



МИНОБРНАУКИ РОССИИ

**федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технологический университет
«СТАНКИН»
(ФГБОУ ВО «МГТУ «СТАНКИН»)**

**Институт
информационных
систем и технологий**

**Кафедра
Математики**

Индивидуальное задание

Тема: «Кластерный анализ»

Студент группы ИДБ-19-07:

Касьян А.И.

Преподаватель:

Саркисова И.О.

March 29, 2021

Оглавление

1 Предисловие	3
2 Обозначения	4
2.1 Речевые обозначения	4
2.2 Математические обозначения	4
3 Постановка задачи	5
4 Анализ данных	6
4.1 Пропущенные данные	7
4.1.1 Пропущенные записи в director	7
4.1.2 Пропущенные записи в cast	9
4.1.3 Пропущенные записи в date_added	9
4.1.4 Пропущенные записи в rating	10
4.1.5 Пропущенные записи в country	10
4.2 Записи-дупликаты	11
4.3 Базовый анализ данных	11
4.4 Возрастной ценз	12
4.5 Даты	13
4.6 Жанры	15
4.6.1 Мера схожести	15
4.7 Жанры №2	16
4.8 Мини-обобщение	17
4.9 Продолжительность	17
4.10 Актёрский состав	19
4.10.1 Ведьмак	19
4.10.2 The Last of Us	21
4.10.3 Подитог	23
4.11 Синопсис (Описание)	23
4.12 Итоги анализа	25
5 Кластеризация	26
5.1 Нормализация данных	27
5.1.1 Квантильная нормализация	27
5.2 Графическое представление кластеров (уменьшение размерности)	28
5.2.1 Principal Components Analysis (PCA)	28
5.2.2 t-distributed stochastic neighbor embedding (t-SNE)	31
5.3 Первоначальное количество кластеров	35
5.3.1 K-means	35
5.3.2 Разбиваем на 3 кластера	35
5.3.2.1 Elbow Method	36
5.3.2.2 Silhouette method	36
5.3.2.3 Главная проблема K-means	38
5.3.2.4 Другие алгоритмы кластеризации	38
5.3.2.5 Density-based spatial clustering of applications with noise	39
5.3.2.6 Применение DBSCAN	41
5.4 Кластеризация синопсисов	42
5.4.1 Deep Embedding Clustering	43
5.4.1.1 Кластеризация с KL-divergence	43
5.4.1.2 Инициализация параметров	45
5.4.1.3 Автоэнкодер	45
5.4.2 Применение шайтан-машины (DEC)	46
5.5 Итоговая кластеризация	49

5.5.1	Спектральная кластеризация	49
5.5.2	Применение Спектральной кластеризации	51
6	Оценка и итог	52
7	Послесловие	53

1 Предисловие

Здравствуйте! Я ввёл данную секцию для прояснения некоторых моментов в данной работе и буду очень благодарен, если вы уделите ей внимание:)

Первое, я хочу оформить данную работу в виде некой исследовательской, но вольном формате, в виде монолога, изложения всех идей которые у меня появляются во время работы.

Я постараюсь по-меньше ссылаться на сторонние ресурсы, объясняя различные темы самостоательно. Но без единой отсылки не получится, т.к. и я не всеведущ и, если разбираться в каждой теме достаточно детально данная работа, наверное, не закончится.

Все ссылки будут на англоязычные ресурсы, поскольку в большинстве случаев описание на английском более детальное и понятное, да и информации на английском во много раз больше.

Инструментарий, который я буду использовать:

- Tableau - для визуализации данных.
- Python - для анализа, визуализации и самой кластеризации.

Весь код, который был написан для этой работы можно найти, перейдя по [ссылке](#).

2 Обозначения

Я очень рекомендую прочитать данную секцию, чтобы в дальнейшем быть уверенными, что мы с вами говорим об одних и тех же вещах.

2.1 Речевые обозначения

Я уже привык к профессиональным терминам, поэтому на всякий случай я решил их один раз расписать, чтобы не было непоняток и путаницы.

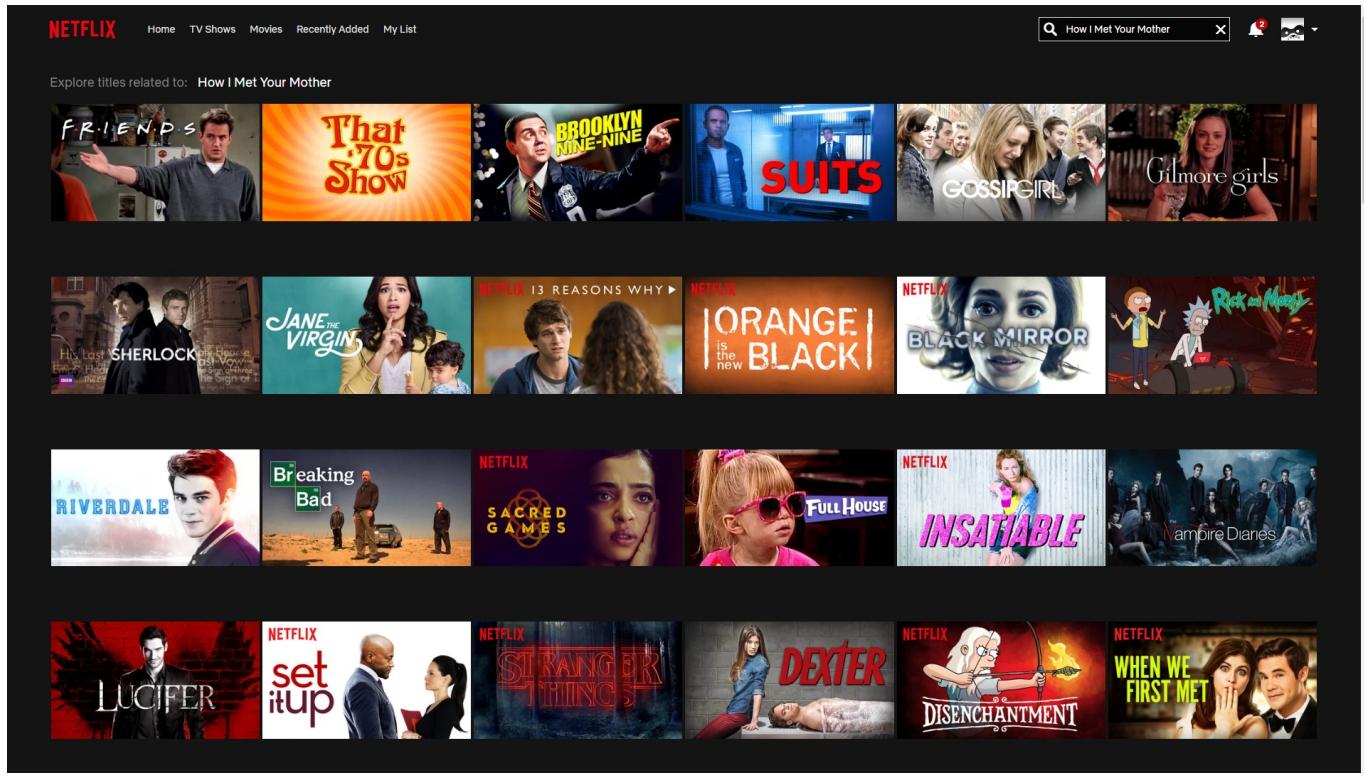
- **датасет, датафрэйм** - технически эти термины обозначают разное. Но в данной работе они равноправны. Они обозначают набор данных, генеральную совокупность.
- **сабсет, сэмпл, слайс** - в данной работе они обозначают некую выборку по определённому/-ым признакам из основного набора данных.
- **фитча** - какой-либо признак/характеристика набора данных либо выборки.
- **элемент** - основной исследуемый объект. Например, в данной работе по кластеризации фильмов и сериалов, элемент - фильм или ТВ-сериал. Соответственно набор характеристик для элементов одинаковый.
- **эмбединг** - на самом деле это слово позаимствовано из NLP. Но в данном контексте оно используется как векторное представление какого-либо объекта
- **паттерн** - Паттерн понимается в этом плане как повторяющийся шаблон или образец. Элементы паттерна повторяются предсказуемо.

2.2 Математические обозначения

- \mathcal{D} - в дальнейшем я буду использовать данный символ для датасета. Если речь идёт о сабсете, то обозначение будет следующим \mathcal{D}_* , где звёздочка будет обозначать признак или набор признаков по которому/-ым сабсет был выбран.
- χ^2 - [хи-статистика Пирсона](#).
- V, \tilde{V} - [V статистика Крамера](#), Скорректированная V статистика Крамера соответственно.
- P - целевое распределение
- Q - вычисляемое распределение (в одном конкретном моменте матрица схожести)
- μ - центройд
- θ - обучаемые параметры
- A - функция-индикатор
- SC - [Silhouette coefficient](#)

3 Постановка задачи

Для начала определимся с задачей. Я решил взять подзадачу не рассмотренную в моём реферате. Если там я рассматривал сильно упрощённую рекомендательную систему для каждого конкретного пользователя, то здесь я хочу рассмотреть рекомендательную систему связных элементов. Область исследования - медиа, а точнее фильмы и ТВ-сериалы. Т.е. нам потребуется найти схожие фильмы. В промышленных масштабах это выглядит как-то так:



Или так:

A screenshot of the ökko app interface. At the top, there's a navigation bar with "öko" and "ПРОМОКОД" buttons. Below it, a search bar and a menu with "Рекомендации", "Новинки", "Каталог", "Подборки", and "Подписки". The main content area is titled "Похожие на сериал «Как я встретил вашу маму»" with a right-pointing arrow. It displays eight show cards: "Офис" (Office), "Два с половиной человека" (Two and a Half Men), "Бруклин 9-9" (Brooklyn Nine-Nine), "Американская семейка" (American Family), "Лузеры" (Lazers), "Клава, давай!" (Clava, Come On!), "Бесстыдники" (Shameless), "Университет" (University), and two additional cards at the bottom: "В Филадельфии всегда солнечно" (Philadelphia Always Sunshine) and "Город Хищниц" (City of Cannibals). A purple message icon is in the bottom right corner.

Теперь нам надо собрать данные. Я не стал их собирать вручную, а взял уже готовый датасет с Kaggle. Датасет достаточно новый, обновлялся 2 месяца назад.

Этот набор данных состоит из телешоу и фильмов, доступных на [Netflix](#) по состоянию на 2020 год. Набор данных собирается из [Flixable](#), сторонней поисковой системы Netflix. В 2018 году они выпустили интересный [отчет](#), который показывает, что с 2010 года количество фильмов на потоковом сервисе уменьшилось более чем на 2000 наименований, а количество телешоу увеличилось почти втрое. Будет интересно изучить, какие другие идеи можно получить из того же набора данных.

Изначальный датасет выглядит следующим образом

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	TV Show	3%	Brazil	August 14, 2020	2020	TV-MA	4 Seasons
1	s2	Movie	7:19	Mexico	December 23, 2016	2016	TV-MA	93 min
2	s3	Movie	23:59	Singapore	December 20, 2018	2011	R	78 min
3	s4	Movie	9	United States	November 16, 2017	2009	PG-13	80 min
4	s5	Movie	21	United States	January 1, 2020	2008	PG-13	123 min

Рассмотрим каждую колонку и что она в себя включает.

- **show_id** - уникальный идентификатор элемента.
- **type** - тип элемента. Всего есть два типа:
 - **TV Show** - многосерийные произведения, такие как сериалы, ТВ-шоу и т.п.
 - **Movie** - полнометражные произведения, такие как фильмы.
- **title** - название произведения.
- **director** - ФИО режисёра.
- **cast** - перечисленные через запятую ФИО актёрского состава.
- **country** - перечисленные через запятую названия стран.
- **date_added** - т.к. мы рассматриваем датасет Netflix'а, то в данной колонке находится дата добавления произведения на сервис.
- **release_year** - год выпуска произведения. Для сериалов - год выпуска последнего сезона.
- **rating** - возрастное ограничение. Возможны следующие значения: **TV-MA, R, PG-13, TV-14, TV-PG, NR, TV-G, TV-Y, TV-Y7, PG, G, NC-17, TV-Y7-FV, UR**
- **duration** - продолжительность произведения. В целом тут два базовых значения, для фильмов - количество минут, а для сериалов - количество сезонов.
- **listed_in** - перечисленные через запятую жанры к которым произведение относится.
- **description** - синопсис произведения.

Наш датасет содержит 7787 записей, чего вполне достаточно для нас, в ознакомительных целях. Чуть ранее у меня была идея скачать [базу данных IMDB](#), и взвешанный рейтинг фильмов, но, во-первых, из-за сильной детальности IMDB появлялось очень много коллизий, и, во-вторых, некоторых фильмов не было в базе IMDB. Тем самым я терял, практически, 1700 записей. Т.к. рейтинг на IMDB показывает на сколько "качественное" произведение, то это очень важный показатель для рекомендации конкретному пользователю. А мы ищем именно максимально похожие произведения, а вот среди них можно выбрать топ лучших, так сказать. Поэтому IMDB-рейтинг не самая значащая характеристика на данный момент, по крайней мере она не стоит 1700 потенциально потерянных записей. Далее, может быть, я добавлю функционал поиска топ *n* фильмов в кластере, но это потом. Так, как набор данных у нас есть - их нужно проанализировать.

4 Анализ данных

Начнём с анализа, а точнее с очистки данных. Первое, что пришло в голову - посмотреть:

- Пропущенные значения
- Записи-дубликаты
- Типы данных

4.1 Пропущенные данные

В нашем датасете есть 5 полей с пропущенными данными:

- director: 2389 записей, что составляет 30% от всего датасета
- cast: 718 записей, что составляет 9% от всего датасета
- country: 507 записей, что составляет 6% от всего датасета
- date_added: 10 записей, что составляет 0.1% от всего датасета
- rating: 7 записей, что составляет 0.08% от всего датасета

Ну, понятно, что 0.08% или 0.1% это не то чтобы сильно "весомые" пропуски. А вот 30%, это да, с этим надо что-то решать. Но прежде чем применять разные методологии замены пропущенных данных, давайте оценим на сколько значимо ФИО режисёра для наших данных.

4.1.1 Пропущенные записи в director

Напомню, что мы определяем схожесть фильмов и сериалов между собой, и нам нужно выяснить на сколько на эту схожесть влияет режисёр. Переходим к тесту.

Вопрос: есть ли сильная взаимосвязь между ФИО режисёра и схожестью фильмов?

Чтобы ответить на этот вопрос, надо определить, что такое схожесть фильмов. Ну интуитивно, можно предположить, что схожие фильмы имеют схожие жанры, схожий смысл синопсиса, иногда схожий актёрский состав. Ещё бы я добавил возрастной ценз, но это не основной параметр схожести, а дополнительная фильтрация, так сказать. Для анализа будет достаточно жанров, поскольку жанры и синопсисы самые репрезентативные. Ну о синопсисах мы поговорим [позже](#).

Начнём с жанров, я выделил все жанры, представленные в столбце `listed_in`. Для каждого жанра мы посчитали количество фильмов этих жанров для каждой группы (1 - есть ФИО режисёра; 2 - нет ФИО режисёра).

	International TV Shows	TV Dramas	TV Sci-Fi & Fantasy	Dramas	...	TV Thrillers
director	99	52	4	3	...	3
nan director	1100	652	72	24	...	47

- H_0 : связь между фильмов не зависит от наличия ФИО режисёра
- H_a : связь между фильмов зависит от наличия ФИО режисёра

Мы имеем две номинальные переменные. И хотим проверить насколько вероятно, что разница между группами (1 - есть ФИО режисёра; 2 - нет ФИО режисёра) возникла случайно. Для теста выбираем [критерий Хи-квадрат Пирсона](#). Напомню, что это такое и как мы это получаем. Рассмотрим таблицу сопряжённости:

χ^2 - оценивает статистическую значимость между ожидаемой частотой и исследуемой частотой для каждой категории в таблице сопряжённости.

$$\chi^2 = \sum_{i=1}^n \frac{(O_i - E_i)^2}{E_i}$$

	International TV Shows	TV Dramas	total
director	99	52	151
nan director	1100	652	1752
total	1199	704	1903

где

- O_i - количество (частота встречаемости) наблюдений категории i
- E_i - ожидаемое (теоретически) количество категории i .

$$E_i = \frac{\text{row total} \times \text{column total}}{\text{total}} = \frac{\sum_{k=1}^n O_{k,i} \times \sum_{k=1}^n O_{i,k}}{\sum_{k=1}^n \sum_{j=1}^n O_{k,j}}$$

Перенесём на наш мини-пример:

$$E_{\text{director}} = \frac{151 \times 1199}{1903} \approx 95.1387$$

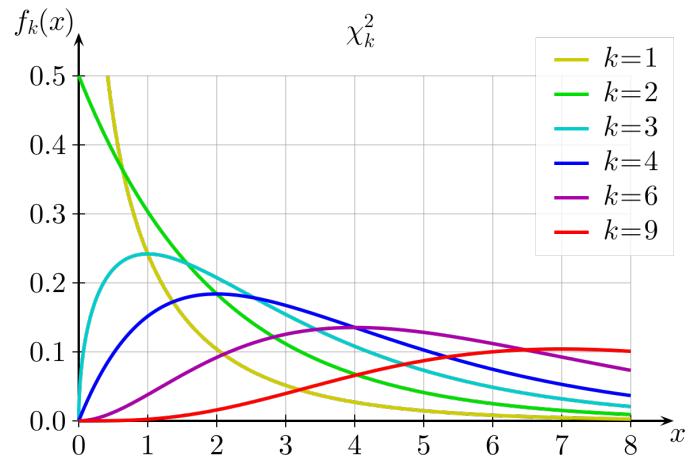
$$E_{\text{nan director}} = \frac{1752 \times 704}{1903} \approx 648.1387$$

Вычисляем статистику:

$$\chi^2 = \sum_{i=1}^n \frac{(O_i - E_i)^2}{E_i} = \frac{(99 - 95.1387)^2}{95.1387} + \frac{(52 - 95.1387)^2}{95.1387} + \frac{(1100 - 648.1378)^2}{95.1387} + \frac{(652 - 648.1378)^2}{648.1378} = 334.76$$

Вычисляем количество степеней свободы $k = (\text{количество строк} - 1)(\text{количество столбцов} - 1) = 1$. И подставляем полученную статистику в χ^2 -распределение.

И вот тут появляется проблема, при таком большом значении статистики, p стремится к нулю, и оно перестаёт быть показательным. На самом деле это достаточно большая проблема на практике, когда набор данных достаточно велик, (нету каких-то точных границ, всё зависит от распределения) такой статистический показатель как $p - value$ перестаёт быть показательным. На эту тему ещё в 2013 году была опубликована исследовательская работа [Too Big to Fail: Large Samples and the p-Value Problem](#). Там всё очень хорошо расписано, советую прочитать. Чтобы обойти эту проблему мы будем использовать другой показатель, основанный на χ^2 Пирсона. **Cramer's V статистика**. Это мера связи между двумя номинальными переменными, дающая значение от 0 до +1. Cramer's V варьируется от 0 (что соответствует отсутствию ассоциации между переменными) до 1 (полная ассоциация) и может достигать 1 только тогда, когда каждая переменная полностью определяется другой. Проблема χ^2 в получении слишком высокого значения статистики, тем самым делая $p - value$ незначимым. Крамер же минимизировал данное значение следующим образом:



где

- φ - Фи коэффициент
- n - общая сумма наблюдений
- k - количество столбцов
- r - количество строк

V Крамера может быть сильно предвзятой оценкой по сравнению с её популяционным аналогом, и будет иметь тенденцию переоценивать силу ассоциации. Коррекция, используя приведенные выше обозначения, задается следующим выражением:

$$\tilde{V} = \sqrt{\frac{\tilde{\varphi}^2}{\min(\tilde{k}-1, \tilde{r}-1)}}$$

где

$$\begin{aligned}\tilde{\varphi}^2 &= \max\left(0, \varphi^2 - \frac{(k-1)(r-1)}{n-1}\right) \\ \tilde{k} &= k - \frac{(k-1)^2}{n-1} \\ \tilde{r} &= r - \frac{(r-1)^2}{n-1}\end{aligned}$$

Затем \tilde{V} оценивает тот же сабсет, что и V Крамера, но обычно с гораздо меньшей [среднеквадратичной ошибкой](#). Исправление заключается в условии независимости

$$E[\varphi^2] = \frac{(k-1)(r-1)}{n-1}$$

Применив V Крамера к нашему мини-примеру, мы получаем:

$$\begin{aligned}\varphi^2 &= \frac{334.76}{1903} = 0.175 \\ \tilde{\varphi}^2 &\max\left(0, \varphi^2 - \frac{(2-1)(2-1)}{1903-1}\right) = 0.174 \\ \tilde{k} &= 2 - \frac{(2-1)^2}{1902} = 1.9995 \\ \tilde{r} &= 2 - \frac{(2-1)^2}{1902} = 1.9995 \\ \tilde{V} &= \sqrt{\frac{0.175}{\min(1.9995, 1.9995)}} = \sqrt{0.08752} = 0.2958\end{aligned}$$

Т.е. наши жанры в мини-примере (International TV Show; TV Dramas) на 30% зависят от наличия ФИО режисёра.

Проведя тест V Крамера для целого сабсета $\mathcal{D}_{\text{genre}\&\text{director}}$ я получил $\tilde{V} = 0.907 \approx 90\%$. Т.е. жанры зависят от группы на 90%, что мне кажется многовато, но выкидывать данный столбец будет глупо. Но т.к. у нас нету пропущенных ни описания ни жанров, то пока мы оставим данный столбец в текущем виде, точнее заменим все пропущенные значения на "No Director", для более удобного анализа.

4.1.2 Пропущенные записи в cast

Если разбить актёров на каждого отдельного актёра а не на каст, да и даже если на каст мы получим минимум 6832 уникальных группы. Т.е. получится слишком диверсионная таблица сопряжённости и оценка не будет показательной. Далее при анализе мы взглянем, важны ли для нас пропущенные переменные в колонке cast. А пока объединим их в группу "No Cast".

4.1.3 Пропущенные записи в date_added

Здесь всё легко, даже не проводя статистический анализ. Он здесь не нужен. Мы просто избавимся от записей с промущенными значениями, поскольку таковых всего 10. Теоретически, можно было бы сравнить разницу между годом релиза и датой добавления на сервис. И выдвинуть предположения на тему "качества" фильмов, т.к. сам знаю, что Netflix не любит убыточные проекты. Качество взято в кавычки, т.к. качество по мнению глав Netflix - то, что приносит деньги (а сейчас это

ещё и очень толерантно), а качество по мнению зрителя - интересный сюжет, качественная съёмка, звукопостановка, актёрская игра и т.п. Вот пример Netflix 2020: количество LGBTQ Movies больше чем научных или научно-фантастических фильмов. Так же можно было найти среднюю скорость принятия решения глав Netflix, продифференцировать полученную функцию, чтобы получить динамику, в зависимости от периода. Далее определить меру схожести, в зависимости от скорости роста добавления на сервис. Вобщем с этой фитчей можно ещё поиграться, но об этом в [анализе](#).

