# Clothing Recommendation & Rating Prediction

Adrian Apsay, Sean Isayama, Ryan Izadshenas and Julia Jung
University of California San Diego
adapsay@ucsd.edu | hisayama@ucsd.edu | rizadshenas@ucsd.edu | jmjung@ucsd.edu

## 1 INTRODUCTION

### 1.1 Dataset Overview

The dataset we have chosen for our analysis is the Clothing Fit Data from Rent the Runway[1], part of the Recommender Systems Datasets[2]. This dataset includes essential features such as 'user_id', 'body type', 'category', 'age', 'review_text', and 'size', which will be used to recommend items based on 'item_id'. Before developing the recommendation models, we will perform exploratory data analysis to understand the dataset's structure and identify key features.

### 1.2 Key Features in the Dataset

This dataset includes the data on ratings and reviews, fit feedback, user/item measurements, and category information with the following being an example of one data point from the dataset:


```
{
  "fit": "fit",
  "user_id": "420272",
  "bust size": "34d",
  "item_id": "2260466",
  "weight": "137lbs",
  "rating": "10",
  "rented for": "vacation",
  "review_text": "An adorable romper! Belt and
zipper were a little hard to navigate in a full day
of wear/bathroom use, but that's to be expected.
Wish it had pockets, but other than that--
absolutely perfect! I got a million compliments.",
  "body type": "hourglass",
  "review_summary": "So many compliments!",
  "category": "romper",
  "height": "5' 8\"",
  "size": 14,
  "age": "28",
  "review_date": "April 20, 2016"
}
```


The 'user_id' and 'item_id' will be essential when recommending items to users, while the 'fit', 'weight', 'rating', 'rented for', 'body type', 'category', 'height', 'size', and 'age' features will be used to provide essential details about the user, allowing for feature-based comparisons. To ensure effective use of these features, we converted these features to appropriate data types (ages, weight, rating, to integers, etc.), enabling efficient analysis and model performance. The 'review_text' and 'review_summary' features will also provide qualitative insights into user satisfaction and preferences, where NLP and sentiment analysis on these fields can uncover valuable themes and sentiments, enhancing personalization and refining recommendations.

One feature that we expect will be significant for our recommendations is 'rating' as it directly reflects user satisfaction and is often a key indicator of a product's quality or performance. Additionally, ratings can provide valuable insights into user preferences and trends, helping to inform product improvements or marketing strategies.

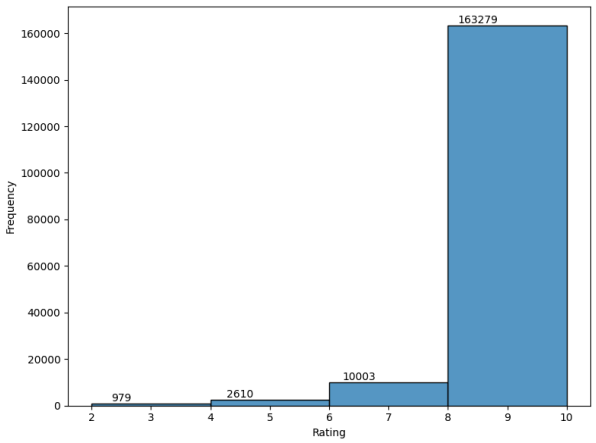



***Figure 1:*** *Distribution of user ratings, showing frequency across values.*

Another feature that we expect will be significant is 'body type', as body type can influence product fit, comfort, and style preferences, which are crucial factors in customer satisfaction. Understanding the distribution of body types

[1]https://www.renttherunway.com/
[2]https://cseweb.ucsd.edu/~jmcauley/datasets.html#clothing_fit

can also help tailor recommendations and improve personalization for users.

