МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра информационных технологий

Допустить к защите
Заведующий кафедрой
канд. физ.-мат. наук, доц.
_____ В.В. Подколзин
(подпись)
_____ 2023 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

ИСПОЛЬЗОВАНИЕ МЕТОДА КОЛЛАБОРАТИВНОЙ ФИЛЬТРАЦИИ ПРИ РАЗРАБОТКЕ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ

Работу выполнил _____ Е.А. Иванов
(подпись)

Направление подготовки 01.03.02 Прикладная математика и информатика
(код, наименование)
Направленность (профиль) Программирование и информационные технологии

Научный руководитель
канд. физ.-мат. наук, доц. _____ Е.П. Лукащик
(подпись)

Нормоконтролер
канд. пед. наук, доц. _____ А.В. Харченко
(подпись)

Краснодар
2023

РЕФЕРАТ

РЕКОМЕНДАТЕЛЬНЫЕ СИСТЕМЫ, КОЛЛАБОРАТИВНАЯ ФИЛЬТРАЦИЯ, ДАТАСЕТ, МАТРИЦА ПРЕДПОЧТЕНИЙ, ОБУЧАЮЩАЯ ВЫБОРКА.

В выпускной квалификационной работе 51 стр, 24 рисунка.

Цель выпускной квалификационной работы – разработать рекомендательный сервис фильмов на основе метода коллаборативной фильтрации.

В теоретической части работы изучены основные принципы построения рекомендательных систем, их базовые алгоритмы: контентная фильтрация, коллаборативная фильтрация, фильтрация, основанная на знаниях. Изучены различные метрики определения схожести объектов по множеству характеристик.

В оригинальной части работы на языке программирования Python разработан рекомендательный сервис, который на основе свободно распространяемых датасетов с фильмами позволяет для некоторого пользователя построить рекомендацию из 10 фильмов по его любимому жанру. Определение наиболее предпочтительного жанра пользователя также выявляется сервисом.

Разработанный сервис может быть применен в системе работы онлайн кинотеатров.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время для обеспечения эффективных продаж различных товаров и услуг необходимо не только отслеживать изменение цены, вводить различные акции, но и предлагать покупателю сопровождающие товары, товары, которые могут заинтересовать клиента.

Но просто предложение похожих товаров и услуг может вызвать обратную реакцию клиента, как навязчивую рекламу товара. Поэтому при формировании списка предлагаемых товаров следует грамотно оценивать настроения пользователя, его интересы. Для этого разработаны алгоритмы рекомендательных систем. Эти алгоритмы, базируясь на больших объемах информации о других пользователях, пытаются составить портрет текущего клиента и определить на каких, уже имеющихся пользователей он похож.

Использование отзывов и рекомендаций по товарам и услугам способствует более качественному построению рекомендаций. Алгоритмы контентной фильтрации, коллаборативной фильтрации хорошо зарекомендовали себя в решении задачи построения рекомендаций.

Первая глава выпускной квалификационной работы содержит перечень основных задач рекомендательной системы.

Вторая глава посвящена описанию методов, используемых при реализации рекомендательных систем.

В третье главе приведены примеры рекомендательных систем, определены их преимущества и недостатки.

В четвертой главе рассмотрены меры схожести, которые можно использовать в рекомендательных системах, для определения близости характеристик объектов.

Пятая глава посвящена этапам проектирования собственного рекомендательного сервиса фильмов.

В шестой главе приведены фрагменты реализации рекомендательного сервиса и даны примеры его работы.

1 Основные задачи рекомендательной системы

Под рекомендательной системой понимают систему фильтрации информации, позволяющую основываясь на предпочтениях пользователя предложить ему тот или иной объект [5]. Рекомендательные системы чаще всего используются в ситуациях, когда среди большого ассортимента товаров необходимо выбрать один.

Основная задача рекомендательной системы — это формирование предложения для пользователя о товаре, который должен вызвать у него наибольший интерес. Клиент получает информационную услугу, доходы сервиса могут складываться из рекламных услуг, процента с продаваемых товаров. Рекомендательная система может быть ядром сервиса, например, при подборе экскурсионных предложений и отелей (сервис TripAdisor) или быть дополнительным сервисом, например, в интернет-магазине.

Перечислим основные характеристики рекомендательной системы [6].

1) предмет рекомендации. указывается тип рекомендуемого товара в широком смысле. это может быть реальный товар для интернет-магазинов или маркетплейсов, видео и фотоизображения, музыкальные треки, фильмы, новости;

2) цель рекомендации. указывается цель рекомендации. для рекомендуемых товаров это покупка, для новостей и изображений это получение информации, для музыкальных треков и фильмов это проведение досуга;

3) содержание рекомендации. определяется поведенческое состояние пользователя в момент общения к рекомендательной системе: слушает музыку, смотри фильмы, подбирает товар, просматривает новости;

4) источник рекомендации. от кого исходит рекомендация, на основании чего она формируется. это может быть мнение большинства, мнение экспертов;

5) степень персонализации. рекомендации могут быть персональные и не персональные. при формировании не персональных рекомендаций пользователь получает рекомендацию. не учитывающую личные предпочтения. здесь основой рекомендации может выступать регион или временной отрезок. персонализированные рекомендации учитывают историю пользователя, его предыдущие сессии, схожесть уже выбранных товаров;

6) прозрачность. рекомендации должны обладать высокой степенью достоверности. необходимо бороться с накрученными и недобросовестными отзывами;

7) вид сформированной рекомендации. формат рекомендации может быть в виде списка, набора фотографий, перечня товаров;

8) алгоритмы. основой рекомендательной системы является алгоритм фильтрации информации. существует ряд популярных подходов: модель, основанная на описании товаров, коллаборативная фильтрация, методы матричного разложения.

Рассмотрим перечисленные алгоритмы подробнее.

2 Алгоритмы фильтрации информации

2.1 Метод матрицы предпочтений

Одним из методов построения рекомендаций является матрица предпочтений. В матрице по горизонтали расположены клиенты, по вертикали – товары, предлагаемые к рекомендации. На пересечении строк и столбцов находятся числовые значения, являющиеся оценками, отражающими заинтересованность пользователя в товаре из столбца. При формировании оценки используется некоторая шкала, на рисунке 1 используется шкала [1, 5].

	Товар 1	Товар 2	Товар 3	Товар 4	Товар 5
Клиент 1		3		5	
Клиент 2	1		1	1	
Клиент 3	2			3	2
Клиент 4		4			5
Клиент 5	5		2	3	4

Рисунок 1 – Матрица предпочтений

В реальной ситуации клиенты не будут оценивать товары всего перечня, поэтому рекомендательная система должна уметь обобщать имеющиеся данные и пролонгировать оценки на пустующие ячейки. Для нормальной работы рекомендательной системы вся матрица предпочтений должна быть заполнена.

Кроме того, при построении рекомендаций упор может быть сделан на новые товары или наоборот на имеющиеся ранее. Выделяют повторяемые товары в рекомендациях, сюда относятся товары, которые покупаются с некоторой периодичностью. Здесь часто клиенты отдают предпочтение любимым маркам. Другая группа рекомендуемых товаров, это товары неповторяемые. Сюда относятся товары, покупаемые единожды. Например, книги.

Необходимо учитывать индивидуальные предпочтения клиентов, кто-то верен любимым фирмам-производителям, а кто-то любит приобретать все новое.

При заполнении матрицы предпочтений оценки пользователей можно получить двумя способами. Пользователь явно проставляет оценку купленному товару, указывает его рейтинг среди множества других. Либо мы получаем неявную оценку, товар интересен, если клиент его приобрел.

2.2 Метод контентной фильтрации

Основой контентной фильтрации является содержание, т.е. информация, которая определяет предпочтения пользователя [3]. Рекомендательные системы, основанные на методе контентной фильтрации, подбирают максимально близкий профиль к профилю текущего пользователя. Например, если пользователь интересуется программными продуктами и часто рассматривает какую-либо информацию об 1:С, то метод контентной фильтрации учитывает связь запросов о программном обеспечении и 1:С. В дальнейшем такая информация может быть рекомендацией для других пользователей, интересующихся программным обеспечением.

Этот же метод хорошо работает в интернет-магазинах и на маркетплейсах, если пользователи при выборе товара покупают часто второй

сопутствующий товар, то эта информация сохраняется и потом второй товар может выдаваться как рекомендация при покупке первого.

Основой метода контентной фильтрации является набор характеристик пользователя и перечень его предпочтений – например, покупаемых товаров. Перечень товаров или услуг представляет собой описание характеристик этих объектов. В итоге мы получаем кортеж характеристик, где для каждой характеристики указывается ее вес.

Схематично алгоритм метода контентной фильтрации можно представить следующими шагами (рисунок 2):

1) сбор характеристик объектов – формирование кортежа характеристик – анализ контента;

2) построение профиля пользователя, содержащего наборы предпочтений пользователя – анализ профиля:

3) соотнесение профиля текущего пользователя с перечнем характеристик объектов – построение рекомендаций.

Рисунок 2 – Схема алгоритма контентной фильтрации

Выделим основные преимущества алгоритма контентной фильтрации [3]. Прежде всего, этот метод дает хорошие результаты даже на небольшом наборе пользователей, на основе которых строятся рекомендации. И второе преимущество, как только появляется набор рекомендаций по новому объекту, его сразу можно добавлять в стек рекомендуемых. Тем самым наборы рекомендуемых объектов расширяются с появлением рекомендаций по ним. Такая ситуация удобна в маркетплейсах, как только появится отзыв на товар, его можно учитывать для рекомендации товара другим клиентам.

Однако у метода контентной фильтрации существуют и недостатки.

Если параметры выбираемых объектов у текущего пользователя не совпадают с имеющимися рекомендациями, то достаточно сложно определить направление приоритетов пользователя. Для того чтобы можно было конструировать рекомендации требуется соблюдение одного условия: набор характеристик текущего пользователя и набор характеристик рекомендаций должны совпадать. Такое требование позволяет сравнить наборы. Самым существенным недостатком метода является невозможность включать в рекомендуемые товары те товары, на которые не существует отзывов. Возникает проблема так называемого холодного старта.

