Toxic Comment Classification Challenge

Identify and classify toxic online comments

Павел Плюснин
Московский физико-технический институт
plyus.pavel2010@yandex.ru

ABSTRACT

Задачи классификации текстов обладают проблемой высокой размерности, поэтому затруднительно применение сложных методов. В рассматриваемой задаче необходимо по тексту комментария понять, не является ли он «токсичным». Для решения этой задачи было применено два подхода. Первый – решать задачу в лоб, в условиях высокой размерности, описывая входные данные через матрицу tf-idf. Второй – понизить размерность, выполнив тематическое моделирование и описав какждый текст его распределением вероятностей принадлежности к различным темам. Также была произведена детальная обработка данных, поиск и исключение шумовых объектов из обучающей выборки. Лучшая модель была оценена значением 0.9777 по метрике mean column-wise ROC AUC.

KEYWORDS

Machine learning, text classification

ACM Reference Format:
Павел Плюснин. 2018. Toxic Comment Classification Challenge: Identify and classify toxic online comments. In *Proceedings of Machine learning MIPT course*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 ПОСТАНОВКА ЗАДАЧИ

Полное описание задачи доступно по ссылке:
https://www.kaggle.com/c/ jigsaw-toxic-comment-classification-challenge

Необходимо по тексту комментария определить, является ли он непреемлемым, т.е является ли он токсичным(toxic), очень токсичным(severe_toxic), непристойным(obscene), содержащим угрозы (threat), оскорбления (insult) или ненависть (identity_hate).
В обучающей выборке дана таблица, где для каждого сообщения дан его id, текст сообщения, а также принадлежит ли он к каждому из 6 вышеуказанных типов. Необходимо для тестовой выборки по id и тексту сообщения, предсказать вероятности принадлежности к каждому из 6 типов «плохих» сообщений.
Метрикой качества является среднее значение всех ROC AUC каждого прогнозируемого столбцов (mean column-wise ROC AUC).

Machine learning MIPT course, May 2018, Moscow, Russia

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM. . . $0
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 BASELINE

Будем считать, что id никак не влияет на токсичность сообщения, поэтому будем анализировать только сам текст.
В качестве baseline рассмотрим следующую модель: для имеющихся данных (для обучающей и тестовой выборок) получим их Tf-idf представление (воспользуемся TfidfVectorizer из sklearn.feature_extraction.text). На основе полученной матрицы обучим 4 следующих модели со стандартными параметрами: наивный байесовский классификатор (sklearn.naive_bayes.MultinomialNB), логистическую регрессию (sklearn.linear_model.LogisticRegression), линейный SVM классификатор (sklearn.linear_model.LinearSVC) и линейный классификатор на основе стохастического градиентного спуска(sklearn.linear_model.SGDClassifier). Из-за высокой размерности мы не можем применять многие более сложные методы, такие как случайный лес или бустинг. Случайный лес будет обучаться слишком долго, чтобы добиться достаточного качества, а бустинг, обычно строящийся на неглубоких деревьях, не сможет уловить сложные зависимости между частотами вхождений слов, их комбинациями, результат тоже будет неудовлетворительным. Линейные же модели обучаются быстро и, как известно, выдают неплохой результат.
Качество построенных моделей будем отслеживать методом кросс-валидации, разбив обучающую выборку на 3 фолда, метрика - ROC AUC.

Как мы видим, наши модели дали очень хороший результат. LogisticRegression и LinearSVC дали очень близкие значения функционала качества ≈ 0.970, SGDClassifier отработал чуть хуже, его результат примерно равен 0.967. На их фоне MultinomialNB дал довольно плохое качество предсказаний - всего 0.863

3 FEATURE ENGINEERING

Теперь переходим к вдумчивому построению модели. Сперва построим несколько вспомогательных признаков:

- CAPS - отношение слов, написанных в верхнем регистре, ко всем словам в сообщении (возможно, злоупотребление CAPSом влияет на вероятность того, что сообщение не будет токсичным)
- n_words - Количество слов в сообщении (возможно, длина сообщения влияет на его токсичность)

