# Как проанализировать отзывы о компании с помощью машинного обучения

Сегодня мы часто слышим о повсеместном применении машинного обучения (**Machine Learning**), больших данных (**BigData**) и нейронных сетях (**Neural Net**). «Непогруженному» человеку бывает непросто разобраться, чем, например, машинное обучение отличается от глубокого обучения (**Deep Learning**), и почему не для каждой задачи нужны нейросети.

Поэтому мы постараемся у нас на сайте доступным языком рассказывать о применении методов машинного обучения для решения реальных задач, которые могут появиться у компании любого размера.

В данной статье разберемся, как с помощью методов машинного обучения можно провести анализ отзывов клиентов о товарах/услугах вашей компании.

## Почему мы стали этим заниматься?

Традиционно Фобизнес много лет помогал клиентам с решением задач автоматизации учета на базе программных продуктов фирмы «1С». При этом зачастую от клиентов мы получали запросы, выходящие за рамки деятельности среднестатистического франчайзи «1С». Одним из таких кейсов стал запрос от нашего заказчика, предоставляющего b2c услуги для широкого круга потребителей, который был связан с анализом отзывов клиентов об услугах.

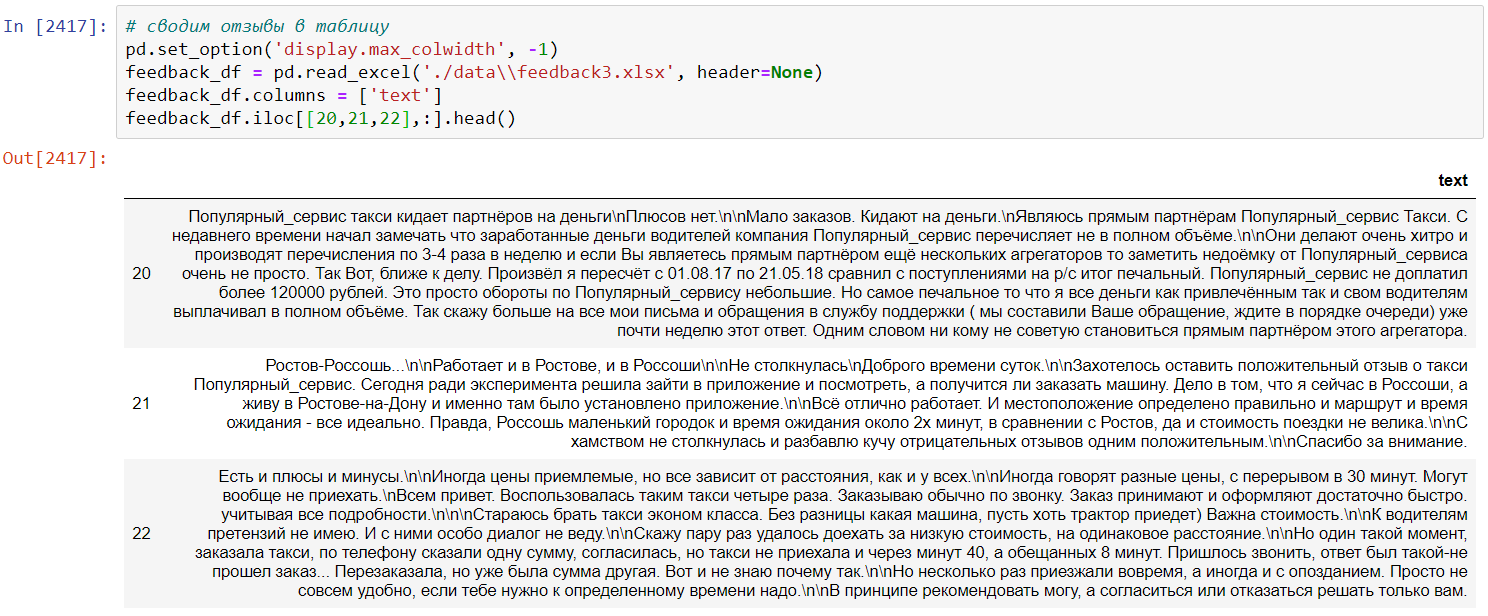
ИТ-специалисты клиента на тот момент уже написали алгоритм, собирающий отзывы о нем с сайтов-отзовиков, нужно было эти отзывы проанализировать и понять, о чем в основном пишут в своих отзывах клиенты. Следующим шагом заказчик хотел проанализировать тональность всех отзывов (положительный/нейтральный/отрицательный). Количество отзывов исчислялось тысячами, и стало понятно, что анализировать их вручную будет очень трудозатратно.

## Исходные данные и инструменты.

По просьбе клиента в этой статье мы будем использовать отзывы о совершенно другом сервисе, полученные с открытых сайтов-отзовиков (возьмем 100 отзывов о сервисе такси). Но это никак не помешает нам оценить эффективность описываемого подхода.

Сразу оговорим, что в рамках этой статьи мы не будем создавать модель, которая делает какие-то прогнозы или что-то классифицирует на основе данных. Мы научимся выделять наиболее вероятные темы отзывов и оценим значимость использованных в этих отзывах слов и выражений. Создание классификатора, который определяет тональность отзывов (положительный или отрицательный), разберем в следующей статье.

В качестве инструментов для работы с данными будем использовать только бесплатные библиотеки на языке **Python**.



Отзывы в таком виде обработать большинству алгоритмов не под силу. Необходимо их «оцифровать», потому что алгоритмы машинного обучения «любят» числа и «не понимают» буквы.

Один из наиболее простых, но эффективных подходов называется **«мешок слов»** (**bag - of - words**). Он состоит из двух основных этапов:

1. Разбиение каждого отзыва на отдельные слова. Например: текст «Иногда цены приемлемые» превращается в список [«иногда», «цены», «приемлемые»]. Заметим, что все слова приводятся к нижнем регистру. Этот шаг называется ***токенизация***. Отдельные слова в списке [«иногда», «цены», «приемлемые»] будем называть ***токенами***.
2. Создание таблицы, в которой столбцы – **токены**, колонки – номера отзывов от 1 до 100. Таблица выглядит примерно так:

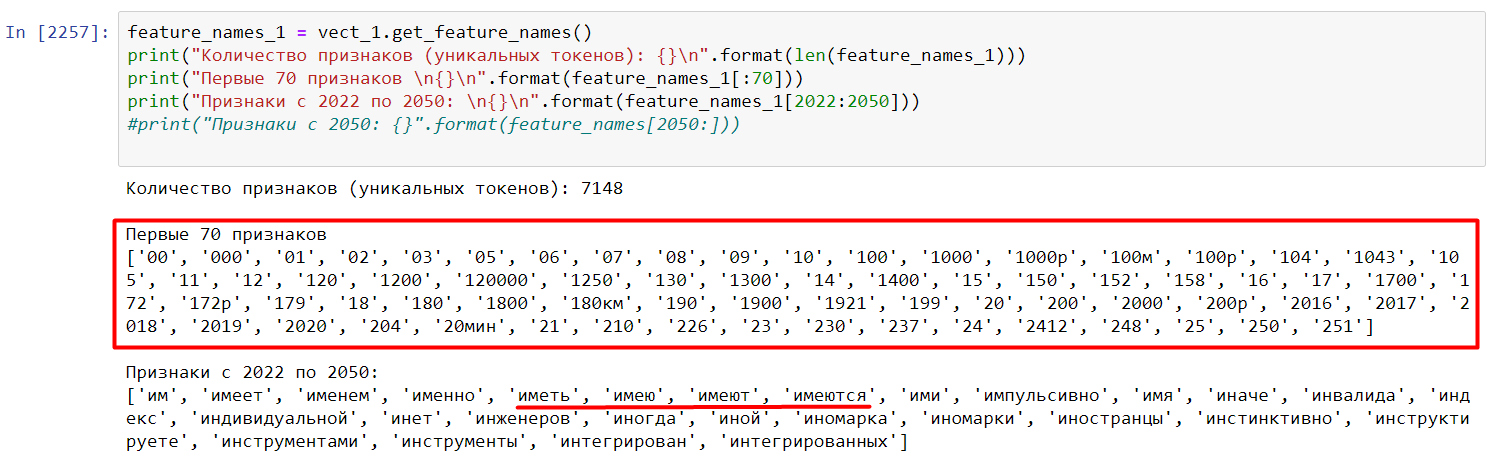


Для каждого отзыва записывается количество вхождений каждого ***токена***. На самом деле в таком виде, как показано на рисунке, данные стараются не хранить, т.к. в основном получившаяся матрица состоит из нулей, а хранить в памяти кучу нулей не оптимально. Поэтому используются другие структуры хранения данных – разреженные матрицы (**sparse matrix**), но на них останавливаться в этой статье не будем.

Итак, мы получили большую таблицу на 100 строк и 7148 столбцов:



Взглянув на ***токены***, становится ясно, что в таблицу попали разные цифры, а также, что однокоренные слова («иметь», «имею», «имеют», «имеются») стали отдельными ***токенами*** и будут рассматриваться как разные признаки в будущей модели. Более того, различные опечатки в одном слове создают новые ***токены***.



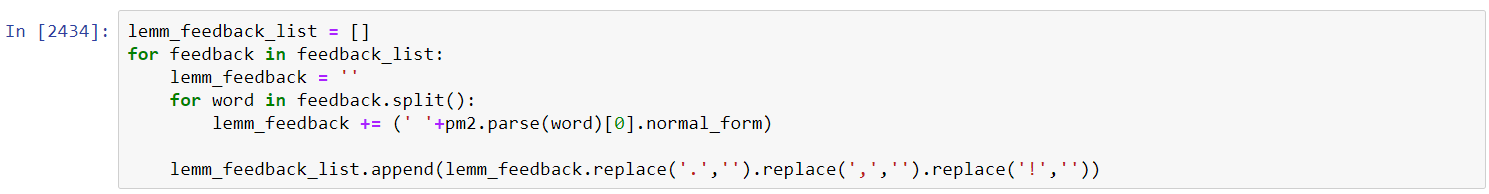
С таким явлением давно уже научились бороться с помощью такой процедуры как ***лемматизация***. Определение из Википедии:

***Лемматиза́ция*** — процесс приведения словоформы к лемме — её нормальной (словарной) форме.

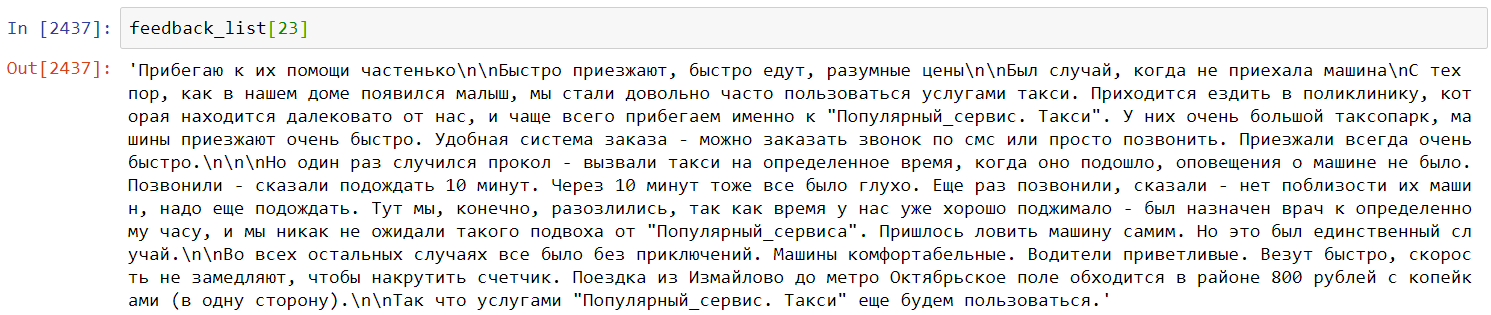
Иными словами, у каждого слова есть нормальная форма - ***лемма***, например:

* кошке, кошку, кошкам, кошкой **→ кошка**
* бежал, бежит, бегу **→** **бежать**
* белому, белым, белыми **→ белый**

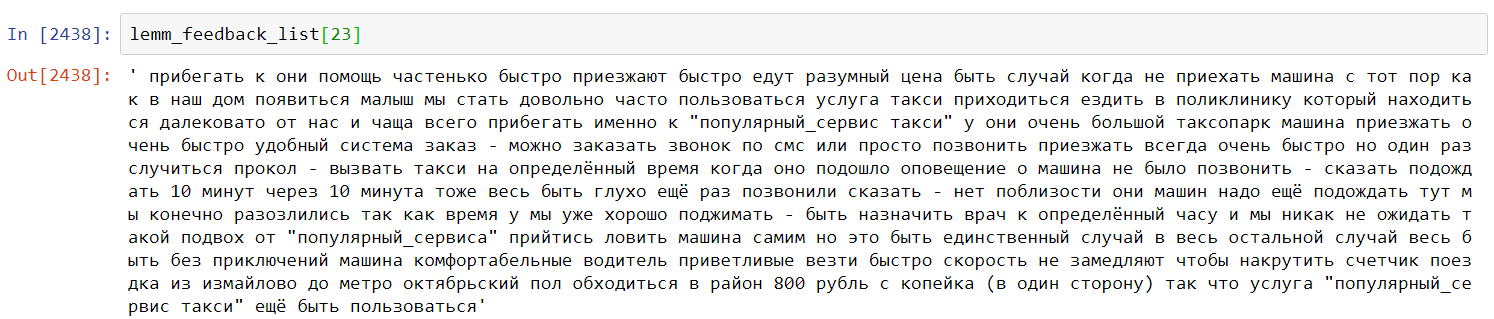
С помощью метода **CountVectorizer** из библиотеки **Scikit-Learn** выполним ***лемматизацию*** изначального текста отзывов (заодно удалим знаки препинания):