Использование метода контентной фильтрации учитывает не только характеристики похожих пользователей, но и историю текущего пользователя. Так при рекомендациях покупок можно использовать информацию о предыдущих покупках. Описание купленных товаров учитывается при описании предпочтений пользователя. Чем больше купленные товары и их характеристик описывают предпочтения пользователя, тем точнее будут дальнейшие рекомендации.

При описании рекомендуемых объектов удобно использовать числовые характеристики, например размер, диапазон стоимости товара. Некоторые характеристики можно привести к числовым значениям, например, цвет. Но существуют характеристики, у которых отсутствует числовая интерпретация, например, описание фильма.

Для успешной реализации рекомендательной системы для каждого объекта в системе должно быть описание. Составляется числовое или текстовое описание. Неструктурированное описание оформляется в виде вектора, каждые элемент которого соответствует некоторой характеристике объекта.

2.3 Метод коллаборативной фильтрации

Этот метод учитывает данные о стратегии поведения пользователей. Различают два основных направления метода [2].

Построение рекомендаций на основе знаний о поведении пользователей, у которых такие же интересы, что и у текущего пользователя. Здесь основой выступает похожесть пользователей. Подход получил название user-based.

Второй подход учитывает предыдущее поведение пользователя. В этом случае рекомендуются объекты, которыми ранее интересовался текущий пользователь. Такой подход называется item-based.

Первое направление основано на идее о том, что если у нескольких пользователей есть общие интересы, то и в будущем они будут интересоваться одним и тем же. В этом случае система должна выбрать из общего множество пользователей пользователей-соседей. Для того, чтобы оценить уровень похожести, обычно вводят систему оценок для товаров. Схожесть пользователей определяется по близким значениям оценок, установленных при оценивании товаров. Зная систему оценивания предыдущих товаров и учитывая историю покупок текущего пользователя можно спрогнозировать как, он отнесется к новому товару. Если новым товаром заинтересовались соседи, то и текущий пользователь, скорее всего, проявит интерес к этому товару, а, следовательно, этот новый товар можно включить в рекомендации.

Аналогично можно привести пример по подпискам в социальных сетях. Если выделить близкие по своим характеристикам множество пользователей, то подписки соседей можно рекомендовать текущему пользователю. В этом случае для соседей можно построить матрицу предпочтений (рисунок 3), в ней столбцы — это подписки пользователей, а в строках указываются пользователи. Если в матрице в ячейке находится 1, то пользователь подписан на подписку, 0 – не подписан.

	Подписка 1	Подписка 2	Подписка 3	Подписка 4
Пользователь 1	0	1	1	1
Пользователь 2	0	1	1	0
Пользователь 3	1	0	0	0
Пользователь 4	1	1	0	1
Пользователь 5	0	0	1	0

Рисунок 3 —Матрица предпочтений для user-based подхода

Пользователи 1, 2 и 4 по своим интересам могут составлять одну группу, у них достаточно много общих подписок. Пользователь 2 на основании этих данных может получить рекомендацию на подписку 4, так как эта подписка фигурирует в их группе схожести.

Подход item-based рассматривает схожесть объектов, а не пользователей [2]. Для каждого оцененного объекта, формируется набор объектов с такими же оценками и близкими характеристиками. Таким образом формируется множество соседей-объектов.

Объекты, отмеченные текущим пользователем, исключаются из множества объектов-соседей. А оставшиеся объекты могут выступать как рекомендация. Например, если большинство клиентов выбирает два товара, то всем клиентам, которые выбрали один товар из пары можно рекомендовать оставшийся товар из пары.

Если рассматривать подписки в социальной сети, то можно рассмотреть несколько подписок, на которые подписано несколько одних и тех же пользователей. Такие пользователи объединяются в группу. Затем

подписки, выходящие из общего набора, предлагаются остальным пользователям группы. Пользователи в группе на основе множества пересекающихся подписок имеют схожие интересы и поэтому дополнительные индивидуальные подписки с большой вероятностью понравятся членам группы.

Таким образом, оба подхода используют один и тот же прием поиска схожести, но группировка выполняется либо по пользователям, либо по объектам. Матрица предпочтений тоже может быть одна и та же (рисунок 4).

	Подписка 1	Подписка 2	Подписка 3	Подписка 4
Пользователь 1	0	1	1	1
Пользователь 2	0	1	1	0
Пользователь 3	1	0	0	0
Пользователь 4	1	1	0	1
Пользователь 5	0	0	1	0

Рисунок 4 — Матрица предпочтений для item-based подхода

И для item-based, и для user-based подходов матрица потребностей используется одна и та же, но группирование происходит по разным элементам (рисунок 4).

Подписки 2, 3 и 4 следует объединить в одну группу схожести, так как они находятся в перечне подписок этих пользователей. А затем Пользователю 2 можно порекомендовать подписку 4. Эта подписка находится в общей группе схожести.

Сформулируем алгоритм коллаборативной фильтрации (рисунок 5).

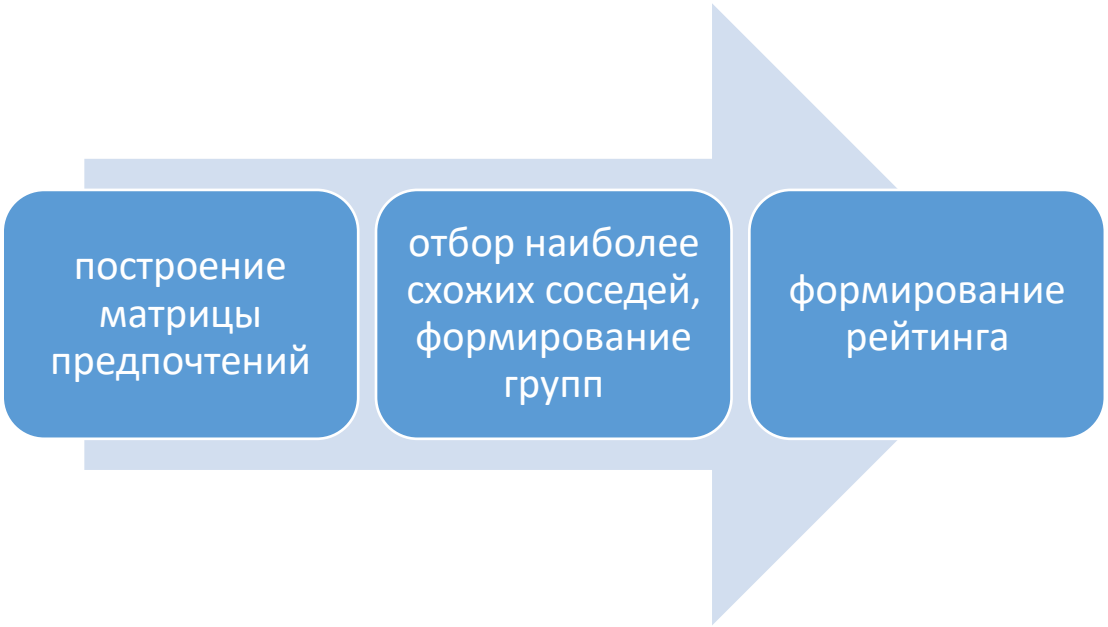

Рисунок 5 – Алгоритм коллаборативной фильтрации

На первом этапе с помощью различных метрик схожести строится матрица предпочтений [7]. В зависимости от подхода выбирается схожесть пользователей (user-based) или схожесть объектов (item-based). Матрица предпочтений содержит строки – объекты, столбцы – пользователи. В ячейке матрицы находится значение схожести.

На следующем этапе по матрице выбираются соседи. Определяется множество наиболее схожих пользователей или наиболее схожих объектов. Они объединяются в группы схожести, группы соседей. Если данных мало, то множество соседей можно построить на основе ненулевого значения схожести.

На завершающем этапе на основе интересов пользователей и объектов формируется их рейтинг. Наиболее часто встречаемые выдаются пользователю как рекомендации.

Определим преимущества и недостатки двух подходов.

К достоинствам подхода, ориентированного на пользователей, является разнообразие в рекомендациях, этот подход легко программируется. В этом случае нет необходимости иметь подробную информацию о рекомендуемых объектах.

Недостатками подхода является, прежде всего, не желание пользователей выставлять оценки объектам. Только малый процент покупателей тратит время на оценивание товара, а тем более на развернутый отзыв. Другой проблемой является недостаточность данных о новых пользователях. При появлении нового пользователя не скоро соберется информация о нем, а значит сравнить его с другими пользователями затруднительно.

При большом количестве пользователей затратно отыскивать группы схожести. На это необходимо выделять много ресурсов, например, хранение и поиск в базе данных. Таким образом, в больших системах это затратно. Кроме того, необходимо сравнивать пользователей попарно и хранить все их ненулевые схожести.

Подход, основанный на объектах более выгоден, так как сходство в товарах меняется гораздо реже, чем сходство в пользователях. Все товары и услуги обычно и так хранятся в базе данных, и поиск по ней не требует дополнительных затрат.

Рекомендации на основе объектов несколько точнее, чем по оценкам пользователей. Пользователи эмоциональны и не всегда дают адекватные реакции.

Недостатком подхода является ситуация рекомендации очевидного объекта, что не способствует эффективным продажам. Кроме того, добавление новых объектов связано с недостатком информации и гарантирует проблему холодного старта.

Выделим общие проблемы при разработке рекомендательных систем.

Основная проблема — это проблема холодного старта, т.е. проблема недостаточности данных. Малое количество данных не позволяет строить корректные рекомендации.

Если исходных данных мало, то группы схожести могут не соответствовать действительности, и тогда рекомендации не будут отвечать интересам пользователей.

При малом числе оценок товара можно вычислять среднее арифметическое имеющихся оценок, но более эффективно вычислить сглаженное среднее. В этом случае за среднюю оценку берется не математическая середина, а та оценка, к которой расположено ближе большинство оценок.

Можно вычислять доверительные интервалы. В этом случае вычисляется рейтинг интервалов достоверности. Рейтинг рассматривается как нижняя граница интервала.