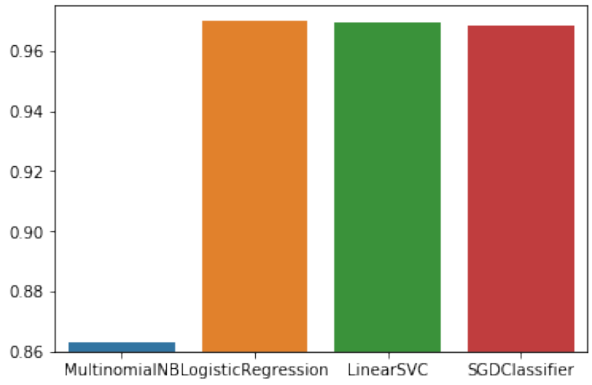

Рис. 1: CrossValScore на <toxic> baseline модели

- n_punct - количество различных знаков препинания в сообщении (возможно, использование знаков препинания как-то влияет на вероятность того, что сообщение не будет токсичным)
- punct_to_word - отношение n_punct к n_words (возможно, уровень грамотности, который грубо можно оценить таким отношением, влияет на вероятность того, что сообщение не будет токсичным)

Для вычисления этих признаков, а также для полного контроля над происходящей обработкой текста, будем все проводить в ручном режиме, используя библиотеку nltk

После вычисления новых признаков на всех объектах тестовой выборки, определем их качество: Для этого построим те же линейные модели для той же таблицы с tf-idf, дополненной новыми признаками
Для каждой модели выведем CrossValScore, а также коэффициенты полученной модели у нескольких последних признаков. Как мы видим в наивной байесовской модели, каждое слово имеет вклад порядка 11, когда наши признаки имеют вклад всего 6, 2 или вообще 0.6. Такая же ситуация и с остальными моделями. Таким образом получили, что придуманные признаки, похоже, не являются удачными, они довольно слабо учитываются в модели, хоть они и агрегируют информацию о всем сообщении, всего одно слово в сообщении имеет вес больший, чем веса этих признаков. Это подтверждается и CrossValScorom - значения только упали, получается, что наши новые признаки скорее мешают. Более детальный анализ проведем далее, когда введем все новые признаки.

3.1 Тематическое моделирование

Теперь построим тематическую модель для наших данных. Кажется логичным, что агрессивные комментарии чаще появляются в теме "политика чем в теме "дикая природа". Поэтому, получив для каждого сообщения его распределение тем, мы сможем более точно классифицировать наши сообщения. Но для этого сначала надо получить сами темы, а уже в построеной тематической модели матрица Θ^T и является нашей таблицей признаков.

sbj14	sbj25	sbj18	sbj1
seem	help	fuck	song
fact	question	u	music
case	ask	shit	im
their	thank	category	eat
opinion	welcome	yourself	band
both	hope	hey	na
evidence	message	jew	album
faggot	place	vandalize	record
quite	hello	fucking	play
matter	write	fat	movie
pov	check	asshole	fan
actually	date	piece	rock
clear	contribution	cunt	night
rather	answer	mother	hit
whether	sign	dog	son

Для построения тематической модели я выбрал библиотеку BigARTM

Было построено 30 тем, одна из которых - общая (в нее войдут общеупотребительные слова). Из словаря модели были исключены слишком частые слова (например, a, to, by, I, he, и т.д), чтобы они не засоряли темы, а также слишком редкие слова. Порог, по которму исключались такие слова определялся экспериментально, по значению token_df у бесполезных и полезных слов.

Детали построения тематической модели можно увидеть в подробном отчете.

Модель сработала даже лучше, чем ожидалось. Мы получили скорее типы сообщений - высказывание мнения (sbj14), обращение и просьбы о помощи (sbj25), оскорбление (sbj18) и т.д. Темы в классическом понимании тоже получились, например тема музыки (sbj1), политического устройств а(sbj12), Америки (sbj24).

После этого оценим качество полученного представления текстов через темы. Для этого Построим линейные модели:

- на всех новых признаках (4 признака: CAPS, n_words, n_punct, punct_to_word + темы)
- только на темах
- только на CAPS, n_words, n_punct

Результаты можно увидеть на Рис.2. Использование всего 30 признаков-тем дает почти такой же результат, как и использование всей огромной tf-idf таблицы с сотней тысяч признаков: Таким образом, мы эффективно произвели понижение размерности нашего признакового протранства. Темы являются очень качественными и информативными признаками. Использование еще 4 признаков (CAPS, n_words, n_punct, punct_to_word) немного ухудшает качество модели. Использование только CAPS, n_words, n_punct, punct_to_word приводит к плохим результатам, что неудивительно.