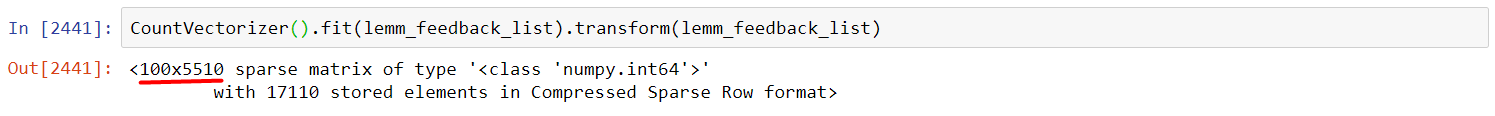
Было:



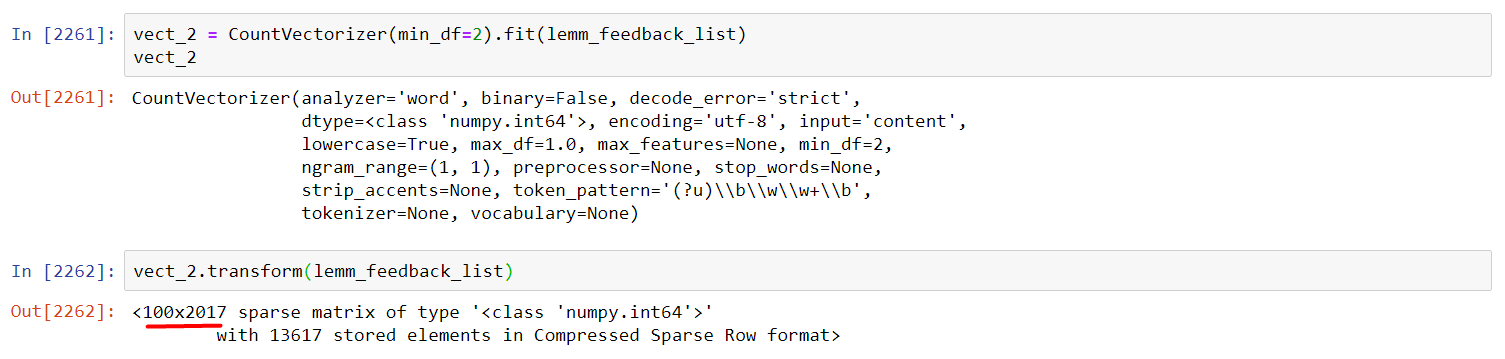
Стало:



Немного похоже на переведенную китайцами инструкцию к бытовой технике, но с точки зрения машинного обучения мы только что уменьшили размер обрабатываемых данных с 7148 ***токенов*** до 5510 практически без потери информации. Почти на четверть!

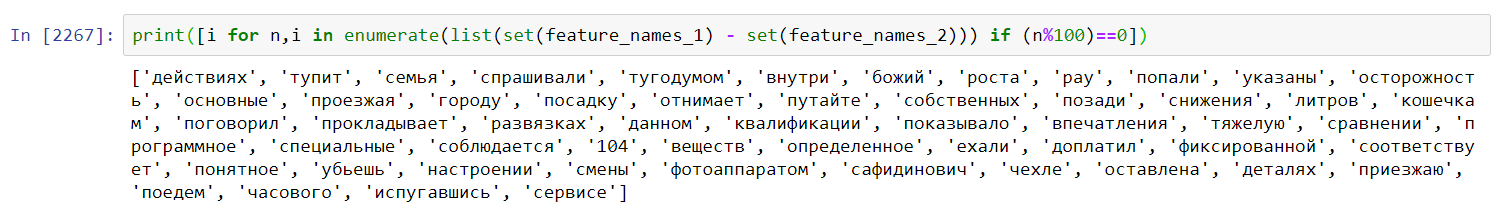


А что, если мы не хотим, чтобы наш будущим алгоритм работал с такими словами как «таскист», «ездиют» и прочими опечатками? Предполагаем, что такие слова не могут встречаться уж очень во многих отзывах из нашего набора. Отфильтруем такие «аномалии» по принципу: «если слово встречается хотя бы в 2 документах, то будем с ним работать, если только в 1 документе – то отбросим его». Установим соответствующие параметры в **CountVectorizer** и посмотрим на результат:



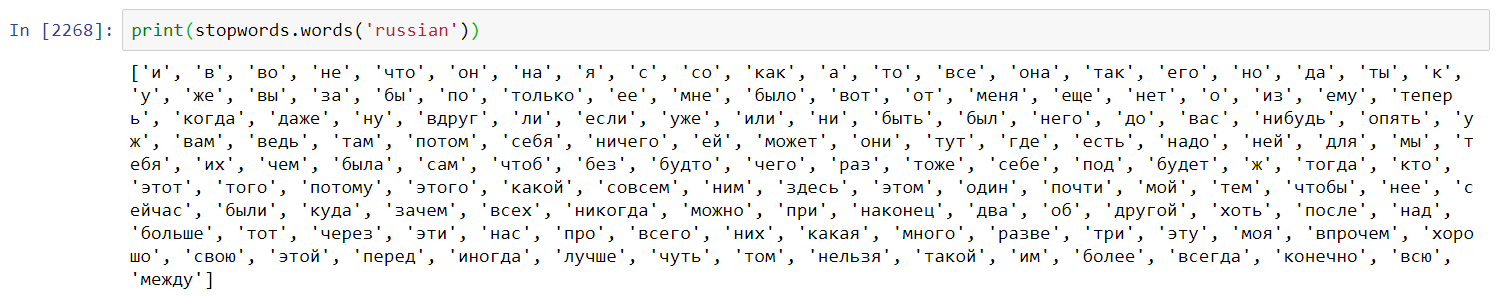
Вместо 5510 ***токенов*** у нас осталось только 2017! Мы снова уменьшили количество признаков, с которыми предстоит работать будущей модели (более чем в 3,5 раза от первоначального количества).