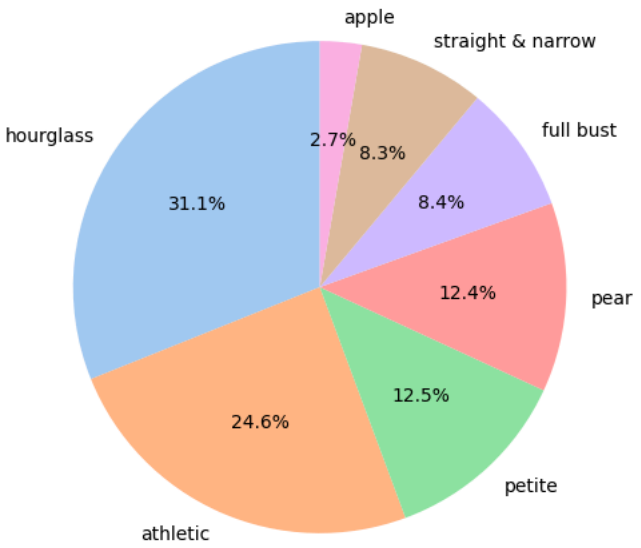



***Figure 2:*** *Distribution of body type by proportion*

## 1.3 Missing Data

This dataset contains several features with missing values.

| Ratings | # of Null values |
|---|---|
| fit | 0 |
| user_id | 0 |
| bust size | 18,411 |
| item_id | 0 |
| weight | 29,982 |
| rating | 82 |
| rented for | 10 |
| review_text | 0 |
| body type | 14637 |
| review_summary | 0 |
| category | 0 |
| height | 677 |
| size | 0 |
| age | 960 |
| review_date | 0 |
| Total # of rows: 192,544 | |
| Total # of rows with null values: 46,163 | |

**Table 1: Rent the Runway Dataset**

To address the issue of missing values in the dataset (Table 1), we utilized two primary strategies tailored to the nature and importance of the features. For features deemed critical to the analysis or model performance (rating, rented for, etc.), but with relatively low proportions of missing values, we chose to drop rows containing null values. This approach ensured the dataset remained robust without introducing potential bias. On the other hand, for features with a high number of missing values (weight, height, etc.), where dropping rows would lead to significant data loss, we applied mean imputation. By replacing null values with the feature's mean, we preserved its overall distribution and statistical properties while effectively reducing the total number of null values. As for null values in categorical features and the bust size column, we decided to drop the feature and null rows as it is still a small proportion of the dataset and there are better features for item recommendation. Together, these methods helped significantly clean the dataset, preparing it for further analysis (Table 2).

| Features | # of Null values |
|---|---|
| fit | 0 |
| user_id | 0 |
| item_id | 0 |
| weight | 0 |
| rating | 0 |
| rented far | 0 |
| body type | 0 |
| category | 0 |
| height (in) | 0 |
| size | 0 |
| age | 0 |
| Total # of rows: 176871 | |
| Total # of rows with null values: 0 | |

**Table 2: Task 1 (Recommendations) Dataset**

## 1.4 Interesting Findings

One observation that we found interesting was that the distribution of 'ratings' is heavily right-skewed, with the majority of ratings being 10s (Figure 1). This suggests that most users are highly satisfied with the product, and there may be fewer instances of extreme dissatisfaction, which could indicate a generally positive reception or a potential bias towards leaving positive feedback. Another interesting observation is that the 'body type' column contains only 7 unique values, indicating a well-defined set of categories (Figure 2). This suggests that the data is organized into

distinct groups, which could streamline the modeling process. At the same time, it provides a valuable opportunity to explore meaningful patterns and relationships within these categories.

| | count | mean | max | min | std |
|---|---|---|---|---|---|
| **weight** | 176871 | 137.494 | 300.0 | 50.0 | 20.63290 |
| **height (in)** | 176871 | 65.3041 | 78.0 | 54.0 | 2.668024 |
| **size** | 176871 | 12.2182 | 58.0 | 0.0 | 8.480389 |
| **age** | 176871 | 33.9635 | 117.0 | 0.0 | 8.057023 |

**Table 3: Summary Statistics**

It is important to note that we are performing two different predictive tasks (task 1, task 2) which will be iterated on later. One of our predictive tasks is to predict the rating of an item based on 'review_text' and 'review_summary'. This did not require as much data cleaning as the dataset imposed no null values for both features. However, our target variable, 'rating', had 82 null values (Table 1). Due to the relatively small number of null values compared to the overall dataset size of over 190,000 rows, their impact was deemed negligible, so we decided to drop them (Table 4).

