VIETNAM NATIONAL UNIVERSITY OF HOCHIMINH CITY

THE INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

MACHINE LEARNING BASED SOLUTIONS FOR FIT CLOTHING SIZE RECOMMENDATION

By

Tạ Thị Phương Thảo - ITDSIU20082

A thesis submitted to the School of Computer Science and Engineering

In partial fulfillment of the requirements for the degree of

Bachelor of Computer Science

Ho Chi Minh City, Viet Nam

2024-2025

MACHINE LEARNING BASED SOLUTIONS FOR FIT CLOTHING SIZE RECOMMENDATION

SUPERVISED BY:

________________________,

Ph.D. Nguyễn Thị Thanh Sang.

ACKNOWLEDGEMENT

This thesis marks my initial endeavor toward completing the final requirement for the Bachelor’s Degree in Data Science at the School of Computer Science and Engineering. I am deeply grateful to Dr. Nguyen Thi Thanh Sang for her invaluable support and guidance throughout the semester. Her willingness to supervise my work and her insightful advice whenever I faced challenges have been instrumental in shaping this thesis proposal.

I also want to extend my heartfelt thanks to my friends, who have stood by me throughout my journey at this university. Their encouragement and unwavering support during difficult times have made my academic experience far more enjoyable and meaningful. I truly treasure the memories and friendships we have built together.

Working on this proposal has been a rewarding process, as it reflects my ability to apply the knowledge I have acquired to create something of value.

LIST OF TABLE

LIST OF FIGURES

ABBREVIATION

Notation	Description
TF-IDF	Term Frequency - Inverse Document Frequency
RS	Recommendation system
DT	Decision Tree
RF	Random Forest
MMR	Maximal Marginal Relevance
NLTK	Natural Language Toolkit
MAE	Mean Absolute Error
MSE	Mean Squared Error
TSS	Total Sum of Squares
RSS	Residual Sum of Squares
ASW	Average Silhouette Width

ABSTRACT

In recent years, the convenience of e-commerce has driven its rapid growth. Shopping online eliminates the need to visit physical stores, allowing individuals to save time and effort. However, one challenge remains: finding the right size and fit without trying items on. This often leads to consumers ordering incorrect sizes, resulting in returns. Such returns not only create inconvenience for users but also increase costs and reduce revenue for companies.

This research holds significance to analyze various personal attributes and associates them with specific items and sizes to provide consumers with accurate size recommendations. Additionally, we plan to develop a recommendation system that suggests clothing items based on users' interests and previous interactions. This approach helps users discover items they might like while ensuring they select the right size, making the shopping experience more efficient and enjoyable. By enhancing user satisfaction, this system can encourage more purchases and contribute to higher revenue.

CHAPTER 1: INTRODUCTION

1.1. Background

1.1.1. E- Commerce Recommendation System

E-commerce has become incredibly influential in recent years. As the product market continues to expand and diversify, users often need to invest significant time and effort to select the right items for them [1]. What sets e-commerce apart from traditional shopping is its ability to provide users with highly relevant product suggestions based on just a few clicks. This is made possible by recommendation systems, which assist users in making decisions, even when they lack the knowledge to assess a particular item effectively [1].

Recommendation systems create a win-win scenario for both businesses and customers. They not only boost sales for companies but also save buyers time by helping them quickly find products that match their preferences. In today's highly competitive and dynamic market, businesses can thrive only if they effectively understand and cater to customer preferences, ensuring strong customer retention. This is why recommendation systems are essential tools, designed to learn and adapt to customer tastes, and their value cannot be overstated.[2]

1.1.2. Clothing Recommendation System Based on User Data

A clothing recommendation system discovers user data like size, weight, height, age, rating, and reviews to recommend size and suitable items to the target user. The data gathered from users improves the decision-making process and helps identify the clothing that fits each person the best. The system can suggest products that fit the user's taste and body type by better understanding preferences through the analysis of user reviews and ratings. Reducing size mismatches, lowering clothing selection uncertainty, and enhancing the user experience are the objectives. By doing this, the system hopes to provide a more customized shopping experience, facilitating customers' search for clothing that fits properly and suits their particular style.

1.2. Problem statement

Online shopping has become increasingly popular, especially in the fashion industry, but consumers often struggle with choosing the right size clothing. One of the main issues is the difference in size selection between brands and product types, which causes customers to frequently face the problem of purchasing clothes that do not fit, leading to returns. This not only causes inconvenience for buyers but also increases costs for businesses and affects revenue. Therefore, the problem to address is how to develop a size recommendation system that accurately and efficiently helps consumers find clothing that fits their body type.

1.3. Scope and Objectives

In the scope of this study, we will focus on developing a clothing size recommendation system by combining user data (such as size, weight, height, body type, and reviews) with machine learning techniques. The primary goal is to enhance the clothing size recommendation process and provide better product suggestions based on personal preferences and rental history. The proposed approach will use clustering algorithms and text analysis techniques to improve the accuracy of size predictions and product suggestions.

This thesis has three main goals:

- Apply clustering algorithms (K-Means) to group users who have similar body types, rental history, and their reviews. It will help the system find out what clothing sizes usually work for each cluster of users.

- Use text analysis (sentiment analysis and TF-IDF) to understand customer feedback from reviews. This makes it easier to adjust the size suggestions to match real customer experiences.

- Apply KeyBERT to extract keywords from user reviews after clustering. These keywords are used to interpret and visualize user groups, helping to understand the distinct characteristics of each cluster.

1.4. Assumptions and Solution

K-Means, TF-IDF, and KeyBERT are used to operate different types of data. K-Means is good at structuring data such as height, weight, body type, or rental history of customers. Besides that, TF-IDF and KeyBERT will help handle unstructured data, especially from user reviews, where people often talk about size, fit, or problems with the clothes.

K-Means groups users who have similar body features and past behavior, so the system can suggest what size might fit for them. TF-IDF looks at the words users mention more often, KeyBERT will be used after clustering to interpret user segments and generate WordClouds, providing better insights into customer preferences. When these are combined, the system gets a better view of what each group of users needs.

By putting structured and unstructured data together, the size prediction using RF machine learning model becomes more accurate. This way, the system can reduce size mismatches and suggest clothes that are more likely to fit the user well.

1.5. Structure of Thesis

This is the flow of the thesis. This part briefly indicates how does each component of the report contribute to the acknowledgement of the readers:

Introduction: Provides an overview of the background, objectives, and scope of the clothing recommendation system based on user data.

Literature Review: Presents key techniques like K-Means clustering, TF-IDF, KeyBERT and their applications in recommendation systems, along with the role of machine learning models like Decision Tree and Random Forest.

Methodology: Describes the architecture of the recommendation system and how clustering algorithms and text analysis are combined to create personalized suggestions.

Implementation and Results: Discusses the implementation process and initial results, which form the foundation for future improvements due to the code is not yet completed.

Conclusion and Future Work: Summarizes key findings and suggests future development directions for the recommendation system.

CHAPTER 2: LITERATURE REVIEW

2.1. Recommendation System

2.1.1. Introduction to Recommendation System:

In the era of technology, Recommendation Systems (RS) filter personalized information to understand user preferences and suggest relevant content based on their interactions with the platform. RS can also assist decision-making by narrowing down a wide range of choices to those most likely to be chosen by the user.

RS models the compatibility between users and items using historical data like clicks, purchases, and likes. Its goal is to improve e-commerce performance by recommending products based on user preferences, thereby boosting sales and platform popularity. In short, RS is a key component in e-commerce, enhancing the user experience.

2.1.2. Categories of Recommendation System

Recommendation Systems can be classified into several types, each with a different approach to generating personalized suggestions for users. Overall, RS includes five types: content-based, collaborative filtering, hybrid, knowledge-based, and demographic-based [2]. However, the first two categories are more commonly used than the others.

Content- Based Filtering*: *this technique is built on the customer preferences and description about the items.

This kind of RS focuses on user preferences and product descriptions. It uses historical or current data from users, analyzing their interactions and preferences to identify patterns in the products they’ve previously engaged with. By examining product attributes, the system suggests items that are most similar to those the user has shown interest in. Since it centers on the product’s content, it does not necessarily assess the quality of the products. Additionally, this method does not require personal information, making it particularly useful in privacy-sensitive situations, such as healthcare or finance applications

Collaborative Filtering: ***makes suggestions by considering users' behavior and decisions made by other users.***

Collaborative Filtering pays attention to the behavior of other users and the decisions they have made. Collaborative filtering focuses on the relationships between users or between products. It also factors in product ratings, which is a significant improvement over content-based filtering. It analyzes how different users rate various items and uses these patterns to recommend items to users with similar preferences. Collaborative filtering can be further divided into two categories:

User-based: Recommendations are made based on the interests of users who are similar. Users are grouped based on their interactions and ratings, and products are recommended in a "You might also like..." format. This method is effective when item attributes are limited or insufficient, relying on groups of similar users to provide highly correlated recommendations.

Item-based: The second type of Collaborative Filtering RS

focuses on the historical behavior of users to find relationships between items. It identifies items similar to those the user has interacted with in the past. Unlike the user-based method, item-based collaborative filtering finds higher similarity between the ratings of different items since users tend to rate similar items similarly.

Demographic Filtering:

As the name suggests, this method uses demographic data, such as age, gender, and education level, to filter recommendations. It relies on these characteristics to suggest relevant content for specific demographic groups. It is particularly useful for applications where confidentiality isn't a concern, and users are willing to share their personal information to improve recommendations.

Knowledge-based:

Knowledge-based RS are used when item ratings are less important. They narrow the preferences down where there is limited purchase history. This approach is based on explicit knowledge about users' tastes and requirements. It often involves asking users directly about their preferences and using these answers to recommend products.

Hybrid:

Hybrid RS combines multiple recommendation techniques to create a more robust system. By combining methods, hybrid systems overcome the limitations of individual approaches and capitalize on their strengths. A common combination is content-based filtering and collaborative filtering. This allows the system to use product content to identify similar users and then apply collaborative filtering to recommend items that those users have interacted with.

2.2. Clustering documents with TF - IDF, K-Means and KeyBERT

2.2.1. TF - IDF

The TF-IDF (Term Frequency-Inverse Document Frequency) is a valuable tool because it provides a numerical representation of documents, capturing both their semantic content and the importance of specific terms. The Term Frequency (TF) component measures how often a word appears in a document, while the Inverse Document Frequency (IDF) component reduces the weight of terms that are common across all documents. Together, these components form the TF-IDF matrix, which is particularly useful for document clustering. It highlights terms that are both unique and frequent within a document, making them significant for describing the document's content.

Term Frequency (TF):

To construct the TF-IDF matrix, the pre-processed data is transformed into a numerical format that highlights the significance of terms within each document relative to the entire corpus.

For each document, Term Frequency (TF) is computed to measure how often each term appears in the text:

$$\mathrm{tf(t,d)} = \frac{f(t,d)}{max\{f(w,d) : w\in d\}} \qquad [3]$$

Inverse Document Frequency (IDF):

It is calculated for each term, which evaluates its rarity across the corpus:

$$\mathrm{idf(t,D)} = \log\frac{|D|}{|\{d\in D : t\in d\}|} \qquad [3]$$

TF - IDF:

In the last steps, the TF-IDF values for each term-document pair are calculated by multiplying the TF and IDF values. This process converts in a sparse matrix where rows correspond to documents, columns correspond to terms, and each entry represents the TF-IDF value of a term in a document.

$$\mathrm{tfidf(t,d,D) = tf(t,d)\ x\ idf(t,D)}\ [3]$$

Table 2.1. Notation and definition for TF-IDF decomposition formula

Notation	Definition
t	Term (word)
d	Document
D	Corpus (collection of all documents)

$f(t, d)$	Frequency of term t in document d
$\max\{f(w, d) : w \in d\}$	Maximum frequency of any term w in document d
$\|D\|$	Total number of documents in the corpus
$\|\{d \in D : t \in d\}\|$	Number of documents containing term t

2.2.2. K-Means Clustering

2.2.2.1 Overview

The *K-Means* clustering is suitable for data sets with large amounts of data and high feature dimensions, and its dependence on data is low. Therefore, *K-Means* has become a widely used clustering method. However, *K* of the traditional *K-Means* that needs to be determined in advance when it is initialized is determined only by the experience of the developer, and such subjectivity will affect the clustering efficiency and the credibility of the results. The random selection of the initial clustering center will cause the instability of the clustering results [4].

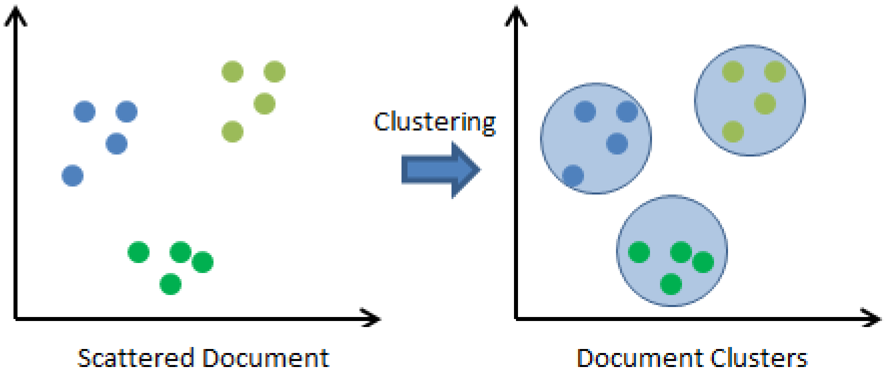


Figure 2.1 *From Scattered Data to Document Clusters using K-Means*

In text mining and recommendation systems, a document-term matrix is often constructed using the TF-IDF method. This matrix represents documents (e.g., user reviews or textual data) as rows, with columns corresponding to unique terms in the dataset. Each cell reflects the importance of a term in a specific document, balancing its frequency within the document and its rarity across all documents.

Once the TF-IDF matrix is generated, K-Means clustering is employed to group documents or users into clusters based on textual similarities. The algorithm iteratively assigns documents to clusters, recalculates the centroids, and continues until achieving an optimal grouping.

The output of K-Means includes cluster labels, which assign each document to a specific group, and cluster centroids, representing the central points of each cluster in the high-dimensional TF-IDF space.These results are valuable for grouping users with similar textual behaviors, such as review patterns, or for identifying clusters most relevant to specific topics or content.

2.2.2.2 Traditional K-Means Algorithm

The distance between data:

$$\mathrm{dis}(y_1,y_n) = \lim_{\lambda \to 0} \sqrt{\sum_i (y_i(p), y_{i+1}(p))^2} \quad , \lambda \ = \ max\{\left|y_{i+1}(p) \ - y_i(p)\right|\} \quad [4]$$

The center point:

$$k_j = \lim_{n \to \infty} \frac{\Sigma y_i(p)}{n_j} \qquad [4]$$

nj refers to the number of same class

Checking Convergence: The convergence flag can use the following formula to compute:

$$l = \Sigma\Sigma \lim_{\lambda \to 0} \sqrt{\sum_{i=1}^{n-1} (y_i(p), y_{i+1}(p))^2} \qquad [4]$$

Traditional K-Means achieves the purpose of clustering by carrying out the cyclic calculation on all the data. However, this process takes a lot of time.

Table 2.2. Notation and definition for traditional K-Means Clustering formula

Notion	Definition
D	The set of all data points, where each *yi* represents a data point in the dataset.
K	The number of clusters to be formed, predefined before the algorithm starts.
k_j	The centroid of cluster *j*, representing the mean of all points within the cluster.
c_j	A specific cluster *j*, formed by assigning points closest to the centroid k_j.
y_j	A specific data point in the dataset.
dis(y_j,k_j)	The distance metric , measuring the distance between *yi* and *kj*.
n_j	The number of data points assigned to cluster *j*.
l	The convergence criterion, measuring the changes in centroids between iterations.
λ	Maximum change in values during the recalculation process to ensure convergence accuracy.
μ	The mean of data points in a cluster, used as the cluster centroid.
σ^2	The variance of data points within a cluster, measuring the spread of the cluster.
p	Parameter to define the distance metric (*p*=2 for Euclidean, *p*=1 for Manhattan).

2.2.3. KeyBERT

KeyBERT is a Python library that helps extract keywords from a given text. It builds on top of the hugging face library, which provides a user-friendly environment for working with pre-trained BERT models to turn both the document and candidate phrases into high-dimensional vectors. Then, it compares those vectors using cosine similarity to figure out which phrases are most relevant to the document.[8]

What makes KeyBERT more useful is that it also supports traditional methods like TF-IDF and MMR (Maximal Marginal Relevance), which help pick out keywords that are not only relevant but also diverse, avoiding redundancy. Based on the similarity scores, the top-ranked phrases are selected as the final keywords.

KeyBERT can also be used for other tasks like document clustering or text classification, since they capture the semantic meaning of the content pretty well.

$$\text{Similarity(A, B)} = \frac{A \cdot B}{||A|| * ||B||} = \frac{\Sigma_{i=1}^{n} Ai*Bi}{\sqrt{\Sigma_{i=1}^{n} A_i^2} * \sqrt{\Sigma_{i=1}^{n} B_i^2}} \quad [6]$$

Table 2.3. Notation and definition for Cosine Similarity formula used in KeyBERT

Notion	Definition
A	Vector representation of the document
B	Vector representation of the candidate keyword/phrase
A_i	The i-th element of vector A
B_i	The i-th element of vector B
$\sum A_i \cdot B_i$	Dot product between vectors A and B
$\|\|A\|\|$	Magnitude (length) of vector A
$\|\|B\|\|$	Magnitude (length) of vector B
Similarity(A, B)	Cosine similarity score between vector A and B

2.3. Decision Tree & Random Forest models

2.3.1. Decision Tree Models

The DT model originated from statistics and has been extensively refined for applications in data mining, pattern recognition, and machine learning. It represents decisions and outcomes in a tree-like structure, where each node represents a test on a feature, each branch indicates the result of that test, and each leaf node provides a classification or prediction. The structure of DT is highly intuitive and easy to understand, which makes it popular for classification and regression tasks across various domains, such as predicting user behavior or categorizing data based on attributes.

The advantages of DT include its clear and interpretable structure, making it accessible to both technical and non-technical users. It is flexible enough to handle both numerical and categorical data and provides fast computation, especially for smaller datasets. However, DT also has notable disadvantages. It is prone to overfitting, particularly when the tree becomes too deep and starts to memorize noise in the data. Additionally, small changes in the dataset can lead to entirely different tree structures, making it unstable. Furthermore, it struggles with high-dimensional data, where feature interactions can become intricate. To address these issues, researchers introduced ensemble methods such as RF, which leverage multiple DTs to build more reliable models [7].

2.3.2. Random Forest models

RF is an ensemble learning method designed to overcome the weaknesses of DT while retaining its strengths. Instead of relying on a single tree, it builds multiple DTs and combines their predictions to improve accuracy and robustness. The key principles behind RF are that each tree is trained on a randomly selected subset of the data, using a method called bootstrap sampling. Additionally, at each split within a tree, a random subset of features is considered, reducing overfitting and increasing model generalization.