Посмотрим на некоторые слова, которые мы исключили:

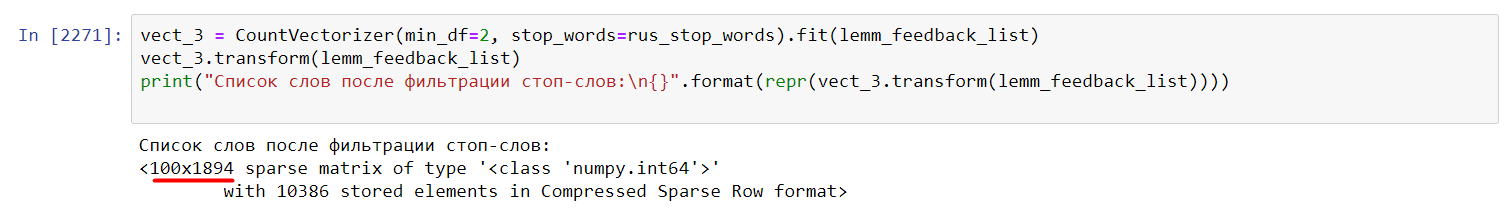


Заметим, что помимо исключения таких интересных слов как «сафидинович» и «тупит», исключены вполне обычные слова. Подход с удалением слов, встречающихся в менее n документах довольно грубый, но зачастую он вполне оправдан, т.к. редкие слова многими моделями воспринимаются как более значимые, заучиваются, но могут не встречаться в новых поступающих данных. Мы же не хотим, чтобы будущий алгоритм считал значимым признаком слово «бдзынь», «манатки», «васильева» и «3269».

Более тонким методом можно назвать использование ***стоп-слов*** (**stop-words**). По сути, это список слов, которые встречаются практически в каждом предложении, но не несут много информации (при этом нецензурные слова также часто включаются в список). Например, ***стоп-слова*** для русского языка:



Ну что ж, избавимся еще и от ***стоп-слов***:

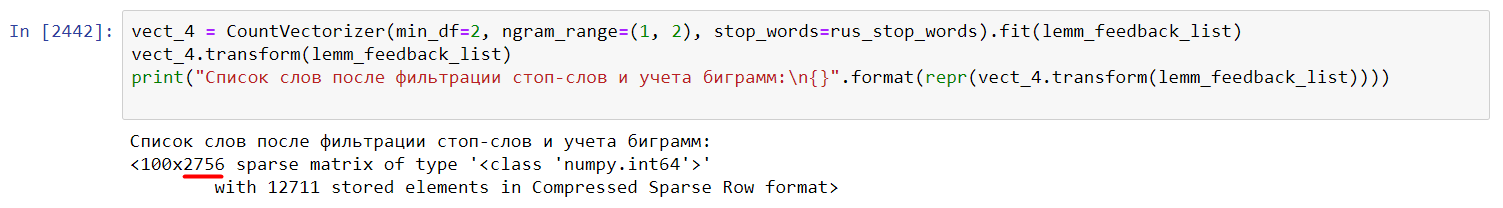


Количество ***токенов*** сократилось до 1894. Все это время мы планомерно уменьшали количество ***токенов*** для анализа.

Но кто-то может сказать: Подождите, кажется, разбивать предложения на отдельные слова это значит терять информацию, содержащуюся в словосочетаниях! Например: словосочетание «не так плохо» при таком подходе разбивается на слова [«не», «так», «плохо»], которые по-отдельности несут уже совсем другой смысл. И будет прав.

***Мешок слов***, о котором мы говорили вначале, можно сформировать не только из отдельных слов, но и из словосочетаний. Словосочетание из 2 слов называется ***биграммой***, из 3 слов – ***триграммой***. из n слов – ***n-граммой***. Таким образом, чтобы учесть контекст, который несут словосочетания, мы должны формировать не ***мешок слов***, а ***мешок слов и словосочетаний***.

Установим для **CountVectorizer** ранее использованные параметры (минимальное количество отзывов, в которых встречается слово, и использовать стоп-слова) и новый параметр (в качестве ***токенов*** взять как отдельные слова, так и ***биграммы***):



Количество ***токенов*** выросло до 2756, но сейчас мы уверены, что стоп-слова и редкие слова мы отфильтровали.

В итоге всех манипуляций мы получили таблицу (**мешок слов)** для 2756 слов и словосочетаний, содержащую частоты их вхождения для каждого отзыва. Каким образом мы можем проанализировать эти данные?

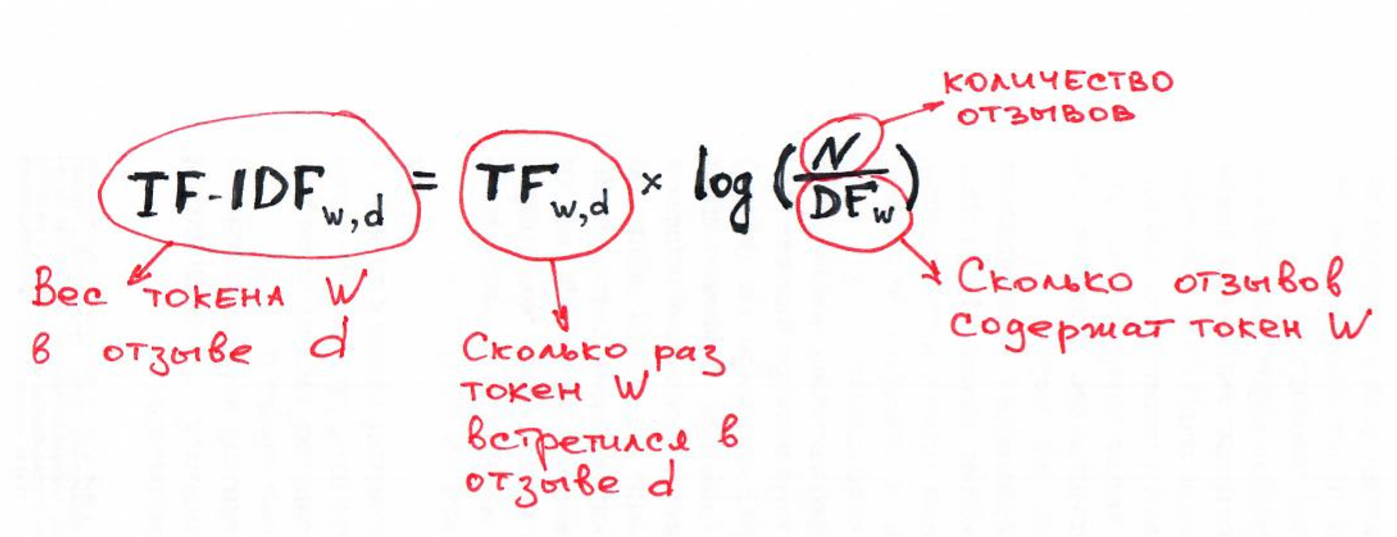
## Применение меры TF-IDF

Зададимся вопросом, а равноценны ли все 2756 токенов с точки зрения количества передаваемой информации? К примеру, если в каком-то отзыве ***токен*** «таксист» встречается 2 раза, а ***токен*** «удобство» только 1 раз, значит ли это, что один ***токен*** является более информативным, чем другой? И если мы продолжим избавляться от малоинформативных признаков, то от какого из двух ***токенов*** будем избавляться в первую очередь? И нужно ли учитывать, насколько эти ***токены*** часто встречаются во всех 100 отзывах?

