**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине

«Прикладные интеллектуальные системы и экспертные системы»

Студент Первушин О. С.

Группа М-ИАП-23-1

Руководитель Кургасов В. В.

доцент, канд. пед. наук

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:
   1. Векторизовать обучающую и тестовую выборки простым подсчетом слов (CountVectorizer) и значеним max\_features = 10000.
   2. Вывести и проанализировать первые 20 наиболее частотных слов всей выборки и каждого класса по-отдельности.
   3. Применить процедуру отсечения стоп-слов и повторить пункт b.
   4. Провести пункты a – c для обучающей и тестовой выборки, для которой проведена процедура стемминга.
   5. 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.
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

На рисунке 1 представлены импортируемы необходимые библиотеки и модули для работы программы.

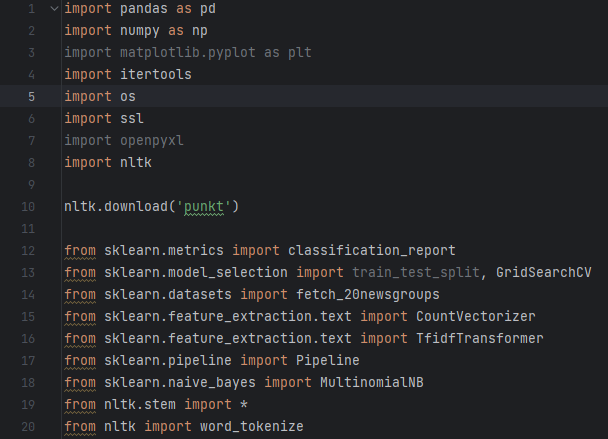


Рисунок 1 — Импортируемые библиотеки и модули

Загрузим обучающую и экзаменационную выборку в соответствии с вариантом. Для этого воспользуемся написанной ранее функцией (рис. 2) и передадим в нее параметры, соответствующие варианту. Также выведем небольшую часть текста из каждой категории (рис. 3).



Рисунок 2 — Функция для получения данных

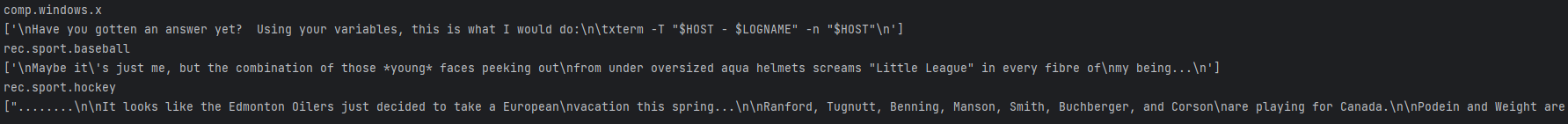


Рисунок 3 — Демонстрация текста категорий, заданных по варианту

Приведем код функции, реализующей стемминг (рис. 4).

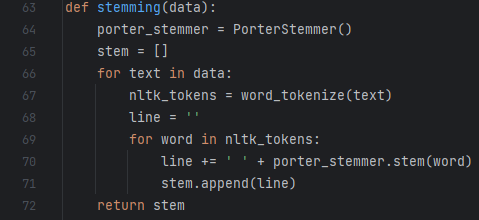


Рисунок 4 — Функция стемминга

Векторизуем обучающую и тестовую выборки простым подсчетом слов (CountVectorizer) и значеним max\_features = 10000, выведем и проанализируем первые 20 наиболее частотных слов всей выборки и каждого класса по отдельности, применим процедуру отсечения стоп-слов, проведем пункты для обучающей и тестовой выборки, для которой проведена процедура стемминга, векторизируем выборки с помощью TfidfTransformer (с использованием TF и TF-IDF взвешиваний) и повторить пункты. Заполним таблицы наиболее частотными терминами обучающей выборки и каждого класса по отдельности. Получилось по 4 таблицы для выборки, к которой применялась операция стемминга и 4 таблицы для выборки, к которой операция стемминга не применялась. Таблицы представлены на рисунках 5 — 12.