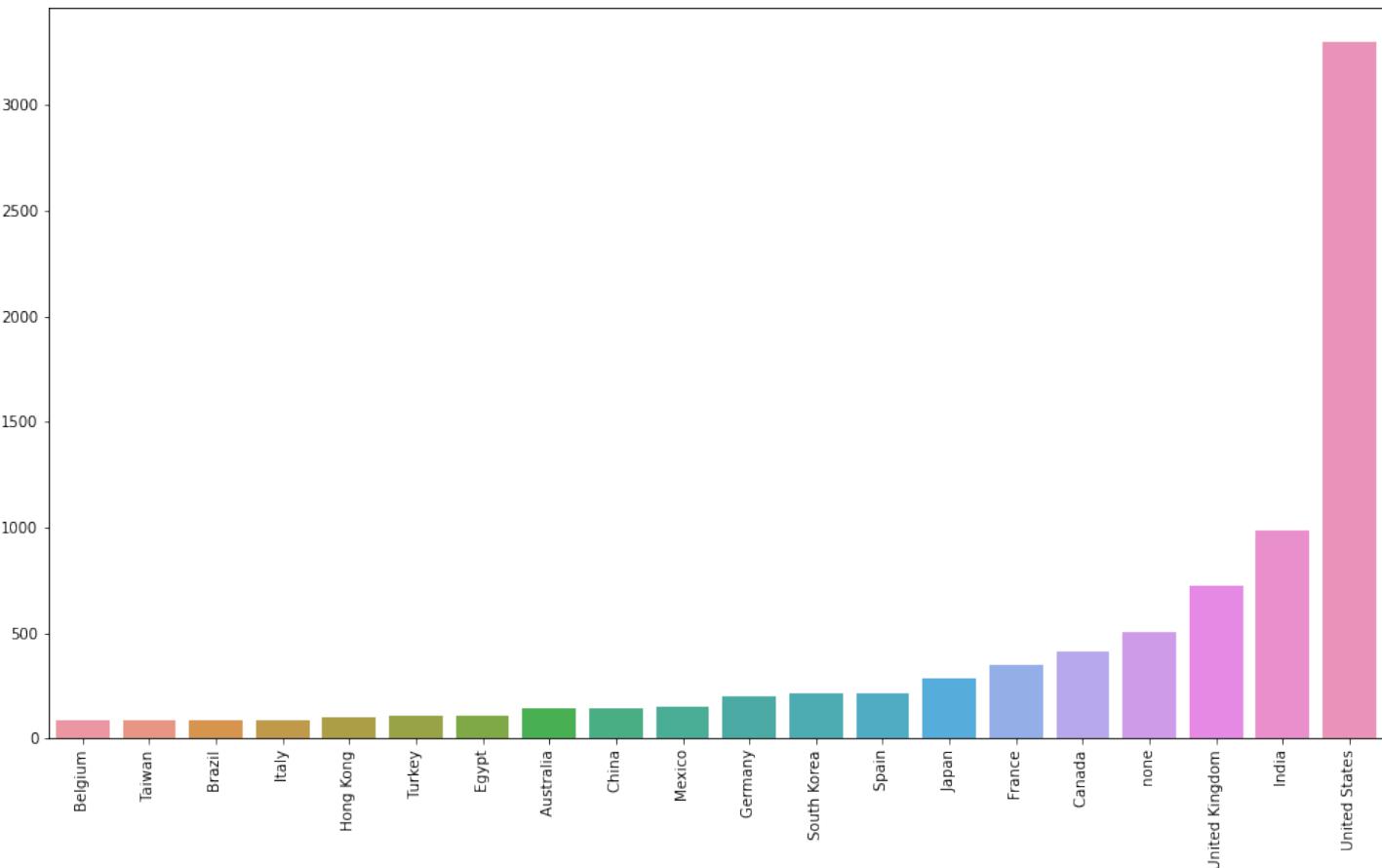
4.1.4 Пропущенные записи в rating

Здесь всё тоже очень легко, с помощью информации о возрастных цензах с [Amazon](#) и непосредственно Netflix находим недостающие рейтинги. Пропущенные рейтинги получились следующие:

- 13th: A Conversation with Oprah Winfrey & Ava DuVernay: TV-PG
- Gargantia on the Verdurous Planet: PG-13
- Little Lunch: PG
- Louis C.K. 2017: TV-MA
- Louis C.K.: Hilarious: TV-MA
- Louis C.K.: Live at the Comedy Store: TV-MA
- My Honor Was Loyalty: PG-13

4.1.5 Пропущенные записи в country

Всего мы имеем 6% пропущенных значений в данной колонке. Взглянем на распределение данных по странам



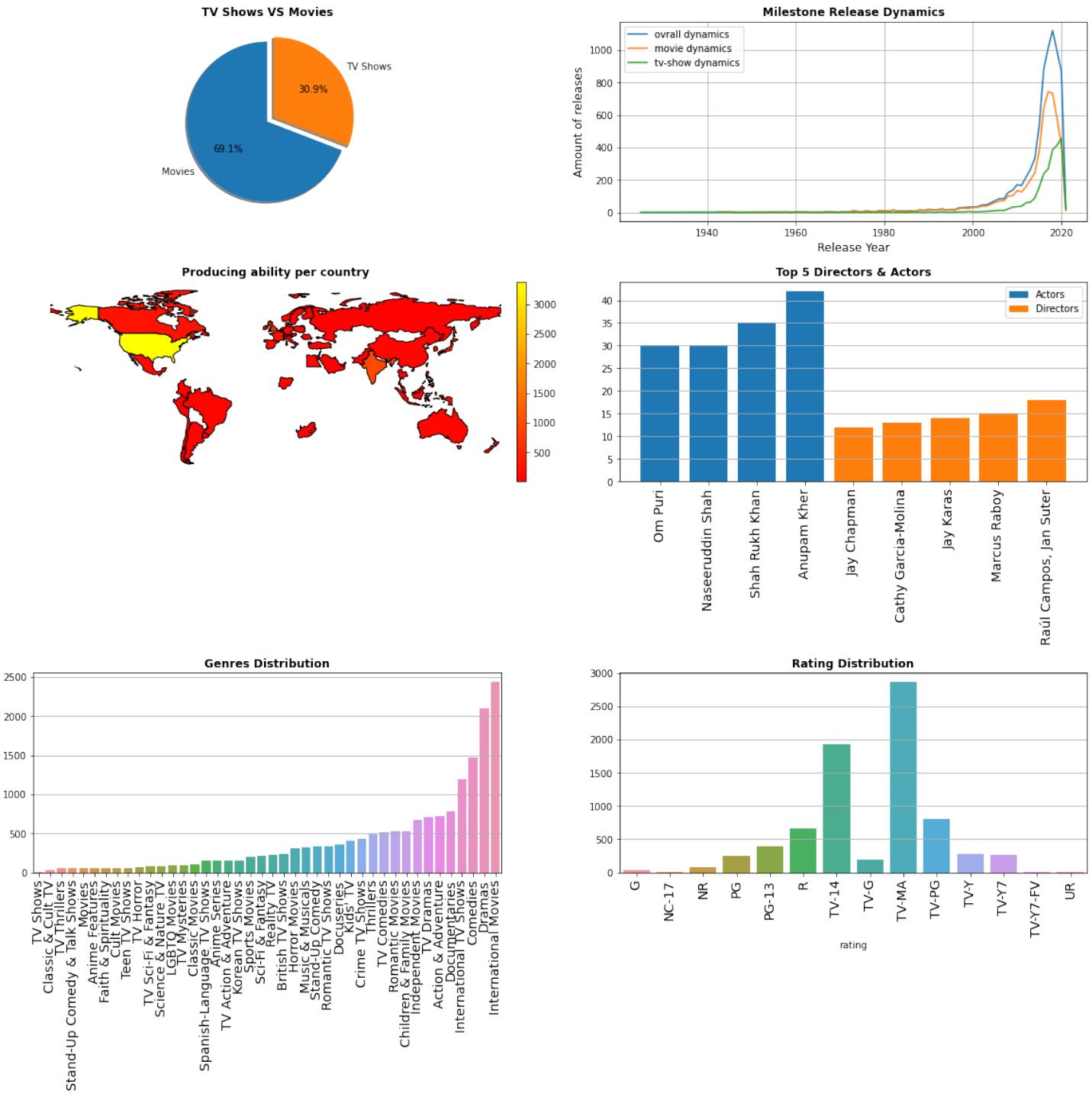
Мда, думаю доминация США очевидна, и если дополним ещё 6% данных, ситуация особо не поменяется.

4.2 Записи-дупликаты

дупликатов в датасете не было.

4.3 Базовый анализ данных

Для начала взглянем на данные и их распределения, которые пришли мне в голову



Разберём, что видим:

- ТВ-шоу и сериалы составляют 30% от всего датасета, соответственно фильмы составляют 70%. Проверив, посредством χ^2 удостоверяемся, что выборка репрезентативна.
- Больше всего продукции, допускаемых на Netflix производят США.
- Где-то с 2005 года производство и выпуск кино-продукции очень сильно возросло.
- Как я уже говорил, т.к. Netflix гонится за прибылью, мы можем определять по количеству представителей каждого жанра заинтересованность зрителя в том или ином жанре.

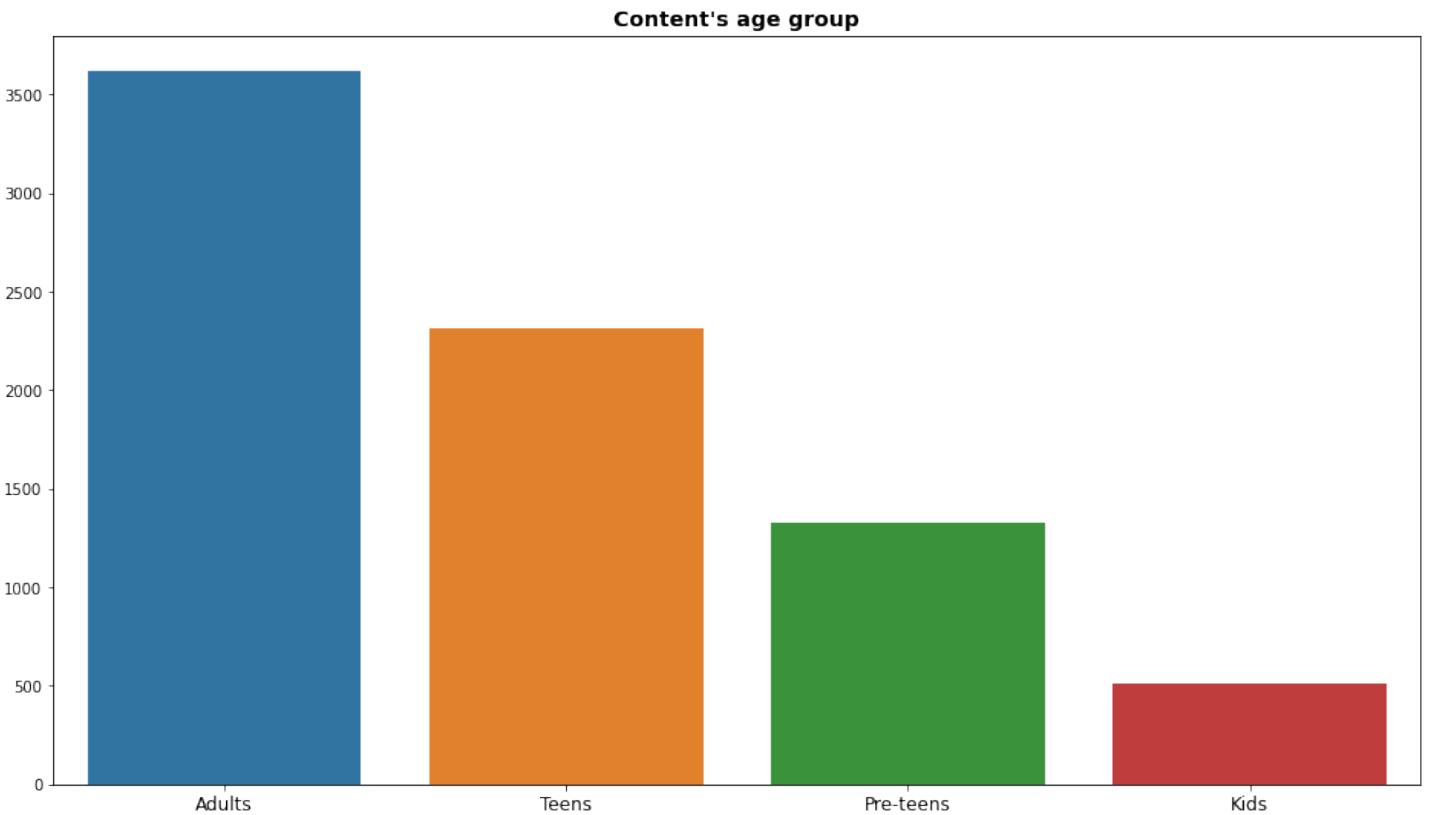
- Так же имеем возрастной рейтинг (сертификат), который ни чего не говорит человеку, далёкому именно от киноиндустрии. Рассмотрим это позже.
- В дополнение я взял топ 5 режисёров и актёров. Так же рассмотрим позже.

Пока я визуализировал данные, я добавил новую фитчу **main_country**. Она показывает основную страну, это первая представленная страна у каждого элемента в столбце **country**.

4.4 Возрастной ценз

Начнём с возрастного рейтинга. Т.к. большинству незнакомы возрастные ограничения в представленном виде, разделим их на группы. Каждая группа подразумевает, что произведения с данным рейтингом могут смотреть, представители данной группы и старше. Ну понятно, что взрослые могут смотреть детские мультики:

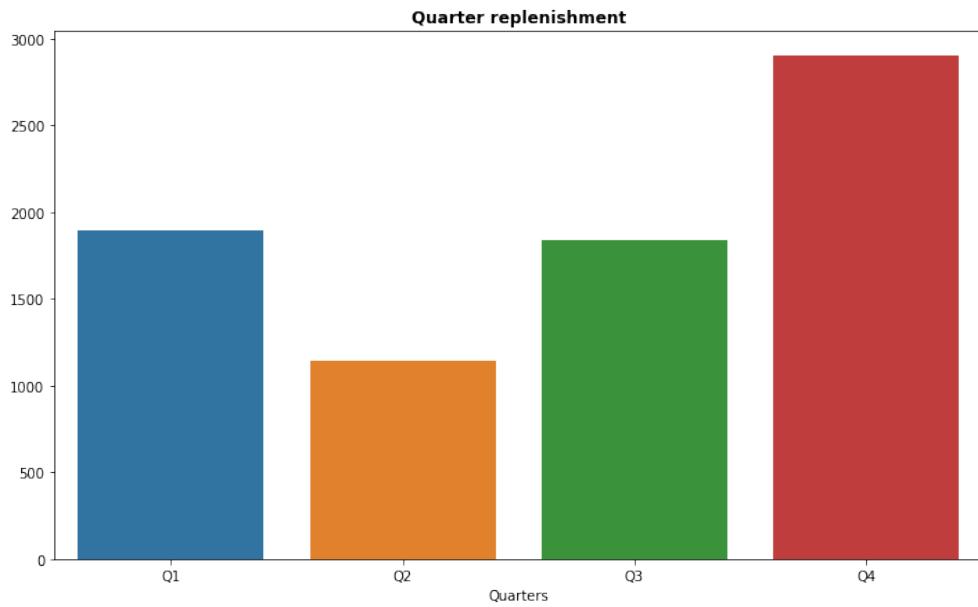
$$\begin{aligned}
 \text{Kids (Дети)} & - \left\{ \begin{array}{l} \text{G} \\ \text{TV-G} \end{array} \right. & \text{Pre-teens (Предподростковый возраст)} & - \left\{ \begin{array}{l} \text{PG} \\ \text{TV-Y7} \\ \text{TV-Y7-FV} \\ \text{TV-PG} \end{array} \right. \\
 \text{Teens (Подростки)} & - \left\{ \begin{array}{l} \text{TV-14} \\ \text{PG-13} \end{array} \right. & \text{Adults (Взрослые)} & - \left\{ \begin{array}{l} \text{TV-MA} \\ \text{R} \\ \text{NR} \\ \text{UR} \\ \text{NC-17} \end{array} \right.
 \end{aligned}$$



Сделаем пометочку, что на мнужно поменять тип данных, данной фитчи, т.к. на данный момент она записана как **object**, а на самом деле это категория. Также логично, что данные 5 групп будут иметь большую корреляцию с жанрами, нежели полноценное разбиение на возрастной ценз, следовательно мы можем избавиться от колонки **rating**. На самом деле данное утверждение можно проверить, но т.к. вариация у **rating** и **age_group** разная, да и нельзя сказать, что они независимы, то провести параметрический t-тест не вйдет, есть его замена **t-тест Уэлча**, но не уверен, что этот тест будет сильно показательным.

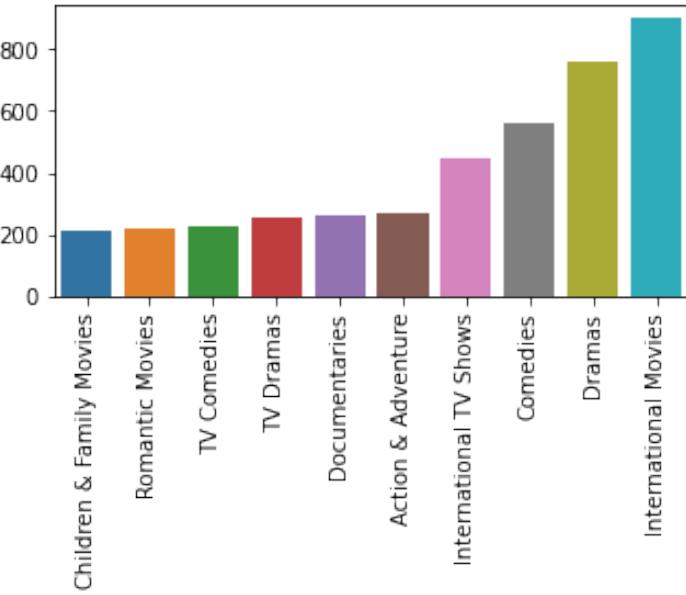
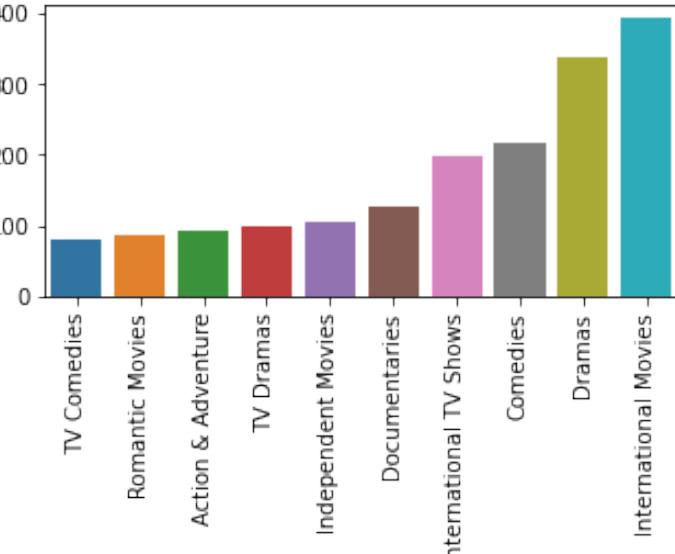
4.5 Даты

Я решил объединить в данной секции и дату релиза и дату добавления на сервис. Для начала я хочу разбить дату добавления на сервис на отдельные части: **день**, **месяц**, **год**. Сразу же избавляемся от колонки **date_added**, и добавим кварталы. Если разобрать кварталы, то выходит, что в Q4 идёт самый большой приток контента



year_added	quarter_max	count_max	quarter_min	count_min
2008	Q1	2	Q1	2
2009	Q4	2	Q4	2
2010	Q4	1	Q4	1
2011	Q4	12	Q3	1
2012	Q4	2	Q1	1
2013	Q4	7	Q1	1
2014	Q4	15	Q2	3
2015	Q4	49	Q1	10
2016	Q4	207	Q2	41
2017	Q4	425	Q2	190
2018	Q4	643	Q2	196
2019	Q4	857	Q2	346
2020	Q4	679	Q2	355
2021	Q1	117	Q1	117

И по таблице видно, что Q4 - это самый пополняемый контентом квартал, а Q2 - чередуется с Q1 как самые не пополняемые, на протяжение 12 лет. Хотя самыми показательными являются 2014 - 2020 гг., т.к. с 2008 - 2014 сервис работал более избирательно, что вполне заметно по [росту их доходов](#). Но, т.к. мы не выясняем причину роста их доходов, а пытаемся выявить фитчи, которые максимально точно определяют меру схожести, то нам скорее всего кварталы не понадобятся. Предположим, что Netflix работает в интересах зрителя, и добавляет тот контент, который интересен зрителю, однако мы на вряд ли сможем разглядеть какое-то изменение в интересах на протяжении года. Это можно доказать, [посмотрев](#) на соотношение жанров, добавленных в 4-ом квартале и во 2-ом квартале. Если бы причина была в жанре, то 1 - 4 место в топе занимали бы разные представители, однако поскольку они одинаковы, то мы можем предположить, что главы компаний по какой-то причине, **не связанный** с интересами зрителя выкладывают где-то на 400 произведений меньше во 2-ой квартал, нежели чем в 4-ый.

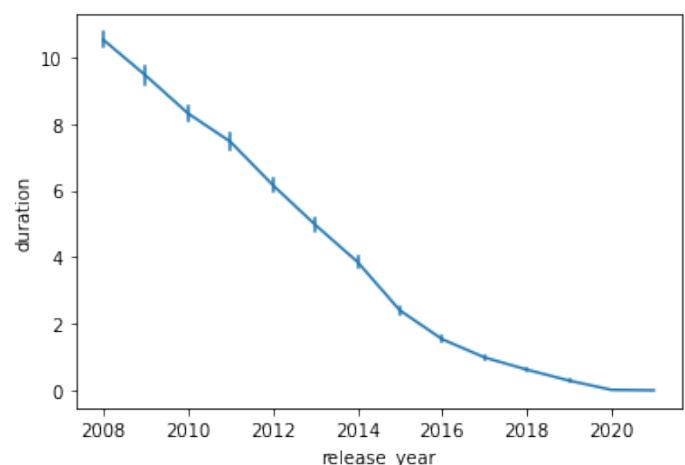
Q4 TOP 10 Genres**Q2 TOP 10 Genres**

Так, я сейчас залез в жанры, чего не хотелось бы здесь делать. Поэтому посмотрим, что ещё можем сделать с нашими датами. Ну как я и говорил выше, можно посмотреть среднюю скорость принятия решений Netflix, и возможно причина скрывается в самих произведениях.

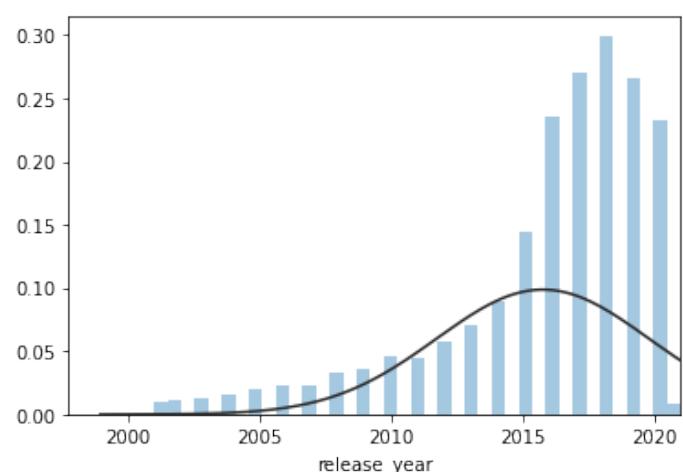
Судя по [Википедии](#) Netflix был основан в 1997 году. Я решил взять интервал с 2008 по 2020 т.к. Netflix именно в 2008 был образован, как стриминговый сервис (ранее продукция доставлялась почтой в виде DVD). Именно этим объясняется такой резкий рост, показанный на графике [Milestone Release Dynamics](#). Судя же по данному графику мы можем заметить, что скорость принятия решения о добавлении контента последовательно росла с годами. Опять-таки, найдя для каждого года самый популярный жанр, у меня вышло следующее

- 2008: Comedies
- 2009: Comedies
- 2010: Dramas
- 2011: Comedies
- 2012: Comedies
- 2013: Dramas
- 2014: Dramas
- 2015: Dramas
- 2016: Dramas
- ...

И на последок взглянем на частоту загрузки контента на сервис, начиная с 2000 года. Чуть выше я писал, что Netflix доставлял DVD-диски по почте до 2008. Однако уже в тот момент у них была подписка, правда, на бессрочной основе. В 2001 году лопнул [пузырь доткомов](#), плюс ещё произошли теракты 11 сентября, что вынудило сократить штат компании, однако продажи DVD-плееров в конце концов взлетели, поскольку они стали более доступными, продаваясь по цене примерно 200 долларов за штуку, став одним из



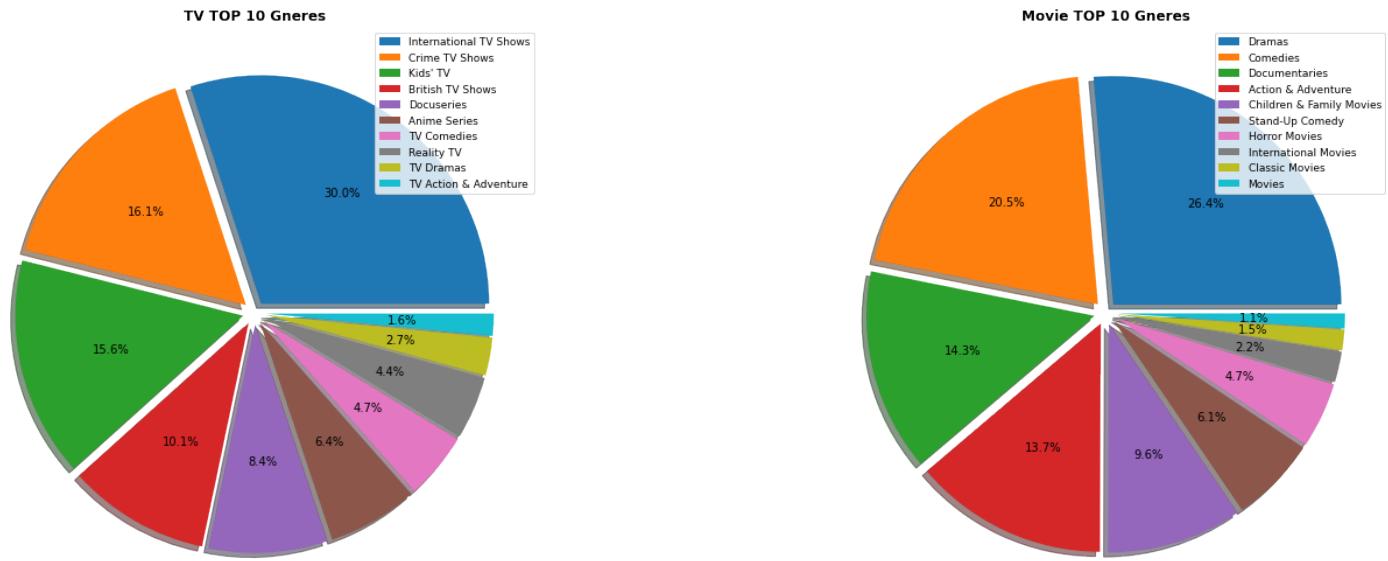
Какого-то резкого изменения в скорости принятия решения при изменении жанра нету. Чтобы это доказать - можно продифференцировать участки и сравнить с общей динамикой. Но и по графику всё видно, да, там есть совсем небольшое снижение в скорости в промежутке с 2010-2011, но не думаю, что оно как-то сильно связано с жанром.



