	{'the': 284.6029885967805}	{'wmbxlt': 52.138731148601934}	{'the': 110.15007259688721}	{'wmbxlt': 52.138731148601934}
1	{'to': 3088}	{'use': 734}	{'to': 179.69612017633773}	{'k9': 34.439374615801476}	{'to': 74.6970778664183}	{'k9': 34.439374615801476}
2	{'and': 2435}	{'00': 583}	{'and': 142.57862070311032}	{'jd0': 33.66646041364366}	{'and': 59.452672256420016}	{'jd0': 33.66646041364366}
3	{'of': 2139}	{'window': 564}	{'for': 122.798938654992}	{'v5': 31.87961880209994}	{'of': 54.694210335382635}	{'v5': 31.87961880209994}
4	{'is': 1838}	{'file': 380}	{'of': 119.02507446606452}	{'introductori': 30.841892061552258}	{'it': 52.94695312834682}	{'introductori': 30.841892061552258}
5	{'it': 1711}	{'work': 361}	{'is': 108.53282199946798}	{'cy': 28.71858022009789}	{'is': 51.959038586981414}	{'cy': 28.71858022009789}
6	{'for': 1707}	{'ani': 336}	{'it': 106.99220738574931}	{'titl': 28.010190251053274}	{'for': 51.36681718395666}	{'titl': 28.010190251053274}
7	{'in': 1528}	{'ha': 326}	{'in': 94.69206460134237}	{'dk': 27.356607275584427}	{'that': 44.33877517975381}	{'dk': 27.356607275584427}
8	{'that': 1284}	{'wa': 326}	{'that': 76.82189998574334}	{'cxs': 24.509161821966384}	{'you': 42.705239546726474}	{'cxs': 24.509161821966384}
9	{'you': 1264}	{'like': 325}	{'you': 76.8202764406269}	{'mower': 24.466160631107165}	{'in': 42.419008264604095}	{'mower': 24.466160631107165}
10	{'have': 1066}	{'know': 289}	{'have': 69.84627566717808}	{'d1': 24.422073918787923}	{'have': 36.75793858768384}	{'d1': 24.422073918787923}
11	{'with': 921}	{'new': 278}	{'or': 54.50905103825732}	{'rmw': 24.36581862630775}	{'do': 34.455101685751444}	{'rmw': 24.36581862630775}
12	{'do': 913}	{'drive': 267}	{'with': 54.247596317509235}	{'v5e': 23.16783154652809}	{'on': 30.18515384300593}	{'v5e': 23.16783154652809}
13	{'thi': 869}	{'just': 261}	{'do': 53.57089000049436}	{'like': 21.926491062038906}	{'or': 30.111057936497932}	{'like': 21.926491062038906}
14	{'on': 863}	{'10': 256}	{'on': 51.37623605842871}	{'lite': 20.570809342527713}	{'if': 28.29711228464798}	{'lite': 20.570809342527713}
15	{'or': 818}	{'run': 254}	{'thi': 48.19411532114574}	{'tr': 19.925039649403804}	{'with': 28.247127641386854}	{'tr': 19.925039649403804}
16	{'be': 791}	{'doe': 250}	{'if': 47.61979715095931}	{'g9v4e': 19.677782012899794}	{'thi': 27.88182171437507}	{'g9v4e': 19.677782012899794}
17	{'if': 783}	{'pleas': 236}	{'use': 44.05763662308672}	{'toler': 19.18157693518253}	{'window': 27.20067077113115}	{'toler': 19.18157693518253}
18	{'are': 781}	{'program': 236}	{'are': 43.93510968605607}	{'morn': 18.717487462169895}	{'be': 27.19403159253378}	{'morn': 18.717487462169895}
19	{'use': 734}	{'make': 235}	{'be': 43.627460539008794}	{'00': 18.247527832473146}	{'are': 26.861145726964967}	{'00': 18.247527832473146}

Рисунок 17 – Таблица результата векторизации для тестового набора данных с применением стемминга

Используя конвейер (Pipeline) реализуем модель наивного байесовского классификатора и выявим на основе показателей качества (значение полноты, точности, f1-меры и аккуратности), какая предварительная обработка данных обеспечит наилучшие результаты классификации. Полученный результат оптимальных параметров поиска представлен на рисунке 18.

```
from sklearn.metrics import classification_report
from sklearn.naive_bayes import MultinomialNB
```

```
stop_words = [None, 'english']
max_features_values = [100, 500, 1000, 2000, 3000, 4000, 5000]
use_tf = [True, False]
use_idf = [True, False]
```

```
from sklearn.model_selection import GridSearchCV

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

```
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       0.92      0.80      0.86       450
           1       0.89      0.88      0.88       396
           2       0.75      0.89      0.81       331

    accuracy                           0.85      1177
   macro avg       0.85      0.86      0.85      1177
weighted avg       0.86      0.85      0.85      1177
```

```
gscv.best_params_
```

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

Рисунок 18 – Результат классификации после нахождения оптимальных параметров через конвейер

Вывод

В ходе выполнения данной лабораторной работы я приобрел навыки предварительной обработки текстовых данных. В практической части исследования были использованы различные методы подсчета слов, включая как использование стемминга, так и без него. Кроме того, был применен метод векторизации с использованием TfidfTransformer с разными способами взвешивания. С использованием конвейера и сетки решений были найдены оптимальные наборы параметров для классификации, метрика которых базируется на оценках качества.

В результате исследования, наиболее лучшим способом предварительной обработки данных является векторизация TfidfTransformer с использованием TF-IDF взвешиваний и количество информативных терминов = 5000.

Приложение А

Исходный код

