

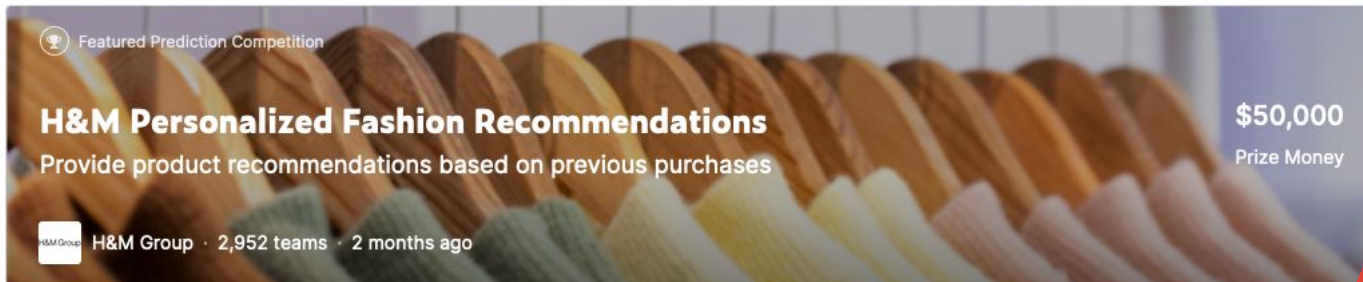
PERSONALISED FASHION RECOMMENDATIONS

H&M ONLINE SHOP

**A deep-learning solution
to customer-based product recommendations**

Nádia Carmona, Marika Erasmus & Stefanie Sturm

THE DATASET



Featured Prediction Competition

H&M Personalized Fashion Recommendations

Provide product recommendations based on previous purchases

\$50,000
Prize Money

H&M Group · 2,952 teams · 2 months ago

Overview **Data** Code Discussion Leaderboard Rules Team

My Submissions **Late Submission** ..

Data Description

For this challenge you are given the purchase history of customers across time, along with supporting metadata. Your challenge is to predict what articles each customer will purchase in the 7-day period immediately after the training data ends. Customer who did not make any purchase during that time are excluded from the scoring.

Files

- `images/` - a folder of images corresponding to each `article_id`; images are placed in subfolders starting with the first three digits of the `article_id`; note, not all `article_id` values have a corresponding image.
- `articles.csv` - detailed metadata for each `article_id` available for purchase

Data Explorer

34.56 GiB

- `images`
- `articles.csv`
- `customers.csv`
- `sample_submission.csv`
- `transactions_train.csv`

THE DATASET

articles.csv

- 25 columns of information (product name, product group, graphical appearance...)
- 105542 different items available in the online shop
- Unique product ID identifies each item

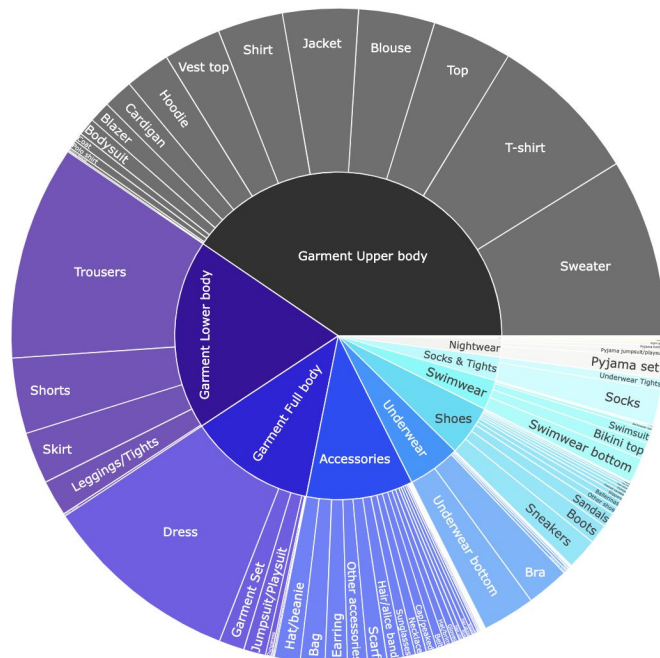
customers.csv

- 7 columns of information (age, membership status, postal code...)
- 1371980 customers that have interacted with the website
- Unique customer ID identifies each customer