The benefits of RF include its ability to compensate for the weaknesses of individual DTs by aggregating their predictions, leading to improved stability and accuracy. The method is highly scalable and can handle large datasets efficiently. Moreover, RF works well with imbalanced data and noisy datasets, making it suitable for diverse applications. However, RF also has its challenges. It sacrifices interpretability, as the model is essentially a collection of multiple trees.

Training a large number of trees can also be computationally expensive, especially with large datasets.

The construction of an RF model involves several steps. First, the dataset is divided into random subsets through bootstrap sampling, ensuring diversity among the training sets for each tree. Second, individual DTs are trained on these subsets, considering only a random subset of features at each split. Finally, the predictions of all trees are combined to produce the final output, using majority voting for classification tasks or averaging for regression tasks [8].

CHAPTER 3: METHODOLOGY

3.1. General Idea

The main idea of this study is to develop a Recommendation System based on TF-IDF and clustering techniques. TF-IDF is used to extract key features from customer reviews and product descriptions, quantifying the importance of words in the dataset. Clustering algorithms then group similar products or users based on these features, forming meaningful clusters that enhance the recommendation process.

Random Forest model is utilized to predict suitable clothing sizes for users based on attributes such as height, weight, and age. By combining text analysis, clustering, and regression, the proposed system not only recommends relevant products but also personalizes size suggestions for each individual user.

The proposed methodology ensures that the system focuses on relevant patterns, reducing noise and improving prediction accuracy. This integration of TF-IDF, clustering, combined with RF serves as a scalable and effective approach, laying the groundwork for future enhancements in recommendation systems.

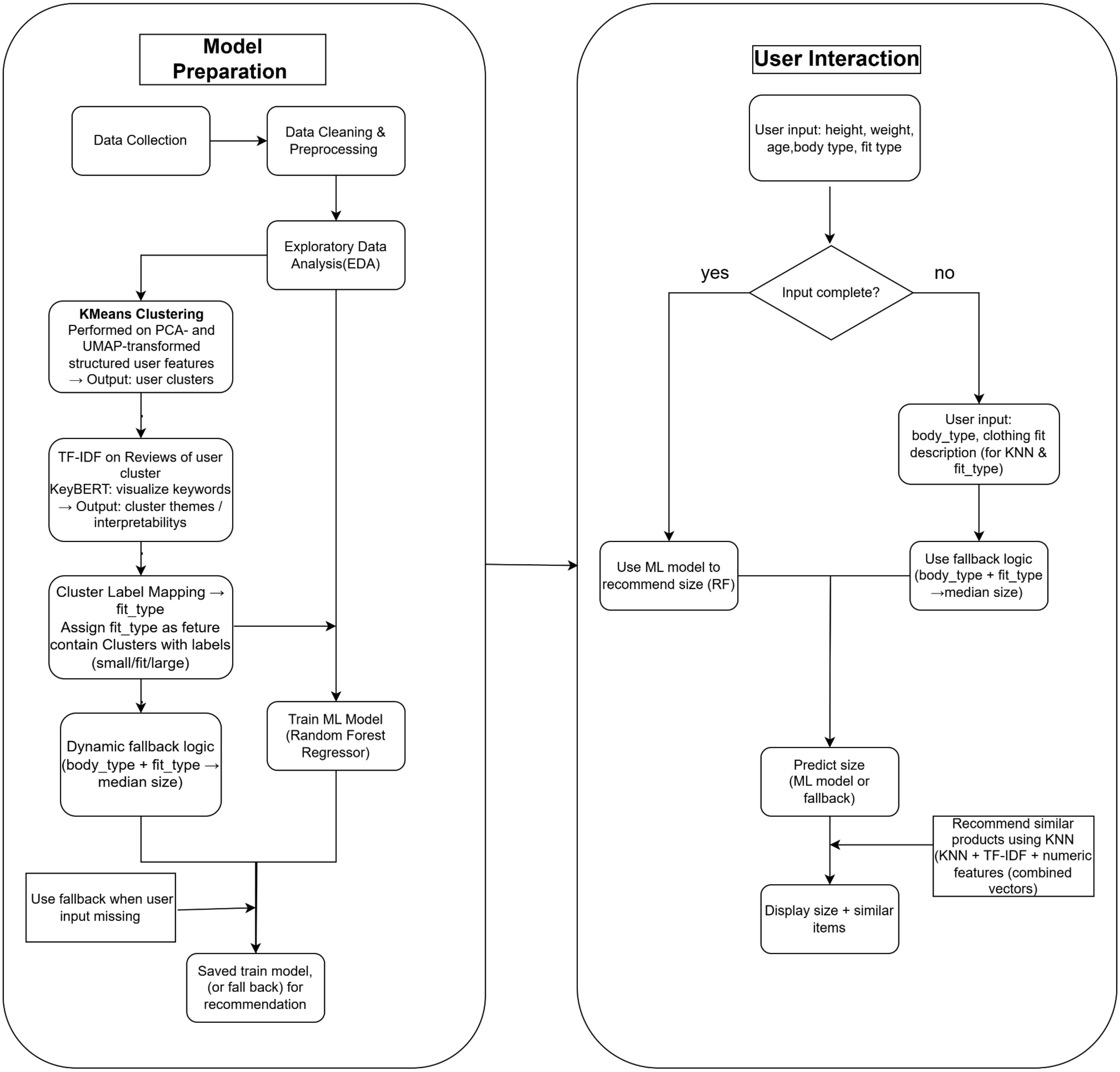

Model Preparation
Data Collection
Data Cleaning & Preprocessing
Exploratory Data Analysis(EDA)
KMeans Clustering
Performed on PCA- and UMAP-transformed structured user features
→ Output: user clusters
TF-IDF on Reviews of user cluster
KeyBERT: visualize keywords
→ Output: cluster themes / interpretabilitys
Cluster Label Mapping → fit_type
Assign fit_type as feture contain Clusters with labels (small/fit/large)
Dynamic fallback logic (body_type + fit_type → median size)
Train ML Model (Random Forest Regressor)
Use fallback when user input missing
Saved train model, (or fall back) for recommendation
User Interaction
User input: height, weight, age,body type, fit type
Input complete?
yes
no
User input: body_type, clothing fit description (for KNN & fit_type)
Use ML model to recommend size (RF)
Use fallback logic (body_type + fit_type →median size)
Predict size (ML model or fallback)
Recommend similar products using KNN (KNN + TF-IDF + numeric features (combined vectors)
Display size + similar items

Figure 3.1 *Overall Framework of the Recommendation System*

3.2. Data Collecting, Preprocessing and EDA

3.2.1 Data Collecting, Preprocessing

The dataset that is used in the scope of this thesis proposal is from cseweb platform. The original set can be taken from https://cseweb.ucsd.edu/~jmcauley/datasets.html#clothing_fit (RentTheRunway dataset). The dataset is stored in the form of a .json file and can be loaded to Google Collaboratory using Python library pandas.

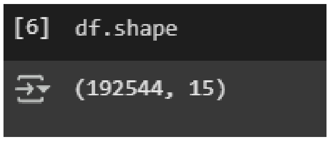
This is a brief description of the data:

Table 3.1 Description of the “TheRentway” dataset

Column name	**Data Types**	**Description**
fit	object	Describes the fitting (e.g., 'fit', 'small', 'large').
user_id	Int64	Unique id of customer
bust_size	object	bust size of the customer
item_id	Int64	Unique id of item.
weight	object	Weight of the customer.
rating	float64	Customer's rating for the rented item (e.g., 1 to 5).
rented_for	object	Purpose for which the item was rented (e.g., 'party', 'work').
review_text	object	Detailed text review provided by the customer.

'body_type'	object	Describes the body type of the customer (e.g., 'athletic', 'curvy').
review_summary	object	Summary of the customer's review.
height	object	Height of the customer.
size	Int64	Size of the item rented.
age	float64	Age of the customer.
review_date	object	Date when the review was submitted.

Shape of Raw Data:

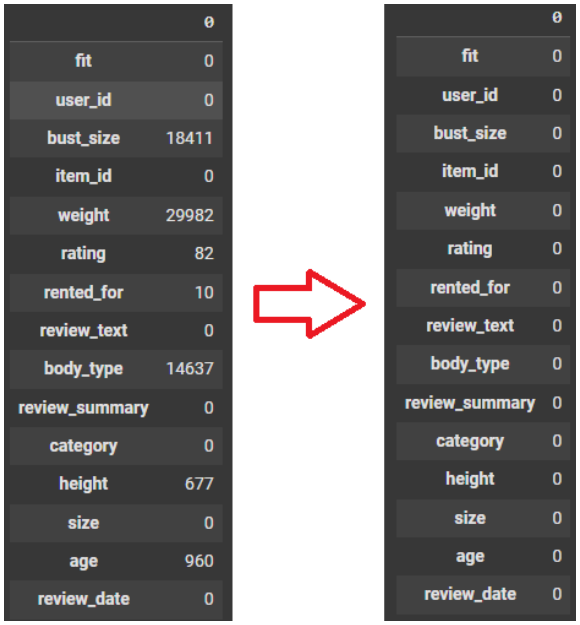


```
[6]  df.shape

(192544, 15)
```

Here are the preprocessing tasks that I performed to format the data:

- Handle missing value (NAN)

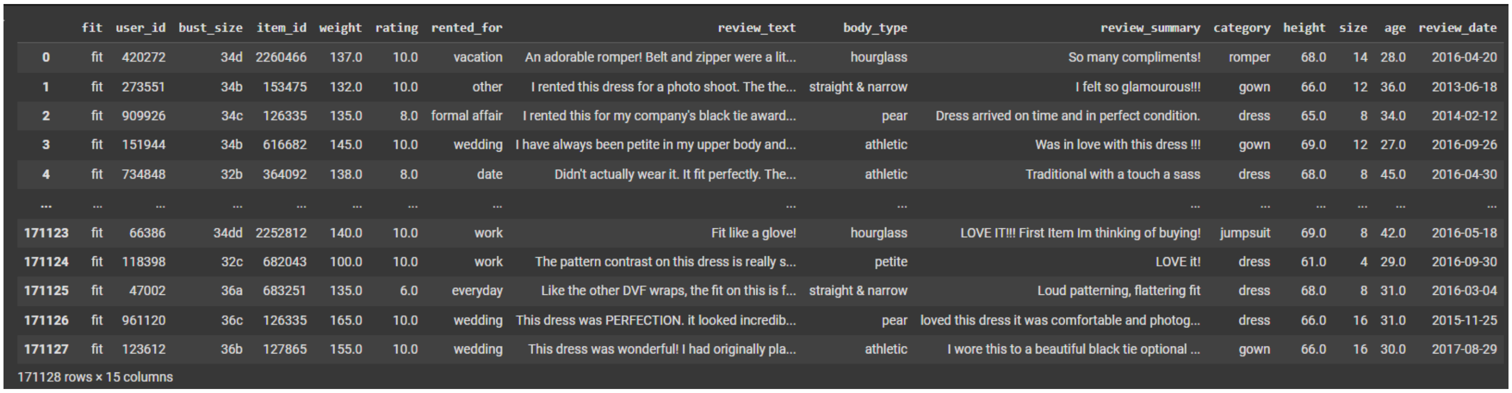


	0
fit	0
user_id	0
bust_size	18411
item_id	0
weight	29982
rating	82
rented_for	10
review_text	0
body_type	14637
review_summary	0
category	0
height	677
size	0
age	960
review_date	0

	0
fit	0
user_id	0
bust_size	0
item_id	0
weight	0
rating	0
rented_for	0
review_text	0
body_type	0
review_summary	0
category	0
height	0
size	0
age	0
review_date	0

Figure 3.2 *Missing value before and after handling.*

- Transform categorical features ("fit", "rented_for", "category", "bust_size", etc.) into numerical representations..
- Tokenization and text cleaning for columns like “review_text” and “review_summary”. Also removing stopwords, digits and special characters.
- Converting textual data from "review_text" or "review_summary" into numerical format.



	fit	user_id	bust_size	item_id	weight	rating	rented_for	review_text	body_type	review_summary	category	height	size	age	review_date
0	fit	420272	34d	2260466	137.0	10.0	vacation	An adorable romper! Belt and zipper were a lit...	hourglass	So many compliments!	romper	68.0	14	28.0	2016-04-20
1	fit	273551	34b	153475	132.0	10.0	other	I rented this dress for a photo shoot. The the...	straight & narrow	I felt so glamourous!!!	gown	66.0	12	36.0	2013-06-18
2	fit	909926	34c	126335	135.0	8.0	formal affair	I rented this for my company's black tie award...	pear	Dress arrived on time and in perfect condition.	dress	65.0	8	34.0	2014-02-12
3	fit	151944	34b	616682	145.0	10.0	wedding	I have always been petite in my upper body and...	athletic	Was in love with this dress !!!	gown	69.0	12	27.0	2016-09-26
4	fit	734848	32b	364092	138.0	8.0	date	Didn't actually wear it. It fit perfectly. The...	athletic	Traditional with a touch a sass	dress	68.0	8	45.0	2016-04-30
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
171123	fit	66386	34dd	2252812	140.0	10.0	work	Fit like a glove!	hourglass	LOVE IT!!! First Item Im thinking of buying!	jumpsuit	69.0	8	42.0	2016-05-18
171124	fit	118398	32c	682043	100.0	10.0	work	The pattern contrast on this dress is really s...	petite	LOVE it!	dress	61.0	4	29.0	2016-09-30
171125	fit	47002	36a	683251	135.0	6.0	everyday	Like the other DVF wraps, the fit on this is f...	straight & narrow	Loud patterning, flattering fit	dress	68.0	8	31.0	2016-03-04
171126	fit	961120	36c	126335	165.0	10.0	wedding	This dress was PERFECTION. it looked incredib...	pear	loved this dress it was comfortable and photog...	dress	66.0	16	31.0	2015-11-25
171127	fit	123612	36b	127865	155.0	10.0	wedding	This dress was wonderful! I had originally pla...	athletic	I wore this to a beautiful black tie optional ...	gown	66.0	16	30.0	2017-08-29

171128 rows × 15 columns

Figure 3.3 *Data after preprocessing*

3.2.2. Exploratory Data Analysis (EDA)

Bar chart:

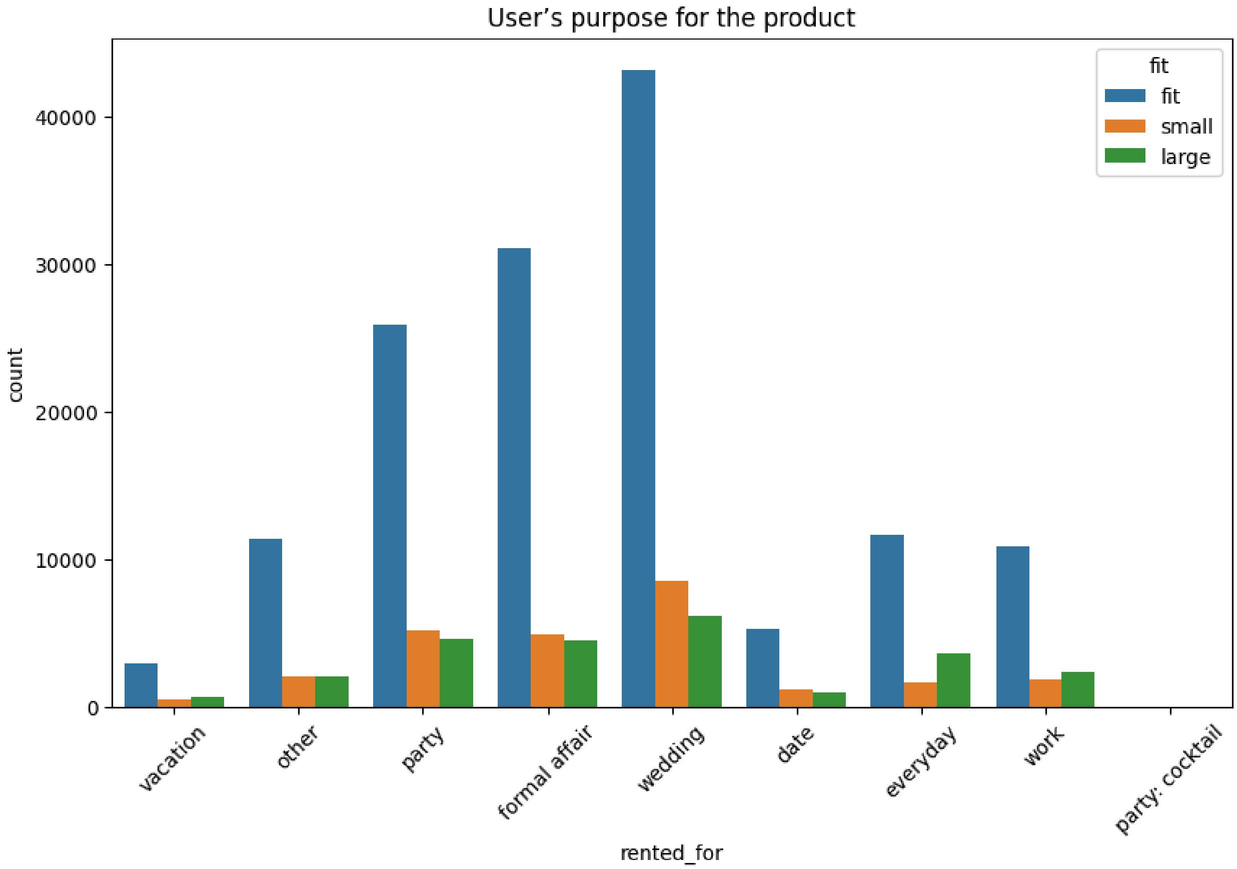


Figure 3.4 *Size Fit Distribution by Renting Purpose.*

From the graph above, we can see that most people rent clothes for weddings and formal affairs. Across all renting purposes, the majority of users reported that the size fit their body.

However, there is still a noticeable proportion of users who rated the items as “small” or “large,” especially for formal occasions. This highlights the importance of choosing the right size, and a size recommendation system can help reduce such mismatches in these scenarios.