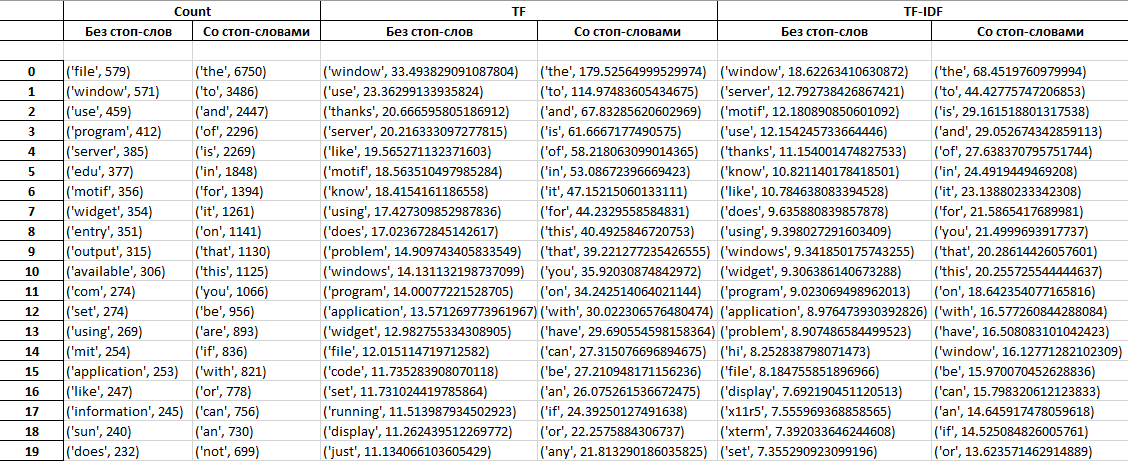


Рисунок 5 — Без стемминга для comp.windows.x

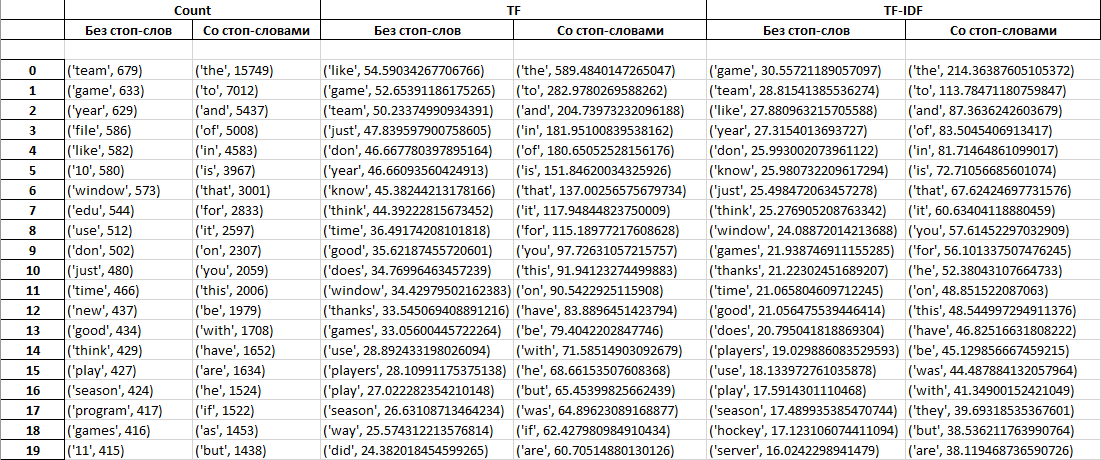


Рисунок 6 — Без стемминга для всех категорий

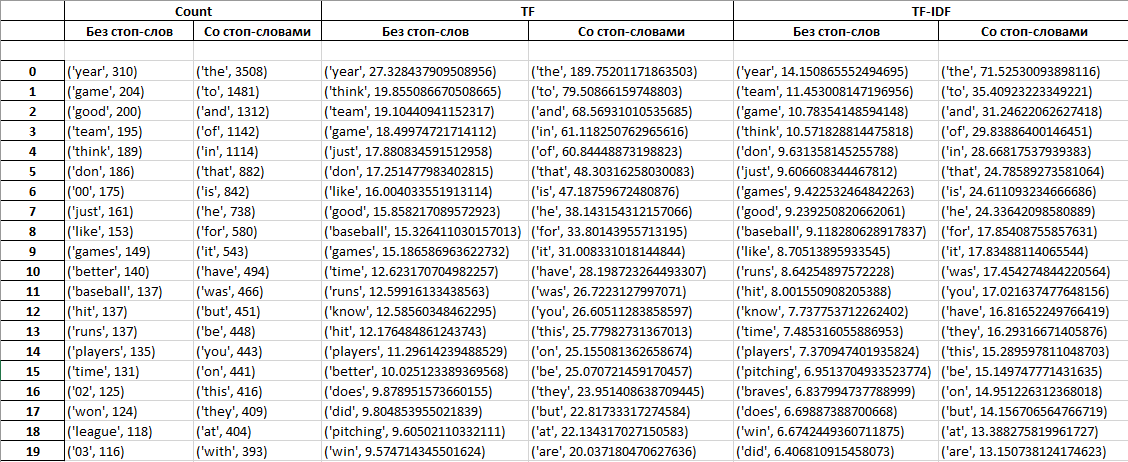


Рисунок 7 — Без стемминга для rec.sport.baseball

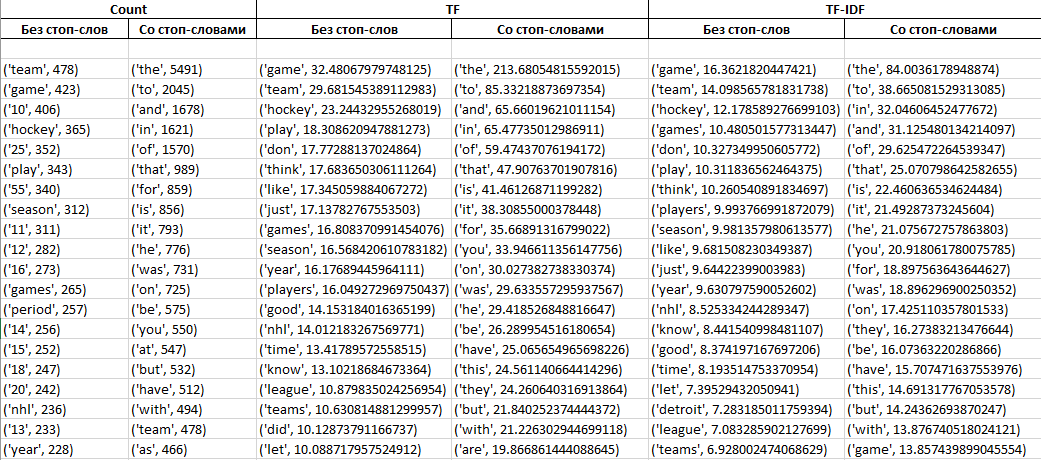


Рисунок 8 — Без стемминга для rec.sport.hockey

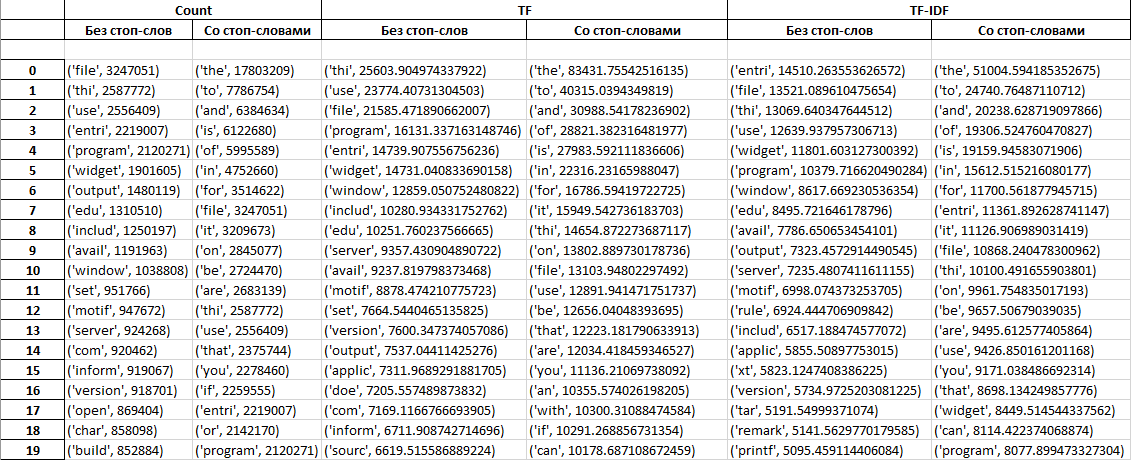


Рисунок 9 — Со стеммингом comp.windows.x

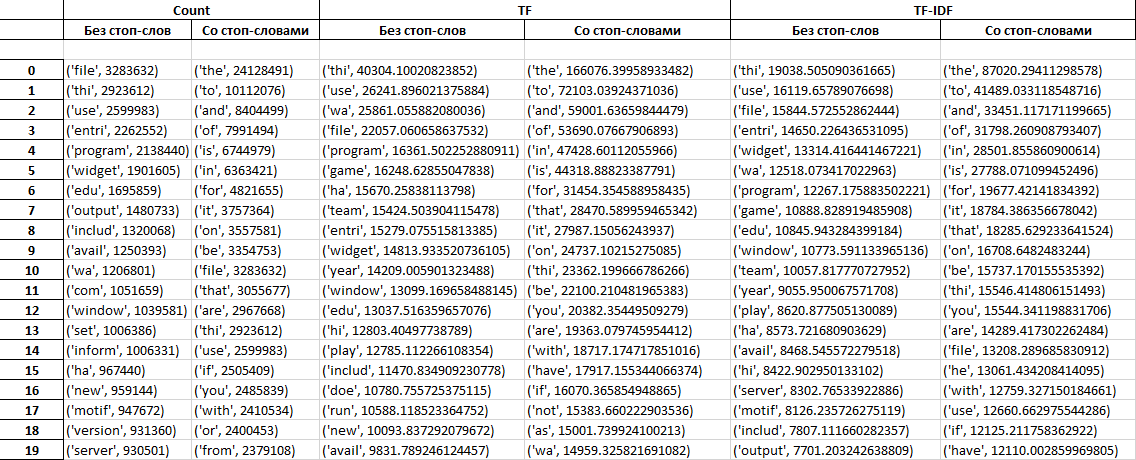


Рисунок 10 — Со стеммингом для всех категорий

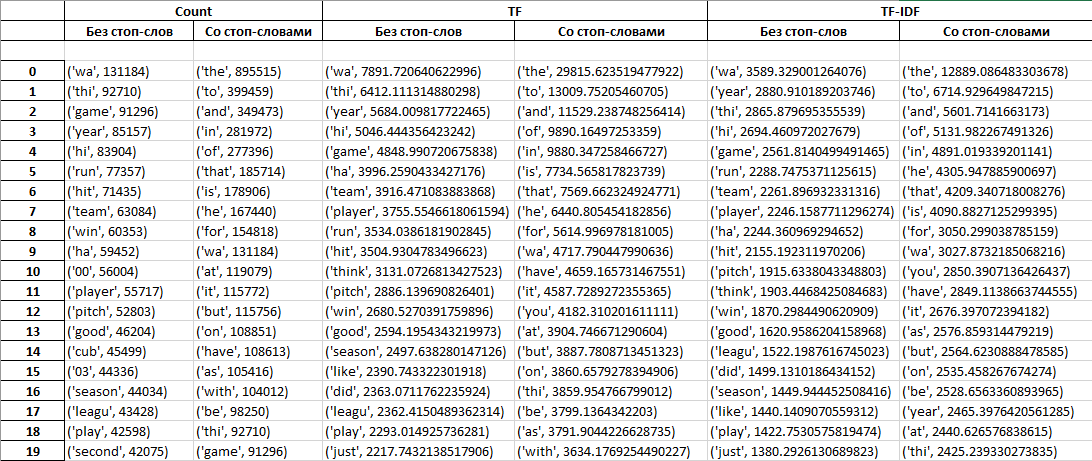


Рисунок 11 — Со стеммингом для rec.sport.baseball

