САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ – ПРОЦЕССОВ УПРАВЛЕНИЯ
КАФЕДРА ТЕОРИИ УПРАВЛЕНИЯ

Малышев Илья Александрович

Выпускная квалификационная работа магистра

Моделирование предпочтений в персонализированной рекомендательной системе онлайн сервиса

НАПРАВЛЕНИЕ 01.04.02 «ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»
ОБРАЗОВАТЕЛЬНАЯ ПРОГРАММА ВМ.5517.2017 «МЕТОДЫ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ В ЗАДАЧАХ УПРАВЛЕНИЯ»

Научный руководитель,
доктор технических наук,
профессор Буре В. М.

Рецензент Степанов П. Н.

Санкт-Петербург
2019

Содержание

Введение

Современные крупные компании, а также мелкие индивидуальные предприниматели используют мировую сеть Интернет для продвижения и расширения своего бизнеса. Большинство подобных организаций располагают очень большим объемом данных различного рода, при этом значительная часть информации располагается на всевозможных интернет ресурсах, принадлежащих компании. И, зачастую, пользователь тратит слишком много времени, изучая контент полностью самостоятельно. Чтобы организовать свою деятельность наиболее эффективно, каждое предприятие любого уровня, использует различные инструменты для поиска, обработки, систематизации имеющихся данных. Основной целью описываемых инструментов является помощь пользователям в выборе тех или иных действий, при условии наличия некоторой информации о них. Поэтому возникает проблема выбора подходящего способа навигации по данным, размещенным в веб-сервисе. Актуальность данной задачи обусловлена слишком большим объемом информации о товарах или услугах, предоставляемой на сайтах различных компаний или интернет- магазинов. Решением этой задачи служат рекомендательные системы.

В работе [1] рекомендательные системы определены как программные инструменты и методы, предлагающие пользователям те объекты, которые им могут быть интересны. Объект в данном случае – это общий термин, используемый для обозначения того, что система рекомендует пользователям. Объектами рекомендаций могут служить товары и услуги в интернет-магазине, медиа-контент, другие пользователи веб-сервиса и так далее. Рекомендательные системы анализируют поведение пользователей или анкетные данные клиентов интернет-сервиса, после чего могут давать оценку предпочтения пользователем того или иного объекта рекомендаций. Если каждому пользователю предлагается свой набор рекомендованных

объектов, отличный от рекомендаций других пользователей, то такой функционал называют персонализацией.

Результатом внедрения рекомендательной системы является очевидное улучшение взаимодействия между пользователем и веб-сервисом. С помощью этого инструмента могут быль улучшены многие показатели эффективности работы, такие как объемы продаж, коэффициент дохода, степень вовлеченности пользователя, лояльность клиентов к сервису.

В настоящей работе будет рассмотрен интернет ресурс букмекерской конторы. Сервис позволяет зарегистрированным клиентам компании делать ставки на спорт в режиме онлайн из любого места страны.

При входе в личный кабинет пользователя, клиенту предлагается список событий, на каждое из которых он может сделать ставку. В виду широкого спектра видов спорта, а также большого количества мероприятий и участников в каждом из них, клиент тратит много времени для поиска интересующего его события.

Цель данной работы заключается в создании персонализированной рекомендательной системы для предложения спортивных событий клиенту, основанной на поведении пользователя на сайте букмекерской конторы.

Работа состоит из пяти глав. В первой главе вводится формальное определение рекомендательной системы, производится обзор методов сбора данных и подходов к формированию рекомендаций.

Во второй главе рассматривается задача совместной фильтрации сначала в контексте memory-based алгоритма, затем в контексте model-based подхода.

В третьей главе описывается формирование модели, начиная с общей модели SVD, с последующей модернизацией в современную модель

латентных векторов, с учетом особенностей веб-сервиса букмекерской конторы.

В четвертой главе производится разбор особенностей букмекерских ставок, с тем чтобы адаптировать построенную модель для улучшения рекомендаций. Также описывается метод построения решения.

В пятой главе проводится анализ полученных результатов, дается оценка качества рекомендаций, а также иллюстрация результатов работы демонстрационной версии рекомендательной системы на примере двух разных наборов данных.

1 Принцип построения рекомендательных систем

1.1 Представление данных

Задача рекомендательной системы – проинформировать пользователя об объекте, который может его заинтересовать. Для выполнения поставленной задачи важно рассмотреть весь процесс формирования рекомендаций: от получения информации до ее представления пользователю.

В основе разрабатываемой рекомендательной системы находится матрица данных, называемая матрицей предпочтений[2].

Это матрица, у которой по вертикальной оси отложены все пользователи сервиса (Users), а по горизонтальной – объекты рекомендации (Items). На пересечении некоторых пар (user, item) данная матрица заполнена числовыми оценками (Ratings) – это известный показатель заинтересованности пользователя в данном объекте.

Вводятся обозначения: множество пользователей U, а множество объектов I. Таким образом, имеется набор пар $R = \{(u, i)\} \in U \times I$, для которых известна вещественная оценка предпочтения r_{ui}.

	Item1	Item2	Item3	Item4
User1	27	4		
User2		24	8	
User3		2		15
User4	35	20		
User5		20	3	18
User6			25	
User7	14		5	
User8		13		19
User9	3		31	

Рис. 1: Табличное представление матрицы предпочтений

Пользователи могут оценить(выбрать) только часть объектов, предлагаемых сервисом в виду обширного перечня наименований объектов, и задача рекомендательной системы - обобщить эту информацию и

предсказать отношение клиента к другим объектам, про которые неизвестно пользователю. То есть заполнить пропуски в матрице предпочтений. При этом не обязательно рекомендовать неизвестные позиции, иногда имеет смысл предложить пользователю те объекты, которые он выбирает на постоянной основе.

1.2 Сбор данных для рекомендательной системы

Сбор данных для рекомендательной системы может быть реализован двумя способами: явный и неявный.

Для явного сбора данных характерно то, что пользователь сам предоставляет всю необходимую для дальнейшей обработки информацию. Это может быть либо при регистрации на ресурсе, либо с помощью заполнения предлагаемых анкет. К таким данным могут относиться пол, возраст, геолокация, а также явные оценочные суждения пользователя о контенте (шкалированная оценка фильма, рейтинг продукта и тд.).

