|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**Отчет**

**по лабораторной работе № 4**

**Вариант 3**

**Название**: Автоматическая обработка текстов

**Дисциплина:** Прикладной анализ данных

Студент гр. ИУ6-51Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** К. А. Татаренко

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** М. А. Кулаев

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

**Цель работы** ­– изучение методов предобработки текстовых документов (очистки), применение лемматизации и стемминга к тексту, построение модели на основе tf-idf, сравнение полученных результатов.

1. **Оставьте в выборках только строки с классами positive и negative.**



Рисунок 1 ­– Код получения определенных строк и определения процентного соотношения

Было оставлены строки, которые помечены positive или negative. 4 набора (2 тестовых и 2 обучающи) из-за того, что если сделать один и второй равный этому, то второй сохрнаится как ссылка на 1 и все изменения применятся и ко второму, поэтому было создано 2 обучающих – для задачи лемматизации и стемминга (аналогично и для тестовых).

Можем заметить, что процент positive и negative строк примерно одинаковое (примерно сбалансированное) – 42 на 58 %. Это значит, что мы **можем применять метрику accuracy** для оценки качества модели.

1. **Определите и реализуйте креативные методы очистки набора данных. Например, в твитах часто встречаются ссылки на аккаунты других пользователей, оформленные однотипным образом – кажется, что это лишняя информация.**

Была составлена функция очистки данных, в которой осуществлено:

* Приведение к нижнему регистру
* Обработка твита (удаление ссылок, упоминаний других пользователей, хэштегов(так как после # словапишутсяподряд)
* Удаление знаков препинания (кроме ! и ? знаков, так как они могут влиять на окрас эмоциональный окрас твита)
* Удаление стоп-слов (полученных из библиотеки)
* Удаление цифр (например, дат), так как они не вносят никакой эмоциональной состовляющей
* Удаление подряд идущих символов, не считая согласных (согласные не берем, так как в русском языке есть слова с 2 подряд согласными, удаляем подобное – яяяя->я, тыыыы->ты, !!!!->!, 🤧 🤧 🤧-> 🤧)
* Отделение смайликов от слов (было замечено, что люди пишут смайлики сразу после слов, не отделяя их пробелами: чимина🤧 -> чимина 🤧)



Рисунок 2 – Самописная функция предобработки текста

1. **Осуществите стемминг подготовленного набора данных и преобразуйте каждый твит в мешок слов. Помните, что кастомные преобразования обучаются только на train выборке. Если они необучаемые, то нужно взять один и тот же тип преобразования для обеих выборок (один и тот же метод из одной библиотеки).**



Рисунок 3 – Стемминг текста, сбор всех строк в массив

Стемминг данных осуществляется за счет применения медота stem из библиотеки SnowBallStemmer. Далее все записи записываются в один массив, где каждый элемент ­– отдельный твит.

1. **Составьте Count-матрицу и рассчитайте на ней tf-idf. Обратите внимание, что tf-idf – это обучаемое преобразование, которое нужно зафитить на обучающих данных и применить затем к тестовым.**

Получим tf-idf с помощью библиотеки sklearn, а именно с помощью объекта TfidVectorizer. TfidfVectorizer преобразует коллекцию необработанных документов в матрицу функций TF-IDF.