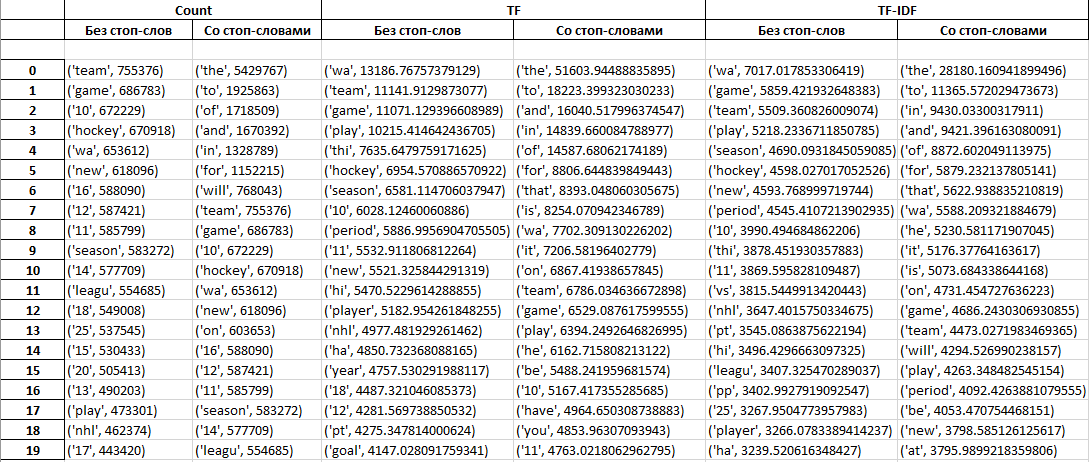


Рисунок 12 — Со стеммингом для rec.sport.hockey

Используя конвейер (Pipeline), реализуем модель наивного байесовского классификатора и выявим на основе показателей качества (значения полноты, точности, f1-меры и аккуратности) какая предварительная обработка данных обеспечит наилучшие результаты классификации. Исследуем следующие характеристики: отсечение и не отсечение стоп-слов, количество информативных терминов (max\_features), взвешивание (Count, TF, TF-IDF). На рисунках 13 — 53 приведены примеры работы программы с указанными параметрами.

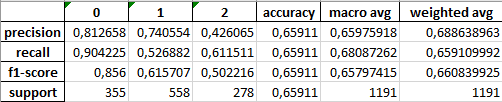


Рисунок 13 — Пример работы программы с параметрами «max\_features=100\_stop\_words=english\_use\_tf=False\_use\_idf =False»

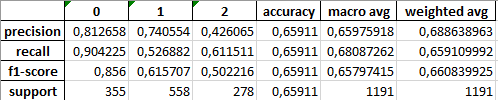


Рисунок 14 — Пример работы программы с параметрами «max\_features=100\_stop\_words=english\_use\_tf=False\_use\_idf =True»

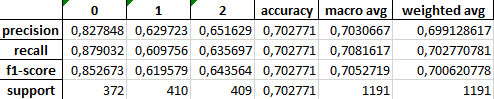


Рисунок 15 — Пример работы программы с параметрами «max\_features=100\_stop\_words=english\_use\_tf=True\_use\_idf =False»

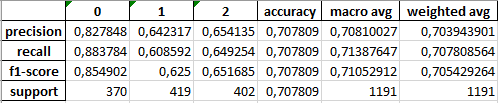


Рисунок 16 — Пример работы программы с параметрами «max\_features=100\_stop\_words=english\_use\_tf=True\_use\_idf =True»