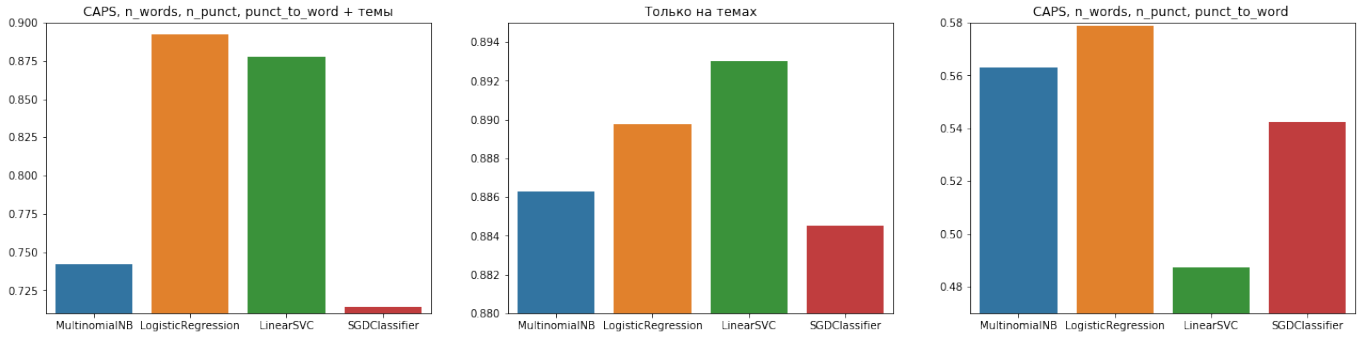

Рис. 2: CrossValScore на <toxic> в зависимости от использованных признаков

3.2 Использование дополнительных модальностей

Найдем самые значимые слова для классификации. Затем будем использовать количество вхождений этих слов в сообщения и типы токсичности сообщения как модальности. Построим 5 «тем», основанных на таких модальностях и добавим полученную матрицу Θ_{modal} к нашим признакам. Новые признаки действительно имеют смысл, их добавление улучшает качество модели предсказания поля <toxic> до 0.9075777 при использовании логистической регрессии. К сожалению, новые признаки и превносят некоторые проблемы: они довольно сильно коррелируют с 30 главными темами, а существенным, в основном, является только mod1 - полный аналог subj18. Поэтому использовать 5 новых признаков будем с осторожностью.

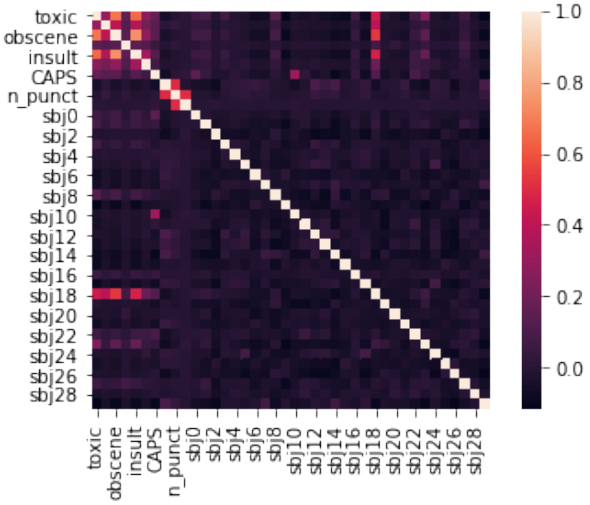

Рис. 3: Корреляции признаков

4 АНАЛИЗ ДАННЫХ

Теперь мы, наконец, можем провести подробный анализ данных, имея 39 вещественных признаков.

На гистограммах признаков видим, что нормально распределенных признаков нет, все они имеют сильный перекос в сторону 0, принимая значения 0 в большом количестве объектов. Выборка несбалансирована - большинство сообщений не являются токсичными.

На графике корреляций признаков (Рис.3) видим, что разные виды "токсичности коррелируют между собой, что понятно. С целевыми признаками больше всего коррелирует sbj18. Интересно, что sbj10 коррелирует с CAPS. Взглянув на тему sbj10 понятно, что это тема чисел (в нее входят различные года и числа). Числа мы трактовали как слова, а все числа, можно сказать, написаны CAPS (их uppercase совпадает с ними самими). Поэтому этот результат легко объясним. Корреляция не являеься слишком большой, поэтому предпринимать по этому поводу мы ничего не будем. Сильно коррелируют n_punct и punct_to_word, что тоже понятно

Из 5 модальных тем больше всего коррелирует с целевыми признаками mod2. Все эти 5 тем значительно коррелируют с 30 темами. Это тоже неудивительно, ведь и то, и другое является тематическим моделированием. Но эта корреляция может иметь негативный эффект в дальнейшей работе, надо иметь в виду Для большей наглядности, посмотрим какие 5 признаков сильнее всего коррелируют с каждым из целевых (кроме него самого и остальных целевых). Столбчатые диаграммы, соответствующие таким признакам можно увидеть на Рис.4.

