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 прикладной математики

КУРСОВАЯ РАБОТА

РАЗРАБОТКА РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ В СРЕДЕ PYTHON

Работу выполнил _____ Г.К. Садунян
(подпись)

Направление 09.03.03 Прикладная информатика

Направленность Прикладная информатика в экономике

Научный руководитель,
д-р физ.-мат. наук, проф. _____ Е.Н. Калайдин
(подпись)

Нормоконтролер,
преподаватель _____ Е.В. Горбачева
(подпись)

Краснодар
2024

РЕФЕРАТ

Курсовая работа 24 с., 14 рис., 3 табл., 7 источн.

РЕКОМЕНДАТЕЛЬНЫЕ СИСТЕМЫ, PYTHON, ДАННЫЕ, ФИЛЬТРАЦИЯ

В данной работе была изучена теоретическая сторона реализации алгоритмов для рекомендательной системы, рассмотрены библиотеки Python и приведены примеры использования системы в различных областях.

Цель данной курсовой работы – познакомиться с рекомендательными системами и инструментами для их реализации.

Задачи:

– изучить теоретические аспекты рекомендательной системы и обработки данных;

– проанализировать имеющиеся программные средства для реализации рекомендательных систем;

– изучить библиотеки для реализации рекомендательных систем в выбранной программной среде.

Объект исследования – рекомендательные системы.

Предмет исследования – типы рекомендательных систем и инструменты для работы с рекомендательными системами.

Итог проделанной работы – реализации рекомендательной системы в среде Python.

СОДЕРЖАНИЕ

Введение 4
1. Обзор задачи предсказания поведения или потребления 5
1.1 Данные и алгоритмы для рекомендательных систем 6
1.2 Коллаборативная фильтрация 7
1.3 Контентные рекомендации 9
1.4 Гибридные системы 11
2 Практическая часть 13
2.1 Выбор средств разработки и подготовка данных 13
2.2 Подбор рекомендаций 17
2.3 Оценка качества рекомендаций 21
Заключение 23
Список использованных источников 24

ВВЕДЕНИЕ

В современном цифровом мире системы рекомендаций играют ключевую роль в улучшении пользовательского опыта. Они используются в различных областях, таких как электронная коммерция, потоковые платформы, социальные сети и образовательные системы, для предоставления пользователям персонализированных рекомендаций.

Python является одним из наиболее популярных языков программирования для разработки рекомендательных систем благодаря своей простоте, богатству библиотек и инструментов, таких как NumPy, pandas, scikit-learn. Использование этих инструментов позволяет реализовывать алгоритмы различной сложности, от простых эвристик до глубоких нейронных сетей.[6]

Целью данной курсовой работы является разработка рекомендательной системы, которая сможет предложить пользователям персонализированные рекомендации на основе их предпочтений и исторических данных. В процессе выполнения работы предполагается изучение теоретических основ рекомендательных систем, анализ алгоритмов и их реализация в среде Python.

Пять задач работы:

1) изучить основы и теоретические аспекты рекомендательных систем;
2) ознакомиться с существующими алгоритмами рекомендаций;
3) подготовить данные для построения системы рекомендаций;
4) разработать рекомендательную систему в среде Python, реализовав один алгоритм;
5) провести тестирование системы и оценить её эффективность.

1 Обзор задачи предсказания поведения или потребления

Рекомендательные системы — программные средства, которые пытаются предсказать какие объекты (фильмы, музыка, книги, новости, веб-сайты и т. д.) будут интересны пользователю, если имеется определенная информация о его предпочтениях. Рекомендации формируются отдельно для каждого человека на основе прошлой активности. Кроме того, имеет значение и поведение остальных пользователей системы [1]. Существует два основных подхода к построению рекомендаций:

– На основе коллаборативной фильтрации (англ. collaborative filtering), которая использует информацию о поведении пользователей в прошлом, например, перечень покупок или оценок объектов, сделанных ранее на сайте интернет-магазина пользователями из той же группы интересов.[1]

– На основе фильтрации содержимого (англ. content-based information filtering), при этом в системе содержатся профили, включающие личную информацию пользователей: социальный статус, возраст, место проживания, род деятельности, а также характеристики, выражающие интерес пользователя к объекту; профили объектов интереса включают характеристики, интересующие пользователя [1].