| **Features** | **# of Null values** |
|---|---|
| review_text | 0 |
| review_summary | 0 |
| rating | 0 |
| Total # of rows: 192462 | |
| Total # of rows with null values: 0 | |

**Table 4: Task 2 (Rating Prediction) Dataset**

# 2 Predictive Task and Model Development

## 2.1 Task 1: Recommender Systems

For our first task, the goal is to develop systems to recommend clothing items to users based on their past ratings, interactions, and personal attributes in the dataset. We will use various features, including user-related attributes ( 'user_id', 'age', 'body type', 'height (in)', 'rating', 'size', etc.), and item-related features ('item_id', 'category', 'fit', 'rented for', etc.). To achieve this, we will implement two recommendation approaches:

1. Using **Jaccard Similarity**, and
2. Using **Item2Vec**

to calculate the similarity scores. We decided to use Jaccard Similarity as one of our approaches as it is a simple yet effective method for measuring the similarity between items based on user interactions, making it useful for identifying items with overlapping user preferences for this dataset. On the other hand, Item2Vec allows us to capture deeper relationships between items by embedding them into a vector space, enabling the model to learn latent features from user behavior and item attributes, providing a more powerful and scalable approach for recommendation tasks.

## 2.2 Task 2: Rating Prediction

The goal of this task is to predict a user's rating (10, 8, 6, 4, or 2) for items they have rented, using 'review_text' and 'review_summary' as key features. Ratings provide insights into user sentiment about specific items, and predicting these values can guide recommendations and user feedback analysis. Since there are only five possible outcomes, we frame this as a multi-class classification problem rather than a regression task. During data processing, null values in the rating column were removed, and text data was preprocessed by removing stop words and punctuation. We utilize Word2Vec to generate vector embeddings from 'review_text' and 'review_summary', converting textual data into numerical representations. These embeddings are combined into a unified feature set to incorporate insights from both sources. As a baseline, we use Decision Trees for their simplicity and interpretability. We will then adopt Random Forest, a more robust ensemble model, to enhance predictive performance through generalizing by aggregating multiple decision trees.

## 2.3 Model Evaluation / Validity of Predictions

**Predictive Task 1:** To assess the validity of recommendations made by our models, we use Precision@K and Recall@K, two key metrics for understanding how well the recommendations align with user preferences. Precision@K measures how many of the top K recommended items are actually relevant, focusing on the accuracy of the suggestions. Recall@K, on the other hand, measures how many of the relevant items for a user

are included in the top K recommendations, capturing the model's ability to cover what users might like. These metrics are especially helpful in recommendation systems, where it's important to balance accuracy with providing a diverse range of meaningful options they potentially haven't rented prior. By focusing on these practical and user-focused measures, we can better understand how effective our model is in delivering personalized and impactful recommendations.

**Predictive Task 2:** To evaluate the performance of Decision Trees and Random Forests for predicting ratings, we will focus on accuracy, precision, recall, and F1-score as key metrics. Accuracy provides an overall measure of correctly predicted ratings, while precision and recall highlight the model's ability to correctly predict specific rating classes, particularly for minority ratings (imbalanced classes that have small occurrences). F1-score, as the harmonic mean of precision and recall, balances these two aspects, offering a robust evaluation metric for imbalanced datasets. Since the rating classes are imbalanced, we will also report macro-averaged metrics to ensure fair evaluation across all rating categories. These metrics will help determine how effectively the models predict ratings across all five classes (10, 8, 6, 4, and 2) based on review text data. The models will be tested on a separate test set that was not used during training to ensure generalizability to new data.

### 2.4 Baselines for Comparison

For our task 1 baselines, we will use the following to compare with our models:

1. A **Random Recommender**, which randomly suggests items to users, without considering any past behavior, ratings, or item features.
2. A **Popularity-Based Recommender**, which recommends the most popular items based on average ratings.

These baseline models will serve as reference points to gauge the performance of the proposed models. For task 2, we will use a Decision Tree classifier as a baseline model because it is simple, interpretable, and provides a good starting point for understanding the relationships in the data before exploring more complex models.