Как мы видим, на первом месте уверенно лидирует sbj18. Также, во всех топах присутствует признак CAPS. Он единственный информатиынй из всей четверки CAPS, n_words, n_punct, punct_to_word. Поэтому в будущем будем использовать только CAPS и 30 тем (а также, иногда, 5 «модальных» тем), остальные 3 признака являются шумовыми.

Также построим scatterplot для пар наших признаков. Для того, чтобы сократить количество графиков, для визуализации возьмем только 5 признаков с наибольшей корреляцией с целевым. Каких то явных закономерностей обнаружено не было, видно лишь что точки сильнее прижаты к левому нижнему углу. *График можно увидеть в подробной версии отчета.*

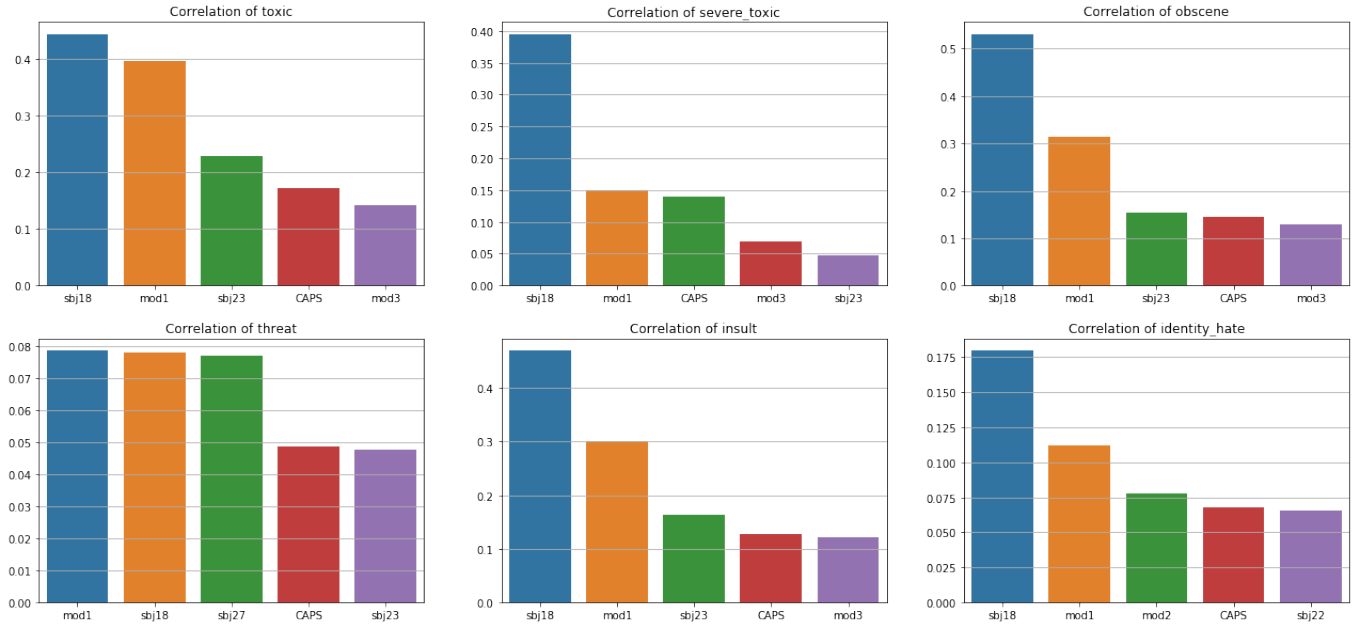

Рис. 4: Признаки, которые сильнее всего коррелируют с целевыми

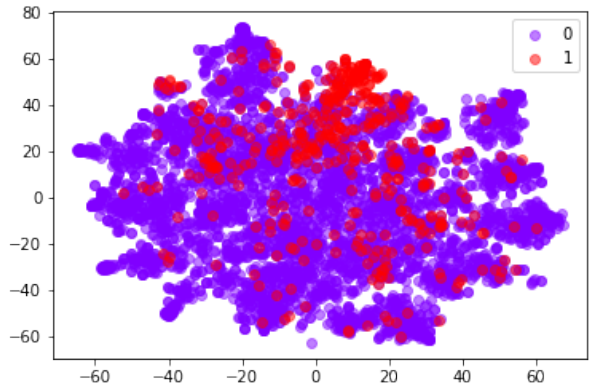

Рис. 5: Понижение размерности данных методом tSNE.