1.1 Данные и алгоритмы для рекомендательных систем

Пять шагов для построения рекомендательной системы:

1) Сбор данных. Данные для анализа можно разделить на две основные категории. Явные данные включают предпочтения, которые пользователь явно указывает, такие как оценки фильмов, лайки и отзывы. Неявные данные собираются из действий пользователя, например, из просмотра контента, покупок или времени, проведенного на странице;

2) Оценка похожести. Для определения степени схожести объектов или пользователей используются различные подходы. Среди них популярны косинусное сходство, корреляция Пирсона и метрики расстояний, такие как Евклидово расстояние;

3) Построение профиля пользователя. Профиль пользователя формируется на основе собранной информации. Он включает личные предпочтения, например, любимые жанры фильмов, и историю взаимодействий, такие как купленные товары или просмотренные видео;

4) Алгоритмы и методы машинного обучения. Современные системы используют машинное обучение для предсказания предпочтений. Наиболее востребованными являются:

- линейные модели, включая линейную регрессию;
- нейронные сети, применяемые для анализа сложных данных;
- факторизация матриц, позволяющая находить скрытые зависимости между пользователями и объектами;
- графовые подходы, представляющие данные в виде графов и анализирующие связи между пользователями и объектами.

5) Персонализация. Для обеспечения индивидуального подхода рекомендательные системы учитывают:

- контекст, например местоположение, время суток и устройство;
- динамическую адаптацию, позволяющую подстраиваться под изменяющиеся предпочтения пользователя;

– качество рекомендаций, которое оценивается с помощью метрик, отражающих точность работы системы.

1.2 Коллаборативная фильтрация

Рекомендация на основе поведения пользователей. Идея заключается в том, что если пользователи вели себя схоже в прошлом (например, ставили одинаковые оценки товарам или выбирали схожие категории), то их предпочтения в будущем также могут совпадать.[5]

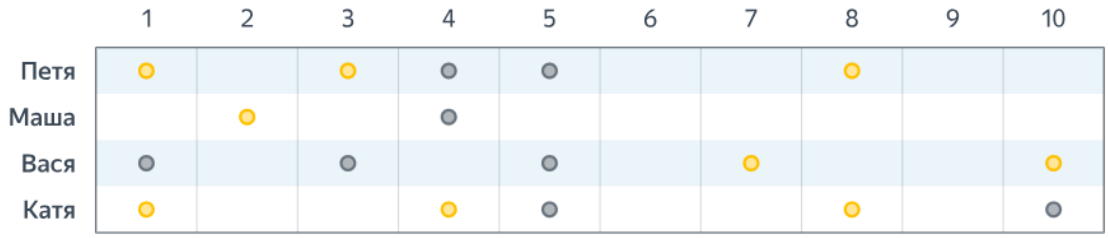

Рисунок 1 – Таблица рекомендаций

Рассмотрим матрицу взаимодействий пользователя, приведённую выше. Что можно порекомендовать Кате, исходя из исторических данных? Можно заметить, что взаимодействия Кати похожи на взаимодействия Пети (так как они оба «лайкали» объекты 1 и 8). Иными словами, их интересы в чём-то похожи, поэтому Кате можно порекомендовать, например, объект 3 (так как он понравился Пете). Можно проделать аналогичное упражнение с Петей и сделать вывод, что ему не стоит рекомендовать объект 10.

Можно решать и транспонированную задачу: для «лайкнутого» пользователем объекта искать похожие, то есть те, которые пользователи достаточно часто лайкали вместе с ним. Например, объекты 1 и 8 похожи друг на друга, так как их лайкали одни и те же пользователи, и точно так же похожи 1 и 3.

