

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

Посаднев В.В.

Группа М-ИАП-22-1

Руководитель

Кургасов В.В.

Доцент

Липецк 2022 г.

Задание кафедры

- 1) В среде Jupyter 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 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.
- 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:
 - a. Наличие – отсутствие стемминга
 - b. Отсечение – не отсечение стоп-слов

c. Количество информативных терминов (max_features)

d. 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 №11

Классы: misc.forsale, sci.med, talk.religion.misc

Ход работы

Загрузим обучающую и экзаменационную выборку в соответствии с вариантом №11, а именно misc.forsale, sci.med, talk.religion.misc. Код для загрузки данных представлен на рисунке 1.

```
categories = ['misc.forsale', 'sci.med', 'talk.religion.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 1 – Код для загрузки данных в соответствии с вариантом №11

После успешной загрузки данных посмотрим на записи путем вывода одного значения, который представлен на рисунке 2.

```
In 4 1 twenty_train_full.data[0]

Out 4 1 "Not to mention the thread about selling someone's wife. I am a guy, therefore\nnot overly bummed
        by it, but a little common sense would dictate that this\nis offensive to many women, and not really
        necessary.\n\n-- \n-----\n\nScott
        Ferguson                      Exxon Research & Engineering Co.\nProject Engineer
        New Jersey"

In 5 1 twenty_test_full.data[0]

Out 5 1 'I have two brand new Dayna Etherprint Adapters (10baset) for sale.\nThey convert ethertalk to
        localtalk. This is useful when wanting to\nhook up a localtalk network printer to a ethertalk
        (10baset) network.\nThey sell for $350 each in Mac Warehouse. Will take $100 each.\nGuaranteed
        .\n\nemail response to atg@virginia.edu'
```

Рисунок 2 – Пример загруженных данных

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

```

In 7 1 def stemming(data):
      2     porter_stemmer = PorterStemmer()
      3     stem = []
      4     for text in data:
      5         nltk_tokens = word_tokenize(text)
      6         line = ''.join([' ' + porter_stemmer.stem(word) for word in nltk_tokens])
      7         stem.append(line)
      8     return stem

In 8 1 stem_train = stemming(twenty_train_full.data)
      2 stem_test = stemming(twenty_test_full.data)

In 9 1 stem_train[0]

Out 9  ✓ " not to mention the thread about sell someon 's wife . i am a guy , therefor not overli bum by it ,
          but a littl common sens would dictat that thi is offens to mani women , and not realli necessari .
          -- -- -- -- --
          -- -- -- scott ferguson exxon research & engin co. project engin new jersey"

In 10 1 stem_test[0]

Out 10  ✓ ' i have two brand new dayna etherprint adapt ( 10baset ) for sale . they convert ethertalk to
          localtalk . thi is use when want to hook up a localtalk network printer to a ethertalk ( 10baset )
          network . they sell for $ 350 each in mac warehous . will take $ 100 each . guarante . email respons
          to atg @ virginia.edu'

```

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': 7706},
 {'of': 4314},
 {'to': 4227},
 {'and': 3922},
 {'in': 2670},
 {'is': 2596},
 {'that': 2302},
 {'for': 2017},
 {'it': 1819},
 {'you': 1541},
 {'have': 1230},
 {'with': 1157},
 {'are': 1149},
 {'this': 1149},
```

⋮

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

Применим процедуру отсеечения стоп-слов и повторим вывод полученных результатов. Код для обработки данных путем отсеечения стоп-

слов представлен на рисунке 6. Результат векторизации обучающей и тестовой выборки простым подсчетом слов с отсечением стоп-слов представлен на рисунке 7.

```
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)
```

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

```
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  
  
{'00': 560},  
{'10': 351},  
{'god': 328},  
{'like': 314},  
{'new': 306},  
{'know': 301},  
{'don': 292},  
{'people': 288},  
{'just': 249},  
{'good': 242},  
{'20': 239},  
{'time': 228},  
{'edu': 220},  
{'50': 214},
```

