E-commerce recommender system

Report

The problem

What products to recommend to which customers as e-commerce retailer?

- how to ensure recommendations are ‘relevant’ to new customers not having history?
- and similarly, how to increase the likelihood of products with limited reviews/purchase to be seen and bought?

The Approach

Develop a recommender system that learns from user/items interactions as well as from items features to recommend products users are most likely to buy

Benefits

The recommender system will increase sales through:

- Increased basket size (value per purchase)
- Increased sales for new products
- Higher conversion rate for new customers
- Overall better experience leading to higher customers turnover

The Data

Dataset source: Kaggle (https://www.kaggle.com/arashnic/marketing-bias-dataset)

Description: the dataset contains attributes about products sold on ModCloth Amazon.
ModCloth is an indian clothing brand selling online through Amazon

Data columns includes item/user interactions (rating) and some useful data regarding products description
Data size: 99893 entries and 7 columns
Unique values: 1020 items and 44783 users

Columns:
item_id
user_id
Rating
size
fit
user_attr
model_attr
category
year
split

Data Wrangling

Data cleaning and tidy:

- user_id: There is 1 missing user_id value and the entire row has been deleted
- Timestamp: the entire column has been dropped as it won't be useful for any of our analysis
- size: There are about 22% missing values. We applied the most frequent value imputer strategy to populate these missing values
- Fit: There are about 18% of missing values. We will apply the same strategy as size for missing values.
- User_attr. there are 8% of missing values. We will apply the same strategy as size for missing values.
- Brand: There are about 74% of missing values which means this column will be useless. we will simply drop it.

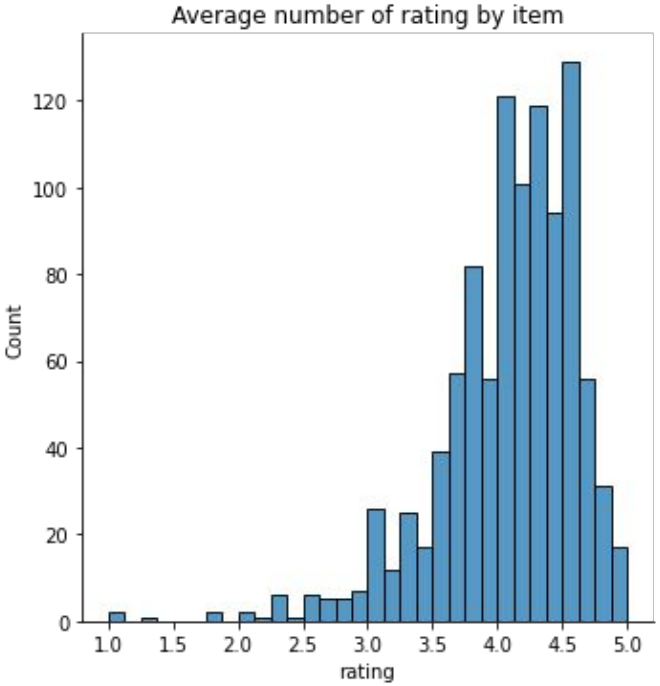
	missing	cat	unique
item_id	0.000000	int64	1020
user_id	0.000010	object	44783
rating	0.000000	int64	5
timestamp	0.000000	object	14741
size	0.217833	float64	9
fit	0.185258	object	5
user_attr	0.083760	object	2
model_attr	0.000000	object	2
category	0.000000	object	4
brand	0.740592	object	31
year	0.000000	int64	10
split	0.000000	int64	3

Key Findings

- There are far more customers than products with a ratio of 43 purchases per customers
- There is considerable number of products with low ratings which is the case for the 'cold start' problem
- We have information regarding items features but not users features
- There are few 'very popular' items with high numbers of rating while others have much lower number of rating
- The data available can be used for different modelling scenario using both collaborative filtering and content base:
- Collaborative filtering: here we focus on the interactions user/item/ratings to predict products
- Contents base: we can use the items features to provide recommendations to new users or for new products