Для того, чтобы оценить значимость каждого ***токена*** в каждом отдельном отзыве, применяется специальная мера **TF-IDF** (**TF** - частота слова, **IDF** — обратная частота документа).

**TF-IDF (TF — term frequency, IDF — inverse document frequency)** - статистическая мера, используемая для оценки важности слова в контексте документа (в нашем случае – отзыва), являющегося частью коллекции документов или корпуса (в нашем случае – набора из 100 отзывов).

Мера вычисляется по формуле\*:



Вес некоторого слова пропорционален частоте употребления этого слова в документе и обратно пропорционален частоте употребления слова во всех документах коллекции. Чем популярнее слово (чем оно чаще встречается в наборе отзывов), тем ближе логарифм, на который умножается встречаемость слова в отзыве, к нулю, и тем меньше вес ***токена***. И наоборот.

\*- Основание логарифма в приведенной формуле не принципиально, возьмем основание, равное 10.

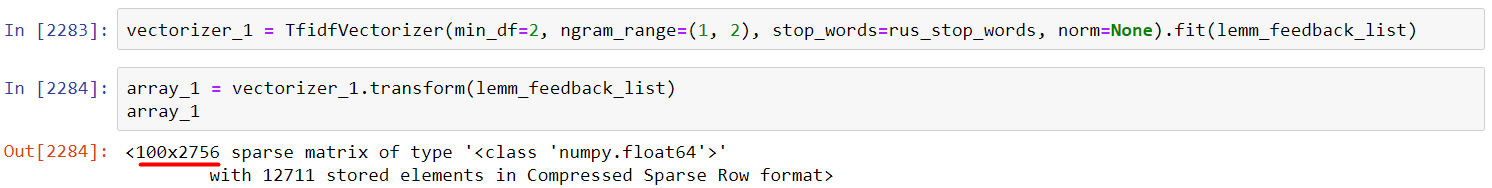
Пусть «таксист» встречается в 10 отзывах, а «удобство» - только в 4. Вычислим TF-IDF вес ***токенов***:

**TF-IDF («таксист», отзыв №7)** = 2 x log(100/10) = 2 x log(10) = 2 x 1 = **2**

**TF-IDF («удобство», отзыв №7)** = 1 x log(100/4) = 1 x log(25) = 1 x 1.4 = **1.4**

***Токен*** «таксист» в отзыве №7 имеет больший вес, чем токен «удобство».

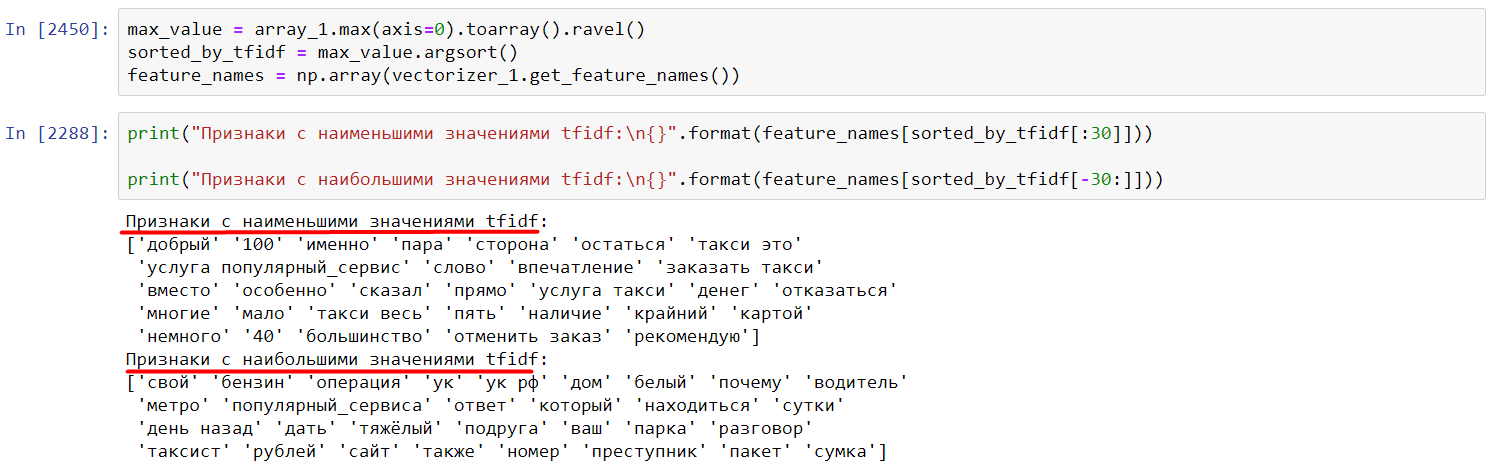
Применим метод **TfidfVectorizer** из библиотеки **Scikit-Learn**, чтобы получить **TF-IDF** веса для каждого из 2756 ***токенов***:



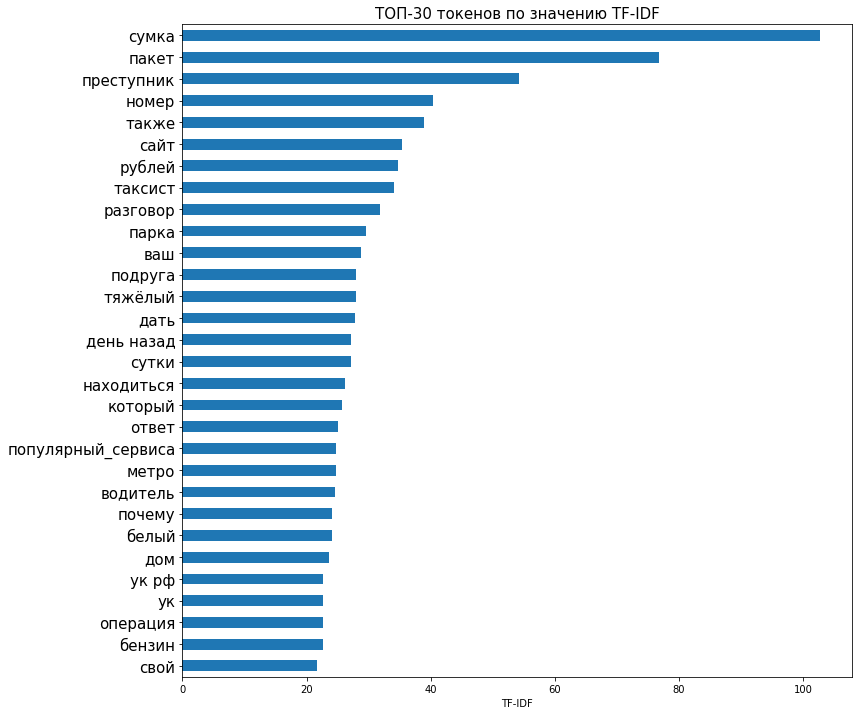
Соответственно значения в таблице ***мешок слов*** изменятся: вместо целых чисел, показывающих число вхождений ***токена*** в каждый отзыв, записываем полученные значения **TF-IDF**. Получаем примерно такую таблицу:



Рассортируем все ***токены*** по их максимальному значению веса **TF-IDF**:

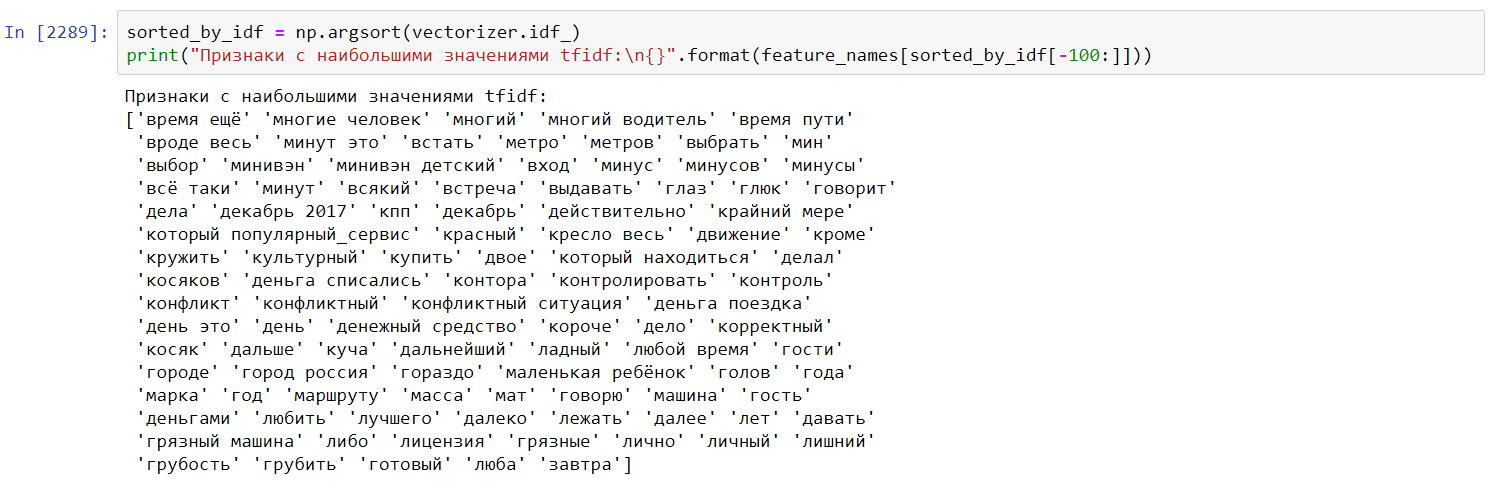


Визуализируем 30 наиболее весомых ***токенов*** по значению **TF-IDF**:



***Токены*** с низкими значениями **TF-IDF** встречаются в отзывах часто, ***токены*** с высокими значениями как правило встречаются в отзывах, тема которых хорошо описывается такими терминами (например: «сумка», «преступник», «парка»), т.е. автор пишет о чем-то, о чем не часто пишут остальные.

В целом по набору отзывов мы можем оценить каждый ***токен*** по значению **IDF** (вторая составляющая в формуле TF-IDF, логарифм от деления общего количества отзывов на количество отзывов, содержащих тот или иной ***токен***):



Такие ***токены*** в нашем наборе отзывов встречаются часто, и мы будем их считать типичными и не очень важными. Печально, что слово «конфликт» является типичным для отзывов о такси.

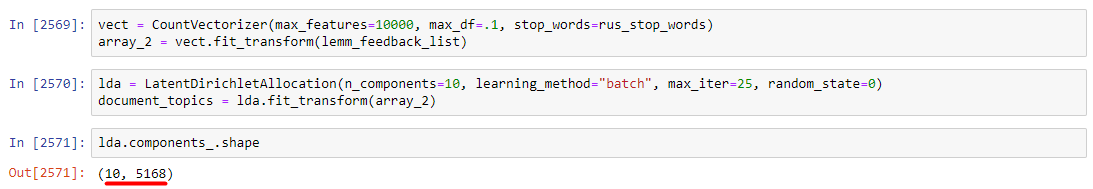
# Латентное размещение Дирихле (LDA)

В самом начале статьи мы запланировали извлечь полезные знания из текстов отзывов, а именно - выделить наиболее вероятные темы отзывов. Наконец применим машинное обучение!

