Recommender System for Amazon Beauty Products

Providing the best shopping experience for customers is a top priority for eCommerce companies. To this end, constructing a sufficient recommender system is crucial. Amazon is one of the biggest eCommerce companies that sell a variety of products from beauty to home decoration to name a few.

In this project, we build a recommendation system for Amazon beauty products based on the product, user, and rating information.

Data

The dataset is one of Tensorflow's ready-to-use datasets that has the variables 'customer_id', 'helpful_votes','marketplace', 'product_category', 'product_id', 'product_part', 'product_title', 'review_body', 'review_date', 'review_headline', 'review_id', 'star_rating', 'total_votes', 'verified_purchase'

For our recommendation system, we use only the variables customer_id, product_id, and rating.

Method

There are three main types of recommenders used in practice today:

1. Content-based filter: Recommending future items to the user that have similar innate features with previously "liked" items. Basically, content-based relies on similarities between features of the items & needs good item profiles to function properly.
2. Collaborative-based filter: Recommending products based on a similar user that has already rated the product. Collaborative filtering relies on information from similar users, and it is important to have a large explicit user rating base (doesn't work well for new customer bases).
3. Hybrid Method: Leverages both content & collaborative based filtering. Typically, when a new user comes into the recommender, the content-based recommendation takes place. Then after interacting with the items a couple of times, the collaborative/user-based recommendation system will be utilized.

We use a user-based recommendation system that predicts the rating that a user would give to a certain product.

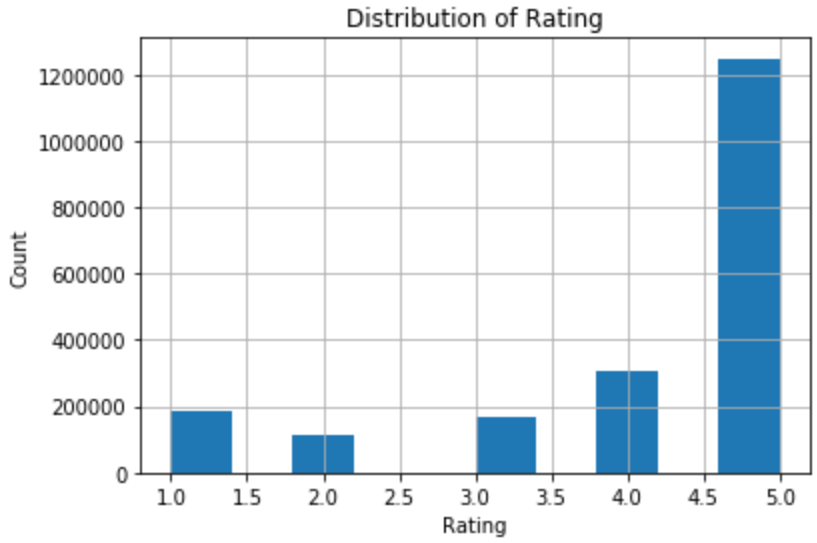
Data Cleaning

The model is based on three variables, namely: customer ID, product ID, and rating. The data does not require extensive cleaning as the data types and format are appropriate for the ML model.

EDA

1. The Distribution of Rating

The following graph displays the distribution of rating



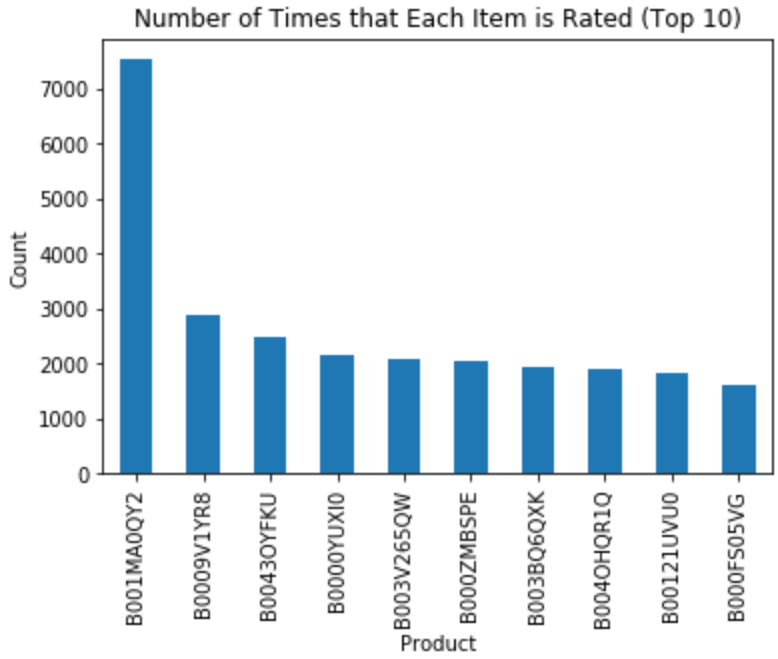
Most of the rating values are above 4. There are almost 200,000 out of 2M items that received a rating of 1. And, almost 300,000 items rated as 4. The majority of the rating values are above 4.5