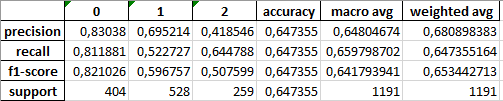


Рисунок 17 — Пример работы программы с параметрами «max\_features=100\_stop\_words=None\_use\_tf=False\_use\_idf =False»

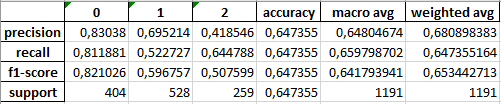


Рисунок 18 — Пример работы программы с параметрами «max\_features=100\_stop\_words=None\_use\_tf=False\_use\_idf =True»

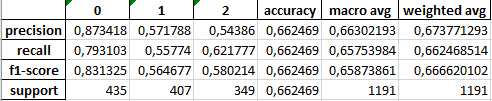


Рисунок 19 — Пример работы программы с параметрами «max\_features=100\_stop\_words=None\_use\_tf=True\_use\_idf =False»

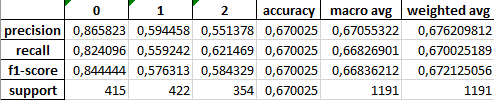


Рисунок 20 — Пример работы программы с параметрами «max\_features=100\_stop\_words=None\_use\_tf=True\_use\_idf =True»

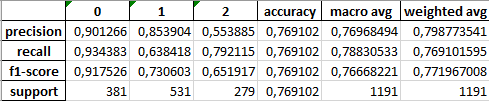


Рисунок 21 — Пример работы программы с параметрами «max\_features=500\_stop\_words=english\_use\_tf=False\_use\_idf =False»

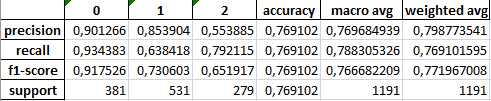


Рисунок 22 — Пример работы программы с параметрами «max\_features=500\_stop\_words=english\_use\_tf=False\_use\_idf =True»

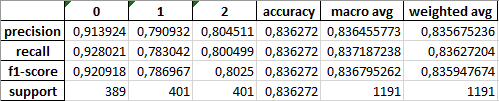


Рисунок 23 — Пример работы программы с параметрами «max\_features=500\_stop\_words=english\_use\_tf=True\_use\_idf =False»

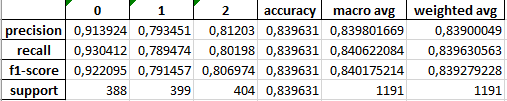


Рисунок 24 — Пример работы программы с параметрами «max\_features=500\_stop\_words=english\_use\_tf=True\_use\_idf =True»

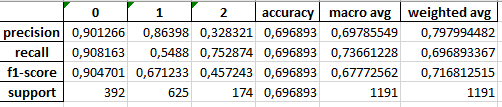


Рисунок 25 — Пример работы программы с параметрами «max\_features=500\_stop\_words=None\_use\_tf=False\_use\_idf =False»

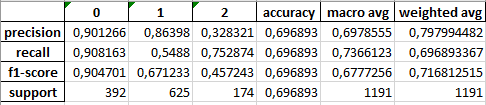


Рисунок 26 — Пример работы программы с параметрами «max\_features=500\_stop\_words=None\_use\_tf=False\_use\_idf =True»

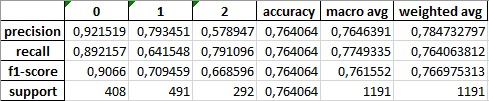


Рисунок 27 — Пример работы программы с параметрами «max\_features=500\_stop\_words=None\_use\_tf=True\_use\_idf =False»

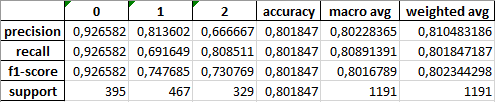


Рисунок 28 — Пример работы программы с параметрами «max\_features=500\_stop\_words=None\_use\_tf=True\_use\_idf =True»

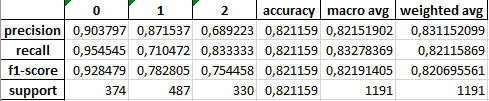


Рисунок 30 — Пример работы программы с параметрами «max\_features=1000\_stop\_words=english\_use\_tf=False\_use\_idf =False»

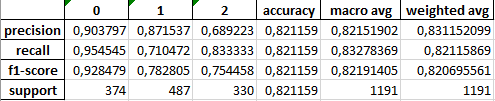


Рисунок 31 — Пример работы программы с параметрами «max\_features=1000\_stop\_words=english\_use\_tf=False\_use\_idf =True»

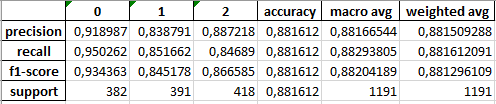


Рисунок 32 — Пример работы программы с параметрами «max\_features=1000\_stop\_words=english\_use\_tf=True\_use\_idf =False»

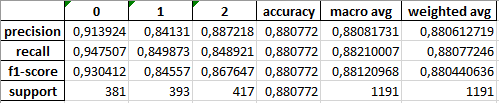


Рисунок 33 — Пример работы программы с параметрами «max\_features=1000\_stop\_words=english\_use\_tf=True\_use\_idf =True»