В случае если пользователь не предоставляет никакую информацию, неявный сбор данных становится более актуальным. В этой ситуации собирается доступная информация о взаимодействиях пользователя с веб-сервисом. Это может быть история покупок, количество «лайков», комментарии, отзывы или просто просмотры. Данные, полученные в результате того или иного метода, называют обратной связью пользователя. Наиболее важные аспекты подхода с неявным сбором описываются в статье [3] на примере телевизионного сервиса.

В силу особенностей рассматриваемого веб-сервиса в данной работе будет описываться система с неявной обратной связью. Безусловно, в связи с этим возникают некоторые проблемы, которые в частном случае конкретной букмекерской конторы решаемы. Во- первых, возникает так называемая проблема отсутствия негативной реакции. Она заключается в том, что, исходя из полученных данных о пользователе, нельзя однозначно сделать вывод о каких объектах клиент не знает, а какие ему именно не нравятся. Во-

вторых, гораздо более остро проявляется проблема шума. При пассивном отслеживании поведения пользователей, можно только строить предположения относительно их собственных предпочтений и других причин выбора того или иного объекта. В рассматриваемой сфере букмекерского бизнеса игрок может сделать ставку на конкретную команду по причине наличия арбитражной ситуации в событии. Подобного рода выбор является шумом, мешающим объективно оценить предпочтения клиента. В-третьих, строго говоря, числовое значение неявной обратной связи в матрице предпочтений характеризует степень уверенности, в то время как значение явной обратной связи дает чистую оценку предпочтения. Чем выше это значение для конкретного объекта, тем активнее пользователь выбирает именно его. То есть, тем больше уверенность в заинтересованности данного клиента. Поэтому очень важно выбрать хорошо масштабируемую характеристику заинтересованности пользователя в объекте, чтобы отличать одиночные взаимодействия от стабильно повторяющихся.

1.3 Формирование рекомендаций

Различают несколько подходов к формированию рекомендаций:

- На основании признаковых описаний или содержания (content-based)

Такой подход предполагает наличие достаточного объема информации про рекомендуемые объекты или про пользователей. Каждый пользователь $u \in U$ и каждый объект $i \in I$ поддаются содержательному описанию признаковыми векторами $x(u)$ и $y(i)$. Формирование рекомендаций происходит для похожих объектов, заказанных одним пользователем, либо для схожих пользователей, заказавших один товар. Степень сходства всех объектов и пользователей оценивается по соответствующим характеристикам. Например, для букмекерских ставок это могут быть: вид спорта, вид соревнования, близкие коэффициенты. При этом подходе по имеющимся данным о взаимодействии пользователей и объектов на сервисе

строится обучающая выборка и задача предсказания предпочтений сводится к задаче обучения по прецедентам. [6]

Применение такого подхода на практике весьма затратно, так как сбор описательных данных достаточно ресурсоемкая процедура, которую, к тому же, возможно реализовать не на любом сервисе.

- Совместная фильтрация (collaborative filtering)

Совместная, или коллаборативная, фильтрация используется, когда при создании рекомендательной системы нет никакой информации о свойствах объектов и пользователей. Прогнозирование происходит исключительно на основании истории поведения пользователей на сервисе, то есть по истории взаимодействия их с объектами.

- Комбинированный подход

В данном случае совмещаются результаты методов основанных на признаковых описаниях и совместной фильтрации. Композиция результатов коллаборативной и содержательной фильтрации зачастую позволяет повысить точность рекомендаций. Например, предполагаемые предпочтения исследуемого пользователя вычисляются с помощью коллаборативной фильтрации, при этом предпочтения остальных пользователей учитываются в алгоритме на основании признаковых описаний.

Также различают еще два подхода выделяемых по иному принципу [7]:

- Основанный на всех записях (Memory-based)

Прогнозирование происходит через вычисление некоторой меры по всем накопленным данным, непосредственно через элементы матрицы предпочтений. Данный подход достаточно прост с точки зрения понимания, он показывает высокую точность прогнозирования (для небольшого набора данных), а также обладает свойством инкрементального учета новых данных.

Однако, этот подход очень затратный с точки зрения времени и ресурсов памяти.

- Основанный на моделях (Model-based)

По имеющимся данным строится описательная модель предпочтений пользователей, объектов, и их взаимодействий, и рекомендации формируются на основании полученной модели. Таким образом, процесс формирования рекомендаций в общем виде состоит из двух этапов: обучение модели и вычисление рекомендаций в реальном времени. Основным недостатком подхода является отсутствие поддержки инкрементального обучения, т.е. появление новых объектов требует перестроения модели.

2 Задача совместной фильтрации

2.1 Постановка задачи совместной фильтрации

Как было сказано выше, коллаборативная фильтрация — подход, при котором предсказание предпочтения происходит исключительно с использованием информации о поведении пользователей на веб-сервисе. Именно поэтому такой подход наиболее предпочтителен при работе с неявной обратной связью.

Данные об известной обратной связи представлены в виде набора троек («пользователь», «объект», «оценка предпочтения»), как записи в базе данных.

$$D = \{(u, i, r_{ui})\}_{(u,i)} \in R, \quad (1)$$

где $r_{ui} \in$ R — числовое значение предпочтения объекта $i \in I$ пользователем $u \in U$; $R \subseteq$ U × I — множество пар («пользователь», «объект»), для которых известна степень заинтересованности. Таким образом формируется матрица предпочтений R, описанная выше, элементы которой принимают значения r_{ui}, , если : $(u, i) \in R$, иначе $- \emptyset$.

Ставится задача: по известной информации D построить предсказание предпочтения $\widehat{r_{ui}} \approx r_{ui}$ для новых пар $(u, i) \notin R$. Другими словами, задача совместной фильтрации заключается в заполнении пропущенных значений в матрице предпочтений.

Кроме определения числовой оценки предпочтений пользователя метод совместной фильтрации позволяет вычислить степень сходства объектов и пользователей соответственно между собой.

Методы решения задачи совместной фильтрации разделяются на две группы, соответственно вышеописанным подходам Memory-based и Model-based.

2.2 Memory-based алгоритмы

К группе Memory-based относятся такие алгоритмы, в которых неизвестные значения рейтинга объектов выражаются непосредственно через имеющиеся элементы матрицы предпочтений. При этом оценка неизвестных рейтингов может производится на основе схожих объектов, либо схожих пользователей. Такие подходы называют предметно-ориентированными. В случае, когда вычисление производится на схожих объектах алгоритм принято называть Item-based, в случае же схожих пользователей соответственно User-based. Это стандартные примеры Memory-based алгоритма, в которых производится взвешивание предпочтений.

