Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Северо-Восточный федеральный университет имени М.К. Аммосова»
Институт математики и информатики

КУРСОВАЯ РАБОТА на тему:
«Построение рекомендательной системы с помощью методов машинного обучения и с предобработкой табличных данных»

Выполнил: студент 3 курса
группы БА-ПМИ-18-2 ИМИ СВФУ _____ Н.В. Федоров
подпись, дата

Руководитель: ученая степень,
звание, должность ИМИ СВФУ _____ М.Ю. Антонов
подпись, дата

Якутск 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время объем информации, доступной человеку стал настолько велик, что даже используемые сервисы для поиска информации не всегда справляются с задачами людей. Для осуществления поиска объектов необходимо знать об их существовании. С ростом объема доступной информации знать обо всем становится затруднительно. Данную задачу решают рекомендательные системы, которые связывают между собой объекты и пользователей и ищут для пользователей объекты, вероятно, их интересующие.

Большинство крупных веб-сайтов рекомендует своим пользователям различные предложения, например товары для дальнейшего изучения или людей, к которым целесообразно обратиться. Рекомендательные механизмы сортируют огромные объемы данных (целесообразно говорить о «Больших данных») для выявления потенциальных предпочтений пользователей. Amazon.com утверждает, что 40% продаж генерируются через механизмы рекомендаций.

Рекомендательные системы изменили способы взаимодействия веб-сайтов со своими пользователями. Вместо предоставления статической информации, когда пользователи ищут и, возможно, покупают продукты, рекомендательные системы увеличивают степень интерактивности для расширения предоставляемых пользователю возможностей.

Рекомендательные системы формируют рекомендации независимо для каждого конкретного пользователя на основе его прошлых действий, а также на основе поведения других пользователей.

Актуальность работы заключается в актуальности современных рекомендательных систем и отсутствии хорошо разработанных

русскоязычных работ, посвященных анализу алгоритмов, с помощью которых можно построить рекомендательную систему.

Целью моей работы является исследование и разработка эффективных алгоритмов рекомендательных систем, позволяющих отбирать рекомендации с приемлемым уровнем релевантности у большого количества пользователей с неполной или отсутствующей информацией об их предпочтениях.

Для решения поставленной цели исследования были определены следующие **задачи**:

1. Анализ методов МО для прогнозирования данных.
2. Анализ данных и их предобработка для дальнейшей работы.
3. Реализация выбранных методов и построение модели.
4. Разработка алгоритма выбора рекомендаций по ближайшим соседям.

Объект исследования: набор данных о рекомендациях пользователей, взятых с сайта https://myanimelist.net/.

Предмет исследования: методы и алгоритмы машинного обучения, используемые в рекомендательных системах.

Методы исследования: методы структурного системного анализа, методы интеллектуального анализа данных и машинного обучения, подходы, применяемые при построении рекомендательной системы.

ГЛАВА 1. РЕКОМЕНДАТЕЛЬНЫЕ СИСТЕМЫ

1.1. Что такое рекомендательные системы

Рекомендательная система – комплекс алгоритмов, программ и сервисов, задача которого предсказать, что может заинтересовать того или иного пользователя. В основе работы лежит информация о профиле человека и иные данные.

Такая система включает в себя весь процесс — от получения информации до её представления пользователю. Важен каждый этап. От информации, которая будет обрабатываться, зависит, какие алгоритмы окажутся более подходящими. Хорошие алгоритмы дают хорошие, полезные рекомендации. Критерии оценки результата позволяют выбрать наиболее подходящие алгоритмы. Первоначально задача кажется простой, но создать хорошую систему сложно. Нужно очень бережно подходить к построению системы. Даже лучшие алгоритмы не всегда дают подходящие результаты.

1.2. Как работают рекомендательные системы

Существуют два уровня:

- Особенности и предпочтения, не меняющиеся месяцами или годами; глобальные оценки; зависимость от характерных пользовательских черт: пол, место проживания; интересные страницы и т.п.
- Тренды и быстрые изменения интересов.

Данные собираются явным и/или неявным способами. В первом случае определяется предпочтения пользователя анкетами, опросами и т.п. Метод эффективный, только пользователи не всегда соглашаются.

Во втором методе фиксируется поведения потребителя на сайте или в приложении: просмотр страниц/разделов, добавление в корзину,

комментарии, отзывы и т.п. При правильном сборе данных и аналитике метод дает хорошие результаты.

1.3. Типы рекомендательных систем

Есть четыре разных типа рекомендательных систем:

1. Коллаборативные (collaborative filtering)
2. Основанные на контенте (content-based)
3. Основанные на знаниях (knowledge-based)
4. Гибридные (hybrid)

Коллаборативные рекомендательные системы:

Это системы, в которых рекомендации пользователю рассчитывается на основе оценок других пользователей. Здесь существует множество алгоритмов, но наиболее популярные - User/User(поиск соседей по оценкам), Item/Item(определение схожести предметов по оценкам пользователей) и SVD(самообучающийся алгоритм).

Суть этого типа – нахождение ближайших соседей. Близость двух пользователей или предметов определяется метриками схожести.

Преимущество – высокая теоретическая точность.

Минус – невозможно порекомендовать новым пользователям, т. к. отсутствует информация об этих пользователях (холодный старт).

Пример: Саша и Вова любят покушать суши и позаниматься спортом. А еще они оба любят машины марки Mercedes и ездят на них. Еще есть Никита, который тоже любит покушать суши после тяжелого дня и позаниматься спортом вечером. Но у него нет машины. Исходя из одинаковых интересов, Никите можно порекомендовать машины марки Mercedes.

Рекомендательные системы, основанные на контенте:

Этот тип лежит в основе многих рекомендательных систем. В отличие от коллаборативной фильтрации, этап знакомства с пользователем опускается. Система работает на основе: проанализировать контент предметов и составить набор его критериев (жанры, тэги, слова), узнать какие критерии нравится пользователю, сопоставить эти данные и получить рекомендации. Критерии составляют из пользователей и предметов точки в системе координат, и если точка пользователя и предмета рядом, то с высокой вероятностью предмет понравится пользователю.

Рекомендательная система, основанная на знаниях:

Этот тип работает на основе знаний о какой-то предметной области: о пользователях, товарах и других, которые могут помочь в ранжировании. Существуют разновидности: case-based, demographic-based, utility-based, critique-based, whatever-you-want-based и т. д. Пользователи указывают предмету другие схожие предметы. На основе этих данных создаются рекомендации. Рекомендательная система, основанная на знаниях, сможет расширить географию выбора пользователя за счет рекомендаций, которые оставили предыдущие пользователи.

Очевидное преимущество системы — высокая точность.

Минус — для разработки этой системы требуется много времени и ресурсов.

Гибридные рекомендательные системы:

Эти системы объединяют несколько выше представленных алгоритмов в один. У всех описанных ранее типов есть определенные недостатки.

Комбинирование нескольких алгоритмов в рамках одной платформы позволяет если не устранить их полностью, то хотя бы минимизировать.

