Recommendation System with Spark

Problem:

How to recommend users/items for each items/users using past items' ratings data from different users?

Data:

reviewerID - ID of the reviewer, e.g. A2SUAM1J3GNN3B

asin - ID of the product, e.g. 0000013714

reviewerName - name of the reviewer

vote - helpful votes of the review

style - a disctionary of the product metadata, e.g., "Format" is Hardcover"

reviewText - text of the review

overall - rating of the product

summary - summary of the review

unixReviewTime - time of the review (unix time)

reviewTime - time of the review (raw)

image - images that users post after they have received the product

title - name of the product

feature - bullet-point format features of the product

description - description of the product

price - price in US dollars (at time of crawl)

related - related products (also bought, also viewed, bought together, buy after viewing)

salesRank - sales rank information

brand - brand name

categories - list of categories the product belongs to

tech1 - the first technical detail table of the product

tech2 - the second technical detail table of the product

similar - similar product table

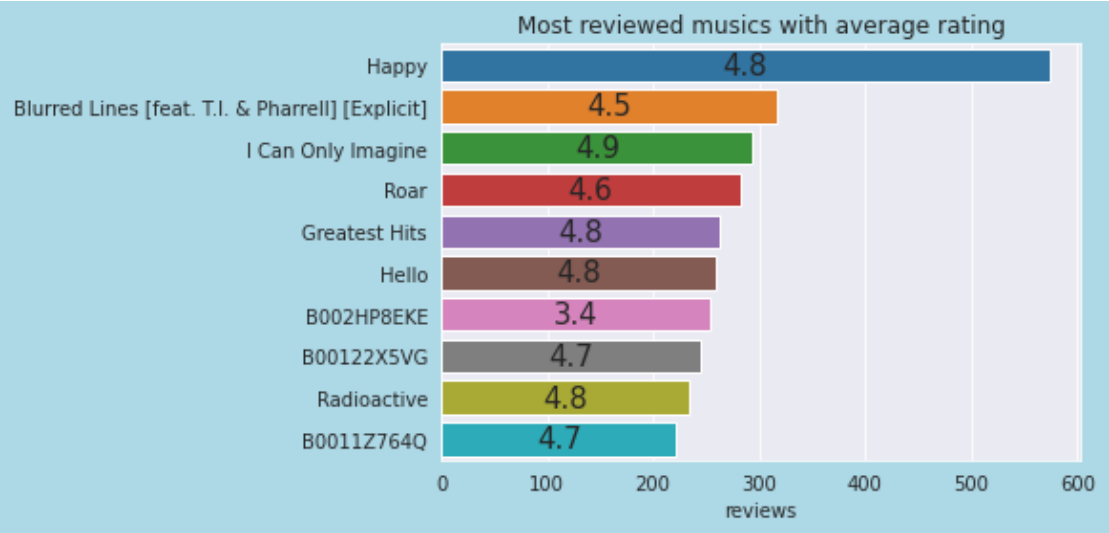
Approach:

A recommender system is built with PySpark using ALS algorithm.

Preprocessing:

- Some columns contain too much missing values, so they are not used.
- Titles of the music are in wrong format, which is corrected with the help of regular expression.
- Rename and create some new columns like “rating”, “use”(ids), and “item”(ids).

EDA:



This graph shows that Happy, Blurred Lines, and I Can Only Imagine are the top 3 most reviews music. They also have high ratings. Happy has by far the most reviews. Blurred Lines seems to have the lowest rating than the others top 3 music, but it’s still a high rating.

This graph shows that mp3, and audio CD music styles are by far the most review music. These music styles also both have high ratings.

Model:

The recommender system is built with ALS algorithm using Spark with Python. A grid search method using CV score of RMSE is used for hyperparameter tuning. The model has an MAE score of about 0.27 on the testing data. MAE is used for easier explanation to non-technical people.

Tuning parameters:

- Rank – 5, 10, 15
- Max iterations – 18, 19, 20
- Regularization parameter – 0.17, 0.18, 0.19

Model parameters:

- blockSize – 4096
- coldStartStrategy – drop
- rank – 15

Recommendation:

The model can be used to give recommendations of users/items to different items/users with some ratings data.

Further improvements:

- The model could be further improved with more complex algorithms to handles issues such as new users/items, and items/users who's ratings are very different from past users/items.
- The model only uses collaborative filtering. Perhaps if more data about the music like genre, era, etc. could be obtained, content base filtering could be combined with collaborative filtering to provide better recommendations.