transactions_train.csv

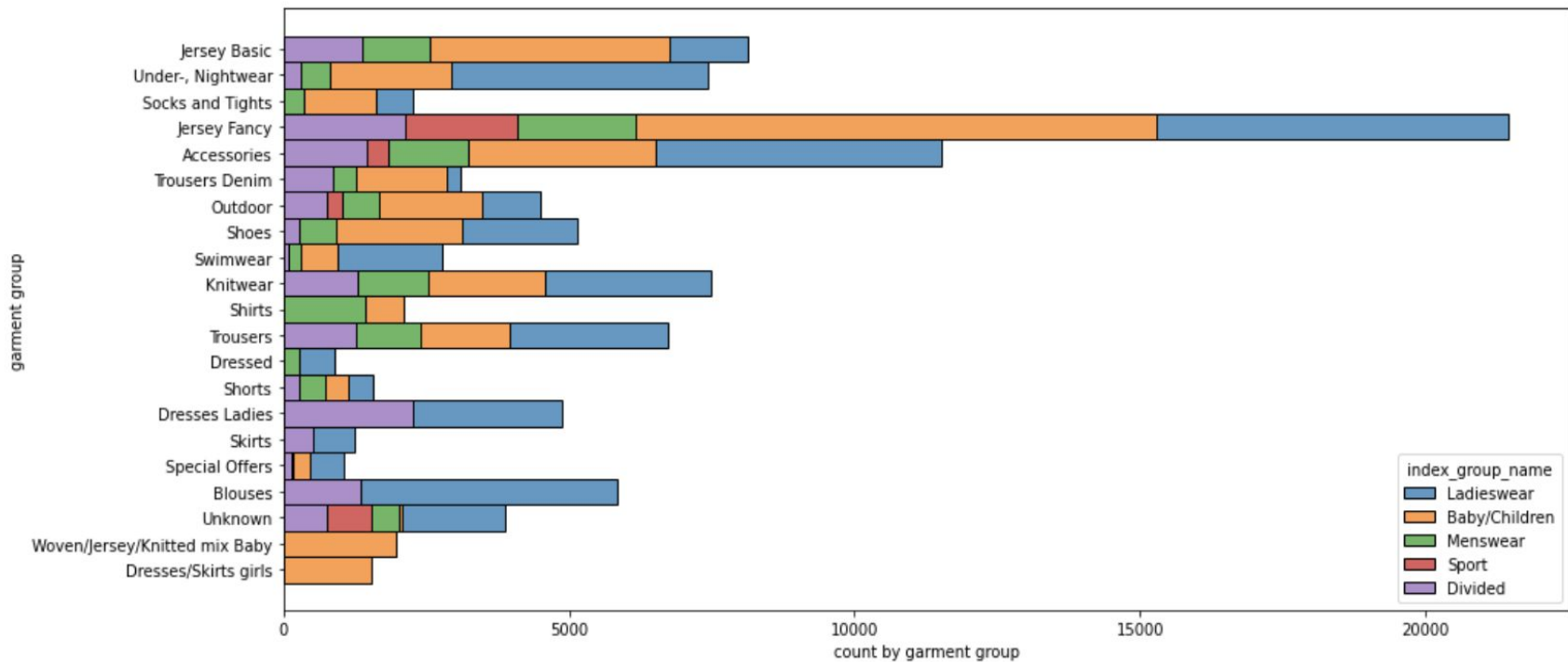
- History of past transactions: items bought by customers (5 columns, including customer ID, article ID and date)
- Dataset was too large so we used subset of 2000 items for development

EXAMPLE OF EXPLORATORY ANALYSIS



- Sub-category “Trousers” prevails
- “Dress”, “T-shirt” & “Sweater” category follows
- “Accessories” composition is various: “Bag”, “Hat”, “Earrings” are the most represented

PRODUCT GROUP-PRODUCT STRUCTURE



CONTENT-BASED NLP RECOMMENDER

-> Suggest articles that are similar to an item that is currently in the basket based on article description

```
[8]: # We will work here with the detailed descriptions available of articles  
articles['detail_desc'].head()
```

```
[8]: 0      Jersey top with narrow shoulder straps.  
     1      Jersey top with narrow shoulder straps.  
     2      Jersey top with narrow shoulder straps.  
     3  Microfibre T-shirt bra with underwired, moule...  
     4  Microfibre T-shirt bra with underwired, moule...  
     Name: detail_desc, dtype: object
```

TF-IDF VECTORS FOR FINDING SIMILARITY BETWEEN DESCRIPTIONS

Goal: extract features from the text data in order to compute the similarity between descriptions; transform natural language into vectors