самых популярных рождественских подарков в этом году. Это обстоятельство способствовало новому росту продаж подписки Netflix в начале 2002 года.

4.6 Жанры

Жанры-жанры, всё с ними связано, давайте уже взглянем на них. Прежде всего, т.к. записи в колонке `listed_in` представлены в виде строки, то сначала мы создадим новую колонку `genres`, которая хранит в себе жанры в виде списка, а потом создадим дополнительную колонку `main_genre`, которая хранит в себе главный жанр (обычно его указывают первым в списке жанров). На самом деле жанры очень показательны, и сами по себе они уже являются некими кластерами, а точнее эмбедингами. Ранее мы рассматривали жанр, как отдельную фитчу, однако у неё есть два крупных подразделения: TV Show и Movie.



И давайте немного уделим времени синей доле, а т.е. жанру, который встречается чаще чем остальные. На диаграмме выше мы рассмотрели пропорции жанров в интервале от 1925 до 2021 гг. Однако, т.к. мы ищем схожесть произведений, давайте рассмотрим как менялись жанры на протяжении лет. Я же взял интервал от 2000 до 2021 года, и посмотрел самый просматриваемый жанр в каждом году. В целом у меня получилось следующее:

- **Movie:** с 2000-2012 это были комедии, а с 2013-2020 это драмы.
- **TV Show:** с 2000-2020 превалируют ТВ-шоу.

Что вобщем-то доказывает, что год выпуска это достаточно важный показатель в мере схожести.

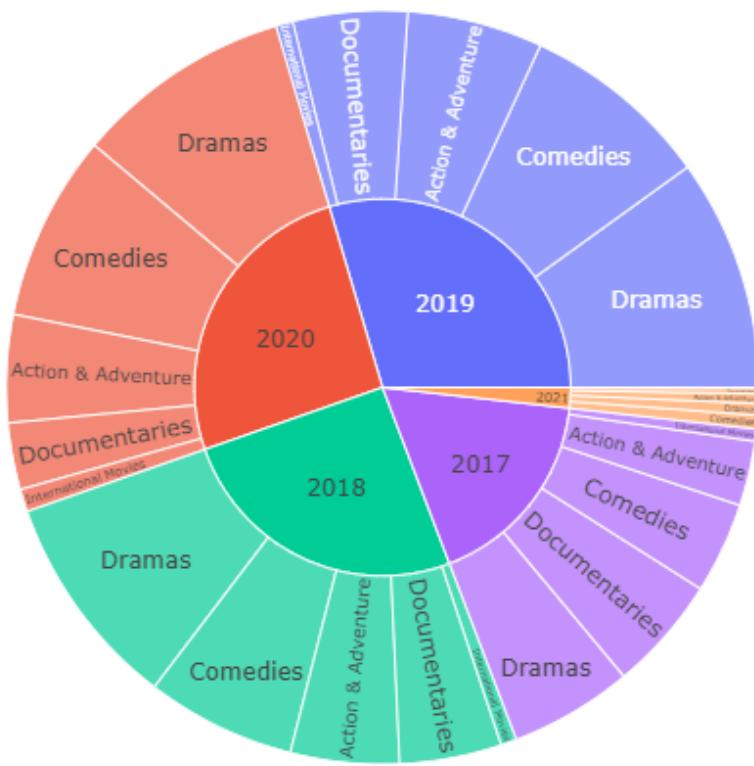
4.6.1 Мера схожести

Я тут так много полагаюсь на меру схожести, не пояснив что это за мера такая. Мера схожести - это функция $s : E \times E \rightarrow \mathbb{R}$ которая берёт пару эмбедингов и возвращает скаляр, измеряющий их сходство. Эмбединги могут использоваться для определения схожести следующим образом: при встраивании запроса $q \in E$ мы ищем вложения элементов $x \in E$, которые близки к q , то есть вложения с высокой степенью сходства $s(q, x)$. Чтобы определить степень сходства, большинство рекомендательных систем полагаются на одно или несколько из следующих:

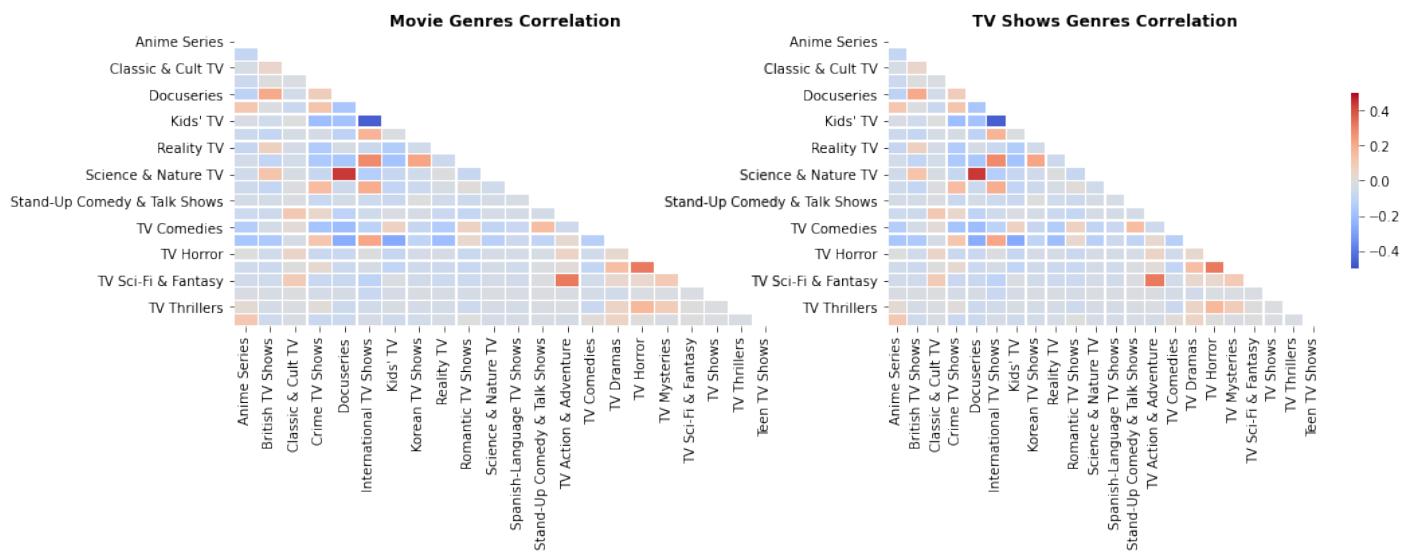
- Косинус $s(q, x) = \cos(q, x)$
- Скалярное произведение $s(q, x) = \sum_{i=1}^d q_i x_i = \|x\| \times \|q\| \cos(q, x)$
- Евклидова метрика $s(q, x) = \|q - x\| = \left[\sum_{i=1}^d (q_i - x_i)^2 \right]^{\frac{1}{2}}$

4.7 Жанры №2

Вернёмся к жанрам, взглянем на следующий график



Как мы можем заметить количество превалирующих жанров растёт на протяжение лет. Следовательно интерес пользователя растёт к данным жанрам, особенно это заметно на **Комедии**, т.к. она переместилась с 3-го места на второе и уже на протяжении 3 лет удерживает свою позицию. А следовательно появляются много похожих фильмов в жанре комедии, которые интересны зрителю. И на последок в жанрах, взглянем на корреляцию



Ну кое-где есть корреляция, ну она достаточно логична: Триллер - Хоррор, Наука и Природа - Документалки, Мистика - Хоррор и т.п.

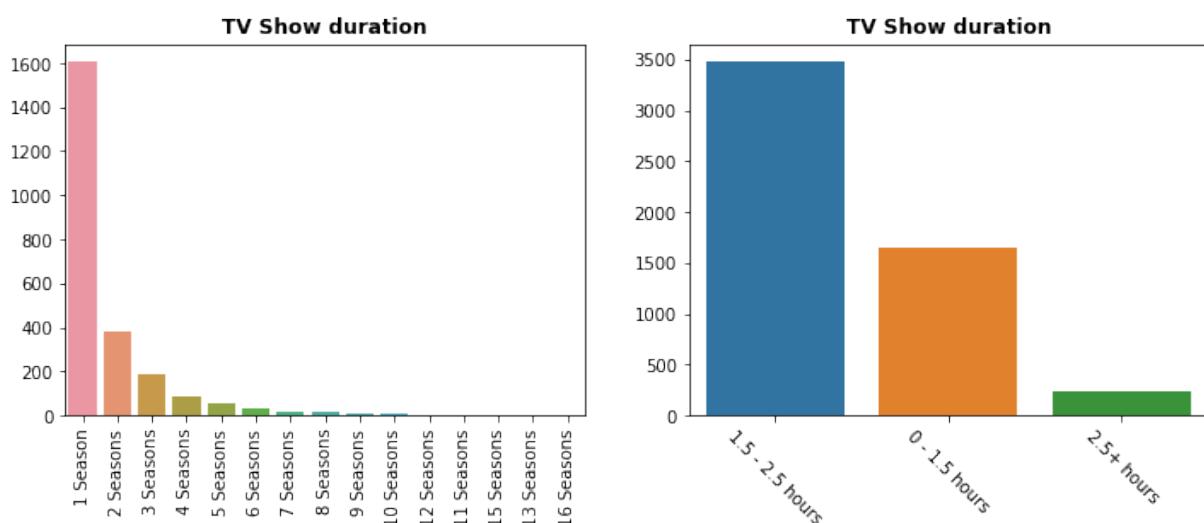
4.8 Мини-обобщение

Так, давайте немного остановимся и сделаем обобщение. Я то понимаю свои мысли, а вот остальные могут и запутаться.

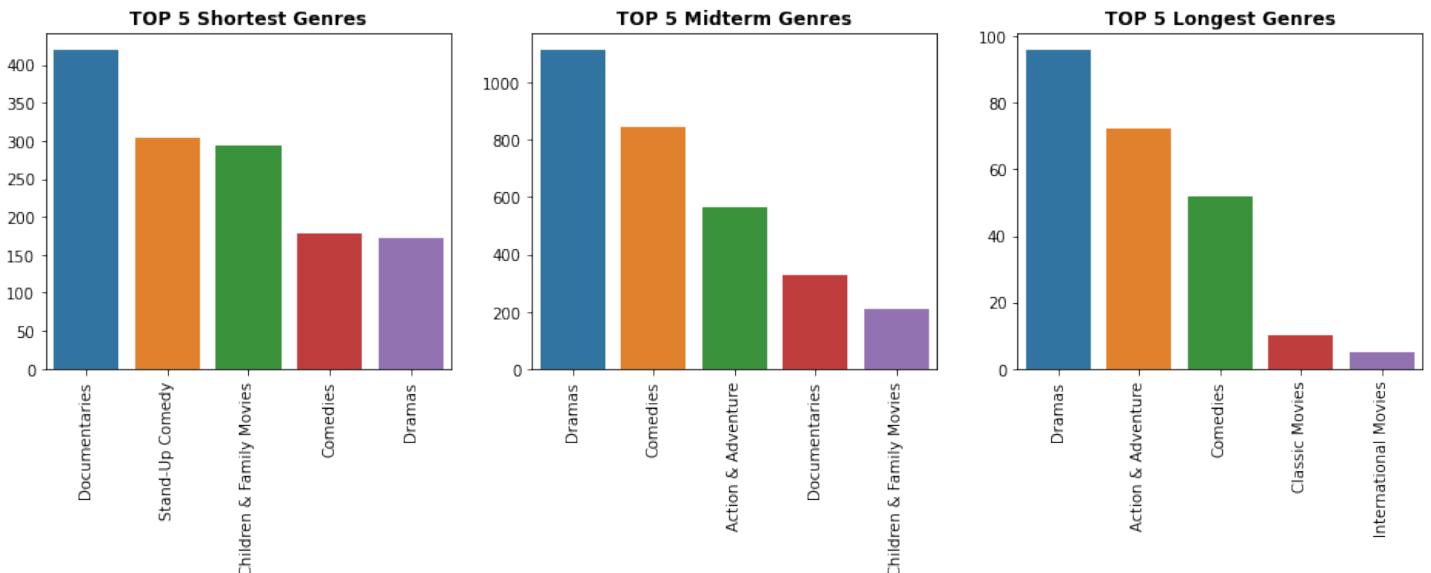
1. Мы определились, что фитча **director** является существенной, подкрепив данное предположение статистикой V Крамера и удалять её не стоит. Поэтому мы заменили пропущенные значения на "No Director".
2. В поле **cast** было 718 пропущенных записей, путём всё тех же манипуляций мы получаем $\tilde{V} = 0.46$, что тоже не очень чтобы плохой показатель. Поэтому заменяем пропущенные записи на "No Cast".
3. В поле **date_added** было 10 пропущенных записей, которые мы успешно удалили, т.к. 10 записей не сильно влияют на датасет из 7787 записей.
4. В поле **country** было 507 пропущенных записей, которые мы заменили на моду. Поскольку США далеко впереди по производству кинокартин, 6% данных не особо изменят распределение.
5. В поле **rating** было 7 пропущенных записей, которые мы заменили, предварительно загуглив данные продукты.
6. Соотношение фильмов к ТВ-шоу где-то $\frac{3}{1}$.
7. Далее мы рассмотрели возрастной ценз (**rating**), и объединили данную фитчу в более крупные подгруппы (**age_group**).
8. Дату добавления (**date_added**) разделили на день/месяц/год. Проанализировав месяцы и дни, удостоверяясь, что они не особо влияют на схожесть контента и удаляем данные фитчи. Однако год добавления я предпочту оставить, т.к. если тот или иной фильм или сериал добавили недавно, и по всем остальным характеристикам он является одним из ближайших соседей, то данное произведение можно продвинуть в топе, как одно из самых предлогаемых.
9. Также убеждаемся, что разница между годом добавления на сервис и годом выпуска так же не связана с жанром, а т.е. с мерой сходства.
10. Однако мы замечаем, что количество представителей жанра растёт от года в год. Доказываем посредством того же χ^2 .

4.9 Продолжительность

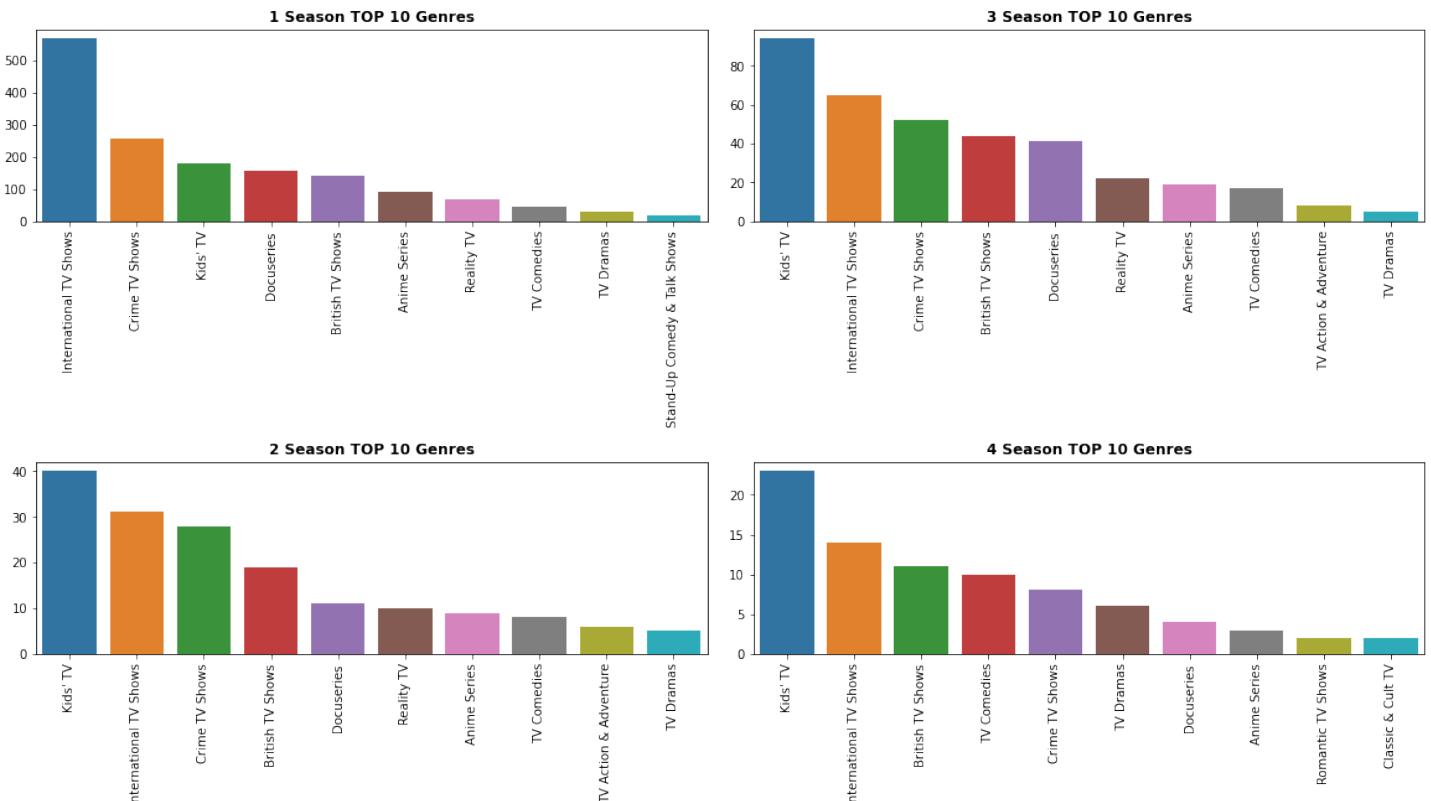
В датасете представлено два вида продолжительности, первая для односерийных, полнометражных картин - **n min**, вторая же для многосерийных представителей - **n Seasons**. С сезонами всё легко. А вот фильмы представлены в минутах. Поэтому конвертируем в самые частоиспользуемые интервалы продолжительностей.



Как мы видим самые популярные фильмы, это 1.5 - 2.5 часовые картины. Давайте посмотрим топ 5 самых популярных жанров в каждом интервале.



Да, жанры зависят от продолжительности произведения. Вобщем-то, что логично, обычно Драмы с большой продолжительностью - сериалы.



Поскольку в обычном виде продолжительность фильмов слишком диверсифицирована, там 201 уникальный класс, что за собой влечёт "предвзятость" в кластеризации, мы заменим слайс столбца **duration** для фильмов на соответствующий интервал продолжительности. Если же говорить про сериалы, вот здесь я бы оставил всё как есть. Объясню, рассмотрим такую ситуацию: есть сериал или мульт-сериал, который нас очень заинтересовал, или он очень похож на просматриваемый нами фильм или сериал, и если он не закончен, а в основном сериалы не заканчиваются на первом - втором сезоне то нам придётся ждать продолжения, а на продолжение уходит минимум 1 - 1.5 года. За это время интерес к произведению может подостыть, что нам, как крупнейшему стриминговому сервису не очень выгодно, нужно чтобы пользователь продолжал платить за подписку. Поэтому продолжительность сериала - важная характеристика, которая должна остаться в текущем виде.

4.10 Актёрский состав

Здесь что-то сложно говорить. Давайте, я попробую разобрать почему.

ДИСКЛЕЙМЕР! Следующие рассуждения не являются оскорблением актёров и их внешности, также я не рассматриваю их актёрскую игру. Ни каких претензий к самим актёрам я не имею. Всё, что будет сказано далее - претензия к главам кинокомпаний, которые утверждают каст.

Рассмотрим два произведения

1. [Ведьмак от Netflix](#), мы будем сравнивать с оригиналом (книгами) и игрой.
2. [The Last of Us от HBO](#), сравнивать будем с почти оригиналом (игрой).

4.10.1 Ведьмак

Начнём с Ведьмака - это идеальный пример и чёрной и белой стороны. Для начала надо ознакомиться с сеттингом. Ведьмак является восточноевропейским фэнтези, или другими словами славянским фэнтези. Действия произведения происходят в мире похожем на средневековье, в котором присутствует магия, эльфы, гномы, междоусобные распри разделённых королевств, предательства и измены. Начнём с персонажа, который у Netflix вышел достаточно неплохо. Описание в книгах следующее:

«Родилась Цири в момент, когда ее матери исполнилось лишь 15 лет. С детства девчонка проявляла бойкость и ярость в характере. Цири повторяла легендарную красоту своей не менее известной матери: женственная форма лица, красивые и утонченные скулы, глубокие и большие глаза зеленого цвета, а также исключительные волосы пепельного цвета. Цирilla обожает мальчишеские игры и практически не выносит глупые девичьи занятия.»

В игре данного персонажа представили следующим образом:



Figure 1: Adult Ciri



Figure 2: Child Ciri

Если сравнить с книжным описанием, выглядят похоже. И славянские/скандинавские черты лица сохранены. Поэтому тем, кто сначала ознакомился с книгами, а потом пошёл играть в игры было приятно играть. На профессиональном жаргоне такое сохранение образа других, оригинальных произведений называется "**канон**".

Давайте оценим на сколько по канону выглядит Цири из сериала от Netflix:



Figure 3: Netflix Ciri 1



Figure 4: Netflix Ciri 2

Ну в принципе, меня, как любителя данной серии произведений, Цири от Netflix вполне устраивает. У неё достаточно подходящие, скандинавские черты лица, да, выражение лица слишком детское, но её актёрская игра всё перекрывает. В дополнение, можно сделать скидку на то, что это первый сезон, и в данном сезоне - Цири это бойкая дочка королевы, она ещё много чего не видела и т.п. И это не плохой пример каста в исполнении глав Netflix.

А теперь рассмотрим мисскаст, который просто сбивает повествование. Трис Меригольд.

«Трисс является светлым и мягким персонажем, ей не чужды стеснение, преданность, милосердие, сочувствие и сострадание. Трисс всегда жаждет помочь людям, не отказывает в помощи, также Меригольд очень любит людей и детей.

Чародейка испытывает нелюбовь лишь к тем, кто пытается воспользоваться ею или открыто льстит, предпочитая рассказывать правду в глаза. Однако, совесть еще долго может заставлять девушку отводить взгляд от собеседника. С друзьями и близкими Трисс мила и добродушна, всегда позитивна и жизнерадостна. Ей владеет желание добиться мира и добра для всех вокруг, а особенно – для дорогих для нее людей.

Девушке иногда ведом страх, наивность и неуверенность, однако ее самоотверженность с лихвой компенсирует это.»

«Две из них были в платьях, очень скромно застегнутых под горлышико, — суровая, одетая в чёрное Шеала де Танкарвиль и молоденькая Трисс Меригольд с голубыми глазами и изумительно красивыми каштаново-рыжими волосами»

А теперь сравним игру и сериал.



Figure 5: Triss Game



Figure 6: Triss Netflix

И вот это уже мисскаст, и это **не только моё мнение**. Ну это ещё ладно, а вот следующая проблема, это то за, что я не долюблю глав Netflix. Напомню, что Ведьмак это славянское фэнтези, действие происходит в Скандинавии и Восточноевропейских странах, и тут



Figure 7: Black Man



Figure 8: Dryads

Вот от куда в Скандинавии темнокожие. Я ни в коем случае не против темнокожих, но они должны быть в тему, а не для того чтобы потом на конференциях орать, посмотрите, какие мы толерантные. А еще больше всего раздражает вторая фотография, это Дриады.

«Дриады очень красивы, хотя и разительно отличаются от привычных людям канонов. В первоисточнике нет никаких упоминаний об их цвете кожи; известно лишь то, что истинные дриады внешне похожи на эльфиек (высокий рост, мелкие зубы и т. д.), а дриады, обращённые с помощью Брокилонской воды, выглядят, как люди. В игровой вселенной же они имеют зеленоватую кожу, помогающую им идеально сливаться с листвой деревьев; цвет их волос колеблется между цветом корицы и изумрудной зеленью. Черты лица очень острые, лица узкие, равно как и очень лёгкие жилистые тела.»

Окей, хоть в первоисточнике не сказано про цвет кожи, но вы ведь то же, если хотите позагорать едете в лес, а не на пляж? Каким образом они стали такими загорелыми в лесу, куда солнечные лучи еле-еле попадают? Вобщем тренды 21 века.

4.10.2 The Last of Us

Рассмотрим одно произведение. Сериал ещё не вышел, но каст уже известен. Начнём с сеттинга. Действие основной части происходит спустя 20 лет после глобальной катастрофы — пандемии, вызванной мутировавшим грибом кордицепсом. Согласно сюжету, кордицепс попадает в человеческий организм в виде спор в результате вдыхания, через кровь или слону заражённого. Затем гриб вызывает необратимые изменения в человеческом организме — жертва теряет разум и превращается в подобие зомби. Пандемия привела к упадку цивилизации и гибели большей части населения США и всей Земли. Часть выживших укрывается в изолированных карантинных зонах, охраняемых военными. Города за пределами карантинных зон превратились в опасные руины, населённые заражёнными, бандитами и каннибалами. Террористическая организация «Цикады» пытается поднять население карантинных зон на восстание против военных, а также стремится разработать вакцину против инфекции. Сюжет посвящён путешествию главных героев — контрабандиста Джоэла и девочки-подростка Элли.