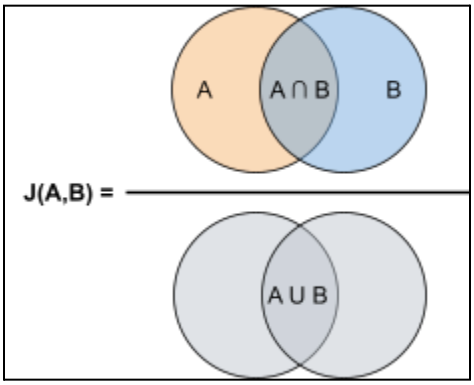
Following these model evaluations, we will also validate our results by ensuring that the model's predictions align with real-world expectations and generalize well to unseen data. Additionally, we will assess the model for overfitting and bias, examining if it is making consistent, unbiased predictions across various scenarios and not just memorizing the training data.

## 3 Model Description and Optimization

### 3.1 Model Design

For our first task, we will use Jaccard Similarity (1) to calculate the similarity scores for our recommender system.. This approach calculates the similarity between items based on shared attributes, and recommends items that are most similar to those a user has interacted with.

$$Jaccard(s1, s2) = \frac{|s1 \cap s2|}{|s1 \cup s2|} \quad (1)$$




Next, we will use Item2Vec, which is an adaptation of the Word2Vec model from natural language processing, tailored for recommender systems. Item2Vec generates vector embeddings for items (e.g., clothing products) based on patterns of co-occurrence in user interactions, such as renting history. These embeddings represent items in a continuous vector space, capturing their latent similarities. Recommendations are generated by retrieving the top five items most similar to a given item (e.g., one the user rented) using cosine similarity (2) in the embedding space.

$$Cosine(A, B) = cos(\Theta) = \frac{AB}{||A||||B||} = \frac{\Sigma AB}{\sqrt{\Sigma A^2 \Sigma B^2}} \quad (2)$$

Using these two models tailored for recommendations and tuned for optimized performance (Figure 3, Figure 4), we will compare the performances against our baseline models.