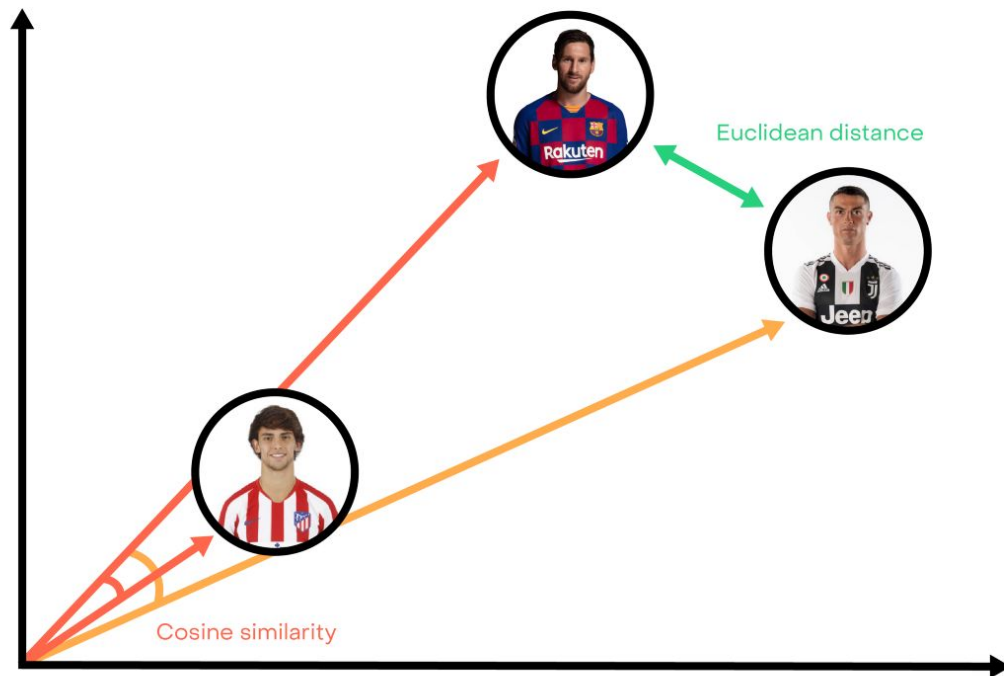
Count Vectorizer

	blue	bright	sky	sun
Doc1	1	0	1	0
Doc2	0	1	0	1

TD-IDF Vectorizer

	blue	bright	sky	sun
Doc1	0.707107	0.000000	0.707107	0.000000
Doc2	0.000000	0.707107	0.000000	0.707107

FIND SIMILAR VECTORS



BUILD A RECOMMENDER FUNCTION

1:

```
# Function that takes in an article ID as input and outputs most similar articles
def get_recommendations(article_id, cosine_sim=cosine_sim):
    # Get the index of the article that matches the ID
    idx = indices[article_id]

    # Get the pairwise similarity scores of all articles with that article
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort the articles based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the scores of the 10 most similar articles
    sim_scores = sim_scores[11:21]

    # Get the articles indices
    article_indices = [i[0] for i in sim_scores]

    # Return the top 10 most similar articles
    return articles['article_id'].iloc[article_indices]
```

CHOOSE A RANDOM ARTICLE AND RETRIEVE SIMILAR ARTICLES

Item in basket:



Top 3 recommendations:



PREPARING TRANSACTION DATA FRAME FOR DEEP LEARNING

```
[39]: # Creating a sparse pivot table with customers in rows and items in columns
customer_items_matrix_df = df.pivot(index = 'customer_id',
                                     columns = 'article_id',
                                     values = 'bought').fillna(0)

customer_items_matrix_df.head(10)
```

