IEOR 4523: Data Analytics

Recommender Systems for Rent The Runway

Columbia University in the City of New York

Aditi Khandelval (ak4426)
Aneesh Goel (ag4182)
Neeraj Ramkumar (nr2728)
Skand Upmanyu (su2236)

1. Abstract

Rent the Runway has a unique business model compared to other ecommerce websites. It provides designer clothing and accessory rentals as opposed to retailing them. In this business model, a customer can select a dress or an accessory online and can rent it for a fraction of the retail price. A large proportion of e-commerce customers are involved in window shopping and are indecisive about the products they need to buy. In such situations, it is imperative for the company to capitalize on this opportunity and display the right product to the customer at the right time to maximize profits and revenues. For our use case, it's particularly necessary because Rent The Runway has close to 18000 products listed on their site. Therefore, we will try to create a solution by developing recommendation systems for the company.

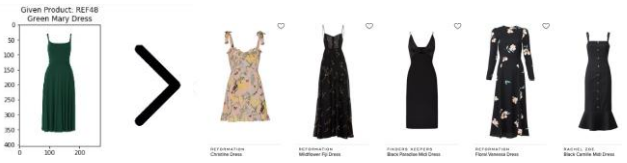
2. Introduction

Our objective with this project is to recommend the top 5 products for a user in 2 particular situations:

1. Showing similar products when the company has prior knowledge about what kinds of products the user is viewing on the website. On e-commerce websites, this is generally displayed as the "You may also like" section of the page
2. Showing recommended products on the landing page of the website based on prior knowledge about the user's preferences. Again, on e-commerce websites, this falls under the "Recommended for you" section on the landing page

3. Related work and research

We explored the recommendations of products on the website of Rent the Runway and identified certain areas which can be improved. An example is shown below:



Here, we identified certain shortcomings in the model like it does not take into account the color of the product and certain other attributes. These are the areas which we would like to improve upon by creating a new recommendation system.

4. Content-based filtering

4.1. Methodology

For content-based filtering, we find similar products for the product that is being viewed by the user and recommend the top 5 similar products to the user. For calculating this similarity, we used the cosine formula as follows:

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \, \|\mathbf{B}\| \cos\theta$$

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

where A_i and B_i are components of vector A and B respectively. This cosine measure gives us the similarity score for every product with every other product.

4.2. Data collection

We initially used an online dataset [1]. However, there were certain shortcomings with this data as mentioned below:

1. Incomplete data: The dataset had limited products and limited users
2. Missing pictorial representations
3. Missing references to the actual product on the website
4. Limited user and product attributes
5. Large proportion of missing values

To overcome this obstacle, we changed our data source to the actual website and scraped the review page of each product using the *selenium* package in Python. We used Google Collaboratory [2] for sharing code with the group members and for higher computational capabilities.

Since a limited number of products are loaded on the website when we go to the products page, we used a driver to get the page source after adding a sleep for 2 seconds.

[1] Source: https://cseweb.ucsd.edu/~jmcauley/datasets.html#clothing_fit
[2] Source: https://colab.research.google.com/

This was carried out as follows:

```
driver.get(url)
time.sleep(2)
response_content = driver.page_source
results_page = \
        BeautifulSoup(response_content,'lxml')
```

We then found the 'div' tags using the BeautifulSoup object to get the product and user attributes. We also stored the image links which can be used for model validations.

This process was carried out in batches with a batch size of 10 pages with each page containing 60 products. The total number of pages scraped were 301.

Value	Attribute
Platform	Google Collaboratory
Package for scraping	Selenium, BeautifulSoup
#Pages	301
#Products	16,751
Batch size	600 products

4.3. Attributes captured

Value	Attribute
Designer_name	Designer (categorical)
FormalityScore	1-10 identifies formal clothes (cont.)
Price_base	Base price (cont.)
Price_adjusted	Offer price (cont.)
AgeRanges	Appropriate age range (categorical list)
BodyTypes	Appropriate body types (categorical list)
Colors	Colors in the product (categorical list)
Formality	Identifies formal clothes (categorical list)
Occasions	Appropriate occasions (categorical list)
Embellishments	Pockets, beads etc. (categorical list)
AverageRating	Average rating of the product (cont.)
CountRatings	No. of ratings received (cont.)
Retail_price_value	Retail price (cont.)
Length	Length of dress (categorical)
Neckline	Neck type (categorical)
Season	Appropriate seasons (categorical list)
Sleeve	Sleeve length (categorical)
Product_age_years	Age of the product (cont.)