Существуют несколько распространенных типов комбинирования:

- реализация по отдельности коллаборативных и контентных алгоритмов и объединение их предположений;
- включение некоторых контентных правил в коллаборативную методику;
- включение некоторых коллаборативных правил в контентную методику;
- построение общей модели, включающей в себя правила обеих методик.

Пример: рекомендательная система Netflix построена на 27 алгоритмах.

Основной недостаток гибридных систем – сложность разработки этих рекомендательных систем.

1.4. Постановка задачи

Дана обучающая выборка $XY_m = \{(X1, Y1), (X2, Y2), \dots, (Xm, Ym)\}$
$F(x): X \to Y, X_i = \{x_{i,1}, x_{i,2}, \dots, x_{i,N}\}, x_{i,j} \in D_j, Y \in D_Y$
Если D_j = {0,1} –бинарный признак.
Если D_j = {A, B,…,D} –категориальный признак.
Если D_j = R, N –числовой признак.
Если D_Y – категориальный, то это задача классификации (см. рис. 1.1.).
Если D_Y – числовой, это задача регрессии (см. рис. 1.2.).

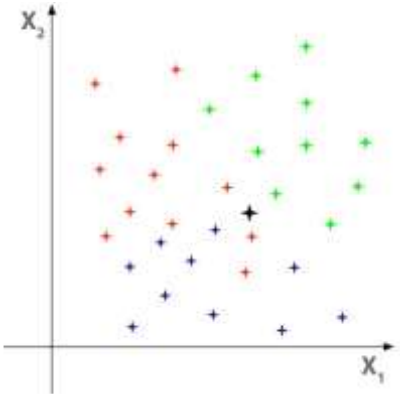

Рис. 1.1. Классификация

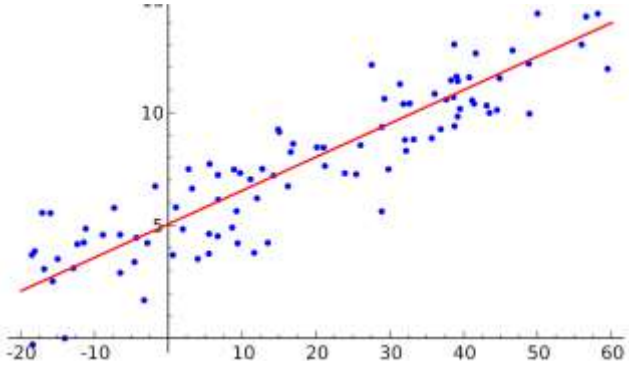

Рис. 1.2. Регрессия

ГЛАВА 2. МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ПРОГНОЗИРОВАНИЯ

2.1. Линейная регрессия

Линейная регрессия (Linear regression) — модель зависимости переменной x от одной или нескольких других переменных с линейной функцией зависимости. Линейная регрессия относится к задаче определения «линии наилучшего соответствия» через набор точек данных:

Предположим, нам задан набор из 7 точек (см. рис. 2.1).

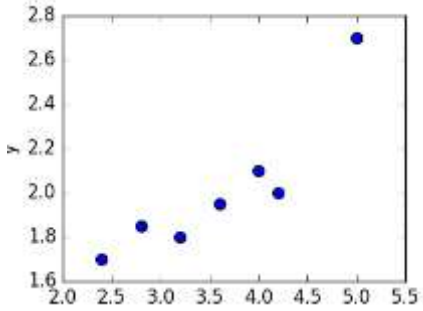

Рис. 2.1. Набор точек

Цель линейной регрессии - найти линию, которая лучше всего соответствует этим точкам. Напомним, что общее уравнение для прямой: $f(x) = m \cdot x + b$, где m - наклон прямой, а b - ее сдвиг по оси y. Таким образом, решение линейной регрессии определяет значения m и b, так что f (x) становится как можно ближе к y. Подходящая линия задаётся уравнением прямой $f(x) = 0.52 \cdot x + 0.1$ (см. рис. 2.2). Более углубленно можете ознакомиться по [16]

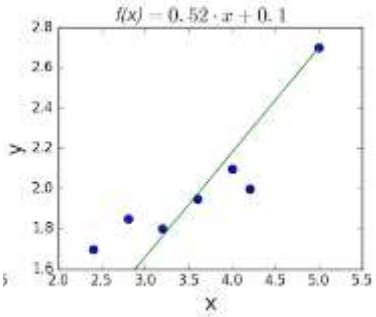

Рис. 2.2. Линия, подходящая для данного набора точек

2.2. Метод k-ближайших соседей (KNN)

Метод k-ближайших соседей – алгоритм классификации или регрессии объектов. В нашем случае используется для классификации, где объекту присваивается среднее значение по k ближайшим к нему объектам.

1. Вычислить расстояние до каждого объекта обучающей выборки
2. Выбрать k самых близких объектов
3. Предсказать класс наиболее часто встречающихся во множестве

Метрики:

1. Эвклидово $\sum |x-y|$
2. Манхэттенское расстояние $\sqrt{\sum (x-y)^2}$
3. Расстояние Чебышева $max|x-y|$
4. Расстояние Минковского $(\sum |x-y|^p)^{\frac{1}{p}}$

При k=1 алгоритм неустойчив к «выбросам», в результате получим ошибочные значения для объектов, близких к «объектам-выбросам» (не встречавшиеся ранее в алгоритме данные).

При k=N метод вырождается в константу, т.к. с какого-то элемента существенных изменений в значениях не будет (см. рис. 2.3).

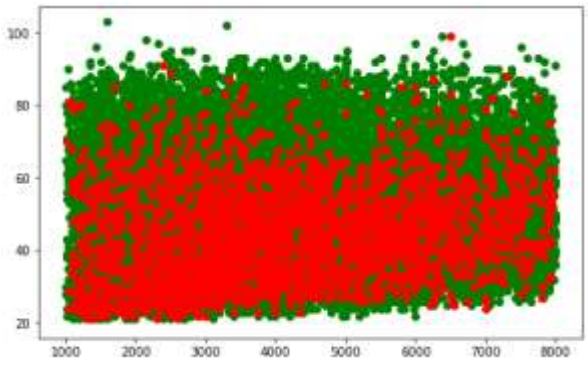

Рис.2.3. Метод KNN

2.3. Деревья решений

Дерево решений — логический алгоритм классификации, решающий задачи классификации и регрессии. Представляет собой объединение логических условий в структуру дерева. Примеры иллюстрированы на рис. 2.4 и рис. 2.5.

Рис. 2.4. Пример 1. Дерево решений

Рис. 2.5. Пример 2. Дерево решений

Как вырастить свое дерево?

1. Разбиваем по какому-то параметру так, чтобы «разнообразие данных» в итоговых множествах максимально уменьшилось (см. рис. 2.6).
2. Повторить пункт 1 (см. рис. 2.7).
3. Успех!