Коллаборативная фильтрация включает несколько подходов, каждый из которых имеет свои особенности:

– Фильтрация на основе пользователей (User-Based Collaborative Filtering). Этот метод ищет похожих пользователей, анализируя их действия или оценки. Например, если два пользователя часто ставят одинаковые оценки на фильмы, система может рекомендовать одному из них фильм, который посмотрел другой;

– Фильтрация на основе предметов (Item-Based Collaborative Filtering). Здесь система определяет схожесть между объектами, такими как товары или фильмы, на основе оценок или взаимодействий пользователей. Например, если два фильма часто оцениваются одинаково, пользователь, посмотревший один из них, вероятно, захочет посмотреть и второй;

– Матрица взаимодействий (Matrix Factorization). Данные представляются в виде матрицы (пользователи × предметы), а для анализа используются методы, такие как SVD (сингулярное разложение), позволяющие выявить скрытые зависимости между пользователями и объектами.

Преимущества коллаборативной фильтрации:

– Отсутствие необходимости в данных о самих товарах (например, описаниях или жанрах), достаточно информации о взаимодействиях пользователей;

– Возможность предоставления неожиданных, неочевидных рекомендаций.

Несмотря на преимущества, у коллаборативной фильтрации есть ограничения:

– Проблема холодного старта. Отсутствие данных делает невозможным рекомендации для новых пользователей или товаров.

– Разреженность данных. Если данные о взаимодействиях редки, рекомендации становятся менее точными.

– Масштабируемость. Обработка больших массивов данных требует значительных вычислительных ресурсов.

1.3 Контентные рекомендации

Контентные рекомендации — это метод рекомендаций, который основывается на анализе характеристик (атрибутов) объектов, чтобы предлагать пользователю товары или контент, похожие на то, что он уже предпочитал.[6] Чаще всего используются там, где есть детализированные данные о продуктах, например, в интернет-магазинах, где товары имеют подробные описания и характеристики.

Основные принципы работы контентных рекомендаций:

- Сравнение характеристик объектов: для каждого объекта (фильма, товара, книги и т. д.) определяются его атрибуты, такие как жанр, описание, цена, цвет, бренд и т. д.;
- Профиль пользователя: Система создает профиль пользователя, используя его предыдущие взаимодействия, чтобы понять, какие характеристики ему нравятся;
- Схожесть объектов: сравниваются атрибуты объектов, чтобы находить те, которые максимально соответствуют предпочтениям пользователя.

Пример работы контентных рекомендаций: В стриминговом сервисе пользователь смотрел фильмы, жанры которых — «комедия» и «фантастика». Система проанализирует атрибуты этих фильмов и предложит ему другие фильмы с аналогичными жанрами.

Преимущества контентных рекомендаций:

- Персонализация: Рекомендации базируются на уникальных предпочтениях каждого пользователя.

– Работа с новыми пользователями: можно сразу делать рекомендации, используя атрибуты контента, даже если у пользователя нет истории взаимодействий.

– Контролируемость: Система даёт объяснимые рекомендации (например, "мы рекомендуем это, потому что вам нравятся комедии").

Недостатки контентных рекомендаций:

– Ограниченность разнообразия: Система может предлагать слишком похожие объекты, не учитывая, что пользователь может захотеть попробовать что-то новое (проблема «узкого тоннеля»).

– Требуются качественные данные: для создания профилей объектов нужны точные и полные метаданные.

– Игнорирование коллективного мнения: Рекомендации не учитывают популярность или предпочтения других пользователей.

Сравнение с коллаборативной фильтрацией:

Таблица 1 – Сравнение рекомендательных систем

	Контентные рекомендации	Коллаборативная фильтрация
Данные	Метаданные объектов	Взаимодействие пользователей
Проблема нового объекта	Не возникает, если есть метаданные	Объекту сложно получить рекомендации
Проблема нового пользователя	Требует минимальных данных о действиях	Существенная сложность
Диверсификация	Меньше (узкий выбор)	Больше (учёт разных предпочтений)

1.4 Гибридные системы

Гибридный метод — это подход к созданию рекомендательных систем, который сочетает в себе несколько различных методов (например, коллаборативную фильтрацию, контентные рекомендации и другие), чтобы устранить недостатки каждого из них и повысить точность и релевантность рекомендаций.[7]

Гибридные методы разрабатываются для преодоления ограничений отдельных подходов. Например, коллаборативная фильтрация сталкивается с проблемой холодного старта, а контентные рекомендации ограничиваются узким выбором объектов. Совмещение методов улучшает качество рекомендаций, учитывая больше факторов и скрытых закономерностей. Гибридные системы также обладают высокой гибкостью, что позволяет адаптироваться к различным сценариям и данным.