4.4. Data pre-processing

Web scraping data from the website provided us with the json object which we converted to python dictionaries. We then found the relevant product attribute keys and their values to create a structured python dictionary. This was then converted to a pandas dataframe.

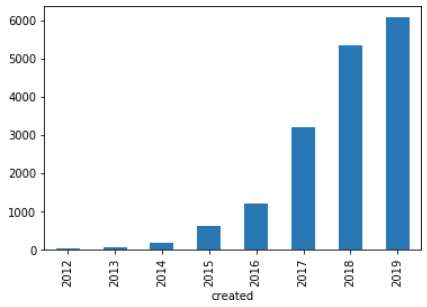
We then imputed the missing values by replacing these values by mean for numerical variable and by the mode for categorical variables. These categorical variables were then converted to one hot encodings for model creation.

Below is the snapshot of the dataframe (16,751 x 101):

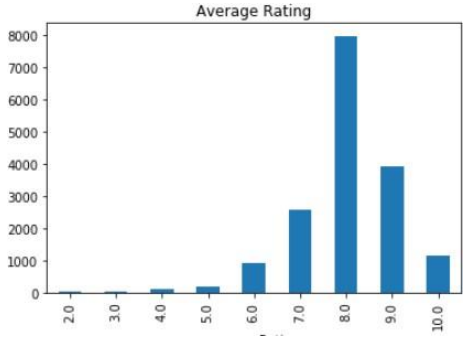
	designer_name	displayName	formalityScore	price_base	price_adjusted
FLN35	Flynn Skye	Maiden Mini Dress	3.0	40.0	30.0
RMB77	Ramy Brook	Silver Lori Top	5.0	50.0	50.0
KPL81	The Kooples	Leopard Robe Dress	3.0	95.0	85.0
STS90	STYLESTALKER	Olive Velvet Leila Jumpsuit	5.0	45.0	35.0
PR125	Proenza Schouler	Crocodile Print Matte Jersey Dress	3.0	295.0	285.0

4.5 Data visualization

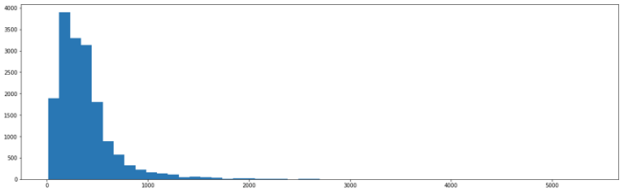
- No. of products for every year on Rent The Runway:



- Rating distribution:

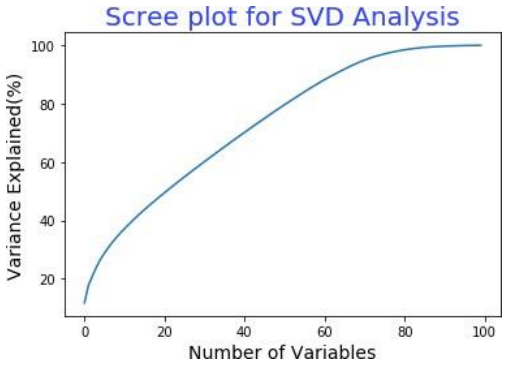


- Price distribution:



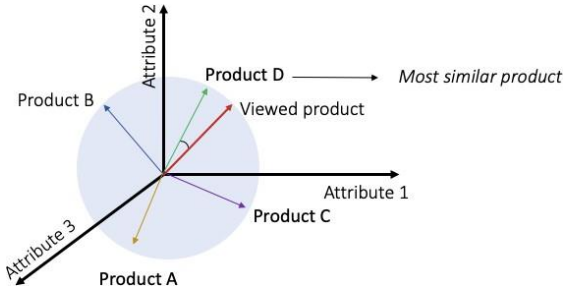
4.5 Principal Component Analysis (PCA)

Since the data has 101 variables and have high correlation, we first standardized our data and carried out Principal Component Analysis to reduce the number of dimensions and to get orthogonal axes before feeding the data into the model.