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': 7706},
 {'of': 4314},
 {'to': 4227},
 {'and': 3923},
 {'in': 2671},
 {'is': 2633},
 {'that': 2306},
 {'for': 2017},
 {'it': 1916},
 {'you': 1540},
 {'have': 1317},
 {'thi': 1199},
 {'are': 1167},
 {'with': 1157},

```

Рисунок 8 – Результат векторизации обучающей и тестовой выборки после стемминга простым подсчетом слов без 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': 1199},
 {'wa': 689},
 {'00': 560},
 {'use': 521},
 {'ha': 498},
 {'god': 378},
 {'10': 351},
 {'ani': 347},
 {'like': 342},
 {'know': 330},
 {'hi': 325},
 {'new': 320},
 {'peopl': 289},
 {'doe': 276},

```

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 стоп-слов представлен на рисунке 10, с использованием стоп-слов представлен на рисунке 11.

```
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': 103.92578878716677},
 {'to': 67.46243226585233},
 {'of': 62.14728629487148},
 {'and': 55.67573051098585},
 {'is': 47.115771882782795},
 {'that': 46.15570171423424},
 {'for': 45.38556743347154},
 {'you': 43.75246255205398},
 {'in': 43.31207482495192},
 {'it': 41.98574454787257},
 {'have': 32.082368823761044},
 {'or': 26.963146304013232},
 {'not': 26.798398081584132},
```

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

[{'know': 16.24730869218536},
 {'like': 15.489751099616834},
 {'just': 14.386831590993486},
 {'don': 14.029111887245111},
 {'sale': 13.80560024422138},
 {'00': 13.797739615372626},
 {'god': 13.258610971213697},
 {'good': 12.953128824156405},
 {'new': 12.114588533328869},
 {'think': 11.774347917159494},
 {'mail': 11.7469651636439},
 {'people': 11.217160674133005},
 {'time': 10.76555628971949},

```

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 стоп-слов 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': 102.4167020332997},
 {'to': 66.80087036635662},
 {'of': 61.0606141251206},
 {'and': 54.86535021843155},
 {'is': 47.55257992890465},
 {'that': 45.61780513240577},
 {'for': 44.97402862941401},
 {'you': 43.505968937738125},
 {'it': 43.107902079866456},
 {'in': 42.66363758672977},
 {'have': 33.69279408264143},
 {'not': 26.86917716237715},
 {'or': 26.76815482715484},

