**НИЯУ “МИФИ”**

**Индивидуальная работа**

по теме: “Классификация медицинских текстов на наличие побочных эффектов при помощи методов машинного обучения”

Группа: М19-117

Студент: Кайгородов Александр

Преподаватели: Сбоев А.Г. , Рыбка Р.Б.

Москва, 2020

**О соревновании   
“*Social Media Mining for Health Applications (#SMM4H) Shared Task 2020”***

Задания данного соревнования включают в себя задачи тематики NLP распространяющиеся на практические цели из сферы здравоохранения. Общая суть указанных заданий данного соревнования окружена вопросами извлечения информации из больших наборов твитов написанных пользователями в сети.

Тематика представленных наборов твитов склоняется к написанию отзыва о каком либо медицинском препарате, побочные эффекты которого нужно занести в базу данных для фармакологических служб. Правильное извлечение огромного числа выявленных побочных эффектов может значительно улучшить разработку новых лекарств, но проблема заключается в работе со сложной структурой информации представленной в виде естественного языка.

Для решения данной задачи можно прибегать к использованию машинного обучения, в частности нейронных сетей. Использование классических базовых методов МЛ допустимо, но использование глубинного обучения представляется наиболее приоритетным в данном случае, т.к позволяет получать большую точность при предсказании.

Основная группа задач данного соревнования сводится к следующим пунктам.  
1. Классификация твита на наличие или отсутствие побочного эффекта (ADR)

2. Выделение спана для конкретного побочного эффекта

3. Нормализация побочного эффекта (строгое формулирование)

4. Соотнесение полученного ADR с идентификатором базы данных MedDRA.

Мной (при активной моральной поддержке Романа Борисовича) были выбраны следующие два таска из соревнования:

*(названия и описания тасков оставляются в оригинальном виде)*

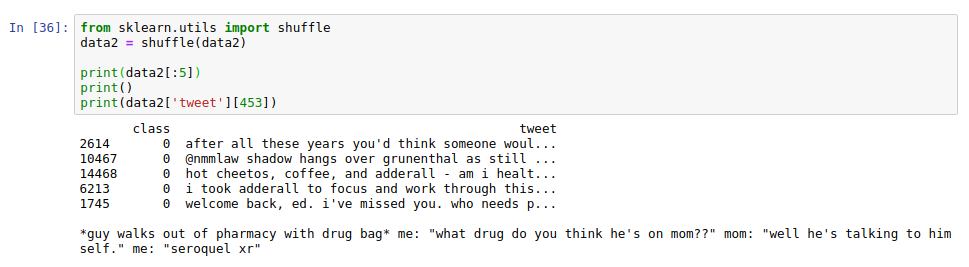
* Task 2: Automatic classification of multilingual tweets that report adverse effects
  + Training data: 25,672 tweets (2,374 “positive” tweets; 23,298 “negative” tweets)
  + Test data: ~5,000 tweets. (! в действительности полученных данных меньше)
* Task 3: Automatic extraction and normalization of adverse effects in English tweets
  + Training data: 2,376 tweets (1,212 “positive” tweets; 1,155 “negative” tweets).
  + Test data: ~1,000 tweets.
  + Evaluation metric: F1-score for the “positive” class (i.e., the correct AE spans and MedDRA IDs for tweets that report AEs.

Оригинальная ссылка на соревнование:  [#SMM4H Shared Task 2020](https://healthlanguageprocessing.org/smm4h-sharedtask-2020/)

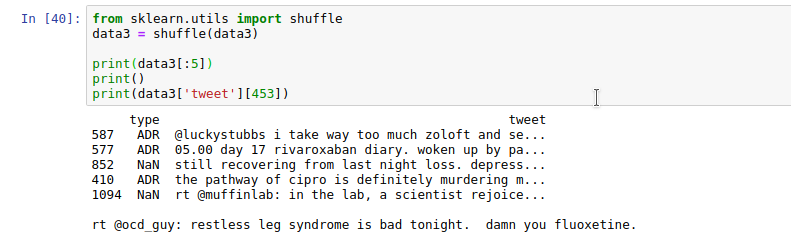
**Описание и визуализация представленных датасетов**

Описание состава используемых датасетов (данных) уже приведено в описании соревнования. В данном пункте будет представлен более детальный разбор имеющихся данных.