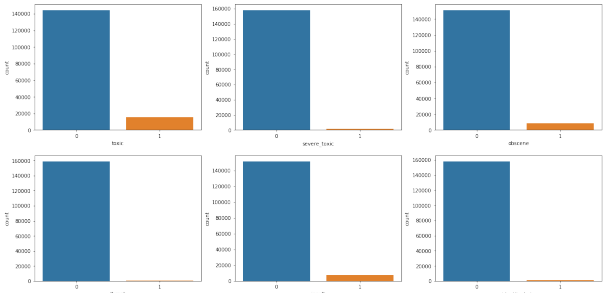

Рис. 6: Гистограммы количества объектов

Еще раз убедимся в том, что выборка несбалансированная (см Рис.6), токсичные объекты составляют одну десятую часть от всей выборки.

5 ВЫЯВЛЕНИЕ АНОМАЛИЙ (ВЫБРОСОВ)

Будем искать аномалии среди признаков, корреляция которых с целевыми переменными > 0.05. Использовать «модальные» темы не будем - было экспериментально получено, что их использование только ухудшает отбор аномальных объектов. Опытным путем было установлено, что OneClassSVM с параметрами gamma=10 и nu=0.01 дает самый подходящий результат. Получили, что 6.6% объектов признаются выбросами. Построим на графиках пар признаков, какие объекты мы посчитали аномальными (они выделены фиолетовым на фоне обычных - желтых): см Рис.7. Доля токсичных сообщений среди аномальных объектов совпадает с долей токсичных сообщений во всей выборке, значит, аномальные объекты действительно являются шумовыми и не вписываются в модель. Исключим эти объекты из обучающей выборки. Наша модель, действительно стала работать лучше (в качестве признаков - только темы и CAPS), что видно на Рис.8

6 ПОСТРОЕНИЕ ОСНОВНОЙ МОДЕЛИ

Использование тем в качестве признаков оказалось продуктивной идеей, таким образом мы смогли серьезно понизить размерность пространства. Но исключая матрицу tf-idf мы все же теряем значительную часть информации. Вернемся к этому представлению и продолжим наши исследования. До этого момента мы строили модели только для "toxic"для краткости изложения. С этого момента будем строить модель для каждого из целевых признаков.