Я хочу рассмотреть главную героиню. Сначала нужно кратко взглянуть на её судьбу, т.к. её судьба является очень хорошим описанием персонажа. Я постараюсь очень кратко, но советую полностью прочитать описание персонажа [здесь](#).

О детстве Элли известно немного: она стала сиротой вскоре после рождения и до 13 лет жила в суровых милитаристских условиях карантинной зоны. Образцовой воспитанницей назвать её было нельзя: девочка регулярно ввязывалась в драки и приключения, хамила старшим и т.п. Характерная черта Элли того периода — закрытость. Героиня напрочь отказывалась от общения с другими подростками, предпочитая проводить время наедине с собой и музыкальным плеером. Примерно в том же возрасте Элли на своих руках потеряла дорогого человека, который помог ей раскрыться с другой стороны.

Оказалось, что девочке страшно интересна поп-культура докарантинного мира; что она любит танцевать и обладает яркой фантазией. Какой бы мрачной ни была сцена прощания, она же стала спасением для героини — ведь именно тогда выяснилось, что у неё есть иммунитет. Мысль о том, что она способна помочь миру, придала Элли сил — ведь у неё впервые появился настоящий смысл жить. Именно такую Элли мы встретим в начале The Last of Us — да, потерявшую свою первую любовь, но все же заряженную на борьбу за будущее человечества. Во время путешествия игроку предстоит многое узнать об Элли, и самым интересным из этих фактов мне кажется тот, что в душе девочки осталась любопытным ребенком, несмотря на все перенесенные невзгоды: выбравшись из города, она начнет озираться по сторонам и радоваться тому, что перед ней открылся дивный новый мир. По ходу сюжета Элли будет раскрываться — не в новом свете, а просто как персонаж. Мы узнаем больше о её любви к комиксам и поп-культуре, выясним, что ей нравятся глупые шутки с играми слов. А еще увидим очень брутальную сцену сражения, в процессе которой Элли буквально разрубит голову противника, но даже этот стресс не повлияет на её стремление спасти человечество. Вообще Элли на протяжении путешествия докажет, что она может постоять за себя, не смотря на её возраст.

На подъезде к Солт-Лейк-Сити девочка скажет: «Всё то, через что мы прошли... То, что я сделала... Это не может быть зря». Для неё так важно оправдать своё существование в этом мире, и именно эту, единственную настоящую мечту Джоэл у неё отберет. Это травмирует девочку и навсегда разрушит её отношения с фактически приемным отцом.

Вобщем, понятно, что Элли это персонаж с очень трудной судьбой, который вроде-бы нашёл назначение в мире, но нет его у неё отобрал самый близкий человек. И данного персонажа будет играть [данная актриса](#). Причём это не плохая актриса. Так персонаж выглядит в игре



Figure 9: Ellie generation



Figure 10: Ellie Child

Однако вот сравнение игрового персонажа и актрисы, которая будет её играть.

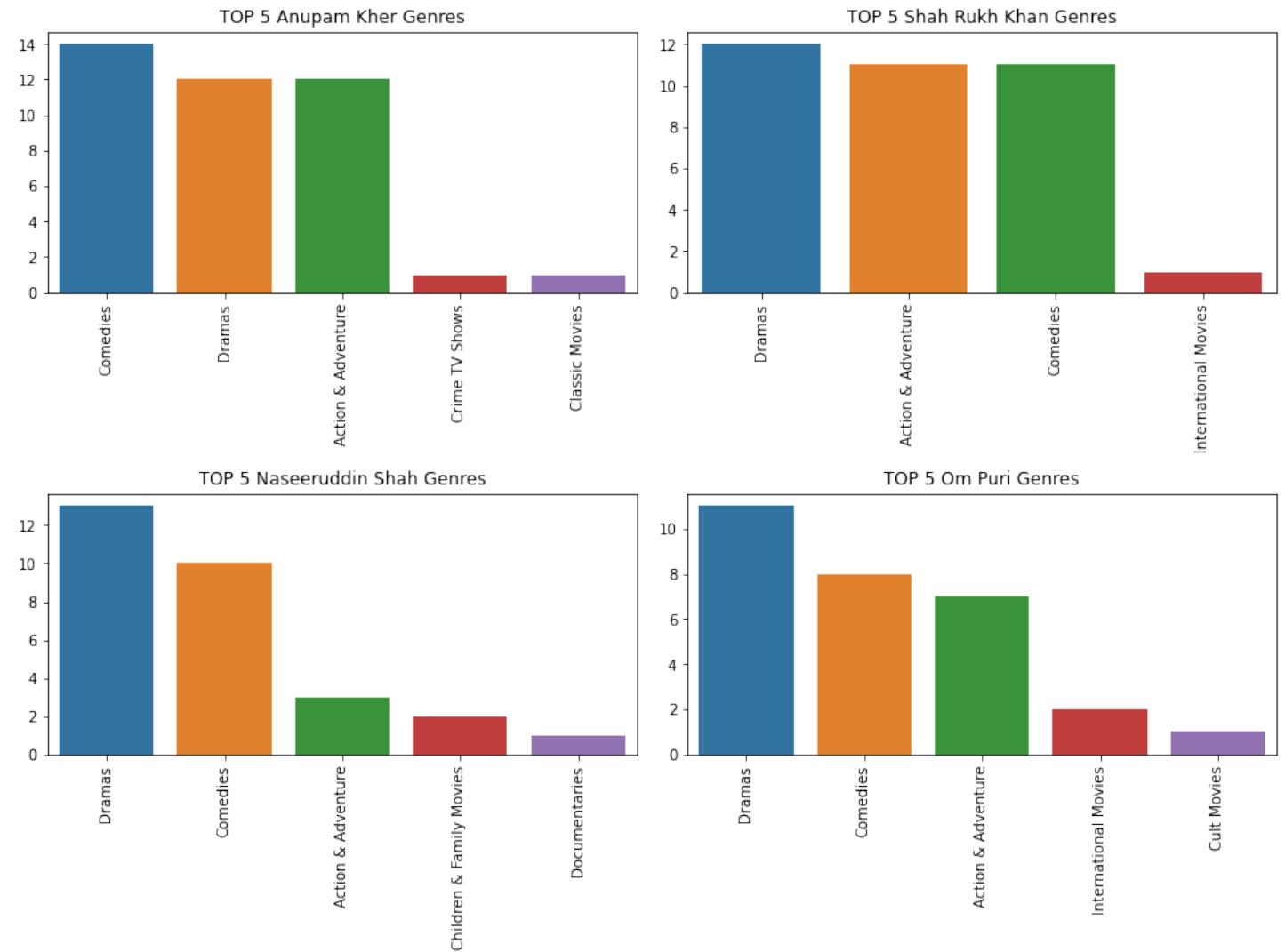


И вот, как по мне, лицо актрисы слишком добре. Элли слишком многое пережила, чтобы быть

такой доброй. Хотя, я повторюсь, данная актриса - хорошая актриса, она играла в самом крупном и крупнобюджетном сериале в мире: «*Игра Престолов*». Возможно, конечно, она сможет своим актёрским мастерством перекрыть неподходящий внешний образ. Но на данный момент я отношусь к ней скептически.

4.10.3 Подитог

И к чему я такую тираду развёл. Как-то глубоко анализировать каст не стоит. Безусловно, у актёров есть жанры в которых они чаще принимают участие, чем в других.



Поэтому удалять данную фичу будет глупо, однако, зная нынешние ценности кинематографии, особенно западные ценности, которые вносят сильные поправки в каст. Где, в дополнение к актёрскому мастерству и внешности оценивают сексуальную ориентацию, цвет кожи, национальность, политические взгляды и т.п., что в перспективе никак не влияет на актёрскую игру.

4.11 Синопсис (Описание)

Синопсис - представляет собой краткое изложение основных моментов произведения в прозе или в виде таблицы; сокращение или сжатие произведения.

Здесь не будет никакого анализа, а скорее подготовка к анализу фичи **description**. В машинном обучении за анализ смысловой нагрузки текста и не только отвечает **natural language processing (NLP)**. Прежде чем переходить к комплексным алгоритмам, сначала извлекается семантически значимые части текста, будь-то предложения, слова или символы. Мы здесь не будем использовать сложные алгоритмы на подобие **Word2vec**, а просто токенезируем слова.

Токенизация - один из первых шагов в NLP, и это задача разбиения последовательности текста на блоки с семантическим значением. Эти блоки называются токенами, и сложность токенизации

заключается в том, как получить идеальное разбиение, чтобы все токены в тексте имели правильное значение и не было пропущенных.

На большинстве языков текст состоит из слов, разделенных пробелами, где отдельные слова имеют семантическое значение. В данном случае мы работаем с английским, поэтому опустим такие языки, где используются кандзи, которые заключают в себе очень обширные значения.

Существует несколько разных способов токенезации, самые популярные следующие:

- BPE
- Unigram LM
- WordPiece
- SentencePiece

Мы рассмотрим один из них - Word Level Tokenization. В данном методе текст разделяется на блоки, где делитель это пробел. Соответственно блоки - слова. Далее убираются знаки пунктуации и всё переводится в нижний регистр (иногда). Создаётся словарь токенов. Рассмотрим на примере:

Оригинальное предложение: "This looks quite beautiful, I like that."

Строим словарь: [1: this, 2: looks, 3: quite, 4: beautiful, 5: i, 6: like, 7: that]

Тогда следующее предложение: "I like that" в токенизированном виде выглядит так: [5, 6, 7].

Минусы у Word Level Tokenization есть и они не маленькие:

- Если в слове допущена ошибка.
- Если встречаются такие блоки как "don't", "we'll" и т.п.
- Слишком большой размер словаря.
- Иногда в обучающей выборке отсутствует какое-то слово, которое есть в тестовой выборке.

Однако для нашего случая эти минусы отпадают, т.к.

- вероятность встретить ошибку в синопсисе - очень мала
- Такие блоки учитываются как целое слово, что в большинстве случаев так и есть.
- У нас в целом выборка не большая (7777 записей), так что и словарь будет не большой.
- Наша тестовая и обучающая выборка - одна и та же выборка.

Далее, чтобы упростить компьютеру жизнь отшкалируем каждый токен по следующей формуле

$$T_\sigma = \frac{T - \min(T)}{\max(T) - \min(T)}$$
$$T_{\text{scaled}} = T_\sigma(\text{min} - \text{max}) + \text{min}$$

Где min и max - минимальное и максимальное для всего датасета токенов.

Если применять всё вышеописанное к нашем датасету, а точнее к первой записи (фильм под названием "3%"), то мы имеем:

1. Оригинальный синопсис: "In a future where the elite inhabit an island paradise far from the crowded slums, you get one chance to join the 3% saved from squalor."
2. Токенезированный синопсис:

[6, 1, 193, 57, ..., 315, 2, 2669, 2670, 16, 9793]

3. Нормализованный синопсис:

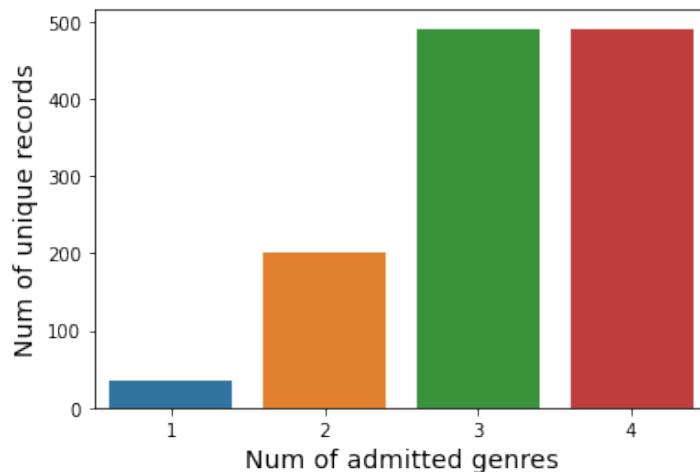
[$6.04e^{-4}$, $1.00e^{-4}$, $1.93e^{-2}$, $5.72e^{-3}$, $2.00e^{-4}$, ..., $1.00e^{-4}$, $2.67e^{-1}$, $2.67e^{-1}$, $1.50e^{-3}$, $9.82e^{-1}$]

И так мы проделываем для каждого синописса в нашем основном датасете. В итоге я получил 19068 уникальных токенов. Вы можете прочитать про остальные типы токенизации и про сами алгоритмы токенизации [здесь](#) и [здесь](#).

4.12 Итоги анализа

Хочу пояснить, что я проделывал дополнительный анализ не расписанный выше. Поэтому, что да как, поясню здесь. И так:

1. Для кластеризации нам не нужна колонка **title**, поэтому дропаем её.
2. Я всё же решил исключить один из годов, либо **year_added** либо **release_year**. Чтобы их сравнить, сначала пытаемся использовать [paired t-Test](#), понимам, что данные распределены не нормально (ха-ха). Поэтому используем [Wilcoxon Signed-Ranks Test](#) Получив статистику Wilcoxon W+ равной 11658.5. Что доказывает, что **release_year** сильно весомей чем **year_added**. Поэтому выкидываем **year_added**.
3. Я всё же решил избавиться от каста. Режисёр более показательный. Поэтому дропаем каст, слишком я в нём сомневаюсь.
4. Удаляем список стран (**country**), оставляя только основную страну (**main_country**).
5. С жанрами всё посложнее. У нас есть две колонки: **genres** и **main_genre**. Понятно, что **main_genre** мы оставляем, т.к. это один из основных показателей в мере сходства. А вот со списком жанров я решил немного поиграть. Я не особо здесь что-то тестировал, но я решил посмотреть на количество уникальных списков жанров в колонке, в зависимости от количества допустимых жанров в одном списке. Ну например, если у нас для какого-то фильма даны следующие жанры: [Drama, Thrillers, Action], тогда при количестве допустимых жанров равных одному, мы для этого фильма получим: [Drama], для двух: [Drama, Thrillers] и т.п. Соответственно и искать во всём датасете мы будем: в первом случае [Drama], а во втором [Drama, Thrillers]. Порядок важен, [Drama, Thrillers] не равен [Thrillers, Drama]. У меня получилась следующая диаграмма (максимальное количество жанров в списке - 3 жанра)



Тут чисто на ваше усмотрение, если хотите ужесточить границы кластера, сделать их более "чёткими" - тогда оставляйте 3. Если хотите сделать их более мягкими - выбирайте 2 допустимых жанра. Я выбрал 3, т.к. в дальнейшем мы будем получать выгоду от синописса.

6. И напоследок, для колонок (фитч) **director**, **duration**, **main_country**, **type**, **age_group**, **genres** и **main_genre** меняем тип на **categorical**, а затем заменяем значения этих колонок на их уникальные id. Например для колонки **type** мы имеем два уникальных значения [TV Show, Movie], соответственно для дальнейшей кластеризации данные значения заменяются на [0, 1]. Для колонки **genres** мы заменяем значения на [0,...,490]. И так далее.

В итоге получилась следующая таблица

show_id	type	director	release_year	duration	description	main_country	age_group	genres	main_genre
s1	1	2716	2020	12	[...]	6	0	397	16
s2	0	1840	2016	2	[...]	42	0	309	12
s3	0	1289	2011	0	[...]	60	0	335	13
s4	0	3446	2009	0	[...]	75	1	47	0
s5	0	3177	2008	2	[...]	75	1	293	12

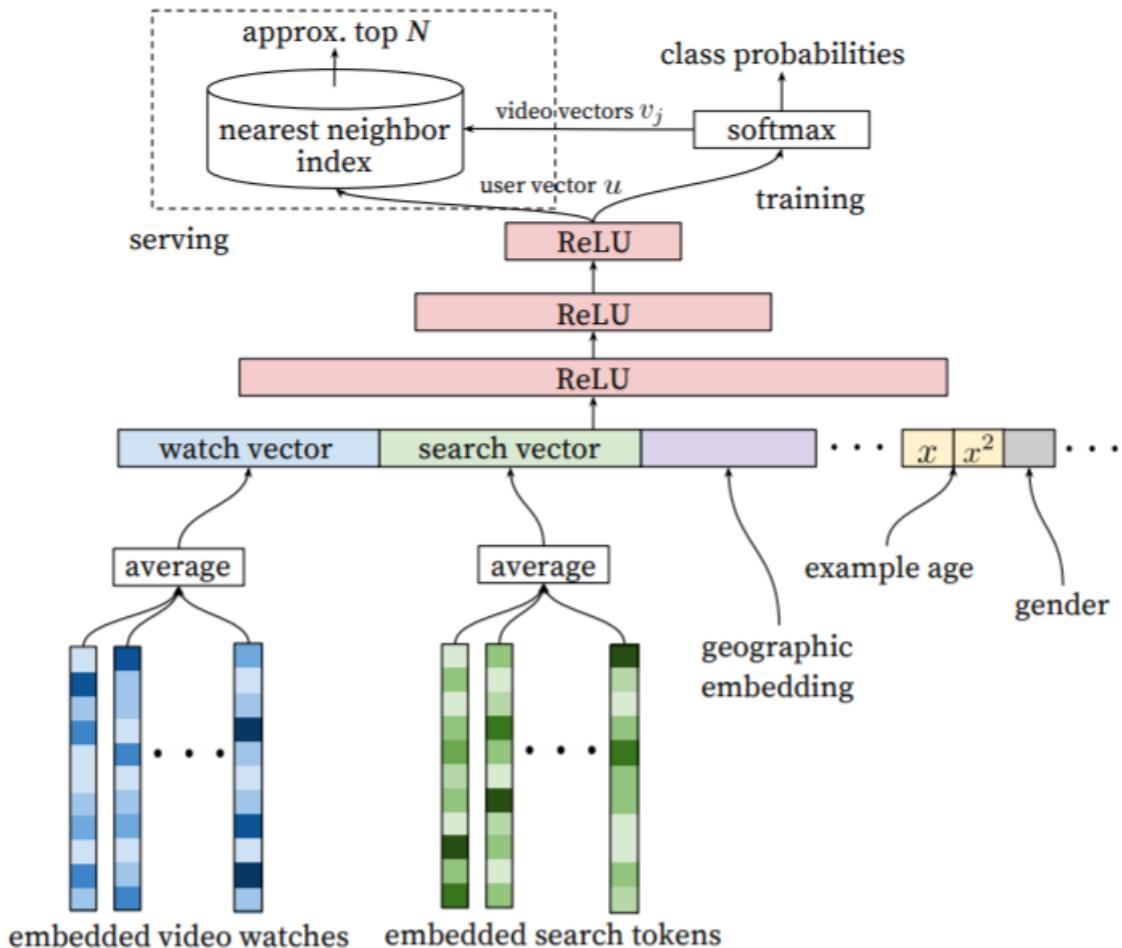
В целом - всё, данные к кластеризации мы подготовили. Переходим к оной.

5 Кластеризация

Кластеризация будет состоять из трёх шагов:

1. Узнаём начальное количество кластеров.
2. Создаём эмбединг для синопсисов. По сути это ещё одна кластеризация, только здесь мы попробуем разбить синопсисы на кластеры, а далее вместо синопсиса будем вставлять id кластера.
3. Итоговая кластеризация.

Хочу подметить, что для таких датасетов обычных алгоритмов кластеризации не достаточно. Далее приведена плюс-минус реальная система генерации кандидатов YouTube'a



Немного посложнее, не правда ли? Ну да ладно, опустим мелочи:)

Прежде чем перейти к самой кластеризации, обсудим две очень важные темы.

- Нормализация данных

- Графическое представление кластеров (уменьшение размерности)

5.1 Нормализация данных

Немного поговорим о данных, которые мы имеем. Для более точного понимания давайте визуализируем зависимость двух фич: **director** и **genres** (Figure 11).

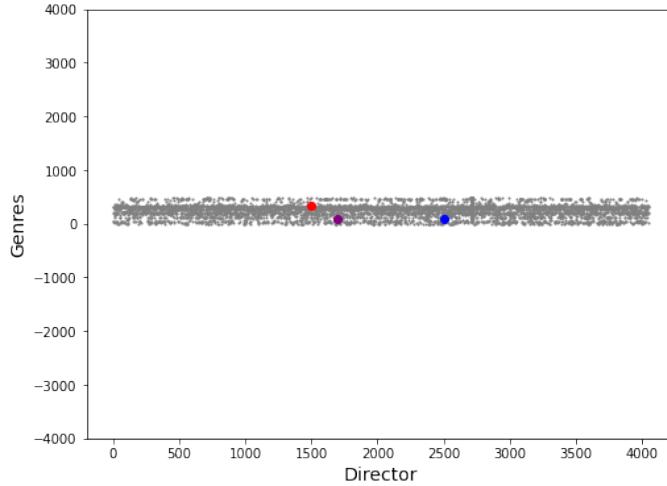


Figure 11: Non-normalized data

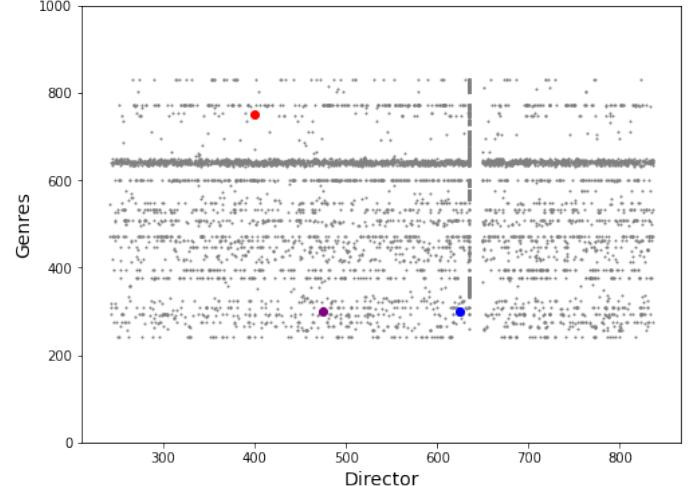


Figure 12: Normalized data

На левом графике чётко видно, что фиолетовая точка намного ближе к красной, чем к синей. А теперь проделаем некую магию, у-у-у-у-у. И взглянем на то, что получилось (Figure 12). А теперь феолетовая точка ближе к синей чем к красной. И в этом то и проблема. При кластеризации важно расстояние между представителями каждого кластера. А, если мы оставим данные без нормализации, тогда получается что год выпуска добавляет артефакты в данные и сильно влияет на разбиение по кластерам. Окей, о важности нормализации поговорили, давайте обсудим магию, которую мы сотворили. Есть много способов нормализации, но их общий плюс - данные принимают вид Гауссовского распределения, что даёт большой простор для работы над данными. [Здесь](#) есть описание некоторых алгоритмов. Я же распишу, тот который я использовал.

5.1.1 Квантильная нормализация

Рассмотрим набор данных $\mathcal{D} = X_1, X_2, \dots, X_n$, где $X_{i,j} \in \mathbb{R}$. И $X_{s,j} \ll X_{i,j} \ll X_{k,j}$ для $s = i-1 = k-2$. Данный набор данных мы сортируем по убыванию.

$$\mathcal{D}_{\text{sorted}} = s(\mathcal{D}), \quad \{X_{i,j}^{\text{sorted}} \geq X_{i+1,j}^{\text{sorted}} | X_{i,j}^{\text{sorted}} \in \mathcal{D}_{\text{sorted}}\}$$

Далее для каждой строки высчитываем среднее арифметическое.

$$\begin{aligned} \mathcal{D}_{\text{row mean}} &= m(\mathcal{D}_{\text{sorted}}); \\ \bar{X}_i &= \frac{\sum_{j=0}^n X_{i,j}^{\text{sorted}}}{n} \end{aligned}$$

Теперь возвращаем датасет в неотсортированный вид, сохраня значения среднего арифметического.

$$\mathcal{D}_{\text{row mean}} \rightarrow \mathcal{D}, \quad X_{i,j} \in \mathcal{D}_{\text{row mean}}$$

Если в какой нибудь колонке встретились два одинаковых значения, то они оба заменяются на среднее значение между данным и максимальным построчным значением. Это проще показать на примере из [Википедии](#)

A	5.67	4.67	2.00
B	2.00	2.00	3.00
C	3.00	4.67	4.67
D	4.67	3.00	5.67

В данном случае во второй колонке есть два значения 4.67. Тогда способы их замены:

1. $\frac{5.67+4.67}{2} = 5.17$
2. $\frac{2.00+4.67}{2} = 3.335$
3. $\frac{3.00+4.67}{2} = 3.835$
4. $\frac{4.67+4.67}{2} = 4.67$

Мы выбираем максимальную сумму. Соответственно это наш первый вариант. Тогда таблица принимает вид

A	5.67	5.17	2.00
B	2.00	2.00	3.00
C	3.00	5.17	4.67
D	4.67	3.00	5.67

Вот! Ну давайте теперь применим нормализацию и сравним наши данные.