1. **Содержимое**

Data2 ( содержимое второго датасета)

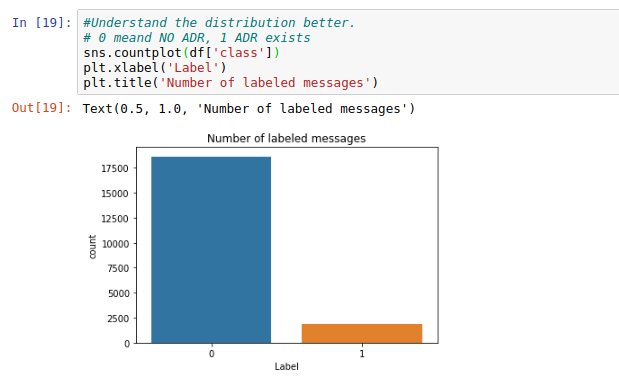
Data3 (содержимое третьего датасета)

Прошу заметить что во втором датасете класс твита представлен в численном виде: 0 - нет adr, 1 - есть adr; а тем временем в третьем датасете пометки следующие: NaN - нет adr, ADR - есть adr; Следует указать что в дальнейшем все будет приведено к единственному представлению как во втором датасете, для удобства обработки.

1. **Баланс классов**

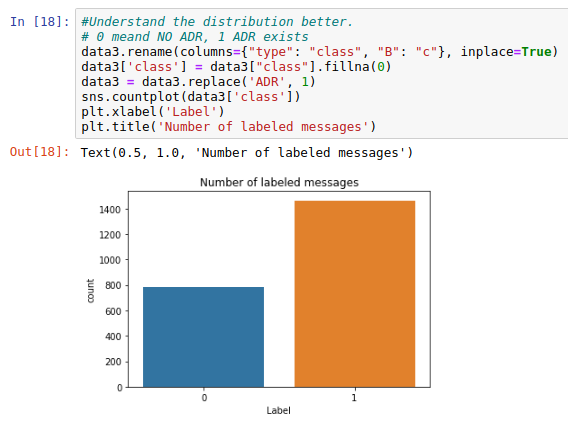
Факт дисбаланса данных является очень важным при обучении модели. Как и оказывается, такая проблема существует и в одном из действительных датасетов.

Анализ баланса данных провести очень легко, можно просто получить численное представление пропорции числа экземпляров классов или же просто представить числа экземпляров одного или другого класса.



Взглянув на эту пикчу не сложно понять что во втором датасете присутствует дисбаланс (довольно выраженный) в сторону меток об отсутствии adr в твите.

!. При построении модели это сыграет свою роль, а также будет важно это учитывать при ее оценке.

Выполняя то же самое и для другого набора данных (третий датасет) получаем следующую иллюстрацию.

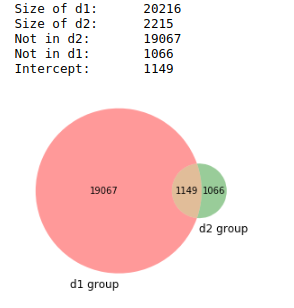
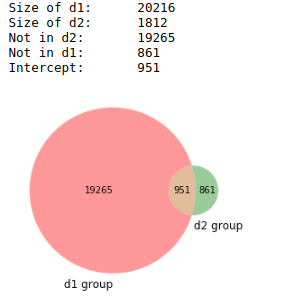
На этом представлении ситуация уже кажется повеселей, не такой выраженный дисбаланс данных.

Казалось бы что использование третьего датасета для классификации было бы лучше чем использование второго, но он обладает не самыми явными, но понятными недостатками относительно второго.

Конечно же стоит упомянуть что второй датасет используется для создания модели выделения спана adr, но если более качественно проанализировать данные (вполне можно и визуально), то можно заметить что тексты в нем повторяются с пометками на разные спаны. Следовательно, можно сделать вывод что не такой уж это и сбалансированный корпус твитов для нашей классификации. Также нужно учитывать объем датасетов ( = корпусов). Во втором датасете общее количество экземпляров равняется ~21к, в то время как в третьем ~2.3к. Это важно!

