E-commerce Recommender System

Capstone 3

Introduction

Since the advent of Covid-19, e-commerce sales has skyrocketed. For many consumer goods companies and retail, online sales has been the only alternative and the expectation is that the new normal still continues to shape habits around online sales.
Unlike physical stores, online shops have limited 'space' to display products to customers over a large range of products. In addition, users are more impatient and can easily switch to a competitor.
This pose a major challenge to e-commerce retailers having wide range of products:
How can we ensure we always display products that customers are most likely to purchase whether they are new or returning customers?

Business opportunity

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

Dataset

- 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.

The outcome of this phase is 2 datasets to pursuit different modelling approaches:

- Rating : having item_id, user_id and rating
- Features: having all items features

Both datasets have been saved under data/ processing subfolder.

EDA (Exploratory Data Analysis)

General purpose analysis

- Analysis of ratings score distribution
- Analysis of average number of ratings by items
- Analysis of total number of ratings by items
- Unique values for each series
- Calculation of average number of ratings by user

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

Pre-processing and Modelling

Modelling part 1: Collaborative Filtering using KNN and Matrix Factorization

Pre-processing:

- Given the cold start problem, we have will only considered users having at least 3 reviews (average is 43 reviews per user) in this first part of modelling
- Train and test split: partition was done with 25% train and 75% test

Modelling scenario

- NearestNeighbors with cosine as measure of similarity
- Matrix Factorization using SVD decomposition

Evaluation

Model	KNN	MF
Pros	Easy to implement	Better alternative to KNN by solving the dimensionality issue (converting the matrix into a dense object) Possible to evaluate (RMSE score of 15) which is not a great performance.

Cons	High dimensionality that may impact model performance Suffer from sparsity resulting from low number of users/items interactions (grayship problem) Is more inclined to recommend popular products which means negative bias for new products with products with few reviews Difficult to evaluate performance as no 'ground truth' is available	The cold start issue is not addressed

Modelling part 2: Collaborative Filtering using scikit surprise library

Pre-processing:

- We used the rating dataset as is
- Train/Test split: partition was done with 20% train and 75% test

Modelling scenario

Model selection

- We trained 3 models (SVD, KNNBasic and Normal Predictor) and used cross-validation based on RMSE performance to determine the best algorithm
- Based on this test, the SVD model was selected following its performance below:

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

Hyperparameters tuning

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

- GidSearch was also used to determine the best number of recommendations (k) for a threshold of 4 (average rating by item) for the best K
- In addition to RMSE we have also evaluated the model recall and precision performance
- Best k is 10 for a precision of 86% and recall of 85%

k	threshold	precision	recall
10	4	0.860676	0.850516
9	4	0.860743	0.850017
8	4	0.860867	0.849411
7	4	0.860968	0.848529
6	4	0.861127	0.847327
5	4	0.861391	0.845535

Evaluation

- Our final model outperformed matrix factorization with a RMSE of 1.0422 (vs 15 for the matrix factorization)
- Though we have a better and more robust model that addresses most of the weaknesses from part 1, we still need to solve the cold start issue.

Modelling part 3: Content Based Recommender

Pre-processing:

- We assume the most important item features are: size, model_attr and category
- A new string column was created with as value a concatenated string of the 3 features mentioned above

Modelling scenario

- TfidfVectorizer was used to extract features from text data
- Similarity was calculated using cosine_similarity

Evaluation

- The model is capable of making recommendations for any given item.

- This fully addresses the cold start problem but completely misses the strength of collaborative filtering as we saw in the modelling part 2

Modelling part 4: Hybrid Recommender

Pre-processing:

- User_id stored as string have been converted into integer using LabelEncoder
- Items features used are fit, model_attr and category
- No user features have been considered in this model as no data was available
- Train and test split: partition was done with 20% train and 80% test

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%
We were finally able to implement a model combining benefits of both collaborative filtering and content base
AUC score looks great, however, precision and recall are not as performing as our Surprise model

Final recommendations

Our final recommendation is threefold:

- 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

- Finally,

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 behaviour