Heat map:

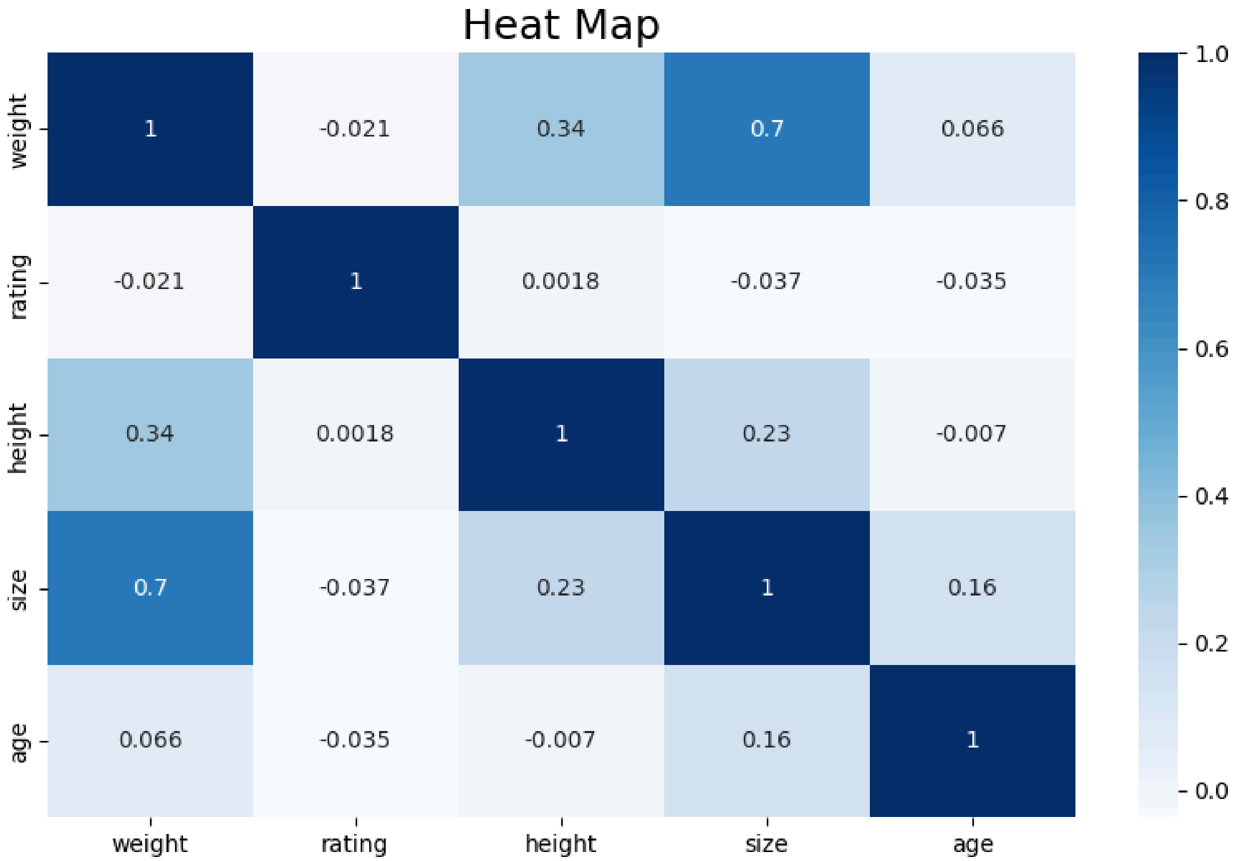


Figure 3.5 *Correlation Heatmap between Size and User Attributes.*

When exploring the data, We also created a heatmap to help study the correlation between each feature. It is observed that the clothing size is positively correlated with the user's weight and height. Especially weight, it has a significant positive correlation with the size (0.7), making it a key factor in size prediction.

Distribution:

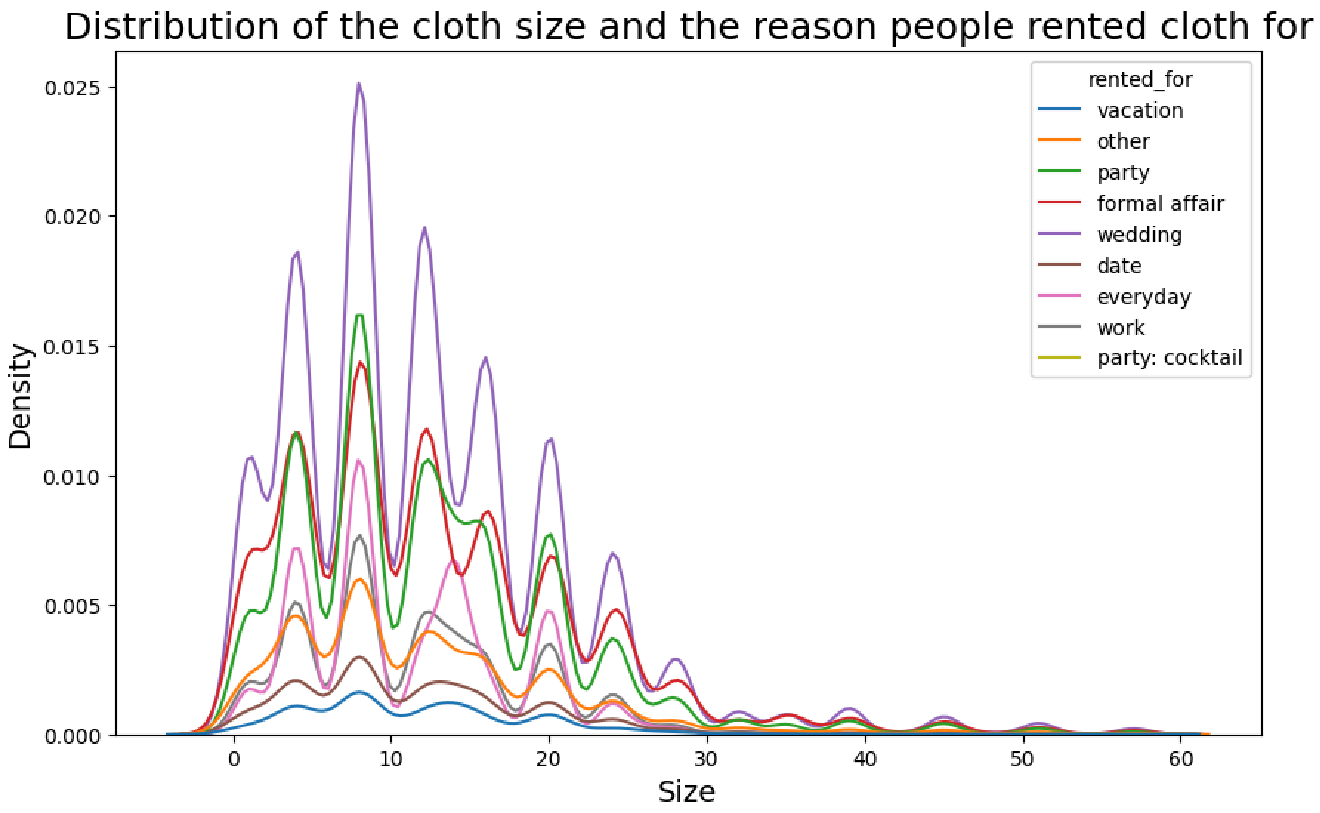


Figure 3.6 *Distribution of Clothing Sizes by Renting Purpose*

The graph on the left illustrates the distribution of the cloth size and the situation people rented cloth for. It can be observed that most sizes are in the range from 3 to 18 regardless of the occasion.

Scatter plot:

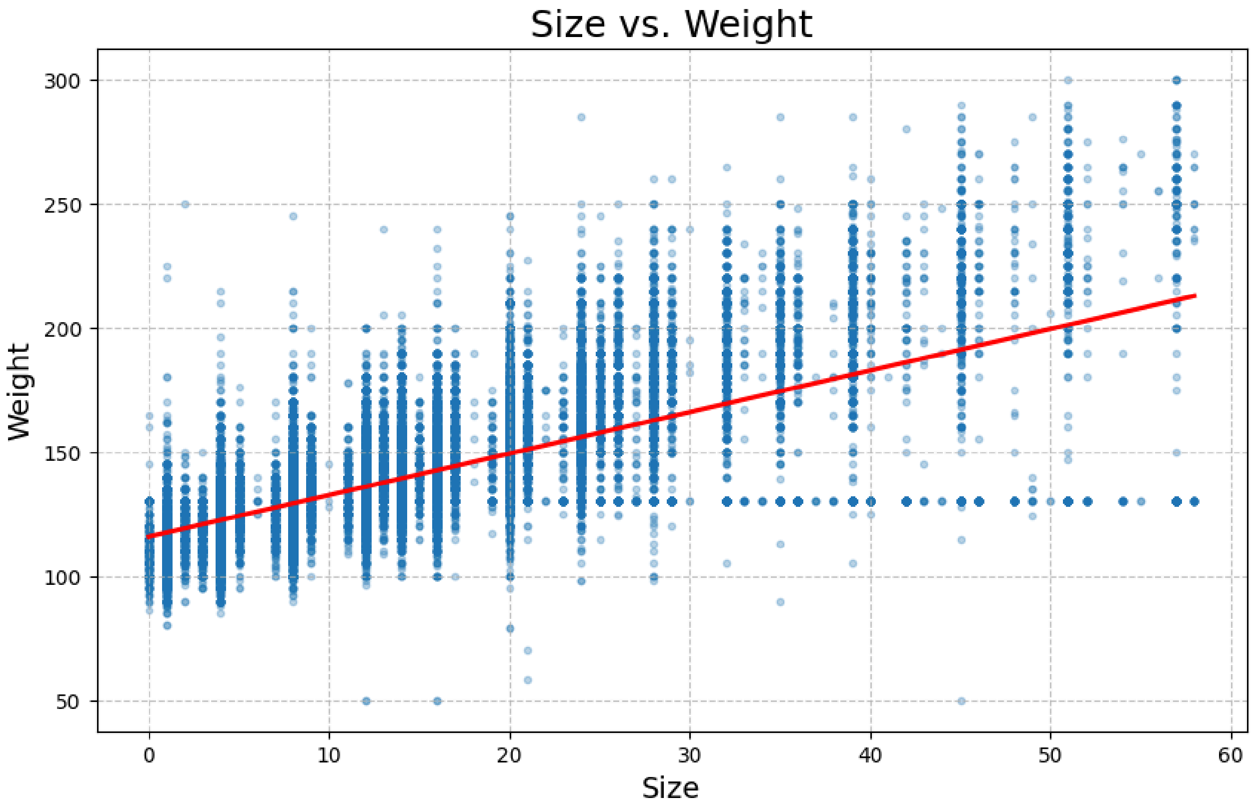


Figure 3.7 *Relationship between Weight and Clothing Size*

The original data had some outliers in size and weight. Using z-score (|z| < 3) to remove the extreme ones (those far from the average), the dataset can be clearly visualized, also as the regression line showed the trend more obviously.

Scatter plot after removing outliers:

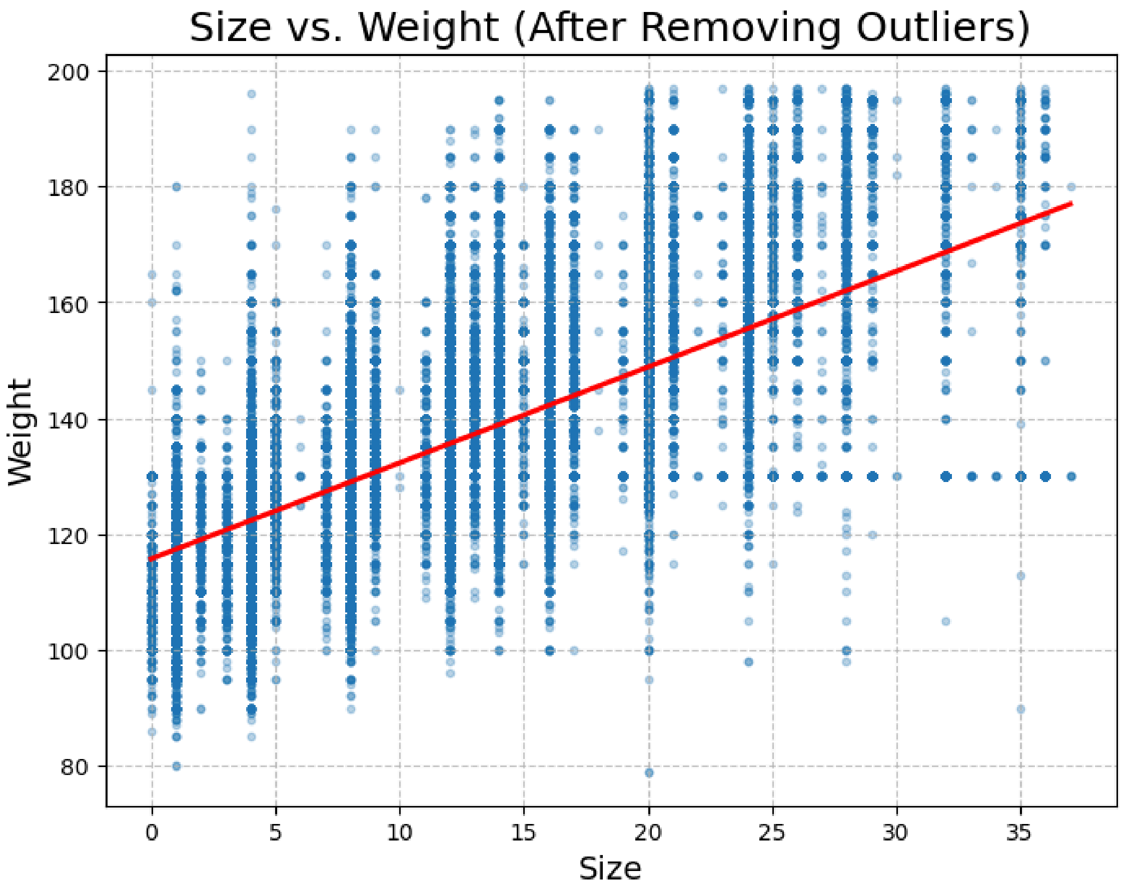
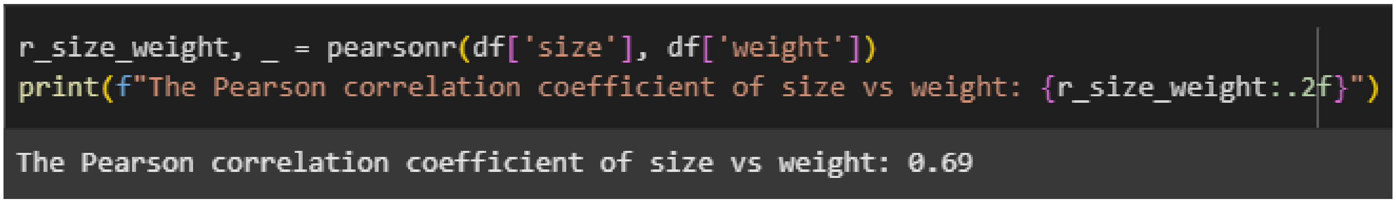


Figure 3.8 *Relationship between Weight and Clothing Size (using Z-score)*

The Pearson correlation coefficient of size vs weight:



```
r_size_weight, _ = pearsonr(df['size'], df['weight'])
print(f"The Pearson correlation coefficient of size vs weight: {r_size_weight:.2f}")
```

```
The Pearson correlation coefficient of size vs weight: 0.69
```

Illustrated by the graphs above, it can be seen that a user's weight is positively correlated with clothing size — the heavier the person, the larger the size they tend to wear. This explains why individuals with higher body weight often choose larger-sized clothing.

Violin plot:

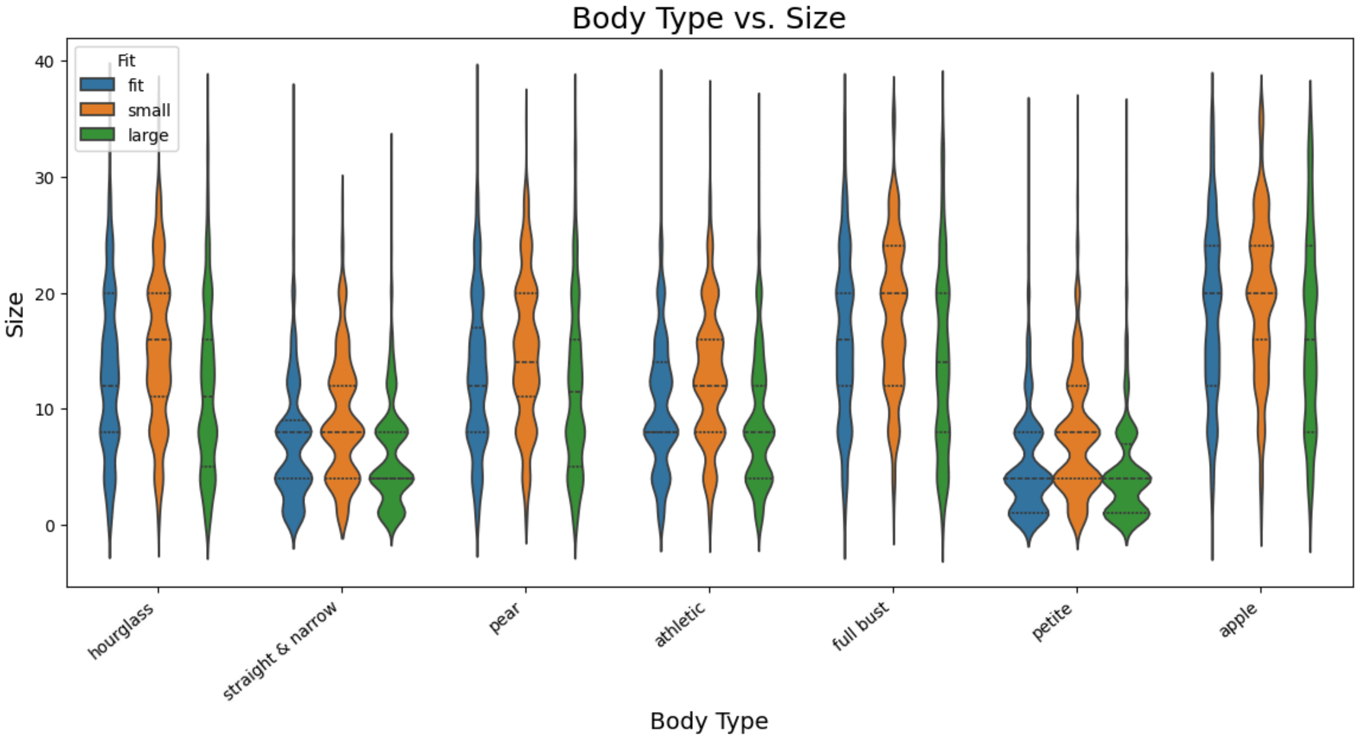


Figure 3.9 *Distribution of Clothing Size across Body Types*

About whether people's body types affect the cloth size they chose.

Based on the plot, it can be concluded that most people in this dataset have a body type of full bust and apple.

Except for people with straight & narrow and petite body types who will choose a smaller size, there is not much difference in the clothing size between other body types.