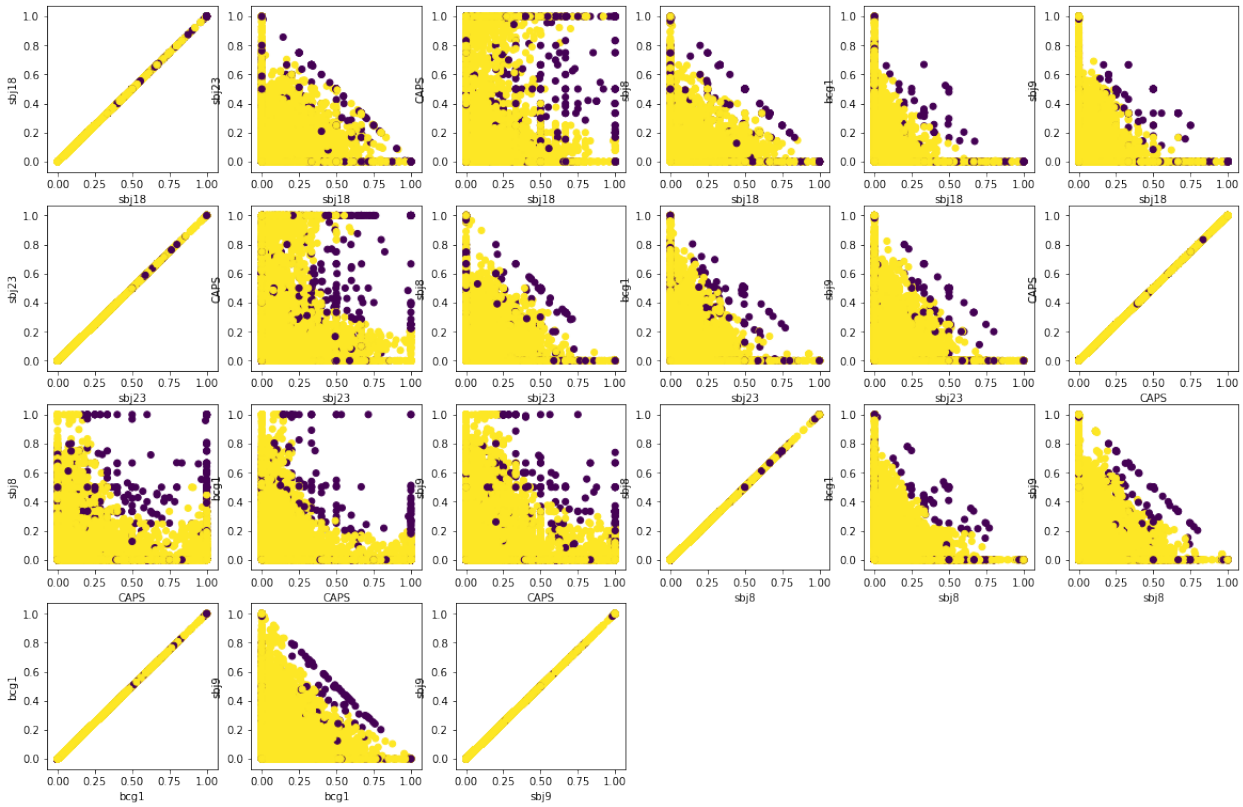

Рис. 7: Объекты-выбросы и «нормальные» объекты на графиках пар признаков, 6.6% объектов признаются аномальными

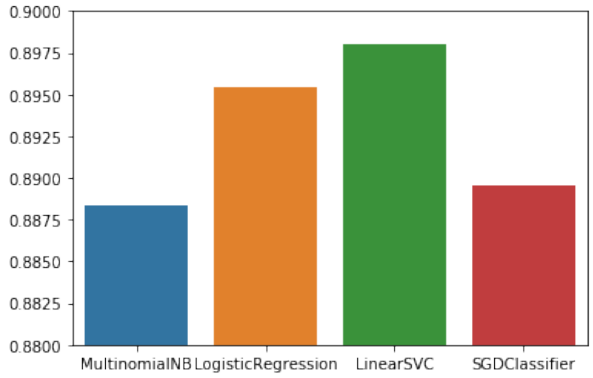

Рис. 8: CrossValScore на <toxic> после удаления шумовых объектов из выборки

6.1 Первое приближение

Построим матрицу объекты-признаки, объединив представление tf-idf с нашим новым 31 признаком. От этой модели и будем отталкиваться. Здесь и далее в качестве тестовой выборки выступает тестовая выборка с kaggle. Следует отличать CrossValScore вычисленный на обучающей выборке от результата, полученного от kaggle на тестовой выборке.

CrossValScore = 0.9783
ROC AUC на тесте = 0.9714

6.2 Второе приближение

Теперь добавим n-граммы для символов. Возможно, это поможет обходить орфографические ошибки в словах, а также учитывать смайлы и прочие символы

CrossValScore = 0.9835
ROC AUC на тесте = 0.9758

Добавление 5 «модальных» тем еще немного увеличивает качество

CrossValScore = 0.9810
ROC AUC на тесте = 0.9777

6.3 Понижение размерности модели предыдущего пункта

Будем понижать размерность при помощи TruncatedSVD – SVD разложения. CrossValScore на <toxic> в зависимости от количества компонент разложения можно увидеть на Рис.9.

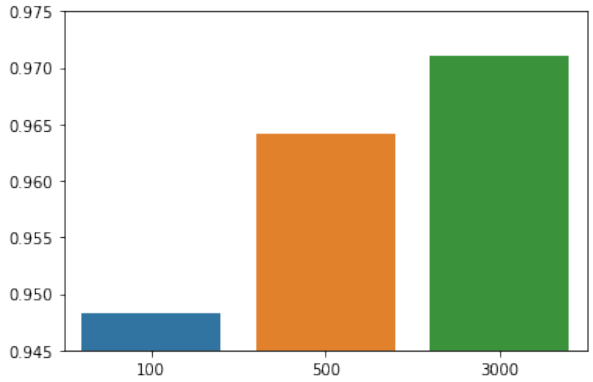

Рис. 9: CrossValScore на <toxic> в зависимости от количества компонент разложения

Результаты на всех целевых признаках при количестве компонент = 3000:

CrossValScore = 0.9808
ROC AUC на тесте = 0.9741

Теперь разберемся с SVD разложением серьезнее. Разложение на 800 компонент позволяет объяснить около 40% всей дисперсии. Графики зависимостей объясненных дисперсий от количества использованных компонент можно увидеть на Рис.10.