```

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

[{'massag': 29.948081204577424},
 {'ll': 29.773079799267403},
 {'earn': 26.879153230636003},
 {'leadership': 26.70362435365944},
 {'sound': 23.870063892545577},
 {'pale': 22.13119244763357},
 {'grind': 21.984139363868216},
 {'tronic': 20.103879527019693},
 {'port': 20.08814309983904},
 {'typefont': 18.825287345123005},
 {'dylan': 17.934617752788245},
 {'endur': 17.86368099065268},
 {'gregori': 17.39807864633475},

```

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.

	A	B	C	D	E	F	G
1		Count		TF		TF-IDF	
2		Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
3							
4	0	{'the': 11301}	{'00': 640}	{'the': 410.6466942091253}	{'don': 42.19836218494229}	{'the': 158.7095535302558}	{'don': 21.80243478947723}
5	1	{'of': 6613}	{'people': 517}	{'to': 251.7142371449254}	{'like': 41.94033888116838}	{'to': 100.59334482026873}	{'like': 21.37695831609965}
6	2	{'to': 6208}	{'new': 504}	{'of': 224.7563651678267}	{'new': 38.49189557572986}	{'of': 96.82672169042559}	{'know': 20.766576231972007}
7	3	{'and': 5710}	{'edu': 502}	{'and': 213.50667797912118}	{'know': 38.43222500691624}	{'and': 86.39016862609914}	{'new': 20.66361310613387}
8	4	{'in': 3962}	{'don': 467}	{'is': 171.8087951834235}	{'just': 36.441798363953794}	{'is': 78.0737463854439}	{'edu': 20.304021113881046}
9	5	{'is': 3857}	{'like': 461}	{'in': 154.16287320333123}	{'people': 36.20820897355665}	{'it': 71.51597224999692}	{'people': 19.788592619256864}
10	6	{'that': 3485}	{'good': 420}	{'for': 150.42899781013273}	{'edu': 34.694153609728126}	{'that': 69.39495091529038}	{'just': 19.258803504577592}
11	7	{'it': 2943}	{'just': 417}	{'it': 144.3601581889401}	{'sale': 33.10807332706017}	{'in': 67.86182033825528}	{'sale': 18.979968418043185}
12	8	{'for': 2894}	{'know': 394}	{'that': 129.90078758447194}	{'good': 32.20601101295828}	{'for': 62.588326880465274}	{'00': 18.701502112747598}
13	9	{'you': 2402}	{'10': 358}	{'you': 107.74646955575315}	{'think': 28.946781478743482}	{'you': 60.005846885521684}	{'good': 18.189793531301607}
14	10	{'this': 1766}	{'use': 356}	{'with': 76.73952240379423}	{'does': 26.955539997275125}	{'this': 41.37365799621488}	{'think': 16.800161577150433}
15	11	{'are': 1753}	{'god': 338}	{'have': 74.76904182863568}	{'time': 26.0987325817474}	{'are': 40.00414095201544}	{'offer': 16.522127130213185}
16	12	{'with': 1736}	{'time': 336}	{'this': 73.98236585775292}	{'offer': 25.200411794386508}	{'have': 39.4882262676624}	{'does': 16.12614063719485}
17	13	{'not': 1711}	{'think': 328}	{'are': 72.3717564019299}	{'used': 25.150208862674478}	{'with': 38.8104458820901}	{'god': 15.548477911354187}
18	14	{'have': 1632}	{'does': 313}	{'or': 70.35049990487211}	{'00': 22.847118231150784}	{'not': 38.444939640920175}	{'used': 14.77771917391587}
19	15	{'be': 1555}	{'20': 285}	{'not': 65.67375220517485}	{'make': 22.24617480778731}	{'be': 37.677253780707396}	{'time': 14.743163825137326}
20	16	{'or': 1504}	{'used': 275}	{'be': 63.48328583893221}	{'use': 21.080078313079557}	{'or': 37.30203300467406}	{'shipping': 14.37929479105371}
21	17	{'as': 1433}	{'50': 261}	{'if': 57.567519020817166}	{'god': 21.00989280692224}	{'as': 32.73502619945886}	{'email': 13.466442391427682}
22	18	{'on': 1314}	{'com': 259}	{'on': 54.056582370621975}	{'interested': 20.947258397082738}	{'if': 32.43874285684031}	{'make': 13.3106674232951}
23	19	{'but': 1143}	{'jesus': 258}	{'as': 50.53911703806337}	{'shipping': 20.37436327922064}	{'on': 31.670689631446457}	{'condition': 13.255658572127189}

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

	A	B	C	D	E	F	G
1		Count		TF		TF-IDF	
2		Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
3							
4	0	{'the': 7706}	{'00': 560}	{'the': 264.43850063653423}	{'like': 29.948081204577424}	{'the': 103.92578878716677}	{'know': 16.24730869218536}
5	1	{'of': 4314}	{'10': 351}	{'to': 165.4520748999254}	{'know': 29.773079799267403}	{'to': 67.46243226585233}	{'like': 15.489751099616834}
6	2	{'to': 4227}	{'god': 328}	{'of': 142.71459444454686}	{'don': 26.879153230636003}	{'of': 62.14728629487148}	{'just': 14.386831590993486}
7	3	{'and': 3922}	{'like': 314}	{'and': 135.3763341296705}	{'just': 26.70362435365944}	{'and': 55.67573051098585}	{'don': 14.029111887245111}