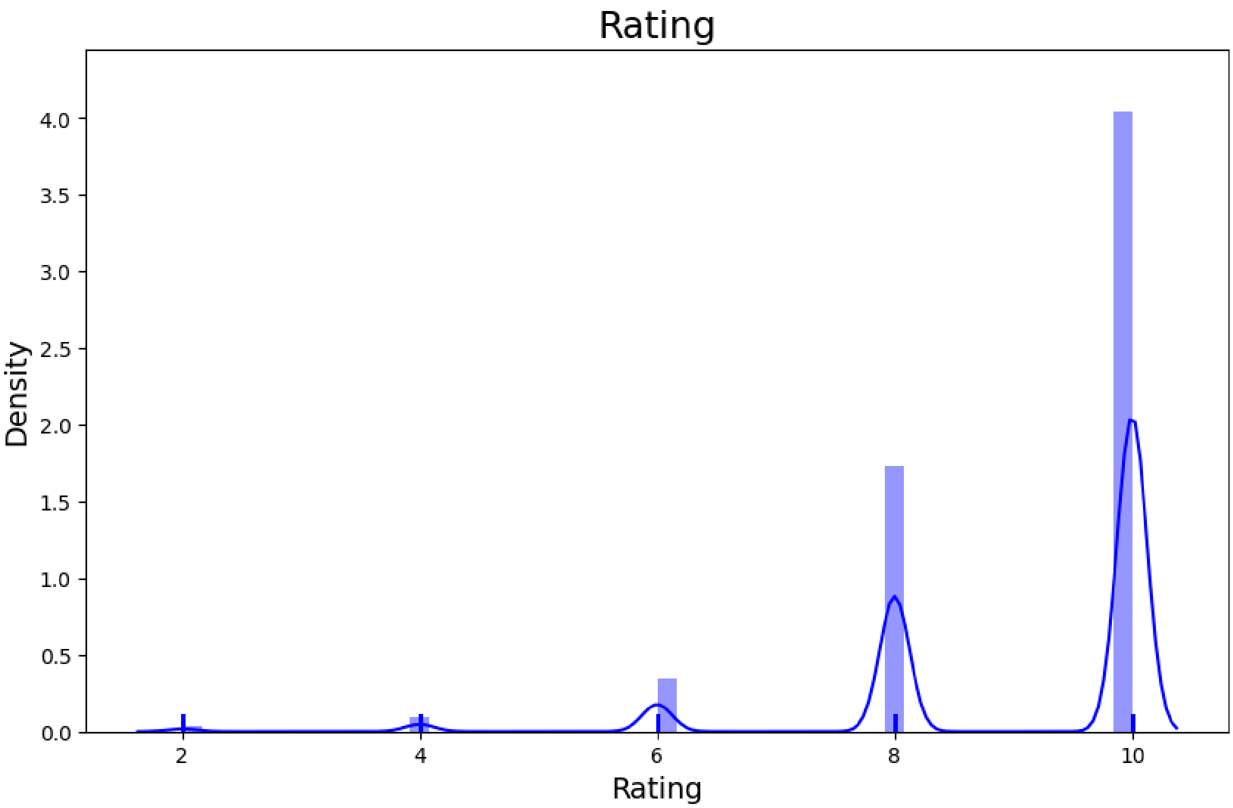


Figure 3.10 *Distribution of Customer's Ratings for Purchased Items*

The graph above illustrates the distribution of people's ratings on their purchases. It is clear that there are many more high ratings than low ratings. The highest rating is 10, also appears the most in the rating column. It can be shown that customers were generally satisfied with their purchases and had positive feedback.

3.3. Text Analysis

3.3.1 Affin score

To explore what motivates users to leave extremely positive or negative reviews, we first calculated sentiment scores using the Afinn library. This tool assigns sentiment values to each 'review_summary', helping identify whether a review expresses praise or dissatisfaction.

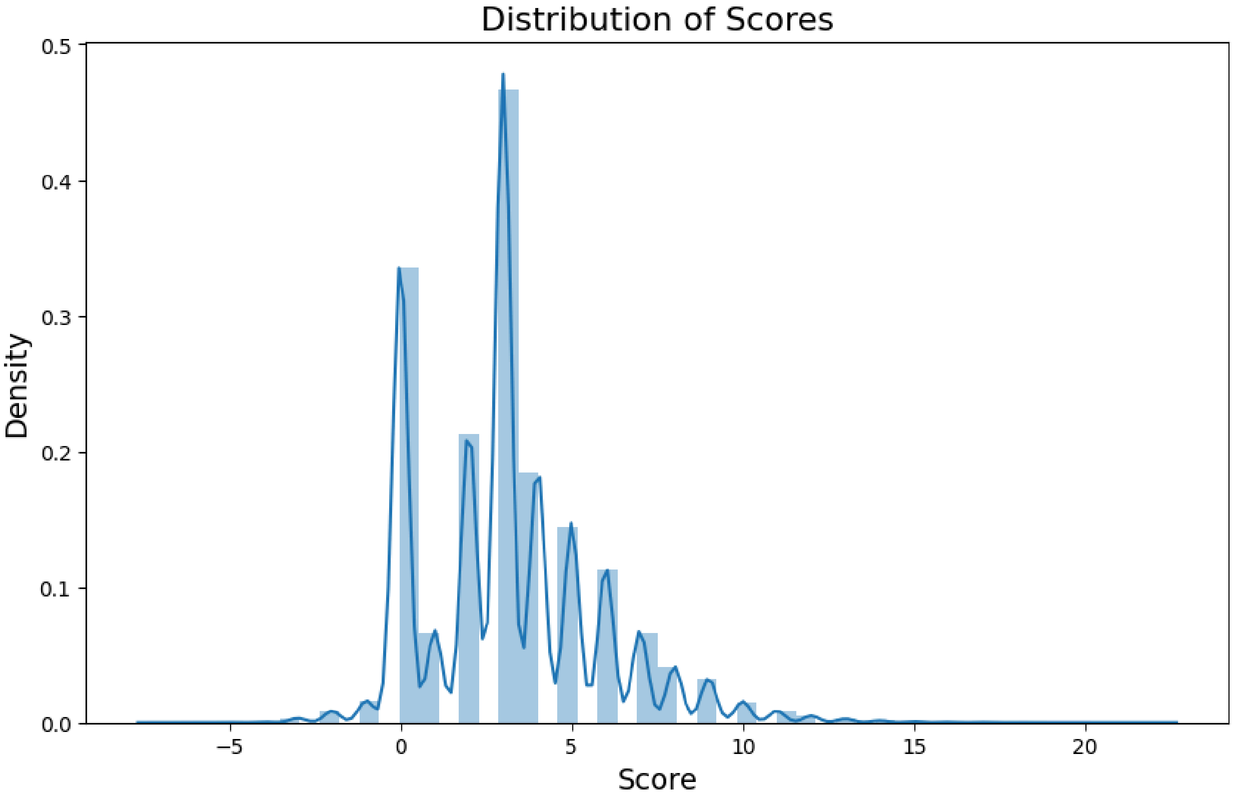


Figure 3.11 *Distribution of Sentiment Scores in Customer reviews.*

After obtaining sentiment scores (‘sent_score’), there are separated into two groups:

- Highly positive reviews (score > 5)
- Strongly negative reviews (score < 0)

We found out that most people tend to give a moderate review (afinn score between 0 to 5) to the clothes they rent. But in some cases, customers would like to give harsh comments on their renting experience (negative score). There are also some situations where people praise the clothes. Therefore our goal is to find out why some people would criticize or praise after receiving their clothes and improve the quality of service.

3.3.2 Frequent words

To discover the most common expressions used in customer reviews, a bigram frequency analysis was performed using CountVectorizer. This technique identifies word pairs (bigrams) that appear most frequently in the review text.

By applying it on the review text, the system extracts the most frequent bigrams such as “true size”, “fit true”, and “many compliments”, which reflect key reasons behind customer satisfaction.

Only reviews with positive sentiment (high rating score) were included to ensure relevance.

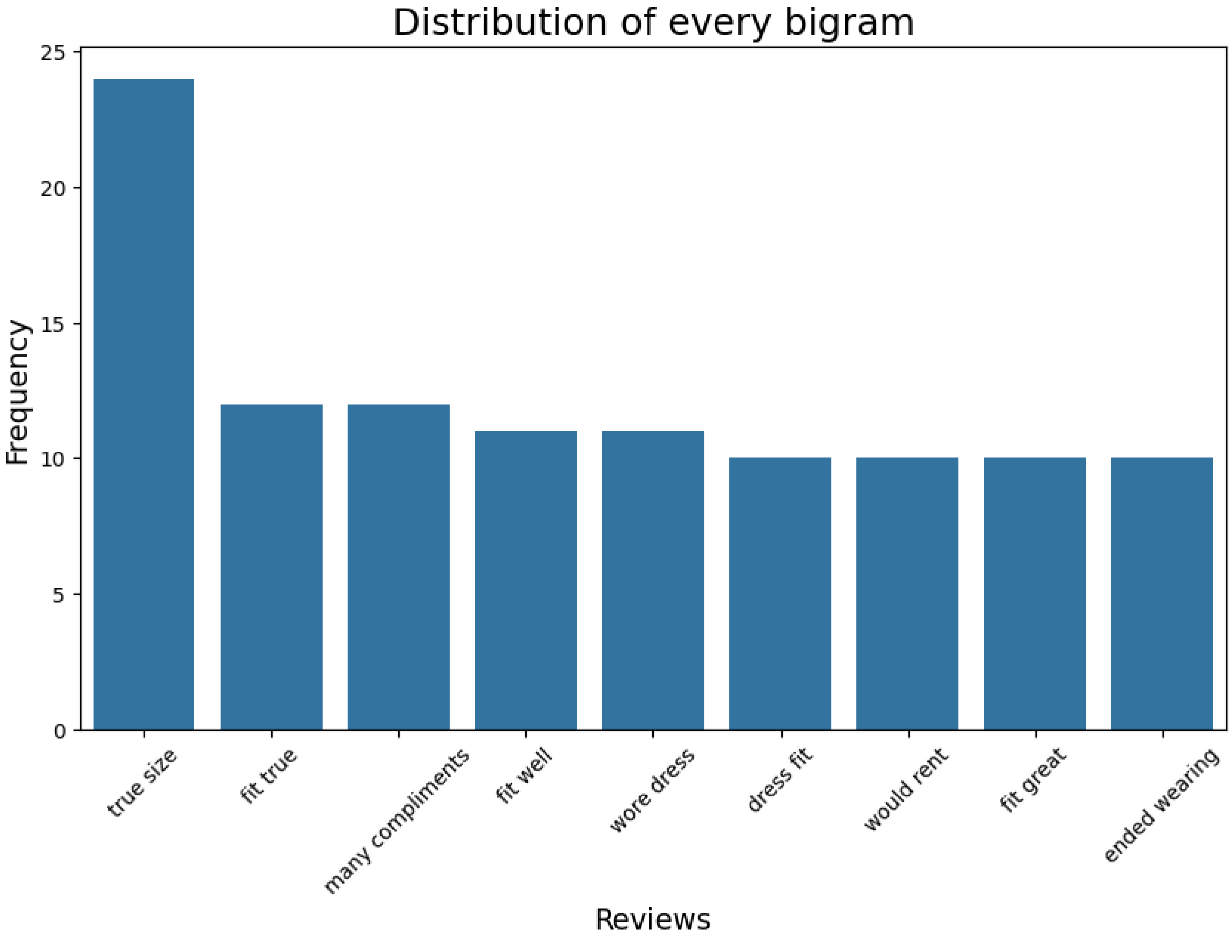


Figure 3.12 *Good Comments of every Bigram*

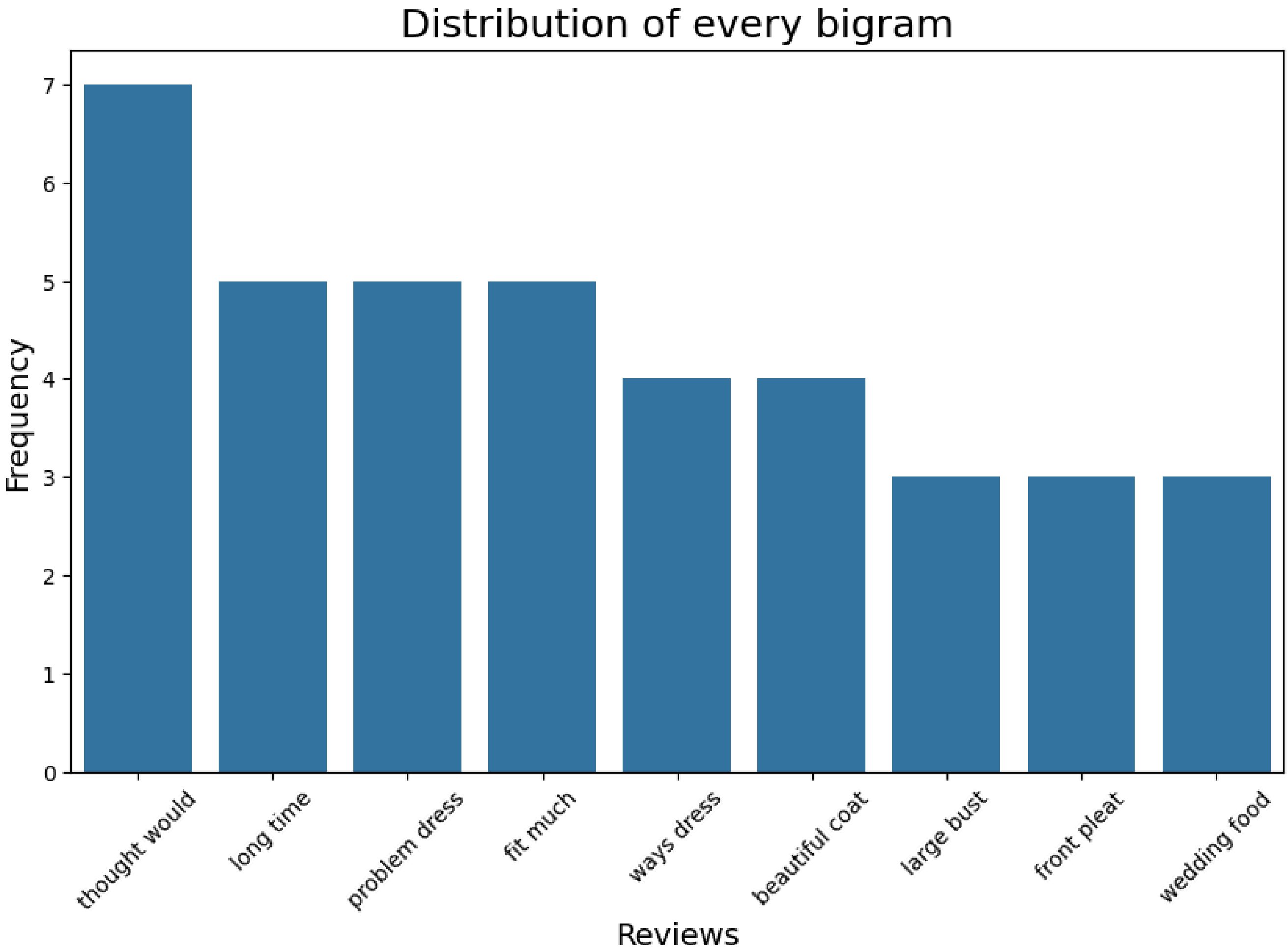


Figure 3.13 *Bad Comments of every Bigram*

From the graphs above, it can be seen that the most important thing for a customer is whether the clothes fit them well. So it's necessary to conduct a prediction on the size of clothes before new customers place orders.

Wordcloud (for good and bad comments) are also made to show what customers are saying in their renting experiences.

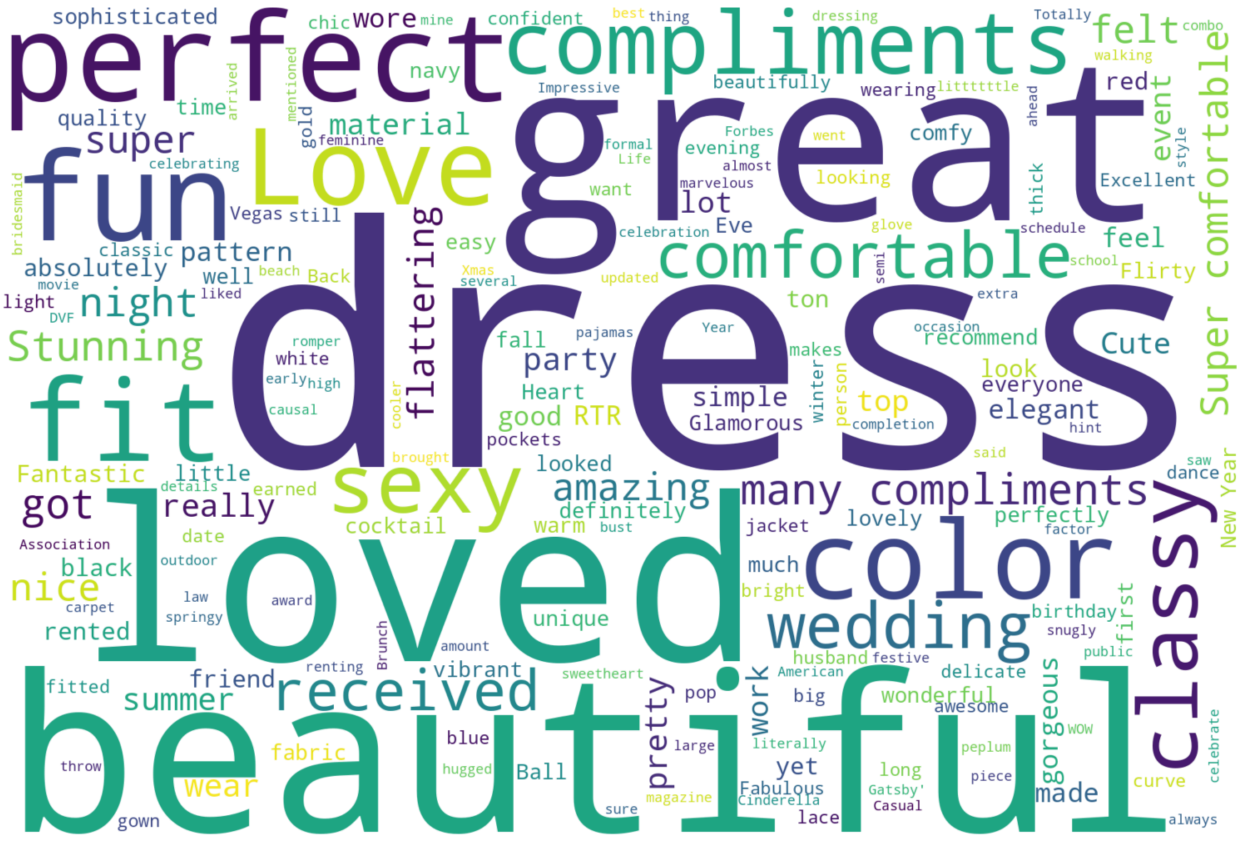


Figure 3.14 *Word cloud for Good Comments*

3.4. Clustering Analysis

Due to a large dataset (over 192,544 observations), a sample of 1,000 rows was taken. This number was determined by testing how long it would take to run code with various dataset sizes. 1,000 seemed to yield the best cost versus benefit.

3.4.1 PCA and Umap in Dimensionality Reduction

PCA reduces the complexity of multi-dimensional data by capturing its main patterns and trends. It achieves this by converting the original data into a smaller number of dimensions that effectively summarize the key features.

