**Липецкий государственный технический университет**

Факультет Автоматизации и Информатики

Кафедра Автоматизированных систем управления

Лабораторная работа №2

по прикладным интеллектуальным системам и экспертным системам

Предварительная обработка текстовых данных

Студент Ледовских Д.В.

Группа М-ИАП-22

Проверил

Доцент Кургасов В.В.

Липецк 2022г.

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

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 | Названия классов |
| 6 | 'alt.atheism', 'sci.space', 'soc.religion.christian' |

Ход работы

Для данной лабораторной работы нам необходимы библиотеки: pandas, numpy, matplotlib, NLTK, itertools, scikit-learn.

Pandas — программная библиотека на языке Python для обработки и анализа данных;

NumPy нужен для работы с большими многомерными массивами и матрицами, в данной библиотеке реализованы различные математические функции и операции с массивами данных;

Matplotlib разработан для визуализации данных в понятном пользователю виде за счет обширного функционала настроек итогового изображение, представленного двумерной или трехмерной графикой.

Scikit-learn – это один из наиболее широко используемых пакетов Python для Data Science и Machine Learning. Он содержит функции и алгоритмы для машинного обучения: классификации, прогнозирования или разбивки данных на группы;

NLTK — пакет библиотек и программ для символьной и статистической обработки естественного языка, написанных на языке программирования Python. Содержит графические представления и примеры данных;

Itertools стандартизирует основной набор быстрых эффективных по памяти инструментов, которые полезны сами по себе или в связке с другими инструментами.

Для выполнения лабораторной работы в среде Jupiter Notebook были импортированы следующие модули:

import pandas as pd

import numpy as np

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

from sklearn.metrics import accuracy\_score

from sklearn.pipeline import Pipeline

from sklearn.datasets import fetch\_20newsgroups

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

from sklearn.metrics import accuracy\_score

from sklearn.feature\_extraction.text import TfidfTransformer

from nltk.stem import \*

from nltk import word\_tokenize

import os

import itertools

Разбиение данных на обучающую (train) и тестовую (test) выборки выполнено следующим способом:

categories = ['alt.atheism', 'sci.space', 'soc.religion.christian']

remove = ('headers', 'footers', 'quotes')

t\_train = fetch\_20newsgroups(subset='train', shuffle=True, random\_state=2, categories = categories, remove = remove)

t\_test = fetch\_20newsgroups(subset='test', shuffle=True, random\_state=2, categories = categories, remove = remove)

twenty\_train = dict()

twenty\_test = dict()

for c in categories:

twenty\_train[c] = fetch\_20newsgroups(subset='train', shuffle=True, random\_state=2, categories = [c], remove = remove)

twenty\_test[c] = fetch\_20newsgroups(subset='test', shuffle=True, random\_state=2, categories = [c], remove = remove)

twenty\_train[c] = twenty\_train[c].data

twenty\_test[c] = twenty\_test[c].data

twenty\_train['full'] = t\_train.data

twenty\_test['full'] = t\_test.data,

где twenty\_train данные для обучения, twenty\_test данные для тестирования.

Вывод на экран по одному документа каждого класса:

print (twenty\_train['alt.atheism'][0], "\n----------------")

print (twenty\_train['sci.space'][0], "\n----------------")

print (twenty\_train['soc.religion.christian'][0], "\n----------------")