Как бонус, проведем анализ этих датасетов относительно друг друга, возможно получится найти что-то интересное.

*(на первой картинке data2 and data3\_train, на второй data2+(data3\_train+data3\_validation))*



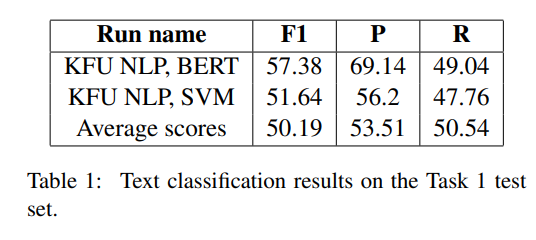
В принципе это было вполне ожидаемо, мы берем датасеты из одного соревнования, почему бы им еще и не пересекаться… Как видим на пикче, пресечение обоих множеств равняется 951 элементам, НО!, второй датасет (d1 group) не включается в себя некоторые элементы третьего (d2 group). Следовательно, можно немного схитрить и попробовать получить модель на слегка расширенном датасете, это может привести с небольшому выигрышу (не стоит забывать о повторении элементов в третьем датасете, поэтому при слиянии выполняем функцию *drop\_duplicates*() библиотеки Pandas)

**Опыт построения моделей предыдущих лет**

Организаторы предоставили ссылку на сборник статей с воркшопа предыдущего года на котором можно легко ознакомиться с работами других участников. Стоит понимать что по сути задания остались в той же тематике, но слегка изменили свои формулировки, так сказать мутировали, и все подходы и методы описанные там не распространяются на текущее соревнование. Тем не мене, работы опубликованные в этом сборнике могут оказаться весьма полезными для ознакомления с работающими подходами и при оценке свой модели относительно полученных равнее. Это даст лучшее представление об эффективности проделанной работы.

*LINK ->* [*WORKSHOP*](https://www.aclweb.org/anthology/W19-32.pdf)

Хоть и владея знаниями английского на достаточном уровне для понимания всех описанных работ (возможно, отсутствием только академических знаний) я все же выбрал работу участника гражданина РФ, [Елены Тутубалиной](https://www.aclweb.org/anthology/W19-3207.pdf).

В работе Елены приведены следующие оценки качества выполненной модели:

*(Task1 прошлого года = Task2 сегодняшнего года)*

Данные оценки взяты в качестве мерки разумности полученной моей модели.

**Поиск базового решения**

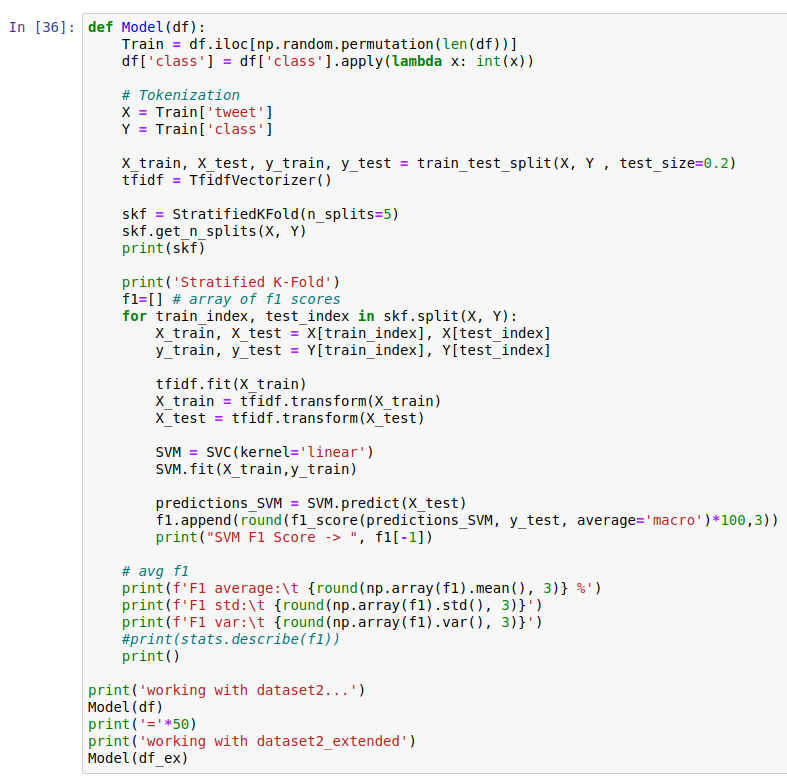
Это важная часть перед построением любой нейронной сети. Зачастую в задачах можно получить относительно простое и эффективное базовое решение которое уже будет удовлетворять всем требованиям заказчика. Для данных примеров были выбраны следующие модели: SVM, MLP, GBM. Вдобавок, к ним была попытка использовать TPOT.