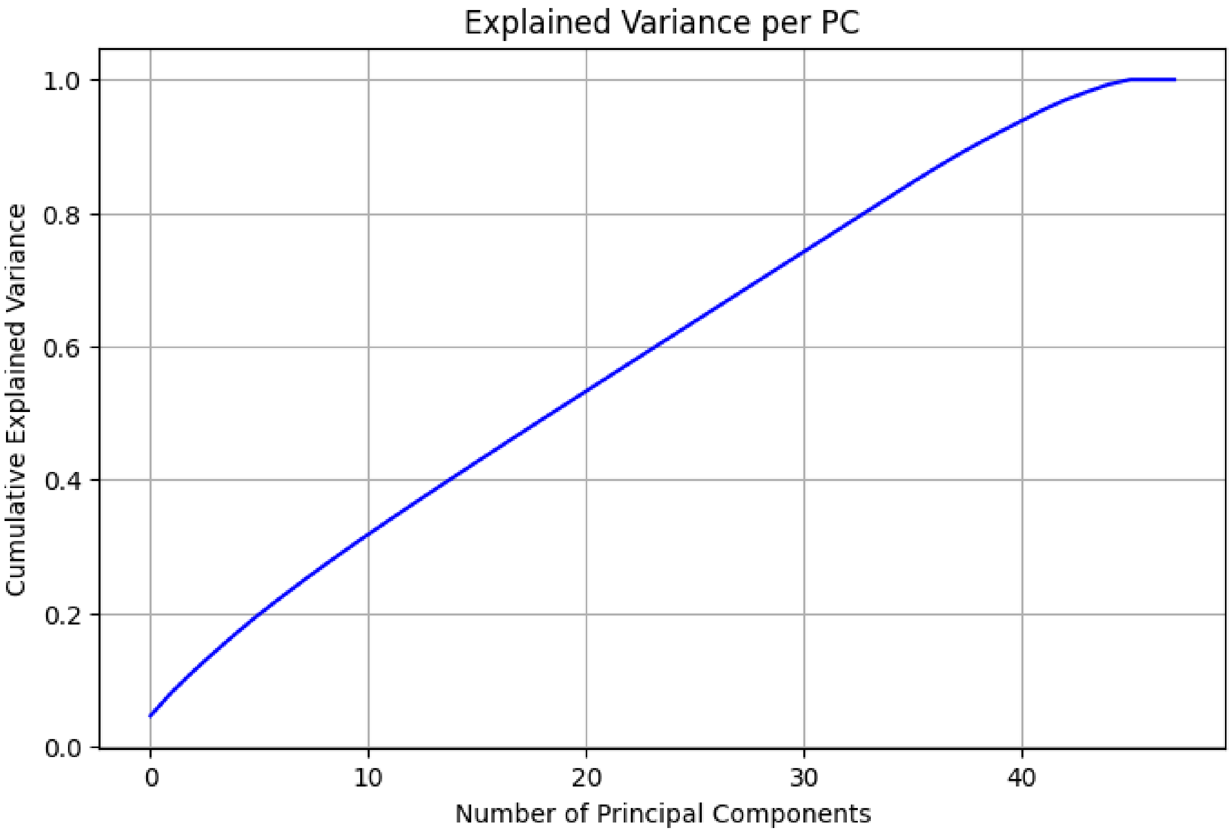
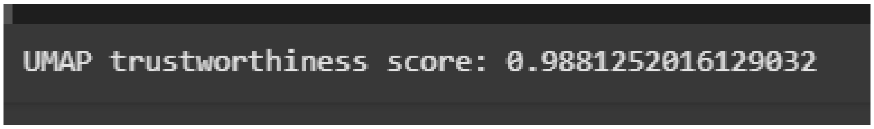


Figure 3.15 *Explained Variance by Principal Components*

After using PCA to conduct Dimension Reduction. We can see from below that the graph illustrates the explained variance per PCA component. We can see from the graph that with approximately the first 48 components we can explain the majority of the variance.

Providing nonlinear structure guarantees and high performance on big data, UMAP is increasingly widely used in many fields with complex data velocity. [10]

To further assess the quality of UMAP's dimensionality reduction, the trustworthiness score was computed, as shown below:



```
UMAP trustworthiness score: 0.9881252016129032
```

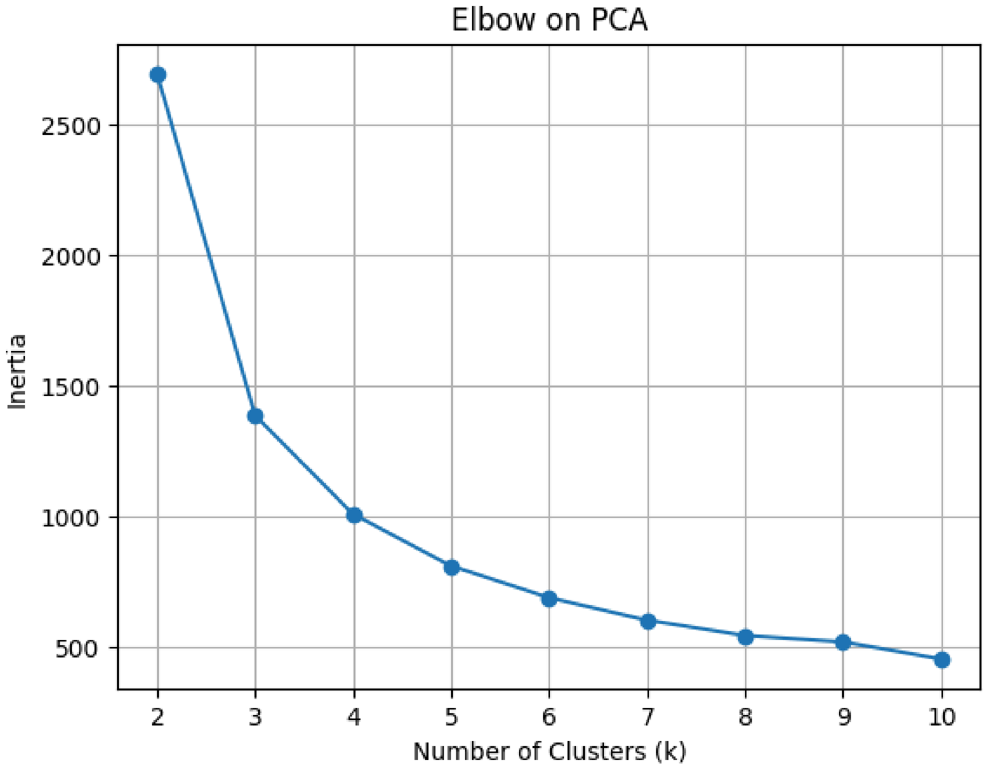
With a score of **0.988**, it indicates that UMAP has preserved the local structure of the original high-dimensional data extremely well.

A trustworthiness score close to **1.0** suggests that the distances between neighbors in the reduced space are consistent with the original space. This reinforces the reliability of using UMAP output for downstream tasks such as clustering or visualization.

3.4.2 Clustering and Evaluation

3.4.2.1 Optimal Number of Clusters (Elbow Method)

The Elbow Method is a popular technique used to estimate the optimal number of clusters for the K-Means algorithm and some other unsupervised learning algorithms. Since this method relies on graph observation, human judgment is required to determine the "elbow point" and thus the appropriate number of clusters in the data.[11]



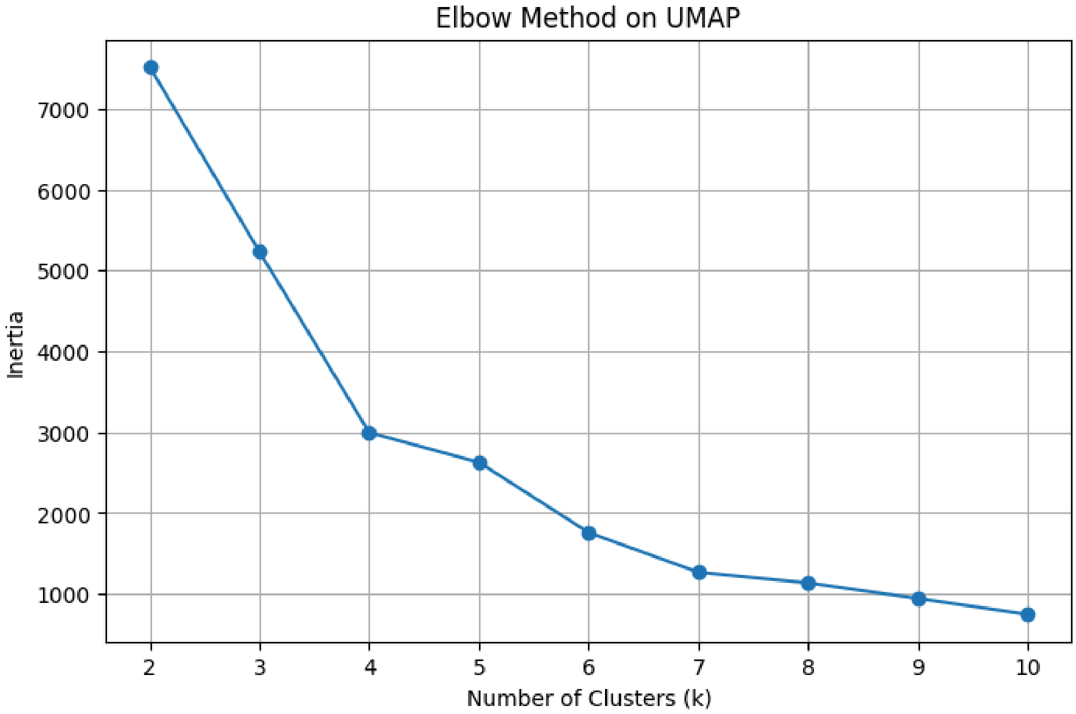


Figure 3.16 *Elbow Method to Determine Optimal Number of Clusters*

The optimal number of clusters was determined to be 3, as inertia significantly decreased up to 3 clusters and flattened out afterward.

This means that dividing the data into three clusters strikes a good balance between complexity and effectiveness.

Adding more clusters beyond this point does not significantly improve the results, so “three” is a reasonable and practical choice.

3.4.2.2 Clustering Quality Evaluation (Silhouette Score)

The Average Silhouette Width is a popular measure used to evaluate the goodness of clusters and estimate the optimal number of clusters. This study examines whether ASW can be used as a general objective function for optimization in the clustering process [12]

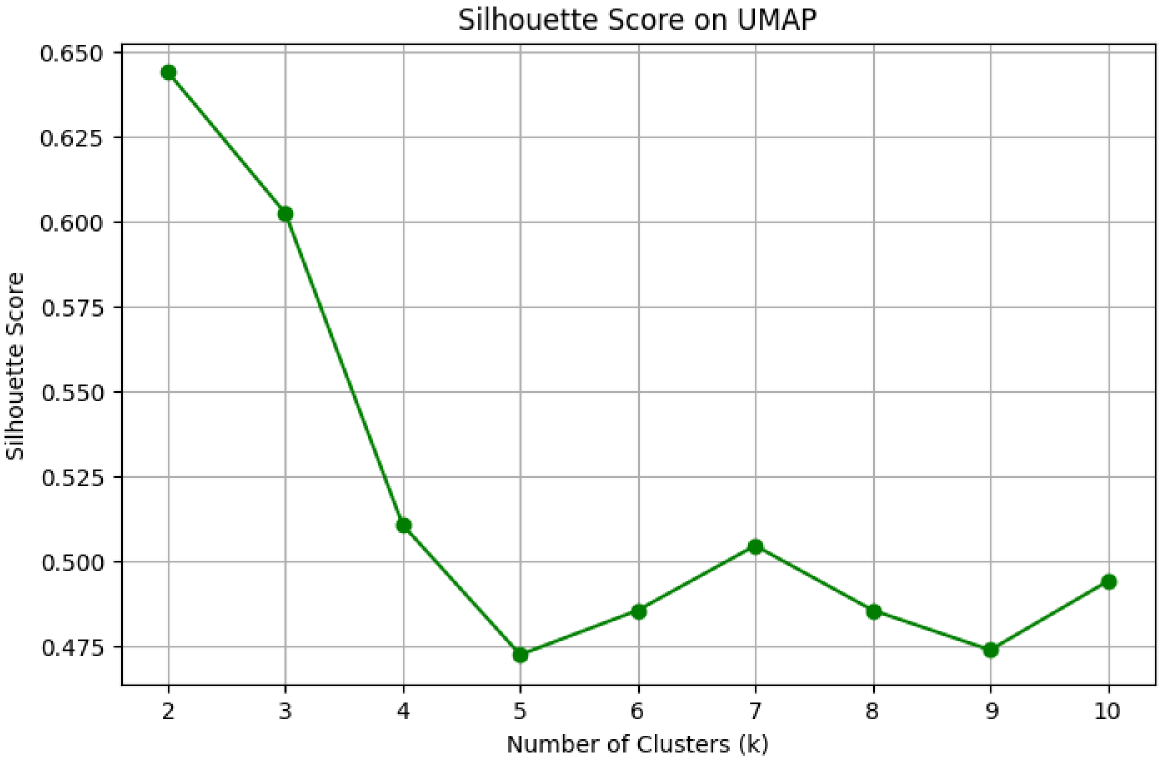


Figure 3.17 *Silhouette Scores for Different Cluster Counts*

From the chart above, it can be observed that:

- ➔ When **k=2**, the Silhouette Score was the highest (**~0.645**), indicating that splitting the dataset into 2 clusters results in the most clearly separated groups.
- ➔ The score for **k =3** remains relatively high (**~0.605**), which makes it an acceptable choice when a slightly more detailed segmentation is desired.
- ➔ But from **k=4** onwards, the Silhouette Score continues to decrease and does not show any significant improvement. This suggests that adding more clusters causes overlapping between groups, making them harder to distinguish and reducing the overall clustering quality.

=> When comparing both Elbow and Silhouette methods, it also shows that **k=3** is a good point, the curve bends at 3 and flattens after that.

So basically, both methods agree that two or three clusters are the best choices. "**k =2**" for the cleanest split, "**k=3**" is acceptable if the clusters need a bit more detail.

3.4.3 PCA vs UMAP in K-Means Clustering Visualization.

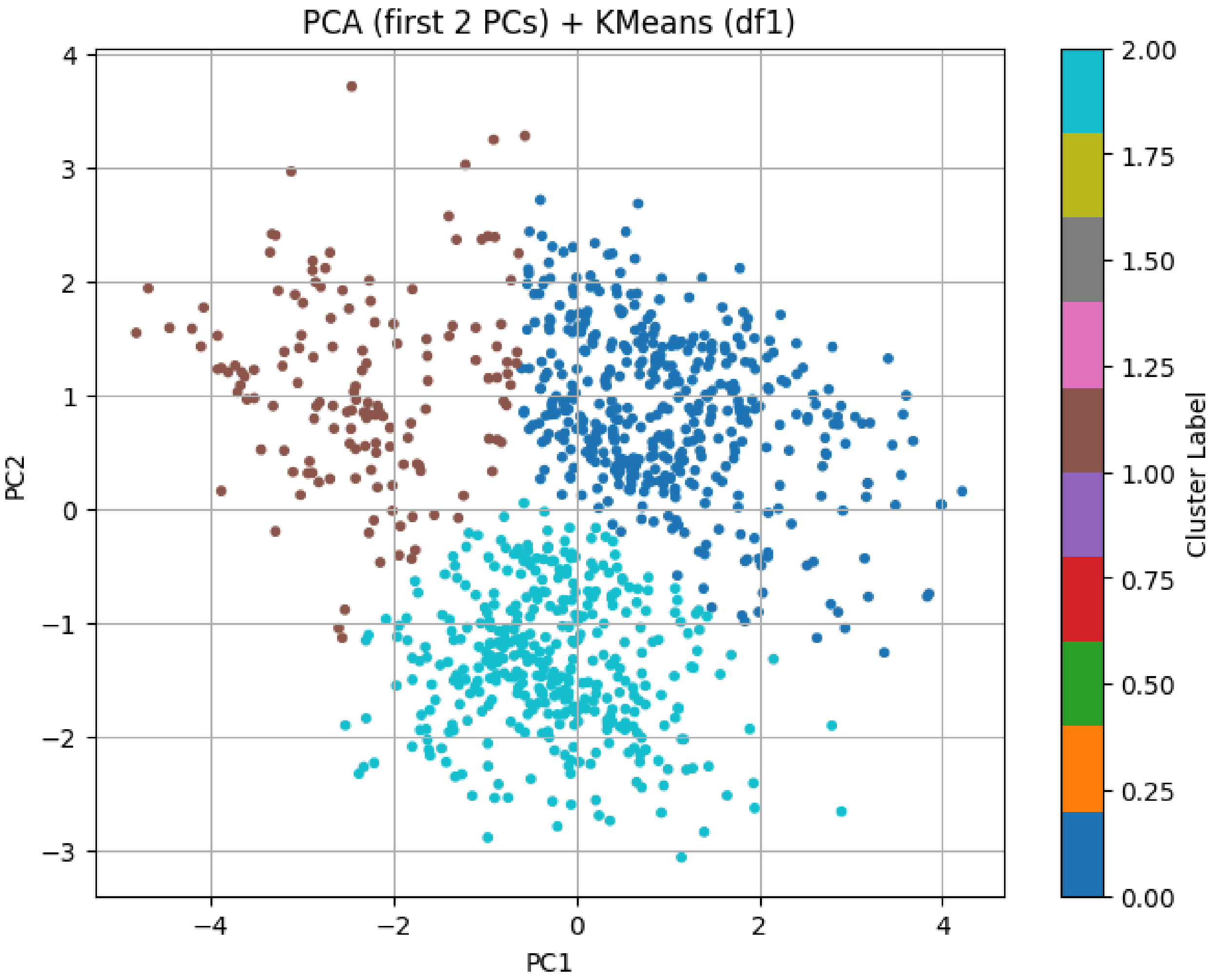


Figure 3.18 *K-Means Clustering Visualization using First 2 Principal Components (PCA)*

After applying PCA to reduce dimensions, the chart above shows the clustering results using K-Means. Each color represents a different group.

The data was separated into five clusters with fairly clear boundaries, which means PCA kept the important features for clustering. Although there is some slight overlap between clusters, they are still visually distinguishable.

Combining PCA with K-Means made the visualization easier to understand and helped improve the clustering quality.