Объединение подходов может осуществляться по-разному. Взвешенное объединение использует результаты разных методов с определёнными весами, например, 70% рекомендаций от коллаборативной фильтрации и 30% от контентного подхода. Последовательное применение методов позволяет сначала сократить выборку одним подходом, а затем уточнить результаты другим. Например, контентная фильтрация формирует список похожих объектов, а коллаборативная фильтрация выбирает самые популярные из них.

В некоторых случаях применяется метод переключения, где система выбирает подходящий метод в зависимости от ситуации, например, контентный метод для новых пользователей и коллаборативный для опытных. Ещё одним вариантом является обогащение данных, когда один метод помогает улучшить данные для последующей обработки другим. Например, контентный метод создаёт профиль пользователя, который используется в коллаборативной фильтрации.

Современные гибридные системы активно используют машинное обучение. Разные методы становятся входными данными для моделей, таких как градиентный бустинг или нейронные сети, которые обучаются предсказывать релевантные рекомендации.

Ярким примером является Netflix, где комбинируются контентные рекомендации (учитывающие жанры, режиссёров, актёров) и коллаборативная фильтрация (анализ действий пользователей, таких как оценки и просмотры). Итоговые рекомендации формируются на основе сочетания этих данных, что позволяет достигать высокой точности.

Преимущества и недостатки гибридных методов: Гибридные системы обеспечивают повышенную точность, универсальность и лучше справляются с проблемой холодного старта, что делает их особенно эффективными в работе с новыми пользователями и объектами. Однако такие системы сложны в реализации, требуют значительных вычислительных ресурсов и правильного выбора комбинации методов, чтобы избежать конфликта между ними.

Гибридные методы находят широкое применение в крупных платформах, таких как Amazon, YouTube и Spotify, где разнообразие данных и предпочтений пользователей играет ключевую роль.

2 Практическая часть

2.1 Выбор средств разработки и подготовка данных

Для практической части определю рекомендации на основе User-Based коллаборативной фильтрации. Буду использовать набор данных с оценками пользователей для различных фильмов [2]. Мне понадобятся библиотеки NumPy, pandas, math и sklearn. Запишу набор данных “ratings.csv” в переменную df (dataframe). DataFrame - двумерная структура данных в библиотеке pandas.

```
import pandas as pd
import numpy as np
from sklearn import model_selection as cv
from sklearn.metrics import mean_squared_error
from math import sqrt
from scipy.spatial import distance
from  sklearn.metrics.pairwise import pairwise_distances
```

Рисунок 2 – Используемые библиотеки

Определю количество строк и столбцов, подсчитаю число пользователей и число фильмов, и выведу первые 5 строк моей структуры для наглядности:

```
df = pd.read_csv('ratings.csv')
df = df.drop(['timestamp'], axis = 1)
print('Количество строк и столбцов: {}'.format(df.shape), "\n")

n_users = len(df['userId'].unique())
n_movies = len(df['movieId'].unique())
print("Число пользователей:", n_users,"\n", "Число фильмов:", n_movies, "\n")
df.head()
```

Рисунок 3 – Чтение данных из файла

```
Количество строк и столбцов: (99983, 3)

Число пользователей: 702
 Число фильмов: 8227
```

	userId	movieId	rating
0	1	2	3.5
1	1	29	3.5
2	1	32	3.5
3	1	47	3.5
4	1	50	3.5

Рисунок 4 – Скриншот вывода

Пояснение столбцов в таблице:

– userId – уникальный идентификатор каждого пользователя;

– movieId – уникальный идентификатор каждого фильма;

– rating – оценка, которую поставил пользователь фильму.

Как видим, в структуре представлено 702 уникальных пользователя и 8227 уникальных фильмов, а общее количество записей – 99 983.