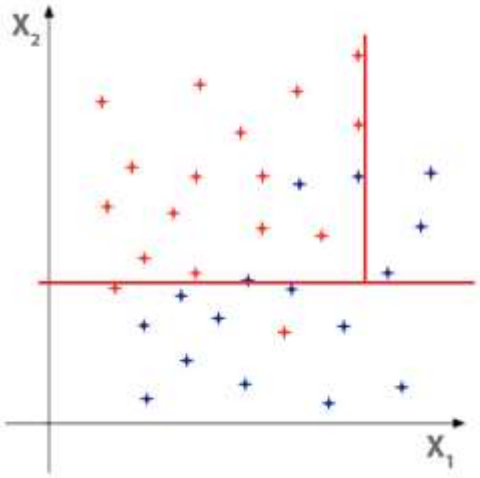

Рис. 2.6. Разбивка в дереве решений

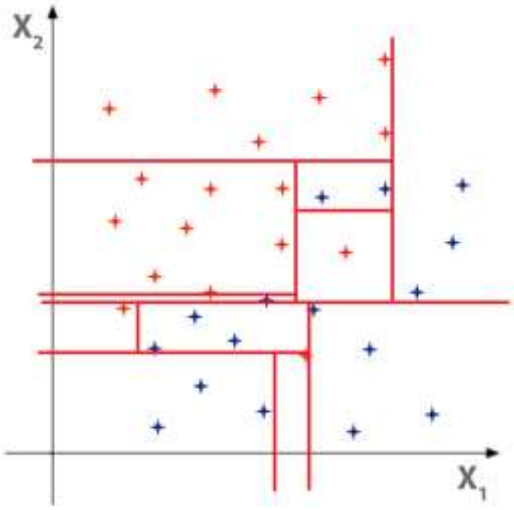

Рис. 2.7. Результат нескольких разбивок

Проблемы:

- Деревья решений строятся жадно. В случае неудачного выбора алгоритм не способен вернуться на уровень вверх и заменить неудачный предикат.
- Деревья решений склонны к переобучению. Алгоритм будет строить дерево до тех пор, пока в листе не останется объектов одного класса.

2.4. Случайный лес (Random forest)

Алгоритм случайного леса основан на использовании большого количества деревьев решений, которые по отдельности не являются «сильными» алгоритмами, но объединенный ансамбль таких деревьев покажет более высокие результаты, чем просто один «сильный» алгоритм. Алгоритм обучается путем построения деревьев решений по случайно выбранным параметрам обучающей выборки. Целевой класс для нового объекта определяется большинством голосов построенных деревьев решений. При работе с большим объемом данных показывает высокие результаты точности, также хорошо работает с малым количеством объектов и большим количеством переменных. Из-за построения большого количества деревьев интерпретируемость результатов низкая, а время на построение всех деревьев высокое.

Случайный лес, как мы узнали, состоит из большого количества отдельных деревьев решений, которые работают как ансамбль методов. Каждое дерево в случайном лесу возвращает прогноз класса, и класс с наибольшим количеством голосов становится прогнозом леса (см. рис. 2.8).

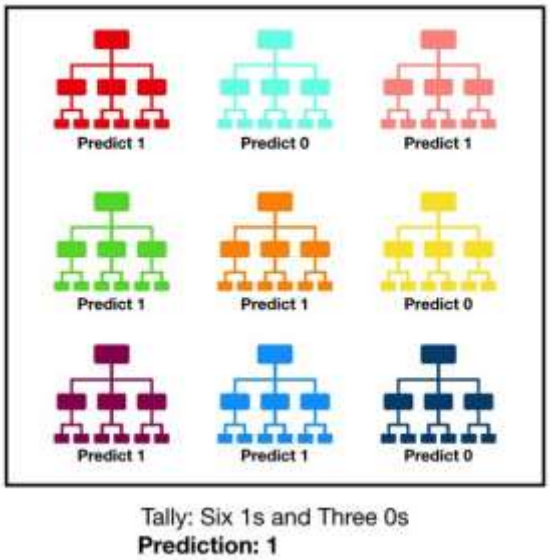

Рис. 2.8. Ансамбль деревьев решений для алгоритма Случайный лес. Иллюстрация по [14].

2.5. Измерение ошибок

Средняя абсолютная ошибка	$MAE = \frac{1}{l}\sum_{i=0}^{l}\|(f(x_i) - y_i)\|$
Среднеквадратическая ошибка	$MSE = \frac{1}{l}\sum_{i=0}^{l}(f(x_i) - y_i)^2$
Квадратный корень из MSE	$RMSE = \sqrt{MSE}$

RMSE – это среднее расстояние точки данных от подобранной линии, измеренное вдоль вертикальной линии (см. рис. 2.9).

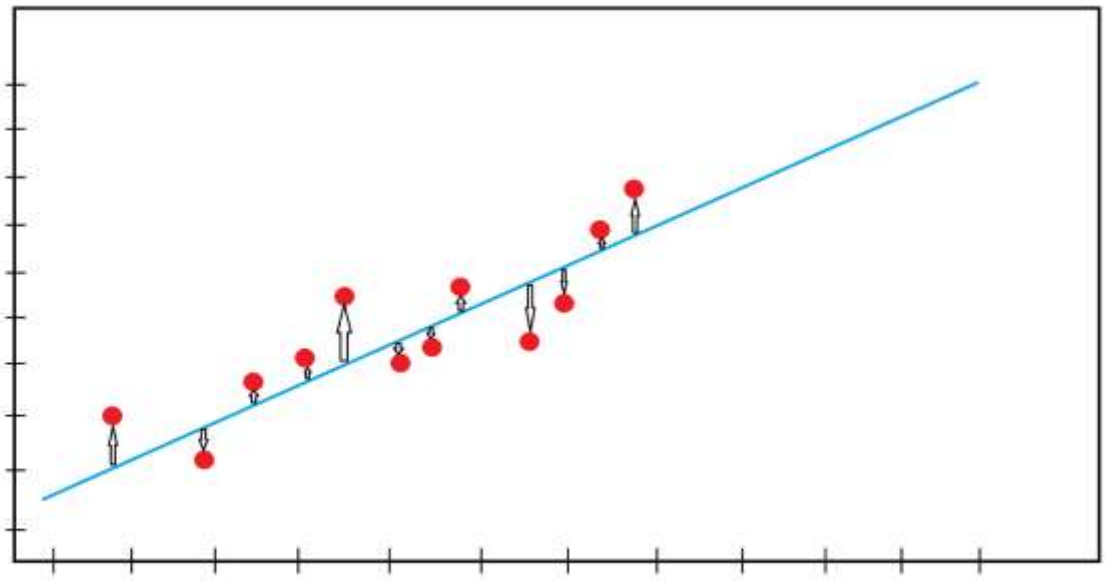

Рис. 2.9. RMSE

2.6. Сравнение

Многообразие существующих алгоритмов машинного обучения не позволяет говорить об однозначно лучших, поскольку каждый из алгоритмов имеет свои слабые и сильные стороны. На основе анализа методов классификации составлена таблица сравнения основных методов

классификации по нескольким параметрам (Таблица 1), где каждый из них оценивался по шкале от 0 до 2, где 2 - максимальный балл.