From the above Scree plot we identified that 75 variables were able to capture 97.1% of the variance in the data. Therefore, we reduced the number of dimensions to 75 using Singular Value Decomposition (SVD).

4.6 Model creation



In the graph above, each product is a unit vector and is described by 3 attributes in vector space. We find the cosine of the angle between the given product and all other products to calculate product similarity. Higher cosine value indicates higher similarity of the product.

We extended this idea to multiple dimensions and calculated the cosine similarity for all the products to create a cosine matrix as follows:

	FLN35	RMB77	KPL81	STS90	PR125
FLN35	1.000000	-0.022735	0.458811	0.023956	0.242897
RMB77	-0.022735	1.000000	0.033126	0.318313	0.052850
KPL81	0.458811	0.033126	1.000000	0.107864	0.541722
STS90	0.023956	0.318313	0.107864	1.000000	-0.088791
PR125	0.242897	0.052850	0.541722	-0.088791	1.000000

Dimension: 16,751 x 16,751

4.7 Results

We found interesting results of the model for which some samples are shown below:

We can observe from these results that our system is able to identify similar products. For clutches, it shows similar clutches, for winter clothes, it shows similar winter clothes and for leopard prints, it shows similar printed products. We also observe that the model takes into account the color and texture of the product to give better recommendations.

We carried out these recommendations for 500 sample prodcuts which can be viewed in the attached jupyter notebook.

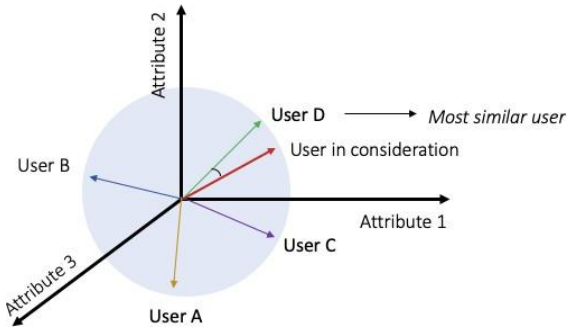
5. Discussion

The model created above provides recommendations based on the prior knowledge of the products that the customer is viewing at any instance. But in certain situations, the customers visit the website for window shopping and are indecisive about the products they want to view. Moreover, we also want to recommend products on the landing page of the website. In these situations, we do not have any prior knowledge about the current viewing behavior of the customer. To create a solution for this problem, we create a recommendation system based on collaborative filtering.

6. Collaborative filtering

6.1. Methodology

For collaborative filtering, we find similar users for the current user and recommend the products that are liked by the similar user but have not been rated by the current user. For calculating the user similarity we use the same approach as we used in content-based filteing.



6.2. Attributes captured

Value	Attribute
Age	Age of the user (cont.)
NumReviewsByUser	No. of reviews by the user (cont.)
HeightInches	Height of the user (cont.)
USStandardSize	Usual size the user wears (cont.)
WeightPounds	Weight of the user (cont.)
Bust_size	Bust size of the user (cont.)
BodyType	Body type of the user (categorical)
Fit	Usual fit of the user (categorical)
Bust_type	Bust type of the user (categorical)
Product rating	Rating for every product

After data pre-processing and missing value treatment, the final dataframe looked as follows:

	user_id	age	numReviewsByUser	heightInches	usStandardSize	weightPounds
0	16844044	24.0	3	65.0	4.0	141.310627
1	17725300	29.0	44	61.0	4.0	125.000000
2	19431324	21.0	4	66.0	6.0	155.000000
3	10992590	30.0	9	67.0	6.0	150.000000
4	12064814	40.0	10	65.0	10.0	164.000000
5	19240875	26.0	5	63.0	4.0	141.310627
6	18524231	23.0	9	66.0	4.0	150.000000
7	6382654	30.0	205	67.0	0.0	118.000000

Dimension :13,497 x 14,925 sparse matrix

Since the dataframe is really huge to store the matrix, we used sklearn's sparse matrix to store this data.