Разделю весь набор данных на две выборки: тренировочную и тестовую. Первая будет использоваться для обучения, а на второй будет измеряться качество предсказанных оценок. Разделить набор можно при помощи функции train_test_split из модуля scikit-learn:

```
train_data, test_data = cv.train_test_split(df, test_size=0.2)

print('Тренировочная выборка: {}'.format(train_data.shape))
print('Тестовая выборка: {}'.format(test_data.shape))
```

```
Тренировочная выборка: (79986, 3)
Тестовая выборка: (19997, 3)
```

Рисунок 5 – Разделение на тренировочную и тестовую выборки

Итого возьму 79 986 записей в тренировочную выборку и 19 997 в тестовую. Сформирую матрицы значений размера для обучающего и тестового наборов таким образом, чтобы элемент в ячейке [i, j] отражал оценку i-го пользователя j-му фильму. На рисунке (8) представлен вывод части тестовой матрицы. Ноль в матрице означает, что оценка отсутствует. Когда в большой матрице много нулевых значений и мало данных она называется разреженной. Излишне раздутые размеры при малом количестве данных сильно замедляют вычисления, однако на мои данные это абсолютно не влияет и вычисления не ломает, поэтому оставлю как есть:

```
train_data_matrix = train_data.pivot(index='userId', columns='movieId', values='rating').fillna(0).values
test_data_matrix = test_data.pivot(index='userId', columns='movieId', values='rating').fillna(0).values

print(test_data_matrix)
```

Рисунок 6 – Создание матриц

```
[[0.  0.  0.  ... 0.  0.  0. ]
 [0.  0.  4.  ... 0.  0.  0. ]
 [0.  0.  0.  ... 0.  0.  0. ]
 ...
 [0.  0.  0.  ... 0.  0.  0. ]
 [0.  0.  0.  ... 0.  0.  0. ]
 [3.5 0.  0.  ... 0.  0.  0. ]]
```

Рисунок 7 – Вывод части тестовой матрицы

Один из важных моментов в коллаборативной фильтрации — найти похожих пользователей для User-Based и похожие объекты (в нашем случае фильмы) для Item-Based коллаборативной фильтрации. Для этого существуют различные подходы. Один из них — использовать косинусное расстояние между векторами, описывающими пользователей и объекты. В модуле scikit-learn существует готовая функция.

```
user_similarity = pairwise_distances(train_data_matrix, metric='cosine')
```

Рисунок 8 – Косинусное расстояние

```
[[0.         0.8768216  0.75174074 ... 0.83498451 0.95143169 0.83972267]
 [0.8768216  0.         0.81318866 ... 0.85428699 0.96086753 0.85111879]
 [0.75174074 0.81318866 0.         ... 0.7615284  0.97090613 0.7669648 ]
 ...
 [0.83498451 0.85428699 0.7615284  ... 0.         0.95230082 0.80790213]
 [0.95143169 0.96086753 0.97090613 ... 0.95230082 0.         0.95157217]
 [0.83972267 0.85111879 0.7669648  ... 0.80790213 0.95157217 0.        ]]
```

Рисунок 9 – Матрица расстояний между пользователями

user_similarity[i][j] представляет собой косинусное расстояние между i-й и j-й строками. Косинусное расстояние можно интерпретировать как меру схожести: чем больше сходство между пользователями или фильмами, тем меньше значение косинусного расстояния.

2.2 Подбор рекомендаций

В примере рассмотрю коллаборативную фильтрацию, основанную на похожести пользователей. Простейшая рекомендательная система считает предсказанную оценку пользователя u фильму i по формуле (1):

$$r_{u,i} = \frac{\sum_{u' \in U} r_{u',i}}{N} \quad (1)$$

где

N — количество пользователей, похожих на пользователя u;

U — множество из N похожих пользователей;

u' — пользователь, похожий на пользователя u (из множества U);

r (u', i) — оценка пользователя u' фильму i;

r (u, i) — предсказанная оценка фильма i.

По этой формуле предсказываемая оценка фильму i пользователя u равняется средней оценке фильма i от N пользователей, наиболее похожих на пользователя u.