TPOT - отличный интерфейс реализующий идеи AutoML. Данная библиотека самостоятельно осуществляет поиск подходящей модели с оптимизированными параметрами. Данных подход является очень универсальным и простым для программиста, но требует значительных вычислительных мощностей. Именно поэтому при использовании данного инструмента, запуск получения подходящей модели осуществлялся для третьего датасета (он на порядок меньше второго что значительно облегчает задачу).

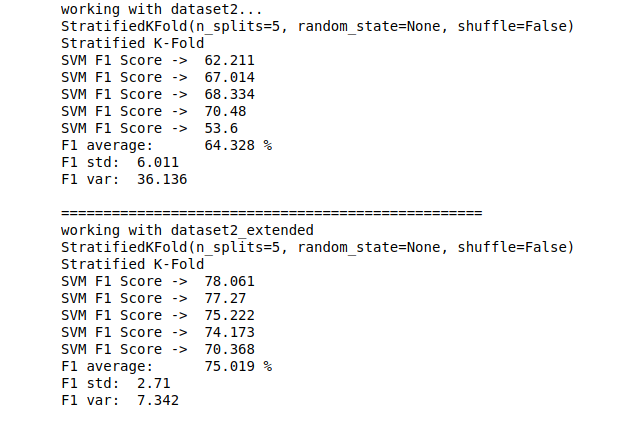
Необходимо упомянуть одну важную вещь, такую как векторизация текста. В качестве способов векторизовать текстовое представление были выбраны TFIDF и набор инструментов разработанный командой Facebook в лице FastText. Методы TFIDF были применены для всех базовых решений, в то время как FastText использовался только для модели SVM, которая мне показалась наиболее обещающей и перспективной из всех (без учета ТРОТ).

Далее приведены расчетные коды данных методов:

1. **TFIDF + SVM.**

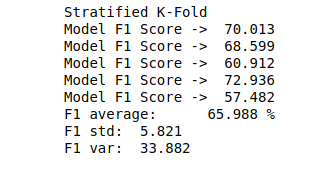
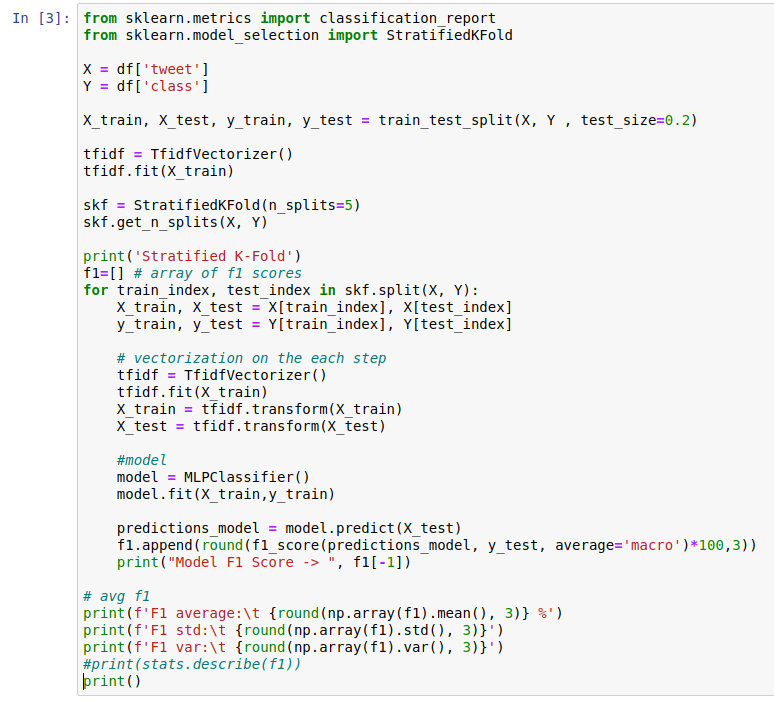