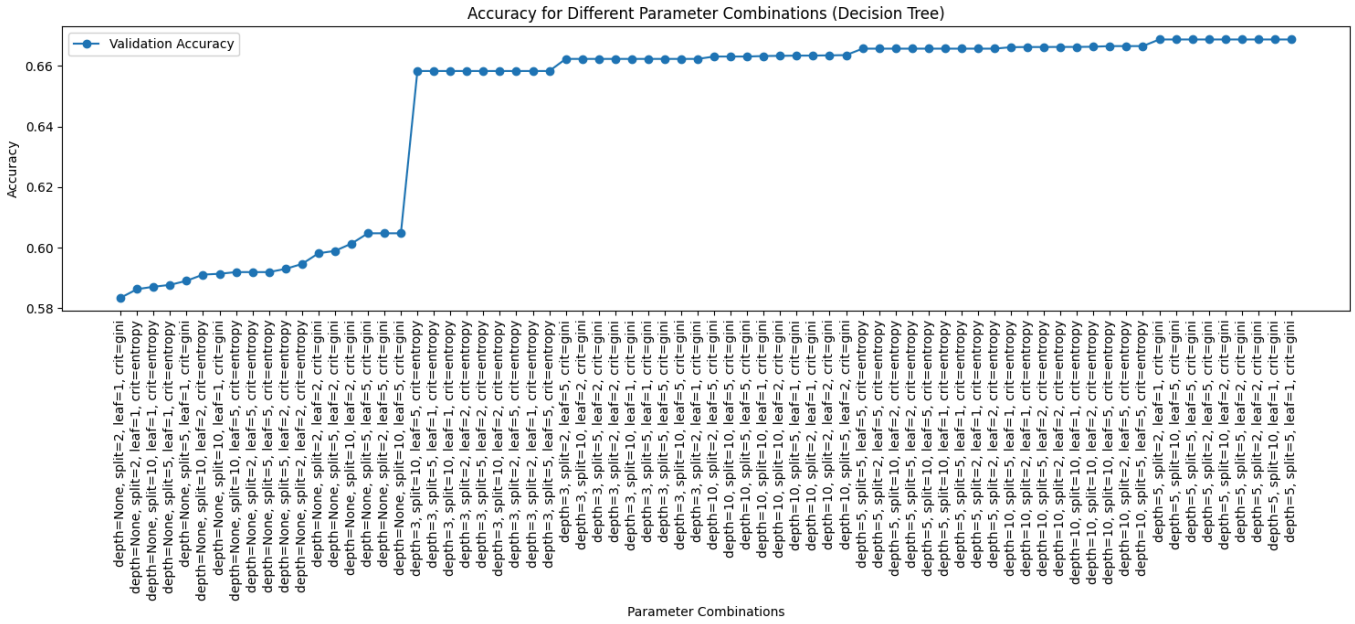



***Figure 3:*** *Decision Tree model accuracy, optimized using different parameter combinations*

| Ratings | Precision | Recall | f1 -score | support |
|---|---|---|---|---|
| 10 | 0.71 | 0.92 | 0.80 | 24818 |
| 8 | 0.44 | 0.26 | 0.33 | 10763 |
| 6 | 0.00 | 0.00 | 0.00 | 2146 |
| 4 | 0.00 | 0.00 | 0.00 | 549 |
| 2 | 0.00 | 0.00 | 0.00 | 217 |
| accuracy | … | … | 0.67 | 38493 |
| marco avg | 0.23 | 0.24 | 0.23 | 38493 |
| weighted avg | 0.58 | 0.67 | 0.61 | 38493 |

**Table 5: Testing Evaluation Metrics for Tuned Decision Tree**

For our second task of predicting rating, we decided to implement text mining with a Word2Vec model to convert 'review_text' and 'review_summary' to a numerical embedding space. Word2Vec learns vector representations (embeddings) of words from large text corpora. These embeddings capture semantic relationships between words such that similar words (in meaning or usage) are close together in the vector space. We then decided to use a Decision Tree along with Random Forest classification to create a multi-class classifier that can predict a user's rating for a product (either 10, 8, 6, 4, 2) given their literal text review and their review summary in the form of embeddings.

## 3.2 Justification for Model Choice and Optimization

The chosen recommender models—Jaccard Similarity and the Item2Vec—were selected to address both explicit and implicit user preferences. Jaccard Similarity is a simple yet effective method to measure the overlap between user interactions and item attributes, making it suitable for quickly identifying similar items. Item2Vec provides deeper insights by learning vector representations of items based on user renting patterns, capturing subtle relationships between items. Optimization strategies for these models include hyperparameter tuning, such as

adjusting the vector size in Item2Vec, ultimately choosing a vector size of 100 to be our hyperparameter. To optimize our Jaccard recommender, we incorporated popularity scores and enhanced the model's performance by assigning weights to both the Jaccard similarity and popularity values, thereby maximizing the effectiveness of the recommendations.

The chosen rating prediction models—Decision Trees and Random Forest using Word2Vec (for embeddings)—were selected for a variety of reasons. Word2Vec excels at capturing semantic and contextual relationships between words by mapping them into dense vector spaces. For text reviews, this is critical, as similar words (e.g., "comfortable" and "cozy") are represented similarly, which helps in understanding the sentiment and nuances of user feedback. It is useful to convert high-dimensional text into compact embeddings for ease of use with classification models. While Word2Vec captures complex linguistic patterns, decision trees and random forests provide interpretable decision rules that can explain why a particular rating was predicted (e.g., based on the presence of certain keywords in the text), and the forests can scale well to datasets of moderate sizes. More concisely, decision trees and random forests were chosen for their interpretability and robustness. Decision trees were chosen as the baseline model due to their simplicity and interpretability as it provides a foundational understanding of how features influence ratings. Random forests, as the primary model, enhance predictive performance by combining multiple trees to improve generalization, making them well-suited for handling the complexity of the text embeddings generated by Word2Vec.

| Ratings | Precision | Recall | f1 -score | support |
|---|---|---|---|---|
| 10 | 0.72 | 0.96 | 0.83 | 24818 |
| 8 | 0.53 | 0.24 | 0.33 | 10763 |
| 6 | 0.43 | 0.02 | 0.03 | 549 |
| 4 | 0.46 | 0.15 | 0.23 | 2146 |
| 2 | 0.67 | 0.01 | 0.02 | 217 |
| accuracy | | | 0.69 | 38493 |
| marco avg | 0.56 | 0.28 | 0.29 | 38493 |
| weighted avg | 0.65 | 0.69 | 0.64 | 38493 |