Как уже было отмечено, основным недостатком алгоритмов совместной фильтрации является выполнение большого количества операций. Для уменьшения трудоемкости вычислений можно рассматривать не всех пользователей, а лишь некоторое количество k наиболее похожих. Очевидно, что выбор числа k оказывает сильное влияние на точность предсказаний. Общей тенденцией является увеличение точности при начальном увеличении числа k, а затем точность стабилизируется и плавно ухудшается. Ухудшение точности обусловлено тем, что все больше непохожих объектов принимается к рассмотрению.

Например, алгоритм Item-based математически можно сформулировать следующим образом[14]:

$$\widehat{r_{ui}} = \frac{\sum_{j \in S^k(i;u)} S_{ij}\, r_{uj}}{\sum_{j \in S^k(i;u)} S_{ij}}, \qquad (2)$$

где k – количество объектов, похожих на -тый, $S^k(i;u)$ −множество k ближайших i-тому объекту соседей. Таким образом, предсказанное $\widehat{r_{ui}}$ принимается как средневзвешенное значение рейтингов соседних объектов.

Меры схожести вычисляются по матрице предпочтений R. Мера схожести является важным параметром алгоритма. Наиболее общеупотребимые простые метрики схожести — корреляция Пирсона и косинусное расстояние соответствующих строк (столбцов) матрицы предпочтений.

Подробный пример, реализованной на основе memory-based алгоритма, рекомендательной системы приведен в статье [4]. Такие алгоритмы актуальны для единичного прогнозирования предпочтений пользователей, помимо этого, выбор достаточно хорошей меры сходства это дополнительная задача, на решение которой требуется отдельное исследование.

Одна из основных проблем предметно-ориентированных методов – невозможность работать с неявной обратной связью. Они не способны делать различия между пользовательскими предпочтениями и степенью уверенности в рейтинге, с которой приходится иметь дело при работе с неявными данными. Другой немаловажной проблемой Memory-based методов является большая неточность предсказаний при сильной разреженности матрицы предпочтений R. В случае букмекерской конторы матрица предпочтений не сможет стать достаточно плотной, в виду широкой возможности выбора событий, зачастую пользователь отдает предпочтение 5-10 объектам. Поэтому подобные алгоритмы в дальнейшем рассматриваться не будут.

2.3 Model-based алгоритмы

Алгоритмы из второй группы model-based работают на основе описательной модели взаимодействий пользователей с объектами. Например, алгоритмы, работающие с явной обратной связью, используют непосредственно имеющиеся рейтинги, избегая переобучения с помощью регуляризованной модели такой как:

$$\min_{x,y} \sum_{r_{u,i} \in D} (r_{ui} - x_u^T y_i)^2 + \Omega(\theta) \qquad (3)$$

Здесь$(r_{ui} - x_u^T y_i)^2 -$ квадратичные отклонения, $\Omega(\theta)$ – регуляризатор на множестве параметров θ. Для более полного описания будет рассмотрена конкретная модель.

3 Построение модели

3.1 Матричные разложения

Имеющиеся данные в виде матрицы предпочтений с помощью матричных разложений можно интерпретировать как произведение двух других матриц, характеризующих скрытые признаки пользователей и, соответственно, объектов[11]. Например, рассматривается матрица $R \in R^{n \times m}$. Из определения ранга матрицы следует, что для числа $k \geq rank(R)$ найдутся такие $X \in R^{n \times k}, Y \in R^{m \times k}$, что:

$$R = XY^T$$

Но более важен случай когда число $k \leq rank(R)$, в этом случае только о приближении ранга k матрицы R, которое определяется как:

$$XY^T = \tilde{R} \approx R,$$

где $X \in R^{n \times k}, Y \in R^{m \times k}$, матрицы приближения. Матрицы X и Y^T содержат по строке для каждого пользователя и каждого объекта соответственно. Строки имеют k значений, каждое из них сопоставляется латентному свойству в модели. Произведение матриц даёт приближенную полную оценку для целостной и плотной матрицы взаимодействий «пользователь-объект».[17]

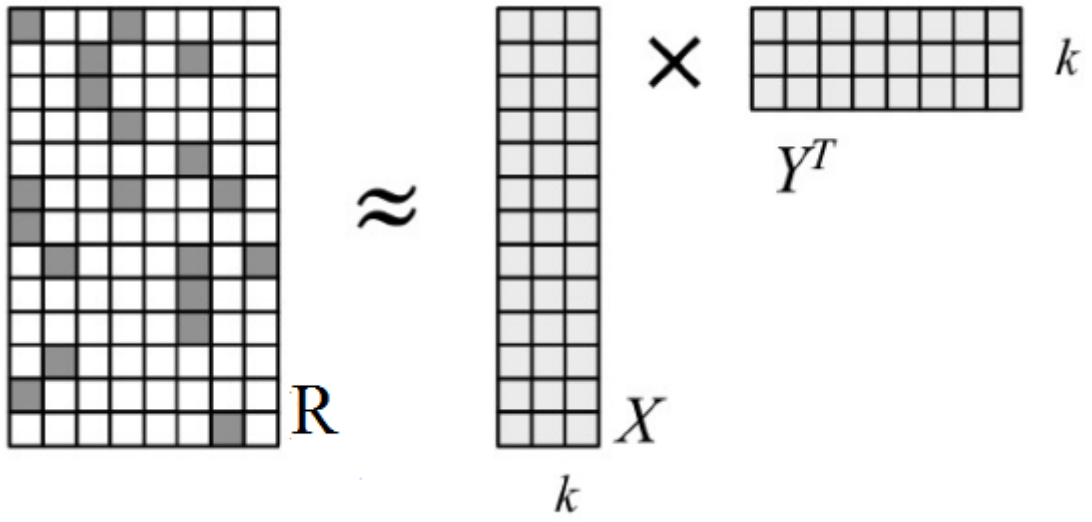

Рис. 2: Схема приближения матрицы предпочтений R с помощью произведения двух матриц.

Приближение ранга k дает значительный выигрыш в занимаемом объеме памяти, потому что позволяет представить матрицу $R^{n\times m}$, используя объем памяти компьютера $O(k(n+m))$, вместо $O(nm)$.