Таблица 1 - Сравнение методов машинного обучения

Параметры \ Методы машинного обучения	Деревья решений	*k*-ближайших соседей	Случайный лес
Интерпретируемость результатов	2	2	0
Высокая скорость обучения	2	2	0
Высокая скорость классификации	2	1	1
Низкое количество параметров для настройки	1	2	1
Работа с малым объемом данных	1	1	0
Постоянство структуры вне зависимости от релевантности параметров	0	1	2

Представленные методы имеют разные области применения, разные требования к набору входных данных и типу параметров, разные вычислительные характеристики и уровень потребления памяти. Использование этих методов классификации в рекомендательных системах также даст разные результаты. Поэтому для каждой задачи и каждого набора данных необходимо использовать свой метод классификации, который может быть основан на ансамблевом подходе и включать множество методов

с последующим объединением результатов при принятии окончательного решения.

Построение ансамблей из различных алгоритмов позволяет устранить один из недостатков машинного обучения, когда структура небольшой обучающей выборки из-за своего размера мало соответствует структуре всего рассматриваемого набора данных, особенно на больших данных. В этом случае обучение каждого алгоритма можно проводить на разных обучающих выборках.

ГЛАВА 3. РЕАЛИЗАЦИЯ МЕТОДОВ И ПОСТРОЕНИЕ МОДЕЛЕЙ

3.1. Импорт библиотек и загрузка данных

Будем работать на Python и первым делом импортируем нужные нам библиотеки:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

NumPy – это open-source модуль для python, который предоставляет общие математические и числовые операции в виде пре-скомпилированных, быстрых функций.

Pyplot — это коллекция функций в стиле команд, которая позволяет использовать matplotlib почти так же, как MATLAB. Она нужна нам чтобы строить графики.

Pandas — это библиотека для работы с данными на Python.

Мы будем работать с **датафреймами** (DataFrame)– это таблицы, состоящие из строк, колонок и ячеек, и **сериями** (Series) – это, технически, колонки датафреймов.

Мы будем работать с данными о рекомендациях от 73 516 пользователей myanimelist.net.

Anime.csv

- anime_id – уникальный идентификатор myanimelist.net, идентифицирующий аниме.
- name – полное название аниме.
- genre – список жанров этого аниме, разделенных запятыми.

- type - фильм, TB, OBA и др.
- episodes – сколько серий в этом шоу (1, если фильм).
- rating – средняя оценка этого аниме из 10.
- members – количество участников сообщества, которые находятся в "группе" этого аниме.

Rating.csv

- user_id – неидентифицируемый случайно сгенерированный идентификатор пользователя.
- anime_id – аниме, которое оценил этот пользователь.
- rating – рейтинг из 10, который присвоил этот пользователь (-1, если пользователь смотрел его, но не выставлял оценку).

Загружаем данные (метод read_csv):

```
anime_df = pd.read_csv("anime.csv")
rating_df = pd.read_csv("rating.csv")
```

3.2. Предобработка данных

Anime.csv

Сперва посмотрим на первые 5 строк с помощью метода head (Таблица 2):

```
anime_df.head()
```

Таблица 2 - Данные из anime.csv

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572
4	9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.16	151266

Каждая строка представляет собой один аниме – это объект исследования. Столбцы – признаки объекта.

Посмотрим на размер данных и типы признаков (см. рис. 3.1)

```
print(anime_df.shape): (12294, 7)
anime_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12294 entries, 0 to 12293
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   anime_id  12294 non-null  int64
 1   name      12294 non-null  object
 2   genre     12232 non-null  object
 3   type      12269 non-null  object
 4   episodes  12294 non-null  object
 5   rating    12064 non-null  float64
 6   members   12294 non-null  int64
dtypes: float64(1), int64(2), object(4)
memory usage: 672.5+ KB
```

Рис. 3.1. Типы признаков датафрейма anime.csv

Посмотрим графически на отсутствующие значения (см. рис. 3.2).

```
plt.figure(figsize=(8,6))
sns.heatmap(anime_df.isnull())
plt.title("Отсутствующие значения в 'anime'", fontsize = 15)
plt.show()
```

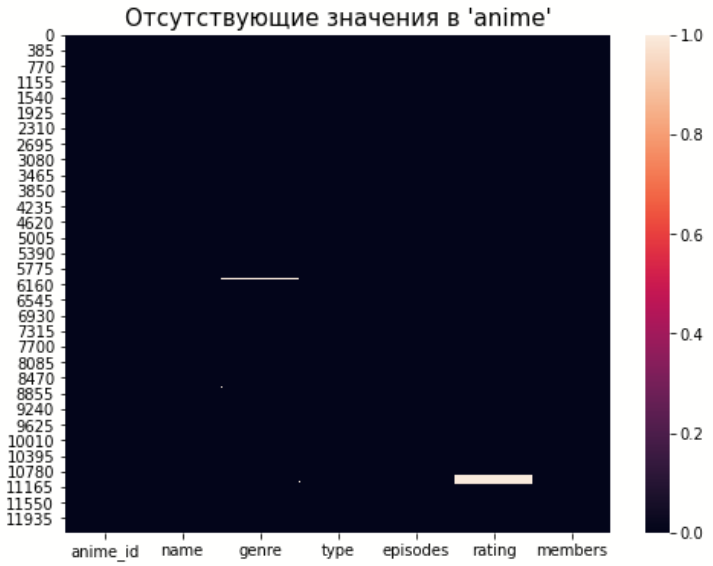

Рис. 3.2. Незаполненные поля в таблице anime.csv

Как мы видим, имеется 3 численных признака (int64(2) и float64(1)) и 4 категориальных (object). А также в признаках “genre”, “type” и “rating” отсутствуют значения. Давайте воспользуемся двумя путями для восполнения недостающих значений.

1) Применим метод dropna, который удаляет недостающие значения и посмотрим информацию об этом датафрейме (см. рис. 3.3).

```
anime_df.dropna(inplace=True)
anime_df.shape: (12017, 7)
anime_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12017 entries, 0 to 12293
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   anime_id  12017 non-null  int64
 1   name      12017 non-null  object
 2   genre     12017 non-null  object
 3   type      12017 non-null  object
 4   episodes  12017 non-null  object
 5   rating    12017 non-null  float64
 6   members   12017 non-null  int64
dtypes: float64(1), int64(2), object(4)
memory usage: 751.1+ KB
```

Рис. 3.3. Информация о датафрейме anime

2) Досконально рассмотрим, изучим и восполним отсутствующие значения (см. рис. 3.4).

```
anime_df.isnull().sum()
```

```
anime_id      0
name          0
genre        62
type         25
episodes      0
rating      230
members       0
dtype: int64
```

Рис. 3.4. Отсутствующие значения в датафрейме anime

Проанализируем “type” (см. рис. 3.5):

```
anime_df.type.value_counts()
```

```
TV         3787
OVA        3311
Movie      2348
Special    1676
ONA         659
Music       488
Name: type, dtype: int64
```

Рис. 3.5. Серия "type" датафрейма anime

Заполним пустые поля значением “TV”, т.к. она является чаще всего встречающимся значением.

```
anime_df.fillna({'type':"TV"},inplace=True)
```

