Amazon products recommendation system

Kamil Madey

Introduction:

According to a report/study done by Mckinsey and Company in 2013, 35% of what people purchase on Amazon and 75% of what people watch on Netflix come from recommendations. Recommendation systems help users discover items they may not have known were previously there and that may be relevant to them. They are seen in many aspects of business, including one we will look at today, Amazon products.

Building recommendation systems has many approaches. There are collaborative based filtering systems, which are defined by an assumption that a user likes items similar to other items they have liked. There are also content based approaches, which follow the information of the item rather than the user. There are also hybrid approaches which combine aspects of each.

Data:

The dataset we use contains reviews for Amazon electronic products. The initial dataset has over 10 million reviews!

The dataset is found on Kaggle but was originally sourced from a website containing various datasets for recommendation system research. There was both a ratings dataset containing the user, item, rating, and timestamp data as well as a metadata file containing information on the product such as name, price, review, etc.

Problem statement and approach:

Some relevant questions a business might ask is how a recommendation system provides value to their business.

Several features of good recommendation systems that improve a business:

1. It can help a user find the right product

2. It can increase engagement from users for various products and/or a website
3. It helps provide the correct items to the correct users.

The approach taken (in terms of value provided to a business) in this project is driven by these ideas.

The approach was to get a look at the data, build out a simple model, then compare several other models and compare them using an evaluation metric and then see how they perform.

In the project, the initial model was built using PySpark while the rest of the models were compared using Pandas. This was simply done to gain experience with PySpark. The same data is used in both cases.

Wrangling and EDA:

Prior to visualizing some of the data, some wrangling needed to be done. This was all done in PySpark using PySpark SQL. Here the datatypes and Schema for each column was specified, formatting for columns was done, and columns that weren't needed were dropped. The metadata dataframe was also joined to the ratings data so the product names were also attached to the ratings.

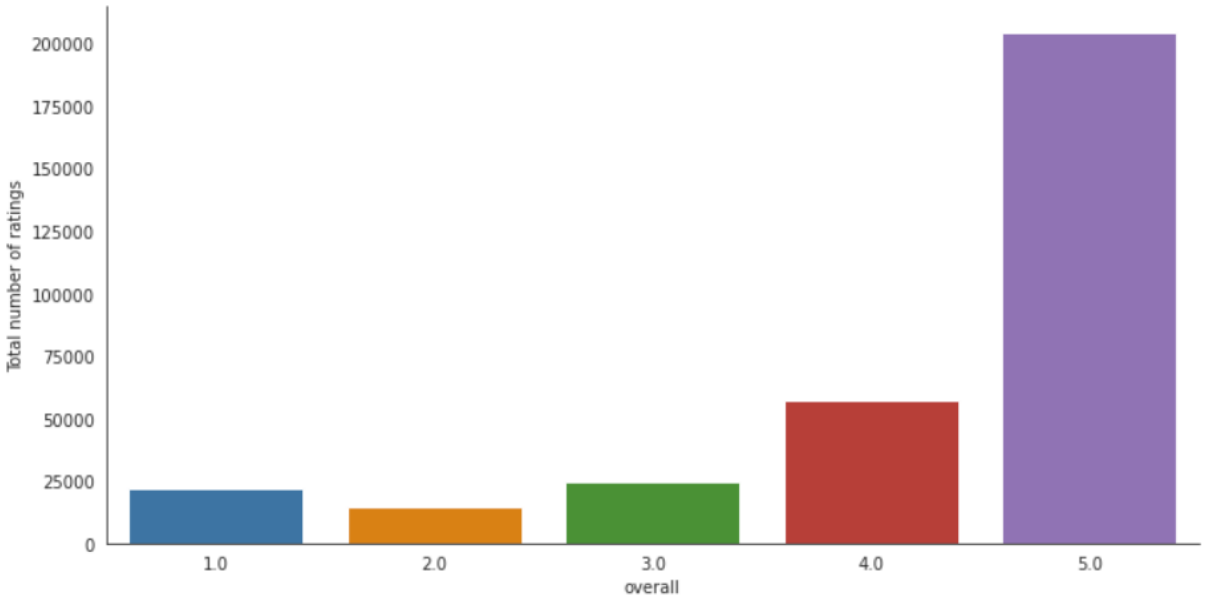
Looking further there is room to explore including some more of the features of the product into the model, especially if there is consideration to build an item based filtering system for a recommender.