3.2 Модель построения предпочтения SVD

На основе матричных разложений была создана общая модель построения предпочтений – SVD модель (Singular Value Decomposition model):

$$\widehat{r_{ui}} = x_u^T y_i, \tag{4}$$

где x_u − вектор матрицы X скрытых($latent$) предпочтений пользователя, yi − вектор матрицы Y скрытых характеристик объекта. Данная модель обучается путем минимизации квадратичных отклонений $(r_{ui} - x_u^T y_i)^2$, избегая проблемы переобучения с помощью регуляризации вида:

$$\Omega(\theta) = \sum_{u\in U}\|x_u\|^2 + \sum_{i\in I}\|y_i\|^2 \tag{5}$$

Название модели (Singular Value Decomposition) появилось благодаря статье Саймона Фанка, посвященной решению Netflix Prize[9]. Поэтому не стоит связывать название модели с сингулярным разложением, они имеют мало общего. В публикации Саймона Фанка модель была оптимизирована методом стохастического градиентного спуска, который является одной из форм стохастического приближения. Его суть заключается в том, что значение градиента аппроксимируется градиентом целевой функции, вычисленном только на одном элементе обучения. Так, после каждого объекта обучения параметры модели изменяются, что дает значительное преимущество в скорости при работе с большими массивами данных, нежели при стандартном градиентном спуске. Однако, позднее был предложен алгоритм, показавший гораздо большую эффективность. Он получил название Alternating Least Squares (ALS)[16]. В настоящей работе он будет подробнее рассмотрен далее.

4 Модернизация модели

Модель SVD не всегда хорошо работает в рекомендательных системах с неявной обратной связью, поскольку в любой сфере применения есть свои особенности, влияющие на достоверность рекомендаций. Поэтому данная модель нуждается в адаптации под конкретную ситуацию. В данной работе рассматривается модель для рекомендательной системы веб-ресурса букмекерской конторы. Необходимо рассмотреть подробно систему букмекерских ставок.

4.1 Система ставок в букмекерских конторах

Букмекерская контора (букмекер) — это организация в статусе юридического лица, которая проводит пари с игроками с последующей выплатой денежных средств. Букмекерские конторы изучают спрос на спортивные события и определяют условия пари: на конкретный исход матча, количество желтых карточек, забитых или пропущенных голов, нестандартных ситуаций на поле и так далее.

Букмекерская контора не преследует цель выиграть пари как можно чаще, а всего лишь зарабатывает на комиссии**.** Специальные спортивные аналитики, работающие в компании, стремятся к тому, чтобы выстроить такую линию (дать оценку клубам, командам, игрокам), чтобы получившиеся коэффициенты выглядели заманчивыми для игроков с противоположными предпочтениями (кто-то ставит на «Спартак», а кто-то на ЦСКА). Это делается для того, чтобы увеличить оборот по каждому событию (т.е. суммарный объем денежных средств, проставленный всеми игроками на конкретное событие). Другими словами, чем больше денег будет поставлено на событие, независимо от исхода, тем большую прибыль получит компания. Кроме того, количество спортивных событий, на которые заключаются пари, огромно. Их классификация очень широка: по видам спорта, по рангу соревнований, по участникам. Именно этим обусловлена актуальность внедрения рекомендательной системы для онлайн- сервиса. И поэтому ставки

могут рассматриваться как товар, предлагаемый пользователю букмекерской конторой.

Итак, клиентам конторы будут предложены спортивные события из предлагаемого в линии списка. По мнению аналитиков, наиболее подходящим критерием заинтересованности игрока в спортивном мероприятии является сумма денежных средств, поставленная на одного из участников данного события. Поэтому показателем предпочтения пользователем спортивного события был выбран оборот денежных средств игрока по текущему событию. Выбранная характеристика обладает хорошим масштабом в смысле показателя заинтересованности пользователя в различных видах спорта. Например, денежный оборот на футбольные матчи игрока, болеющего за футбольный клуб «Зенит», будет во много раз превышать сумму денежных средств, проставленных на другие виды спорта, этим игроком.

В виду особенностей букмекерского бизнеса, нередко возникают арбитражные ситуации, которые позволяют делать ставки на все противоположные исходы состязания в разных букмекерских конторах и получить доход при любом исходе матча. Такие ситуации называют букмекерскими вилками и, соответственно, людей, регулярно пользующихся ими, принято называть вилочниками. Характер игры вилочника очень отличается от стандартного игрока, потому что вилочник всегда стремиться «поймать» арбитражную ситуацию и сделать ставку на событие, в котором она возникла. Определить заинтересованность вилочника в каком-либо спортивном событии зачастую невозможно, поскольку он делает ставки, руководствуясь не своим интересом, а ситуацией. Поэтому большая часть шума в используемых неявных данных генерируется именно вилочниками. Все букмекерские конторы пытаются бороться с вилочниками, добавляя специальный функционал во внутренние антифрод модули. В рассматриваемой конторе клиенты, которых подозревают в нечестной игре, помечаются специальной меткой. Эту метку можно использовать как

дополнительный вес к степени уверенности в достоверности предпочтения игрока. Таким образом будет фильтроваться большая часть шума.

4.2 Адаптация модели

Для того чтобы получить адекватные результаты модель будет дополнена следующим образом.

Во-первых, необходимо сформировать оценку предпочтения, для этого вводится набор двоичных параметров p_{ui}, который однозначно указывает на предпочтения пользователя. Значения p_{ui} получаются путем бинаризаци значений в матрице предпочтений по следующему правилу:

$$p_{ui} = \begin{cases} 1, & r_{ui} > 0 \\ 0, & r_{ui} = 0 \end{cases}$$

Другими словами, если пользователь u выбирает объект i ($r_{ui} > 0$), то считается, что однозначно нравится данный объект. В то же время, все остальные невыбранные объекты $r_{ui} = 0$ считаются неинтересными пользователю. Таким образом, степень уверенности, которую задают известные нам неявные данные, преобразуется в чистую оценку предпочтения. Набор параметров p_{ui} формирует матрицу P, которая будет использоваться для решения стандартной проблемы «холодного старта».

Проблема «холодного старта» - это проблема, связанная с отсутствием какой-либо информации о новых пользователях или объектах в системе. По определению метода совместной фильтрации можно понять, что новые пользователи и объекты никак не учитываются при построении рекомендаций. В рассматриваемой системе эта проблема решается с помощью матрицы P. Так как все значения p_{ui} матрицы P единообразно определяют интерес пользователя u к объекту i, то по ним можно однозначно определить самых активных игроков и самые популярные спортивные события. Далее, уже вне алгоритма, рекомендовать новым пользователям самые популярные события, а новые объекты – самым активным пользователям.

