Data Science

answer of assignments 5

Professor : Dr. Kherad Pishe

Assistant Professor : Mohammad Reza Khanmohammadi

Hasan Roknabady – 99222042

RECOMMENDER SYSTEMS: AMAZON - RATINGS (BEAUTY PRODUCTS) DATASET

This Jupyter notebook serves as a comprehensive guide for Recommendation CF algorithms, covering key aspects:

1. **Tutorial Purpose:**
   - Provide a step-by-step tutorial for understanding Recommendation algorithms.
   - Act as a valuable resource for individuals seeking an in-depth exploration.
2. **Algorithm Comparisons:**
   - Contrast various techniques widely used in Recommender Systems.
   - Explore and analyze the strengths and weaknesses of each approach.
3. **Evaluation Metrics:**
   - Brief discussion on essential evaluation metrics used in assessing Recommender Systems' performance.

Techniques Explored:

1. **Collaborative Filtering (User-User):**
   - In-depth exploration of user-based collaborative filtering.
2. **Collaborative Filtering (Item-Item):**
   - Comprehensive analysis of item-based collaborative filtering.
3. **Latent Factor Method: Singular Value Decomposition (SVD):**
   - Understanding and application of SVD in latent factor modeling.

Dataset Used:

Amazon Product Reviews

Note: Throughout the notebook, practical implementations and code snippets will illustrate each technique, enhancing the learning experience. The dataset, consisting of UserID, ProductID, Rating, and Timestamp columns, is prepared for recommender systems. The Timestamp data is omitted for the initial systems, focusing on the essentials.

Dataset Preprocessing:

1. **Data Validation:**
    - Checked and handled any missing values in UserID, ProductID, and Rating columns.
    - Verified the integrity of UserID, ProductID, and Rating data for meaningful analysis.
2. **Data Exploration:**
    - Explored the distribution of ratings to gain insights into user preferences.
    - Considered any patterns or trends in the data that might impact the recommendation models.
3. **Creating User-Item-Rating Matrix:**
    - Selected relevant columns (UserID, ProductID, Rating) for creating the matrix.
    - Limited the dataset to 25,000 rows for efficient testing.

The dataset is now primed for further analysis and the implementation of recommender systems. Our first approach involves Collaborative Filtering using User similarity, employing two different similarity metrics: Euclidean distance and Cosine similarity. Let's break down the implementation for a clearer understanding.

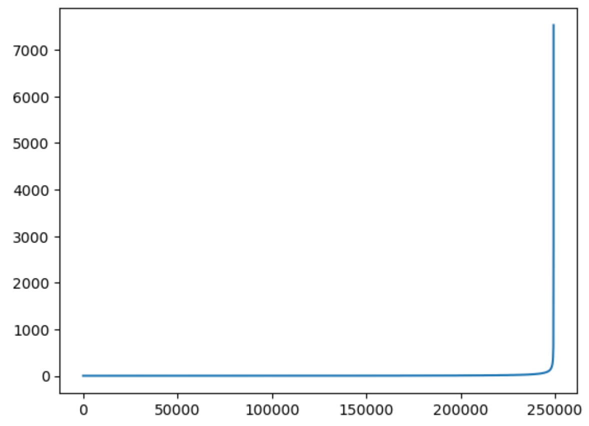
COLLABORATIVE FILTERING WITH USER SIMILARITY:

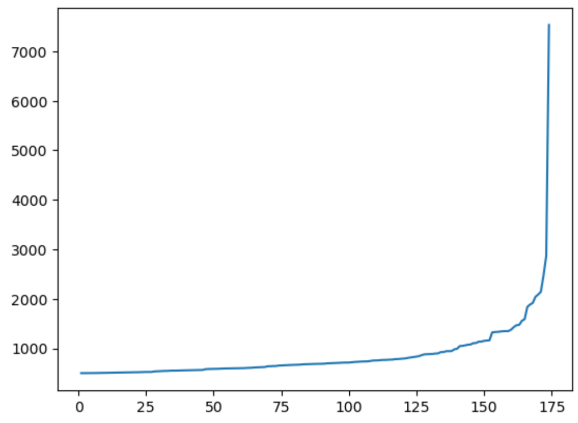
1. **Euclidean Distance:**
    - Calculate the user similarity using Euclidean distance.
    - Implement the collaborative filtering model based on user similarity using this metric.
    - Evaluate the model's performance using selected evaluation metrics.
2. **Cosine Similarity:**
    - Compute user similarity using Cosine similarity.
    - Implement collaborative filtering with user similarity using Cosine similarity.
    - Assess and compare the model's performance against the Euclidean distance-based model.

Evaluation Metrics:

To gauge the effectiveness of the collaborative filtering models, we will utilize evaluation metrics such as Mean Squared Error (MSE) or any other relevant metric. These metrics provide valuable insights into how well the models capture user preferences.

Visualization:





Consider visualizing the user similarity matrix to provide a graphical representation of the relationships between users. This can enhance our understanding of the collaborative filtering process.