Опрделеим оптимальное количество компонент для разложения. Для этого посмотрим на разности в дисперсиях в отсортированном ряде $\lambda_{(1)} > \lambda_{(2)} > \cdots > \lambda_{(D)}$: $\lambda_{(1)} - \lambda_{(2)}, \ldots, \lambda_{(D-1)} - \lambda_{(D)}$, и удалим те компоненты, начиная с самой большой разности. Получили, что формально мы должны оставить всего одну компоненту - именно на ней наблюдается наибольший скачок. Но мы попробуем также и вариант с 6 компонентами, который из графика кажется более логичным. Графики представлены на Рис.11

Результаты при использовании 6 главных компонент:

CrossValScore = 0.9796
ROC AUC на тесте = 0.9722

Результаты при использовании 1 главной компоненты:

CrossValScore = 0.9783
ROC AUC на тесте = 0.9717

6.4 Оптимизация параметров модели

Наконец приступим к настройке параметров модели. В связи с высокой размерностью данных, мы строим линейные модели, у них не так много параметров для настройки. Таким образом, мы будем настраивать параметр C отвечающий за регуляризацию, а также параметр, отвечающий за необходимость балансировки обучающей выборки. Параметры модели будем подбирать при помощи поиска по сетке (GridSearchCV).
В результате получили следующие результаты: Логистическая регрессия предпочитает, чтобы классы были сбалансированы, а C=1, SVM предпочитает несбалансированные классы, а C=0.1

Значения метрики качества:

CrossValScore = 0.9820
ROC AUC на тесте = 0.8270

Эта посылка была оценена в 0.827, хотя по Кросс-валидации получился результат намного выше. Очевидно, что мы переобучились.

6.5 Модель на основе XGBoost

Посмотрим, на что способен бустинг над решающими деревьями. В нашем случае придется воспользоваться только 31 признаком - темами и CAPS. Проверим, сможет ли модель на основе бустинга оказаться лучше чем простая линейная. В связи с тем, что модель строится довольно долго, а качество ее довольно сомнительно, не будем настраивать параметры модели - построим с параметрами по умолчанию. Значения метрики качества:

CrossValScore = 0.9431
ROC AUC на тесте = 0.9319

Действительно, результат лучше чем у простой линейной моделью над теми же признаками. И это без оптимизации.

7 ВЫВОДЫ И ЗАКЛЮЧЕНИЕ

Таким образом, наша лучшая модель имеет значение метрики качества 0.9777 на тестовой выборке. Значения метрики качества для различных использованных методов видны на Рис.12.

В рассмотренной задаче огромная размерность данных и к разрешению этой проблемы можно подойти с разных сторон. Во-первых, можно понизить размерность. Для этого мы провели тематическое моделирование и описали каждый текст вероятностным распределением. Также можно просто понизить размерность матрицы tf-idf известными методами, такими как SVD разложение. Оба эти метода довольно эффективны и не слишком портят качество модели с прикладной точки зрения, но в условиях соревнования потеря информации существенна. Поэтому, даже применение сложных методов не дает выигрыша. Во-вторых, можно ничего не делать с высокой размерностью, а просто строить простые модели. Такой вариант хуже интерпретировать, он сложнее в применении, хранении данных, но дает лучшее качество. В зависимости от конкретной прикладной задачи можно выбирать одно из этих двух решений.