Во-вторых, чтобы формализовать понятие доверия, и тем самым улучшить точность рекомендаций, необходимо ввести набор параметров, который выражает степень уверенности в актуальности рейтинга, в виде:

$$c_{ui} = 1 + \alpha_u r_{ui}, \quad (6)$$

где α_u −параметр пользователя, определяющийся внутренним антифрод модулем компании с помощью специальной метки. Таким образом регулируется достоверность каждого элемента в матрице предпочтений.

Также вводится параметр $\boldsymbol{\lambda}$, выражающий силу регуляризации, для дополнительного контроля за проблемой переобучения. Его точное значение подбирается с помощью кросс-валидации.

Итак, задача найти векторы x_u для каждого пользователя и векторы y_i для каждого объекта. С учетом вышеописанных особенностей их можно найти путем минимизации следующей функции:

$$min_{x,y} \sum_{u,i} c_{ui} (p_{ui} - x_u^T y_i)^2 + \boldsymbol{\lambda}(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2) \quad (7)$$

Похожая, но более общая задача была описана для системы с явной обратной связью в статье[7]. Ее суть в том, что необходимо инициализировать один из наборов векторов x_u или y_i случайными значениями, и по ним рассчитать значения для второго набора. Далее, с помощью значений второго набора пересчитать первый набор и так далее. Каждый шаг гарантированно снижает квадрат отклонения. Схематично это описывается так:

$$x \to y \to x \to y \to \cdots$$

Такой подход называется методом чередующихся наименьших квадратов (Alternating Least Squares (ALS)). Однако, для системы с неявными данными требуется иной подход, в который можно встроить показатель достоверности рейтинга. В статье [8], на примере данных о просмотрах телешоу, описывается способ, который позволяет не только в явном виде учесть степень достоверности предпочтений пользователя, но и значительно ускорить работу алгоритма.

Для начала вычисляются векторы пользователей x_u. Для этого векторы $y_i \in Y$, где $Y \in R^{m \times k}$ инициализируются случайными значениями(меньше 1). Для каждого пользователя u определяется диагональная матрица C^u размера $m \times m$ с элементами $c_i(u)$ по главной диагонали. Так же выделяется вектор $p(u)$, который содержит оценки предпочтений пользователя u (значения p_{ui}). Путем дифференцирования функции (7) находится выражение для x_u:

$$x_u = (Y^T C^u Y + \boldsymbol{\lambda} E)^{-1} Y^T C^u p(u) \quad (8)$$

В данном выражении слабым местом является произведение $Y^T C^u Y$, так как имеет временную сложность $O(k^2 m)$ для каждого из n пользователей. Эту ситуацию можно разрешить, представив данное произведение в виде:

$$Y^T C^u Y = Y^T Y + Y^T (C^u - E) Y \quad (9)$$

В таком виде выражение разбивается на две части. Первая $Y^T Y$ – не зависит от u и может быть вычислена заранее для всех пользователей. Произведение $Y^T Y$ это матрица размера $k \times k$, его временная сложность вычисления составляет $O(k^2 m_u)$.А вторая часть $Y^T (C^u - E) Y$, в которой можно заметить, что $C^u - E$ содержит m_u ненулевых элементов, то есть m_u – количество объектов, для которых $r_{ui} > 0$, причем $m_u < m$. Точно так же произведение $C^u p(u)$ содержит только m_u ненулевых элементов. Таким образом, временная сложность вычисления x_u равняется $O(k^2 m_u + k^3)$, где $O(k^3)$ – сложность вычисления обратной матрицы $(Y^T C^u Y + \boldsymbol{\lambda} E)^{-1}$. Описанные вычисления проводятся для каждого из n пользователей, соответственно, общая временная сложность вычислений составляет $O(k^2 M + k^3 n)$, где M- общее количество ненулевых элементов $M = \sum_u m_u$.[13]

Вычисление каждого вектора y_i для объектов производится последовательно после расчета x_u соответственно. Принцип точно такой же как и в предыдущем пункте, только в качестве известного параметра берутся вектора $x_u \in X$, где $X \in R^{n \times k}$. И для каждого объекта i определяется диагональная матрица C^i размера $n \times n$ с элементами $c_u(i)$ по главной

диагонали, и, соответственно, вектор $p(i)$, который содержит оценки всех пользователей, выбравших данный объект. Далее рассчитываются векторыy_i аналогично формуле (б):

$$y_i = \left(X^T C^i X + \boldsymbol{\lambda} E\right)^{-1} X^T C^i p(i) \qquad (10)$$

Временная сложность этих вычислений составляет $O(k^2 N + k^3 m)$, где $N = \sum_i n_i$ есть общее количество ненулевых элементов по всем объектам.

После вычисления всех векторов пользователей и объектов формируются рекомендации путем скалярного произведения соответствующих векторов по формуле (4):

$$r_{ui} = x_u^T y_i,$$

где r_{ui} есть прогнозируемое предпочтение объекта i пользователем u.

5 Реализация демонстрационной версии рекомендательной системы

5.1 Методы оценки точности результатов.

В качестве результата работы реализованной системы получаются упорядоченные списки предпочтений для каждого пользователя, отсортированные в соответствии с прогнозом от наиболее интересных объектов к наименее интересным для каждого пользователя.

В качестве метода был выбран способ оценивания AUC-Recall (значение площади под кривой ROC).

ROC-кривая (Receiver Operating Characteristic) – это графическая характеристика бинарного классификатора[15]. Для ее использования в терминах рекомендательной системы необходимо перевести полученные рейтинги в матрице предпочтений в бинарную шкалу: «Рекомендовать» и «Не рекомендовать» . В настоящей работе это проделано следующим образом: так как уже получены упорядоченные списки предпочтений, следовательно можно выбрать несколько наибольших показателей заинтересованности и рекомендовать соответствующие объекты, остальные не рекомендовать. При этом среди полученных результатов выделяются следующие группы, которые можно представить в табличном виде:

Таблица 1

	Положительные	Отрицательные
Положительные	True Positives	False Positives
Отрицательные	False Negatives	True Negatives

True Positives(TP)- рекомендации, верно классифицированные как положительные, то есть предложенные пользователю и выбранные им.
False Positives(FP) - нерелевантные рекомендации, предложенные системой пользователю, их называют ложноположительные результаты или ошибка второго рода.