Рассмотрим небольшой пример. У нас имеется набор оценок пользователей следующего вида:

Таблица 2 – Пример

	Фильм1	Фильм2	Фильм3	Фильм4	Фильм5
Анна	5	5	5	0	0
Вова	4	1	0	5	3
Инна	1	0	0	5	0
Иван	5	0	5	0	4

Как говорил выше, для меры близости будем использовать косинусное расстояние:

```
demo_data = [[5,5,5,0,0], [4,1,0,5,3], [1,0,0,5,0], [5,0,5,0,4]]
pairwise_distances(demo_data, metric='cosine')
```

Рисунок 10 – Вычисление расстояния для примера

```
# array([[ 0.        ,  0.59577396,  0.8867723 ,  0.28933095],
#        [ 0.59577396,  0.        ,  0.2036092 ,  0.4484398 ],
#        [ 0.8867723 ,  0.2036092 ,  0.        ,  0.87929886],
#        [ 0.28933095,  0.4484398 ,  0.87929886,  0.        ]])
```

Рисунок 11 – Скриншот вывода

В полученной матрице число в ячейке [i, j] отражает похожесть пользователя i и j. В нашем примере число 0.59577396 в ячейке [0, 1] — косинусное расстояние между оценками Анны и Вовы.

Допустим, что N равно двум. Двумя наиболее похожими на Анну пользователями будут Иван (расстояние равно 0.2893) и Вова (расстояние равно 0.5957); для Вовы — Инна и Иван (расстояния 0.2036 и 0.4484 соответственно); для Инны — Вова и Иван (расстояние 0.2036 и 0.8792 соответственно); для Ивана — Анна и Вова (расстояние 0.2893 и 0.4484 соответственно). Следуя формуле в начале раздела, посчитаем ожидаемую оценку Анны для фильмов 4 и 5.

Для фильма4: r = (5+0)/2 = 2,5

Для фильма5: r = (3+4)/2 = 3.5

Конец примера. Теперь представлю код, который реализует коллаборативную фильтрацию на основе похожести пользователей (User-Based коллаборативная фильтрация).

```
def naive_predict(top):

    top_sim_users = []
    #получаю top похожих пользователей
    for i in range(user_similarity.shape[0]):
        # Получаем индексы топ-N пользователей, исключая самого пользователя (i)
        top_users = user_similarity[i].argsort()[::-1][1:top + 1]
        top_sim_users.append(top_users)

    n_users, n_movies = train_data_matrix.shape
    top_similar_ratings = np.zeros((n_users, len(top_sim_users[0]), n_movies))  # Массив для оценок


    # Извлекаю оценки фильмов для топ похожих пользователей
    for i, similar_users in enumerate(top_sim_users):
        top_similar_ratings[i] = train_data_matrix[similar_users]

    pred = np.zeros((n_users, n_movies)) #матрица предсказанных значений

    #реализация формулы
    for i in range(n_users):
        pred[i] = top_similar_ratings[i].sum(axis=0) / top

    return pred
```

Рисунок 12 – Реализация User-Based коллаборативной фильтрации

Функция naive_predict предсказывает оценки фильмов для пользователей, основываясь на оценках топ-k наиболее похожих пользователей.

6 действий, реализуемых в функции по порядку:

1) Функция получает на вход количество похожих пользователей, которых необходимо отобрать из матрицы user_similarity;

2) Создаётся одномерный массив top_users, содержащий топ-к похожих пользователей. Далее этот массив добавляется в top_sim_users;

3) На основе тренировочной матрицы рейтингов train_data_matrix создаётся нулевая трёхмерная матрица top_similar_ratings. Эта матрица будет содержать все оценки топ-к пользователей, похожих на i-го пользователя;

4) Трехмерная матрица заполняется оценками нужных пользователей;

5) Создаётся двумерный массив размером с тренировочную матрицу;

6) Реализация формулы (1).

Функция возвращает двумерную матрицу pred размером (n_users, n_movies), где каждая строка соответствует пользователю, а столбцы — его предсказанным оценкам для фильмов.

2.3 Оценка качества рекомендаций

Для определения качества предсказанных оценок воспользуюсь мерой RMSE (Root Mean Square Error, среднеквадратическая ошибка):

$$RMSE = \sqrt{\frac{1}{|D|}\sum_{(u,i)\in D}\left(\widehat{r_{u,i}} - r_{u,i}\right)^2} \quad (2)$$

где

RMSE — среднеквадратическая ошибка;

$\widehat{r_{u,i}}$— предсказанное значение рейтинга;

$r_{u,i}$ — фактическое значение рейтинга;

D — множество пар (пользователь, предмет);

|D| — количество пар в множестве D.