Summary Statistics for Rating

Mean	4.15
Std. dev	1.31
Min	1
Max	5
Q1	4
Q3	5

2. Most rated products (Top 10)

We also want to know what products received the most ratings

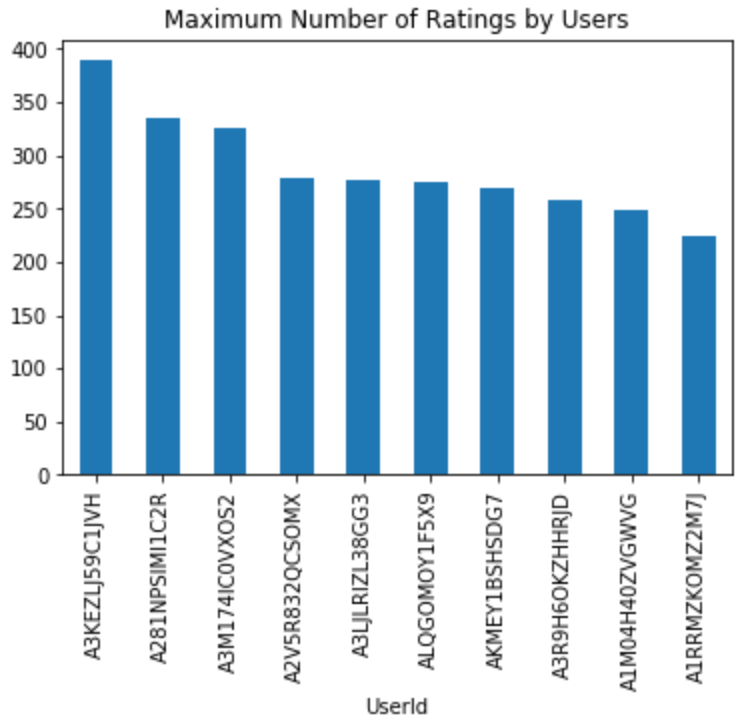


As we can see on the above graph that the product ‘B001MA0QY2’ (‘Ceramic Tourmaline Ionic Flat Iron Hair Straightener’) is rated more than 7000 times with an average rating of 4.32

Number of times that the product is rated and the average rating per product

	mean	count
ProductId		
B001MA0QY2	4.321386	7533
B0009V1YR8	3.568839	2869
B0043OYFKU	4.310456	2477
B0000YUXI0	4.405040	2143
B003V265QW	4.365421	2088
B000ZMBSPE	4.422342	2041
B003BQ6QXK	4.625652	1918
B004OHQR1Q	4.465782	1885
B00121UVU0	4.538085	1838
B000FS05VG	4.159849	1589

3. Average rating and the number of ratings by user



Above, we see the average rating and the number of times that a user rated. As we can see that user 'A3KEZLJ59C1JVH' rated items 389 times with an average rating of 3.7. There is a user who rated 275 times with an average rating of 2.22

Algorithms and Machine Learning

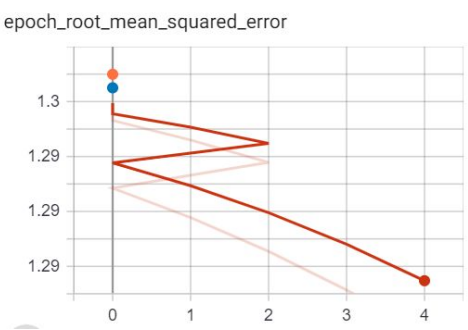
We used Tensorflow to build the recommendation model with the following steps.

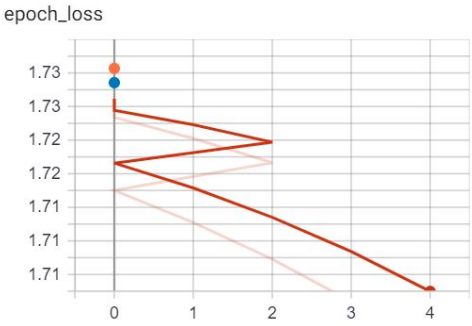
1. Load the data using TensorFlow's load function
2. Choose the basic variables (Customer Id, product ID, and rating
3. Split the data (80-20)
4. Use unique customers and products
5. Create the embeddings
6. Use the embeddings as inputs to make predictions
7. Cache the data before fitting

We use the MeanSquaredError and RMSE Keras loss in order to predict the ratings. Finally, we test the model on our test data.

Training Performance

The following graphs show the performance of the training stage in terms of loss and RMSE





As we can see, on the graphs above, as the model is trained the loss and RMSE decrease.

We evaluated the model on the test data and obtained the following results

root_mean_squared_error	1.296741
loss	1.672668
regularization_loss	0.000000
total_loss	1.672668

Comparison of loss and RMSE for training and test data.

	Train	Test
Loss	1.697	1.672
RMSE	1.289	1.297

Based on the table above, the performance of the model on the training and the test data is quite similar.

Future Improvements

The model above gives us a decent start towards building a ranking system. Of course, making a practical ranking system requires much more effort. In most cases, a ranking model can be substantially improved by using more features rather than just user and candidate identifiers.