True Negatives(TN) - рекомендации, верно классифицированные как отрицательные, то есть корректно отфильтрованные как нерелевантные.
False Negatives(FN) – результаты, ошибочно классифицированные системой как нерелевантные, их называют ложноотрицательные результаты или ошибка первого рода.
На основе описанных групп вводятся понятия полноты системы и ее точности. Под полнотой системы понимают отношение всех релевантных предложений к рекомендованным:

$$Recall = \frac{TP}{TP + FP}$$

А точность системы определяют как долю всех предложенных объектов среди релевантных рекомендаций:

$$Precision = \frac{TP}{TP + FN}$$

ROC-кривая иллюстрирует способность рекомендательной системы предлагать релевантные рекомендации. По виду кривой можно судить о способности системы выделить релевантные данные предпочтения пользователей среди шума. ROC-кривая наглядно отражает зависимость полноты системы от ее точности.

ROC-кривая строится следующим образом:

- все рекомендации упорядочиваются по рейтингу
- для каждой позиции в этом списке рассчитываются полнота и выпадение
- строиться график «полнота-выпадение»

Где под выпадением доля всех ложноположительных рекомендаций среди всех предложенных пользователю:

$$Fallout = \frac{FP}{TP + FN}$$

На рисунке 2 приведен пример ROC-кривых. Диагональная прямая характеризует рекомендательную систему, предлагающую произвольные

рекомендации. Кривые, расположенные над диагональю, указывают на более эффективные рекомендации, чем произвольные. Вычисление площади под кривой позволяет дать количественную оценку производительности и эффективности рекомендательной системы. Такая оценка называется AUC (area under curve). У идеальной рекомендательной системы AUC = 1.

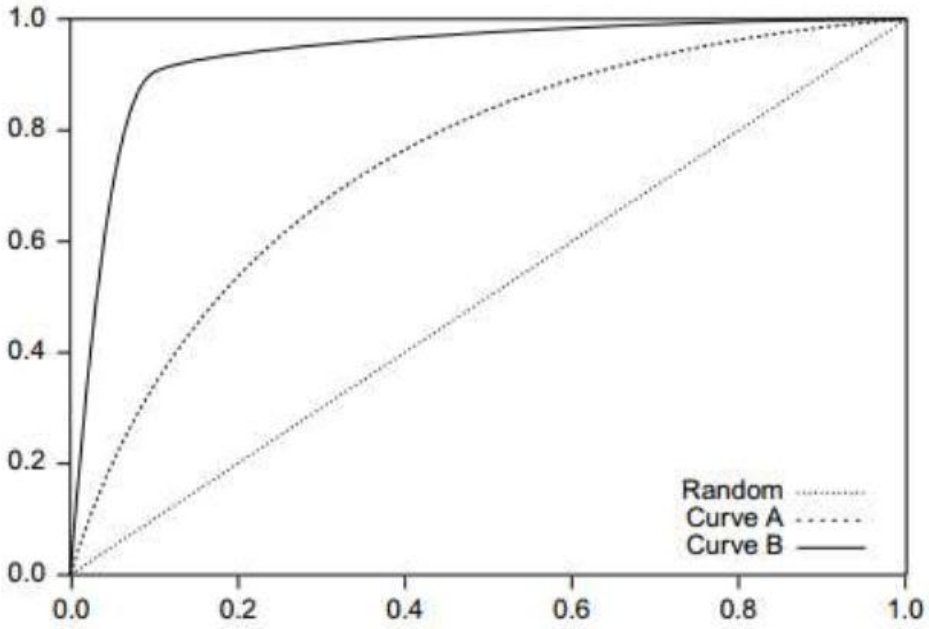

Рис. 3: Примеры ROC-кривых

Важно понимать, что в виду использования неявных данных невозможно получить надежную обратную связь относительно того, какие из предоставленных рекомендаций оказались действительно интересны пользователю. Потому что для объективной оценки необходим длительный сбор данных об очень большом количестве пользователей, сравнимом с первоначальным набором. В связи с этим, без реализации непосредственно на веб-ресурсе компании с помощью стандартных методов можно дать лишь предварительную оценку.

5.2 Полученные результаты. Примеры.

Предложенная модель была реализована для двух наборов данных. Они будут описаны отдельно.

В первом наборе были сгенерированы данные по ставкам 51642 игроков на спортивные события футбола, сгруппированных в соответствии с расположением в базе данных конторы, за период 3 месяца. На рисунке 4 приведена часть сформированной матрицы предпочтений.

Spanish	German	Tov_match	Champions League	Greec	Scotland	France	Portugal	Mexico
850000	300000	540212	1695000	0	0	200000	100000	0
634343	375339	86862	1129386	24691	33752	702298	73625	300
653330	391006	318949	1111896	0	0	1085418	0	0
447942	601743	45600	1531097	0	2933	384805	1277	0
0	0	0	750000	0	0	0	0	0

Рис. 4: Фрагмент матрицы предпочтений

Для полученного набора данных производился расчет модели с разными параметрами силы регуляризации $\boldsymbol{\lambda}$ (в диапазоне от 0,1 до 1) и количеством скрытых характеристик пользователей и объектов k. После чего для набора аккаунтов с реальной историей ставок было проведено сравнение предложенных рекомендаций и дальнейшего стиля игры пользователей, так чтобы вычислить полноту, точность и выпадение для системы. Далее по полученным значениям для каждого пользователя вычислялся параметр AUC и затем, для получения общей оценки, рассчитанные параметры усреднялись. В таблице 2 приведены полученные результаты для различных параметров. Как видно из полученных результатов величина регуляризации сильнее влияет на точность предсказания предпочтений.

Таблица 2

	λ=1; k=10	λ=0,1; k=7	λ=1; k=14	λ=0,5; k=14
User1	0,854	0,732	0,822	0,75
User2	0,895	0,789	0,831	0,802
User3	0,795	0,721	0,808	0,799
User4	0,666	0,701	0,678	0,611
User5	0,708	0,731	0,654	0,689
User6	0,654	0,625	0,677	0,686
User7	0,863	0,813	0,799	0,768
User8	0,567	0,511	0,536	0,555
User9	0,712	0,698	0,733	0,745
User10	0,701	0,655	0,722	0,861
Average	0,74	0,6976	0,726	0,7266

При этом ROC-кривая для большинства пользователей принимает вид как показано на рисунке 5.