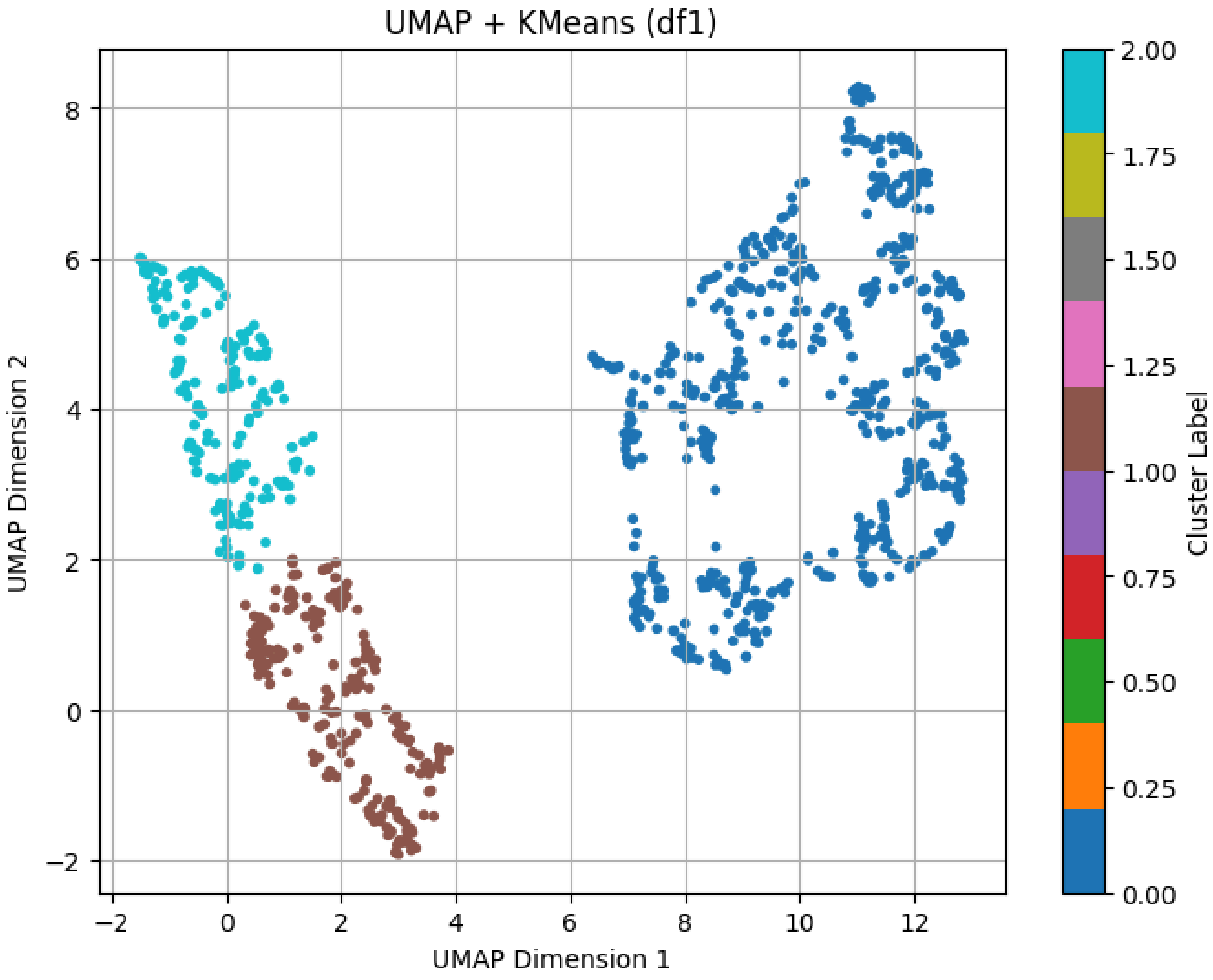


Figure 3.19 *K-Means Clustering Visualization using UMAP*

The chart illustrates how clusters are formed after applying UMAP for dimensionality reduction, followed by K-Means clustering. Each color represents a different cluster. This graph shows the result of using UMAP to reduce dimensions, then applying K-Means to group the data. Each color is a different cluster.

Compared to the PCA version, UMAP looks easier to read — the groups are more spread out and separated. UMAP seems to do a better job at showing how the data points actually relate to each other.

The clusters also look tighter and more organized, so overall UMAP gives a clearer view than PCA for this case.

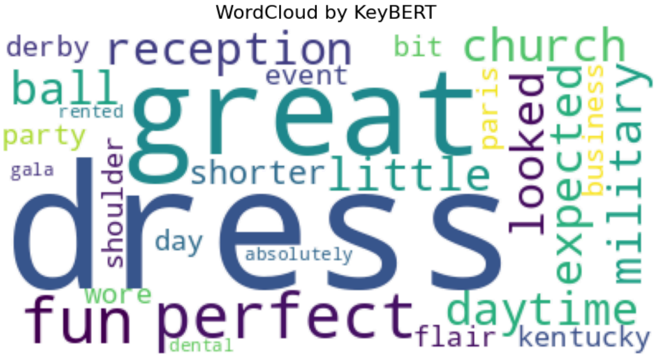
3.5. Word cloud for different clusters (TF-IDF and KeyBert)

To better understand the nature of each cluster, we performed a semantic analysis of the reviews by applying TF-IDF vectorization on the review_summary field within each cluster. This allowed us to extract meaningful textual patterns specific to each group.

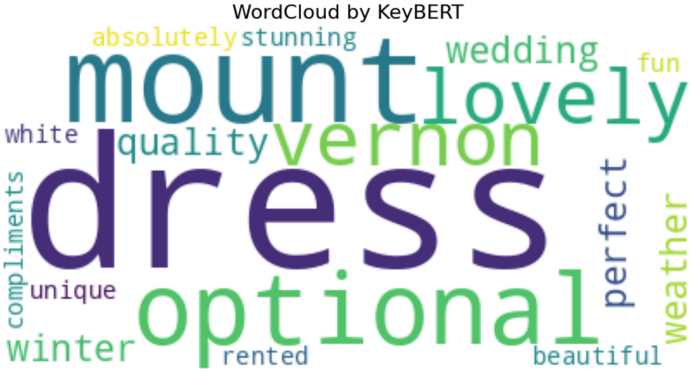
We then used KeyBERT to generate word clouds based on the TF-IDF vectors, highlighting the most representative keywords for each cluster. This step did not assign labels to the clusters but rather served as a visual interpretation tool to explore how users in each cluster described their experiences.

As shown in **Figure 3.5.1**, the word clouds provide insight into the common themes and contexts discussed in each group, supporting later steps like naming clusters or interpreting user fit preferences.

Cluster = 0:



Cluster = 1:



Cluster = 2:

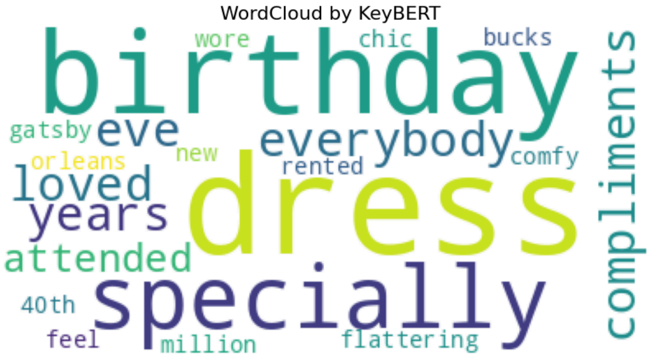
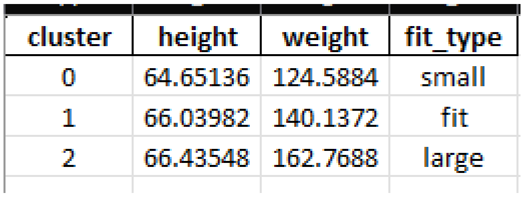


Figure 3.20 *WordClouds for Top Keywords in Each Cluster*

The three clusters (Clusters 0–2) are used in the official system, because these are clusters that are consistent with the results from the Elbow method and reflect enough differences needed to assign a ‘fit_type’, core feature used in Fallback, to suggest size when height, weight, age are missing.The following table presents the average height and weight for each cluster:



cluster	height	weight	fit_type
0	64.65136	124.5884	small
1	66.03982	140.1372	fit
2	66.43548	162.7688	large

- Cluster 0: This group has the lowest average height (**64.65**) and weight (**124.59**), indicating smaller body sizes. Therefore, it is labeled as "small".
- Cluster 1: This group has medium average height (**66.04**) and weight (**140.14**). It represents average or standard body sizes and is labeled as “fit".
- Cluster 2: With the highest average height (**66.44**) and weight (**162.77**), this group corresponds to larger body sizes and is labeled as "large".

This mapping allows the system to categorize users into appropriate size groups, improving recommendation reliability.

3.6. Clothing item and size recommendation

3.6.1 Apply Random Forest Regressor to find Size Recommendations

Random Forest Regressor is used to predict the clothing sizes that best fit a customer based on various measurements and user attributes.

Framework of the model:

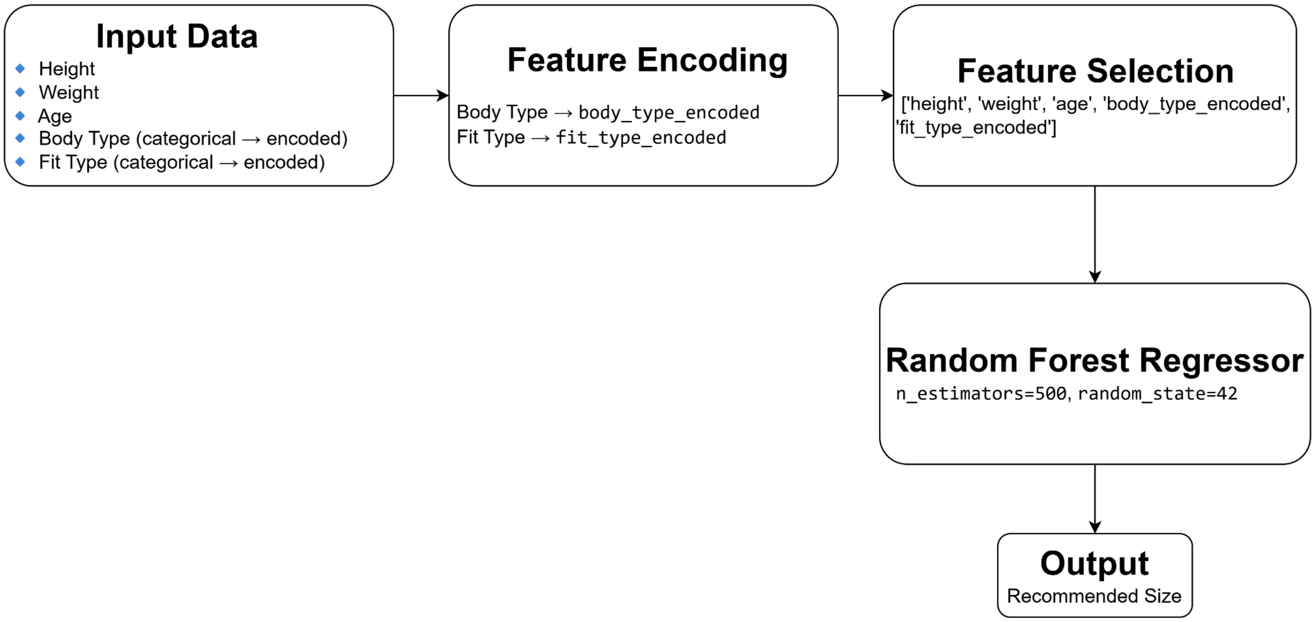

Input Data
Height
Weight
Age
Body Type (categorical → encoded)
Fit Type (categorical → encoded)
Feature Encoding
Body Type → body_type_encoded
Fit Type → fit_type_encoded
Feature Selection
['height', 'weight', 'age', 'body_type_encoded', 'fit_type_encoded']
Random Forest Regressor
n_estimators=500, random_state=42
Output
Recommended Size

Figure 3.21 Modeling Pipeline – Random Forest for Size Recommendation

After having inputs from the user (such as **age**, **height**, and **weight, body type,** and **fit type**) it will use this data to predict the most suitable size. If some of them are missings, the system will use Fallback instead of the main model.

3.6.2 Fallback

In cases where the user does not provide sufficient information for the prediction model (height, weight or age), the system will use a fallback mechanism to estimate the appropriate size based on the available data.

Specifically, this fallback mechanism relies on two features:

- 'body_type': entered directly by the user
- 'fit_type': inferred through K-Means clustering on the review data vectorized using TF-IDF

Each review cluster is assigned a corresponding 'fit_type' label ('small', 'fit', 'large') based on the characteristic keywords in that cluster **(Figure 3.20** – WordCloud). The system will then group users with the same 'body_type' and 'fit_type' to calculate the median size they have chosen.

This median value is used as a fallback size recommendation. In this way, the system can still make informed size suggestions based on similar user behavior and perception even when the main model is not working.

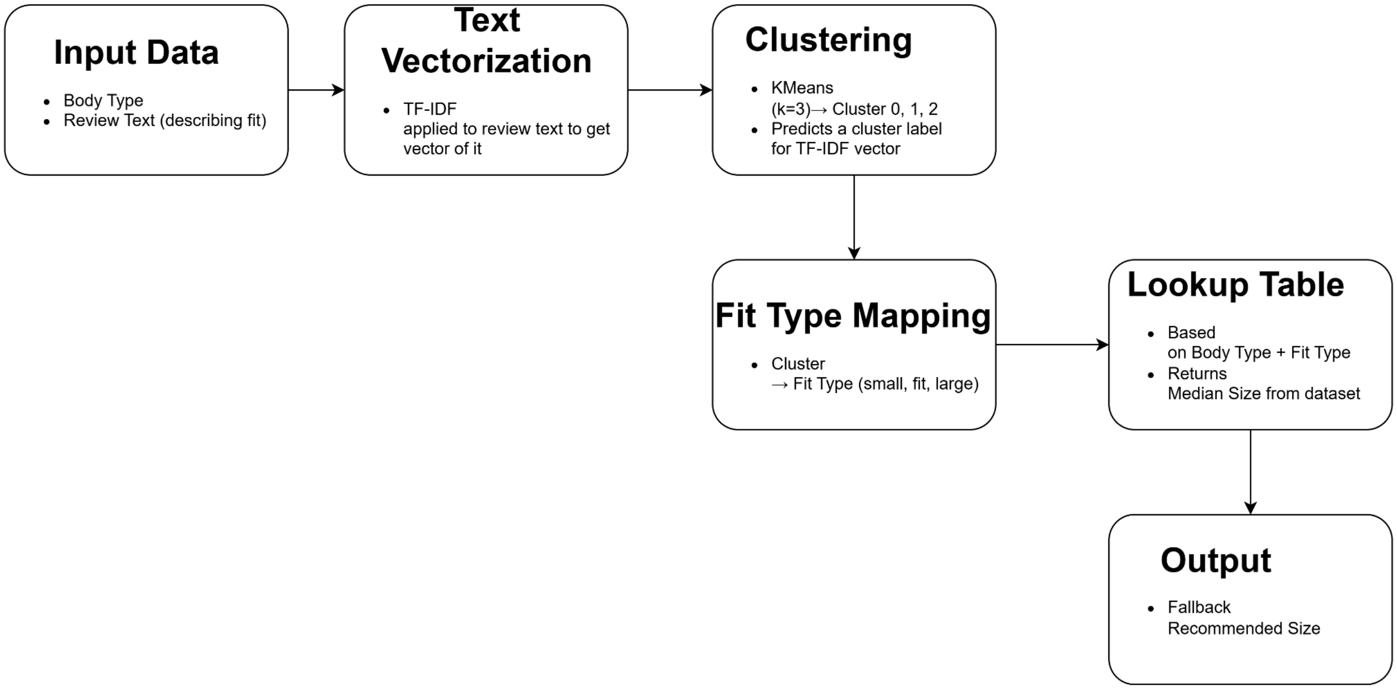

Input Data
Body Type
Review Text (describing fit)
Text Vectorization
TF-IDF applied to review text to get vector of it
Clustering
KMeans (k=3)→ Cluster 0, 1, 2
Predicts a cluster label for TF-IDF vector
Fit Type Mapping
Cluster → Fit Type (small, fit, large)
Lookup Table
Based on Body Type + Fit Type
Returns Median Size from dataset
Output
Fallback Recommended Size

Figure 3.22 *Fallback Flow – Size Recommendation via body_type and fit_type*

To assign a ‘fit_type’ to each review, we applied K-Means clustering (k=3) to the vectorized ‘review_summary’. The resulting clusters were named as ‘small’, ‘fit’, and ‘large’ based on their dominant keywords, identified via WordClouds.

3.6.3 Supplementary K-Nearest Neighbors (KNN) to recommend similar Items

K-Nearest Neighbors (KNN) is a commonly used algorithm in recommendation systems [13] to suggest similar items based on feature similarity. In this implementation, each product is first converted into a feature vector called combined_vectors, which includes both textual data (from review summaries via TF-IDF) and numerical attributes such as height, weight, age, rating, and sentiment score.

After the system displays the recommended size, it shows a list of available items from various categories. When the user selects an item, the system retrieves its corresponding vector and uses the KNN model to find other items that are close in vector space, indicating high similarity in terms of characteristics.

To ensure contextual relevance, the similarity search is limited to products within the same category. For example, dresses are compared with dresses, blazers are compared with blazers. The originally selected item (based on item_id) is excluded from the results, and up to five alternatives are shown. This approach ensures that the recommendations remain relevant and diverse while avoiding redundant suggestions.

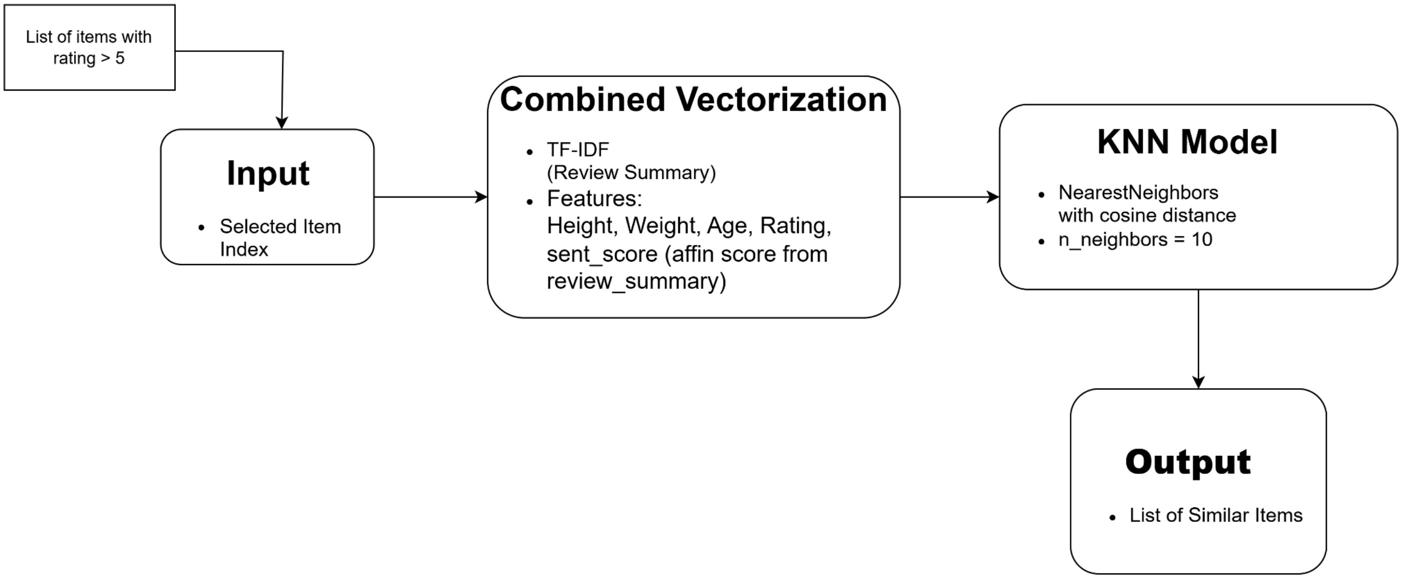

List of items with rating > 5
Input
Selected Item Index
Combined Vectorization
TF-IDF (Review Summary)
Features: Height, Weight, Age, Rating, sent_score (affin score from review_summary)
KNN Model
NearestNeighbors with cosine distance
n_neighbors = 10
Output
List of Similar Items

Figure 3.23 *Framework of KNN model*

3.7. Implementation platform and libraries

All data processing and model training were done in Jupyter Notebook. The Random Forest and Decision Tree models were built using Scikit-learn. Text data from “review_text” and “review_summary” was cleaned using NLTK and transformed into numerical form using TF-IDF.