Данная модель применялась дважды, на чистом сете данных из второго датасета (df) и на расширенных данных второго и третьего датасета (df\_ex).

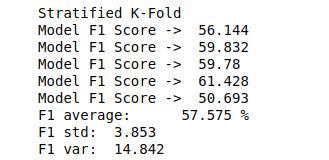
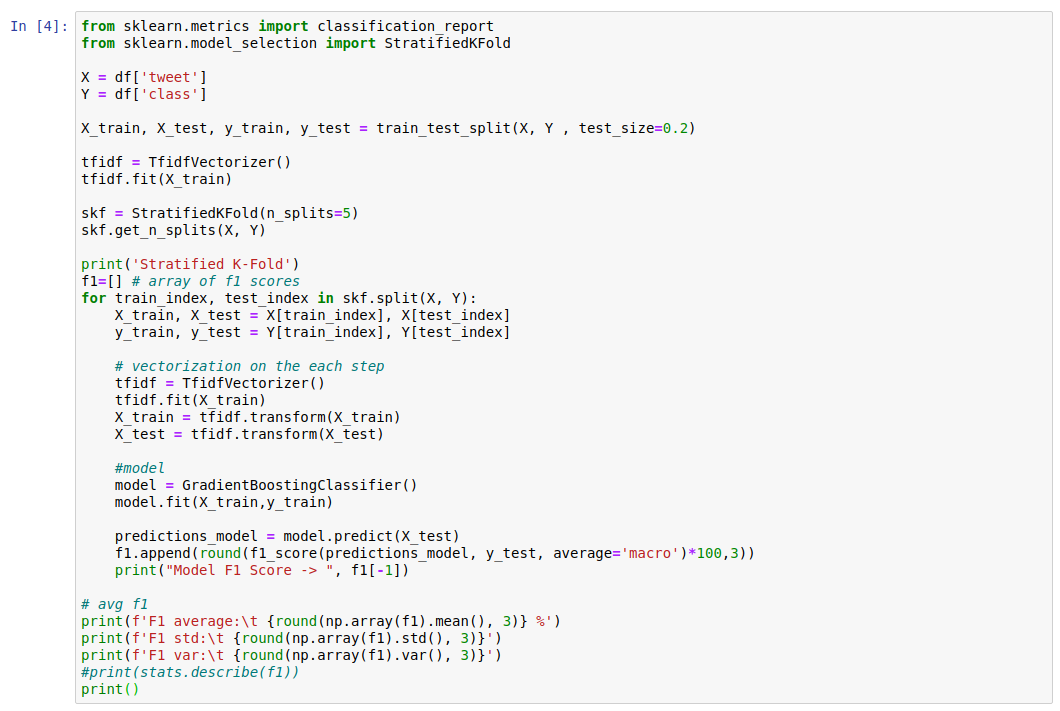
Вывод полученных результатов: 

Как видно на картинке, полученная точность на расширенных данных больше чем на обычном втором. Это не значит что все идеально, но четко говорит о важности объема данных для обучения. (Не исключено что была проведена валидация на примерах которые могли иметь место в тренировочной выборке)