**Table 6: Testing Evaluation Metrics for Random Forest**

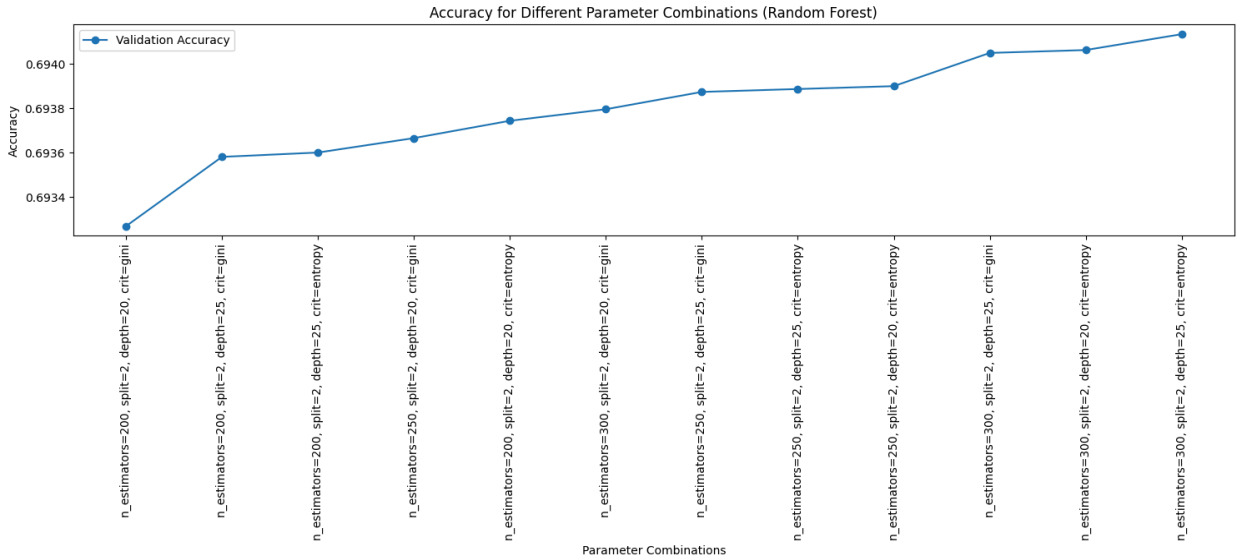



***Figure 4:*** *Random Forest model accuracy, optimized using different parameter combinations*

### 3.3 Scalability, Overfitting, and Challenges

Overfitting is a critical challenge when building models, especially with complex models like decision trees and random forests. Overfitting occurs when a model learns the noise and specific patterns in the training data rather than the general patterns that are likely to apply to unseen data. This is evident in our task #2, where both the decision tree and random forest models achieve nearly perfect accuracy on the training set (Table 7, 8) but perform poorly on the test set (Table 5, 6), particularly for minority ratings. This obstacle highlights that the models are failing to generalize to new data, despite performing grid search and cross-validation to tune hyperparameters. There are numerous factors contributing to overfitting, which includes the high dimensionality of text embeddings, the imbalanced distribution of ratings (as seen in the support column of our classification report), and the complexity of the models. Random Forests, despite being less prone to overfitting than single Decision Trees due to ensemble averaging, can still overfit when the trees in the forest are too deep or when there are too many features. In this case, we are working with many features, as each embedding corresponds to 200 features. We can address these overfitting issues through strategies like limiting tree depth, balancing the dataset by aggregating data, using simpler baseline models for comparison, and using lower vector sizes for our embeddings.

| Ratings | Precision | Recall | f1 -score | support |
|---|---|---|---|---|
| 10 | 1.00 | 1.00 | 1.00 | 99719 |
| 8 | 1.00 | 1.00 | 1.00 | 42628 |
| 6 | 1.00 | 1.00 | 1.00 | 2242 |
| 4 | 1.00 | 1.00 | 1.00 | 8551 |
| 2 | 1.00 | 1.00 | 1.00 | 829 |
| accuracy | … | … | 1.00 | 153969 |
| marco avg | 1.00 | 1.00 | 1.00 | 153969 |
| weighted avg | 1.00 | 1.00 | 1.00 | 153969 |