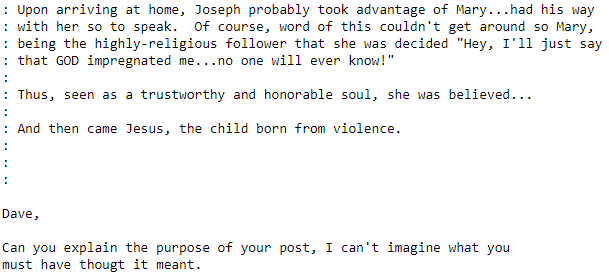


Рисунок 1 – вывод документа первого класса

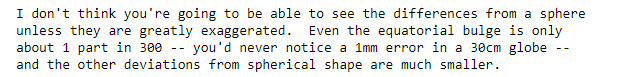


Рисунок 2 – вывод документа второго класса

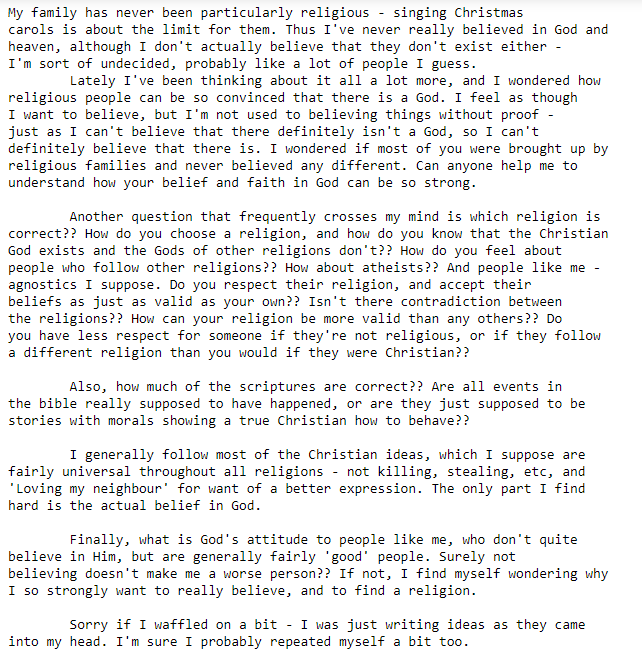


Рисунок 3 – вывод документа третьего класса

Функция стемминга реализована следующим образом:

def stem(text):

porter\_stemmer = PorterStemmer()

result = []

for t in text:

nltk\_tokens = word\_tokenize(t)

line = ''

for word in nltk\_tokens:

line += ' ' + porter\_stemmer.stem(word)

result.append(line)

return steresultm

Применение стеминга:

train\_s = dict()

test\_s = dict()

for с in categories:

train\_s[с] = stem(twenty\_train[с])

test\_s[с] = stem(twenty\_test[с])

train\_s['full'] = stem(twenty\_train['full'])

test\_s['full'] = stem(twenty\_train['full'])

Проведем векторизацию выборки по пунктам необходимыми способами, а именно:

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.