Дальше заполним “rating” нулями (0):

```
anime_df.fillna({'rating':0},inplace=True)
```

Посмотрим на серию “genre” датафрейма anime (см. рис. 3.6).

```
anime_df.genre.value_counts()
```

```
Hentai                                                        823
Comedy                                                        523
Music                                                         301
Kids                                                          199
Comedy, Slice of Life                                         179
                                                             ...
Drama, Magic, Mecha, Romance, Shoujo Ai, Shounen, Supernatural  1
Comedy, Drama, Ecchi, Romance                                 1
Demons, Historical                                            1
Action, Comedy, Fantasy, Magic, Shoujo, Slice of Life         1
Adventure, Comedy, Kids, Sci-Fi, Shounen                      1
Name: genre, Length: 3264, dtype: int64
```

Рис. 3.6. Genre

Можно было заполнить пустые поля значением "Unknown". Но давайте просто выбросим колонку "genre", т.к. как-то оптимально и правильно заполнить его очень сложно, даже если это возможно.

```
anime_df.drop('genre', axis=1, inplace=True)
anime_df.isnull().sum()
```

Как мы видим, недостающих значений не осталось (см. рис. 3.7 и рис. 3.8).

```
anime_id    0
name        0
genre       0
type        0
episodes    0
rating      0
members     0
dtype: int64
```

Рис. 3.7. Отсутствующие значения в датафрейме anime

```
plt.figure(figsize=(8,6))
sns.heatmap(anime.isnull())
plt.title("Отсутствующие значения в 'anime'", fontsize = 15)
plt.show()
```

Рис. 3.8. Гистограмма отсутствующих значений датафрейма anime

Посчитаем кол-во пользователей и аниме.

```
n_users = rating.user_id.unique().shape[0]
n_items = rating.anime_id.unique().shape[0]
print(n_users): 73515
print(n_items): 11200
```

Rating.csv

Посмотрим на первые 5 строк rating.csv (см. рис. 3.9)

```
rating_df.head()
```

	user_id	anime_id	rating
0	1	20	-1
1	1	24	-1
2	1	79	-1
3	1	226	-1
4	1	241	-1

Рис. 3.9. Датафрейм rating

```
print(rating_df.shape): (7813737, 3)
```

Объединение таблиц:

Следующее что сделаем объединим две таблицы в один по колонке “anime_id” (см. рис. 3.10).

```
df = pd.merge(anime_df, rating_df, on = 'anime_id')
df.head()
```

	anime_id	name	genre	type	episodes	rating_x	members	user_id	rating_y
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	99	5
1	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	152	10
2	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	244	10
3	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	271	10
4	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	278	-1

Рис. 3.10. Датафрейм объединения anime и rating

```
print(df.shape): (7813727, 8)
```

Узнаем самые популярные Аниме (см. рис. 3.11) и самые просматриваемые типы аниме (см. рис. 3.12):

```
plt.figure(figsize = (8,6))
df_members = df.sort_values(by = "members", ascending = False).copy()
sns.barplot(data = df_members.iloc[0:10], y = "name", x = "members")
plt.title("Самые просматриваемые Аниме", size = 12)
plt.xlabel("Участники")
plt.ylabel("")
plt.show()
```

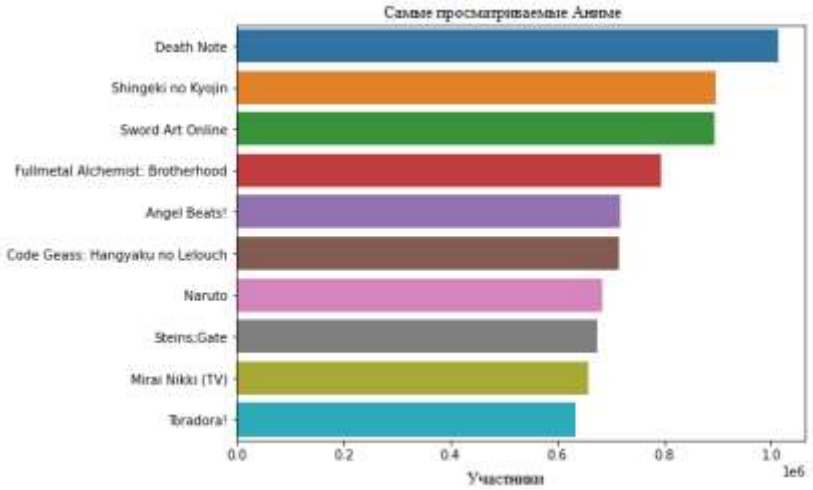

Рис. 3.11. Самые популярные аниме

```
fig = plt.figure(figsize=(12,10))
sns.countplot(df['type'], palette='gist_rainbow')
plt.title("Самые просматриваемые типы Аниме", fontsize=20)
plt.xlabel("Тип", fontsize=20)
plt.ylabel("Количество просмотров", fontsize = 20)
plt.legend(df['type'])
plt.show()
```

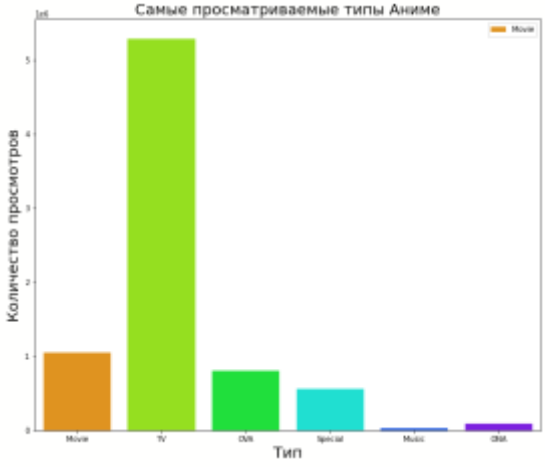

Рис. 3.12. Самые просматриваемы типы аниме

Выбросим из таблицы строки с рейтингом “-1”, т.к. они нам ничего не дают и посмотрим на информацию (info()) по датафрейму df (см. рис. 3.13).

```
df = df[df.rating_y != -1]
print(df.shape): (6337239, 8)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6337239 entries, 0 to 7813724
Data columns (total 8 columns):
 #   Column    Dtype
---  ------    -----
 0   anime_id  int64
 1   name      object
 2   type      object
 3   episodes  object
 4   rating_x  float64
 5   members   int64
 6   user_id   int64
 7   rating_y  int64
dtypes: float64(1), int64(4), object(3)
memory usage: 435.1+ MB
```

Рис. 3.13. Информация по df

Нам нужно категориальные признаки перевести в числовые (см. рис. 3.14).

```
df.drop('name', axis=1, inplace=True)
df = pd.get_dummies(df, columns=['type'])
df = df.loc[df['episodes'] != 'Unknown']
df.episodes = df.episodes.astype('int64')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6337233 entries, 0 to 7813724
Data columns (total 12 columns):
 #   Column        Dtype
---  ------        -----
 0   anime_id      int64
 1   episodes      int64
 2   rating_x      float64
 3   members       int64
 4   user_id       int64
 5   rating_y      int64
 6   type_Movie    uint8
 7   type_Music    uint8
 8   type_ONA      uint8
 9   type_OVA      uint8
 10  type_Special  uint8
 11  type_TV       uint8
dtypes: float64(1), int64(5), uint8(6)
memory usage: 374.7 MB
```

