|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра прикладной математики (ПМ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Модели и методы предиктивной аналитики»

**Практическое занятие № 2**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | *ИМБО-02-22, Ким Кирилл Сергеевич* | (подпись) | |
| Преподаватель | *Крынецкий Борис Алексеевич, преподаватель* | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_202\_\_г. | |  | |

Москва 2025 г.

СОДЕРЖАНИЕ

[ПРАКТИЧЕСКАЯ РАБОТА 2 3](#_Toc210159583)

[Введение 3](#_Toc210159584)

[Шаги выполнения 3](#_Toc210159585)

[Результат работы: 8](#_Toc210159586)

[Вывод: 8](#_Toc210159587)

[Список использованных источников и литературы: 9](#_Toc210159588)

ПРАКТИЧЕСКАЯ РАБОТА 2

Введение

**Цель**

Создать рекомендательную систему, которая будет рекомендовать элементы на основе предпочтений пользователей. Найти данные для рекомендаций, провести их обработку и построить модель для генерации рекомендаций.

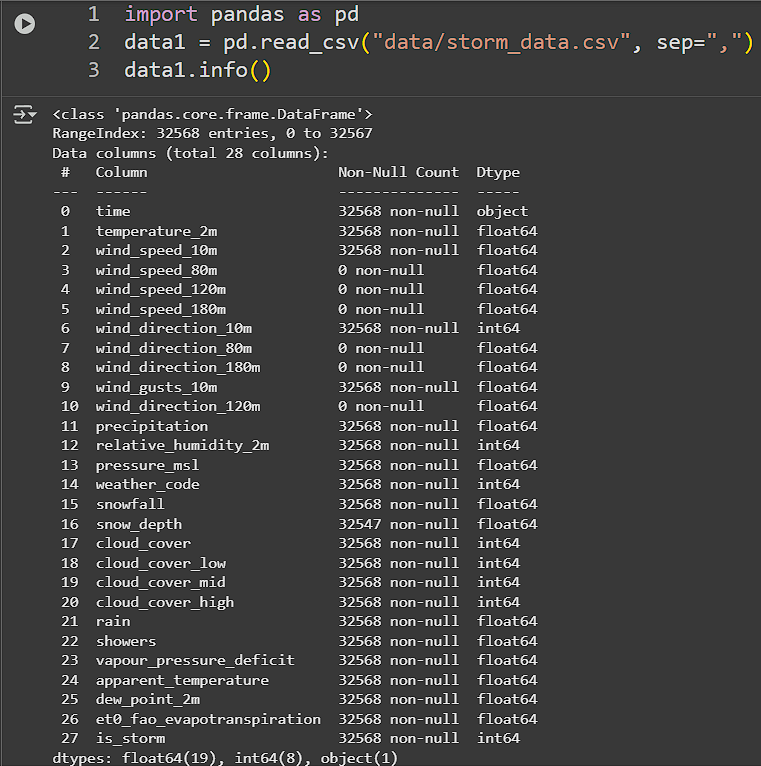
Шаги выполнения

1. Сбор данных для рекомендаций.

Данные были взяты из Kaggle (https://www.kaggle.com/datasets/syedjaferk/book-crossing-dataset?resource=download)

Листинг 1 – для временного ряда

|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  import seaborn as sns  from sklearn.metrics.pairwise import cosine\_similarity  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error  books = pd.read\_csv("BX-Books.csv", sep=";", encoding="latin-1", quotechar='"', on\_bad\_lines="skip", low\_memory=False)  users = pd.read\_csv("BX-Users.csv", sep=";", encoding="latin-1", quotechar='"', on\_bad\_lines="skip", low\_memory=False)  ratings = pd.read\_csv("BX-Book-Ratings.csv", sep=";", encoding="latin-1", quotechar='"', on\_bad\_lines="skip", low\_memory=False) |