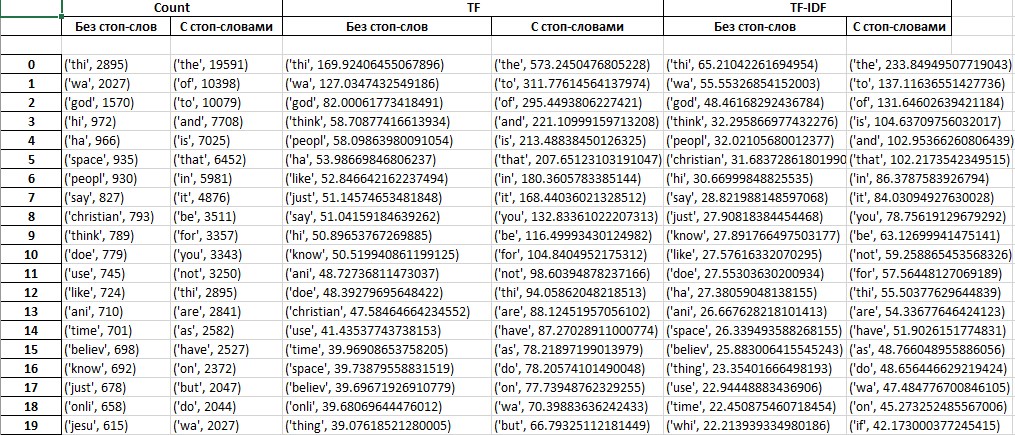


Рисунок 4 – Со стеммингом для всех категорий

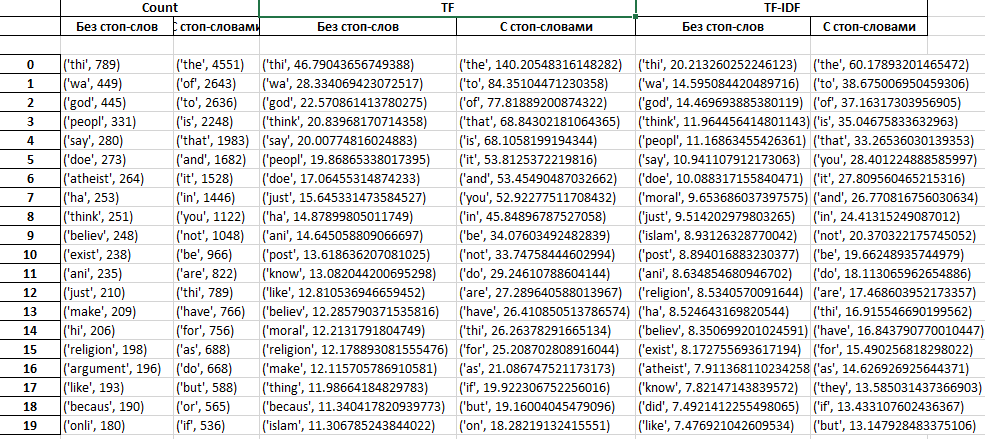


Рисунок 5 – Со стеммингом для alt.atheism

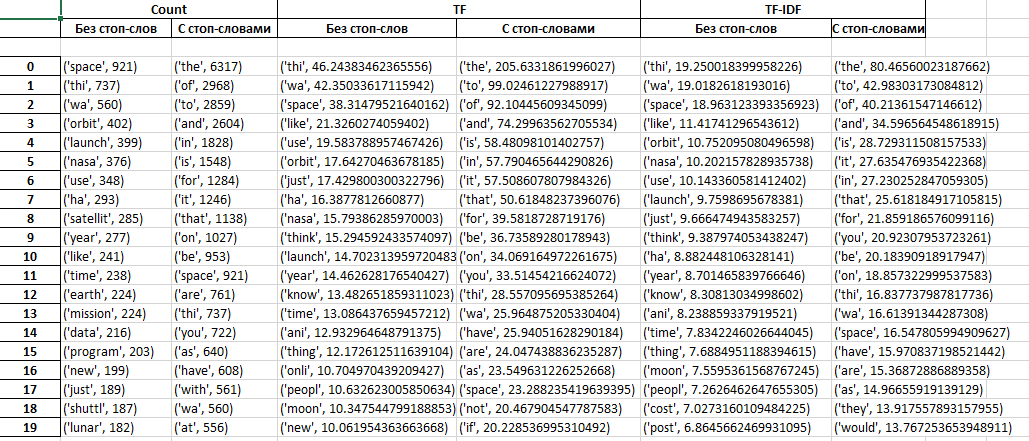


Рисунок 6 – Со стеммингом для sci.space

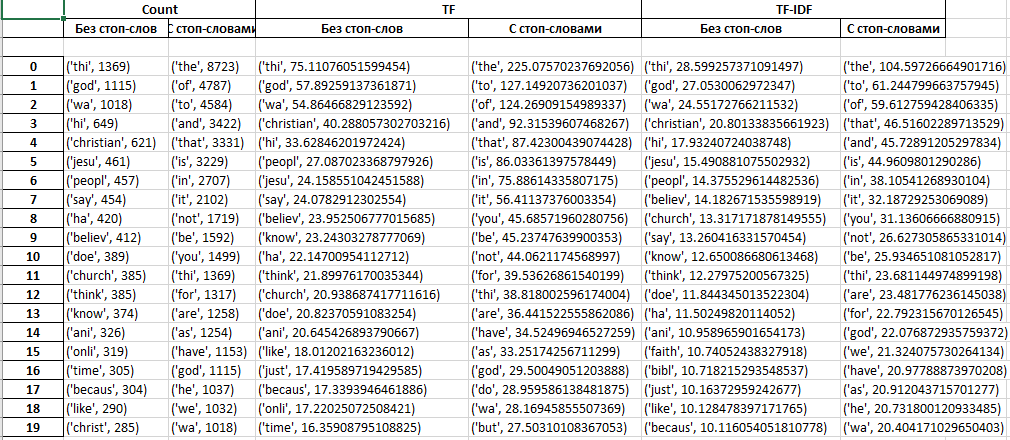


Рисунок 7 – Со стеммингом для soc.religion.christian

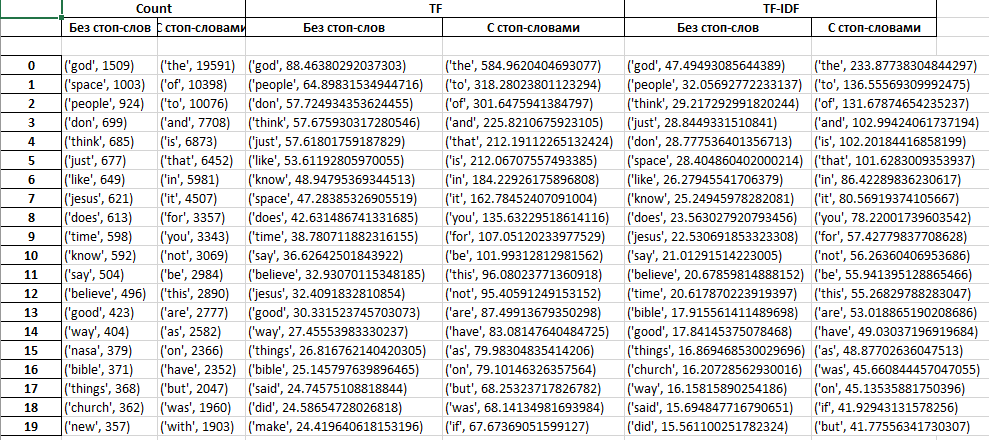


Рисунок 8 – Без стемминга для всех категорий

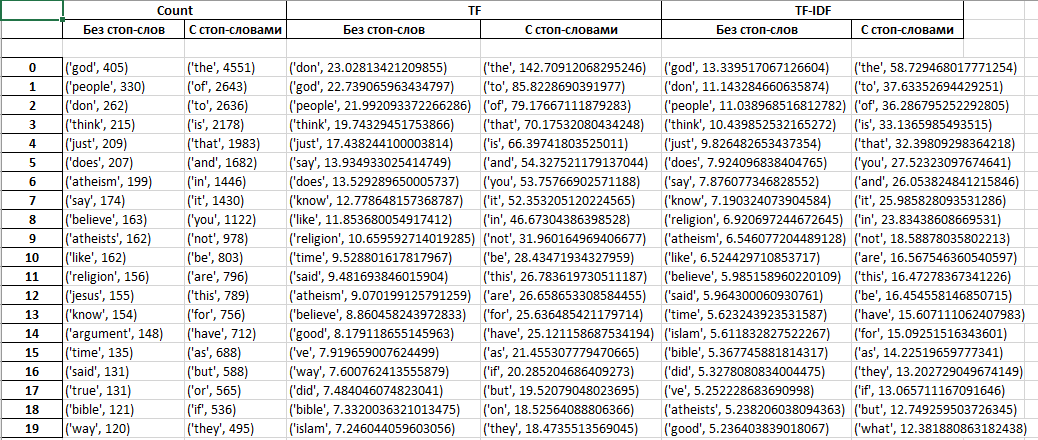


Рисунок 9 – Без стемминга для alt.atheism

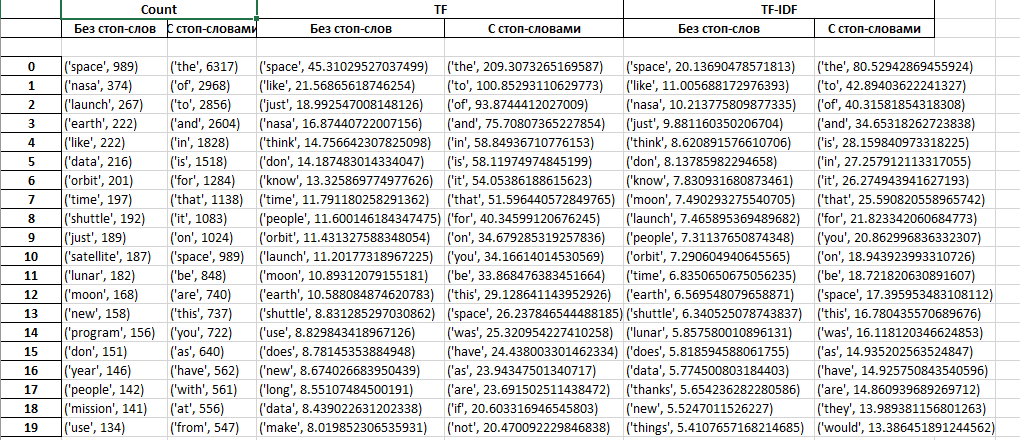


Рисунок 10 – Без стемминга для sci.space