В некоторых предметных областях важна актуальность рекомендации. Например, при рекомендации новостей. В этом случае учитывать историю пользователя не эффективно. Создается топ новостей, который постоянно актуализируется. Новости можно присвоить рейтинг, например по формуле [8]:

$$Rank = \frac{(U-D-1)^{0.8} * P}{T^{1.8}} \quad (1)$$

Где

U это число положительных оценок,

D – число отрицательных оценок,

P – дополнительная корректировка.

Можно принимать в расчет время появления новости и использовать формулу определения рейтинга:

$$Rank = log_{10}(max(1, U - D)) - \frac{|U-D|T}{const} \quad (2)$$

Формулы могут видоизменяться и адаптироваться под конкретную предметную область.

2.4 Фильтрация, основанная на знаниях

В некоторых ситуациях и предметных областях учитывать историю пользователя невозможно. Например, на некоторых сайтах пользователи совершают покупки редко, например, ювелирные изделия или крупная бытовая техника. В этом случае удобно использовать метод фильтрации данных на основе знаний о рассматриваемой предметной области [6].

Необходимо иметь подробную информацию о рекомендуемом объекте. Например, если это ноутбук, то рассматривается множество характеристик: частота процессора, объем памяти, объем жесткого диска, наличие вай-фай, диагональ экрана, вес, количество разъемов и т.д. У пользователя перед выбором рекомендательная система запрашивает набор предпочтительных характеристик и тем самым определяет интересы пользователя.

Схематично алгоритм метода фильтрации, основанного на знаниях, включает следующие этапы (рисунок 6).

Сначала с помощью интерактивной формы собирается информация о начальных предпочтениях пользователя.

На следующем этапе полученные данные сравниваются с характеристиками имеющихся объектов и формируется набор наиболее близких по заявленным характеристикам объектов.

На заключительном этапе пользователь уточняет свои запросы и список рекомендаций обновляется.

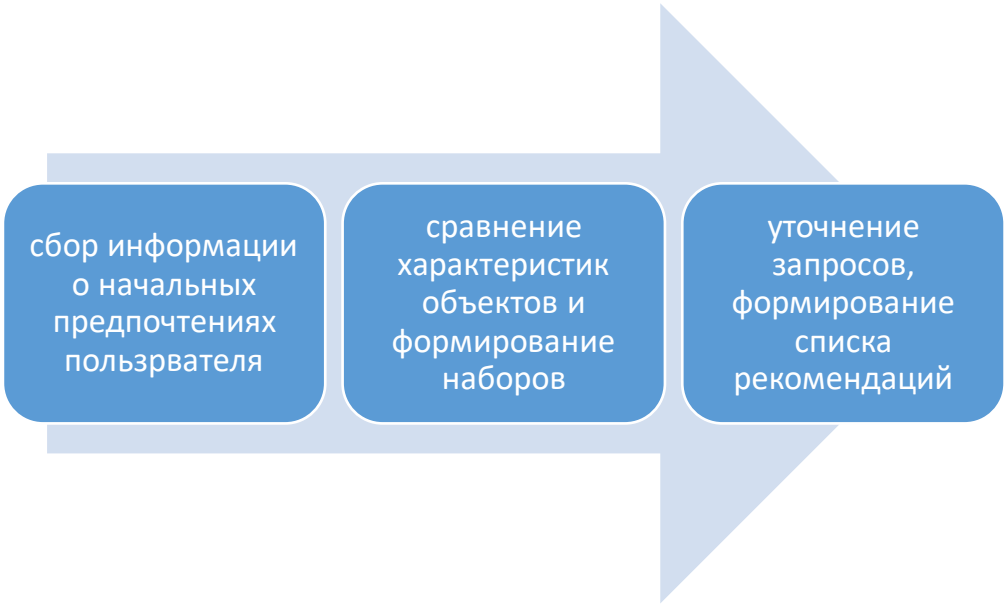

Рисунок 6 – Алгоритм метода фильтрации, основанного на знаниях

К преимуществам подхода фильтрации можно отнести высокую точность рекомендаций. Это связано с тем, что пользователь практически строит фильтр на имеющиеся объекты в соответствии со своими предпочтениями. В этом методе не требуется знание истории покупок и другой предварительной информации о пользователе, не строится рейтинг объектов.

Недостатком метода можно указать сложность реализации системы. Для того чтобы такой подход был действенным, необходимо собрать существенное количество характеристик объектов, учитывать множество параметров объектов.

Если пользовательский выбор не соответствует ни одному объекту в системе, то необходимо определить алгоритм, решающий проблему поиска максимально похожего объекта.

Кроме того, пользователь должен изъявить желание заполнить интерактивную форму и указать все свои предпочтения, выбрав все характеристики объекта.

3 Обзор некоторых рекомендательных систем

Рассмотренные подходы к организации построения рекомендаций имеют недостатки. Чаще всего на практике для преодоления недостатков используются комбинированные методы, которые отражаются в реализации гибридных рекомендательных систем.

Во многих рекомендательных сервисах учитывается как полная информация о товарах, так и история покупок пользователя. В этом случае объединяют метод контентной фильтрации и метод коллаборативной фильтрации. Такое объединение позволяет сделать рекомендации более точными, более адаптивными. В некоторых случаях такое объединение помогает при малом количестве информации, при холодном старте.

Гибридную стратегию реализуют обычно крупные компании, это связано с крупными вложениями в реализацию.

Примером такой гибридной рекомендательной системы является рекомендательная система Cinematch, разработанная Netflix. В этой системе пользователь проходит мини-опрос, указывая свои любимые фильмы в различных категориях. Тем самым собирается некоторая информация о профиле пользователя. Далее для конкретного пользователя формируется лента рекомендаций. Корректировка профиля пользователя осуществляется его реакцией на предложенные фильмы. Пользователь отмечает какие фильмы ему понравились ранее или не понравились, какие бы он хотел посмотреть. Вся эта информация с одной стороны уточняет информацию о пользователе, позволяет определить его к некоторой группе схожих профилей. С другой стороны, накапливается информация об объектах, о фильмах, какие нравятся большинству или не нравятся.

Рисунок 7 – Сервис Cinematch

Сервис Cinematch учитывает достаточно много информации, например, отмечается геолокация пользователя, пол, возраст, время выбора фильма, частота общения с сервисом, время просмотра и длительность. Вся эта информация формирует полноценный профиль пользователя.

Компания Netflix не ограничивается стандартными рецензиями на фильмы. Для каждого видео подбираются ключевые слова, по которым скорее всего пользователь будет выбирать в строке поиска фильм, не зная его названия. Таким образом, рекомендация появляется и по строке запроса содержания фильма.

Сервис учитывает и внешние факторы, например, праздники, день недели, религиозные привязанности.

Достаточно мощным гибридным рекомендательным сервисом является сервис Библа (рисунок 8).

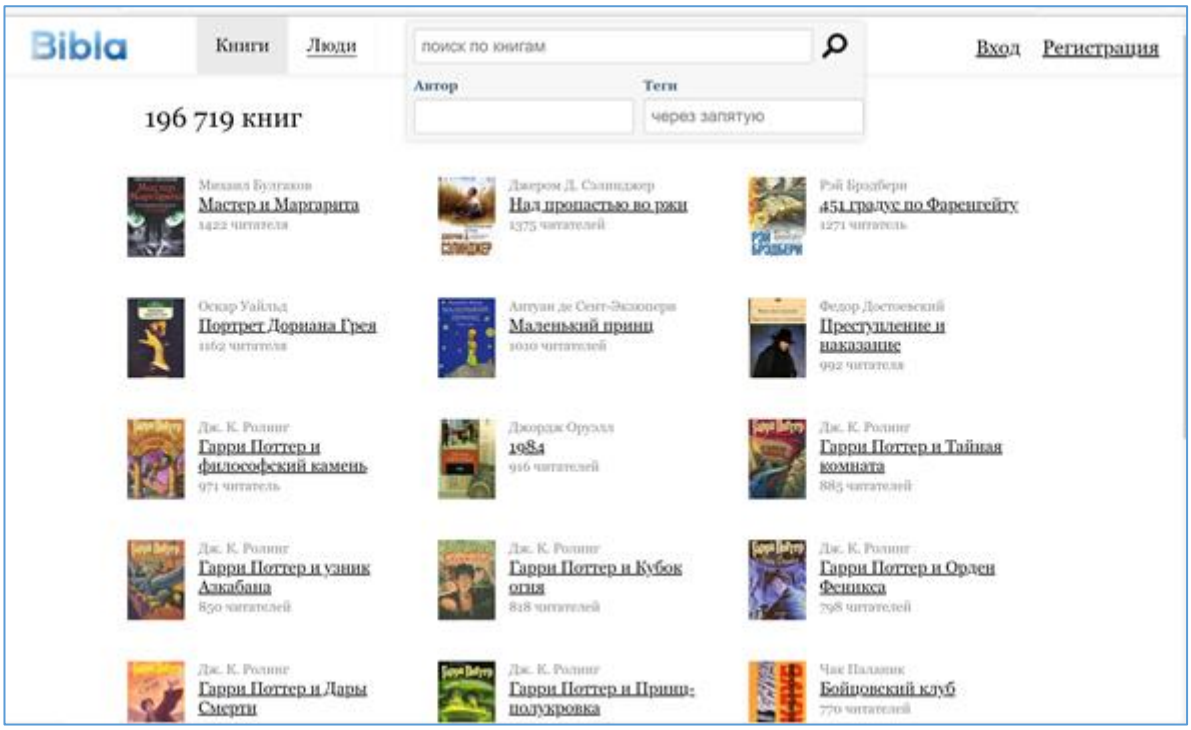

Рисунок 8 – Сервис Библа

Сервис рекомендует книги. Здесь существует учет истории пользователя («Список прочитанных книг»), причем пользователь имеет возможность сделать заметку, чем понравилась или не понравилась книга. Последнее уже относится к формированию профиля объекта. В сервисе формируется список пожеланий («Хочу почитать»), содержащий списки книг, которые планируется прочитать, и также указываются заметки, где отмечено, чем заинтересовала книга.