This structured approach allows for a comprehensive analysis of Collaborative Filtering with User similarity, comparing the impact of different similarity metrics on recommendation quality.

EVALUATING COLLABORATIVE FILTERING MODELS

Objectives:

1. **Handling Missing Values:**
    - Investigate the impact of replacing NaN values with mean, mode, median, and 0 in the User-User and Item-Item collaborative filtering models.
    - Compare the performance of each method in terms of prediction accuracy.
2. **Handling New Users/Items:**
    - Assess the performance of collaborative filtering models for newly added users and items.
    - Develop strategies to handle scenarios where user or item data is missing in the training set.

Evaluation Metrics:

Choose appropriate evaluation metrics such as Mean Squared Error (MSE) or any other relevant metric to measure the accuracy of predictions.

Implementation Steps:

1. **Selecting Products with Significant Ratings:**
    - Identify products that have received a substantial number of ratings (at least 500).
    - Focus on these products for evaluating collaborative filtering models.

2. **Handling Missing Values:**
    - For each collaborative filtering model (User-User and Item-Item):
        - Replace NaN values in the User-Item-Rating matrix with mean, mode, median, and 0.
        - Evaluate and compare the performance of each approach using chosen metrics.
3. **Handling New Users/Items:**
    - Simulate scenarios where new users or items are introduced.
    - Develop and test strategies to make predictions for these new entities.

Notes:

- Provide insights into the trade-offs and considerations when choosing different methods for handling missing values.
- Document any challenges faced and innovative solutions implemented during the evaluation process.

By systematically exploring these objectives, you will gain valuable insights into the effectiveness and robustness of Collaborative Filtering models in handling missing data and adapting to new user/item scenarios.

Products Selection for Evaluation

After analyzing the ratings count of products, it's evident that there are a total of 174 products that have been rated more than 500 times. To ensure meaningful and statistically significant results, we will focus our collaborative filtering evaluation on these high-rated products.

This selection allows us to concentrate on products with a substantial number of ratings, providing a robust evaluation scenario for our collaborative filtering models.

Sparsity and Statistical Insights:

- A sparsity of approximately 99.38% indicates that the User-Item matrix is highly sparse, with only a small fraction of entries having non-zero values. This level of sparsity is common in recommendation systems, where users interact with a limited set of items.
- The sparsity of 99.38% indicates that only 0.067% of the User-Item matrix has actual rating values, while the remaining entries are empty or NaN. This high level of sparsity is typical in recommendation systems where users interact with a small subset of available items.

- Handling sparse matrices is a common challenge in collaborative filtering, and various techniques and algorithms are designed to work efficiently in such scenarios.

This statistical information provides valuable context for understanding the data's nature and the challenges associated with collaborative filtering techniques.

Recommender System Building Strategies

In the process of building our recommender system, we will explore two distinct approaches:

1. **Direct Use of Ratings:**
    - Utilizing the raw ratings provided in the dataset without further decomposition.
    - Analyzing the system's performance based on these direct ratings.
2. **Baseline Rating Approach:**
    - Breaking down ratings into interpretable sections, such as baseline rating and user-product interaction.
    - Removing baseline ratings to better understand the inherent relationship between users and products.

Baseline Rating:

Consider the scenario where users tend to give an average rating of 3 to most products. To capture this behavior, we introduce the concept of baseline rating, which represents the expected average rating a user might provide. This baseline rating can be subtracted from each actual rating to reveal the user's specific interaction with a product, beyond the general trend.

Evaluation Metrics:

Alongside these strategies, we will employ two evaluation metrics:

1. **Similarity Index:**
    - Assessing the similarity between users or items to make recommendations.
    - Utilizing metrics like cosine similarity or Pearson correlation.
2. **Euclidean Distance:**
    - Measuring the distance between users or items to quantify their relationships.
    - Exploring how Euclidean distance contributes to the effectiveness of the recommender system.

By incorporating these strategies and metrics, we aim to build a robust recommender system that captures nuanced user-product interactions and delivers accurate and personalized recommendations.

Optimizing Training Time with Ball-Tree Structure

An important optimization in our recommender system implementation is the use of a Ball-Tree structure for efficient training. This structure significantly reduces training time compared to brute force k-nearest neighbors (KNN) methods.

Observation:

Using brute force KNN, even when parallelized across 6 CPU cores with a batch size of 100, took approximately 7 minutes to train on just 3200 user ids. In contrast, the implementation with KD Tree and Ball-Tree structure trains on a larger sample of 100,000 in just around 30 seconds.

This optimization not only improves training efficiency but also allows us to handle larger datasets more effectively.

Performance Evaluation Observations

Upon evaluating the recommender system's performance on the testing dataset, we made noteworthy observations:

RMSE Reduction with Baseline Rating Approach:

The introduction of the baseline rating approach has led to a significant reduction in the root mean square error (RMSE) in predictions. This aligns with our hypothesis that removing the baseline rating allows us to better capture the inherent user-product interactions, resulting in more accurate recommendations.