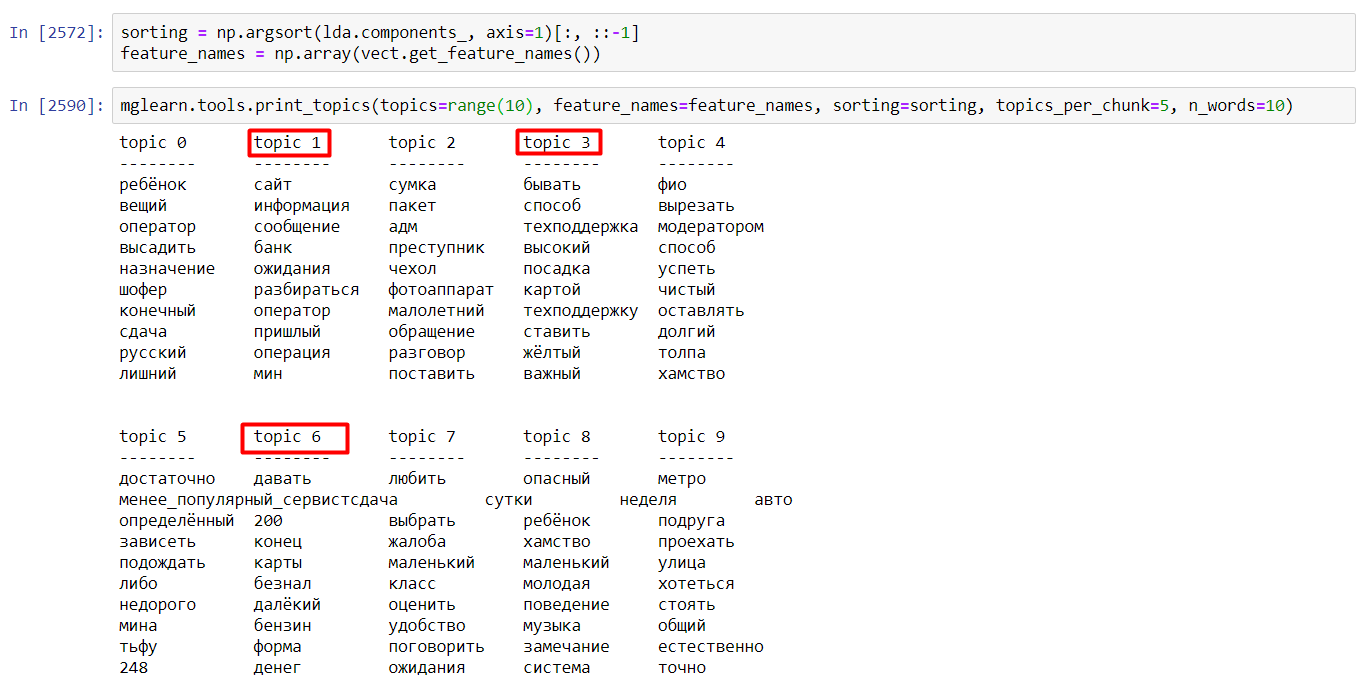
Одной из моделей, позволяющих объединить имеющиеся данные в группы на основании их сходства, является ***Латентное размещение Дирихле*** (**LDA**, от англ. **Latent Dirichlet allocation**). Это не единственный способ кластеризовать данные, но он хорошо работает в обработке текстов.

Смысл заключается в том, что мы рассматриваем каждый отзыв как смесь из некого количества тем. При этом появление какого-либо слова в отзыве вызвано принадлежностью этого отзыва определенной теме. С точки зрения алгоритма понятие темы это не то же самое, что тема для человека. Алгоритм оценивает вероятность появления группы слов вместе (принадлежности группы слов к той или иной теме). Количество тем мы должны указать алгоритму. Образно говоря, мы приказываем машине: «Ты должен прочитать все тексты и определить ровно 10 тем, о которых говорится в этих документах!». Темы, которые мы получим, могут с точки зрения человека выглядеть довольно странными, но чем больше отзывы «поляризованы» (автор пишет преимущественно про какой-то ключевой для него аспект), тем более «осмысленные» группы мы получим. Алгоритм **LDA** относится к алгоритмам машинного обучения без учителя, т.е. нам не нужно сообщать алгоритму: вот эти отзывы относятся к этой теме, те отзывы – к той теме, выучи и когда тебе покажут незнакомый текст, ты должен правильно отнести его к известным тебе темам. Мы лишь сообщаем количество тем, дальше алгоритм работает сам.

Тексты отзывов снова переводим в формат большой таблицы (***мешок слов***), передаем её методу **LatentDirichletAllocation** (реализует алгоритм **LDA** в библиотеке **Scikit-Learn**), и просим выделить 10 тем. Количество **n-грамм** ограничиваем 10 000.

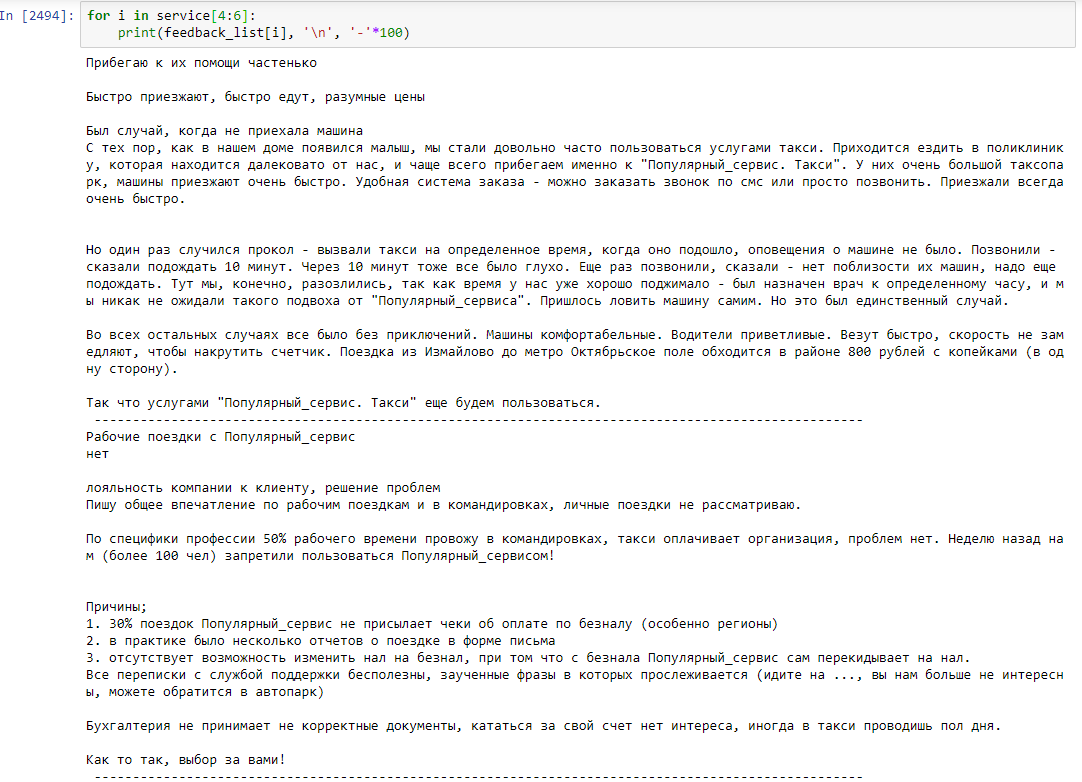


Сортируем все признаки по вероятности принадлежности к каждой из 10 тем и выводим на печать:

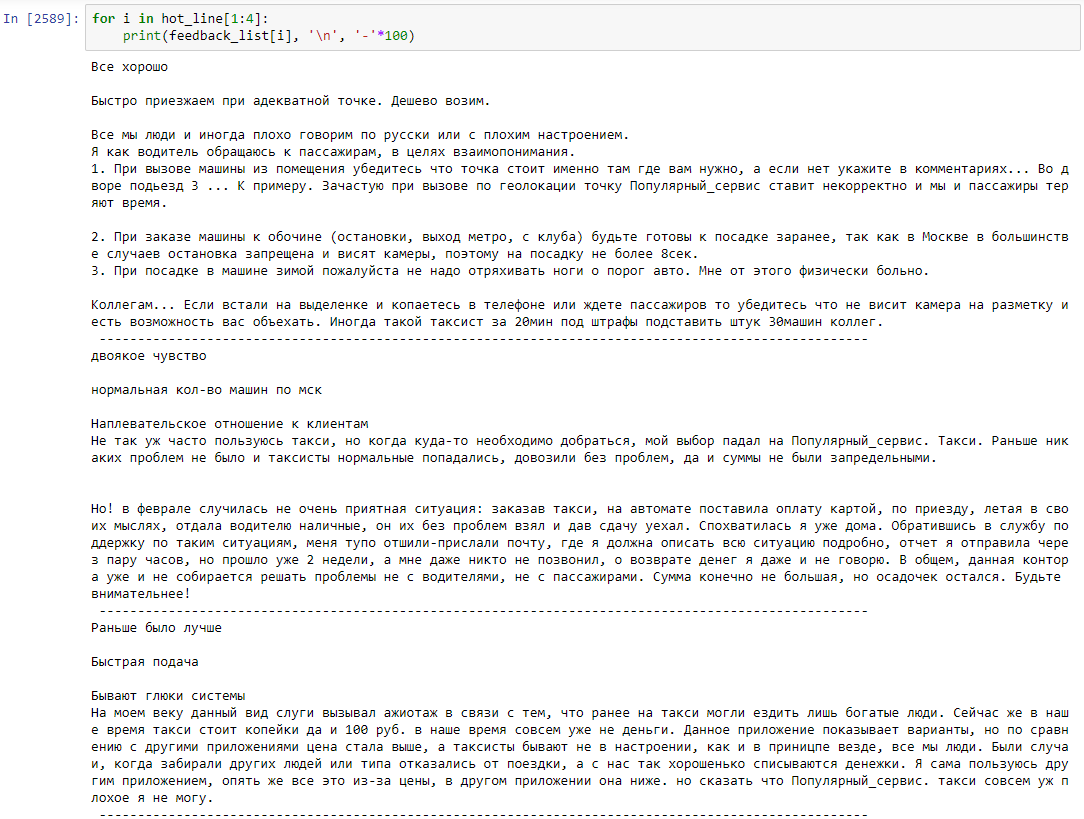


Внимательно смотрим на сформированные темы. Видно, что многие слова попадают сразу в несколько тем. Это объясняется тем, что в рамках одного отзыва пользователь, как правило, описывает несколько аспектов услуги, которые его волнуют. При этом некоторые темы довольно уверенно можно интерпретировать. Рассмотрим лишь 3 из наиболее выраженных на наш взгляд тем. Интерпретацию остальных тем оставим для читателей этой статьи.

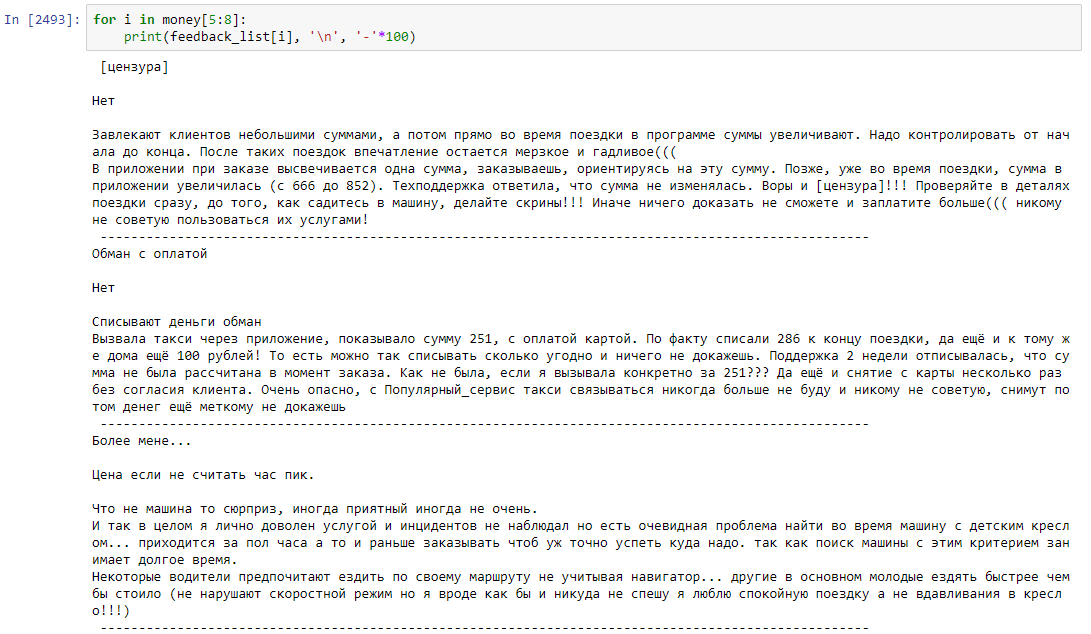
Тема № 1: тут явно что-то про клиентский сервис и информацию на сайте.



Тема №3: оплата и общение с техподдержкой.



Тема №6: оплата услуг.



## Выводы:

Нам удалось с помощью общедоступных инструментов выделить наиболее информативные слова и словосочетания в отзывах клиентов, и что более ценно, без «ручного» прочтения каждого отзыва в отдельности выявить из всего массива информации наиболее популярные темы, которые волнуют пользователей.

Анализ отзывов по мере **TF-IDF** может помочь узнать, какие слова и выражения преобладают в клиентских отзывах. И если там преобладают такие выражения, как «проблема» или «ужасно», стоит об этом знать.

Понятно, что в случае поддающегося прочтению количества отзывов (как в нашем случае с 100 отзывами), имеет смысл просто взять их прочитать, чтобы «услышать» аудиторию. Но когда у вас тысячи отзывов, а руководство хочет получить ответы в сжатые сроки, такой подход может быть неоптимальным.

В следующих статьях мы планируем рассказать о других подходах к анализу текстовой информации, в частности будем создавать модель для предсказания тональности отзывов клиентов на сайте.

**Если у вас есть задачи, связанные с анализом данных, напишите нам и мы поможем извлечь из ваших данных дополнительную ценность**.

Лев Пикулев, руководитель проектов, data scientist.