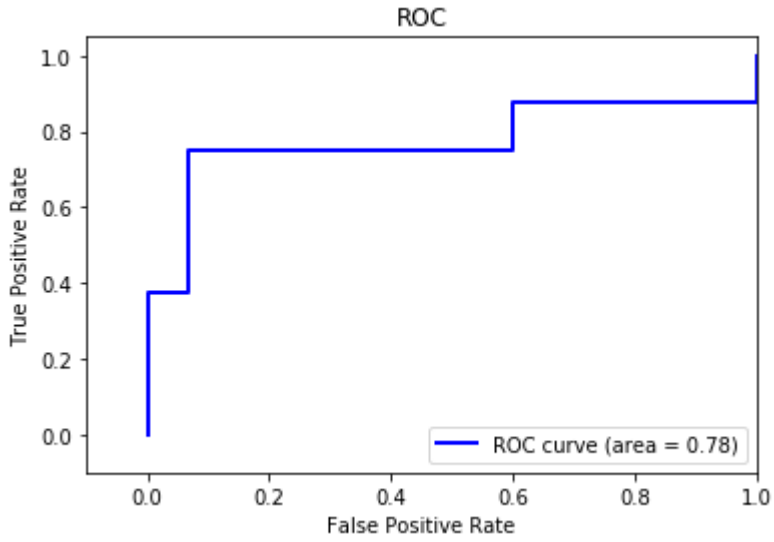

Рис. 5: Пример ROC-кривой для пользователя из первого набора данных.

Полученные результаты для описанного набора данных , в целом, можно оценить как удовлетворитьльные, потому что среднее значение AUC<0.75.

Для повышения точности рекомендаций было принято решение изменить масштаб анализируемых данных. Тогда был сформирован новый

набор данных за период 7 дней, в который попала информция о ставках 4569 пользователей, но уже на различные виды спорта, предлагаемые в линии. Для данного набора данных параметр AUC вычислялся совершенно аналогично. При этом были получены следующие результаты:

Таблица 3

	λ=1; k=10	λ=0,1; k=7	λ=1; k=14	λ=0,5; k=14
User1	0,90524	0,77592	0,87132	0,795
User2	0,90395	0,79689	0,83931	0,81002
User3	0,81885	0,74263	0,83224	0,82297
User4	0,7659	0,80615	0,7797	0,70265
User5	0,72216	0,74562	0,66708	0,70278
User6	0,72594	0,69375	0,75147	0,76146
User7	0,85437	0,80487	0,79101	0,76032
User8	0,7938	0,7154	0,7504	0,777
User9	0,7832	0,7678	0,8063	0,8195
User10	0,74306	0,6943	0,76532	0,91266
Average	0,801647	0,754333	0,785415	0,786436

Показатели рекомендаций для пользователей заметно улучшились, теперь ROC-кривая для подавляющего большинства пользователей имеет вид:

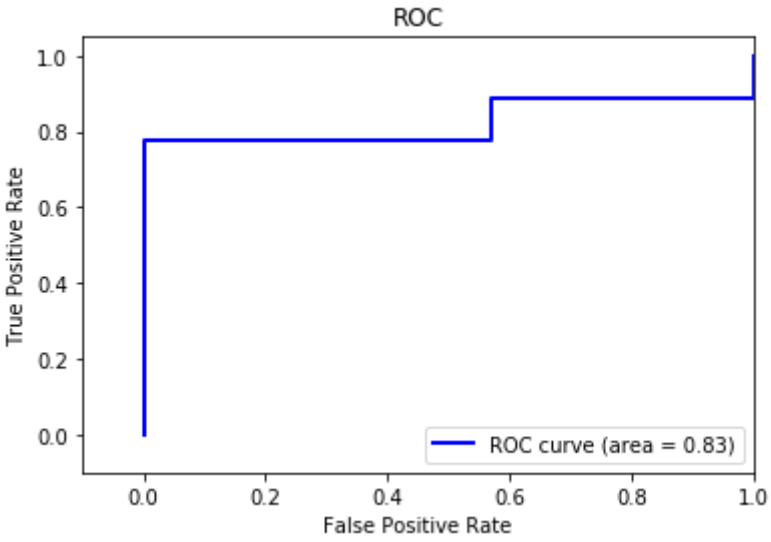

Рис. 6: Пример ROC-кривой для пользователя из второго набора данных.

Факт такого значительного улучшения точности рекомендаций можно объяснить тем, что за выбранный промежуток (7 дней) было проведено количественно меньше спортивных мероприятий (в рамках одного вида спорта), чем за 3 месяца, при таком же количестве видов спорта. Соответственно, увеличивается масштаб оборота по каждому событию, что значительно облегчает фильтрацию шума в данных.

6 Заключение

В рамках настоящей работы были решены следующие задачи:

- Построена модель предсказания предпочтений пользователей для рекомендательной системы онлайн сервиса букмекерской конторы.
- Реализована демонстрационная версия описаной системы, по результатам работы которой были получены адекватные результаты.
- Проведена работа по улучшению качества рекомендаций с учетом проблем неявной обратной связи и масшабирования данных.

Рекомендательная система показала себя как мощный инструмент, который потенциально способен улучшить показатели букмекерского бизнеса за счет увеличения вовлеченности клиентов в ставки. Кроме того, полученные результаты для разных наборов данных показали, что реализованная система имеет широкие перспективы дальнейшего развития и улучшения в рамках букмекерской конторы, за счет возможного дополнения модели и совершенствования антифрод модуля.

Список литературы.

1. Ромов П.А. *Низкоранговые приближения в моделировании данных*[Электронный документ] http://www.machinelearning.ru/wiki/images/9/93/2014_517_RomovPA.pdf

2. Гончаров М. *Data Mining: Рекомендательные системы* [Электронный документ] (https://habr.com/ru/company/lanit/blog/420499/)

3. Yifan Hu, Yehuda Koren, Chris Volinsky. *Collaborative Filtering for Implicit Feedback Datasets.*

4. Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. *In Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001

5. Jun Wang, Arjen P De Vries, and Marcel JT Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. *In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508. ACM, 2006

6. К. В. Воронцов. *Математические методы обучения по прецедентам (теория обучения машин)*. Москва, 2011

7. Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005

8. Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. *In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008

9. Gregory Piatetsky. *Interview with simon funk*. ACM SIGKDD Explorations Newsletter, 9(1):38–40, 2007

10. Francesco Ricci, Lior Rokach and Bracha Shapira. *Recommender Systems Handbook*, 2014

11. Е.С. Соколов. *Семинары по матричным разложениям*. Москва, 2015

12. R. Bell and Y. Koren, *Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpola-tion Weights*, IEEE International Conference on DataMining (ICDM'07), pp. 43–52, 2007.