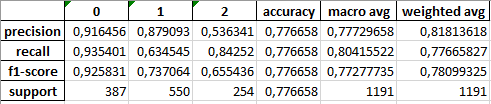


Рисунок 34 — Пример работы программы с параметрами «max\_features=1000\_stop\_words=None\_use\_tf=False\_use\_idf =False»

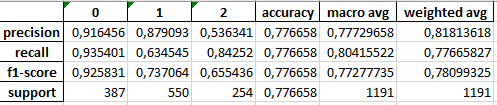


Рисунок 35 — Пример работы программы с параметрами «max\_features=1000\_stop\_words=None\_use\_tf=False\_use\_idf =True»

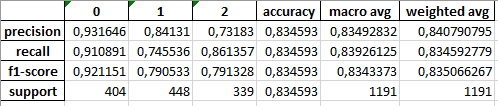


Рисунок 36 — Пример работы программы с параметрами «max\_features=1000\_stop\_words=None\_use\_tf=True\_use\_idf =False»

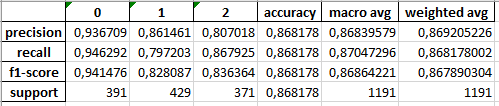


Рисунок 37 — Пример работы программы с параметрами «max\_features=1000\_stop\_words=None\_use\_tf=True\_use\_idf =True»

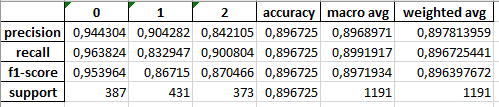


Рисунок 38 — Пример работы программы с параметрами «max\_features=5000\_stop\_words=english\_use\_tf=False\_use\_idf =False»

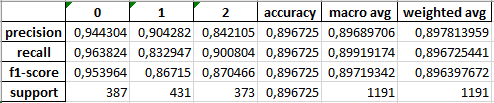


Рисунок 39 — Пример работы программы с параметрами «max\_features=5000\_stop\_words=english\_use\_tf=False\_use\_idf =True»

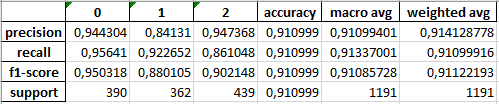


Рисунок 40 — Пример работы программы с параметрами «max\_features=5000\_stop\_words=english\_use\_tf=True\_use\_idf =False»

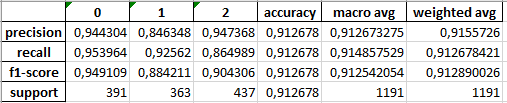


Рисунок 41 — Пример работы программы с параметрами «max\_features=5000\_stop\_words=english\_use\_tf=True\_use\_idf =True»

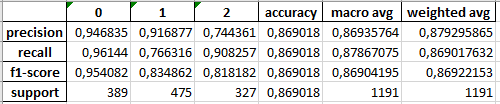


Рисунок 42 — Пример работы программы с параметрами «max\_features=5000\_stop\_words=None\_use\_tf=False\_use\_idf =False»

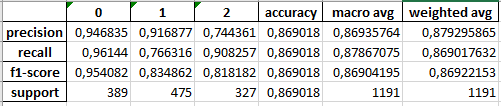


Рисунок 43 — Пример работы программы с параметрами «max\_features=5000\_stop\_words=None\_use\_tf=False\_use\_idf =True»

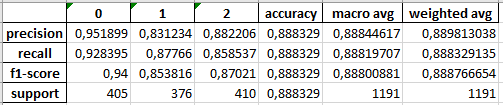


Рисунок 44 — Пример работы программы с параметрами «max\_features=5000\_stop\_words=None\_use\_tf=True\_use\_idf =False»

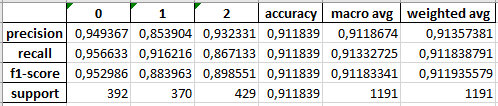


Рисунок 45 — Пример работы программы с параметрами «max\_features=5000\_stop\_words=None\_use\_tf=True\_use\_idf =True»

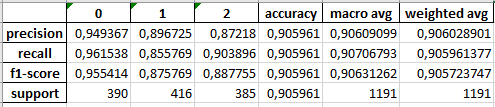


Рисунок 46 — Пример работы программы с параметрами «max\_features=10000\_stop\_words=english\_use\_tf=False\_use\_idf =False»

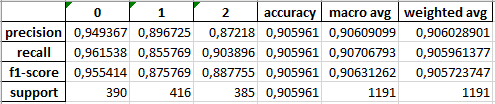


Рисунок 47 — Пример работы программы с параметрами «max\_features=10000\_stop\_words=english\_use\_tf=False\_use\_idf =True»

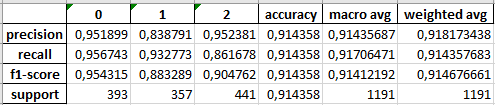


Рисунок 48 — Пример работы программы с параметрами «max\_features=10000\_stop\_words=english\_use\_tf=True\_use\_idf =False»