The cosine matrix, created by the similar approach as before is shown below:

	12247768	10979088	7840766	1060714	1752866
12247768	1.000000	0.026409	0.014369	0.020789	0.051383
10979088	0.026409	1.000000	0.005254	0.021410	0.025631
7840766	0.014369	0.005254	1.000000	0.024412	0.017932
1060714	0.020789	0.021410	0.024412	1.000000	0.036506
1752866	0.051383	0.025631	0.017932	0.036506	1.000000

Dimension: 13,497 x 13,497
(filtered users with at least 5 rated products to have enough data about the users for making recommendations)

4.7 Results

We analyzed the results of collaborative filtering and a sample is show below:

Sample: Recommendations for user: 1233836

```
Top Similar User: 6496966
cosine: [0.15317217]
```

Similarites:

	variable	value_1233836	value_6496966
0	Product_ASI15	8.0	10.0
1	Product_GA33	10.0	10.0
2	Product_RZ37	8.0	10.0
0	age	46.0	36.0
1	numReviewsByUser	182.0	26.0
2	heightInches	67.0	63.0
3	usStandardSize	6.0	4.0
4	weightPounds	140.0	115.0
5	bust_size	36.0	34.0

Both the users have similar attributes and have liked the following 3 products:

Products rated by similar user but not by given user:

	variable	value_6496966	value_1233836
1	Product_ECZ3	10	NaN
4	Product_HB20	10	NaN
3	Product_GEN1	8	NaN
7	Product_RZ63	8	NaN
5	Product_KP91	4	NaN

Only 4 good products found. Therefore, we go to next best similar user for the next recommendation:

```
Top Similar User: 17976123
cosine: [0.14239687]
```

Similarites:

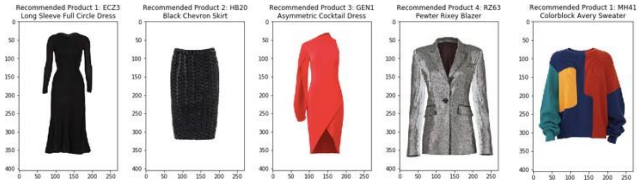
	variable	value_1233836	value_17976123
0	Product_EJ118	10.0	8.0
0	age	46.0	26.0
1	numReviewsByUser	182.0	37.0
2	heightInches	67.0	61.0
3	usStandardSize	6.0	2.0
4	weightPounds	140.0	120.0
5	bust_size	36.0	34.0
6	bust_type_C	1.0	1.0

Both the users have similar attributes and have liked the following 1 product:

Products rated by similar user but not by given user:

	variable	value_17976123	value_1233836
15	Product_MH41	10	NaN
3	Product_DL258	10	NaN
4	Product_DL80	10	NaN
6	Product_EJ225	10	NaN
10	Product_JOA51	10	NaN
1	Product_CLEHNB23	10	NaN
17	Product_PLL34	10	NaN
0	Product_BLNK16	8	NaN
24	Product_VIN101	8	NaN
23	Product_SW611	8	NaN
22	Product_SW537	8	NaN

Final recommendations:

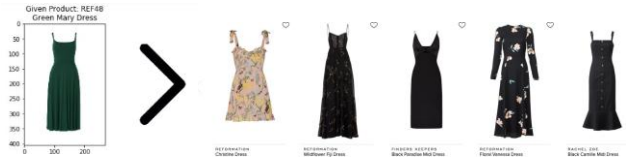


7. Conclusions

Our recommendation system is able to capture the product and user attributes to provide recommendations to the customers which we validated by running several iterations on multiple products. These recommendations were then compared with Rent The Runway's recommendations to validate the results and

to analyze if we were able to solve the issue of the colors and textures that we discussed earlier.
The recommendation comparison for the product is shows below:

Rent The Runway's recommendations:



Our model's recommendations:

Here, we can conclude that our model is now able to capture the colors of the products as well to provide better recommendations.

8. Future work and limitations

The following are the limitations of our project which can lead to future improvements:

1. Our model is created on publicly available data and does not take into account the internal data maintained by the company.
2. Our model is based on reviews of the customers. More data can be incorporated by adding viewing behavior of the customer by tracking activity on the website. This would involve data sharing from the company's end
3. We can also take into account user dislikes in order to remove products from recommendations which are similar to the products that the user has disliked
4. Can explore Jaccard similarity for calculating product and user similarities

9. References

1. Collaborative filtering recommender systems: https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0

2. Recomender systems: http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/itembased.html

3. Recommender systems (Book) by Charu C. Aggarwal

4. Document similarity: https://medium.com/@actsusanli/i-am-preparing-an-article-lda-document-similarity-content-based-recommender-system-f79f857e4b52