Рис. 3.14. Перевод категориальных в числовые

Предобработка данных завершилась. Теперь разделим на y (рейтинги, на которых будем обучаться, а потом предсказывать, т.е. то, что нам надо найти, см. рис. 3.16) и X (все наши данные без рейтингов, см. рис. 3.15).

```
y = df["rating_y"]
X = df.drop("rating_y", axis=1)
```

	anime_id	episodes	rating_x	members	user_id	type_Movie	type_Music	type_ONA	type_OVA	type_Special	type_TV
0	32281	1	9.37	200630	99	1	0	0	0	0	0
1	32281	1	9.37	200630	152	1	0	0	0	0	0
2	32281	1	9.37	200630	244	1	0	0	0	0	0
3	32281	1	9.37	200630	271	1	0	0	0	0	0
5	32281	1	9.37	200630	322	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
7813713	9316	1	4.15	211	58483	0	0	0	1	0	0
7813716	5543	1	4.28	183	49503	0	0	0	1	0	0
7813717	5543	1	4.28	183	58483	0	0	0	1	0	0
7813720	5621	4	4.88	219	49503	0	0	0	1	0	0
7813724	6133	1	4.98	175	60365	0	0	0	1	0	0

6337233 rows × 11 columns

Рис. 3.15. X

```
0          5
1         10
2         10
3         10
5         10
          ..
7813713    1
7813716    4
7813717    1
7813720    6
7813724    4
Name: rating_y, Length: 6337233, dtype: int64
```

Рис. 3.16. y

3.3. Строим модель

```
from sklearn.model_selection import train_test_split
```

Разделяем данные на тренировочную и тестовые наборы данных:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, random_state=0)
```

Линейная регрессия (linreg):

```
from sklearn import ensemble, linear_model
from sklearn.metrics import mean_squared_error
linreg = linear_model.LinearRegression()
linreg.fit(X_train, y_train)
y_test_linreg = linreg.predict(X_test)
```

Посмотрим на гистограммы тестовой и предиктовой (см. рис. 3.17 и рис. 3.18)

```
plt.hist(y_test)
plt.hist(y_test_linreg)
```

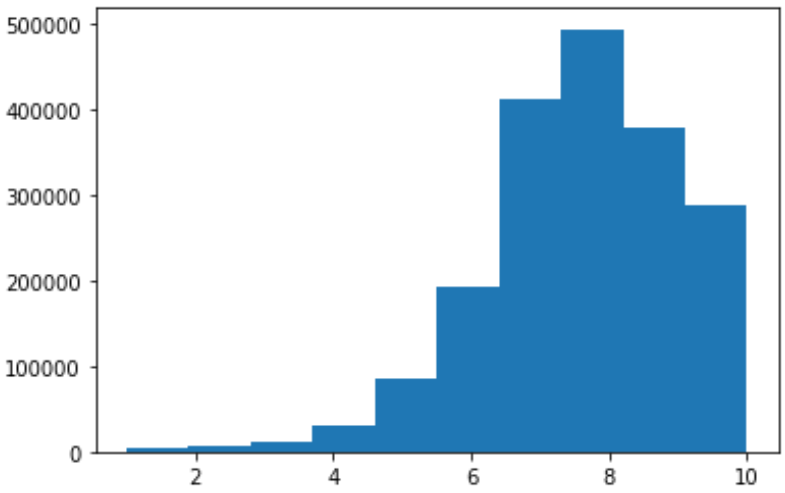

Рис. 3.17. Гистограмма тестовой

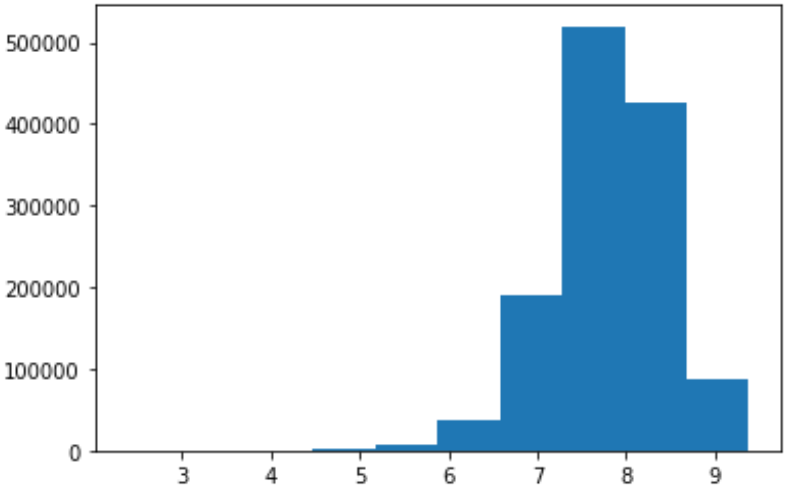

Рис. 3.18. Гистограмма предиктовой

```
print(mean_squared_error(y_test_linreg, y_test)): 2.053301334031836
```

Random forest:

```
rf = ensemble.RandomForestRegressor(n_estimators=100, max_depth=10, min_samples
_split=7)
rf.fit(X_train, y_train)
y_test_rf = rf.predict(X_test)
```

Также посмотрим на гистограммы тестовой и предиктовой (см. рис. 3.19 и рис. 3.20).

```
plt.hist(y_test);
plt.hist(y_test_rf);
```

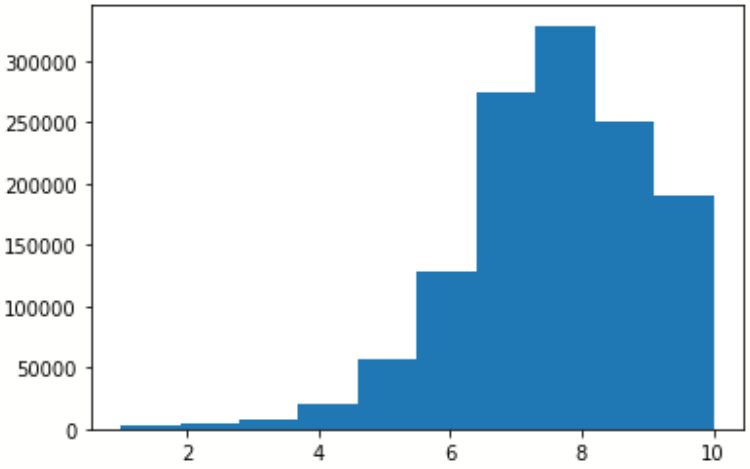

Рис. 3.19. Гистограмма тестовой

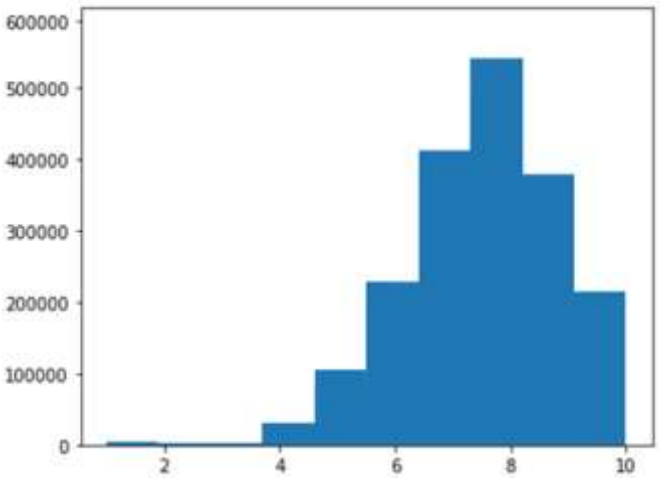

Рис. 3.20. Гистограмма предиктовой