This is how the dataframe looked after wrangling:

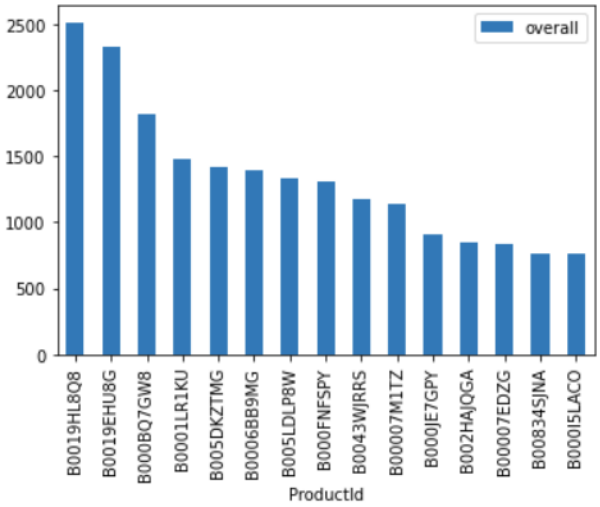
ProductId	overall	reviewerID	title
0446697192	4.0	A1A8QJ282YIUQ0	Hollywood Is like...
0446697192	4.0	AX9C10JD538D9	Hollywood Is like...
0446697192	3.0	A2UAM0ITC30078	Hollywood Is like...
0528881469	2.0	A2CPBQ5W4OGBX	Rand McNally 5288...
0528881469	5.0	A3H86FCI0QZH7T	Rand McNally 5288...

EDA was done in pandas since visualization is much easier through certain python libraries like matplotlib or seaborn than PySpark.

Using the dataframe we can get a look of how the ratings are broken down:



We also can see a breakdown of the most popular products:



With the number of ratings on the y axis and the product on the x axis.

Pre-processing/label encoding/dummy variables

One thing that was done was sparsity was calculated (in Pyspark) of the matrix. This is the matrix containing the users vs products and filled with the ratings. The sparsity calculated was 99%, meaning most of the data were user/product combinations that didn't have a rating.

Additionally, unique integer ids were given for products and users (this was essentially converting the users and products from categorical into numerical values.

Another thing that needed to be done before modeling was reducing the size of the matrix. At least for the purpose of this project. Having a sparse matrix with so many features would be too computationally taxing. The initial data had 20 million ratings!

We filtered the dataframe to products that had at least 50 ratings and users that had given at least 15 ratings. This greatly reduces the size of the data and also allows for a more accurate recommender system since each user/product has multiple ratings.

Modeling:

In our modeling section ,we first compare several different models to produce a test rmse along with computational time. We then take the top 3 from there and build those models using the data.

Algorithm	test_rmse	fit_time	test_time
SVD	0.897758	1.401541	0.073332
SlopeOne	0.911833	0.409658	0.084033
KNNWithMeans	0.922569	0.188611	0.188974
KNNBasic	0.927454	0.071114	0.102396
BaselineOnly	0.937336	0.037194	0.045982
CoClustering	1.020893	1.248085	0.072441
NMF	1.080985	1.518049	0.049352

We then end up comparing SVD, SlopeOne, and KNNBasic algorithms.

```
#SVD
svd = SVD(n_epochs=5, lr_all = 0.05, reg_all = 0.04, random_state= 42)

svdpredictions = svd.fit(trainset).test(testset)
accuracy.rmse(svdpredictions)
```

```
RMSE: 0.8432

0.8431915906914618
```

```
#SlopeOne
slope = SlopeOne()

slopepredictions = slope.fit(trainset).test(testset)
accuracy.rmse(slopepredictions)
```

```
RMSE: 0.8966

0.8966362456635468
```

```
#KNNBasic
knn = KNNBasic(n_epochs=5, lr_all = 0.05, reg_all = 0.04, random_state= 42)

knnpredictions = knn.fit(trainset).test(testset)
accuracy.rmse(knnpredictions)
```

```
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9142

0.9141795201785718
```

Winning model:

SVD ends up as our winning model.