```
#!/usr/bin/env python
# coding: utf-8

# # Лабораторная работа №2
# ### Задание
# ### Вариант №7
# ### Классы 3, 7, 13 ('comp.os.ms-windows.misc', 'misc.forsale',
'sci.electronics')

# In[1]:


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


# In[2]:


categories = ['comp.os.ms-windows.misc', 'misc.forsale', 'sci.electronics']
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)


# In[3]:


twenty_train_full.data[0]


# In[4]:


twenty_test_full.data[0]


# ### Применение стемминга

# In[5]:


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

nltk.download('punkt')


# 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]:


stem_train[0]


# In[9]:


stem_test[0]


# ### Векторизация выборки

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

# In[10]:


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


# In[11]:


vect_without_stop = CountVectorizer(max_features=10000)


# In[12]:


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


# In[13]:


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]
```

```
# In[14]:


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


# ### Отсечение стоп-слов

# In[15]:


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


# In[16]:


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


# In[17]:


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


# ### Для данных после стемминга

# #### Без стоп-слов

# 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


# #### С использованием стоп-слов

# In[21]:


vect_stem = CountVectorizer(max_features=10000, stop_words='english')


# In[22]:


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


# In[23]:


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


# ### Векторизация выборки с помощью TfidfTransformer (TF и TF-IDF)

# #### Без использования стоп-слов

# In[24]:


from sklearn.feature_extraction.text import TfidfTransformer


# In[25]:


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


# In[26]:


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)


# In[27]:


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


# #### С использованием стоп-слов

# In[28]:


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


# In[29]:


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)


# In[30]:


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


# #### Со стеммингом без стоп-слов

# In[31]:


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


# In[32]:
```

```
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)


# In[33]:


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


# #### Со стеммингом с использованием стоп-слов

# In[34]:


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


# In[35]:


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)


# In[36]:


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


# ### Составление таблицы

# In[37]:


import pandas as pd


# In[38]:


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


# #### Без стемминга

# In[39]:


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


# In[40]:


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


# #### Со стеммингом

# In[41]:


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
```

```
# In[42]:
```

```
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
```

```
# #### Запись в файл
```

```
# In[43]:
```

```
import openpyxl
```

```
# In[44]:
```

```
writer = pd.ExcelWriter('result.xlsx', engine='openpyxl')

df1.to_excel(writer, sheet_name='Train, wo stem')
df2.to_excel(writer, sheet_name='Test, wo stem')
df3.to_excel(writer, sheet_name='Train, with stem')
df4.to_excel(writer, sheet_name='Test, with stem')

writer.close()
```

```
# ### Конвейер
```

```
# In[45]:
```

```
from sklearn.metrics import classification_report
from sklearn.naive_bayes import MultinomialNB
```

```
# In[46]:
```

```
stop_words = [None, 'english']
max_features_values = [100, 500, 1000, 2000, 3000, 4000, 5000]
use_tf = [True, False]
use_idf = [True, False]
```

```
# In[47]:
```

```
def prepare(data, max_feature, stop_word, use_tf, use_idf):
    tf = None
    cv = CountVectorizer(max_features=max_feature, stop_words=stop_word)
    cv.fit(data)
    if use_tf:
        tf = TfidfTransformer(use_idf=use_idf)
        tf.fit(cv.transform(data))
    return cv, tf


# In[48]:


result = []

for max_features_value in max_features_values:
    for stop_word in stop_words:
        for ut in use_tf:
            for ui in use_idf:
                options = {}
                cv, tf = prepare(twenty_train_full.data, max_features_value,
stop_word, ut, ui)
                if tf:
                    clf = MultinomialNB()

clf.fit(tf.transform(cv.transform(twenty_train_full.data)),
twenty_train_full.target)
                    prep_test =
tf.transform(cv.transform(twenty_test_full.data))
                else:
                    clf = MultinomialNB()
                    clf.fit(cv.transform(twenty_train_full.data),
twenty_train_full.target)
                    prep_test = cv.transform(twenty_test_full.data)

                options['features'] = max_features_value
                options['stop_words'] = stop_word
                options['use_tf'] = ut
                options['use_idf'] = ui

                result_data = classification_report(clf.predict(prep_test),
twenty_test_full.target, output_dict=True)
                result_df = pd.DataFrame(result_data)
                result.append({
                    'df': result_df,
                    'options': options
                })


# In[49]:


writer = pd.ExcelWriter('result_compare.xlsx', engine='openpyxl')

df = pd.DataFrame(columns=['Номер страницы', 'features', 'stop_words',
'use_tf', 'use_idf'])
for it, item in enumerate(result):
    for key, value in item['options'].items():
        df.at[it, key] = value
    df.at[it, 'Номер страницы'] = it

df.to_excel(writer, sheet_name='Оглавление')
```

```
for it, item in enumerate(result):
    df_new = pd.DataFrame(item['df'])
    df_new.to_excel(writer, sheet_name=f'Страница {it}')

writer.close()


# In[50]:


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())])


# In[51]:


from sklearn.model_selection import GridSearchCV

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


# In[52]:


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


# In[53]:


gscv.best_params_
```