```
print(mean_squared_error(y_test_rf, y_test)): 2.0481371069717444
```

3.4. Простая рекомендательная система на 10 ближайших соседях

Используем те же данные. Сперва удаляем аниме с низким количеством оценок и пользователей, которые дали мало оценок. Потом строим рейтинговую матрицу. Преобразуем матрицу рейтингов в матрицу CSR для экономии памяти. Поместим матрицу рейтинга CSR в KNN. Как результат, получим десять ближайших соседей. Тем самым, выведем десять рекомендованных аниме.

Мы будем рассматривать только популярные аниме (рейтинг>50) и пользователей, которые дали много оценок по разным аниме(>50).

Создадим датафрейм “anime_rating_count” (см. рис. 3.21)

```
anime_rating_count = rating.groupby(by='anime_id').count()['rating'].rese
t_index().rename(columns={'rating':'rating_count'})
anime_rating_count.head()
```

	anime_id	rating_count
0	1	15509
1	5	6927
2	6	11077
3	7	2629
4	8	413

Рис. 3.21. Датафрейм anime_rating_count

Оставляем аниме с количеством рейтингов больше 50 (см. рис. 3.22)

```
filtered_anime = anime_rating_count[anime_rating_count['rating_count']>50
]
filtered_anime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5625 entries, 0 to 11188
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   anime_id      5625 non-null   int64
 1   rating_count  5625 non-null   int64
dtypes: int64(2)
memory usage: 131.8 KB
```

Рис. 3.22. filtered_anime

Также создадим датафрейм “user_rating_count” (см. рис. 3.23)

```
user_rating_count = rating.groupby(by='user_id').count()['rating'].reset_
index().rename(columns={'rating':'rating_count'})
user_rating_count.head()
```

	user_id	rating_count
0	1	153
1	2	3
2	3	94
3	4	52
4	5	467

Рис. 3.23 user_rating_count

Оставляем пользователей с количеством поставленных рейтингов больше 50 (см. рис. 3.24)

```
filtered_user = user_rating_count[user_rating_count['rating_count']>50]
filtered_user.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 39115 entries, 0 to 73513
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   user_id       39115 non-null  int64
 1   rating_count  39115 non-null  int64
dtypes: int64(2)
memory usage: 916.8 KB
```

Рис. 3.24. filtered_user

Объединим filtered_anime и filtered_user (см. рис. 3.25)

```
filtered_rating_anime = rating[rating['anime_id'].isin(filtered_anime['an
ime_id'])]
```

```
filtered_rating = filtered_rating_anime[filtered_rating_anime['user_id'].
isin(filtered_user['user_id'])]
filtered_rating.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7083803 entries, 0 to 7813734
Data columns (total 3 columns):
 #   Column    Dtype
---  ------    -----
 0   user_id   int64
 1   anime_id  int64
 2   rating    int64
dtypes: int64(3)
memory usage: 216.2 MB
```

Рис. 3.25 filtered_rating

Рейтинг-матрица (разреженная матрица):

Воспользуемся pivot_table (создает сводную таблицу, см. рис. 3.26).

```
rating_matrix = filtered_rating.pivot_table(index='anime_id',columns='use
r_id',values='rating').fillna(0)
print(rating_matrix.shape)
rating_matrix.head(10)
```

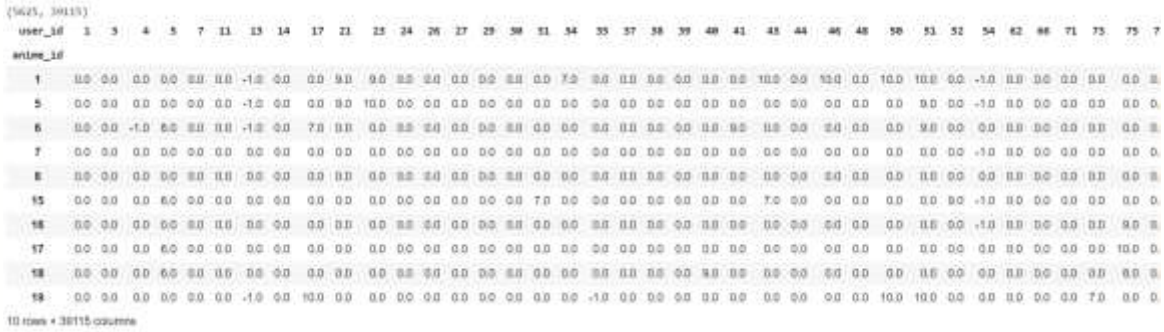

Рис. 3.26. pivot_table

CSR-матрица:

```
from scipy.sparse import csr_matrix
csr_rating_matrix = csr_matrix(rating_matrix.values)
```

Что делает CSR-матрица? Оставляет только ненулевые значения из рейтинг-матрицы.

KNN:

Найдем с помощью KNN k ближайших точек данных, по которым будет рекомендовано аниме. Мы также будем использовать косинусное сходство [15] в качестве метрики для алгоритма. В рис. 3.27 показаны параметры KNN.

```
from sklearn.neighbors import NearestNeighbors
rec = NearestNeighbors(metric='cosine')
rec.fit(csr_rating_matrix)
```

```
NearestNeighbors(algorithm='auto', leaf_size=30, metric='cosine',
                 metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                 radius=1.0)
```

Рис. 3.27. Параметры KNN

Выберем аниме, по которому будем строить рекомендацию (см. рис. 3.28).

```
user_anime = anime[anime['name']=='Naruto']
print(user_anime)
```

	anime_id	name	genre	type	episodes	rating	members
841	20	Naruto	Action, Comedy, Martial Arts, Shounen, Super P...	TV	220	7.81	683297

Рис. 3.28. Выбранное аниме, по которому строится рекомендация

```
user_anime_index = np.where(rating_matrix.index==int(user_anime['anime_id
'']))[0][0]
print(user_anime_index): 10
```

Посмотрим все рейтинги по этому аниме (см. рис. 3.29):

```
user_anime_ratings = rating_matrix.iloc[user_anime_index]
print(user_anime_ratings):
```

```
user_id
1       -1.0
3        8.0
4        0.0
5        6.0
7        0.0
        ...
73503    0.0
73504    0.0
73507    0.0
73510    0.0
73515    0.0
Name: 20, Length: 39115, dtype: float64
```

Рис. 3.29. Все рейтинги по выбранному аниме

Нам нужно преобразовать это в 2D массив (только с 1 строкой), так как алгоритм не принимает 1D массив (см. рис. 3.30).