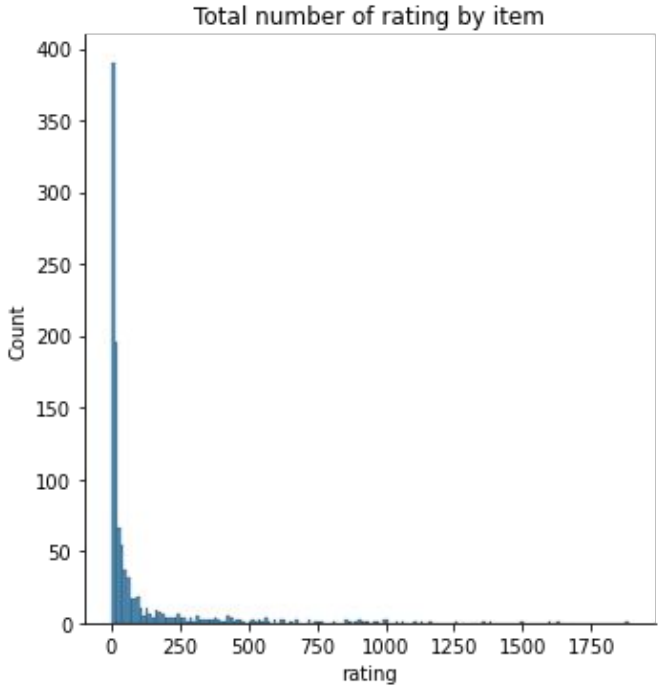
Items can be segmented into 2 categories:

- 'Popular' items’ with high frequency of rating
- ‘Less popular’ items with few ratings



Distribution of average number of rating per item is right skewed which confirms that there are some 'very popular' items with high frequency of rating while others have much lower number of rating

Sparse user/items interaction matrix



Most items have few number of rating which menas that the item/user matrix is very sparse

Modelling scenario

01 Collaborative Filtering using KNN and Matrix Factorization

02 Collaborative Filtering using scikit surprise library

03 Content Based Recommender

04 Hybrid Recommender

Model selection for collaborative filtering

Modelling using Scikit Surprise library

Algorithm	test_rmse	fit_time	test_time
SVD	1.056060	6.689027	0.222243
KNNBasic	1.080752	191.928903	11.370964
NormalPredictor	1.393228	0.234222	0.372735

Our final model outperformed matrix factorization with a RMSE of 1.0422 (vs 15 for the matrix factorization)

We used GridSearch to identify the best parameters below for our best SVD algorithm: {'n_factors': 10, 'n_epochs': 10, 'lr_all': 0.005}

Model selection for content base

Hybrid model using LightFM library

Modelling scenario
LightFM was used to create a model including user/item interactions as well as items features
Loss function used is ‘warp’
Number of epochs: 10

Evaluation
Train auc: 98%
Test auc: 85%
Recall_at_100: 61%
precision_at_100:0%

Recommendations

- Implement the surprise **SVD model** as go to algorithm leveraging the wealth of user/item interaction to recommend items to returning users
- Implement the **LightFM hybrid** model in parallel for items having low number of ratings as well as new users until enough data are gathered to apply the SVD model
- Finally, leverage the **content based** approach strategically for brand new products during launch phase for instance

Further Analysis

- User features: We missed the opportunity to integrate user features in both content based and hybrid models. Information such as prior pages visited, viewed items or comments can be very useful to personalize recommendations to users.
- User profile: Initial collection of information from users can also help to instantly enrich recommendations from content based approaches. This includes information such as age, gender, height or weight.
- Real time implementation: Our implementation assumes batch feeding into the model. However, we can leverage the LightFM hybrid model to feed data partially and update it in real time. This would increase the pace of learning for our model as it could quickly adapt from user behavior

Thank you!