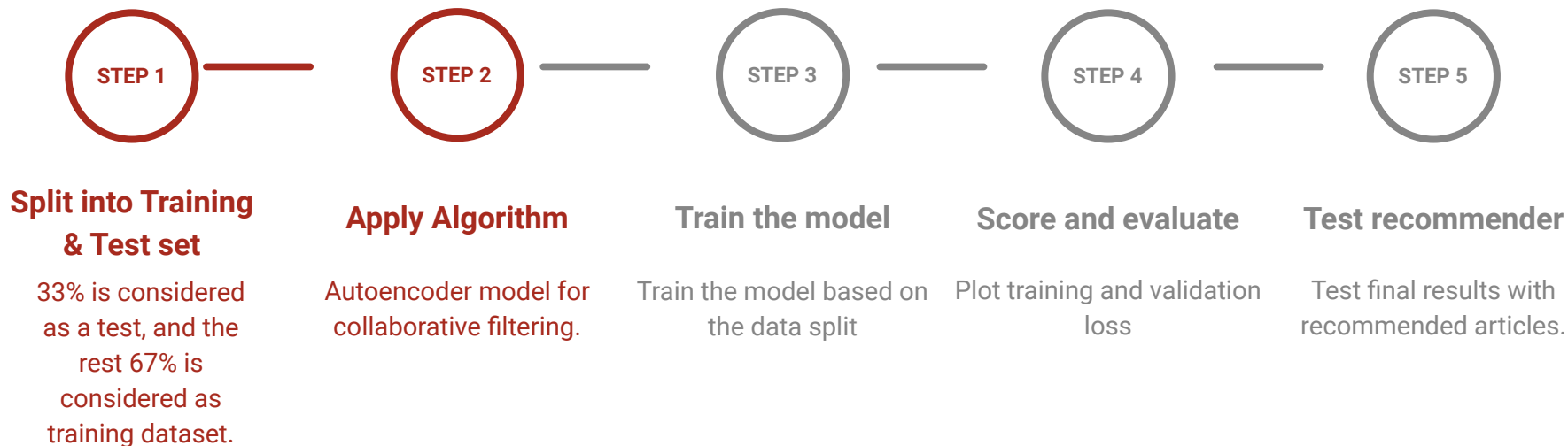
ut[39]:

article_id	108775015	110065001	111593001	153115039	156231001
customer_id					
000058a12d5b43e67d225668fa1f8d618c13dc232df0cad8ffe7ad4a1091e318	0.0	0.0	0.0	0.0	0.0
00007d2de826758b65a93dd24ce629ed66842531df6699338c5570910a014cc2	0.0	0.0	0.0	0.0	0.0
00083cda041544b2fbb0e0d2905ad17da7cf1007526fb4c73235dccbbc132280	0.0	0.0	0.0	0.0	0.0
0008968c0d451dbc5a9968da03196fe20051965edde7413775c4eb3be9abe9c2	0.0	0.0	0.0	0.0	0.0
000aa7f0dc06cd7174389e76c9e132a67860c5f65f970699dacc14425ac31a8	0.0	0.0	0.0	0.0	0.0
001127bffdda108579e6cb16080440e89bf1250a776c6e55f56e35e9ee029a8d	0.0	0.0	0.0	0.0	0.0
001ea4e9c54f7e9c88811260d954edc059d596147e1cf8adc73323aebf571fd8	0.0	0.0	0.0	0.0	0.0
001fd23db1109a94bba1319bb73df0b479059027c182da490e1161b34cd3af61	0.0	0.0	0.0	0.0	0.0
0021da829b89f82269fc51feded4eac2129058ee95bd75bb1591e2eb14ecc79	0.0	0.0	0.0	0.0	0.0
00228762ecff5b8d1ea6a2e52b96dafa198febddbc3bf350eb611f28e61ea6ce	0.0	0.0	0.0	0.0	0.0

DEEP LEARNING RECOMMENDER

- Collaborative filtering technique: suggest articles that a customer might want to buy in the future, based on purchases of customers who have similar profiles.

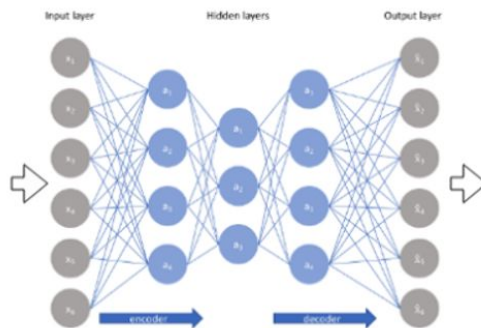
Steps:



HOW DOES THE AUTOENCODER COME UP WITH RECOMMENDATIONS?

- The aim of an Autoencoder is to compress representations and preserve essential information for reconstructing input data. So, we compress an Interaction Matrix which is sparse and reconstruct a complete version of the matrix containing an approximation of all values that were null before. See below:

	Item A	Item B	Item C	Item D	Item E
Customer 1		1	1		
Customer 2	1	1		1	
Customer 3		1		1	
Customer 4		1	1	1	1
Customer 5	1	1		1	
Customer 6			1		1



	Item A	Item B	Item C	Item D	Item E
Customer 1	0.86	1	1	0.65	0.12
Customer 2	1	1	0.65	1	0.1
Customer 3	0.1	1	0.12	1	0.25
Customer 4	0.23	1	1	1	1
Customer 5	1	1	0.65	1	0.65
Customer 6	0.95	0.15	1	0.65	1