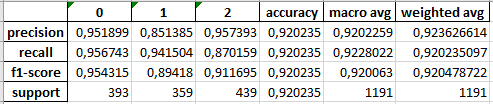


Рисунок 49 — Пример работы программы с параметрами «max\_features=10000\_stop\_words=english\_use\_tf=True\_use\_idf =True»

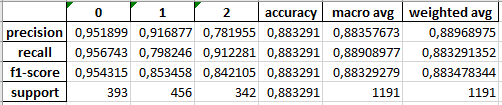


Рисунок 50 — Пример работы программы с параметрами «max\_features=10000\_stop\_words=None\_use\_tf=False\_use\_idf =False»

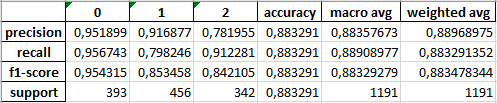


Рисунок 51 — Пример работы программы с параметрами «max\_features=10000\_stop\_words=None\_use\_tf=False\_use\_idf =True»

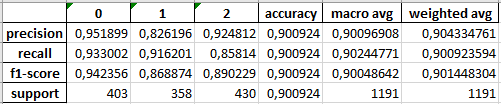


Рисунок 52 — Пример работы программы с параметрами «max\_features=10000\_stop\_words=None\_use\_tf=True\_use\_idf =False»

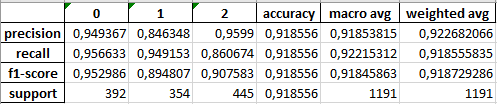


Рисунок 53 — Пример работы программы с параметрами «max\_features=10000\_stop\_words=None\_use\_tf=True\_use\_idf =True»

По результатам классификации наиболее подходящей является предварительная обработка данных со следующими параметрами:

* max\_features=10000
* со стоп словами
* с TF
* с IDF

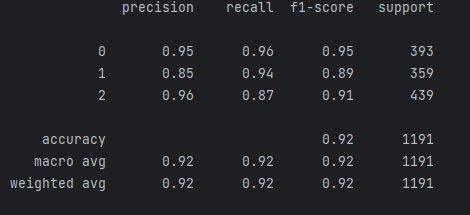


Рисунок 54 — Результат работы программы

Приложение А

Код программы

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import itertools  
import os  
import ssl  
import openpyxl  
import nltk  
  
nltk.download('punkt')  
  
from sklearn.metrics import classification\_report  
from sklearn.model\_selection import train\_test\_split, GridSearchCV  
from sklearn.datasets import fetch\_20newsgroups  
from sklearn.feature\_extraction.text import CountVectorizer  
from sklearn.feature\_extraction.text import TfidfTransformer  
from sklearn.pipeline import Pipeline  
from sklearn.naive\_bayes import MultinomialNB  
from nltk.stem import \*  
from nltk import word\_tokenize  
  
ssl.\_create\_default\_https\_context = ssl.\_create\_unverified\_context  
  
cats = ['comp.windows.x', 'rec.sport.baseball', 'rec.sport.hockey']  
rm = ('headers', 'footers', 'quotes')  
  
parameters = {  
 'vect\_\_max\_features': (100, 500, 1000, 5000, 10000),  
 'vect\_\_stop\_words': ('english', None),  
 'tfidf\_\_use\_idf': (True, False),  
}  
  
text\_clf = Pipeline([  
 ('vect', CountVectorizer()),  
 ('tfidf', TfidfTransformer()),  
 ('clf', MultinomialNB())  
])  
  
  
def get\_data(categories, remove):  
 twenty\_train\_full = fetch\_20newsgroups(subset='train', categories=categories, remove=remove)  
 twenty\_test\_full = fetch\_20newsgroups(subset='test', categories=categories, remove=remove)  
  
 twenty\_train\_full = twenty\_train\_full.data  
 twenty\_test\_full = twenty\_test\_full.data  
  
 twenty\_train = dict()  
 twenty\_test = dict()  
  
 for category in categories:  
 twenty\_train[category] = fetch\_20newsgroups(subset='train', categories=[category], remove=remove)  
 twenty\_test[category] = fetch\_20newsgroups(subset='test', categories=[category], remove=remove)  
 twenty\_train[category] = twenty\_train[category].data  
 twenty\_test[category] = twenty\_test[category].data  
  
 twenty\_train['full'] = twenty\_train\_full  
 twenty\_test['full'] = twenty\_test\_full  
 print('Done!')  
  