8	4	{'in': 2670}	{'new': 306}	{'for': 106.7211060032294}	{'sale': 23.870063892545577}	{'is': 47.115771882782795}	{'sale': 13.80560024422138}
9	5	{'is': 2596}	{'know': 301}	{'is': 102.38992170757815}	{'new': 22.13119244763357}	{'that': 46.15570171423424}	{'00': 13.797739515372626}
10	6	{'that': 2302}	{'don': 292}	{'in': 96.31664067346627}	{'good': 21.984139363868216}	{'for': 45.38556743347154}	{'god': 13.258610971213697}
11	7	{'for': 2017}	{'people': 288}	{'that': 85.72866577881462}	{'think': 20.103879527019693}	{'you': 43.75246255205398}	{'good': 12.953128824156405}
12	8	{'it': 1819}	{'just': 249}	{'it': 84.31251661554674}	{'people': 20.08814309983904}	{'in': 43.31207482495192}	{'new': 12.114588533328869}
13	9	{'you': 1541}	{'good': 242}	{'you': 76.62890164443824}	{'time': 18.825287345123005}	{'it': 41.98574454787257}	{'think': 11.774347917159494}
14	10	{'have': 1230}	{'20': 239}	{'have': 59.782972354009544}	{'does': 17.934617752788245}	{'have': 32.082368823761044}	{'mail': 11.7469651636439}
15	11	{'with': 1157}	{'time': 228}	{'with': 50.54776335071849}	{'edu': 17.86368099065268}	{'or': 26.963146304013232}	{'people': 11.217160674133005}
16	12	{'are': 1149}	{'edu': 220}	{'or': 49.51766448693027}	{'god': 17.39807864633475}	{'not': 26.798398081584132}	{'time': 10.76555628971949}
17	13	{'this': 1149}	{'50': 214}	{'this': 46.257322990834744}	{'used': 17.273875588234212}	{'this': 26.54414001492925}	{'does': 10.75280532883665}
18	14	{'not': 1084}	{'12': 212}	{'are': 45.58212548464762}	{'mail': 17.20653379928399}	{'are': 26.139018385325716}	{'thanks': 10.731058363911187}
19	15	{'or': 1009}	{'does': 205}	{'not': 44.61417757310946}	{'00': 16.990129981097365}	{'with': 26.089883896229}	{'ve': 10.416912964357634}
20	16	{'be': 1002}	{'92': 204}	{'if': 40.58806741479666}	{'make': 16.729737142292908}	{'be': 24.02821218376009}	{'used': 10.294090022947794}
21	17	{'as': 932}	{'use': 204}	{'on': 39.910061357543114}	{'offer': 15.708337776995725}	{'if': 23.8169899007375}	{'offer': 10.221700719462152}
22	18	{'on': 926}	{'25': 202}	{'be': 39.15623548893515}	{'ve': 15.610096759328895}	{'on': 23.72047883885346}	{'edu': 10.177793191710643}
23	19	{'if': 790}	{'medical': 201}	{'as': 32.12860833436083}	{'thanks': 15.274593080166266}	{'as': 21.37691832162967}	{'make': 10.132834220143167}

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

	A	B	C	D	E	F	G
1		Count		TF		TF-IDF	
2		Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
3							
4	0	{'the': 11298}	{'thi': 1770}	{'the': 401.75578691089754}	{'earn': 42.19836218494229}	{'the': 157.79831839320482}	{'earn': 42.19836218494229}
5	1	{'of': 6613}	{'wa': 1069}	{'to': 246.39046279013354}	{'massag': 41.94033888116838}	{'to': 100.54188184209593}	{'massag': 41.94033888116838}
6	2	{'to': 6208}	{'use': 808}	{'of': 219.99708531926314}	{'pale': 38.49189557572986}	{'of': 96.36850919046363}	{'pale': 38.49189557572986}
7	3	{'and': 5712}	{'ha': 732}	{'and': 208.88810525625937}	{'ll': 38.43222500691624}	{'and': 86.04795469898639}	{'ll': 38.43222500691624}
8	4	{'in': 3964}	{'00': 640}	{'is': 171.1104805078897}	{'leadership': 36.441798363953794}	{'is': 79.0033745727237}	{'leadership': 36.441798363953794}
9	5	{'is': 3922}	{'ani': 538}	{'in': 150.9005456172757}	{'port': 36.20820897355665}	{'it': 73.31936518758786}	{'port': 36.20820897355665}
10	6	{'that': 3488}	{'new': 529}	{'it': 147.38739013572695}	{'endur': 34.694153609728126}	{'that': 69.29949321110132}	{'endur': 34.694153609728126}
11	7	{'it': 3111}	{'like': 518}	{'for': 146.88012385811385}	{'sound': 33.10807332706017}	{'in': 67.48075467301989}	{'sound': 33.10807332706017}
12	8	{'for': 2894}	{'peopl': 518}	{'that': 127.31873045176705}	{'grind': 32.20601101295828}	{'for': 62.181913294819346}	{'grind': 32.20601101295828}
13	9	{'you': 2401}	{'edu': 502}	{'you': 105.54217923935947}	{'tronic': 28.946781478743482}	{'you': 59.995166222979435}	{'tronic': 28.946781478743482}
14	10	{'are': 1786}	{'hi': 498}	{'have': 79.37206678905693}	{'dylan': 26.955539997275125}	{'have': 41.708464124313906}	{'dylan': 26.955539997275125}
15	11	{'not': 1780}	{'doe': 440}	{'with': 75.16031613421946}	{'typefont': 26.0987325817474}	{'thi': 41.41806858705026}	{'typefont': 26.0987325817474}
16	12	{'have': 1774}	{'know': 438}	{'thi': 72.6923434645989}	{'perciv': 25.200411794386508}	{'are': 40.730013978082994}	{'perciv': 25.200411794386508}