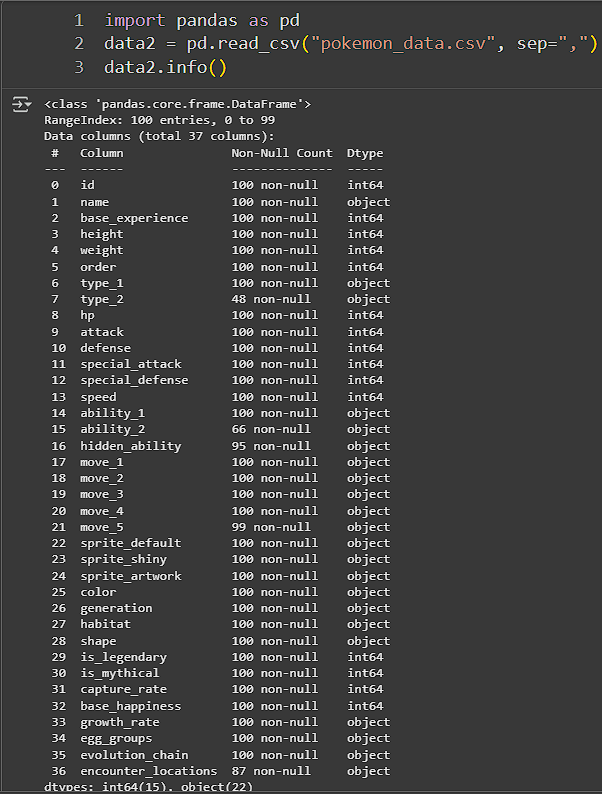


**Рисунок 1 – Временной ряд**

1. Обработка данных

Листинг 2 – для многомерных данных

|  |
| --- |
| # Удаляем "неявные" оценки (0 = не оценено)  ratings\_explicit = ratings[ratings['Book-Rating'] > 0]  # Фильтрация пользователей и книг  print("Фильтрация активных пользователей и популярных книг")  USER\_RATINGS = 45  BOOK\_RATINGS = 25  user\_counts = ratings\_explicit['User-ID'].value\_counts()  book\_counts = ratings\_explicit['ISBN'].value\_counts()  print(f"Пользователей всего: {len(user\_counts)}")  print(f"Книг всего: {len(book\_counts)}")  active\_users = user\_counts[user\_counts >= USER\_RATINGS].index  popular\_books = book\_counts[book\_counts >= BOOK\_RATINGS].index  print(f"Активных пользователей (>={USER\_RATINGS} оценок): {len(active\_users)}")  print(f"Популярных книг (>={BOOK\_RATINGS} оценок): {len(popular\_books)}")  ratings\_filtered = ratings\_explicit[      ratings\_explicit['User-ID'].isin(active\_users) &      ratings\_explicit['ISBN'].isin(popular\_books)  ].copy()  print(f"После фильтрации: {len(ratings\_filtered)} оценок")  print(f"Уникальных пользователей: {ratings\_filtered['User-ID'].nunique()}")  print(f"Уникальных книг: {ratings\_filtered['ISBN'].nunique()}") |

