International Rock Climbing Recommendation System

An Exercise in User-Based Collaborative Filtering

Capstone 1 Project
Kristen N. Colley

The sport of rock climbing has been steadily increasing in popularity. From 2012-2017, the IBISWorld estimates that from average annual growth for the indoor climbing wall industry was 3.9% in the USA. In 2015, it ranked 17th out of 111 out of the most popular sports in the United States. (Physical Activity Council and PHIT America).

Yet, even with this growth in popularity, most of the international rock climbing websites still lack a rock climbing recommendation system. In this project, I will create a recommendation system for the 8a.nu website that will help climbers identify some unique international climbing objectives.

Audience

Any of the climbing apps, websites, or gear websites would reap the rewards of being able to provide a rock route prediction system for their users.

The climbing community is still a small subset of the population; However, they are a passionate group who invest a lot of their time climbing and researching future climbs. This recommendation system would be a good addition to the community to give climbers a quick reference of what they would like to climb next.

Data Source

For the data, I choose a Kaggle data set scraped from the website: 8a.nu, one the world's largest database of rock routes with particular attention to the international climbing community. With over 4 million entries of climbs and ratings, this is a sufficient size to develop a good predictor model. To view the original Kaggle data set, click the link below:

Kaggle Dataset

In order to upload this large dataset, I utilized the Kaggle API in a google colab notebook in order to combat the RAM issue of my personal computer. To view my report on converting the data from an SQLite Kaggle dataset to a pandas dataframe click below:

Data Import Report

About the Data

All of this data is online user-entered and prone to discrepancies which made it a particularly challenging dataset to clean.

There were 4 different tables in this dataset. Below are general descriptions of the four tables:

- Ascent Table
    - 4 million user entered rock climbing entries
- User Table
    - 67,000 user profiles of climbers that log their climbs
- Grade Table
    - A reference table that translates the rock climbing grade to French or American rock ratings
- Methods Table

- Shows a reference of the way in which the climber finished the climb (i.e. did they fall, take, or climb is cleanly)
- I was able to eliminate all together because this was completely irrelevant to the recommendation system.

In order to properly clean these tables, I needed to delete a lot of extraneous information that would not serve the recommendation system. Essentially, I only needed three columns (user_id, item_id, item rating) to feed to the algorithms in the machine learning section. However, I needed to clean several more to serve as a “reference table” so my recommendation system can be filtered based on area and type of climb.

A	B	C
USER_ID	ITEM_ID	RATING
1	100	***
2	101	**
3	102	*

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.

I choose to work with a user-based collaborative filtering system. This made the most sense because half of the 4 million user-entered climbs had an explicit rating of how many stars the user would rate the climb. Unfortunately, the data did not have very detailed "item features". Every rock climbing route had an area, a difficulty grade, and a style of climbing (roped or none). This would not have been enough data to provide an accurate content-based recommendation. In the future, I would love to experiment using a hybrid system to help solve the problem of the cold-start-threshold.

Data Wrangling

Data Cleaning Report

Below is an overview of the main issues I ran into while cleaning the data:

- Problem 1: This dataset is all user-entered information. There are a couple drop down options, but for the most part the user is able to completely make-up, or list something incorrectly.
    - Solution part 1: normalize the data

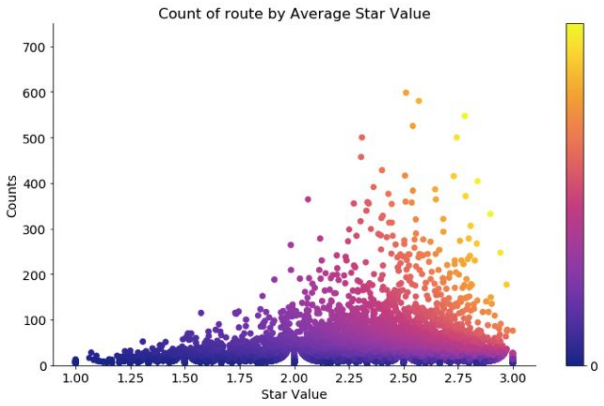
        - Make all characters lowercase
        - Exchange “&” sign for and
        - Get rid of all spaces in the names, and special characters (, . - ! ‘ ?)
        - Take away all accent marks (because there are lots of foreign language names)
        - Create a REGEX expression that only accepts letters and numbers
        - Filter out phrases such as (“I dont know”, “noname”, “none”)
    - Solution part 2:
        - after normalizing & cleaning all the columns, I created a three-tier groupby system that I could then take the mode of each entry and fill in the column with that mode. For example: a route listed 12 times had the country Greece associated with it 11 times, but one person incorrectly listed it located in the USA. By grouping together three other indicator columns and then computing the mode of the country, I was able to catch and change some of the user-entered errors and increase the accuracy of my dataset.
- Problem 2: Being this is an international rock climbing website, the names of the rock climbing routes were differing based on if the user enters accent marks or not.
    - Solution: normalize all names to the ascii standards.
- Problem 3: Spelling issues with the route name. For example: if there was a route named "red rocks canyon" it could be spelled "red rock", "red rocks", "red canyon" etc.