Each word was given a value based on how important it was in the review. Other information such as height, weight, age, rating, and sentiment score was also included as features. These values were scaled to make them easier to combine with the text data.

After that, all features were merged into one dataset for further analysis. Clustering was performed using K-Means, and similarity between users was calculated using the scipy library. Users with a similarity score above 0.5 were placed in the same group or considered similar.

This method made it possible to consider both review content and user attributes when generating suggestions.

Note: All the steps above were applied on a tiny set extracted from big data (1,000 out of 192,544 were sampled) for testing before running on the whole set which needs much more time to execute.

CHAPTER 4: IMPLEMENTATION & RESULTS

4.1. Implementation

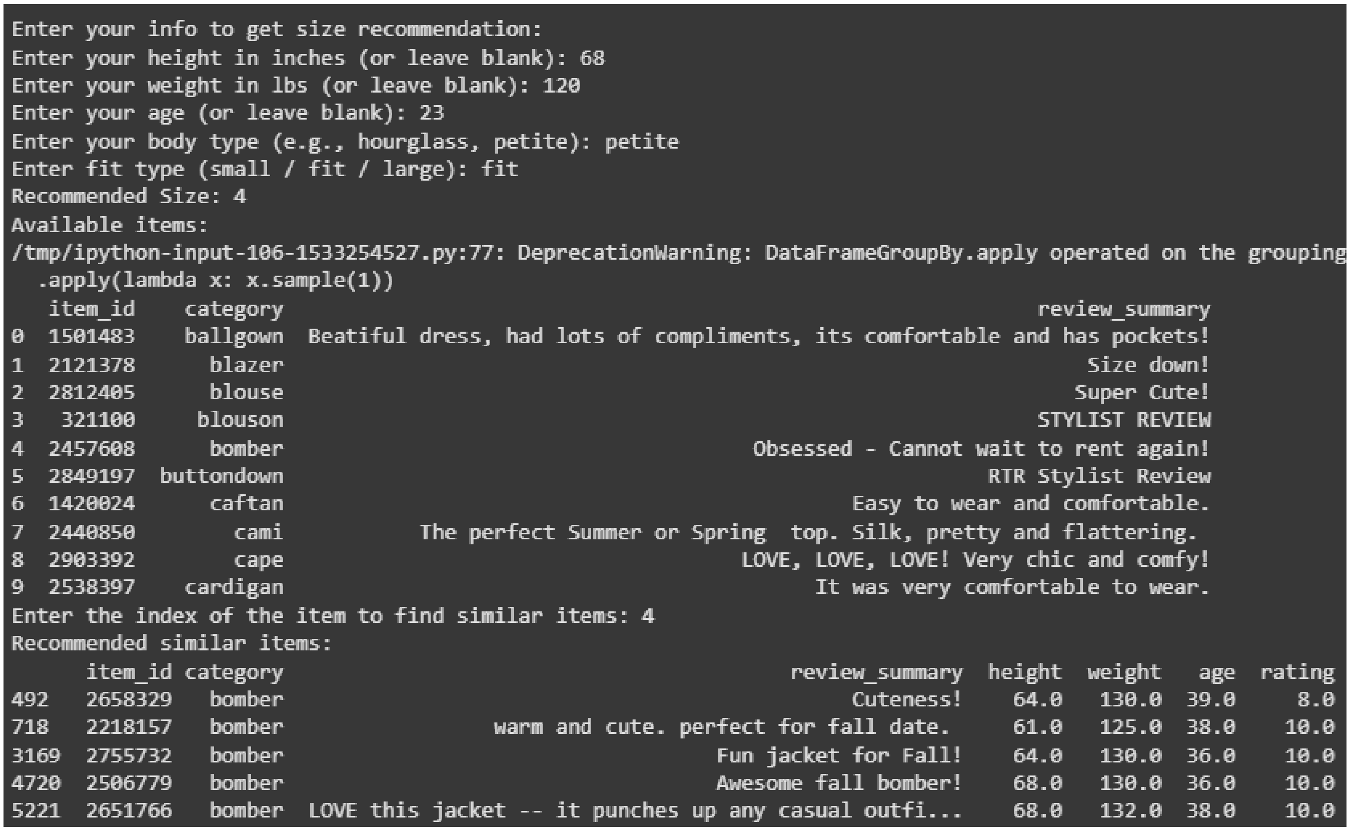
Implementation containing:

- An init method.
- A “pre_process” function which carries out all the preprocess tasks I mentioned previously.
- A “train” Function for Random Forest, Fallback and KNN for prediction tasks.
- “Get_similar_item” function: returns a dictionary where the key is the selected **item_id**, and the value is a list of similar items based on vector similarity. The function filters items within the same category and applies a KNN model using combined TF-IDF and numerical features to retrieve the most relevant alternatives.

4.2. Results

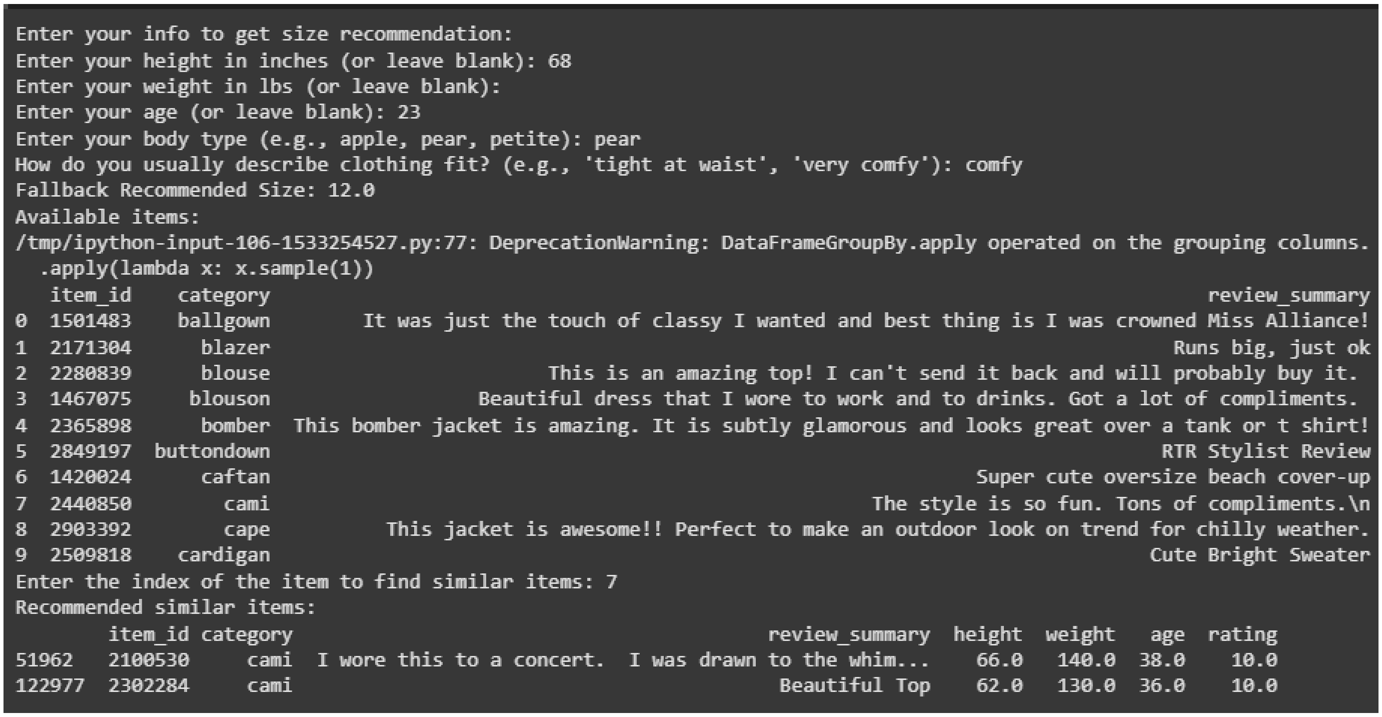
4.2.1 Code Execution Results

After running all the code mentioned above, we will receive the inputs required to enter three values provided by the user: age, weight, height, body_type and fit_type. When users done, a predicted size for them will come out as output



```
Enter your info to get size recommendation:
Enter your height in inches (or leave blank): 68
Enter your weight in lbs (or leave blank): 120
Enter your age (or leave blank): 23
Enter your body type (e.g., hourglass, petite): petite
Enter fit type (small / fit / large): fit
Recommended Size: 4
Available items:
/tmp/ipython-input-106-1533254527.py:77: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping
  .apply(lambda x: x.sample(1))
   item_id    category                                                review_summary
0  1501483    ballgown  Beatiful dress, had lots of compliments, its comfortable and has pockets!
1  2121378      blazer                                                    Size down!
2  2812405      blouse                                                   Super Cute!
3   321100     blouson                                                STYLIST REVIEW
4  2457608      bomber                           Obsessed - Cannot wait to rent again!
5  2849197  buttondown                                            RTR Stylist Review
6  1420024      caftan                                 Easy to wear and comfortable.
7  2440850        cami          The perfect Summer or Spring  top. Silk, pretty and flattering.
8  2903392        cape                           LOVE, LOVE, LOVE! Very chic and comfy!
9  2538397    cardigan                                It was very comfortable to wear.
Enter the index of the item to find similar items: 4
Recommended similar items:
      item_id category                                     review_summary  height  weight   age  rating
492   2658329   bomber                                         Cuteness!    64.0   130.0  39.0     8.0
718   2218157   bomber                    warm and cute. perfect for fall date.    61.0   125.0  38.0    10.0
3169  2755732   bomber                                Fun jacket for Fall!    64.0   130.0  36.0    10.0
4720  2506779   bomber                               Awesome fall bomber!    68.0   130.0  36.0    10.0
5221  2651766   bomber  LOVE this jacket -- it punches up any casual outfi...    68.0   132.0  38.0    10.0
```

Figure 4.1 *Size Prediction Results With Full Inputs (height, weight, age, 'body_type','fit_type')*



```
Enter your info to get size recommendation:
Enter your height in inches (or leave blank): 68
Enter your weight in lbs (or leave blank):
Enter your age (or leave blank): 23
Enter your body type (e.g., apple, pear, petite): pear
How do you usually describe clothing fit? (e.g., 'tight at waist', 'very comfy'): comfy
Fallback Recommended Size: 12.0
Available items:
/tmp/ipython-input-106-1533254527.py:77: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns.
  .apply(lambda x: x.sample(1))
   item_id    category                                                               review_summary
0  1501483    ballgown       It was just the touch of classy I wanted and best thing is I was crowned Miss Alliance!
1  2171304      blazer                                                              Runs big, just ok
2  2280839      blouse                       This is an amazing top! I can't send it back and will probably buy it.
3  1467075     blouson                 Beautiful dress that I wore to work and to drinks. Got a lot of compliments.
4  2365898      bomber  This bomber jacket is amazing. It is subtly glamorous and looks great over a tank or t shirt!
5  2849197  buttondown                                                             RTR Stylist Review
6  1420024      caftan                                                 Super cute oversize beach cover-up
7  2440850        cami                                         The style is so fun. Tons of compliments.\n
8  2903392        cape         This jacket is awesome!! Perfect to make an outdoor look on trend for chilly weather.
9  2509818    cardigan                                                            Cute Bright Sweater
Enter the index of the item to find similar items: 7
Recommended similar items:
        item_id category                                     review_summary  height  weight   age  rating
51962   2100530     cami  I wore this to a concert.  I was drawn to the whim...    66.0   140.0  38.0    10.0
122977  2302284     cami                                      Beautiful Top    62.0   130.0  36.0    10.0
```

Figure 4.2 *Fallback Size Recommendation (using 'body_type' and 'fit_type')*

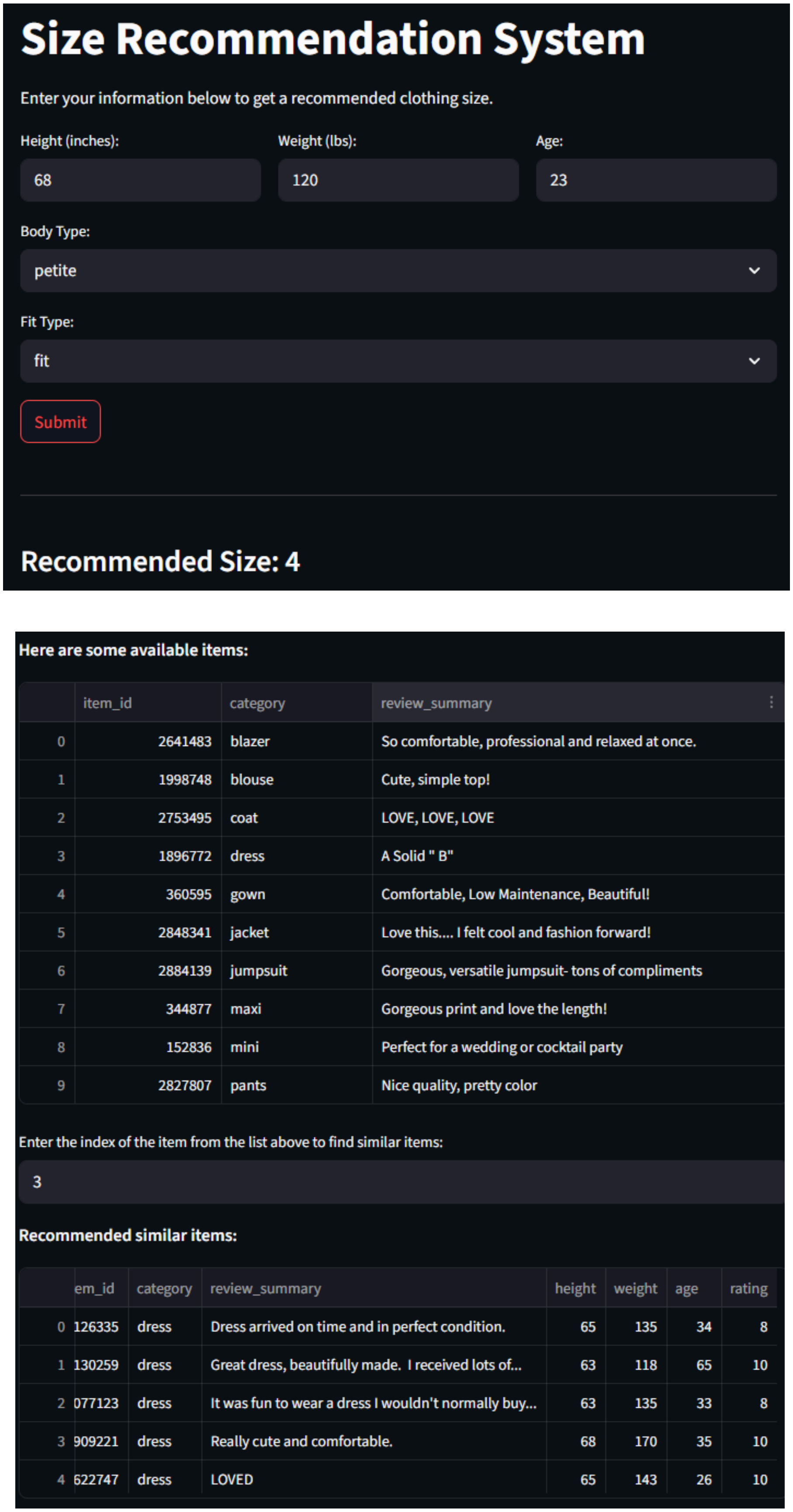
when missing input

When a user does not provide enough information like height, weight or age, the system can not use the machine learning model to predict size. Instead, it uses a Fallback. Understandably, the system looks for data on users with similar characteristics, in this case 'body_type' (pear, apple, petite…), and 'fit_type'(“beautiful”, “tight waist”, “very comfortable”…). From the filtered group, the system calculates the median size from the group of users with the same characteristics to estimate an appropriate size. As shown in the figure above, the “recommended size” is 12. This indicates that the result does not come from the predictive model but is instead inferred from statistical analysis of users with similar traits.

4.2.2 Web-based User Interface (Streamlit)

To demonstrate the functionality of the recommendation system in a user-friendly manner, a web-based user interface was developed using Streamlit. This interface enables users to input their attributes—such as height, weight, age, body type, and fit_type—and receive a recommended clothing size along with a list of similar items.

The UI replicates the logic of both the main prediction model and the fallback mechanism. When full input is provided, the Random Forest model is used to predict the appropriate size. If any key fields are missing, the fallback logic uses only body_type and fit_type to estimate the median size based on similar users



Size Recommendation System

Enter your information below to get a recommended clothing size.