With the SVD model, predictions are based on the equation:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

If a user (u) is unknown, bias (bu) and factors (pu) are assumed to be zero. To make the estimations for the unknown, we minimize the regularized squared error though this equation.

$$\sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2 \right)$$

Minimization is done using stochastic gradient descent, a common ML approach to minimizing a cost function. We can tweak the hyperparameters such as learning rate, regularization, n_epochs of this process.

This famous algorithm was used by Simon Funk in the Netflix Challenge.

Conclusion:

We evaluate the model on our test set and record our best 10 recommendations and worst 10. We see that a common theme among the top recommendations is that they all have a much larger number of ratings.

Best predictions

```
best_predictions = result.sort_values(by='error')[:10]
best_predictions
```

	user_int_id	prod_int_id	rating	predicted_rating	items_rated_by_user	num_ratings_for_item	error	title
2704	1321	1455	5.0	5.0	6	5	0.0	Nikon AF-S Nikkor 50mm f/1.8G Lens
2438	882	1272	5.0	5.0	7	21	0.0	NETGEAR 8-Port Fast Ethernet Unmanaged Switch,...
2466	383	125	5.0	5.0	7	3	0.0	Polk Audio CS10 Center Channel Speaker (Single...
2492	277	1009	5.0	5.0	9	11	0.0	Belkin (F3U133b10) Hi-Speed USB A/B Cable, USB...
2496	277	1009	5.0	5.0	9	11	0.0	Belkin (F3U133b10) Hi-Speed USB A/B Cable, USB...
2499	146	800	5.0	5.0	10	9	0.0	Canon EF 17-40mm f/4L USM Ultra Wide Angle Zoo...
2503	779	800	5.0	5.0	17	9	0.0	Canon EF 17-40mm f/4L USM Ultra Wide Angle Zoo...
2519	1135	531	5.0	5.0	5	18	0.0	C2G 29803 16 AWG 1-to-4 Power Cord Splitter - ...
2522	1136	531	5.0	5.0	9	18	0.0	C2G 29803 16 AWG 1-to-4 Power Cord Splitter - ...
2526	415	3121	5.0	5.0	9	7	0.0	Uxcell DLM-B002B8WVVU Aleratec Female to Femal...

Worst predictions

```
worst_predictions = result.sort_values(by='error')[-10:]
worst_predictions
```

	user_int_id	prod_int_id	rating	predicted_rating	items_rated_by_user	num_ratings_for_item	error	title
3974	1462	2240	1.0	4.771132	6	4	3.771132	Dual-shoulder Camera Neck Strap for Canon Niko...
3779	1434	1668	1.0	4.802158	5	0	3.802158	Zeikos ZE-LCH1 Lens Cap Holder Keeper String S...
3178	13	696	1.0	4.828623	9	0	3.828623	Uxcell 2-Way RJ11 US Telephone Plug to RJ11 So...
1079	239	1883	1.0	4.850841	11	2	3.850841	6ft ProMaxi Premium Gold Series Elegant Dual T...
1649	1030	470	1.0	4.862131	11	0	3.862131	AVerMedia EZMaker 7, Standard Definition USB V...
2889	658	3098	1.0	4.949009	4	2	3.949009	Griffin Elan Form Hard-Shell Leather Case for ...
3811	994	1689	1.0	4.960276	10	2	3.960276	TP-Link TL-WA901ND Wireless N450 3TER Access P...
309	179	65	1.0	4.965269	17	18	3.965269	TP-Link 5 Port Fast Ethernet Switch | Desktop ...
1211	1064	1429	1.0	5.000000	13	2	4.000000	Sony CFDS05 CD Radio Cassette Recorder Boombox...
2716	13	2544	1.0	5.000000	9	14	4.000000	Logitech HD Pro Webcam C920, Widescreen Video ...