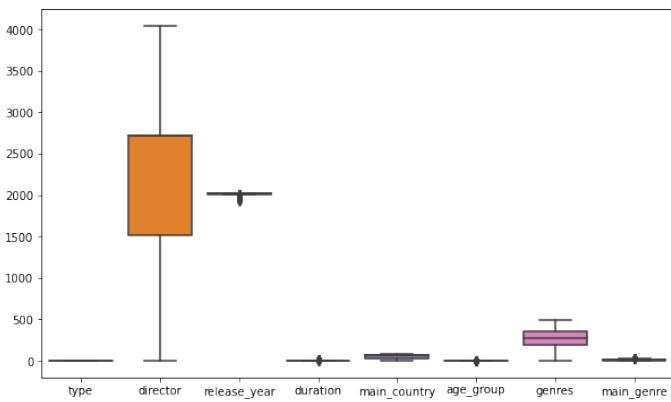


Figure 13: Non-normalized data

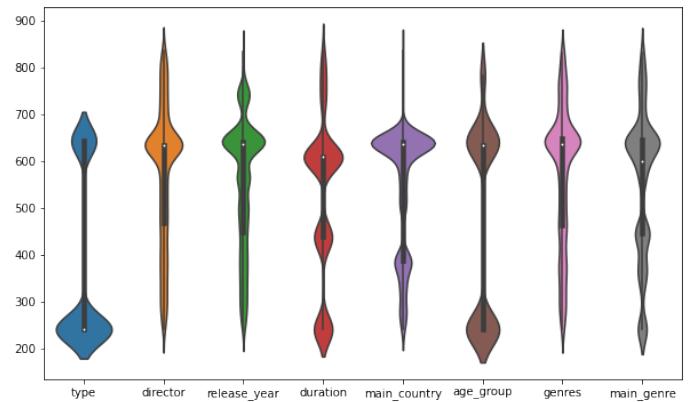


Figure 14: Normalized data

5.2 Графическое представление кластеров (уменьшение размерности)

В примере [выше](#) мы рассмотрели всего лишь две фитчи, которые легко визуализируются. Соответственно, если мы будем проводить кластеризацию только на этих двух фитчах, то мы спокойно изобразим эти кластеры на 2-D графике. Даже для трёх фитч мы можем отобразить кластеры. А что же делать с нашими 9 фитчами? Для этого, очень умные люди, а точнее Карл Пирсон и Лоренс ван дер Маатен вместе с Джоффри Хинтон придумали такие методы, как [Principal Components Analysis \(PCA\)](#) и [t-SNE](#), соответственно. Обсудим каждый.

5.2.1 Principal Components Analysis (PCA)

Чтобы понять, как работает PCA - ознакомтесь с данным [видео](#). Я же опишу всё достаточно кратко.

PCA определяется как [ортогональное линейное преобразование](#), которое преобразует данные в новую систему координат, так что наибольшая дисперсия некоторой скалярной проекции данных приходится на первую координату (называемую первым (главным) компонентом), следующая по величине дисперсии приходится на вторую координату и т. д.

Рассмотрим $\mathbf{X} = \mathbb{R}^{n \times p}$ с нулевым эмпирическим средним значением для столбцов. где каждая из n строк представляет собой одно наблюдение (одну запись в датасете), а каждый из p столбцов определяет какую-нибудь фитчу.

Примечание: Поясню, что обозначает "нулевым эмпирическим средним значением для столбцов". Это значит, что среднее значение для каждой колонки равно нулю. Рассмотрим на примере, что это значит. Пусть есть следующий набор данных \mathcal{D}

x	2	2.5	2.3	0.3	0.5	0.5
y	1	1.2	1.4	0.5	0.7	0.3

Чтобы найти матрицу \mathbf{X} , сперва мы находим барицентр оригинальных данных. Потом мы сдвигаем данный барицентр в центр координат $(0, 0)$. Давайте посчитаем координаты барицентра:

$$\bar{x} = \frac{2 + 2.5 + 2.3 + 0.3 + 0.5 + 0.5}{6} = 1.35$$

$$\bar{y} = \frac{1 + 1.2 + 1.4 + 0.5 + 0.7 + 0.3}{6} = 0.85$$

Чтобы данный барицентр оказался в центре координат - сдвигаем на -1.35 по OX и на -0.85 по OY . Тогда матрица \mathbf{X} равна

$$\mathbf{X} = \begin{bmatrix} 0.65 & 0.15 \\ 1.15 & 0.35 \\ 0.95 & 0.55 \\ -1.05 & -0.35 \\ -0.85 & -0.15 \\ -0.85 & -0.55 \end{bmatrix}$$

Соответственно $\bar{X}_0 = 0$ и $\bar{X}_1 = 0$. Графически это бы выглядело так

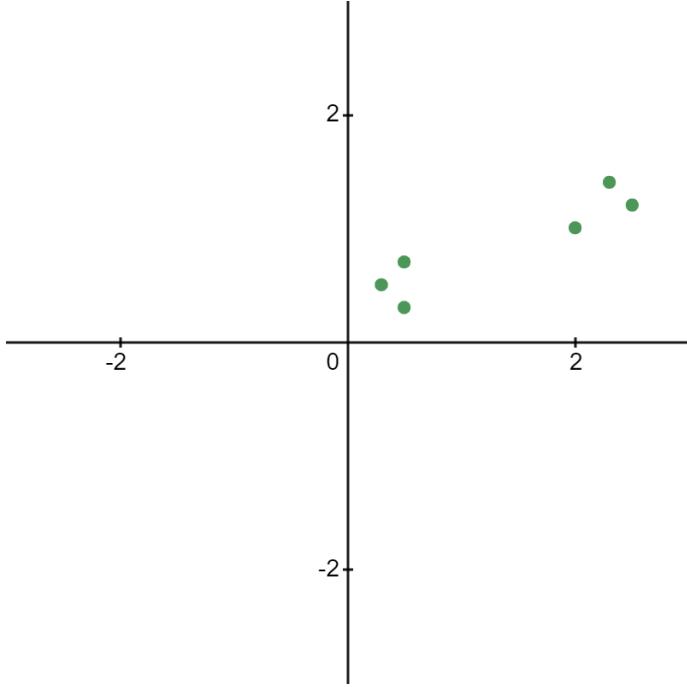


Figure 15: Original Data

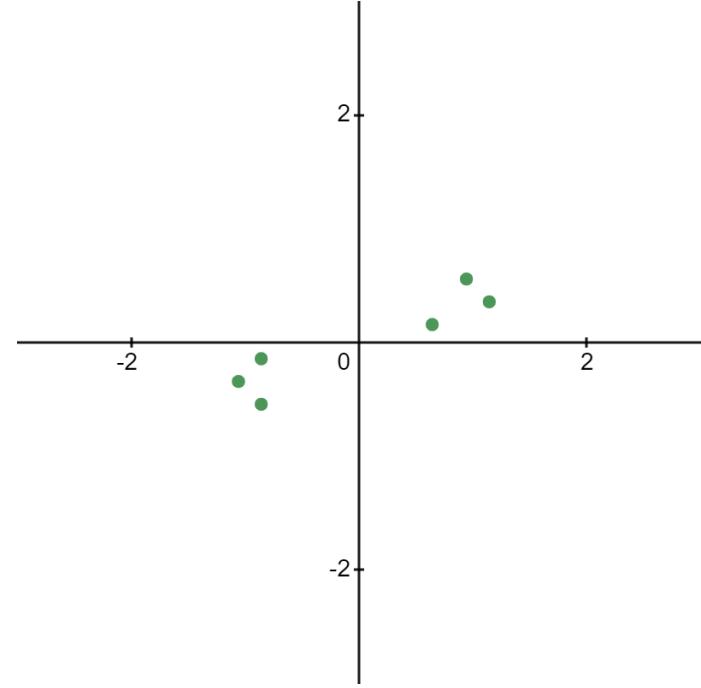


Figure 16: X-matrix Data

Математически преобразование определяется набором p -мерных векторов коэффициентов $\mathbf{w}_{(k)} = (w_1, \dots, w_p)_{(k)}$. Данный набор длиной l сопоставляет каждый вектор-строку $\mathbf{x}_{(i)} \in X$ с новым вектором оценок главных компонент $\mathbf{t}_{(i)} = (t_1, \dots, t_l)_{(i)}$, заданный как

$$t_{k(i)} = \mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)} \quad i = 1, \dots, n \quad k = 1, \dots, l$$

таким образом отдельные переменные $\mathbf{t}_1, \dots, \mathbf{t}_l$, рассматриваемые на всём датасете, последовательно наследуют максимально возможную дисперсию от X , причем каждый вектор коэффициентов \mathbf{w} это юнит вектор. (где l обычно выбирается меньше p для уменьшения размерности).

Таким образом, чтобы максимизировать дисперсию, первый вектор коэффициентов $\mathbf{w}_{(1)}$ должен удовлетворять

$$\mathbf{w}_{(1)} = \arg \max_{\|\mathbf{w}\|=1} \left\{ \sum_i (t_1)_{(i)}^2 \right\} = \arg \max_{\|\mathbf{w}\|=1} \left\{ \sum_i (\mathbf{x}_{(i)} \cdot \mathbf{w})^2 \right\}$$

Перепишем в матричном виде

$$\mathbf{w}_{(1)} = \arg \max_{\|\mathbf{w}\|=1} \{\|\mathbf{X}\mathbf{w}\|^2\} = \arg \max_{\|\mathbf{w}\|=1} \{\mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w}\}$$

Поскольку $w_{(1)}$ был определен как единичный вектор, тогда верно следующее

$$\mathbf{w}_{(1)} = \arg \max \left\{ \frac{\mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w}}{\mathbf{w}^T \mathbf{w}} \right\}$$

Когда $\mathbf{w}_{(1)}$ найден, первая главная компонента вектора данных $\mathbf{x}_{(i)}$ может быть затем задана как $\mathbf{t}_{1(i)} = \mathbf{x}_{(i)} \cdot \mathbf{w}_{(1)}$ в преобразованных координатах или как соответствующий вектор в исходных переменных, $\{\mathbf{x}_{(i)} \cdot \mathbf{w}_{(1)}\} \mathbf{w}_{(1)}$.

k -я компонента может быть найдена путем вычитания первых $k - 1$ главных компонент из X

$$\hat{\mathbf{X}}_k = \mathbf{X} - \sum_{s=1}^{k-1} \mathbf{X} \mathbf{w}_{(s)} \mathbf{w}_{(s)}^T$$

а затем находим вектор коэффициентов, который извлекает максимальную дисперсию из этой новой матрицы данных

$$\mathbf{w}_{(k)} = \arg \max_{\|\mathbf{w}\|=1} \left\{ \|\hat{\mathbf{X}}_k \mathbf{w}\|^2 \right\} = \arg \max \left\{ \frac{\mathbf{w}^T \hat{\mathbf{X}}_k^T \hat{\mathbf{X}}_k \mathbf{w}}{\mathbf{w}^T \mathbf{w}} \right\}$$

Оказывается, это дает оставшиеся собственные векторы $\mathbf{X}^T \mathbf{X}$ с максимальными значениями для значения в скобках, заданными их соответствующими собственными значениями. Таким образом, векторы коэффициентов являются собственными векторами $\mathbf{X}^T \mathbf{X}$.

k -я главная компонента вектора данных $\mathbf{x}_{(i)}$ может быть задана как $\mathbf{t}_{k(i)} = \mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)}$ в преобразованных координатах или как соответствующий вектор в исходных переменных, $\{\mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)}\} \mathbf{w}_{(k)}$, где $\mathbf{w}_{(k)}$ - k -й собственный вектор $\mathbf{X}^T \mathbf{X}$.

Таким образом, полное разложение \mathbf{X} на главные компоненты можно представить в виде

$$\mathbf{T} = \mathbf{X} \mathbf{W}$$

Преобразование $\mathbf{T} = \mathbf{X} \mathbf{W}$ отображает вектор данных $\mathbf{x}_{(i)}$ из исходного пространства p -переменных в новое пространство p -переменных, которые не коррелируют на всём датасете. Однако не все основные компоненты необходимо сохранять. Сохранение только первых L главных компонент, созданных с использованием только первых L собственных векторов, дает усеченное преобразование

$$\mathbf{T}_L = \mathbf{X} \mathbf{W}_L$$

где матрица \mathbf{T}_L теперь имеет n строк, но только L столбцов. Другими словами, РСА изучает линейное преобразование $t = W^T x, x \in R^p, t \in R^L, t = W^T x, x \in R^p, t \in R^L$, где столбцы матрицы \mathbf{W} размера $p \times L$ образуют ортогональный базис для L фитч, которые декоррелированы. По построению всех преобразованных матриц данных только с L столбцами, эта матрица оценок максимизирует

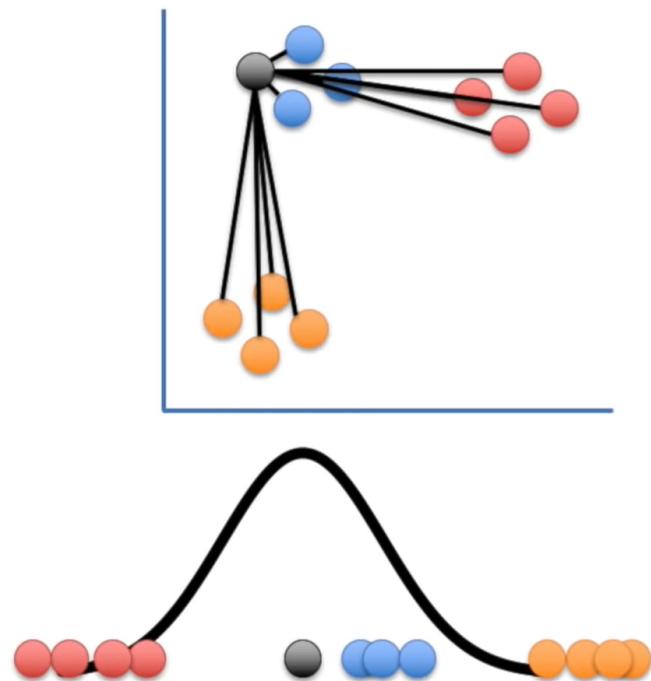
дисперсию в исходных данных, которые были сохранены, при минимизации общей квадратичной ошибки $\|\mathbf{T}\mathbf{W}^T - \mathbf{T}_L\mathbf{W}_L^T\|_2^2$ или $\|\mathbf{X} - \mathbf{X}_L\|_2^2$.

Думаю проблема очевидна - ошибка дисперсии для главных компонент. Другими словами, чем больше главных компонент, тем меньше точность отображения. Поэтому переходим к следующему методу.

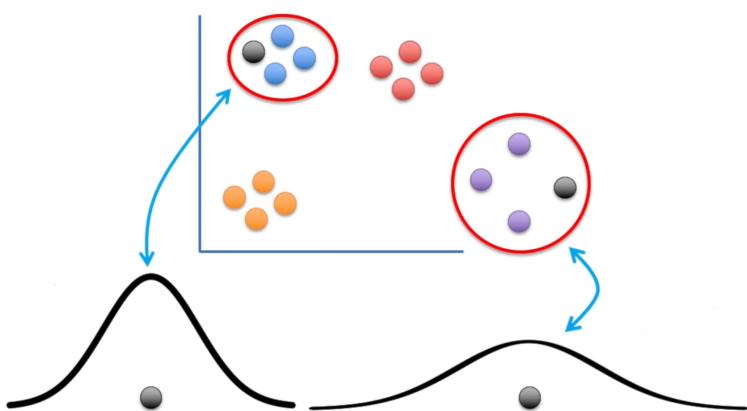
5.2.2 t-distributed stochastic neighbor embedding (t-SNE)

Чтобы понять, как работает t-SNE - ознакомьтесь с данным [видео](#). А я опять кратенько суммирую. Ахаха, судя по РСА - кратенько это сильно сказано, но здесь всё легче. Вообще на эту тему есть великолепная [работа](#), с которой советую ознакомиться. Рассмотрим набор данных \mathcal{D} , состоящий из $\mathbf{x}_1, \dots, \mathbf{x}_n$, где $x_i = \mathbb{R}^2$. Другими словами мы рассматриваем набор данных в 2-D пространстве. Соответственно наша цель уменьшить размерность до 1-D пространства, при этом сохранив плотность распределения. Сперва для каждой точки x_i вычисляется вероятность, что x_j , где $i \neq j$ является соседом. Для этого рассчитываются расстояния от точки x_i до всех остальных точек, проецируя их на Гауссовское распределение.

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)},$$



Сразу закрываем вопрос, почему мы взяли σ_i . Ну в принципе из-за этого



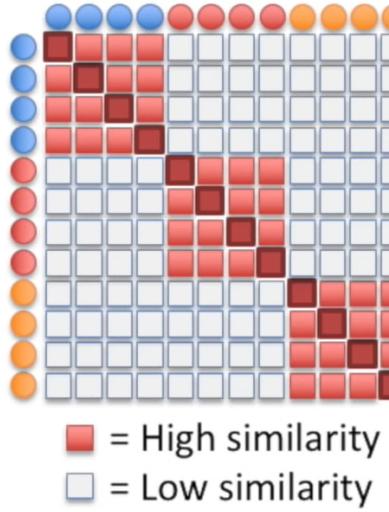
Кластеры с разной плотностью образуют разные распределения. Делением на соответствующую среднеквадратичную ошибку мы, так сказать, уравниваем распределения (меру схожести/вероятности).

Т.к. вероятность, что сама точка является своим соседом равна нулю то $p_{i|i} = 0$, ну и поскольку мы ищем вероятности то $\sum_j p_{j|i} = 1$ для всех i .

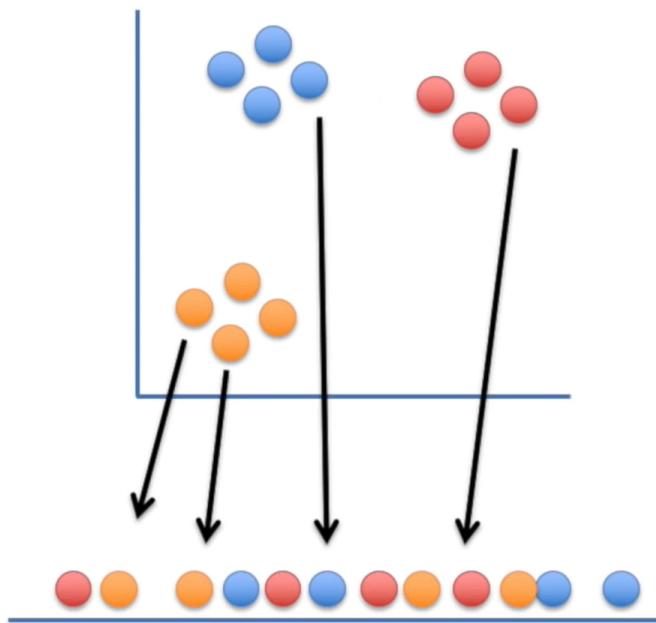
Тогда, чтобы найти вероятность, что x_j является соседом x_i для всех точек x_i , мы строим матрицу схожести, где каждый элемент равен

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

при $p_{ij} = p_{ji}$, $p_{ii} = 0$, и $\sum_{i,j} p_{ij} = 1$. Давайте взглянем на матрицу вероятностей (схожести), которая получилась



Напоминаю, что это матрица схожести для 2-D пространства, образец, так сказать. Теперь все точки рандомно проецируются на линию



Теперь находим матрицу схожести для полученного 1-D пространства. Однако, для поиска данной матрицы для 1-D пространства мы будем использовать не Гауссовское распределение, а t-распределение Стьюдента. Хоть данные распределения и выглядят похоже

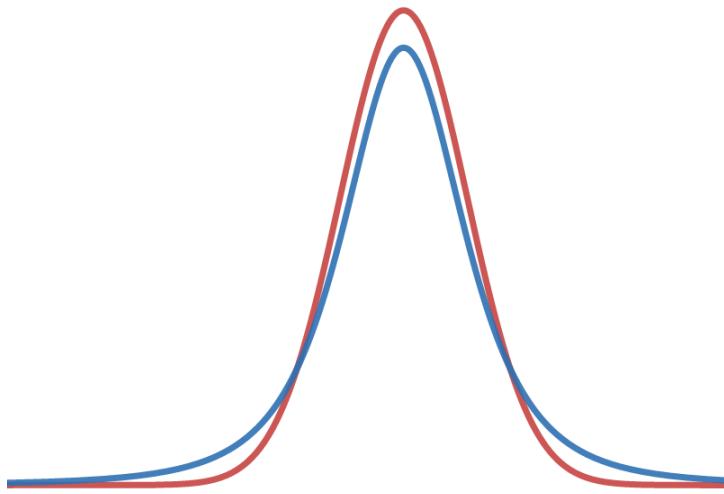


Figure 17: t-Distribution vs Gauss Distribution

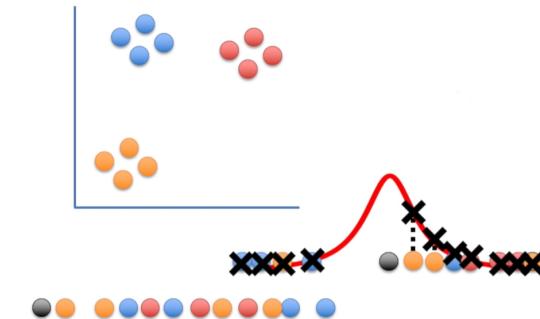
где t-распределение - голубое, а Гауссовское распределение - красное. Поясню, почему мы так делаем. При вычислении матрицы схожести для 2-Д пространства мы использовали Евклидовы расстояния, как x , и проецировали их на Гауссовское распределение. Однако при уменьшении размерности пространства появляется такая проблема, как «Проклятие размерности». Другими словами, если мы будем использовать Гауссовское распределение для выявления соседей (матрицы схожести) на пространстве с меньшей размерностью, тогда наши кластеры в данном пространстве будут перекрывать друг-друга. Связанно это с тем, что при увеличении размерности пространства, данные в любом случае становятся более разряженными (об этом вобщем-то и гласит Проклятие размерности). Однако, как вы могли заметить, хвосты у t-распределения более увесистые, нежели чем у Гауссовского. И центр более узкий, чем у Гауссовского. Что позволяет нам уровнять разряженность данных для разных размерностей. Теперь по, уже, известному нам алгоритму вычисляем матрицу схожести для нашей линии (1-Д пространства).

t-SNE пытается распределить d -размерное пространство $\mathbf{y}_1, \dots, \mathbf{y}_n$, где $\mathbf{y}_i \in \mathbb{R}^d$, и сделать это так, чтобы данное распределение максимально точно отображало схожесть p_{ij} . Другими словами, t-SNE пытается получить матрицу схожести $Q = \mathbb{R}^d$ максимально похожую на матрицу схожести P .

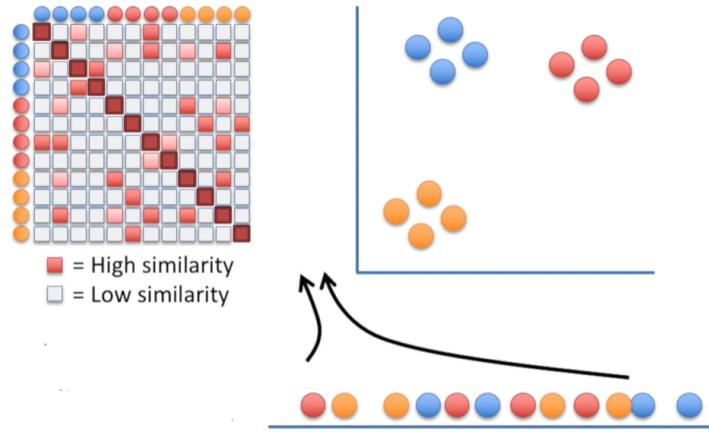
$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$

Как я и говорил, сначала мы проделываем те же шаги, до того момента, пока не получим матрицу схожести для 1-Д пространства. А это

1. расчитываем вероятность, что y_j - сосед y_i , подставляя их расстояния в t-распределение.



2. Нормализуем и строим матрицу схожести.



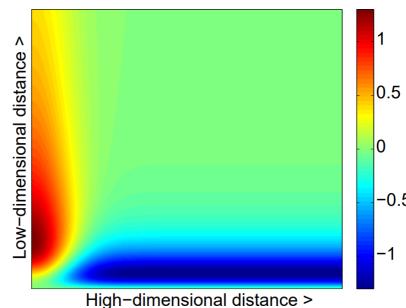
А теперь, путём перемещения элементов в получившейся матрице схожести Q , мы делаем её максимально похожей на матрицу схожести P . Технически, таким способом мы уменьшаем разницу между двумя распределениями. В математике есть целая мера, отвечающая за это - [Расстояние Кульбака — Лейблера](#), или просто KL-divergence. Т.к. число точек у нас конечное, то

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

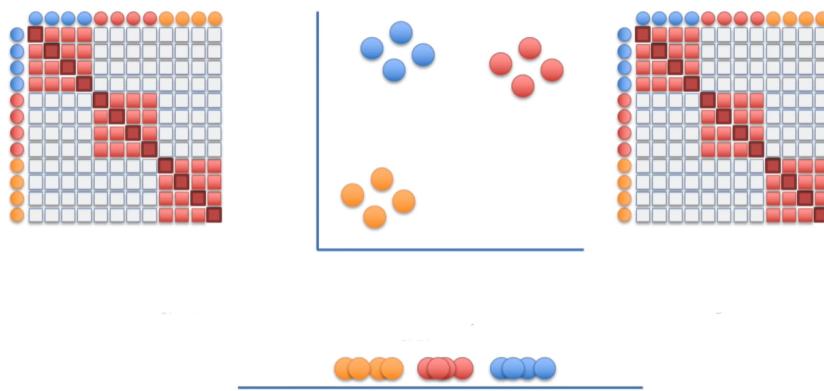
Соответственно, с помощью градиентного спуска ищем локальный минимум KL-divergence

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}$$

Получаем следующий градиент



Переставляем элементы нашей матрицы Q требуемым образом и получаем, уменьшенную размерность с сохранённой плотностью распределения



Вот и всё, говорил же легче:)) У этого метода тоже есть минус, но его обсудим позже. А, пока, мы будем им пользоваться для отображения кластеров нашего датасета.