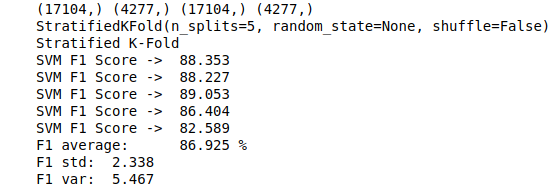
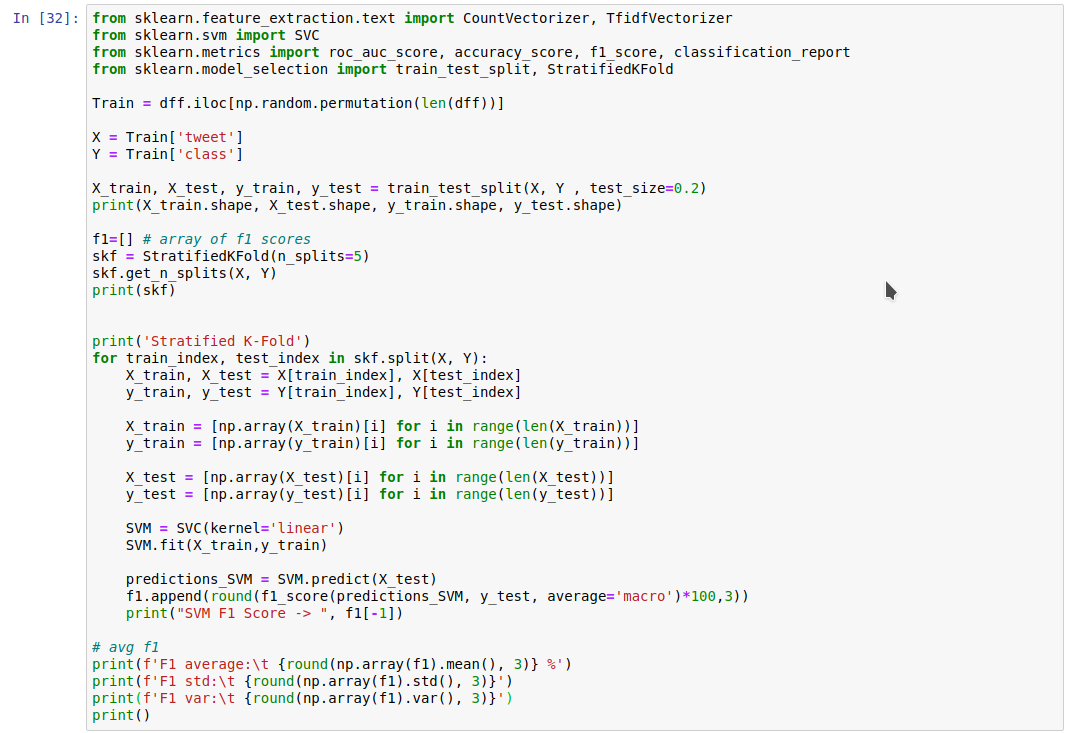
1. **TFIDF + MLP**

Реализация в коде + вывод, второй датасет:

1. **TFIDF + GBM**

Реализация в коде + вывод, второй датасет:

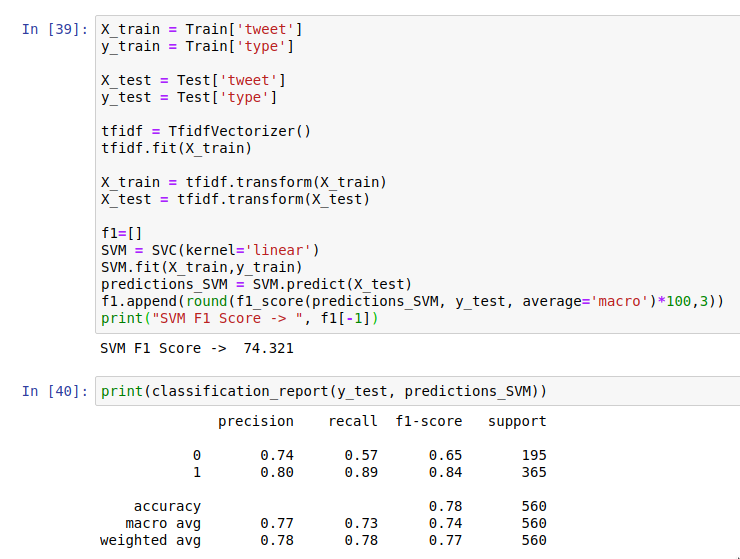
1. **FastText + SVM**

Реализация в коде + вывод, второй датасет:

Замечание: FastText+SVM получает точность предсказаний 84.5 процента, в то время как TFIDF+SVM получает точность предсказаний порядка 64.5 прцоента.