Для удобства ориентации среди множества книг существует пункт «У меня есть», позволяющий помещать в список книги домашней библиотеки. Эти книги также учитываются как книги с положительным профилем.

Сервис позволяет создавать тематические подборки книг с соответствующими заметками. Такая функция является готовой рекомендацией от отдельного пользователя.

Таким образом, гибридный сервис интегрирует в себе построение профилей, как пользователей, так и профилей книг.

Другой сервис книжных рекомендаций Two-books.net основан на идее построения списка лучших книг мира. Каждому пользователю можно

выбрать всего две лучших для него книг. Далее для каждой книги устанавливается рейтинг, чем больше пользователей выбрали книгу в качестве любимой, тем выше у нее рейтинг. Кроме того, сервис сохраняет рецензии и отзывы на книги, пользователи могут составлять свои списки рекомендаций.

Выделим на примере Netflix и Библа достоинства и недостатки гибридных систем.

К преимуществам можно отнести повышенную точность рекомендаций за счет комбинирования методов. Кроме того, исключаются ограничения разных методов, один метод спасает другой.

Основным недостатком гибридной рекомендательной системы является ее ресурсоемкость и высокая стоимость.

4 Основные меры определения схожести в рекомендательных системах

Методы фильтрации рекомендаций основаны на методах определения схожести характеристик объектов и профилей пользователей. Рассмотрим основные из них.

Одним из простых методов определения схожести является Евклидово расстояние. Этот метод заложен в основу построения матриц схожести в алгоритмах коллаборативной фильтрации [7].

Значения признаков объектов распределяются по координатным осям. Точки, внутри системы координат соответствуют пользователям, строящим оценки. Точки располагаются в координатной плоскости. Чем ближе точки, тем ближе их профили.

Евклидово расстояние определяется формуле 3 [4]:

$$d(x, y) = \sqrt{\sum_i^n (x_i - y_i)^2} \quad (3)$$

где x_i и y_i — оценка объекта i пользователями x и y, n — количество оцениваемых объектов.

Чем меньше расстояние между точками, тем более схожи пользователи по своим предпочтениям. Для удобства можно изменить формулу, рассматривая обратную величину. Добавленная единица позволяет избежать деления на ноль. Модифицированная формула имеет вид:

$$\frac{1}{1+\sqrt{\sum_i^n (x_i - y_i)^2}} \quad (4)$$

Модифицированная формула дает значения от 0 до 1. Где 1 обозначает полное сходство.

Рассмотрим применение Евклидова расстояния на примере. Клиент 1 дал оценку ноутбуку Dell на 4,5, а Asus — на 1,0. Клиент 2 дал оценки 4,0 и 2,0 соответственно.

Значение схожести для клиентов 1 и 2 по Евклидову расстоянию дают:

$$\frac{1}{1+\sqrt{(4,5-4)^2+(1-2)^2}} = 0,47 \quad (5)$$

Следующая мера схожести – коэффициент корреляции Пирсона [5].

Корреляция Пирсона более сложная и позволяет определить линейную зависимость между двумя величинами и измеряется от -1 до +1, где значение 1 указывает на то, что объекты данных идеально коррелированы, а -1 — не коррелированы.

При построении коэффициента Пирсона пользователи системы представляют собой координатные оси, а объекты — это точки в координатной плоскости. Чем точнее точки плоскости образуют прямую, тем точнее будет рекомендация. Другими словами, объекты, расположенные ближе всего к прямой можно рекомендовать (рисунок 9).

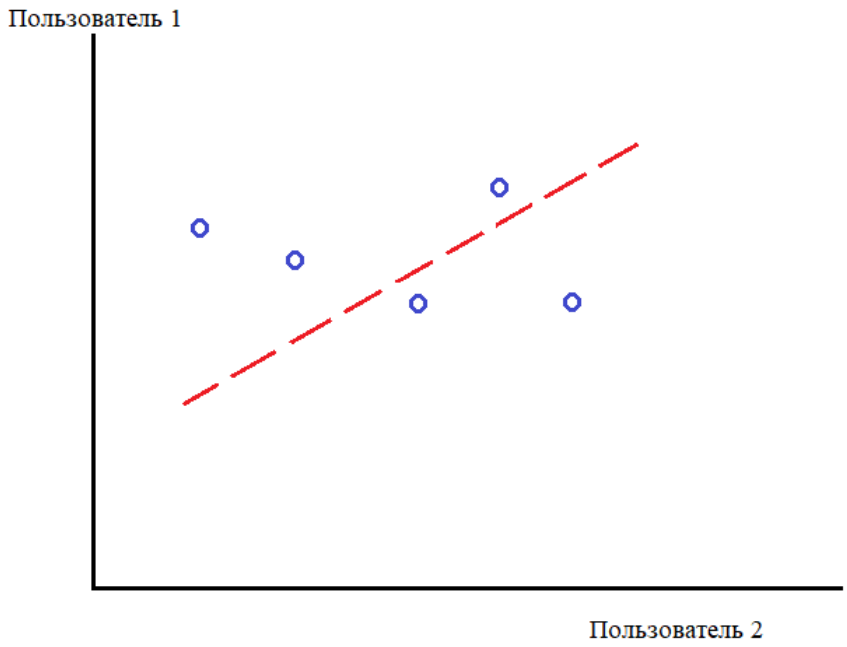

Рисунок 9 - Сравнение пользователей с помощью корреляции Пирсона

Эта прямая, к которой приближено большинство точек, называется прямой наилучшего приближения. Эта линия образует диагональ или стремится к ней, если пользователи выставляют максимально близкие (одинаковые) оценки. Т.е. корреляция равна 1.

Коэффициент корреляции Пирсона удобен тем, что кроме собственно сравнения оценок и вычисления их близости, как Евклидово расстояние, выполняет нормализацию проставленных оценок.

Формула коэффициента корреляции Пирсона следующая:

$$r_{xy} = \frac{\sum_i^n (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_i^n (x_i - \overline{x})^2 \sum_i^n (y_i - \overline{y})^2}} \quad (6)$$

где $\overline{x}$ и $\overline{y}$ — средние арифметические всех оценок пользователей x и y.

Корреляция Пирсона накладывает ограничения на исходные данные – число объектов должно быть строго больше двух.

Еще одним эффективным механизмом определения схожести оценок является косинусная мера [4].

В этом случае признаки объектов следует трансформировать в вектора и затем вычислить расстояние между этими векторами, так называемый косинусный вектор.

Формула косинусной меры следующая.

$$K = \frac{\sum_i^n x_i \times y_i}{\sqrt{\sum_i^n (x_i)^2} \times \sqrt{\sum_i^n (y_i)^2}} \quad (7)$$

Вычисленное значение коэффициента также как и коэффициент Пирсона, нормализовано и находится в пределах от 0 до 1. Абсолютное сходство обозначается значением 1.

Наиболее популярным в силу своей простоты и эффективности при решении задачи определения групп схожести является метод k-средних.

Этот метод рассматривает множество многомерных векторов, содержащих признаки рассматриваемых объектов. Вектора разбиваются на k заданных кластеров. Сначала вектора располагаются в кластерах случайным образом, обычно по мере следования. Дальше на каждом шаге алгоритма центр масс для каждого кластера обновляется, пересчитывается и состав кластеров перегруппировывается. Вектора распределяются в тот кластер, к центру масс которого они относятся ближе всего. Как только состав кластеров перестает изменяться, алгоритм заканчивает свою работу (рисунок 10).

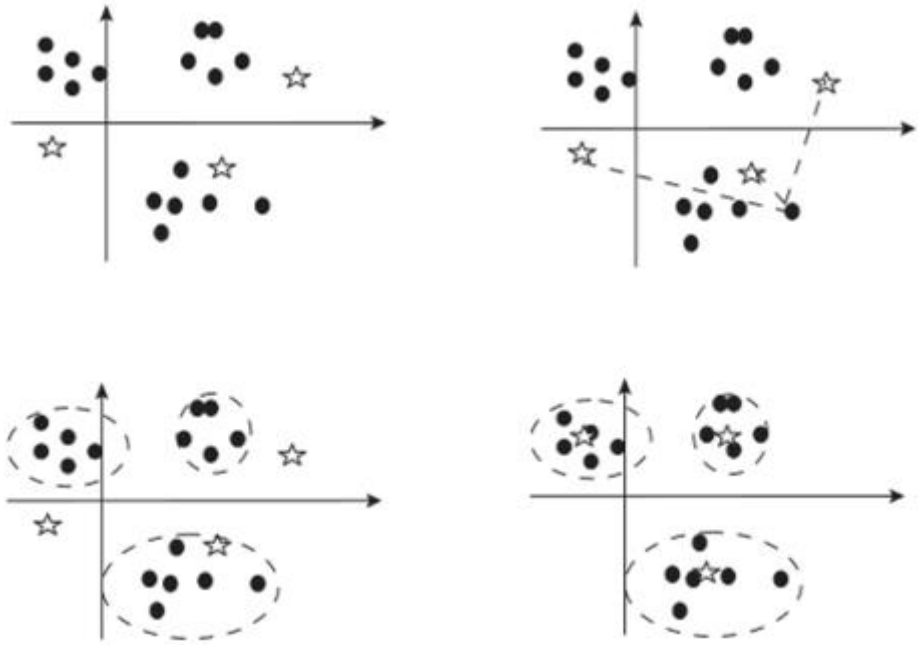

Рисунок 10 – Кластеризация методом k-средних

Алгоритм достаточно прост в реализации, однако у него есть ограничения.

Количество кластеров задается изначально. В связи с этим, чтобы получить наиболее эффективное разбиение, необходимо провести несколько экспериментов, задавая разное число кластеров. Выбор центра масс каждого

кластера в начале алгоритма имеет значение. При неудачном выборе, алгоритм будет работать существено долго. Алгоритм может выдать не верное решение, в ситуации, когда один объект одинаково близок в двум центрам кластеров.