5.3 Первоначальное количество кластеров

Для начала определим первоначальное количество кластеров с помощью K-means.

5.3.1 K-means

Даны n объектов, которые нужно разбить по k кластеров, минимизируя сумму дистанций от объектов до центроидов. Где:

- $A_{nk} = \begin{cases} 1, & \text{если } n\text{-ый экземпляр присвоен } k\text{-ому кластеру} \\ 0, & \text{в ином случае} \end{cases}$
- μ_k - центройд k -ого кластера

Составляем целевую функцию, которую мы будем минимизировать:

$$\min_{A,\mu} = \sum_{n=1}^N \sum_{k=1}^K A_{nk} \|\mu_k - x_n\|^2, \quad \text{где } A_{nk} \in \{0, 1\} \forall n, k \text{ и } \sum_{k=1}^K A_{nk} = 1 \forall n$$

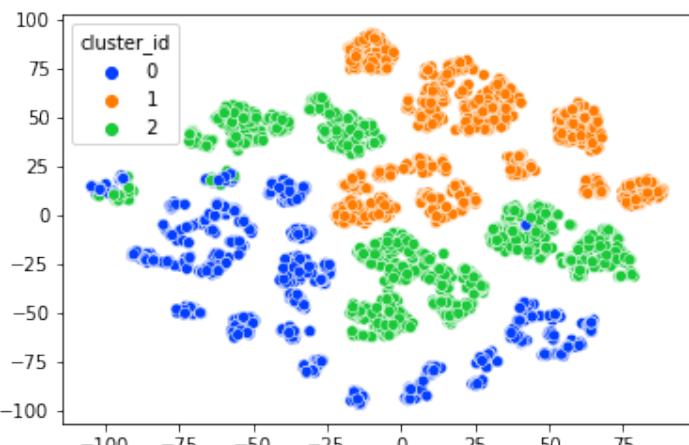
Чтобы минимизировать выражение относительно центроидов кластера, возьмём производную по и приравняем ее к 0.

$$\begin{aligned} C(\mu) &= \sum_{n=1}^N \sum_{k=1}^K A_{nk} \|\mu_k - x_n\|^2 \\ \frac{\partial C}{\mu_k} &= 2 \sum_{n=1}^N A_{nk} (\mu_k - x_n) = 0 \\ \implies \sum_{n=1}^N A_{nk} \mu_k &= \sum_{n=1}^N A_{nk} x_n \\ \mu_k \sum_{n=1}^N A_{nk} &= \sum_{n=1}^N A_{nk} x_n \\ \mu_k = \frac{\sum_{n=1}^N A_{nk} x_n}{\sum_{n=1}^N A_{nk}} \end{aligned}$$

Числитель - это сумма всех расстояний от каждого экземпляра до центроидов в кластере. Знаменатель - это количество экземпляров в кластере. Таким образом, центройд μ_k кластера является барицентром экземпляров кластера. Есть очень неплохая [визуализация](#) K-means от Stanford'a.

5.3.2 Разбиваем на 3 кластера

Для начала я выбрал случайное количество кластеров. Например $k = 3$. Проведя кластеризацию, с помощью K-means мы получили следующее разбиение.



Обсудим немного, что у нас получилось. Ну, очевидно, что количество кластеров равное 3 очень мало. И тут мы понимаем проблему, а как определить количество кластеров? Понятно, что точное количество кластеров мы ни когда не получим. Давайте рассмотрим методы, получения оптимального количества кластеров:

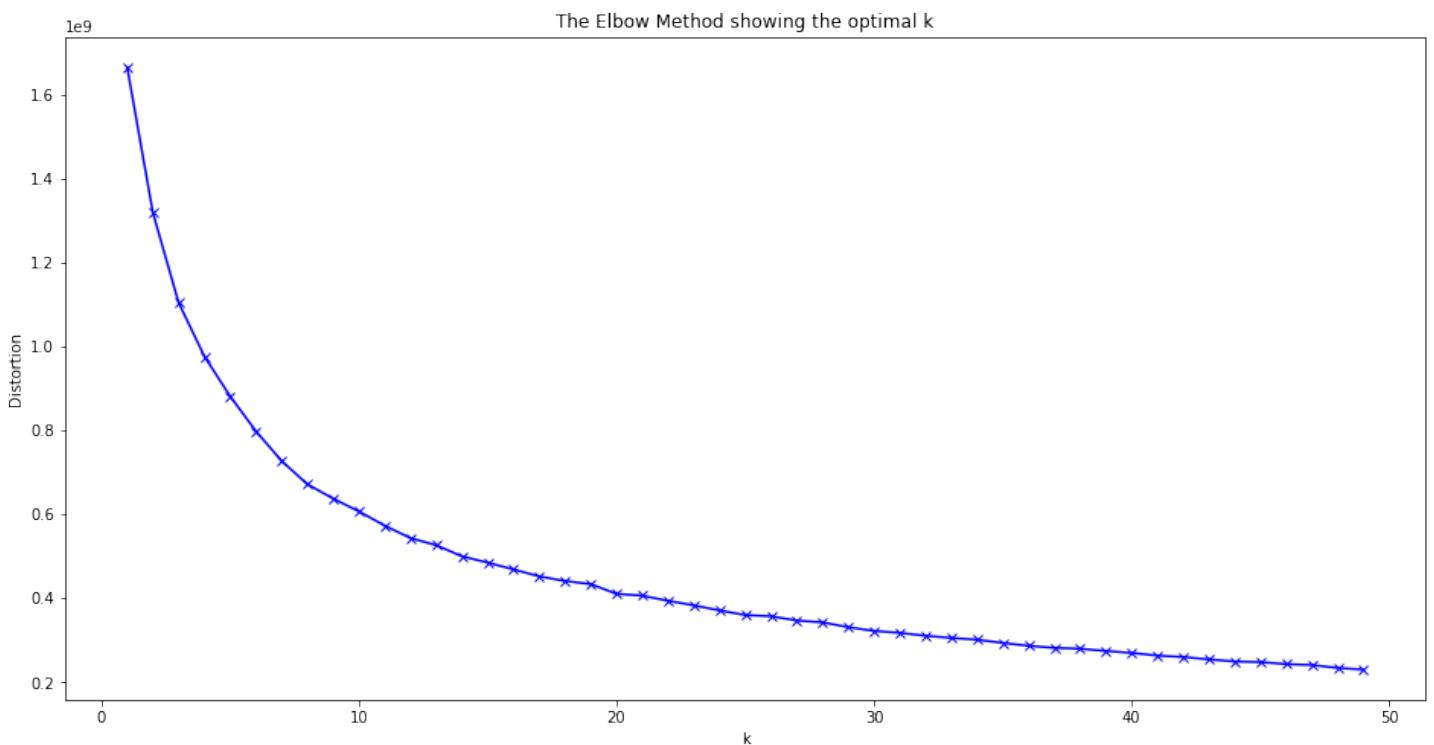
- Elbow Method
- Silhouette Score Method
- Другие алгоритмы кластеризации

5.3.2.1 Elbow Method

Elbow Method - это эвристический метод, поэтому какого-то математического пояснения нету. На глаз, так сказать.

Elbow Method рассматривает процент дисперсии, описываемый как функция от количества кластеров: следует выбрать количество кластеров, чтобы добавление еще одного кластера не давало лучшего моделирования данных. Точнее, если построить график зависимости процента дисперсии, объясняемой кластерами, от количества кластеров, первые кластеры объяснят большую разницу, но в какой-то момент процент объясненных дисперсий упадет, отрисовывая угол на графике. Этот изгиб и определяет количество кластеров, отсюда и «Elbow Method». Этот «локоть» не всегда можно однозначно идентифицировать, что делает этот метод очень субъективным и ненадежным. Процент объясненной дисперсии - это отношение дисперсии между группами к общей дисперсии, также известное как F-тест. Небольшая вариация этого метода отображает кривизну внутригрупповой дисперсии.

Я получил следующий график.



Т.к. какого-то резкого изгиба нету - отказываемся от данного метода.

5.3.2.2 Silhouette method

Silhouette score - это мера того, насколько объект подходит своему собственному кластеру (cohesion) по сравнению с другими кластерами (separation). Silhouette score варьируется от -1 до +1, где высокое значение указывает, что объект вероятнее всего соответствует своему собственному кластеру и не соответствует соседним кластерам. Если большинство объектов имеют высокую Silhouette score, то кластеризация прошла успешно. Если многие точки имеют низкое или отрицательное Silhouette score, то мы выбрали либо слишком много либо слишком мало кластеров.

Предположим, что данные были сгруппированы с помощью любого метода, например K-means, в k кластеров.

Для точки данных $i \in C_i$ (точка данных i находится в кластере C_i), тогда пусть

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j)$$

будет средним расстоянием между i и всеми другими точками данных в том же кластере, где $d(i, j)$ - расстояние между точками данных i и j в кластере C_i (мы делим на $|C_i| - 1$ потому что мы не

включаем расстояние $d(i, i)$ в сумму). Мы можем интерпретировать $a(i)$ как меру того, насколько сильно (близко/хорошо/вероятно, я не знаю как объяснить) i принадлежит своему кластеру (чем меньше значение, тем более вероятно, что данная точка данных принадлежит своему кластеру).

Затем мы определяем среднее несходство точки i с некоторым кластером C_k как среднее расстояние от i до всех точек в C_k (где $C_k \neq C_i$).

Для каждой точки данных $i \in C_i$, теперь определим

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j)$$

быть наименьшим (отсюда оператор \min в формуле) средним расстоянием от i до всех точек в любом другом кластере, членом которого i не является. Кластер с этим наименьшим средним несходством называется «соседним кластером» i , потому что этот соседний кластер, наиболее подходящий для точки i .

Теперь мы определяем Silhouette score одной точки данных i

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{if } |C_i| > 1 \\ |C_i| > 1$$

и

$$s(i) = 0, \text{ если } |C_i| = 1 |C_i| = 1$$

Что в общем-то можно переписать, как

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$

Из определения выше мы понимаем, что

$$-1 \leq s(i) \leq 1$$

Обратите внимание, что $a(i)$ четко не определен для кластеров с размером 1, и в этом случае мы устанавливаем $a(i) = 0$. Этот выбор произвольный, но нейтральный (я имею ввиду, что он находится по середине между, -1 и 1.)

Чтобы $s(i)$ было близко к 1, нам требуется $a(i) \ll b(i)$. Поскольку $a(i)$ является мерой того, насколько i отличается от своего кластера, небольшое значение означает, что он достаточно точно с ним совпадает. Более того, большой $b(i)$ означает, что i плохо соответствует своему соседнему кластеру. Таким образом, значение $s(i)$, близкое к 1, означает, что данные правильно сгруппированы. Если $s(i)$ близко к -1, затем по той же логике мы видим, что i было бы более подходящим, если бы он был сгруппирован в соседнем кластере. $s(i)$ около нуля означает, что датум находится на границе двух кластеров.

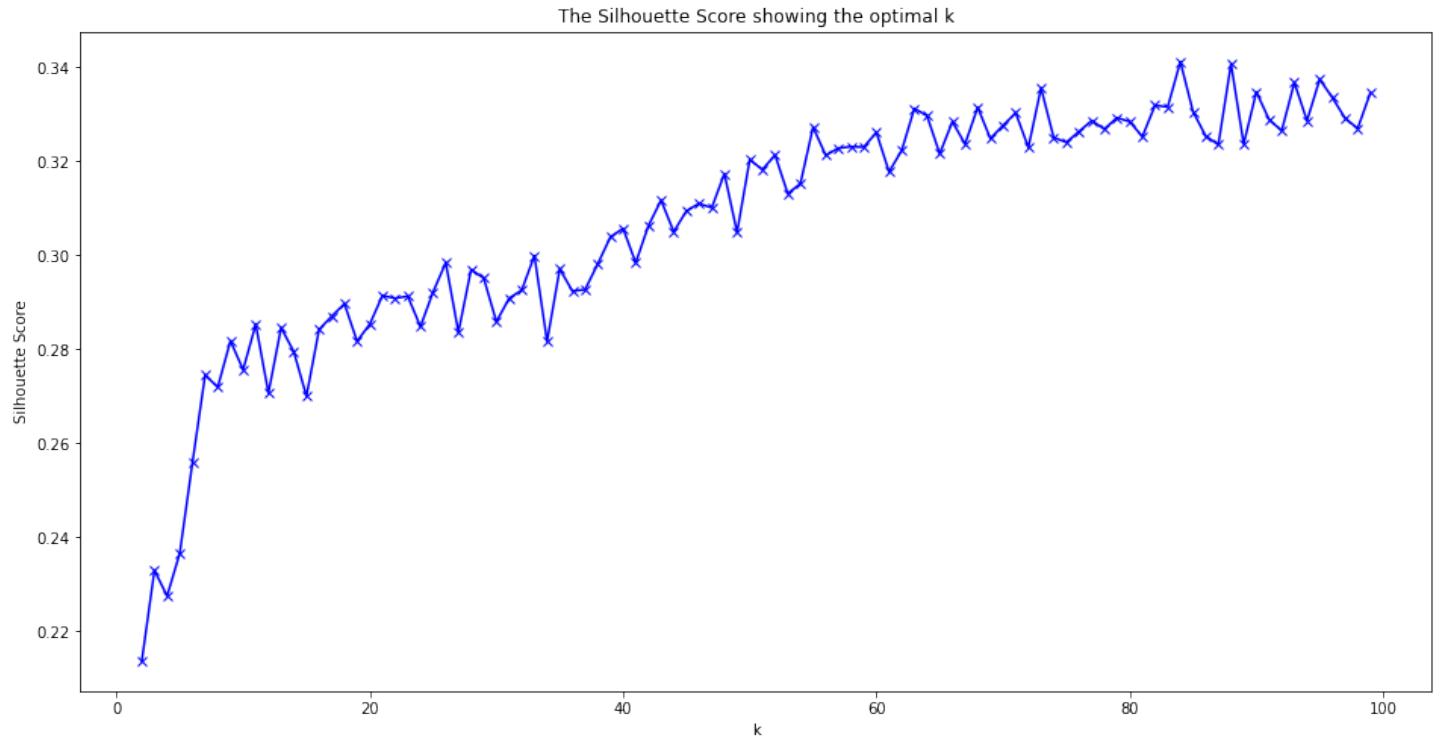
Среднее значение $s(i)$ по всем точкам кластера является мерой того, насколько плотно сгруппированы все точки в кластере. Таким образом, среднее значение $s(i)$ по всем данным всего набора данных является мерой того, насколько правильно были сгруппированы данные. Если кластеров слишком много или слишком мало, что может произойти, когда в алгоритме кластеризации используется неправильный выбор k (например, k-means), некоторые из кластеров обычно будут отображать гораздо более узкие силуэты, чем остальные. Таким образом, графики силуэтов и средние могут использоваться для определения числа кластеров для набора данных. Можно также увеличить вероятность того, что силуэт будет максимизирован при правильном количестве кластеров, путем повторного масштабирования данных с использованием весовых коэффициентов фитч, которые зависят от кластера.

Silhouette coefficient для максимального значения среднего $s(i)$ для всех данных для всего набора данных

$$SC = \max_k \tilde{s}(k)$$

Где $\tilde{s}(k)$ представляет среднее $s(i)$ по всем данным всего набора данных для определенного количества кластеров k .

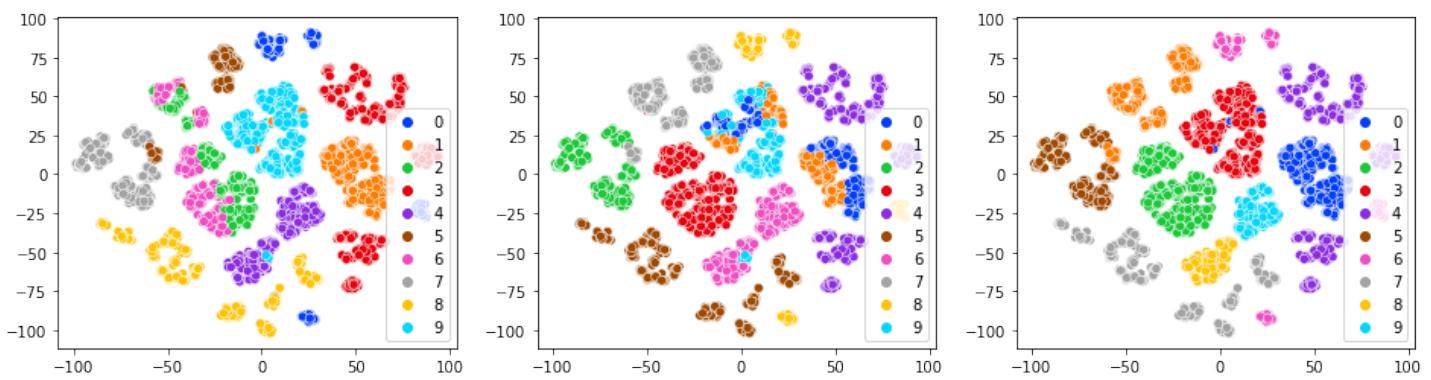
Для нашего набора данных для $k = 1, \dots, 100$ я получил следующий график



Уже 100 кластеров, а он всё растёт и растёт. Ладно, обсудим почему так выходит.

5.3.2.3 Главная проблема K-means

Запустим три раза K-means для 7 кластеров каждый и взглянем на результаты.



Проблема очевидна: разное разбиение на кластеры. Так происходит из-за рандомной инициализации центроидов в начале алгоритма K-means. Однако данный алгоритм всё-равно очень часто применяется на практике, что мы увидим чуть-чуть попозже. А пока нам всё ещё надо определить количество кластеров.

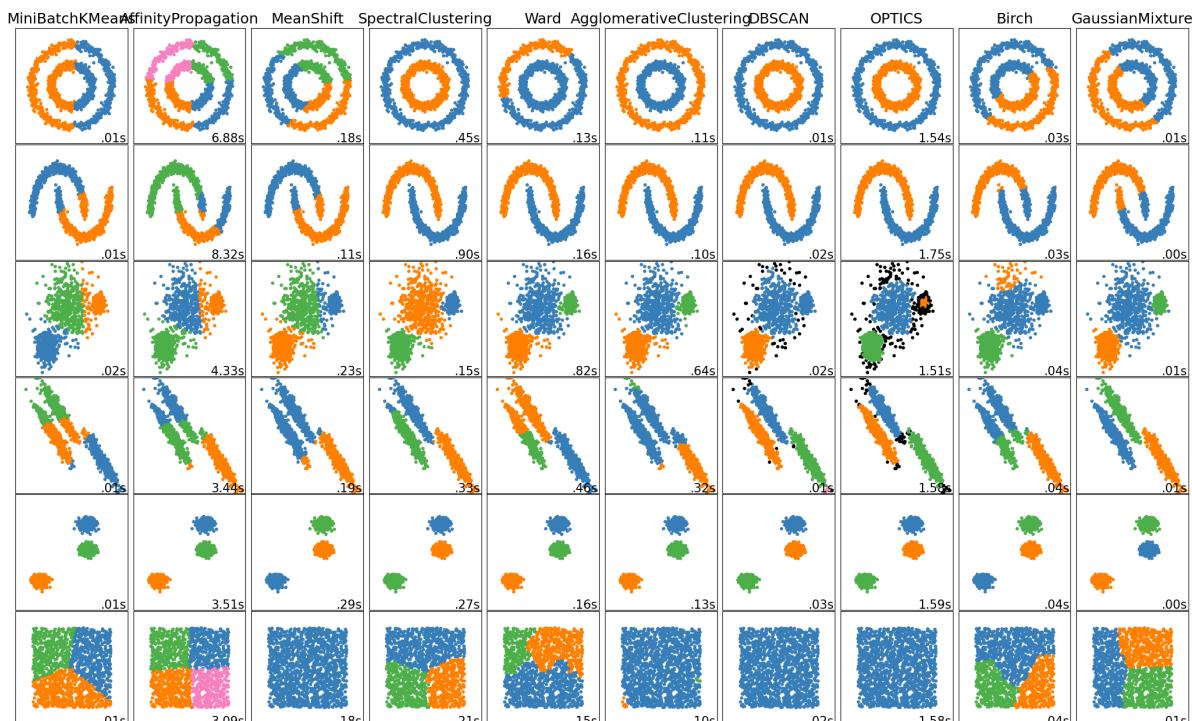
5.3.2.4 Другие алгоритмы кластеризации

Вспомним, какие базовые типы алгоритмов существуют

- Кластеризация основаная на центройдах. (Например: K-means или MeanShift)
- Кластеризация основаная на плотности распределения (Например: DBSCAN)
- Кластеризация основаная на типе распределения (Например: Gaussian Mixture Model)
- Иерархическая кластеризация

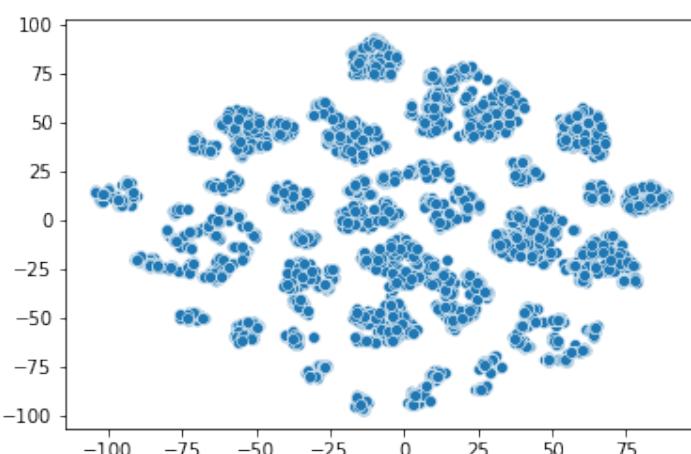
Первый вариант нам не подходит, по причине описанной выше, точнее он подходит, но об этом чуть-чуть попозже. Второй вариант - да, он вполне нас устраивает, поскольку если взглянуть на данные - в целом, на глаз, мы можем сами разбить на кластеры, основываясь на плотности распределения данных. Третий вариант, впринципе тоже можно, но наши данные вроде не распределены, согласно Гауссовскому распределению. Поэтому оставим это на запасной вариант. И последний вариант нам не подходит, ну т.к. тут нету какой-то иерархии.

У [Scikit-Learn](#) есть неплохое сравнение разных алгоритмов на разных паттернах.



5.3.2.5 Density-based spatial clustering of applications with noise

Чтобы понять что такое DBSCAN и почему я решил его использовать, давайте взглянем на данные.



Если оценить на глаз, разные скопления данных имеют разную форму. Более того в одном скоплении данных намного больше, чем другом. Где-то это вообще змейка какая-то. Проблемное распределение данных, здесь явно не подойдёт Gaussian Mixture Model, как я предполагал ранее.

Впринципе основания применения DBSCAN следующие

- Количество кластеров значения не имеет. ✓

- **Core point (Основная точка):** Данная точка должна иметь не менее minPts точек на расстоянии ϵ от себя.
- **Border point (Граничная точка):** Это точка, у которой есть хотя бы одна основная точка на расстоянии ϵ от себя.
- **Nois (Шум):** В нашем случае это зелёная точка. Это точка, которая не является ни основной, ни граничной. И имеет менее minPts точек на расстоянии ϵ от себя.

Теперь пошагово опишем алгоритм.

1. Алгоритм действует путем произвольного выбора точки в наборе данных (до тех пор, пока все точки не будут пройдены).
2. Если есть по крайней мере minPoint точек в радиусе ϵ от точки, мы считаем, что все эти точки являются частью одного кластера.
3. Затем кластеры расширяются путем рекурсивного повторения вычисления окрестности для каждой соседней точки.

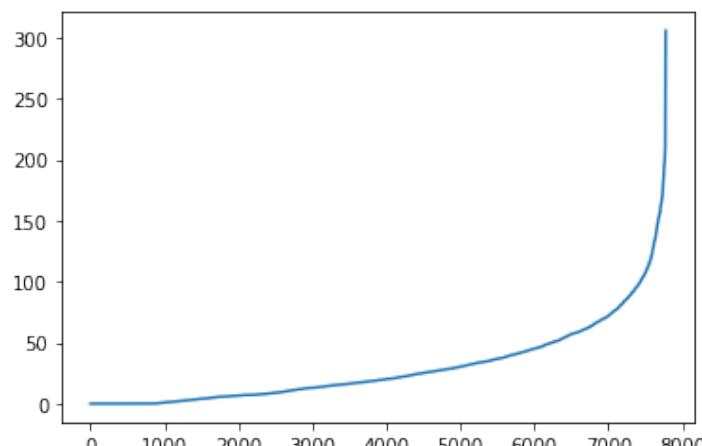
Это всё круто, но как же вычислить параметры.