Height (inches): 68

Weight (lbs): 120

Age: 23

Body Type: petite

Fit Type: fit

Recommended Size: 4

Here are some available items:

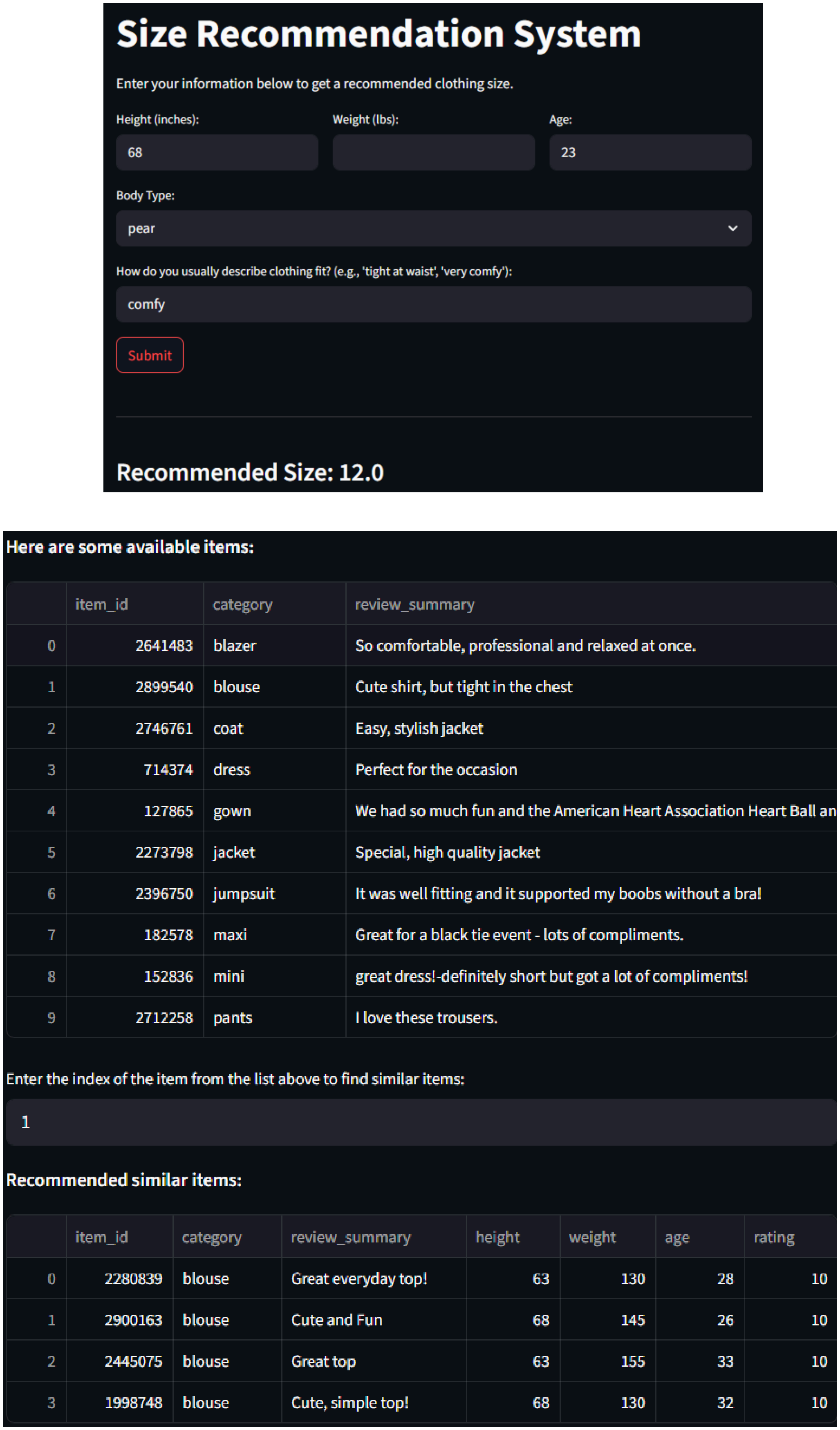
	item_id	category	review_summary
0	2641483	blazer	So comfortable, professional and relaxed at once.
1	1998748	blouse	Cute, simple top!
2	2753495	coat	LOVE, LOVE, LOVE
3	1896772	dress	A Solid " B"
4	360595	gown	Comfortable, Low Maintenance, Beautiful!
5	2848341	jacket	Love this.... I felt cool and fashion forward!
6	2884139	jumpsuit	Gorgeous, versatile jumpsuit- tons of compliments
7	344877	maxi	Gorgeous print and love the length!
8	152836	mini	Perfect for a wedding or cocktail party
9	2827807	pants	Nice quality, pretty color

Enter the index of the item from the list above to find similar items: 3

Recommended similar items:

	em_id	category	review_summary	height	weight	age	rating
0	126335	dress	Dress arrived on time and in perfect condition.	65	135	34	8
1	130259	dress	Great dress, beautifully made. I received lots of...	63	118	65	10
2	077123	dress	It was fun to wear a dress I wouldn't normally buy...	63	135	33	8
3	909221	dress	Really cute and comfortable.	68	170	35	10
4	622747	dress	LOVED	65	143	26	10

Figure 4.3 *User Interface with Full Input (Main Model)*



Size Recommendation System

Enter your information below to get a recommended clothing size.

Height (inches): 68

Weight (lbs):

Age: 23

Body Type: pear

How do you usually describe clothing fit? (e.g., 'tight at waist', 'very comfy'): comfy

Recommended Size: 12.0

Here are some available items:

	item_id	category	review_summary
0	2641483	blazer	So comfortable, professional and relaxed at once.
1	2899540	blouse	Cute shirt, but tight in the chest
2	2746761	coat	Easy, stylish jacket
3	714374	dress	Perfect for the occasion
4	127865	gown	We had so much fun and the American Heart Association Heart Ball an
5	2273798	jacket	Special, high quality jacket
6	2396750	jumpsuit	It was well fitting and it supported my boobs without a bra!
7	182578	maxi	Great for a black tie event - lots of compliments.
8	152836	mini	great dress!-definitely short but got a lot of compliments!
9	2712258	pants	I love these trousers.

Enter the index of the item from the list above to find similar items: 1

Recommended similar items:

	item_id	category	review_summary	height	weight	age	rating
0	2280839	blouse	Great everyday top!	63	130	28	10
1	2900163	blouse	Cute and Fun	68	145	26	10
2	2445075	blouse	Great top	63	155	33	10
3	1998748	blouse	Cute, simple top!	68	130	32	10

Figure 4.4 *User Interface with Missing Input (Fallback Logic)*

4.3. Evaluation Metrics (MAE, MSE, R^2)

To evaluate the performance of the size prediction model, three regression metrics are commonly used are Mean Absolute Error(MAE), Mean Squared Error(MSE), and Coefficient of Determination(R^2) as the following formula:

Mean Absolute Error(MAE):

$$MAE = \frac{1}{n}\sum_{I=1}^{n}\left|Y_i - \hat{Y}_i\right|$$

Mean Squared Error(MSE):

$$MSE = \frac{1}{n}\sum_{I=1}^{n}(y_i - \hat{y}_i)^2$$

Coefficient of Determination(R^2):

$$R^2 = 1 - \frac{\sum_{I=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{I=1}^{n}(y_i - \bar{y})^2} \quad \text{or} \quad R^2 = 1 - \frac{RSS}{TSS}$$

where TSS $= \Sigma(y_i - \bar{y})^2$

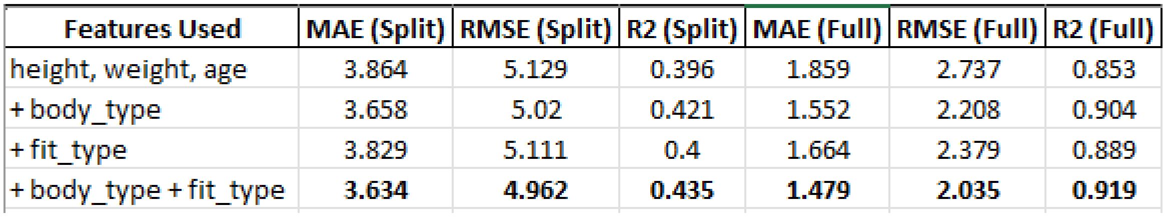
RSS $= \Sigma(y_i - \hat{y}_i)^2$

Table 4.3. Notation and Definition of Regression Evaluation Metrics

Notation	Definition
y_i	The actual value of the target variable at observation i.
$\hat{y}_i$	The predicted value of the target variable at observation i.
$\bar{y}$	The mean of all actual values.
n	The total number of observations or data points.
MAE	The average of the absolute differences between actual and predicted values.
MSE	The average of the squared differences between actual and predicted values.

RSS	Total squared error between actual and predicted values
TSS	Total squared deviation from the mean
R^2	To measure the proportion of variance in the actual values explained by the model

These metrics help check the model's accuracy and its ability to fit the actual data.



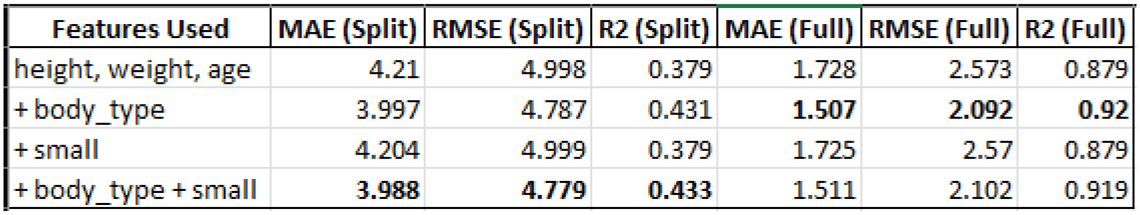
Features Used	MAE (Split)	RMSE (Split)	R2 (Split)	MAE (Full)	RMSE (Full)	R2 (Full)
height, weight, age	3.864	5.129	0.396	1.859	2.737	0.853
+ body_type	3.658	5.02	0.421	1.552	2.208	0.904
+ fit_type	3.829	5.111	0.4	1.664	2.379	0.889
+ body_type + fit_type	**3.634**	**4.962**	**0.435**	**1.479**	**2.035**	**0.919**

Figure 4.5 *Models Performance by Feature Combination.*

It can be seen that the full use of the variables height, weight, age, body_type and fit_type can give the best results on all three metrics MAE, RMSE and R². Specifically, this combination achieves the lowest absolute error (MAE = 1.479), the lowest root square error (RMSE = 2.035), the highest coefficient of determination (R² = 0.919), helps reduce large prediction errors in the data.

Meanwhile, the models that only use a specific part such as ‘body_type’ or ‘fit_type’ individually, although improved compared to the baseline (height, weight, age), still do not achieve optimal performance.

Besides the ensemble model used on the entire dataset, there were models also trained based on clusters of the 'fit_type' feature ('small', 'fit' and 'large'), to investigate whether localized modeling could yield better prediction accuracy.

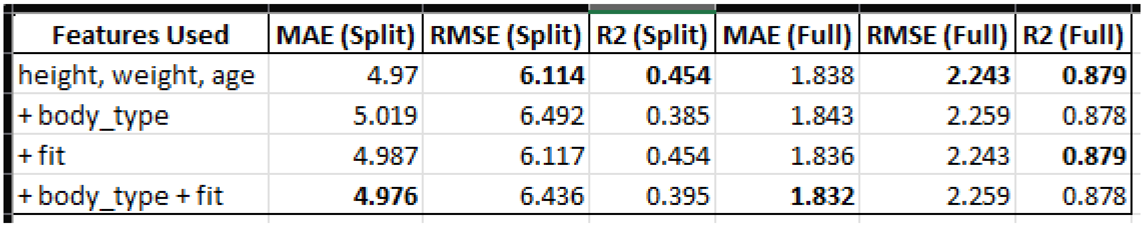


Features Used	MAE (Split)	RMSE (Split)	R2 (Split)	MAE (Full)	RMSE (Full)	R2 (Full)
height, weight, age	4.21	4.998	0.379	1.728	2.573	0.879
+ body_type	3.997	4.787	0.431	**1.507**	**2.092**	**0.92**
+ small	4.204	4.999	0.379	1.725	2.57	0.879
+ body_type + small	**3.988**	**4.779**	**0.433**	1.511	2.102	0.919

Figure 4.6 *Model Performance in Cluster ‘small’*

MAE = 1.511, **RMSE = 2.047**, **R² = 0.919**

Although R² matches the general model, both MAE and RMSE are slightly worse than the main model. The RMSE value in particular suggests the presence of a few large errors that are not well-handled by the localized model.

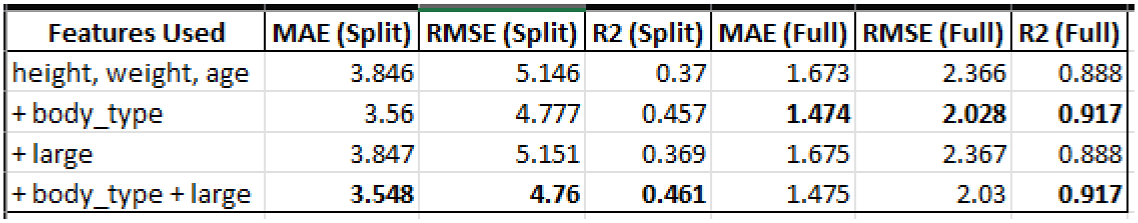


Features Used	MAE (Split)	RMSE (Split)	R2 (Split)	MAE (Full)	RMSE (Full)	R2 (Full)
height, weight, age	4.97	**6.114**	**0.454**	1.838	**2.243**	**0.879**
+ body_type	5.019	6.492	0.385	1.843	2.259	0.878
+ fit	4.987	6.117	0.454	1.836	2.243	**0.879**
+ body_type + fit	**4.976**	6.436	0.395	**1.832**	2.259	0.878

Figure 4.7 *Model Performance in Cluster 'fit'*

MAE ≈ 1.834, **RMSE ≈ 2.641**, **R² ≈ 0.878**

RMSE is significantly higher than the general model, indicating that the 'fit' cluster suffers from larger individual errors and lacks coherent patterns for effective prediction This is also the most inconsistent and least promising cluster. Regardless of the combination of features used.



Features Used	MAE (Split)	RMSE (Split)	R2 (Split)	MAE (Full)	RMSE (Full)	R2 (Full)
height, weight, age	3.846	5.146	0.37	1.673	2.366	0.888
+ body_type	3.56	4.777	0.457	**1.474**	**2.028**	**0.917**
+ large	3.847	5.151	0.369	1.675	2.367	0.888
+ body_type + large	**3.548**	**4.76**	**0.461**	1.475	2.03	**0.917**

Figure 4.8 *Model Performance in Cluster 'large'*

MAE = 1.475, **RMSE = 2.057**, **R² = 0.917**

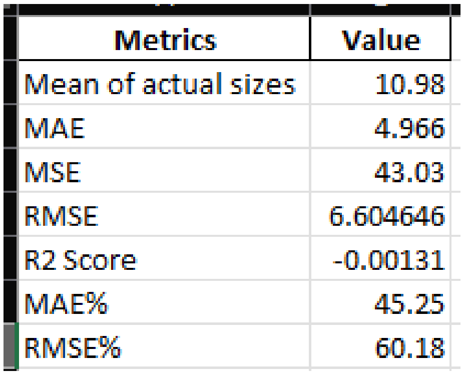
While the MAE is nearly identical to the main model, the higher RMSE implies that some prediction errors are not handled. Therefore, this model still cannot consistently outperform the general model.

The cluster-based models perform slightly worse than the general model, none of them outperform the general model across all dimensions, It can be seen that:

- ➔ RMSE values in the cluster-based models are consistently higher, showing their sensitivity to outliers and instability in error distribution.
- ➔ The ‘fit’ cluster shows that if the split is too large, the model may underperform and the prediction results may be unstable.
- ➔ The general model illustrates its stability, maintainability, and also is better at handling a wide range of cases without the need for segmentation.

Overall, the general model is the most effective, reliable and deployable for size recommendation.

Compared to size prediction with Fallback:



Metrics	Value
Mean of actual sizes	10.98
MAE	4.966
MSE	43.03
RMSE	6.604646
R2 Score	-0.00131
MAE%	45.25
RMSE%	60.18

Figure 4.9 *Fallback Model Performance with Partial Input ('body_type' and 'fit_type' only)*

It can be seen that when the model is provided with complete input data, it will perform more accurately compared to cases where only partial information is given and the size recommendation relies on body type. This is because body type is not a key factor in determining user size; it merely serves as support when there is a lack of input. As a result, the accuracy also decreases when users do not provide enough personal information.

Therefore, using body type for size recommendation is possible, but the result is quite vague. It should run in parallel with the main model, as this approach helps balance both coverage and accuracy, limits cold-start issues in recommendation systems, and still allows the model to prioritize predictions based on more complete and reliable data.

CHAPTER 5: CONCLUSIONS

In conclusion, this is a size recommendation system that combines machine learning and its Fallback. The main model uses the Random Forest Regressor algorithm to predict size from user characteristics such as height, weight, age, body type, and fit type giving accurate results with low average deviation (MAE = 1.479) and high R^2 score ($R^2 = 0.919$), which means that almost 92% of the variation in size can be explained by the model.

When users do not give complete information, the system will switch from the main model (RF) to the Fallback mechanism, using the combination of 'body_type' and 'fit_type' (assigned from the review clustering using TF-IDF and K-Means). The Fallback size is calculated based on the median of the group of users with similar characteristics.

Furthermore, the system also supports similar product recommendations by applying KNN on product review vectorization. This helps improve the user experience, even when the predicted size is not accurate.

Future development directions may include integrating hybrid models, using deep learning networks, which has recently proven to be effective in the fields of information retrieval and recommender system research [14], and upgrading the Fallback system with more sophisticated clustering. Testing on a larger user base will also help the system achieve higher accuracy and generality.

REFERENCES

[1] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining (WSDM '20)*. Association for Computing Machinery, New York, NY, USA.

[2] Debashis Das, Laxman Sahoo and Sujoy Datta. 2017. A Survey on Recommendation System. *International Journal of Computer Applications (0975 – 8887) Volume 160 – No 7*

[3] Paul Sheridan, Mikael Onsj. 2023. In *The hypergeometric test performs comparably to TF-IDF on standard text analysis tasks*

[4] Chunqiong Wu, Bingwen Yan, Rongrui Yu, Baoqin Yu, Xiukao Zhou, Yanliang Yu, Na Chen. 2021. In *K-Means Clustering Algorithm and Its Simulation Based on Distributed Computing Platform*.

[5] Zaira Hassan Amur, Yew Kwang Hooi, Gul Muhammad Soomro, Hina Bhanbhro, Said Karyem, Najamudin Sohu. 2023. In *Unlocking the Potential of Keyword Extraction: The Need for Access to High-Quality Datasets.*

[6] Zaira Hassan Amur, Yew Kwang Hooi, Gul Muhammad Soomro, Hina Bhanbhro, Said Karyem, Najamudin Sohu. 2023. *Unlocking the Potential of Keyword Extraction: The Need for Access to High-Quality Datasets*

[7] Chenmeng Zhang, Can Hu, Shijun Xie1, Shuping Cao. 2020 . In *Research on the application of Decision Tree and Random Forest Algorithm in the main transformer fault evaluation.*

[8] Jehad Ali, Rehanullah Khan, Nasir Ahmad, Imran Maqsood. 2012. In *Random Forests and Decision Trees* .

[9] Shiqiang Lin. 2023. In *Understanding Principal Component Analysis.*

[10] Leland McInnes, John Healy, James Melville. 2020. In *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*.

[11] Adeiza James Onumanyi, Daisy Nkele Molokomme, Sherrin John Isaac, Adnan M. Abu-Mahfouz. 2022. In *AutoElbow: An Automatic Elbow Detection Method for Estimating the Number of Clusters in a Dataset*.

[12] Fatima Batool, Christian Hennig. 2020. In *Clustering with the Average Silhouette Width*.

[13] Badr Hssina, Abdelakder Grota, Mohammed Erritali. 2021. In *Recommendation system using the k-nearest neighbors and singular value decomposition algorithms*.

[14] Shuai Zhang, Lina Yao, Aixin Sun, Yi Tay. 2019. In *Deep Learning Based Recommender System: A Survey and New Perspectives.*