Эту проблему решает другой алгоритм, алгоритм c-средних. В алгоритме высчитывается не просто расстояние от объекта до центра кластера, а определяется вероятность принадлежности этого объекта кластеру. Тогда вектор объекта может принадлежать к кластеру 1 с вероятностью 70%, а к кластеру 2 принадлежать с вероятностью 30%.

Алгоритм c-средних уже более сложен в реализации и требует больше времени при выполнении.

Алгоритмы k-средних и c-средних являются классическими алгоритмами кластеризации. Но для рекомендательных систем разработали узкоспециализированные алгоритмы, например, алгоритм Grouplens [8].

Этот алгоритм разработан в 1992 году. Он на первом этапе выполнял поиск пользователей со схожими предпочтениями. А на втором этапе выполнял прогнозирование оценок текущего пользователя на основе имеющихся оценок пользователей соседей.

Базовая формула, использованная в алгоритме для построения прогноза, имеет вид.

$$\widehat{r_{i,a}} = \overline{r_i} + \frac{\sum_j^n (r_{j,a} - \overline{r_j}) w_{ij}}{\sum_j^n |w_{ij}|} \quad (8)$$

где $\widehat{r_{i,a}}$ — прогнозируемая оценка объекту a, которая будет выставлена пользователем i;

$\overline{r_i}$ и $\overline{r_i}$ — средние арифметические всех оценок пользователей i и j;

$r_{j,a}$ — оценка пользователя j объекту a;

w_{ij} — коэффициент сходства пользователей i и j.

Мера схожести k ближайших соседей (k-NN) — это один из наиболее распространенных алгоритмов машинного обучения.

Ключевая идея метода k-NN заключается в том, что объекты, близкие в пространстве признаков, склонны к принятию схожих решений. Для классификации нового объекта алгоритм k-NN находит k ближайших к нему объектов из обучающей выборки и определяет класс этого объекта на основе классов ближайших соседей.

Мера схожести в методе k-NN обычно выражается в виде расстояния между объектами в пространстве признаков. Наиболее распространенными метриками расстояния являются Евклидово расстояние и манхэттенское расстояние, но в зависимости от задачи и свойств данных могут использоваться и другие метрики.

Оптимальное значение k в методе k-NN зависит от конкретной задачи и может быть выбрано эмпирически с помощью кросс-валидации или оценки ошибки на тестовой выборке.

Метод k-NN может быть применен в широком диапазоне задач машинного обучения, включая распознавание образов, классификацию текстов, анализ данных и многие другие.

Преимущества метода k-NN заключаются в его простоте и относительной эффективности для небольших и средних выборок данных. Кроме того, метод k-NN легко адаптируется для работы с нечисловыми данными, например, с текстами.

Недостатками метода k-NN являются его относительная вычислительная сложность при больших объемах данных и сложности в выборе оптимального значения k. Кроме того, метод k-NN не имеет возможности учиться на основе скрытых зависимостей в данных, поэтому может оказаться неэффективным в некоторых задачах.

Несмотря на некоторые ограничения и недостатки, метод k-NN остается популярным и широко используемым алгоритмом машинного

обучения благодаря своей простоте, универсальности и эффективности на небольших и средних объемах данных.

Алгоритм работы меры схожести k ближайших соседей следующий:

− Определение значения k, количество ближайших соседей, которые будут использоваться для оценки схожести объектов.

− Вычисление расстояний между объектом, для которого необходимо оценить схожесть, и всеми другими объектами в выборке.

− Определение k ближайших объектов на основе расстояний.

− Вычисление меры схожести между двумя объектами. Возможны различные методы вычисления меры схожести, например, Евклидово расстояние.

− Определение уровня схожести между двумя объектами на основе меры схожести, вычисленной на предыдущем шаге.

Мера схожести k ближайших соседей широко используется в задачах классификации и регрессии, а также в задачах рекомендации, например, в рекомендательных системах, чтобы определить, какие товары или услуги могут быть наиболее интересны для пользователя на основе его предыдущих действий. Однако, этот метод может страдать от проблемы переобучения, если число k слишком мало, а также он может быть неэффективным при большом числе объектов в выборке.

5 Проектирование рекомендательной системы

Будем рассматривать рекомендательную систему, построенную на основе метода коллаборативной фильтрации. Предметной областью системы является выбор фильмов.

Выделим основные задачи проектируемой системы (рисунок 11).

− Подбор датасета, содержащего фильмы, предназначенные для оценки. Это может быть фильмотека онлайн-кинотеатра. В работе будем использовать готовый датасет, имеющийся в открытом доступе.

− Формирование датасета, содержащего оценки к фильмам, проставленные пользователями. Необходимо собирать данные о реакции пользователей на просмотренные или выбранные фильмы. В онлайн-кинотеатре такой датасет должен периодически обновляться, при появлении новых оценок или добавлении новых фильмов в фильмотеку. Разрабатываемый рекомендательный сервис будет использовать уже готовый датасет с оценками пользователей.

− Определение наиболее популярного жанра для активного пользователя (для того, которому формируются рекомендации).

− Построение обучающей выборки по конкретному жанру. Эта выборка относится к тому жанру, который является ведущим у активного пользователя. Выборка содержит только оценки таких пользователей, которые выставляли оценки более 15 раз к более 50 фильмам. Таким образом, в построении выборки участвуют только пользователи с большой историей.

− Формирование рекомендаций. Выбираются 10 фильмов на основе рейтинга обучающей выборки.

Первые две задачи не реализуются полноценно в рекомендательном сервисе, данные берутся из готовых датасетов.

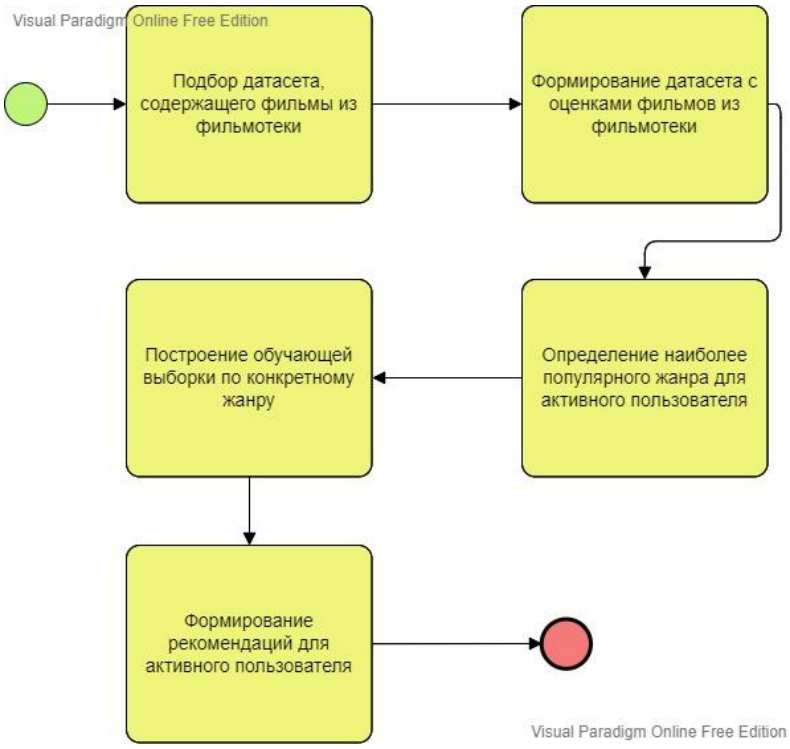

Рисунок 11 – Основные задачи бизнес-процесса

Для реализации рекомендательного сервиса будем использовать язык программирования Python. Диаграмма компонентов приведена на рисунке 12.

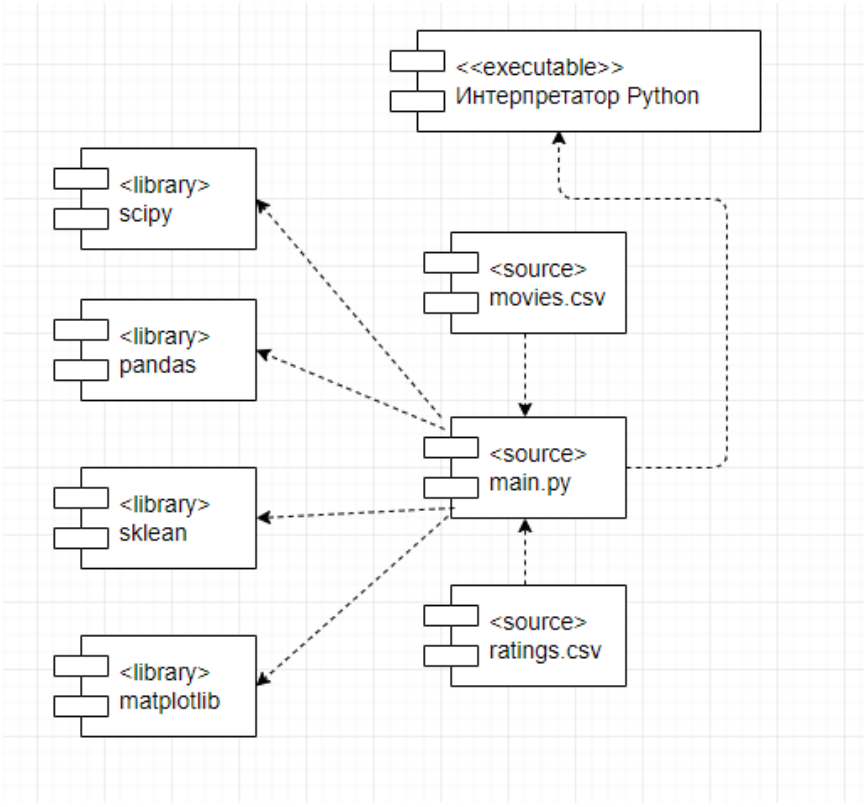

Рисунок 12 – Диаграмма компонентов

6 Разработка рекомендательной системы фильмов

При разработке рекомендательной системы будем использовать язык программирования Python.

Нам понадобится ряд библиотек.