- **minPts :** Как показывает опыт, минимальное значение minPts может быть получено из размерности \mathcal{D} в датасете, поскольку $\text{minPts} \geq \mathcal{D} + 1$. Низкое значение $\text{minPts} = 1$ не имеет смысла, поскольку тогда каждая точка сама по себе уже является кластером. При $\text{minPts} \leq 2$ результат будет таким же, как и при иерархической кластеризации с метрикой одиночной ссылки, с разрезом дендрограммы по высоте ϵ . Следовательно, minPts необходимо выбрать не менее 3. Однако большие значения обычно лучше для наборов данных с шумом и будут давать более значимые кластеры. Как правило, можно использовать $\text{minPts} = 2 \times \text{dim}$, но может потребоваться выбрать большие значения для очень больших наборов данных, для зашумленных данных или данных, содержащих много дубликатов.
- **ϵ :** Значение ϵ может быть выбрано с помощью графа k -расстояний, на котором расстояние до ближайшего соседа $k = \text{minPts}-1$ упорядочено от наибольшего к наименьшему значению. Хорошие значения ϵ находятся там, где этот график показывает «изгиб»: если ϵ выбрано слишком маленьким, большая часть данных не будет кластеризована; тогда как при слишком большом значении ϵ , кластеры объединяются, и большинство объектов будут в одном кластере. Как правило, предпочтительны небольшие значения ϵ , и, как показывает практический опыт, только небольшая часть точек должна находиться на этом расстоянии друг от друга.

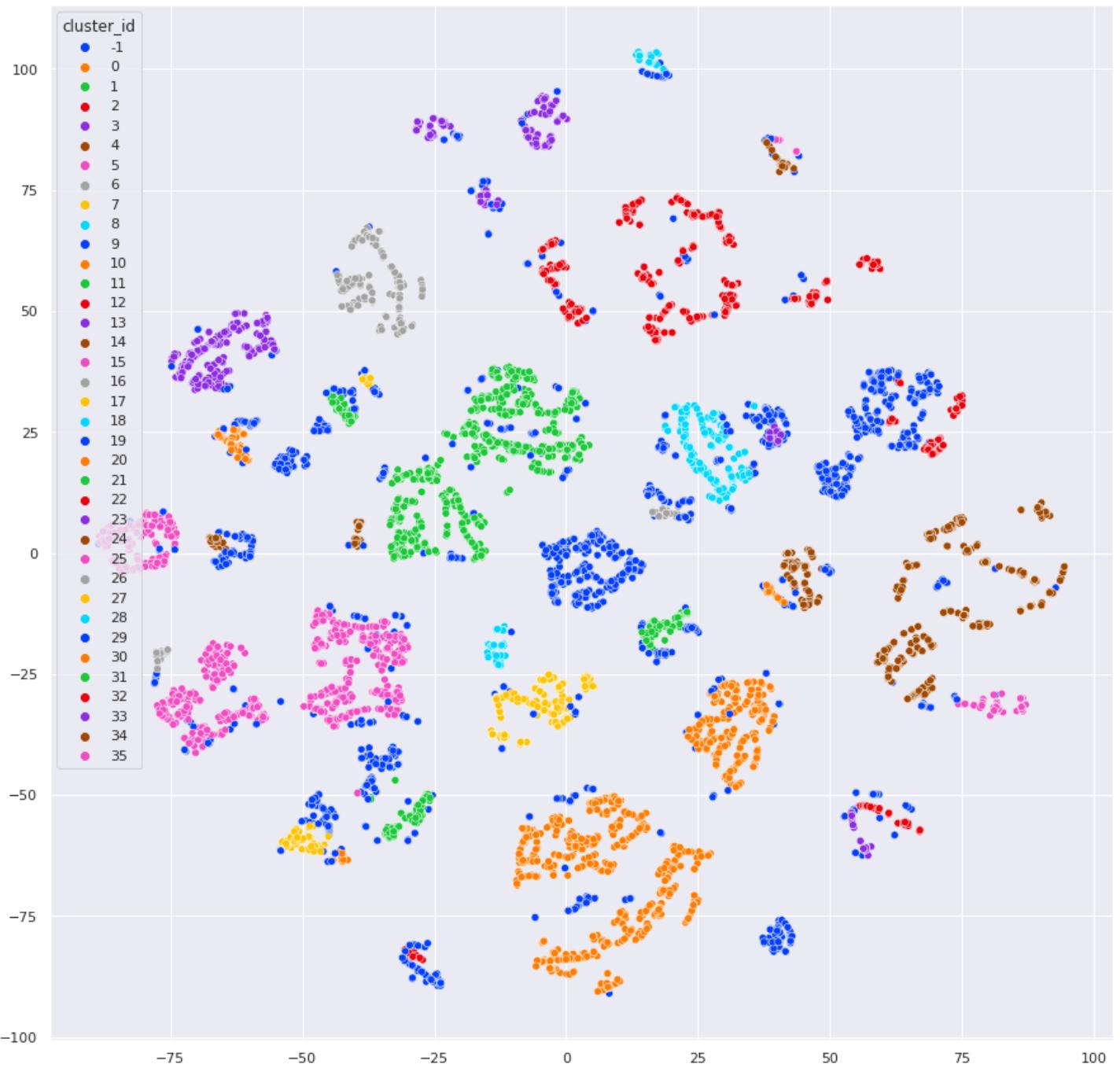
5.3.2.6 Применение DBSCAN

Для начала выбираем параметры. На данный момент размерность моего датасета 7777×8 . Тогда $\text{minPts} = 2 \times 8 = 16$.

Для ϵ строим модель [k-ближайших соседей](#). И получаем график k -расстояний. И строим график данных расстояний.



Примерный $\varepsilon \in [90; 130]$. Я взял 125. Проведя кластеризацию я получил следующие кластеры



Попрошу заметить, что похожие цвета могут обозначать разные кластеры. У меня получилось 36 начальных кластеров. Данное значение мы будем использовать, как опорное количество кластеров. Фухххх, с первой частью покончено.

5.4 Кластеризация синопсисов

Вся кластеризация описанная выше, была проводилась на датасете без учёта синопсиса. Полагаю пришло время поговорить и о нём. Напомню, что мы токенезировали и нормализовали наши синопсисы, получив массивы следующего типа

$$\text{Synopsis} = [6.04107934e^{-4}, 1.00100100e^{-4}, \dots, 1.50210294e^{-3}, 9.82343499e^{-1}]$$

для каждого синопсиса. Теперь нам предстоит найти схожие синопсисы и объединить их в группы. Кхм-кхм, начнём. Чтобы найти схожие синопсисы нам нужно найти схожие по смыслу слова (их численную репрезентацию). Но в данном виде у нас слишком много лишних слов в синопсисе. Предлагаю избавиться от лишних (ненужных) слов. Но как это сделать? Первое, что приходит

в голову это снизить размерность вектора, например с помощью t-SNE. Но, помните [я говорил](#), что у t-SNE есть минус: ему слишком тяжело обрабатывать слишком большую размерность (где-то я читал, что граничное количество фитч для t-SNE это 32), что заметно сказывается на точности. И поэтому мы переходим к более совершенному методу уменьшения размерности - [автоэнкодеры](#).

5.4.1 Deep Embedding Clustering

Я решил объединить всё в одну тему. Начнём с того, что такое DEC.

Идея DEC это параметризованное нелинейное отображение из пространства данных X в пространство признаков Z более низкой размерности, в котором мы оптимизируем нашу кластеризацию. Мы используем стохастический градиентный спуск (SGD), прогоняя через обратное распространение ошибки на основе кластеризации.

Оптимизация DEC - непростая задача. Мы хотим одновременно определить кластер и представление объекта. Однако, в отличие от обучения с учителем, мы не можем обучить нашу глубокую сеть с помеченными (labeled) данными. Вместо этого мы предлагаем итеративно уточнять кластеры с помощью дополнительного целевого распределения, полученного из текущего мягкого распределения кластера.

Существует схожая идея - спектральная кластеризация, однако для неё требуется построить полносвязный граф и Матрицу Кирхгофа для этого графа, тем самым возводя сложность в квадрат квадратов, короче сложно будет нашему компутеру считать это всё. Однако данный метод линейный, соответственно и быстрый.

Определим проблему: нужно разбить n точек $\{x_i \in X\}_{i=1}^n$ на k кластеров, каждый из которых представлен центроидами μ_j , $j = 1, \dots, k$. Вместо того, чтобы проводить кластеризацию на пространстве X , мы сначала трансформируем данные в с помощью нелинейной проекции $f_\theta : X \rightarrow Z$, где θ обучаемые параметры, а Z неизвестное пространство данных. Однако, мы точно знаем, что размерность Z сильно меньше размерности пространства X , это сделано для того, чтобы избежать «проклятия размерности». Чтобы параметризовать f_θ будем использовать глубокие нейронные сети (ГНС).

Данный алгоритм кластеризует данные в то время, пока учится на центroidах кластеров $\{\mu_j \in Z\}_{j=1}^k$ в пространстве фитч Z при параметрах θ , которые проецируют данные на пространство Z . DEC имеет две фазы:

1. инициализация параметров с помощью автоэнкодера.
2. оптимизация параметров (т.е. кластеризация), при которой мы выполняем итерацию между расчитываемым и вспомогательным целевым распределением и минимизируем расхождения Кульбака – Лейблера (KL-divergence) между ними.

Начнём со второго.

5.4.1.1 Кластеризация с KL-divergence

Изначально, имея функцию отображения f_θ и центроиды кластеров $\{\mu_j\}_{j=1}^k$, мы пытаемся улучшить кластеризацию, с помощью алгоритма обучения без учителя, который состоит из двух шагов.

1. На первом этапе мы вычисляем мягкое сопоставление ([мягкая кластеризация](#)) между вложенными точками и центроидами кластера.
2. На втором этапе мы обновляем f_θ уточняем центроиды кластера, исследуя текущие присвоение с использованием вспомогательного целевого распределения.

Мягкое сопоставление Доверимся ван дер Маатену и Хинтону, и согласно [их работе](#) мы будем использовать t-распределение Стьюдента для того, чтобы оценить сходство между вложенной точкой

z_i и центроидом μ_j :

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2 / \alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'} (1 + \|z_i - \mu_{j'}\|^2 / \alpha)^{-\frac{\alpha+1}{2}}}$$

где $z_i = f_\theta(x_i) \in Z$ соответствует $x_i \in X$ после эмбединга, α - степень свободы t-распределения Стьюдента, а q_{ij} можно интерпретировать как вероятность отнесения элемента i к кластеру j . Т.к. кроссвалидацию использовать мы не можем то пусть $\alpha = 1$.

Минимизация KL-divergence Мы будем итеративно уточнять кластеры, обучаясь на их точном присвоении, с помощью дополнительного целевого распределения. В частности, наша модель обучается путем сопоставления мягкого сопоставления целевому распределению. А следовательно, наша цель - минимизировать KL-divergence между мягким сопоставлением q_i и вспомогательным распределением p_i

$$L = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Теперь важный шаг: выбираем P . Т.к. q_i это мягкое сопоставление, мы будем использовать более мягкие целевые вероятности. Точнее, наше целевое распределение должно обладать следующими свойствами:

1. усилить прогнозы (т.е. улучшить точность кластера)
2. уделять больше внимания назначенным точкам данных, которые с большой вероятностью в своём кластере
3. нормализовать вклад каждого центроида в потери, чтобы предотвратить искажение большими кластерами скрытого пространство фитч.

В нашем примере p_{ij} равна

$$p_{ij} = \frac{q_{ij}^2 / f_i}{\sum_{j'} q_{ij'}^2 / f_i}$$

где $f_i = \sum_j q_{ij}$ - частоты мягких кластеров.

Как и при самообучении, мы берем начальный классификатор и немаркованный набор данных, а затем маркируем набор данных классификатором, чтобы обучать его собственным прогнозам с высокой степенью точности.

Оптимизация Мы одновременно оптимизируем центры кластеров $\{\mu_j\}$ и параметры ГНС θ , используя стохастический градиентный спуск (SGD) с импульсом. Градиенты L относительно встраивания пространств признаков каждой точки данных z_i и каждого центроида μ_j кластера вычисляются как:

$$\begin{aligned} \frac{\partial L}{\partial z_i} &= \frac{\alpha+1}{\alpha} \sum_j \left(1 + \frac{\|z_i - \mu_j\|^2}{\alpha}\right)^{-1} \times (p_{ij} - q_{ij})(z_i - \mu_j) \\ \frac{\partial L}{\partial \mu_j} &= -\frac{\alpha+1}{\alpha} \sum_i \left(1 + \frac{\|z_i - \mu_j\|^2}{\alpha}\right)^{-1} \times (p_{ij} - q_{ij})(z_i - \mu_j) \end{aligned}$$

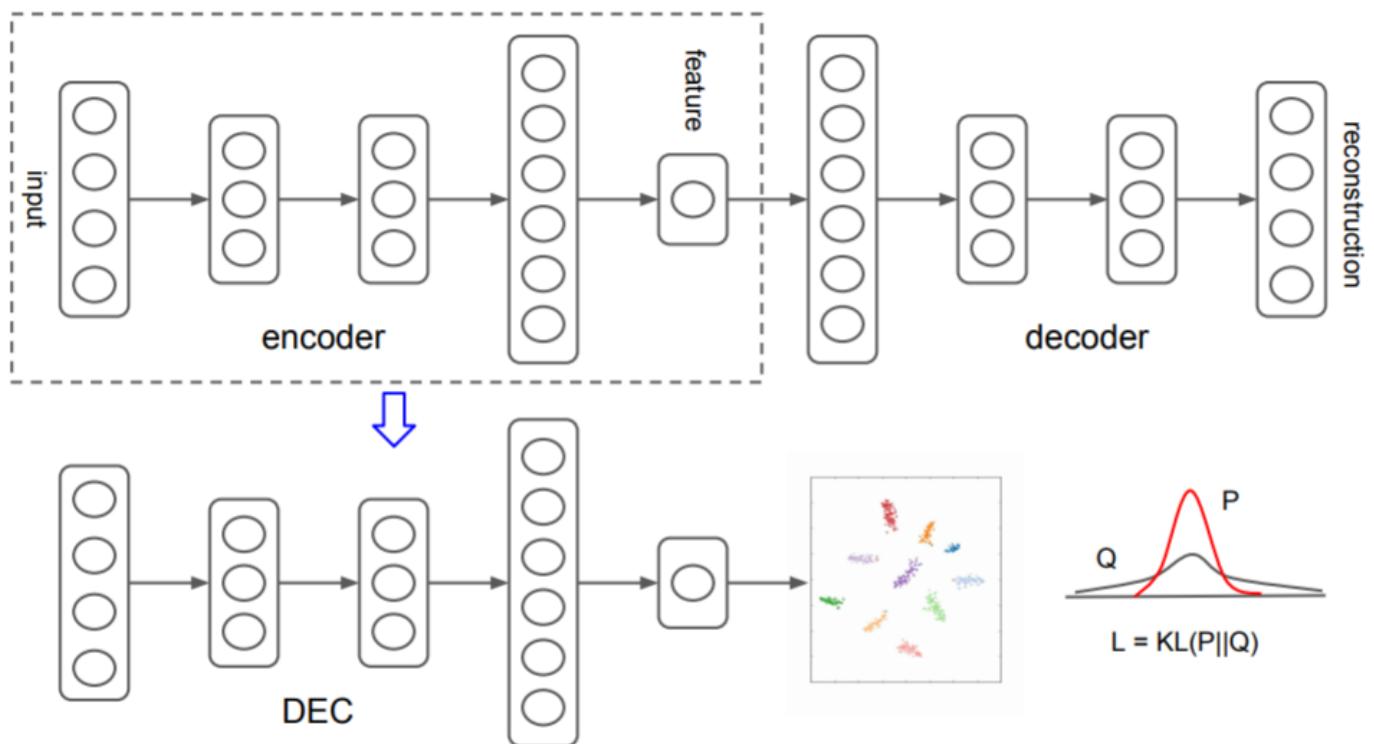
Затем градиенты $\frac{\partial L}{\partial z_i}$ передаются в ГНС и используются в стандартном обратном распространении для вычисления градиента параметра ГНС $\frac{\partial L}{\partial \theta}$. С целью обнаружения кластерных назначений мы останавливаем нашу процедуру, когда менее чем $tol\%$ точек меняют назначение кластера между двумя последовательными итерациями.

5.4.1.2 Инициализация параметров

До сих пор мы обсуждали, как DEC действует при начальных параметрах ГНС θ и центроидов кластера $\{\mu_j\}$. Теперь обсудим, как инициализируются параметры и центроиды.

Мы инициализируем DEC с помощью составного автоэнкодера (SAE), потому что недавние исследования ([Reducing the Dimensionality of Data with Neural Networks, Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion](#)) показали, что они создают семантически значимые и хорошо разделенные представления в реальных наборах данных.

Структура нейронной сети следующая:



5.4.1.3 Автоэнкодер

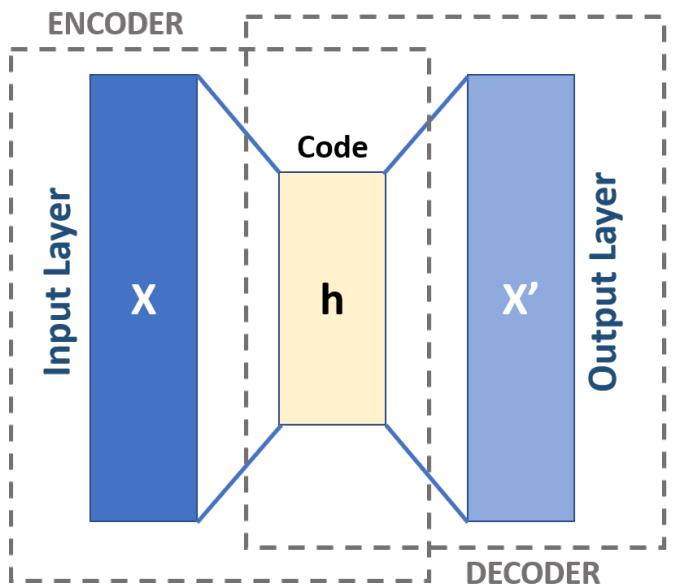
Здесь будет краткое, (ахахахахахахах, простите, это уже истерический смех со слово "краткое") пояснение, что такое автоэнкодер. Мы здесь не будем рассматривать разные виды автоэнкодеров (на [Википедии](#) всё есть). Просто взглянем на базовую архитектуру.

Автоэнкодер это архитектура нейронной сети, которая состоит из двух частей

- $\phi : \mathcal{X} \rightarrow \mathcal{F}$
- $\psi : \mathcal{F} \rightarrow \mathcal{X}$
- $\phi, \psi = \arg \min_{\phi, \psi} \|X - (\psi \circ \phi)X\|^2$

В простейшем случае, учитывая один скрытый слой, этап кодировщика автоэнкодера принимает входные данные $\mathbf{x} \in \mathbb{R}^d = \mathcal{X}$ и проецирует их на $\mathbf{h} \in \mathbb{R}^p = \mathcal{F}$:

$$\mathbf{h} = \sigma(\mathbf{Wx} + \mathbf{b})$$



Это отображение \mathbf{h} обычно называется кодом, скрытыми переменными или скрытым представлением.

Здесь σ - это поэлементная функция активации, такая как сигмовидная функция или RELU. \mathbf{W} - это матрица весов, а \mathbf{b} - вектор смещений (баеса). Веса и смещения обычно инициализируются случайнным образом, а затем обновляются итеративно во время обучения посредством обратного распространения ошибки. После того, этап декодирования автоэнкодера отображает \mathbf{h} на реконструкцию \mathbf{x}' той же формы, что и \mathbf{x} :

$$\mathbf{x}' = \sigma'(\mathbf{W}'\mathbf{h} + \mathbf{b}')$$

где σ' , \mathbf{W}' и \mathbf{b}' для декодера может не иметь отношения к соответствующему σ , \mathbf{W} и \mathbf{b} для энкодера.

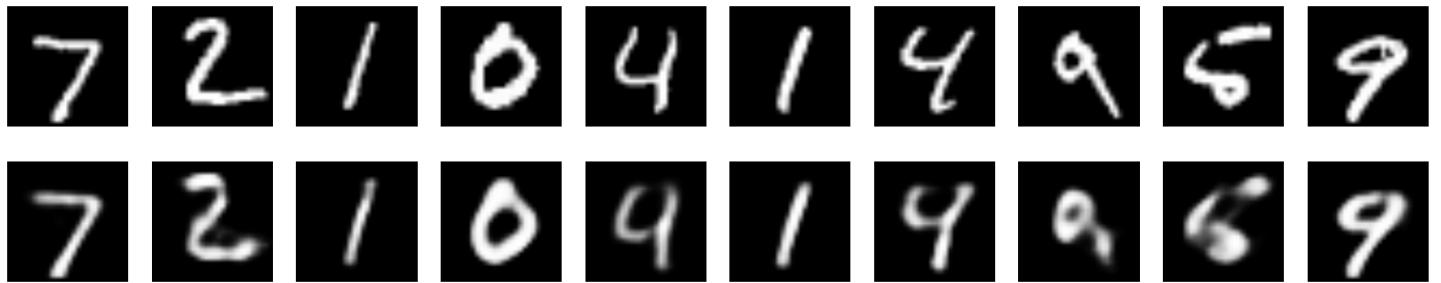
Автоэнкодеры обучены минимизировать потери:

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2 = \|\mathbf{x} - \sigma'(\mathbf{W}'(\sigma(\mathbf{W}\mathbf{x} + \mathbf{b})) + \mathbf{b}')\|^2$$

где \mathbf{x} обычно усредняется по некоторому входному обучающему набору.

Как упоминалось ранее, обучение автокодировщика выполняется путем обратного распространения ошибки, как и в обычной нейронной сети с прямой связью. Если пространство функций \mathcal{F} имеет меньшую размерность, чем пространство ввода \mathcal{X} , вектор фитч $\phi(x)$ можно рассматривать как сжатое представление ввода x . Это случай неполных автоэнкодеров.

Вот пример работы автоэнкодера



А теперь, вернувшись немного назад, помните: я говорил, что нам нужно уменьшить размерность вектора синопсисов, что означает выкинуть ненужные слова, сохранив самые важные. Нуууу, и как вам идея воспользоваться этой шайтан-машиной. Вдобавок для кластеризации используем DEC, чтобы разбить синопсисы на кластеры.

5.4.2 Применение шайтан-машины (DEC)

Прежде, чем что-то применять, нам надо это что-то создать. Сначала я создал симметричный автоэнкодер, с количеством слоёв: 4 - на энкодер, 4 - на декодер. На вход ГНС получает 1500 входных данных (я расширил синопсис до 1500, чтобы получить более разряженные данные). Далее следуют 3 слоя с RELU активацией, которые сходятся к коду - это энкодер. Далее код уменьшает размерность. Потом 3 слоя с RELU активацией которые востанавливают данные - декодер. И выходной слой на 1500 (такой же как и входной). Целевая функция - RMSPROP.

Далее DEC, мы берём энкодер из автоэнкодера и соединяем его со слоем кластеризации. На вход слой кластеризации получает, соответственно, уменьшенную размерность, а на выходе возвращает откластеризованные данные. Количество кластеров выбирается согласно подсчитанному количеству с помощью DBSCAN в [секции выше](#). Ну а сам кластерный слой создан по принципу описанному в DEC, просто минимизируем KL-divergence, между t-распределением Стьюдента и целевым распределением описанным выше.