```
user_anime_ratings_reshaped = user_anime_ratings.values.reshape(1,-1)
user_anime_ratings_reshaped
```

```
array([[-1.,  8.,  0., ...,  0.,  0.,  0.]])
```

Рис. 3.30. Преобразование в 2D

```
distances, indices = rec.kneighbors(user_anime_ratings_reshaped,n_neighbo
rs=11)
```

Индексы ближайших соседей (см. рис. 3.31).

```
array([[  10, 1316, 3976,   98, 1349, 2782,  241, 4444, 3617, 2157,  199]])
```

Рис. 3.31. Индексы

Расстояния ближайших соседей до аниме пользователя (см. рис. 3.32).

```
array([[1.75526260e-13, 3.83547534e-01, 4.60357470e-01, 4.60516627e-01,
        4.62602576e-01, 4.62895693e-01, 4.65323580e-01, 4.68829505e-01,
        4.83850482e-01, 4.84128444e-01, 4.89905062e-01]])
```

Рис. 3.32. Расстояние ближайших соседей до выбранного аниме

Возвращенные индексы будут использоваться для получения аниме id (индекса) в рейтинговой матрице. Эти индексы являются ближайшими соседями. Мы исключаем первый элемент, так как первый ближайший сосед

сам по себе. В итоге получаем 10 рекомендованных аниме по выбранному пользователем аниме (см. рис. 3.33).

```
nearest_neighbors_indices = rating_matrix.iloc[indices[0]].index[1:]

nearest_neighbors = pd.DataFrame({'anime_id': nearest_neighbors_indices})
pd.merge(nearest_neighbors,anime,on='anime_id',how='left')
```

	anime_id	name	genre	type	episodes	rating	members
0	1535	Death Note	Mystery, Police, Psychological, Supernatural, ...	TV	37	8.71	1013917
1	11757	Sword Art Online	Action, Adventure, Fantasy, Game, Romance	TV	25	7.83	893100
2	121	Fullmetal Alchemist	Action, Adventure, Comedy, Drama, Fantasy, Mag...	TV	51	8.33	600384
3	1575	Code Geass: Hangyaku no Lelouch	Action, Mecha, Military, School, Sci-Fi, Super...	TV	25	8.83	715151
4	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
5	269	Bleach	Action, Comedy, Shounen, Super Power, Supernat...	TV	366	7.95	624055
6	16498	Shingeki no Kyojin	Action, Drama, Fantasy, Shounen, Super Power	TV	25	8.54	896229
7	9919	Ao no Exorcist	Action, Demons, Fantasy, Shounen, Supernatural	TV	25	7.92	583823
8	2904	Code Geass: Hangyaku no Lelouch R2	Action, Drama, Mecha, Military, Sci-Fi, Super ...	TV	25	8.98	572888
9	226	Elfen Lied	Action, Drama, Horror, Psychological, Romance,...	TV	13	7.85	623511

Рис. 3.33. Рекомендация. 10 ближайших соседей.

ЗАКЛЮЧЕНИЕ

В ходе курсовой работы мы исследовали и разработали эффективные алгоритмы рекомендательных систем, позволяющие прогнозировать оценки или отбирать рекомендации с приемлемым уровнем релевантности у большого количества пользователей с неполной или отсутствующей информацией об их предпочтениях. Для этого был сделан анализ методов МО для прогнозирования данных, анализ данных и их предобработка для корректной работы. А также мы реализовали выбранные методы и построили модели. Для полной рекомендательной системы был разработан алгоритм выбора рекомендаций по ближайшим соседям.

В итоге удалось создать работающий сервис, который ищет и выдает рекомендации по аниме на основе пользовательских предпочтений и открытых данных с myanimelist.net. Несмотря на простоту текущих алгоритмов, систему можно легко модифицировать для выполнения более сложных методов обработки и для произвольно больших объемов обрабатываемых данных. Система может использоваться как основа для пользовательских рекомендательных приложений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Федоровский, А.Н. Архитектура рекомендательной системы, работающей на основе неявных пользовательских оценок / А.Н. Федоровский, В.К. Логачева // Труды 13й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» - RCDL'2011, Воронеж, Россия, 2011.

2. Правиков, А.А. Разработка и применение метода формализации проектирования рекомендательных систем с естественно-языковым интерфейсом: дис.... канд. технических наук, Москва, 2011. С. 160

3. “How Computers Know What We Want — Before We Do” [электронный ресурс] URL: http://content.time.com/time/magazine/article/0,9171,1992403,00.html [дата обращения: 16 мая 2021]

4. Decision Trees, Random Forests, and Nearest-Neighbor classifiers: https://pages.mtu.edu/~shanem/psy5220/daily/Day13/treesforestsKNN.html [дата обращения: 17 мая 2021]

5. Классификация, регрессия и другие алгоритмы Data Mining с использованием R: https://ranalytics.github.io/data-mining/ [дата обращения: 17 мая 2021]

6. Что такое дерево решений и где его используют?: https://habr.com/ru/company/productstar/blog/523044/ [дата обращения: 17 мая 2021]

7. Breiman L., Friedman J. H., Olshen R. A., Stone C. J. Classification and regression trees. Monterey, CA: Wadsworth & Brooks // Cole Advanced Books & Software, 1984.

8. Patra A., Singh D. A Survey Report on Text Classification with Different Term Weighing Methods and Comparison between Classification Algorithms // International Journal of Computer Applications. 2013. № 7 (75). P. 14–18.

9. Satyanarayana N., Ramalingaswamy C., Ramadevi Y. Survey of classification techniques in data mining // IJISET - International Journal of Innovative Science, Engineering & Technology. 2014. № 9 (1). P. 268–278.

10. Altman N. S. An introduction to kernel and nearest-neighbor nonparametric regression // The American Statistician, 1992. 46 (3). P. 175–185.

11. Crisci C., Ghattas B., Perera G. A review of supervised machine learning algorithms and their applications to ecological data // Ecological Modelling, 2012. (240). P. 113–122.

12. Breiman, Leo. Random Forests //Machine Learning, 2001. 45 (1), pp. 5–32.

13. Understanding Random Forest: https://towardsdatascience.com/understanding-random-forest-58381e0602d2 [дата обращения: 18 мая 2021]

14. Как работает случайный лес? https://nuancesprog.ru/p/6160/ [дата обращения: 18 мая 2021]

15. Косинусное сходство: https://en.wikipedia.org/wiki/Cosine_similarity [дата обращения: 20 мая 2021]

16. Линейная регрессия в машинном обучении: https://neurohive.io/ru/osnovy-data-science/linejnaja-regressija/ [дата обращения: 23 мая]

ПРИЛОЖЕНИЕ

Ноутбуком Colab на код Python можете ознакомиться по https://github.com/nikfed280199/anime_recommendation