Pandas отвечает за работу с матрицами. Эта библиотека позволяет сформировать .csv файлы и совмещать два набора данных. Пакет scipy работает на основе функций пакета Numpy и предназначен для научных вычислений. В программе эти функции используются для модификации разреженной матрицы в список параметров. Пакет scikit-learn необходим для использования алгоритма нахождения k ближайших соседей. Функции пакета позволяют выбрать метрику схожести, алгоритм поиска и количество кластеров. Пакет matplotlib используется для отрисовки графиков.

На первом этапе необходимо загрузить датасеты для построения матрицы предпочтений.

```
load_dir = "movierecommenderdataset/"
movies = pd.read_csv(load_dir + "movies.csv")
ratings = pd.read_csv(load_dir + "ratings.csv")
```

Датасет movies.csv содержит набор оцениваемых фильмов. Датасет представлен в виде таблицы. Поля таблицы: movieId (идентификатор фильма), title (название), genres (метки жанров, к которым относится фильм).

Фрагмент датасета movies.csv представлен на рисунке 13.

	A	B	C
1	movieId	title	genres
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
3	2	Jumanji (1995)	Adventure|Children|Fantasy
4	3	Grumpier Old Men (1995)	Comedy|Romance
5	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
6	5	Father of the Bride Part II (1995)	Comedy
7	6	Heat (1995)	Action|Crime|Thriller
8	7	Sabrina (1995)	Comedy|Romance
9	8	Tom and Huck (1995)	Adventure|Children
10	9	Sudden Death (1995)	Action
11	10	GoldenEye (1995)	Action|Adventure|Thriller
12	11	American President, The (1995)	Comedy|Drama|Romance
13	12	Dracula: Dead and Loving It (1995)	Comedy|Horror
14	13	Balto (1995)	Adventure|Animation|Children
15	14	Nixon (1995)	Drama
16	15	Cutthroat Island (1995)	Action|Adventure|Romance
17	16	Casino (1995)	Crime|Drama
18	17	Sense and Sensibility (1995)	Drama|Romance
19	18	Four Rooms (1995)	Comedy
20	19	Ace Ventura: When Nature Calls (1995)	Comedy
21	20	Money Train (1995)	Action|Comedy|Crime|Drama|Thriller
22	21	Get Shorty (1995)	Comedy|Crime|Thriller
23	22	Copycat (1995)	Crime|Drama|Horror|Mystery|Thriller
24	23	Assassins (1995)	Action|Crime|Thriller
25	24	Powder (1995)	Drama|Sci-Fi
26	25	Leaving Las Vegas (1995)	Drama|Romance
27	26	Othello (1995)	Drama
28	27	Now and Then (1995)	Children|Drama
29	28	Persuasion (1995)	Drama|Romance

Рисунок 13 – Фрагмент датасета movies.csv

Датасет ratings.csv содержит оценки фильмов пользователями. Поля таблицы: userId (идентификатор пользователя, выполнившего оценку), movieId (идентификатор оцененного фильма), rating (оценка пользователя), timestamp (время оценки).

Фрагмент датасета ratings.csv представлен на рисунке 14.

В работе выбран существующий датасет. Ссылка на датасет https://www.kaggle.com/datasets/gargmanas/movierecommenderdataset?select=ratings.csv

	A	B	C	D
1	userId	movieId	rating	timestamp
2	1	1	4.0	964982703
3	1	3	4.0	964981247
4	1	6	4.0	964982224
5	1	47	5.0	964983815
6	1	50	5.0	964982931
7	1	70	3.0	964982400
8	1	101	5.0	964980868
9	1	110	4.0	964982176
10	1	151	5.0	964984041
11	1	157	5.0	964984100
12	1	163	5.0	964983650
13	1	216	5.0	964981208
14	1	223	3.0	964980985
15	1	231	5.0	964981179
16	1	235	4.0	964980908
17	1	260	5.0	964981680
18	1	296	3.0	964982967
19	1	316	3.0	964982310
20	1	333	5.0	964981179
21	1	349	4.0	964982563
22	1	356	4.0	964980962
23	1	362	5.0	964982588
24	1	367	4.0	964981710
25	1	423	3.0	964982363
26	1	441	4.0	964980868
27	1	457	5.0	964981909
28	1	480	4.0	964982346
29	1	500	3.0	964981208

Рисунок 14 – Фрагмент датасета ratings.csv

На следующем этапе необходимо совместить датасет фильмов и датасет рейтингов по полю идентификатора фильма.

```
merged_list = pd.merge(movies, ratings, on="movieId")
```

Теперь датасеты готовы к работе.

Выполним пошагово алгоритм построения рекомендаций.

Шаг 1 – необходимо определить самый популярный жанр для пользователя.

Сначала указываем идентификатор пользователя. В примере ниже для построения рекомендаций выбран пользователь с идентификатором 8.

```
    user_id = 8  # Genre 'Comedy'
    print('MOVIE RECOMMENDATIONS FOR USERID:', user_id, '\n')
    print('---------- (1) GET USER TOP GENRE ----------')
```

Осуществляем выбор всех фильмов, которые оценил указанный пользователь

```
    user_list = merged_list[merged_list['userId'] == user_id]
```

Выполняем генерацию списка с количеством оцененных пользователем фильмов (по жанрам).

```
    print('---------- Displaying all genres for userId:', user_id, '----------')
    genre_list = user_list['genres'].str.split('|', expand=True).stack().value_counts()
    print(genre_list)
```

Получаем наиболее популярный для пользователя жанр.

```
    top_movie_genre = genre_list.index[0]
    print('---------- Top Genre for userId', user_id, ':', top_movie_genre, '----------')
    get_movie_recommendation_by_genre(top_movie_genre)
```

Результат определения популярного у пользователя 8 жанра приведен на рисунке 15.

```
MOVIE RECOMMENDATIONS FOR USERID: 8

---------- (1) GET USER TOP GENRE ----------
---------- Displaying all genres for userId: 8 ----------
Comedy       24
Drama        19
Thriller     16
Romance      14
Action       12
Adventure    11
Crime         9
Fantasy       4
Children      4
Sci-Fi        4
Mystery       3
War           3
IMAX          2
Horror        2
Western       2
Animation     1
Musical       1
dtype: int64
---------- Top Genre for userId 8 : Comedy ----------
```

Рисунок 15 – Определение популярного жанра

Шаг 2 - необходимо составить обучающую выборку. В нее попадут фильмы, которые были оценены более чем 15 раз пользователями, которые оценили более 50 фильмов.

Для этого необходимо сначала задать количество рекомендуемых фильмов.

```
n_movies_to_recommend = 10
```

Строим тренировочный датасет.

```
    print('')
    print('---------- (2) CREATE TRAINING DATA SET ----------')
    # extract movies under input genre (Contains)
    #                                   movie_list        = merged_list[merged_list['genres'].str.contains(movie_genre)]
    # extract movies under input genre (Match)
    print('')
    print('--------------- (2a) Get All Movies Under Genre: ', movie_genre,
                ' ---------------')
```

Список отзывов к фильмам жанра movie_genre - самый популярный в отзывах пользователя с id user_id.

```
    movie_list                                                = merged_list[merged_list['genres'].str.match(movie_genre)]
    print(movie_list)
```

Выбираем фильмы, которые имеют жанр не Триллер.

```
    movie_list                                                = movie_list[~movie_list['genres'].str.contains('Thriller')]
```

Выбираем фильмы по году выпуска.

```
     #                                  movie_list        = movie_list[movie_list['title'].str.contains('19')]
```

Выполняем построение разреженной матрицы оценок пользователями фильмов.

```
    genre_ratings = movie_list.pivot(index='movieId', columns='userId', values='rating')
```

Если фильм пользователем не оценен, то считаем оценку = 0.

```
    genre_ratings.fillna(0, inplace=True)
    print('')
    print(genre_ratings)
```

Из матрицы genre_ratings выбираем фильмы, которые были оценены более 15 раз пользователями, которые оценили как минимум 50 фильмов. Таким образом формирует тренировочный датасет.

Выбираем фильмы, которые были оценены как минимум 15 раз.

```
    print('')
    print('---------------  (2b) Must Be Movies That Have Been Rated At Least 15 Times ---------------')
```

Выполняем подсчет, сколько человек оценило каждый из фильмов.

```
     no_user_voted = movie_list.groupby('movieId')['rating'].agg('count')
        genre_ratings = genre_ratings.loc[no_user_voted[no_user_voted > 15].index, :]
    print(genre_ratings)
```

Выбираем фильмы, которые были оценены как минимум рейтингом 3.0.

```
    #    genre_ratings = genre_ratings.loc["rating"["rating" > 3].index,:]
```

Пользователь должен оценить как минимум 50 фильмов.

```
print('')
print('--------------- (2c) Must Be Users Who Have Rated At Least 50 Times ---------------')
no_movies_voted = movie_list.groupby('userId')['rating'].agg('count')
    genre_ratings = genre_ratings.loc[:,no_movies_voted[no_movies_voted > 50].index]
print(genre_ratings)
```

Визуализируем количество оценок у фильмов из тренировочного датасета.

```
f, ax = plt.subplots(1, 1, figsize=(16, 4))
f, ax.set_xlim(0, 100)
plt.scatter(no_user_voted.index, no_user_voted, color='mediumblue')
plt.axhline(y=15, color='r')
plt.xlabel('Идентификатор фильма')
plt.ylabel('Кол-во оценивших')
plt.title('Количество оценок у фильмов из тренировочного датасета')
plt.show()
```

Строим график количества оценок по пользователям.

```
f, ax = plt.subplots(1, 1, figsize=(16, 4))
```

```
    plt.scatter(no_movies_voted.index,      no_movies_voted,
color='mediumseagreen')
    plt.axhline(y=50, color='r')
    plt.xlabel('Идентификатор пользователя')
    plt.ylabel('Кол-во голосов')
    plt.title('График количества оценок по пользователям')
    plt.show()
```

На рисунке 16 приведен график количества оценок у фильмов выбранного жанра на примере первых 200 фильмов. В обучающую выборку пойдут только те из них, которые будут иметь больше 15 голосов.