**Table 7: Training Evaluation Metrics for Tuned Decision Tree**

| Ratings | Precision | Recall | f1 -score | support |
|---|---|---|---|---|
| 10 | 1.00 | 1.00 | 1.00 | 99719 |
| 8 | 1.00 | 1.00 | 1.00 | 42628 |
| 6 | 1.00 | 1.00 | 1.00 | 2242 |
| 4 | 1.00 | 1.00 | 1.00 | 8551 |
| 2 | 1.00 | 1.00 | 1.00 | 829 |
| accuracy | | | 1.00 | 153969 |
| marco avg | 1.00 | 1.00 | 1.00 | 153969 |
| weighted avg | 1.00 | 1.00 | 1.00 | 153969 |

**Table 8: Training Evaluation Metrics for Random Forest**

### 3.4 Comparison with Other Models and Evaluation

The evaluation metrics for our models reveal a clear tradeoff between baseline simplicity and the complexity of optimized models. Regarding the recommendation system, there is a clear improvement in precision and recall between our baseline and optimized models (Table 9). Although the precision and recall values seem to be low, we determined that the drastic improvement compared to our baseline models demonstrates a clear enhancement in the system's ability to make relevant recommendations. This indicates that, while the models may still have room for improvement, the optimized approaches significantly outperform the simpler baseline models, showcasing their potential for more accurate and effective recommendations in the future.

| Recommendation | Precision@K | Recall@K |
|---|---|---|
| **Random** (B) | 0.003 | 0.0053 |
| **Popularity** (B) | 0.02 | 0.0789 |
| **Jaccard** (O) | 0.3251 | 0.6842 |
| **Item2Vec** (O) | 0.231 | 0.6335 |

**Table 9: Task 1 Results**
(B = Baseline, O = Optimized, K = 5)

Regarding the rating prediction task, we can see a slight improvement in performances when comparing our baseline Decision Tree model to our optimized Random Forest model (Table 5, 6). While Random Forest improved predictive power over Decision Trees due to its ensemble approach, addressing overfitting and imbalanced data remains a weakness.

# 4 Related Work and Literature Review

## 4.1 Existing Dataset Usage

As mentioned previously, the dataset used in this project originates from RentTheRunway, a unique platform that allows women to rent clothing for various occasions. This dataset has been previously utilized to create a predictive framework for product size recommendations by factoring the semantics of customers' fit feedback [1]. This approach enables the system to capture customers' preferences regarding specific product aspects, such as shoulders and waist fit, to provide more accurate and personalized recommendations.

## 4.2 Related Literature

Beyond this dataset, other efforts in clothing recommendation have utilized similar data to address related problems. For instance, FashionNet utilizes a deep neural network with a two-stage training approach: first training a general model for outfit compatibility, followed by user-specific fine-tuning to adapt to individual preferences[2]. This method improves recommendation accuracy by leveraging both shared and unique user data, with the network trained on compatibility rules such as matching tops and bottoms (e.g., t-shirts with jeans) to address the insufficiency of individual data.

Another project introduces a hybrid recommendation system that combines visual analysis and user interaction data to enhance personalized clothing suggestions[3]. Using convolutional neural networks (CNNs), the system extracts visual features such as color and texture from images to generate style-aware embeddings. By integrating collaborative and content-based filtering, it addresses challenges like the cold start problem and offers versatile applications, including occasion-specific recommendations.

A comprehensive study, "Fashion Recommendation Systems, Models and Methods: A Review," highlights the evolution of fashion recommender systems. This review explores content-based and collaborative filtering approaches and the integration of hybrid models. The paper emphasizes the growing role of artificial intelligence, particularly machine learning and deep learning, in analyzing user preferences and product attributes to enhance recommendation quality[4].