 return twenty\_train, twenty\_train\_full, twenty\_test, twenty\_test\_full  
  
  
def stemming(data):  
 porter\_stemmer = PorterStemmer()  
 stem = []  
 for text in data:  
 nltk\_tokens = word\_tokenize(text)  
 line = ''  
 for word in nltk\_tokens:  
 line += ' ' + porter\_stemmer.stem(word)  
 stem.append(line)  
 return stem  
  
  
def sort\_by\_tf(input\_str):  
 return input\_str[1]  
  
  
def top\_list(vect, data, count):  
 x = list(zip(vect.get\_feature\_names\_out(), np.ravel(data.sum(axis=0))))  
 x.sort(key=sort\_by\_tf, reverse=True)  
 return x[:count]  
  
  
def fill\_frame(frame):  
 pass  
  
  
def process(train, categories):  
 cats = categories[:]  
 cats.append('full')  
  
 mux = pd.MultiIndex.from\_product([['Count', 'TF', 'TF-IDF'], ['Без стоп-слов', 'Со стоп-словами']])  
 summary = dict()  
  
 for category in cats:  
 summary[category] = pd.DataFrame(columns=mux)  
  
 stop\_words = [None, 'english']  
 idf = [False, True]  
  
 index\_stop = {  
 'english': 'Без стоп-слов',  
 None: 'Со стоп-cловами'  
 }  
  
 index\_tf = {  
 False: 'TF',  
 True: 'TF-IDF'  
 }  
  
 for category in cats:  
 for stop in stop\_words:  
 vect = CountVectorizer(max\_features=10000, stop\_words=stop)  
 vect.fit(train[category])  
 train\_data = vect.transform(train[category])  
 summary[category]['Count', index\_stop[stop]] = top\_list(vect, train\_data, 20)  
  
 for tf in idf:  
 tfidf = TfidfTransformer(use\_idf=tf).fit(train\_data)  
 train\_fidf = tfidf.transform(train\_data)  
 summary[category][index\_tf[tf], index\_stop[stop]] = top\_list(vect, train\_fidf, 20)  
  
 return summary  
  
  
def print\_classification\_score(clf, data):  
 print(classification\_report(clf.predict(data.data), data.target))  
  
  
def prespocess(data, max\_features, stop\_words, use\_tf, use\_idf):  
 tf = None  
 cv = CountVectorizer(max\_features=max\_features, stop\_words=stop\_words).fit(data)  
 if use\_tf:  
 tf = TfidfTransformer(use\_idf=use\_idf).fit(cv.transform(data))  
 return cv, tf  
  
  
def models\_grid\_search(data\_train, data\_test):  
 max\_features = [100, 500, 1000, 5000, 10000]  
 stop\_words = ['english', None]  
 use\_tf = [True, False]  
 use\_idf = [True, False]  
 res = dict()  
 for param in itertools.product(max\_features, stop\_words, use\_tf, use\_idf):  
 cv, tf = prespocess(data\_train.data, param[0], param[1], param[2], param[3])  
 if tf:  
 clf = MultinomialNB().fit(tf.transform(cv.transform(data\_train.data)), data\_train.target)  
 prep\_test = tf.transform(cv.transform(data\_test.data))  
 else:  
 clf = MultinomialNB().fit(cv.transform(data\_train.data), data\_train.target)  
 prep\_test = cv.transform(data\_test.data)  
  
 name = f'max\_features={param[0]}\_stop\_words={param[1]}\_use\_tf={param[2]}\_use\_idf ={param[3]}'  
 res[name] = pd.DataFrame(classification\_report(clf.predict(prep\_test), data\_test.target, output\_dict=True))  
 return res  
  
  
twenty\_train, twenty\_train\_full, twenty\_test, twenty\_test\_full = get\_data(cats, rm)  
  
stem\_train = dict()  
stem\_test = dict()  
  
for category in cats:  
 stem\_train[category] = stemming(twenty\_train[category])  
 stem\_test[category] = stemming(twenty\_test[category])  
  
stem\_train['full'] = stemming(twenty\_train['full'])  
stem\_test['full'] = stemming(twenty\_test['full'])  
  
summ\_without\_stem = process(twenty\_train, cats)  
summ\_with\_stem = process(stem\_train, cats)  
  
if not os.path.exists('summ\_without\_stem'):  
 os.makedirs('summ\_without\_stem')  
  
if not os.path.exists('summ\_with\_stem'):  
 os.makedirs('summ\_with\_stem')  
  
for cat in ['full'] + cats:  
 summ\_without\_stem[cat].to\_excel('summ\_without\_stem/' + cat + '.xlsx')  
 summ\_with\_stem[cat].to\_excel('summ\_with\_stem/' + cat + '.xlsx')  
  
print('Well Done!')  
  
# Pipelines  
  
twenty\_train\_full = fetch\_20newsgroups(subset='train', categories=cats, remove=rm)  
twenty\_test\_full = fetch\_20newsgroups(subset='test', categories=cats, remove=rm)  
  
scores = models\_grid\_search(twenty\_train\_full, twenty\_test\_full)  
if not os.path.exists('scores'):  
 os.makedirs('scores')  
  
print(len(scores.items()))  
  
for name, score in scores.items():  
 print(name)  
 score.to\_excel('scores/' + name + '.xlsx')  
  
gs\_clf = GridSearchCV(text\_clf, parameters, n\_jobs=-1, cv=3)  
gs\_clf.fit(X=twenty\_train\_full.data, y=twenty\_train\_full.target)  
print\_classification\_score(gs\_clf, twenty\_test\_full)