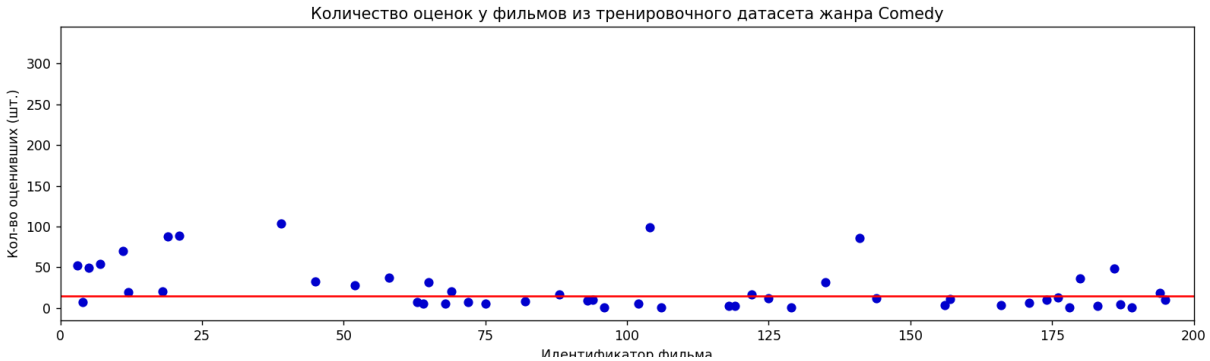

Рисунок 16 – График количества оценок фильмов

На рисунке 17 приведен график количества оценок пользователями фильмов выбранного жанра. Красной линией обозначен порог (50 голосов), необходимый для учета голосов таких пользователей в тренировочном датасете.

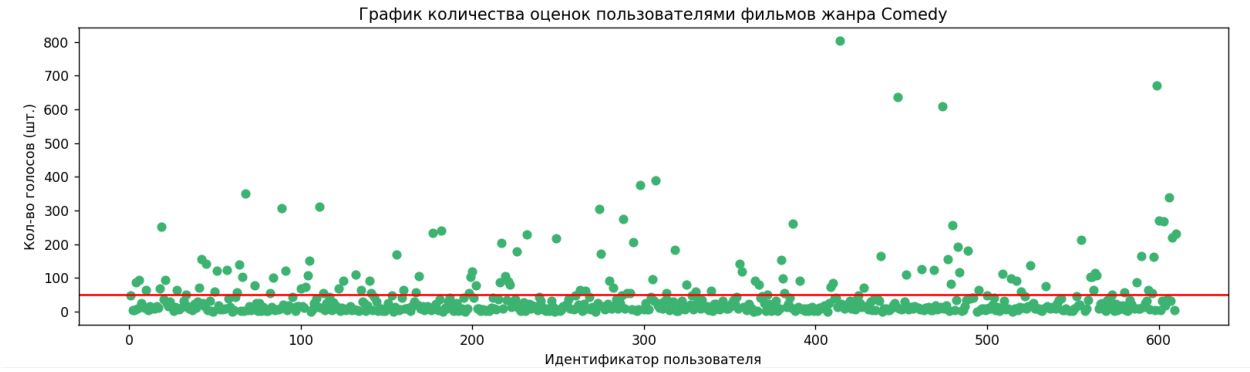

Рисунок 17 – График количества оценок пользователями фильмов выбранного жанра

Убираем разреженность и получить сжатые разреженные строки (ненулевые значения).

```
csr_data = csr_matrix(genre_ratings.values)
print('')
print('--------------- (2d) Training Data Set Created! (In CSR Format: X, Y, Rating) ---------------')
print(csr_data)
# reset index i.e., row index starts at 0 instead of 1
    genre_ratings.reset_index(inplace=True)
```

Устанавливаем параметры knn модели (https://scikit-learn.org/stable/modules/neighbors.html)

Метрика - metric: cosine / manhattan / euclidean (default).

Алгоритм изучения ближайших соседей - algorithm: brute / ball_tree / kd_tree / auto (default).

Количество исследуемых соседей - n_neighbors.

Параллелизм - n_jobs: 1 (no joblib parallelism) / -1 (use all CPUs).

Например: knn = NearestNeighbors(metric='cosine', algorithm='brute', n_neighbors=20, n_jobs=-1).

```
print('')
```

```
    print('---------- (3) START TRAINING ----------')
    knn        =        NearestNeighbors(algorithm='brute',
n_neighbors=20, n_jobs=-1)
```

Загружаем тренировочный датасет.

```
    knn.fit(csr_data)
    print('')
    print('--------------- Training Done! ---------------')
```

Проверяем матрицу genre_ratings на пустоту.

```
    if len(genre_ratings):
       # get movieId with the highest total rating
       #                                    total_rating    =
movie_list.groupby('movieId')['rating'].sum()
       #                                       movie_idx    =
genre_ratings.iloc[total_rating[total_rating > 75].index,:]
       #        print(total_rating)
       #        genre_ratings.sort(reverse=True)
```

Получаем id первого фильма из user_list (список фильмов, оцененных пользователем)

```
    user_movie_list                                        =
user_list[user_list['genres'].str.match(movie_genre)]
    print('')
    print('---------- (4) START PREDICTING ----------')
    print('')
    print('---------------  (4a)  Get  All  Movies  Watched  By
User Under Genre:', movie_genre, '---------------')
    print(user_movie_list)
```

Выполняем построение разреженной матрицы оценок фильмов выбранным пользователем.

```
    user_genre_ratings = user_movie_list.pivot(index='movieId', columns='userId', values='rating')
    user_genre_ratings.fillna(0, inplace=True)
    user_genre_ratings.reset_index(inplace=True)
    movie_idx = user_genre_ratings.iloc[0]['movieId']  # id первого по счету фильма, оцененного пользователем
    # get first movieId from genre_ratings
    #        movie_idx = genre_ratings.iloc[0]['movieId']
    print('')
    print('--------------- (4b) Select A Movie From User List (Independent Variable) ----------')
    print('Movie Id: ', movie_idx, 'selected!')
```

Получаем индекс movieId.

```
    movie_idx = genre_ratings[genre_ratings['movieId'] == movie_idx].index[0]
    print('')
    print('Index No.:', movie_idx)
    print(genre_ratings)
```

Шаг 3 – Вывод рекомендаций.

Результатом работы рекомендательной системы является 10 фильмов, основанные на информации о фильмах из датасета. Количество фильмов в рекомендации указывается в начале программы.

Строим рекомендации.

```
    print('')
    print(f'---------------     (4c)     Getting     Nearest
K={movie_genre} Neighbors To Selected Movie ---------------
')
    print(csr_data[movie_idx])
    distances, indices = knn.kneighbors(csr_data[movie_idx],
n_neighbors=n_movies_to_recommend + 1)
```

Трансформируем результат в удобный формат для вывода.

```
    rec_movie_indices = sorted(list(
    zip(indices.squeeze().tolist(),
distances.squeeze().tolist())),
                                        key=lambda         x:
x[1])[:0:-1]
     #                                rec_movie_indices    =
sorted(list(zip(indices.squeeze().tolist(),distances.squeeze
().tolist())),key=lambda x: x[1],reverse=True)[:0:-1]
    print('')
    print('---------------  (4d)  Getting  Neighbors  Done!
(Movie Index No., Distance) ---------------')
    print(rec_movie_indices)
    recommend_frame = []
    print('')
    print(f'---------------           (4e)           Display
{n_movies_to_recommend} Movie Recommendations -------------
-')
```

Далее строим результирующую таблицу рекомендаций.

```
    for val in rec_movie_indices:
            movie_idx = genre_ratings.iloc[val[0]]['movieId']
            idx = movies[movies['movieId'] ==
movie_idx].index
             print('Processing Movie: ', 'ID -',
int(movie_idx), 'Title -',
movies.iloc[idx]['title'].values[0])
    recommend_frame.append({'ID':movies.iloc[idx]['movieId']
.values[0],'Title':movies.iloc[idx]['title'].values[0],'Genr
es':movies.iloc[idx]['genres'].values[0],'Rating':ratings.il
oc[idx]['rating'].values[0],'Distance':val[1]})
    print('')
    df = pd.DataFrame(recommend_frame, index=range(1,
n_movies_to_recommend+1))
    return df
    else:
       return "No movies found. Please check your input"
```

На рисунке 18 приведены рекомендации – 10 фильмов для пользователя с идентификатором 8.

```
--------------- (4e) Display 10 Movie Recommendations ---------------
Processing Movie:  ID -  838 Title - Emma (1996) Genre - Comedy|Drama|Romance
Processing Movie:  ID - 1381 Title - Grease 2 (1982) Genre - Comedy|Musical|Romance
Processing Movie:  ID -  234 Title - Exit to Eden (1994) Genre - Comedy
Processing Movie:  ID - 5943 Title - Maid in Manhattan (2002) Genre - Comedy|Romance
Processing Movie:  ID -  276 Title - Milk Money (1994) Genre - Comedy|Romance
Processing Movie:  ID -  252 Title - I.Q. (1994) Genre - Comedy|Romance
Processing Movie:  ID -    7 Title - Sabrina (1995) Genre - Comedy|Romance
Processing Movie:  ID -  361 Title - It Could Happen to You (1994) Genre - Comedy|Drama|Romance
Processing Movie:  ID -  852 Title - Tin Cup (1996) Genre - Comedy|Drama|Romance
Processing Movie:  ID -  440 Title - Dave (1993) Genre - Comedy|Romance
```

Рисунок 18 – предлагаемые фильмы для пользователя, оценившего комедии

В качестве других примеров были построены рекомендации для других пользователей.