## 4.3 State-of-the-Art Methods

Some state-of-the-art methods for clothing recommendations include collaborative filtering techniques such as matrix factorization (e.g., SVD and ALS) and neighborhood-based approaches, which rely on user or item similarities. Content-based filtering methods utilize detailed user and item metadata, such as body type, size, and item categories, to align recommendations with user preferences. Hybrid systems combine these approaches to enhance predictive power. Advanced techniques such as deep learning, including deep neural networks (DNNs), recurrent neural networks (RNNs), and convolutional neural networks (CNNs), are increasingly employed to process user-item interactions, sequential data, and visual features of clothing [2,3]. Additionally, natural language processing (NLP) techniques analyze review text to extract sentiments and stylistic preferences, further refining recommendations.

## 4.4 Conclusions from Existing Work

The differences in findings between our work and existing studies stem largely from the methodologies used to build the recommendation systems. While many of the reviewed studies rely on deep learning techniques like convolutional neural networks (CNNs) to analyze visual features of clothing images for personalized recommendations, we chose a different approach [2, 3]. We used similarity scores and random forests that were fine-tuned to recommend items, focusing on matching user preferences with similar items based on past interactions and characteristics. This approach does not require the computational complexity of deep learning models, but still effectively captures user preferences using simpler models.

Despite these methodological differences, the core motivation behind both our work and the reviewed studies remains the same: to enhance the user experience by providing relevant and personalized clothing

recommendations. This goal drives the development of systems that can predict user preferences, improve satisfaction, and solve challenges like the cold start problem, as seen in both the existing literature and our findings. While deep learning models offer powerful tools for feature extraction and prediction, our choice of similarity-based methods and decision trees reflects a preference for interpretability and simpler computational requirements, which can still deliver effective results in personalized recommendation tasks.

# 5 Results, Analysis, and Conclusion

For task 1 (recommender systems), we explored recommendation systems using Jaccard Similarity and Item2Vec models. The results showed that Item2Vec achieved an Average Precision@5 of 0.231 and Recall@5 of 0.633, outperforming both the baseline Random and Popularity recommenders. The Jaccard Similarity model, while also effective, performed slightly better than Item2Vec in terms of Recall@5 (0.684) but fell short on Precision@5 (0.325). These results highlight the strengths of Item2Vec in capturing latent item relationships and its ability to generate meaningful recommendations. However, both models struggled with cold-start users (users with very few interactions), a common challenge in recommender systems. The baselines, while simple, provided useful benchmarks to gauge the effectiveness of our optimized models. Overall, the results emphasize the importance of leveraging user-item interaction data and embedding-based methods to improve recommendation accuracy.

For task 2 (rating prediction), we used text embeddings generated by Word2Vec and applied Decision Trees as a baseline and Random Forest as the primary classification model. While Random Forest achieved a higher overall accuracy (0.69) than Decision Trees (0.67), both models exhibited overfitting, with near-perfect performance on the training set and diminished performance on the test set, particularly for minority ratings like 2, 4, and 6. This reflects challenges in generalizing from an imbalanced dataset, where certain classes are underrepresented. Despite these limitations, the integration of Word2Vec embeddings effectively transformed textual data into numerical features, demonstrating the potential for text-based predictors in rating prediction tasks. In conclusion, while the models for both tasks showed promise, future work should focus on mitigating overfitting, addressing class imbalance, and exploring additional features to improve generalization and overall performance.

## REFERENCES


[1] Misra, R., Wan, M., & McAuley, J. (2018). Decomposing fit semantics for product size recommendation in metric spaces. Decomposing Fit Semantics for Product Size Recommendation in Metric Spaces. https://doi.org/10.1145/3240323.3240398

[2] He, T., & Hu, Y. (2024, March 2). FashionNet: Personalized Outfit Recommendation with Deep Neural Network. https://ar5iv.org/html/1810.02443

[3] Rathod, D., Tundare, A., Shelke, P., Muneshwar, S., & Dawkhar, S. (2024). Fashion recommendation system using deep learning. International Journal of Research Publication and Reviews, 5(5), ISSN 2582-7421. https://ijrpr.com/uploads/V5ISSUE5/IJRPR27038.pdf

[4] Chakraborty, S.; Hoque, M.S.; Rahman Jeem, N.; Biswas, M.C.; Bardhan, D.; Lobaton, E. Fashion Recommendation Systems, Models and Methods: A Review. Informatics 2021, 8, 49. https://doi.org/10.3390/info