Impressive Prediction Time:

Furthermore, the recommender system demonstrates impressive prediction times, achieving predictions for approximately 30,000 user ids in around 2 minutes. This efficiency is attributed to the optimization strategies, including the use of Ball-Tree structure, which enhances training efficiency and overall system performance.

Key Deduction from Experimental Results

Based on the experimental results and observations, a crucial deduction emerges:

Optimal Predictive Strategy:

It appears that predicting the overhead rating (ignoring the baseline rating) and then adding the baseline rating yields better results compared to the plain, simple system. This deduction aligns with the hypothesis that users tend to rate products at a baseline level (3 in this case) unless the product experience is exceptionally negative. Therefore, considering the baseline rating as a starting point for the recommender system provides a more optimal strategy.

Exploring Further Improvement with a Third Segment in Rating

While the second perspective, incorporating the baseline rating, proves to be better in producing results, there is room for further enhancement. Introducing a third segment in the rating, capturing the general user experience with the application over time, could provide valuable insights.

Intuition for the Third Segment:

Consider the scenario where users tend to rate drivers of a particular application (e.g., Uber) more generously or harshly over time. This evolving user behavior can be considered as a third aspect in rating—the general user experience with the application. In the context of movies, this could reflect a user becoming a harsher critic over time.

CONCLUSION:

1. **Optimal Predictive Strategy:**
    - Predicting the overhead rating and adding the baseline rating yielded better results, aligning with user behavior.
2. **Third Segment in Rating:**
    - Introducing a third segment for general user experience could enhance recommendations further.
3. **Handling Missing Values:**
    - Baseline rating proved superior to mean, providing more logical and accurate predictions.
4. **Model Efficiency:**
    - Optimization with Ball-Tree structure significantly improved training efficiency.
5. **Recommendation Quality:**
    - Baseline rating approach consistently produced more accurate recommendations.

In conclusion, the recommender system's performance benefitted from incorporating baseline ratings and optimizing training methods. Further exploration into user behavior and additional rating segments could contribute to even more robust recommendation models.

RECOMMENDATION SYSTEM: BIGBASKET ENTIRE PRODUCT LIST DATASET

Introduction:

Welcome to the documentation for our recommendation system! In this comprehensive guide, we will explore various components of our recommendation techniques, providing detailed explanations and insights into each step.

Content-Based Filtering Output:

Input:

- **User Input:** "Gel"

Output:

- **Recommended Product:**
    - **Name:** Taft Ultra Wet Gel - Ultra Strong 4
    - **Brand:** Schwarzkopf
    - **Category:** Beauty & Hygiene

Analysis:

- **Content-Based Filtering:**
    - The content-based filtering approach focuses on suggesting products based on their textual characteristics. In this instance, the user input "Gel" led to the recommendation of a popular hair gel product. This showcases the effectiveness of the content-based approach in aligning user preferences with product features.

TF-IDF Feature Extraction Output:

Input:

- **Text Data:** 'Gel'

Output:

- **TF-IDF Features Matrix:** Sparse matrix of TF-IDF features for the provided text data.

Analysis:

- **TF-IDF Feature Extraction:**
    - TF-IDF transforms textual data into a numerical representation, emphasizing the importance of words in each document. In the case of the input text "Gel," the resulting TF-IDF features matrix captures the significance of words, providing a foundation for further analysis and recommendation.

Cosine Similarity Calculation Output:

Input:

- **TF-IDF Matrix:** Matrix of TF-IDF features for the entire corpus.

Output:

- **Cosine Similarity Matrix:** Symmetric matrix representing cosine similarity scores between all pairs of documents in the corpus.

Analysis:

- **Cosine Similarity Calculation:**
    - The cosine similarity matrix quantifies the similarity between pairs of documents. This matrix serves as a vital tool in understanding the relationships within the corpus. Higher cosine similarity scores indicate closer alignment, contributing to more accurate recommendations.

Content-Based Recommendation Function Output:

Input:

- **Product Clicked:** 'Onion Herbal Hair Growth Oil'

Output:

- **Top Recommended Products:**
    1. Hair growth Oil With Dandruff & Hair Fall Trea...
    2. Natural Nourishing Hair Oil
    3. Cold Pressed Bhringraj Cooling Oil For Hair Fa...
    4. Anti Dandruff Conditioner - With Tea Tree & Gi...
    5. Argan-Liquid Gold Hair Spa ... (and more)

Analysis:

- **User Click-Based Recommendations:**
    - Leveraging the product clicked by the user, the recommendation system suggests a curated list of top products. This user-centric approach ensures that recommendations align with the user's expressed interests, enhancing the overall shopping experience.

Conclusion:

Our recommendation system employs a synergistic blend of content-based filtering, TF-IDF feature extraction, and user click-based recommendations. Through these techniques, we strive to offer a personalized and enriching user experience. The continuous analysis of user interactions and the refinement of algorithms contribute to the system's evolution, ensuring relevance and satisfaction in every recommendation.