13. V. Kohler. *ALS Implicit Collaborative Filtering* [Электронный документ] https://medium.com/radon-dev/als-implicit-collaborative-filtering-5ed653ba39fe

14. J. L. Herlocker, J. A. Konstan, A. Borchers and John Riedl, *An Algorithmic Framework for Performing Collaborative Filtering*, Proc. 22nd ACM SIGIR Conference on Information Retrieval, pp. 230–237, 1999.

15. К. В. Воронцов. *Кривая ошибок* [Электронный документ] http://www.machinelearning.ru/wiki/index.php?title=ROC-%D0%BA%D1%80%D0%B8%D0%B2%D0%B0%D1%8F

16. T. Hofmann, *Latent Semantic Models for Collabora-tive Filtering*, ACM Transactions on Information Sys-tems22(2004), 89–115.

17. R. Salakhutdinov and A. Mnih, *Probabilistic MatrixFactorization*, Advances in Neural Information Pro-cessing Systems 20 (NIPS'07), pp. 1257–1264, 2008.

Приложение

Разработка демонстрационной версии рекомендательной системы производилась на языке Python 3. Для работы использовалась интерактивная оболочка IPython, инструмент разработки Jupyter Notebook 5.7.8. Такой выбор обоснован наличием свободного доступа к продукту, а также удобством в его эксплуатации.

Код можно символически разбить на несколько блоков. В первом блоке происходит обработка полученных из базы данных и преобразование их в разреженную матрицу.

```
import random
import pandas as pd
import numpy as np

import scipy.sparse as sparse
from scipy.sparse.linalg import spsolve
from sklearn.preprocessing import MinMaxScaler


ratings_sdf = pd.read_csv( 'Turnover.csv', sep=';').to_sparse(fill_value=0.0)
head_rows = 5

ratings_sdf = ratings_sdf.loc[ratings_sdf.All != 0]
Users_acc = ratings_sdf['Account_id']
#Users_acc['Account_id'] = Users_acc.Account_id.astype(str)

print(ratings_sdf[:head_rows])
def all_turn(r):
    if sum(r)>0:
        return r/sum(r)
    else:
        return r

def sparse_df_to_csr(df):
    return sparse.csr_matrix(df.to_coo())

ratings_sdf.drop('Account_id', axis=1, inplace=True)
ratings_sdf.drop('All', axis=1, inplace=True)
ratings_sdf=ratings_sdf.apply(all_turn, axis=1)

ratings_sdf = ratings_sdf.dropna()

print(ratings_sdf[:head_rows])

data = sparse_df_to_csr(ratings_sdf)
print(data.shape)
```

Во втором блоке происходит непосредственно вычисление векторов скрытых свойств пользователей и объектов

```
def nonzeros(m, row):
    for index in range(m.indptr[row], m.indptr[row+1]):
        yield m.indices[index], m.data[index]

def implicit_als_cg(Cui, features=10, iterations=20, lambda_val=0.5):
    user_size, item_size = Cui.shape

    X = np.random.rand(user_size, features) * 0.1
    Y = np.random.rand(item_size, features) * 0.1

    Cui, Ciu = Cui.tocsr(), Cui.T.tocsr()
    for iteration in range(iterations):
        print('{}th iteration is completed**********************'.format(iteration))
        least_squares_cg(Cui, X, Y, lambda_val)
        least_squares_cg(Ciu, Y, X, lambda_val)

    return sparse.csr_matrix(X), sparse.csr_matrix(Y)


def least_squares_cg(Cui, X, Y, lambda_val, cg_steps=3):
    users, features = X.shape

    YtY = Y.T.dot(Y) + lambda_val * np.eye(features)

    for u in range(users):

        x = X[u]
        r = -YtY.dot(x)

        for i, confidence in nonzeros(Cui, u):
            r += (confidence - (confidence - 1) * Y[i].dot(x)) * Y[i]

        p = r.copy()
        rsold = r.dot(r)

        for it in range(cg_steps):
            Ap = YtY.dot(p)
            for i, confidence in nonzeros(Cui, u):
                Ap += (confidence - 1) * Y[i].dot(p) * Y[i]

            alpha = rsold / p.dot(Ap)
            x += alpha * p
            r -= alpha * Ap

            rsnew = r.dot(r)
            p = r + (rsnew / rsold) * p
            rsold = rsnew

        X[u] = x

conf_data = (data * alpha_val).astype('float64')
user_vecs, item_vecs = implicit_als_cg(conf_data, iterations=20, features=10)
```

```
print('done')
```

В третьем блоке в явном виде рассчитываются рекомендации для конкретного пользователя.

```
user_id = 495600

t_idx, = Users_acc[Users_acc == user_id].index

consumed_idx = ratings_sdf.loc[t_idx, :].nonzero()
for ii in consumed_idx:
   print(events[ii])
#то, на что ставил пользователь убираем ниже, чтоб рекомендовать то, на что он не ставил
user_interactions = data[t_idx,:].toarray()
user_interactions = user_interactions.reshape(-1)+1  #Reshape to turn into 1D array
user_interactions[user_interactions > 1] = 0.8

print('user preferense:', user_interactions)
rec_vector = user_vecs[t_idx,:].dot(item_vecs.T).toarray()

#если нужно решейпить
min_max = MinMaxScaler()
rec_vector_scaled = min_max.fit_transform(rec_vector.reshape(-1,1))[:,0]
recommend_vector = user_interactions*rec_vector_scaled
#print('raiting:', recommend_vector)
num_items=14
item_idx = np.argsort(recommend_vector)[::-1][:num_items]
sports = []
scores = []
for idx in item_idx:
      sports.append(events[idx])
      scores.append(recommend_vector[idx])
recommendations = pd.DataFrame({'sport': sports, 'score': scores})

print(recommendations)
```

И в четвертом блоке рассчитывается показатель AUC для текущего пользователя и визуализируется ROC-кривая.

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import roc_curve, auc

score =results['d']
y = np.array([1,1,1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0])

fpr, tpr, Thresholds = roc_curve(y, score)
roc_auc = auc(fpr, tpr)
print('AUC = ', roc_auc)
lw =2
plt.title('ROC')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.xlim([-0.1, 1.0])
plt.ylim([-0.1, 1.05])
plt.plot(fpr, tpr, color='blue',
```

```
        lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
plt.legend(loc="lower right")
plt.show()
```