17	13	{'thi': 1770}	{'good': 430}	{'are': 72.46612914192293}	{'weather': 25.15020882674478}	{'be': 40.60952442256573}	{'weather': 25.15020882674478}
18	14	{'be': 1764}	{'onli': 420}	{'be': 69.19071366723983}	{'00': 22.847118231150784}	{'not': 39.275869509026585}	{'00': 22.847118231150784}
19	15	{'with': 1737}	{'just': 417}	{'or': 68.83200312335784}	{'miner': 22.24617480778731}	{'with': 38.75291671354243}	{'miner': 22.24617480778731}
20	16	{'or': 1504}	{'time': 414}	{'not': 66.22164742443427}	{'wear': 21.080078313079557}	{'or': 37.1367365602891}	{'wear': 21.080078313079557}
21	17	{'as': 1431}	{'say': 400}	{'if': 56.34259136328616}	{'gregori': 21.00989280692224}	{'do': 35.13133751221469}	{'gregori': 21.00989280692224}
22	18	{'do': 1386}	{'think': 393}	{'do': 55.09326254250311}	{'join': 20.947258397082738}	{'as': 32.523517894221236}	{'join': 20.947258397082738}
23	19	{'on': 1320}	{'make': 381}	{'on': 53.14025829598756}	{'stx': 20.37436327922064}	{'if': 32.50089178966867}	{'stx': 20.37436327922064}

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

	A	B	C	D	E	F	G
1		Count		TF		TF-IDF	
2		Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
3							
4	0	{'the': 7706}	{'thi': 1199}	{'the': 257.92253141743}	{'massag': 29.948081204577424}	{'the': 102.4167020332997}	{'massag': 29.948081204577424}
5	1	{'of': 4314}	{'wa': 689}	{'to': 161.58217852697967}	{'ll': 29.773079799267403}	{'to': 66.80087036635662}	{'ll': 29.773079799267403}
6	2	{'to': 4227}	{'00': 560}	{'of': 138.969377940274}	{'earn': 26.879153230636003}	{'of': 61.0606141251206}	{'earn': 26.879153230636003}
7	3	{'and': 3923}	{'use': 521}	{'and': 131.9264434926551}	{'leadership': 26.70362435365944}	{'and': 54.86535021843155}	{'leadership': 26.70362435365944}
8	4	{'in': 2671}	{'ha': 498}	{'for': 104.14329159064731}	{'sound': 23.870063892545577}	{'is': 47.55257992890465}	{'sound': 23.870063892545577}
9	5	{'is': 2633}	{'god': 378}	{'is': 102.16292180234682}	{'pale': 22.13119244763357}	{'that': 45.61780513240577}	{'pale': 22.13119244763357}
10	6	{'that': 2306}	{'10': 351}	{'in': 93.95093720037718}	{'grind': 21.984139363868216}	{'for': 44.97402862941401}	{'grind': 21.984139363868216}
11	7	{'for': 2017}	{'ani': 347}	{'it': 86.50120545821625}	{'tronic': 20.103879527019693}	{'you': 43.505968937738125}	{'tronic': 20.103879527019693}
12	8	{'it': 1916}	{'like': 342}	{'that': 83.82705059369427}	{'port': 20.08814309983904}	{'it': 43.107902079866456}	{'port': 20.08814309983904}
13	9	{'you': 1540}	{'know': 330}	{'you': 74.86410865202238}	{'typefont': 18.825287345123005}	{'in': 42.66363758672977}	{'typefont': 18.825287345123005}
14	10	{'have': 1317}	{'hi': 325}	{'have': 63.54925811141936}	{'dylan': 17.934617752788245}	{'have': 33.69279408264143}	{'dylan': 17.934617752788245}
15	11	{'thi': 1199}	{'new': 320}	{'with': 49.181030980627284}	{'endur': 17.86368099065268}	{'not': 26.86917716237715}	{'endur': 17.86368099065268}
16	12	{'are': 1167}	{'peopl': 289}	{'or': 48.393434221928636}	{'gregori': 17.39807864633475}	{'or': 26.76815482715484}	{'gregori': 17.39807864633475}
17	13	{'with': 1157}	{'doe': 276}	{'thi': 45.45057235122236}	{'weather': 17.273875588234212}	{'thi': 26.448880489356604}	{'weather': 17.273875588234212}
18	14	{'be': 1143}	{'time': 271}	{'are': 45.28657427458017}	{'mildli': 17.20653379928399}	{'are': 26.070506658760067}	{'mildli': 17.20653379928399}
19	15	{'not': 1116}	{'make': 265}	{'not': 44.53968532734706}	{'00': 16.990129981097365}	{'with': 25.715909070863507}	{'00': 16.990129981097365}
20	16	{'or': 1009}	{'say': 263}	{'be': 42.71042458581328}	{'miner': 16.729737142292908}	{'be': 25.66065041027378}	{'miner': 16.729737142292908}
21	17	{'on': 933}	{'just': 249}	{'if': 39.69464615053347}	{'perciv': 15.70833776995725}	{'do': 23.831879096909034}	{'perciv': 15.70833776995725}
22	18	{'as': 931}	{'good': 244}	{'on': 39.214057720326736}	{'whatsoev': 15.610096759328895}	{'if': 23.719008715916885}	{'whatsoev': 15.610096759328895}
23	19	{'do': 800}	{'onli': 241}	{'do': 36.976857767659254}	{'trash': 15.274593080166266}	{'on': 23.57406476332026}	{'trash': 15.274593080166266}

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.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)

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