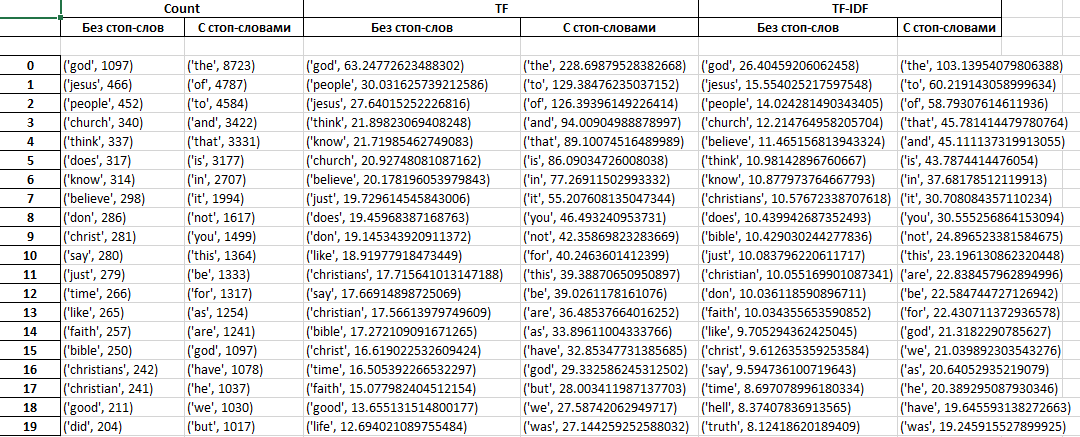


Рисунок 11– Без стемминга для soc.religion.christian

Реализуем модель Наивного Байесовского классификатора с помощью конвейера Pipeline для получения ключевых показателей качества, а именно: значения полноты, точности, f1-меры и аккуратности. Исследуем следующие характеристики:

• Отсечение – не отсечение стоп-слов

• Количество информативных терминов (max\_features)

• Взвешивание: Count, TF, TF-IDF

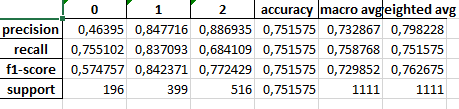


Рисунок 12 – Пример работы программы со следующими параметрами (max\_features = 1000, со стоп словами, TF, без TF-IDF)

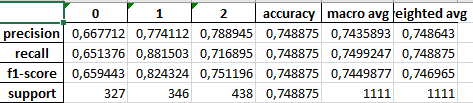


Рисунок 13 – Пример работы программы со следующими параметрами (max\_features = 1000, без стоп слов, без TF, без TF-IDF)

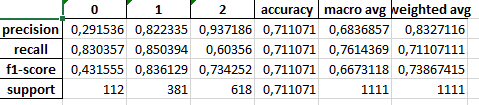


Рисунок 14 – Пример работы программы со следующими параметрами (max\_features = 1000, без стоп слов, TF, без TF-IDF)

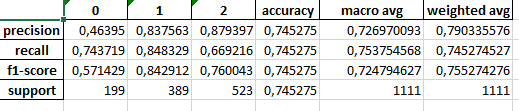


Рисунок 15 – Пример работы программы со следующими параметрами (max\_features = 1000, без стоп слов, TF, TF-IDF)

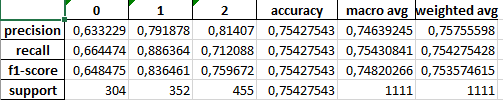


Рисунок 16 – Пример работы программы со следующими параметрами (max\_features = 1000, без стоп слов, без TF, TF-IDF)