Выбраны пользователи с идентификаторами 2 и 3. На рисунке 19 представлен анализ оценок, выставленных пользователем 2, выбран преимущественный жанр для него – Драма.

```
---------- (1) GET USER TOP GENRE ----------
---------- Displaying all genres for userId: 2 ----------
Drama          17
Action         11
Crime          10
Thriller       10
Comedy          7
IMAX            4
Sci-Fi          4
Adventure       3
Documentary     3
Mystery         2
Romance         1
War             1
Horror          1
Western         1
dtype: int64
---------- Top Genre for userId 2 : Drama ----------
```

Рисунок 19 – Анализ оценок пользователя с идентификатором 2

На рисунке 20 представлен рейтинг фильмов пользователя с идентификатором 3 и выбран его любимый жанр.

```
---------- (1) GET USER TOP GENRE ----------
---------- Displaying all genres for userId: 3 ----------
Drama        16
Sci-Fi       15
Action       14
Adventure    11
Comedy        9
Horror        8
Thriller      7
War           5
Romance       5
Children      5
Animation     4
Fantasy       4
Crime         2
Musical       1
Mystery       1
dtype: int64
---------- Top Genre for userId 3 : Drama ----------
```

Рисунок 20 – Анализ оценок пользователя с идентификатором 3

Для выбранного жанра Драма выполнен количественный анализ оценок фильмов этого жанра в тренировочном датасете. Визуализация анализа приведена на рисунке 21.

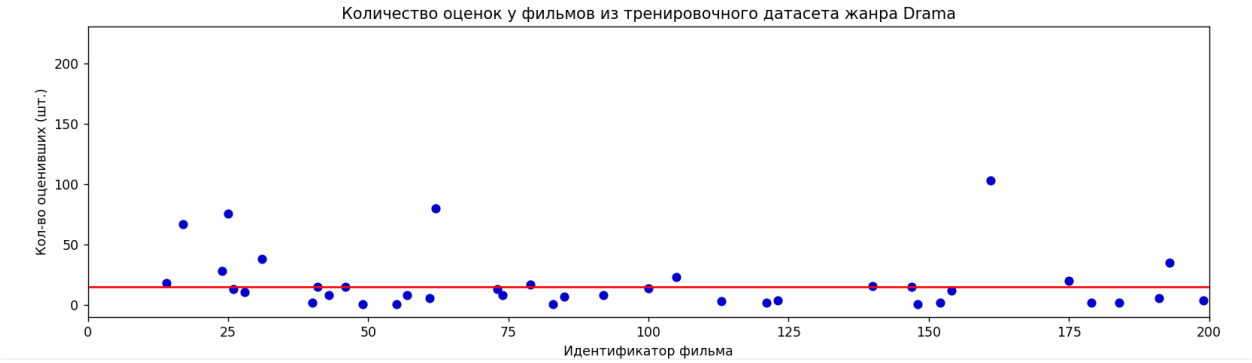

Рисунок 21 – Анализ оценок фильмов в жанре Драма

На рисунке 22 приведен график количественного анализа оценок всех пользователей, оценивших фильмы жанра Драма.

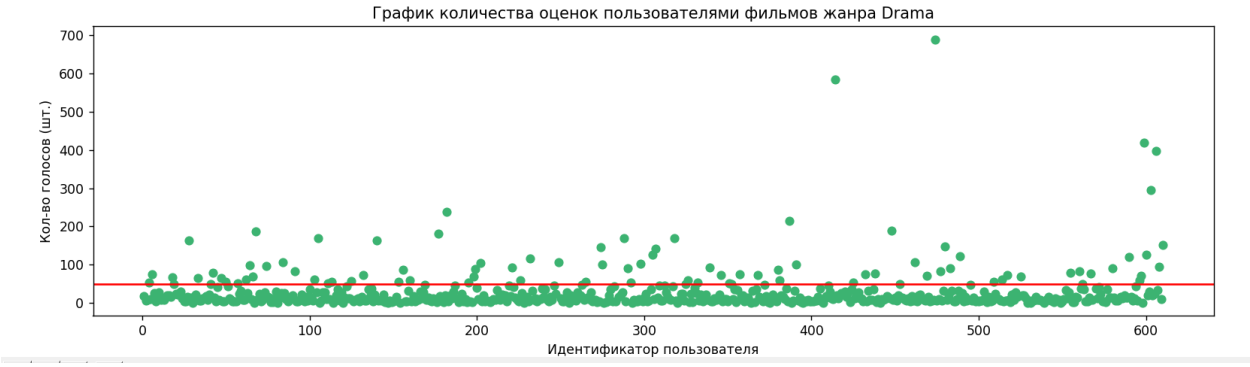

Рисунок 22 – Анализ оценок пользователями фильмов жанра Драма

Далее выполнено построение рекомендаций для пользователя с идентификатором 2 (рисунок 23).

```
--------------- (4e) Display 10 Movie Recommendations ---------------
Processing Movie:  ID - 4720 Title - Others, The (2001) Genre - Drama|Horror|Mystery|Thriller
Processing Movie:  ID - 3994 Title - Unbreakable (2000) Genre - Drama|Sci-Fi
Processing Movie:  ID - 1246 Title - Dead Poets Society (1989) Genre - Drama
Processing Movie:  ID - 7361 Title - Eternal Sunshine of the Spotless Mind (2004) Genre - Drama|Romance|Sci-Fi
Processing Movie:  ID - 1653 Title - Gattaca (1997) Genre - Drama|Sci-Fi|Thriller
Processing Movie:  ID - 2291 Title - Edward Scissorhands (1990) Genre - Drama|Fantasy|Romance
Processing Movie:  ID - 2762 Title - Sixth Sense, The (1999) Genre - Drama|Horror|Mystery
Processing Movie:  ID - 1721 Title - Titanic (1997) Genre - Drama|Romance
Processing Movie:  ID - 4995 Title - Beautiful Mind, A (2001) Genre - Drama|Romance
Processing Movie:  ID - 1961 Title - Rain Man (1988) Genre - Drama
```

Рисунок 23 – Рекомендации для пользователя с идентификатором 2

На следующем рисунке 24 приведены рекомендации для пользователя с идентификатором 3.

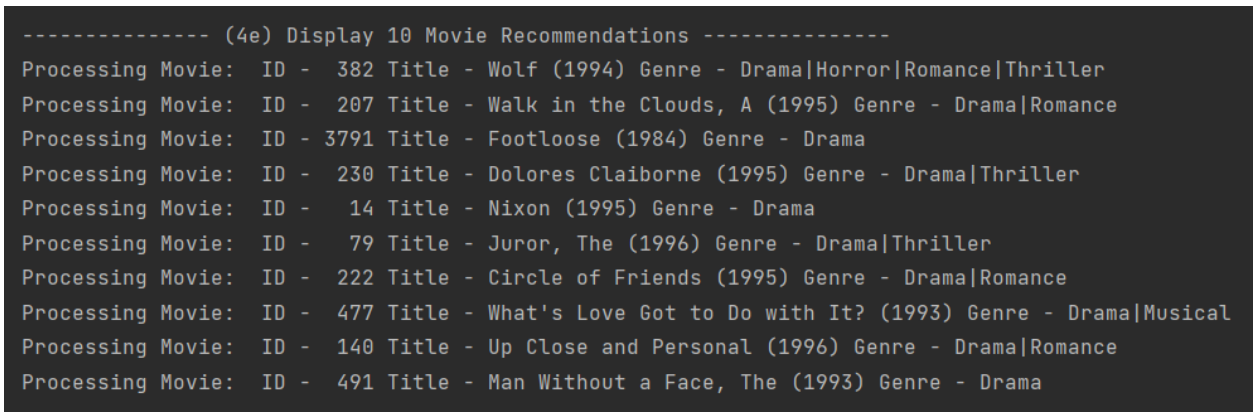

Рисунок 24 – Рекомендации для пользователя с идентификатором 3

Таким образом, рекомендательный сервис на основе метода коллаборативной фильтрации строит рекомендации из 10 фильмов по наиболее популярному у активного пользователя жанру.

ЗАКЛЮЧЕНИЕ

В выпускной квалификационной работе представлена разработка рекомендательного сервиса по фильмам.

При подготовке работы были изучены основные методы построения рекомендательных систем, выполнен обзор нескольких существующих рекомендательных систем, перечислены меры определения схожести объектов, которые можно использовать в рекомендательных системах для определения похожести пользователей.

На языке программирования Python с использованием ряда библиотек выполнена реализация рекомендательного сервиса фильмов. В качестве основного метода взят метод коллаборативной фильтрации, в качестве меры схожести использован алгоритм k-ближайших соседей. Исходные датасеты фильмов и их оценок взяты из открытых источников.

Сервис позволяет определить наиболее популярный жанр фильмов для выбранного пользователя и построить для него рекомендацию из 10 фильмов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Кластеризация: алгоритмы k-means и c-means // Habr [сайт]. – URL: https://habr.com/ru/post/67078/ (дата обращения 10.11.2022).

2. Коллаборативная фильтрация: как предпочтения других пользователей влияют на твои рекомендации // VC [сайт]. – URL: https://vc.ru/marketing/153788-kollaborativnaya-filtraciya-kak-predpochteniya-drugih-polzovateley-vliyayut-na-tvoi-rekomendacii (дата обращения 10.11.2022).

3. Рекомендательные системы: user-based и item-based. // Habr [сайт]. – URL: https://habr.com/ru/company/surfingbird/blog/139518/ (дата обращения 10.11.2022).

4. Рекомендательные системы: идеи, подходы, задачи. // Habr [сайт]. – URL: https://habr.com/ru/company/jetinfosystems/blog/453792/ (дата обращения 10.11.2022).

5. Авхадеев, Б.Р. Разработка рекомендательной системы на основе данных из профиля социальной сети «ВКонтакте» / Б.Р. Авхадеев, Л.И. Воронова, Е.П. Охапкина // Вестник НВГУ. – Нижневартовск: НВГУ, 2014. №3. С. 68-76.

6. Меньшикова, Н.В. Обзор рекомендательных систем и возможностей учета контекста при формировании индивидуальных рекомендаций / Н.В. Меньшикова, И.В. Портнов, И.Е. Николаев // Academy. – Москва. 2016. №6 (9). С.20-22.

7. Трофимова, Е.В. Разработка рекомендательной системы на основе анализа тональности текста / Е.В. Трофимова, К.А. Туральчук // Актуальные проблемы гуманитарных и естественных наук. – Москва: МГУ, 2015. №1-1. С.93-94.

8. Цурко, В.В. Рекомендательные системы в здравоохранении / В.В. Цурко // УБС. – Екатеринбург: УБС, 2019. №82. С. 61-77.