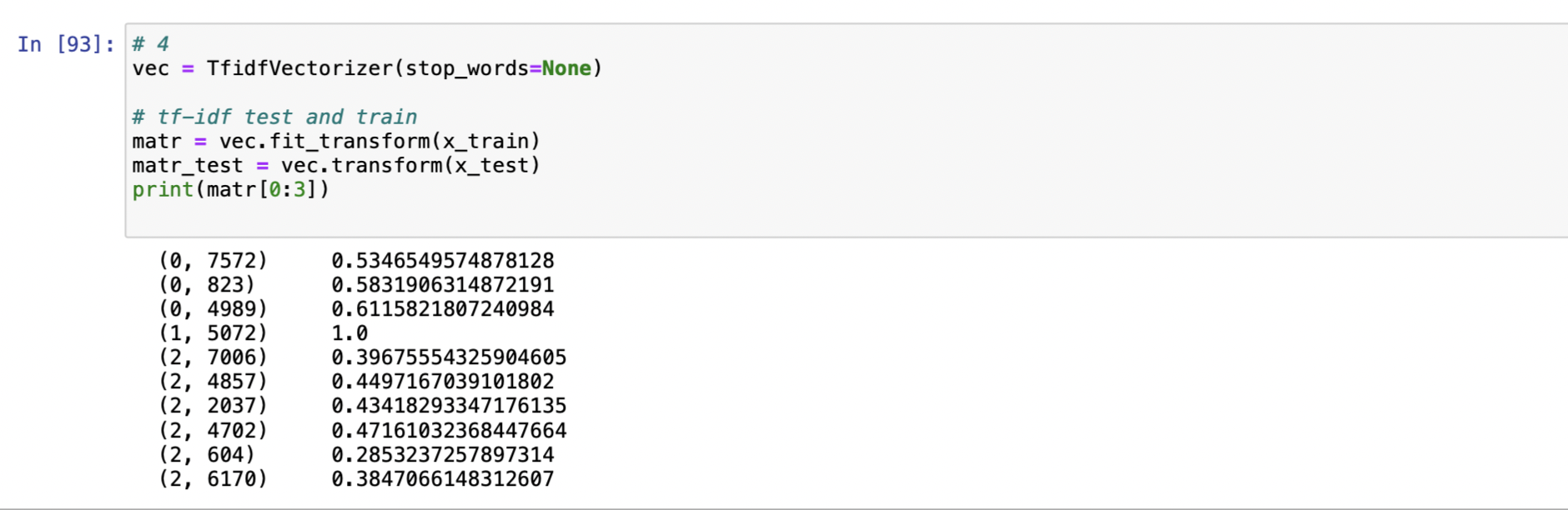


Рисунок 4 – Получение tf-idf

1. **Обучите модели логистической регрессии и случайного леса на обучающей выборке, примените их к тестовым данным. Посчитайте качество на обучающих и тестовых данных, сравните результаты. Определите наиболее важные признаки (слова).**

При первоначальном решении (при значениях параметров по умолчанию) модель была сильна переобучена, что видно на рисунке



Рисунок 5 – Переобученная модель

По этой причине был произведен перебор параметра регуляризации (для логистической регрессии) и максимальной глубины дерева (для случайного леса) для того, чтобы точность моделей на тестовых и обучающих данных была примерно равна.



Рисунок ­6 – Полученные показатели модели после подбора параметра регуляризации логистической регрессии

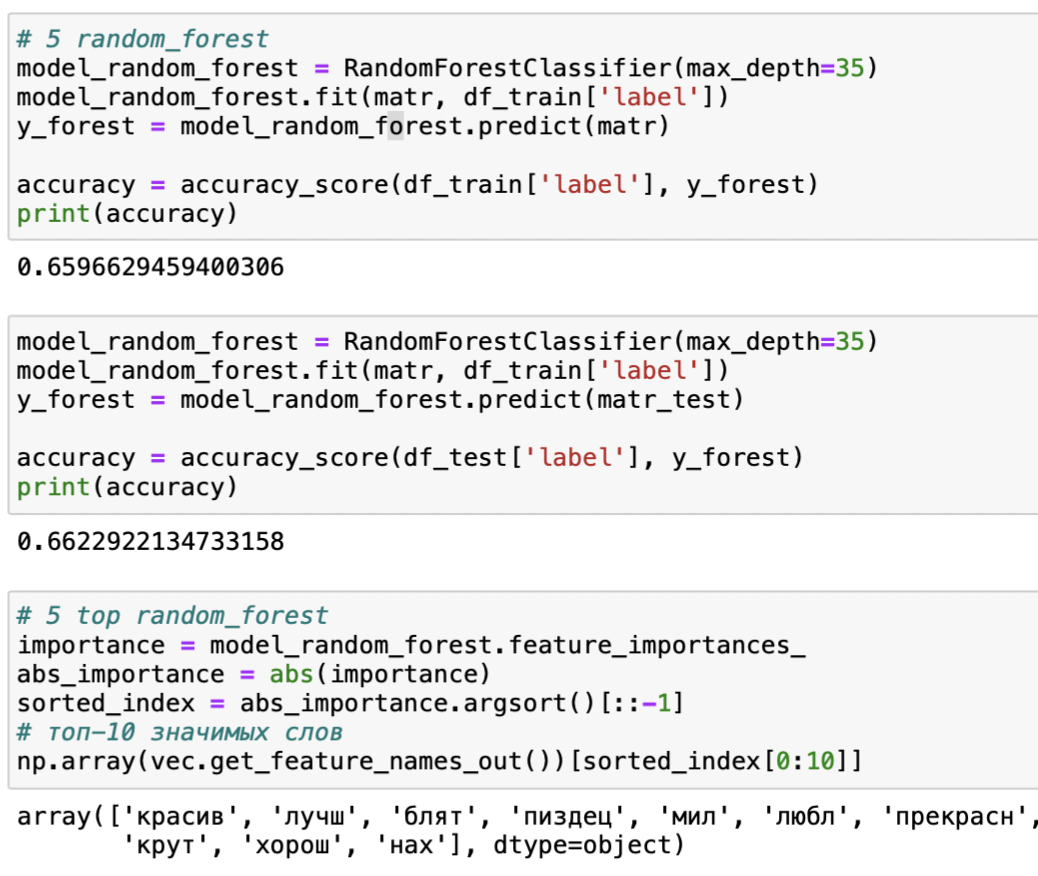


Рисунок ­6 – Полученные показатели модели после подбора параметра максимальной глубины случайного леса

1. **В пункте 3 вместо стемминга осуществите лемматизацию и проделайте пункты 3-5 с учетом другого типа подготовки данных.**

Лемматизация осуществлялась с помощью библитеки pymorphy3, а именно модуля Mystem, который поддерживает контекстную лемматизацию.