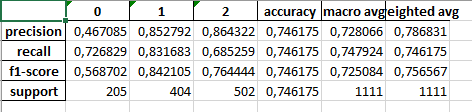


Рисунок 17 – Пример работы программы со следующими параметрами (max\_features = 1000, со стоп словами, TF, TF-IDF)

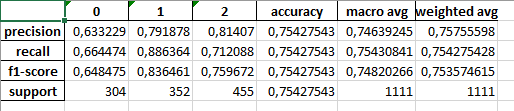


Рисунок 18 – Пример работы программы со следующими параметрами (max\_features = 1000, со стоп словами, без TF, без TF-IDF)

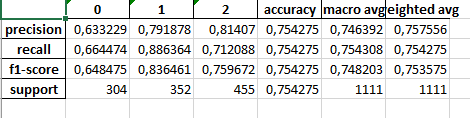


Рисунок 19 – Пример работы программы со следующими параметрами (max\_features = 1000, со стоп словами, без TF, TF-IDF)

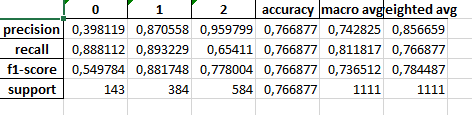


Рисунок 20 – Пример работы программы со следующими параметрами (max\_features = 5000, со стоп словами, TF, без TF-IDF)

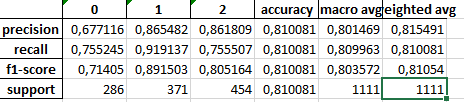


Рисунок 21 – Пример работы программы со следующими параметрами (max\_features = 5000, без стоп слов, без TF, без TF-IDF)

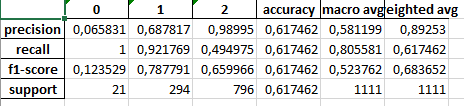


Рисунок 22 – Пример работы программы со следующими параметрами (max\_features = 5000, без стоп слов, TF, без TF-IDF)

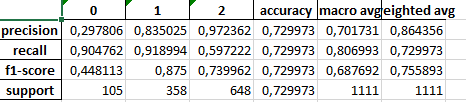


Рисунок 23 – Пример работы программы со следующими параметрами (max\_features = 5000, без стоп слов, TF, TF-IDF)

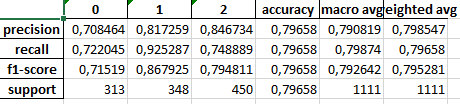


Рисунок 24 – Пример работы программы со следующими параметрами (max\_features = 5000, без стоп слов, без TF, TF-IDF)

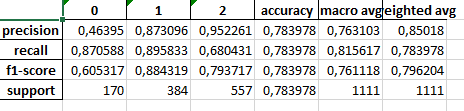


Рисунок 25 – Пример работы программы со следующими параметрами (max\_features = 5000, со стоп словами, TF, TF-IDF)

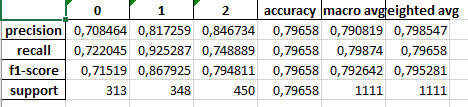


Рисунок 26 – Пример работы программы со следующими параметрами (max\_features = 5000, со стоп словами, без TF, без TF-IDF)

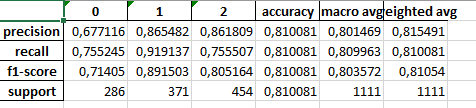


Рисунок 27 – Пример работы программы со следующими параметрами (max\_features = 5000, со стоп словами, без TF, TF-IDF)

По результатам классификации наиболее подходящие предварительные обработки данных являются обработки данных с параметрами (max\_features = 10000, без стоп слов, без TF, TF-IDF) и (max\_features = 10000, без стоп слов, без TF, без TF-IDF)

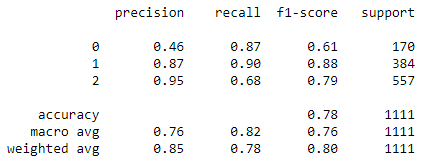


Рисунок 29 – Результат работы программы

Вывод

В ходе лабораторной работы были получены практические навыки обработки текстовых данных в среде Jupiter Notebook. Я научился проводить предварительную обработку текстовых данных и выявлять параметры обработки, позволяющие добиться наилучшей точности классификации.