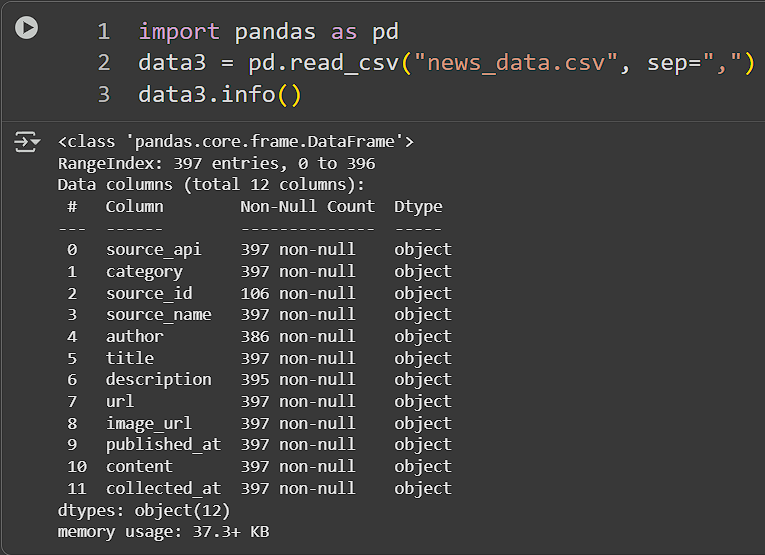


**Рисунок 2 – Многомерные данные**

1. Построение рекомендательной системы

Листинг 3 – для наборов текста

|  |
| --- |
| train\_data, test\_data = train\_test\_split(ratings\_filtered, test\_size=0.2, random\_state=42)  print(f"Обучающая выборка: {len(train\_data)}")  print(f"Тестовая выборка: {len(test\_data)}")  def create\_user\_item\_matrix(df):      # Создание матрицы пользователь-книга      matrix = df.pivot\_table(index='User-ID', columns='ISBN', values='Book-Rating', fill\_value=0)      print(f"Матрица пользователь-книга: {matrix.shape}")      return matrix  def create\_item\_user\_matrix(df):      # Создание матрицы книга-пользователь      matrix = df.pivot\_table(index='ISBN', columns='User-ID', values='Book-Rating', fill\_value=0)      print(f"Матрица книга-пользователь: {matrix.shape}")      return matrix  def compute\_similarity(matrix, method='cosine'):      # Вычисление косинусной близости      sim = cosine\_similarity(matrix)      return pd.DataFrame(sim, index=matrix.index, columns=matrix.index)  # Создание матриц  user\_item\_train = create\_user\_item\_matrix(train\_data)  item\_user\_train = create\_item\_user\_matrix(train\_data)  # Вычисляем схожести  user\_similarity = compute\_similarity(user\_item\_train)  item\_similarity = compute\_similarity(item\_user\_train)  def user\_based\_predict(user\_id, book\_id, user\_item\_matrix, user\_sim\_matrix, k=10):      if user\_id not in user\_item\_matrix.index or book\_id not in user\_item\_matrix.columns:          return 0.0      # Похожие пользователи      sims = user\_sim\_matrix[user\_id].sort\_values(ascending=False)[1:k+1]      numerator = 0.0      denominator = 0.0      for neighbor\_id, sim in sims.items():          if sim <= 0:              continue          rating = user\_item\_matrix.loc[neighbor\_id, book\_id]          if rating == 0:              continue          numerator += sim \* rating          denominator += sim      return numerator / denominator if denominator > 0 else 0.0  def item\_based\_predict(user\_id, book\_id, item\_user\_matrix, item\_sim\_matrix, k=10):      if book\_id not in item\_user\_matrix.index or user\_id not in item\_user\_matrix.columns:          return 0.0      # Похожие книги      sims = item\_sim\_matrix[book\_id].sort\_values(ascending=False)[1:k+1]      numerator = 0.0      denominator = 0.0      for neighbor\_id, sim in sims.items():          if sim <= 0:              continue          rating = item\_user\_matrix.loc[neighbor\_id, user\_id]          if rating == 0:              continue          numerator += sim \* rating          denominator += sim      return numerator / denominator if denominator > 0 else 0.0  def user\_based\_recommend(user\_id, user\_item\_matrix, user\_sim\_matrix, n=10):      if user\_id not in user\_item\_matrix.index:          return []      user\_ratings = user\_item\_matrix.loc[user\_id]      candidates = user\_ratings[user\_ratings == 0].index[:100]  # ограничение      preds = [(book\_id, user\_based\_predict(user\_id, book\_id, user\_item\_matrix, user\_sim\_matrix)) for book\_id in candidates]      preds.sort(key=lambda x: x[1], reverse=True)      return preds[:n]  def item\_based\_recommend(user\_id, item\_user\_matrix, item\_sim\_matrix, n=10):      if user\_id not in item\_user\_matrix.columns:          return []      user\_ratings = item\_user\_matrix[user\_id]      candidates = user\_ratings[user\_ratings == 0].index[:100]      preds = [(book\_id, item\_based\_predict(user\_id, book\_id, item\_user\_matrix, item\_sim\_matrix)) for book\_id in candidates]      preds.sort(key=lambda x: x[1], reverse=True)      return preds[:n] |



**Рисунок 3 – Набор текста**

1. Оценка и интерпретация результатов

Результат работы:

Данную работу можете увидеть в блокноте Jupyter Notebook.

<https://drive.google.com/file/d/1o0rhZqLS5oI57PGN8sMn_clyT9e35E8Q/view?usp=sharing>

Вывод:

Научился парсить через API.

Список использованных источников и литературы:

1. Ростовцев В.С. Искусственные нейронные сети,   
   Издательство "Лань", 2019. — 216 с. — URL: https://e.lanbook.com/book/122180
2. Араки М. Манга: Машинное обучение,   
   Издательство "ДМК Пресс", 2020. — 214 с. — URL: <https://e.lanbook.com/book/179473>
3. Платонов, А. В. Машинное обучение : учебное пособие для вузов / А. В. Платонов. — Москва : Издательство Юрайт, 2022. — 85 с. — (Высшее образование). — ISBN 978-5-534-15561-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/508804