У меня получились следующие архитектуры

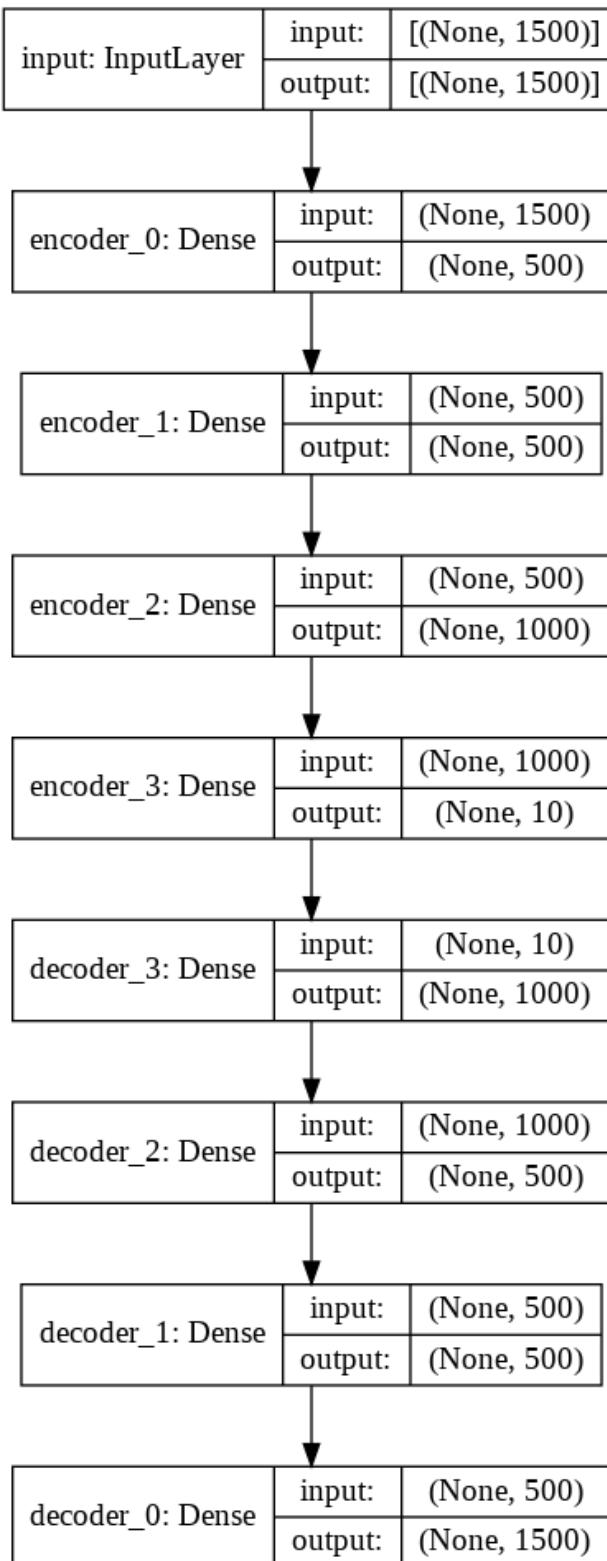


Figure 18: Autoencoder

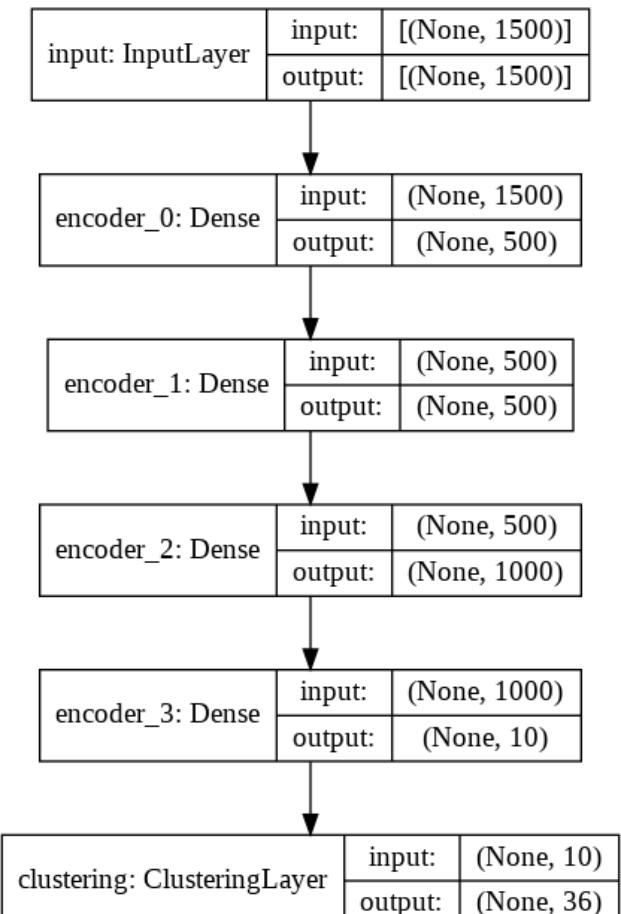


Figure 19: DEC

Я хочу заметить, что на данный момент мы проводим кластеризацию только на синопсисах, т.е. другие фитчи не учитываются и формы кластеров будут совершенно иные.

На следующей странице вы можете увидеть кластеры синопсисов.

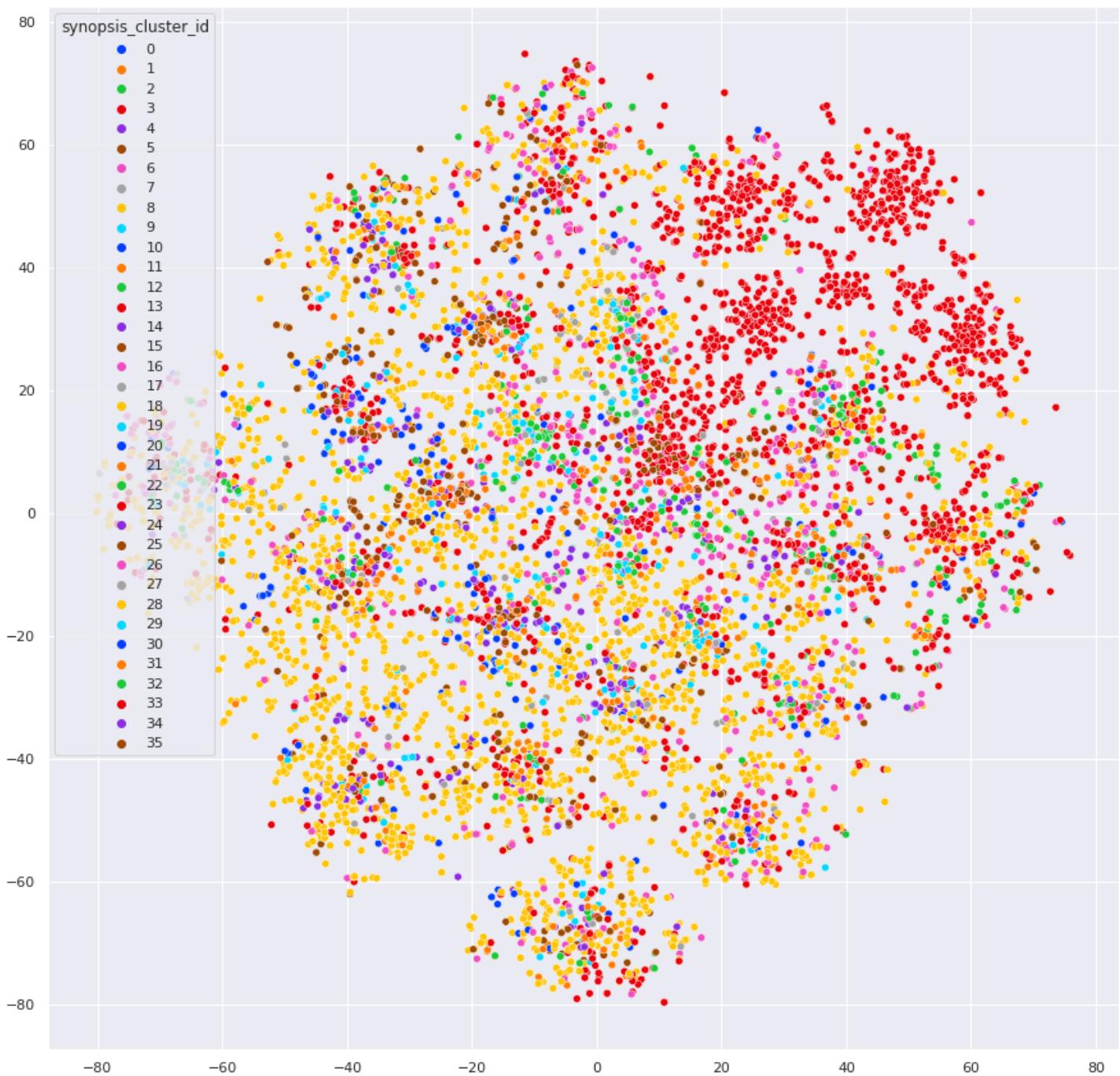


Figure 20: Synopsis clusters

Опять таки, напоминаю, что у разных кластеров могут быть похожие цвета из-за цветовой палитры. Ну и давайте возьмём какойнибудь кластер и сравним два фильма отсортированных по жанрам. И теперь самое интересное: я выбрал разные жанры (я смотрел эти фильмы):

- [Код да Винчи](#)
- [Три икса 2: Новый уровень](#)

Взглянем на их синопсисы:

- **Код да Винчи**

Гарвардского профессора Роберта Лэнгдона подозревают в чудовищном преступлении, которого он не совершал. Лэнгдон знакомится с криптографом парижской полиции Софи Невё и вместе с ней пытается раскрыть тайну, которая может подорвать могущество католической церкви.

- **Три икса 2: Новый уровень**

Группа высокопоставленных заговорщиков в белом доме поднимает мятеж с целью захвата власти. Спецслужбы бессильны — бунтовщики разгромили Агентство Национальной Безопасности

(АНБ). Америка на грани хаоса, но у чудом выжившего офицера АНБ Огастеса Гиббонса остаётся последний козырь, способный спасти положение. Это Дариус Стоун — хладнокровный суперагент, отобранный по сверхсекретной программе «Три Икса». Оставляя после себя смерть и разрушения, этот человек-армия вступает в битву, которая решит судьбу Америки и всего мира.

И, хоть, у них разные жанры - у первого это Детектив/Триллер, а у второго Экшен. Скажу честно, по своей структуре они похожи, кто-то где-то бегает, стреляет, что-то выясняет и т.п.

Ну что, пошли к следующей, последней секции.

5.5 Итоговая кластеризация

Мы получили эмбединги для синописа, и теперь удалим **description**, а добавим **id** соответствующего кластера, для каждого синописа. Взглянем ещё раз на получившийся датасет.

show_id	type	director	release_year	duration	main_country	age_group	genres	main_genre	synopsis_cluster_id
s1	1	2716	2020	12	6	0	397	16	32
s2	0	1840	2016	2	42	0	309	12	26
s3	0	1289	2011	0	60	0	335	13	5
s4	0	3446	2009	0	75	1	47	0	16
s5	0	3177	2008	2	75	1	293	12	32

Проводим квантильную нормализацию и проводим кластеризацию. Однако, сейчас будем применять другой алгоритм, поскольку k-means всё так же не надёжен, а DBSCAN обрабатывает выбросы. Помните, я в описании DEC'а заикался про спектральную кластеризацию, а ещё мы сравнивали алгоритмы, и там четвёртый алгоритм, красавец такой стоит, **Spectral Clustering**. Я думаю все мы знаем, что сейчас будет:)) Правильно! Описание алгоритма.

5.5.1 Спектральная кластеризация

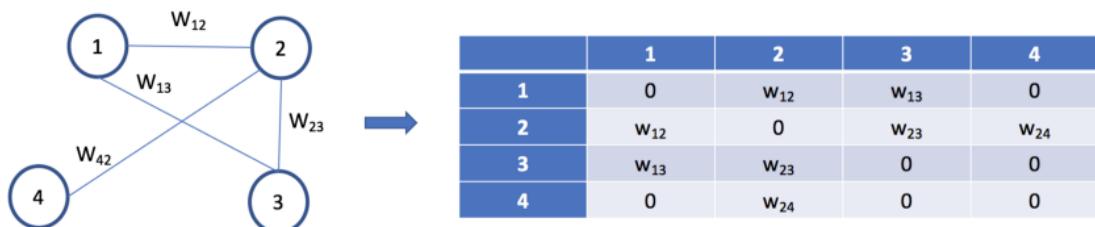
[Ссылка на видео](#) от Stanford'a по которому я когда-то изучал данный алгоритм.

Сперва на перво, нужно понимать, что спектральная кластеризация разбивает граф на части, именно график. Поэтому, первым делом из наших точек данных нужно составить график $G(V, E)$. Чтобы это сделать мы составляем матрицу схожести Q где элементами матрицы являются Евклидовые расстояния (Так же можно использовать К ближайших соседей или составить полно связанный график).

Далее мы задаём матрицу смежности A , по следующему правилу

$$A_{i,j} = \begin{cases} 1 & \text{если } Q_{i,j} > \varepsilon \\ 0 & \text{иначе} \end{cases}$$

где ε - это некоторый лимит расстояния на котором могут быть расположены соседи. В DBSCAN это было тоже ε . Получив данную матрицу мы можем спокойно построить график. (Если $A_{i,j} = 1$ тогда x_i соединена с x_j , иначе они не имеют ребра). В итоге получится, примерно, такая матрица



Следующий шаг, это уменьшение размерности.

Как видно на рисунке, точки данных в одном кластере также могут быть далеко - даже дальше, чем точки, которые находятся в разных кластерах. Наша цель состоит в том, чтобы преобразовать пространство так, чтобы, когда две точки находятся рядом, они всегда находились в одном кластере, а когда они далеко друг от друга, они находились в разных кластерах. Нам нужно спроектировать наши элементы на маломерное пространство. Для этого мы вычисляем матрицу Киргхофа (или Laplacian Graph), который является просто еще одним матричным представлением графа и может быть полезен при поиске интересных свойств графа. Это можно вычислить как:

$$L = D - A$$

где D это диагональная матрица

$$D_{ii} = \sum_j A_{ij}$$

Тем самым мы получаем следующую матрицу Киргхофа

	1	2	3	4
1	d_1	$-w_{12}$	$-w_{13}$	0
2	$-w_{12}$	d_2	$-w_{23}$	$-w_{24}$
3	$-w_{13}$	$-w_{23}$	d_3	0
4	0	$-w_{24}$	0	d_4

$d_1 = w_{12} + w_{13}$
 $d_2 = w_{12} + w_{23} + w_{24}$
 $d_3 = w_{13} + w_{23}$
 $d_4 = w_{24}$

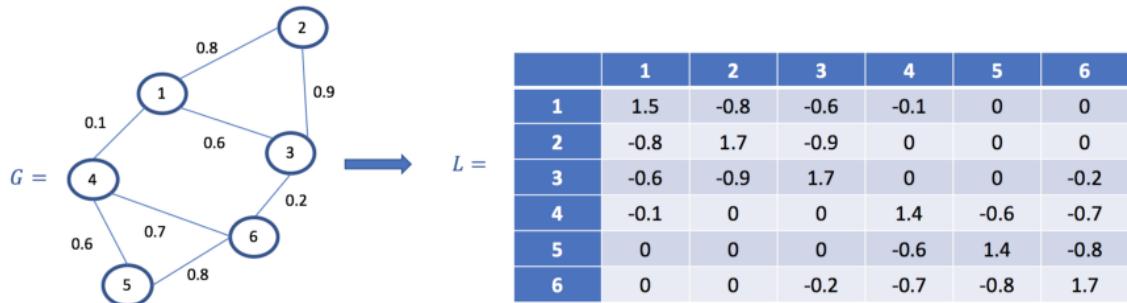
Вся цель вычисления матрицы Киргхофа состояла в том, чтобы найти для него собственные значения и собственные векторы, чтобы встроить точки данных в малоразмерное пространство. Итак, теперь мы можем пойти дальше и найти собственные значения. Мы знаем что:

$$L\lambda = \lambda v$$

где v это собственный вектор матрицы Киргхофа, соответствующий собственному значению λ . Тем самым мы получаем собственные значения $\{\lambda_1, \lambda_2, \dots, \lambda_n\}$, где $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ и собственные векторы $\{v_1, v_2, \dots, v_n\}$.

- Если L имеет собственное значение o с k различными собственными векторами, такое что $o = \lambda_1 = \lambda_2 = \dots = \lambda_k$, тогда граф G имеет k соединённых элементов.
- Если G является соединённым, т.е. $o = \lambda_1$ и $\lambda_2 > o$, тогда λ_2 это алгебраическая связность графа G . Следовательно, чем больше λ_2 , тем больше связность.

Давайте рассмотрим пример с циферками

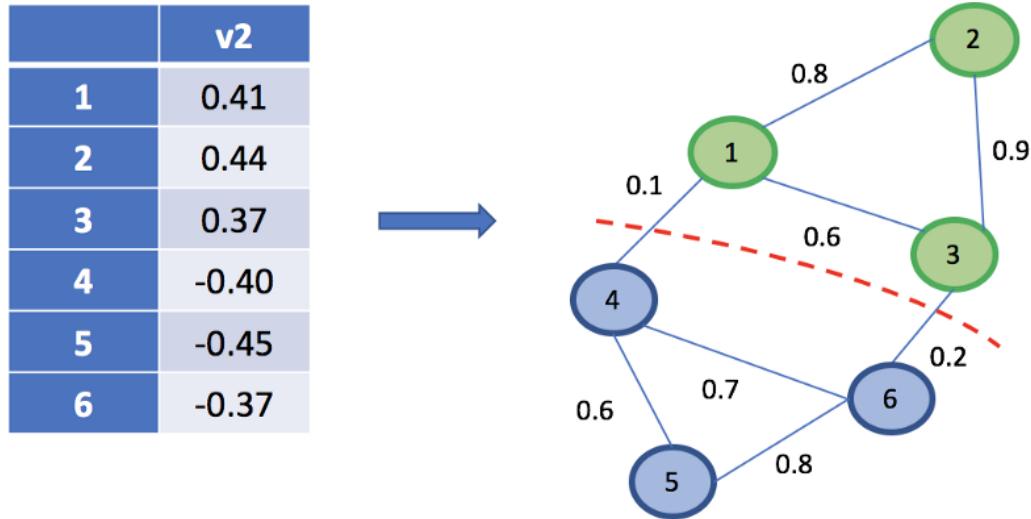


Теперь данный граф нужно разбить на кластеры

На этом шаге мы используем собственный вектор, соответствующий 2-му собственному значению, чтобы присвоить значения каждому узлу. При вычислении второе собственное значение равно 0,189, а соответствующий собственный вектор $v_2 = [0, 41, 0, 44, 0, 37, -0, 4, -0, 45, -0, 37]$. Чтобы получить двустороннюю кластеризацию (2 отдельных кластера), мы сначала назначаем каждый элемент v_2 узлам таким образом, чтобы $\{\text{node}_1 : 0.41, \text{node}_2 : 0.44, \dots, \text{node}_6 : -0.37\}$. Затем мы разделяем узлы

так, чтобы все узлы со значением > 0 находились в одном кластере, а все остальные узлы - в другом кластере. Таким образом, в этом случае мы получаем узлы 1, 2 и 3 в одном кластере, а 4, 5 и 6 во 2-м кластере.

Примечание: Второе собственное значение указывает, насколько тесно связаны узлы в графе. Для хорошего и чистого разделения уменьшите второе собственное значение, получите более качественные кластеры.



Если изначально задано k кластеров, тогда мы должны нормализовать нашу матрицу Киргхофа следующим образом:

$$L_{norm} = D^{-1/2} L D^{-1/2} =$$

	1	2	3	4	5	6
1	1.0	-0.5	-0.4	-0.1	0	0
2	-0.5	1.0	-0.5	0	0	0
3	-0.4	-0.5	1.0	0	0	-0.1
4	-0.1	0	0	1.0	-0.4	-0.5
5	0	0	0	-0.4	1.0	-0.5
6	0	0	-0.1	-0.5	-0.5	1.0

- Для k кластеров вычисляются первые $\{v_1, v_2, \dots, v_k\}$
- Формируем новую матрицу, где векторы - это колонки
- Представляем каждый узел, как соответствующую строку этой новой матрицы. Эти строки составляют вектор фитч для узлов.
- Используем K-Means чтобы разбить на k кластеров.

Вообщем-то всё, в этом и заключается алгоритм.

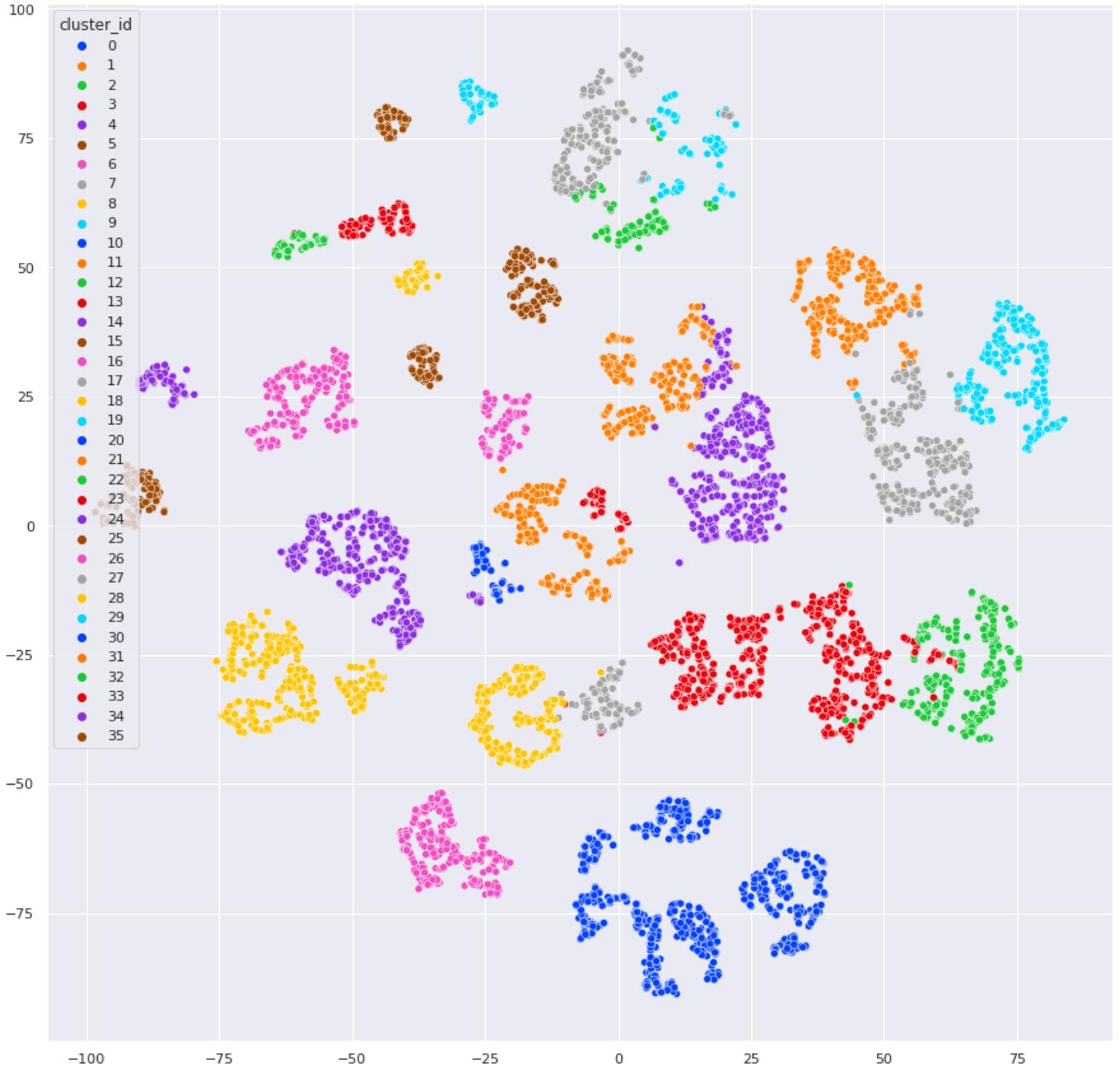
5.5.2 Применение Спектральной кластеризации

Прежде, чем применять предлагаю определиться с параметрами. Мы будем пользоваться уже готовой [реализацией от scikit-learn](#). А там немного усечённая версия.

- Поиском собственных значений и векторов займётся [ARPACK](#).
- Количество кластеров мы установим то же, которое получили с помощью DBSCAN, а точнее 36.

- Создание матрицы смежности будет происходить посредством k-ближайших соседей (там нету Евклидова расстояния)
- И я выбрал минимум 16 ближайших соседей (ну типа как и minPts в DBSCAN)

Проведя кластеризацию мы получили вот такую красотицу



Уфффф, какая красота. Ну давайте протестируем.

6 Оценка и итог

Поговорим немного о том, как себя проверить, ну конкретно в данном наборе данных ни как. Да и вы сами сможете откластеризовать фильмы, здесь скорее всё по наитию. Есть на самом деле много оценок качества кластеризации, например самая часто используемая - [ARI](#). Но, поскольку мы не имеем целевого разбиения, то нам и сравнивать не с чем. Предлагаю, просто взять какойнибудь фильм, и вывести топ 10 на него похожих.

Самый маленький кластер состоит из 24 элементов, т.е. мы спокойно можем получить топ 10

фильмов/ТВ-шоу для любого фильма/ТВ-шоу. Однако, самый маленький кластер синописса - 6 элементов, поэтому мы будем выводить топ 5 фильмов.

Для теста я выбрал фильм [«21»](#). Основные характеристики данного фильма следующие

- год выпуска: 2008
- страна производства: США
- продолжительность: 1.5 - 2.5 часа
- возрастная группа: подростки
- список жанров: Драма

Сразу прошу заметить, это не очень крупный датасет, здесь нету прямо очень похожих фильмов, (я проверял: попытался найти «Игрок» (англ: The Gambler), или «Уолл-стрит: Деньги не спят» (англ: Wall Street: Money Never Sleeps netflix)) а это достаточно похожие фильмы, по крайней мере так говорит реальная рекомендательная система. Датасет из 7777 фильмов и ТВ-шоу - очень мал. Но всё же, что мы имеем.

1. Сначала я выбрал все фильмы кластера, в котором находится наш фильм «21»
2. Потом я выбрал кластер синопсисов, которому принадлежит наш фильм «21»
3. Далее я отсортировал по основному жанру
4. И наконец выбрал топ 5 фильмов

У меня получился следующий список фильмов (я его немного урезал в ширине)

	type	title	duration	age_group	release_year	genres
4854	Movie	Pieces of a Woman	1.5 - 2.5 hours	Adults	2020	[Dramas]
5127	Movie	Recall	0 - 1.5 hours	Adults	2018	[Dramas]
455	Movie	American Beauty	1.5 - 2.5 hours	Adults	1999	[Dramas]
429	Movie	Alone in Berlin	1.5 - 2.5 hours	Adults	2016	[Dramas]
1836	Movie	Doubt	1.5 - 2.5 hours	Teens	2008	[Dramas, Faith & Spirituality]

Если учесть, что я ни как не сортировал по возрасту и продолжительности, то можно сказать, что вышло не плохо. Если почитать описания, они в целом не похожи, однако идея одна, и направлена она в драму. Всёёёёёёёёёёёёёёёё:)))))))))))))))))))

7 Послесловие

Я на самом деле очень рад, что вы до сюда дошли. Я потратил очень много времени, делая эту работу. Я делал её на своей работе, после неё, и в выходные. Вообщем я уделял всё свободное время данной работе, чтобы вам понравилось. Надеюсь, так оно и есть.