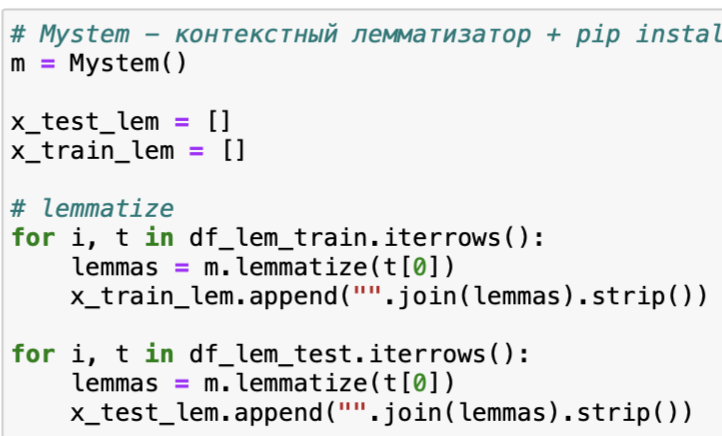


Рисунок 7 – Лемматизация текста

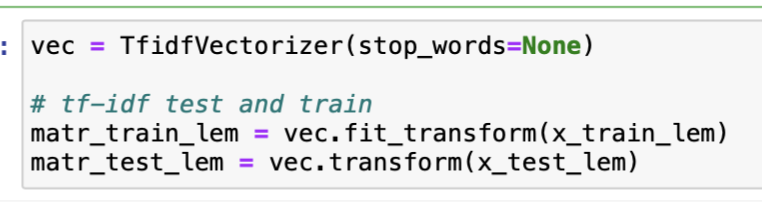


Рисунок ­8 – Получение tf-idf после лемматизации

По аналогии со стеммингом модель с параметрами по умолчанию переобучена, поэтому также подберем параметры для каждой модели.



Рисунок 9 – параметры модели после подбора параметра логистической регрессии при лемматизации



Рисунок 10 – Полученные показатели модели после подбора параметра максимальной глубины случайного леса при лемматизации

1. **Сравните результаты по качеству и по наиболее важным признакам (словам) между 2 обученными вариантами.**

Таблица 1 – Сравнение стемминга и лемматизации

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Стемминг | | Лемматизация | |
| Логистическая | Случаный лес | Логистическая | Случаный лес |
| Accuracy | 0.71 | 0.66 | 0.71 | 0.66 |
| Топ-10 слов | **'любл',**  **'красив',**  **'блят',**  **'пиздец',**  **'мил',**  'крут',  **'ва',**  'прекрасн',  'лучш',  **'хорош'** | **'красив',**  'лучш',  **'блят',**  **'пиздец',**  **'мил',**  **'любл',**  'прекрасн',  'крут',  **'хорош',**  **'нах'** | **'любить',**  **'блять',**  **'хороший',**  **'красивый',**  **'пиздец',**  **'вау',**  **'милый',**  'умирать',  'нравиться',  'сука' | **'любить',**  **'блять',**  **'хороший',**  **'красивый',**  'вау'**,**  **'пиздец',**  **'милый',**  'умирать'**,**  **'нахуй',**  'сука' |

Качество моделей (по метрике accuracy) абсолютно идентичное между лемматизацией и стеммингом. Если говорить про слова – то можно заметить, что в логичстической регрессии различие лишь в трех словах из первой десятки, причем с примерным сохранением порядка; в модели случайного леса – также различие лишь в трех словах, но порядок сохранился хуже.

Если же говорить о различие между самими моделями, то различие в полученной метрике незначительно (0.05).

Таким образом, можно сказать, что глобальных различий между лемматизацией и стеммингом не обнаружилось, а это могло произойти по следующим причинам:

* Не очень большой объем текста
* Неважность грамматической информации (для чего и нужно использовать лемматизацию)

**Вывод** – в ходе выполнения лабораторной работы я научился делать предобработку (очистку) текста, осуществлять лемматизацию, стемминг и нахождение tf-idf на языке python. На основе tf-idf получена модели логистической регрессии и случайного леса, которые в последствии были сравнены для 2 случаев.