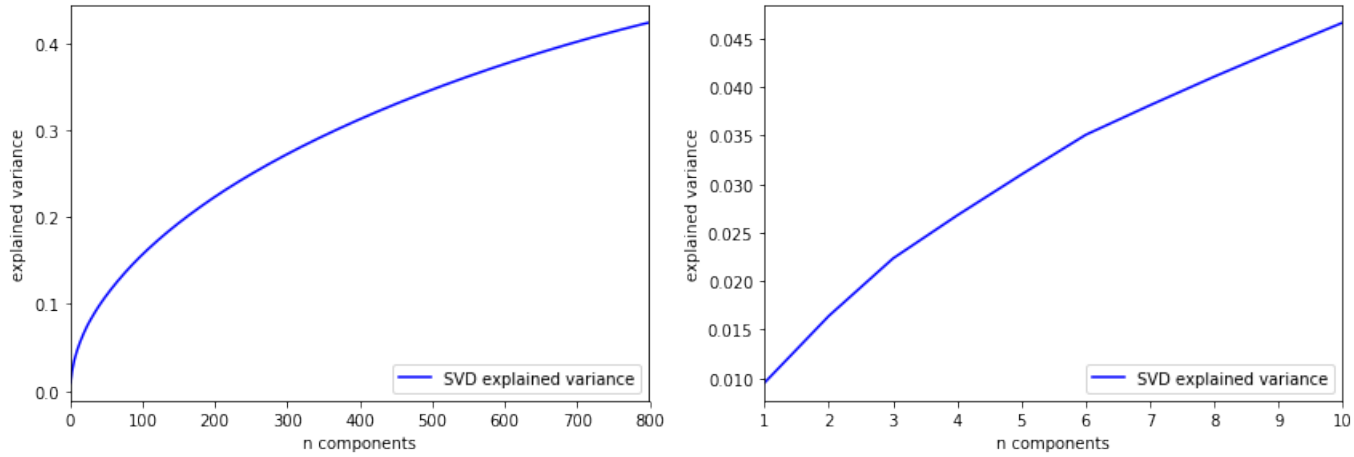

Рис. 10: Графики зависимостей объясненных дисперсий от количества использованных компонент

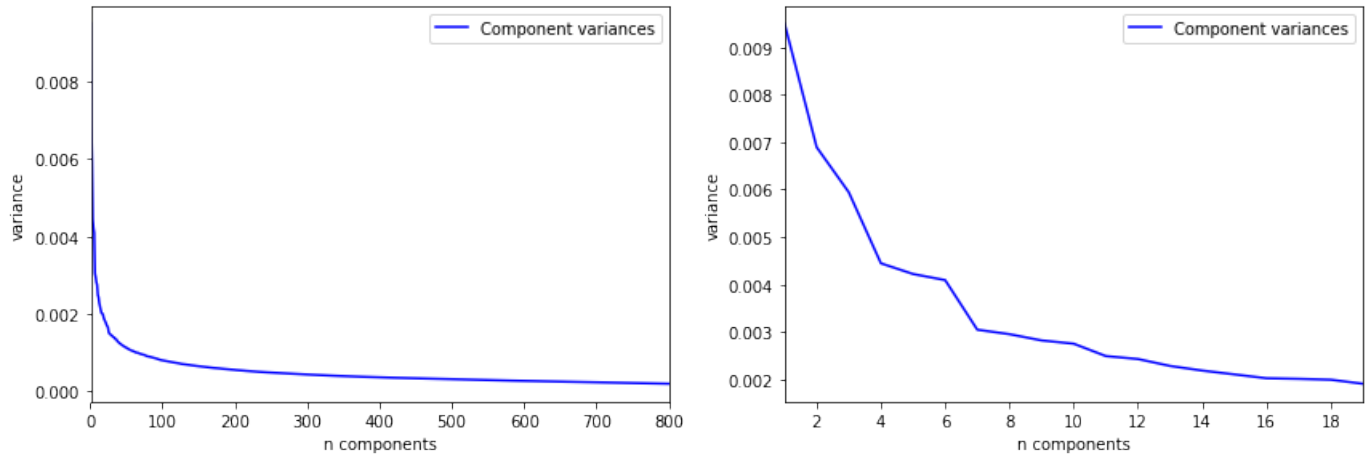

Рис. 11: График разностей в дисперсиях в зависимости от номера компоненты

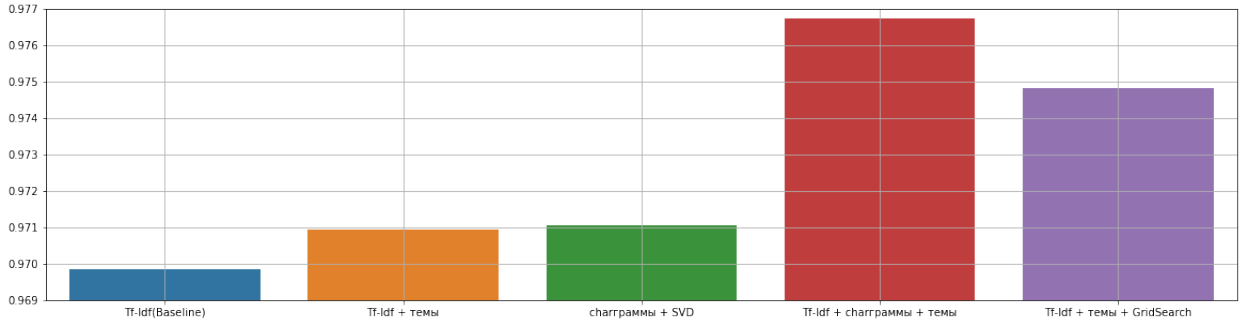

Рис. 12: CrossValScore <toxic> при разных обработках данных