RMSE — это метрика, используемая для оценки точности предсказательной модели, которая работает с числовыми данными. Она измеряет среднюю разницу между предсказанными значениями и фактическими (истинными) значениями, придавая больший вес большим отклонениям. RMSE показывает, насколько в среднем предсказания модели отклоняются от реальных значений. Чем ниже значение RMSE, тем точнее предсказания [3].

Допустим имеется следующий набор оценок:

Таблица 3 – Пример данных

User	Movie	Rating	Predicted
Анна	Фильм1	5	4
Вова	Фильм2	3	5
Вова	Фильм3	1	3
Иван	Фильм4	5	5

Колонка predicted содержит предсказанные алгоритмом оценки. В этом случае мера RMSE будет следующей:

$$\text{RMSE} = \sqrt{\frac{1}{4} * ((4-5)^2 + (5-3)^2 + (3-1)^2 + (5-5)^2)} = 1{,}5$$

Отклонение на 1.5 балла говорит о том, что предсказания значительно хуже, и модель нуждается в улучшении.

Реализация RMSE в среде Python:

```
def rmse(prediction, ground_truth): #RMSE - оценка качества рекомендации
    prediction = np.nan_to_num(prediction)[ground_truth.nonzero()].flatten()
    ground_truth = np.nan_to_num(ground_truth)[ground_truth.nonzero()].flatten()

    mse = mean_squared_error(prediction, ground_truth)
    return sqrt(mse)
```

Рисунок 13 – Реализация RMSE

Prediction – Матрица предсказанных значений. Ground_truth – истинные значения. В моём случае я подаю функции матрицу pred в качестве prediction и test_train_matrix в качестве ground_truth.

```
User-based проверка RMSE:  3.0871402624435453
```

Рисунок 14 – Оценка моей рекомендательной системы

ЗАКЛЮЧЕНИЕ

В ходе курсовой работы была разработана рекомендательная система на основе коллаборативной фильтрации с использованием Python. Изучены теоретические аспекты, включая основные подходы к созданию систем рекомендаций, их преимущества и недостатки. Практическая реализация позволила протестировать алгоритмы на реальном наборе данных и оценить их эффективность.

Результаты работы подтвердили важность персонализированного подхода к рекомендациям и выявили области для дальнейшего улучшения, такие как устранение проблем холодного старта и разреженности данных. Полученные знания и навыки могут быть применены для создания рекомендаций в различных сферах, включая электронную коммерцию и медиа.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Кокачев, В.А. Системы рекомендаций в науке о данных / В.А. Кокачев [Электронный ресурс]. – URL: https://dspace.spbu.ru/bitstream/11701/12104/1/Kokachev_V.pdf (дата обращения: 26.12.2024).

2. Grouplens. Movielens 20M Dataset / Grouplens [Электронный ресурс]. – URL: https://grouplens.org/datasets/movielens/20m/ (дата обращения: 26.12.2024).

3. Root mean square deviation. Root mean square deviation [Электронный ресурс]. – URL: https://en.wikipedia.org/wiki/Root_mean_square_deviation (дата обращения: 11.01.2025).

4. Habrahabr. Системы рекомендаций: введение в методы и подходы / Habrahabr [Электронный ресурс]. – URL: https://habr.com/ru/companies/yandex/articles/241455/ (дата обращения: 26.12.2024).

5. 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. С.20-22.

6. Плас, Дж. В. Python для сложных задач: наука о данных и машинное обучение / Дж. В. Плас. – Москва: ДМК Пресс, 2023. 576 с.

7. Ерёмин, О. Е. Методы реализации гибридных рекомендательных систем / О.Е. Ерёмин, Д.В. Моркулев [Электронный ресурс]. – URL: https://cyberleninka.ru/article/n/metody-realizatsii-gibridnyh-rekomendatelnyh-sistem (дата обращения: 26.12.2024).