```

	precision	recall	f1-score	support
0	0.94	0.92	0.93	396
1	0.91	0.82	0.86	441
2	0.72	0.90	0.80	200
accuracy			0.87	1037
macro avg	0.85	0.88	0.86	1037
weighted avg	0.88	0.87	0.87	1037

```

gscv.best_params_

{'tfidf__use_idf': True,
 'vect__max_features': 2000,
 'vect__stop_words': 'english'}

```

Рисунок 18 – Результат классификации после нахождения оптимальных параметров через конвейер

Заключение

В ходе выполнения данной лабораторной работы мною были получены навыки предварительной обработки текстовых данных. На практике применены способы подсчета слов без применения стемминга и вместе с ним. Также были применен метод векторизации TfidfTransformer с различным способом взвешивания. С помощью конвейера и сетки решений были найдены оптимальные наборы параметров для классификации метрика которых основывается на показателях качества.

Приложение А

Исходный код

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

# In[1]:

# Загрузим обучающую и экзаменационную выборку
## Вариант №11: [misc.forsale, sci.med, talk.religion.misc]

# In[2]:

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

# In[3]:

categories = ['misc.forsale', 'sci.med', 'talk.religion.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)

# In[4]:

twenty_train_full.data[0]

# In[5]:

twenty_test_full.data[0]

# # Применение стемминга

# In[6]:

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

nltk.download('punkt')

# In[7]:

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[8]:

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

# In[9]:

stem_train[0]

# In[10]:

stem_test[0]

# # Векторизация выборки

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

# In[11]:

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

# In[12]:

vect_without_stop = CountVectorizer(max_features=10000)

# In[13]:

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

# In[14]:

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[15]:

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[16]:

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

# In[17]:

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

# In[18]:

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[19]:

vect_stem_without_stop = CountVectorizer(max_features=10000)

# In[20]:

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[21]:

```

```

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[22]:

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

# In[23]:

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

# In[24]:

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[25]:

from sklearn.feature_extraction.text import TfidfTransformer

# In[26]:

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

# In[27]:

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[28]:

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[29]:

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


# In[30]:

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[31]:

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[32]:

```

```

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

# In[33]:

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[34]:

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[35]:

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

# In[36]:

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[37]:

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[38]:

import pandas as pd

# In[39]:

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

# ## Без стемминга

# In[40]:

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[41]:

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[42]:

```

```

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[43]:

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[44]:

import openpyxl

# In[45]:

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.save()

# ## Конвейер

# In[46]:

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

# In[47]:

```



```

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

```

```

# In[48]:

```

```

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[49]:

```

```

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[50]:

```

```

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.save()

```

In[51]:

```

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[52]:

```

from sklearn.model_selection import GridSearchCV

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

```

In[53]:

```

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

```

In[54]:

```

gscv.best_params_

```