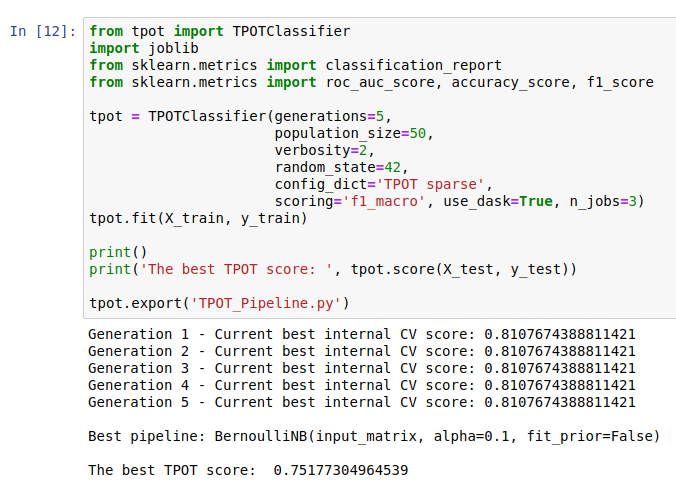
Это связано с “**не знаю с чем оно связано”**.

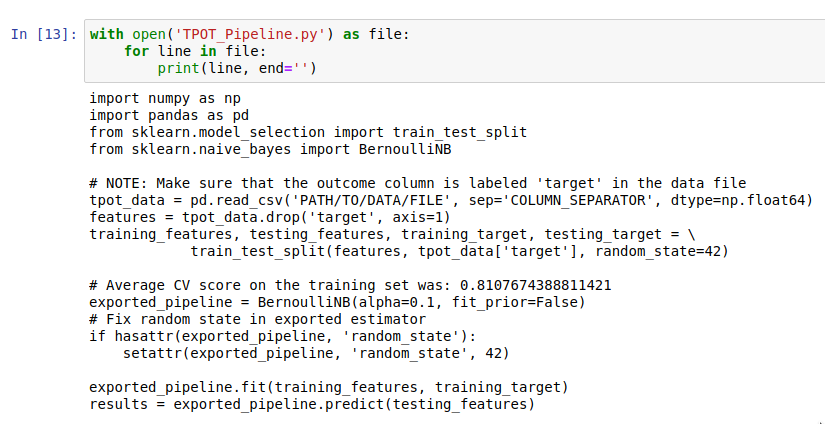
1. **TFIDF + SVM (dataset3)**

Реализация в коде + вывод, третий датасет:

Заданее хочется отметить следующее: проверка по К-фолдам не осуществлялась, т.к для третьего датасета четко представлены тренировочный и тестовый наборы данных.

1. **TPOT (dataset3)**

Реализация в коде + вывод, третий датасет:Полученная точность модели равняется 75%. Как уже указывалось выше, проверка по К-фолдам не осуществяется.

ТРОТ предлагает следующую модель:

1. **Summary по пункту**

Для визуальной оценки полученных результатов, сведем их в таблицу:

|  |  |  |
| --- | --- | --- |
| MODEL | F1 score, % | Std |
| TFIDF + SVM | 64.328 | 6.011 |
| TFIDF + MLP | 65.988 | 5.821 |
| TFIDF + GBM | 57.575 | 3.853 |
| FastText + SVM | 86.925 | 2.338 |
| TFIDF + SVM (d3) | 74.321 | - |
| TFIDF + SVM (d3) | 75.177 | - |

Выводы: среди проведенных оценок, наилучшей на втором датасете оказалась модель с многослойным персептроном (MLP), но вычислительная способность такой модели не выглядит привлекательной, именно поэтому в качестве передовой модели мной была выбрана модель опорных векторов (SVM). Использование векторизации текста по методам FastText оказалось эффективней метода TFIDF.

**Классификатор FastText**

Бла бла, ничего не понял….

Шутка, напишу позже

**Нейронная сеть с LSTM**

Кайфы природные

**Заключение**

Фух щищ