TRAINING THE MODEL

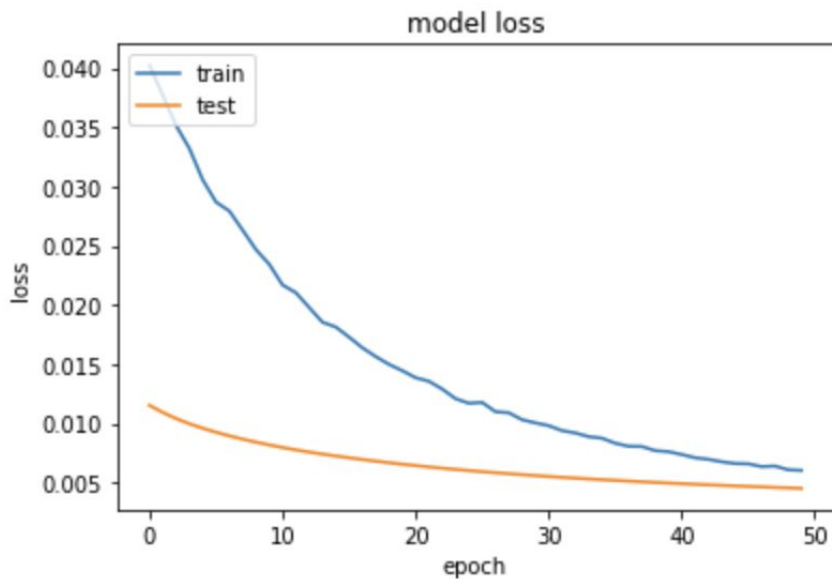
- Then we got 50 epochs and could observe:
 - `val_loss` - 0.0045
 - `loss` - 0.0060
- It confirms that validation loss is consistently lower than training loss (gap shrinks overtime)

```
6/6 [=====] - 0s 24ms/step - loss: 0.0070 - val_loss: 0.0048
Epoch 44/50
6/6 [=====] - 0s 24ms/step - loss: 0.0069 - val_loss: 0.0048
Epoch 45/50
6/6 [=====] - 0s 24ms/step - loss: 0.0067 - val_loss: 0.0047
Epoch 46/50
6/6 [=====] - 0s 24ms/step - loss: 0.0066 - val_loss: 0.0047
Epoch 47/50
6/6 [=====] - 0s 24ms/step - loss: 0.0064 - val_loss: 0.0047
Epoch 48/50
6/6 [=====] - 0s 24ms/step - loss: 0.0064 - val_loss: 0.0046
Epoch 49/50
6/6 [=====] - 0s 27ms/step - loss: 0.0062 - val_loss: 0.0046
Epoch 50/50
6/6 [=====] - 0s 23ms/step - loss: 0.0060 - val_loss: 0.0045
```

After 50 epochs we get a 0.0060 star deviation between the predicted and actual purchases on the test set...

MODEL LOSS GRAPH

- Plot of learning curves is a good fit when training loss decreases to a point of stability.



RECOMMENDER

- **Articles previously purchased** by the customer.

	score	prod_name
article_id		
635957003	1.0	CSP Coronado lace sweater
680912011	1.0	Linni tee
316657008	1.0	Karenina Jacket
555364003	1.0	Spencer Shirt Sweater
679059001	1.0	GABBY KNITTED TUBE
684238002	1.0	Savannah shopper (1)
662628001	1.0	MC Eleven top
640556001	1.0	Bali top BIG
571650001	1.0	Taylor Fancy Denim

- **Recommended articles** for a customer sorted by best scored.

	score	prod_name
article_id		
670063001	0.255884	Flirty Disa ring pk
604100002	0.243205	Nilla Top S/S
606712001	0.242506	Valerie sandalette
643588001	0.227597	Polly shopper
564313003	0.227204	Robin 3pk R basic short trunk
427159007	0.224022	Skinny 5pkt Trash
578816004	0.214845	OL RIRI sneaker
539723001	0.192965	Jade Denim TRS
179208001	0.192820	Control Top 100 den 1p Tights
564312012	0.191794	Chris 3pk Fancy

TEST RECOMMENDER ON UNSEEN DATA

Example: The customer bought those items

Article 657497007



Article 625773001



Article 456163026



Article 674336001



Article 676954001



Example: Next products to be recommended (sorted by best scored)

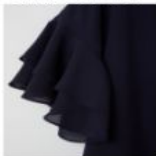
Article 399136061



Article 525500001



Article 617900007



Article 585788001



Article 678073002



Article 613497008



Article 573652001



Article 662741001



Article 610776010



Article 562245042



THANK YOU FOR YOUR ATTENTION!