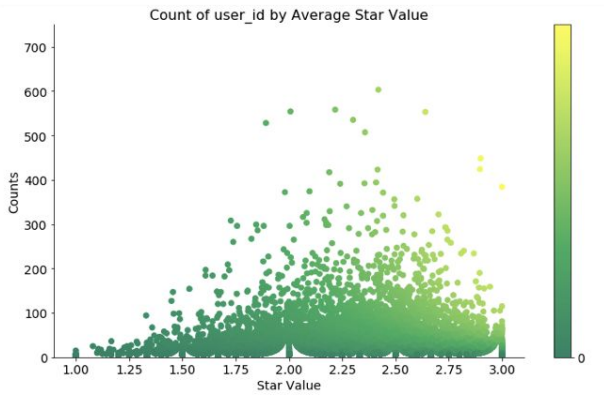
- Solution: at first, I was hopeful and tried two different phonetic spelling algorithms (soundex & double metahpone). However, both of these proved to be too aggressive in their grouping and sometimes would group together up to 20 different individual routes as the same item! My final solution was to create an accurate filter for route names. The logic being that if up to x number of users all entered that *exact same* route name, the chances were good that it was an actual route spelled correctly. I played around with 4 different filters and kept these until I could test their prediction accuracy in the ML portion. I found the greatest prediction accuracy came from the dataset that filtered out any routes listed less than 6 times.

Exploratory Data Analysis

In the EDA, I was able to identify that the dataset will be sufficient for the recommendation system.

Click here for the detailed EDA report of the main dataframe





Below are a couple pertinent findings:

- Star ratings:
    - The ratings of routes had an expected distribution of 1-3 stars. Where there were less one star ratings than two or three star (because people tend to negatively rate things less).
- Users providing ratings:
    - The most ratings a single user provided was 2,088 ratings
    - The majority of users (mode) only provided 1 rating
    - The average number of ratings per user is 50
    - 243 users gave more than 500 ratings
- Routes that were given ratings
    - The most ratings a single route received was 687 ratings
    - The lowest amount was only 1 rating for a route
    - The average number of ratings per route was 30
    - The mode was 10 ratings per route with a total of 3,622 routes with exactly ten ratings
    - Only 353 routes have more than 200 ratings (thus this was excluded from the graph)

Hypothesis Testing

Even though it does not pertain the recommendation system itself, I wanted to explore two intriguing questions about the climbing data:

1. *Do females climb the same grades as males now?*

    a. They have been pushing the envelope in climbing since the 90's, and even though this is traditionally a male dominated and developed sport in the past, I was curious if this has changed.
    b. Female versus Male Climbing Grades Hypothesis Test
    c. Results: According to this dataset, females are climbing on average two grades lower than males. The hypothesis test showed this difference is not due to chance.
2. *Do taller people climb higher grades than shorter people?*
    a. There is a rock climbing stereotype that if you are taller, you are naturally better at climbing. With the height data, I wanted to test this presumption.
    b. Tall vs. Short Climbing Grades Hypothesis Testing
    c. Results: According to this dataset, "short" people are climbing on average two grades higher than "tall" climbers and one grade better than "average" height climbers. The hypothesis test showed this difference is not due to chance. *Thus, according to this dataset, shorter climbers are better at climbing!*

Algorithms

ML Notebook

I chose to work with the Python surprise library scikit for training my recommendation system. I tested all four different filtered datasets on the 11 different algorithms provided, and every time the Single Value Decomposition++ (SVD++) algorithm performed the best. It should be noted that this algorithm,

although the most accurate is also the most computationally expensive, and that should be taken into account if this were to go into production.

Algorithm	test_rmse	fit_time	test_time
SVDpp	0.656549	16.115670	0.648576
BaselineOnly	0.660056	0.066147	0.135528
SVD	0.661268	2.013415	0.109702
KNNBaseline	0.697995	0.582456	0.640773
CoClustering	0.705715	1.005888	0.139992
KNNWithMeans	0.707114	0.610985	0.545816
KNNWithZScore	0.707959	0.649346	0.553792
NMF	0.730429	2.461001	0.122456
SlopeOne	0.733207	0.774625	0.367432
KNNBasic	0.743560	0.553092	0.482563
NormalPredictor	0.939817	0.043034	0.129717

NOTE: I adopted RMSE as the accuracy metric over mean absolute error(MAE) because the errors are squared before they are averaged which gives the RMSE a higher weight to large errors. Thus, the RMSE is useful when large errors are undesirable. The smaller the RMSE, the more accurate the prediction because the RMSE takes the square root of the residual errors of the line of best fit.

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j \right)$$

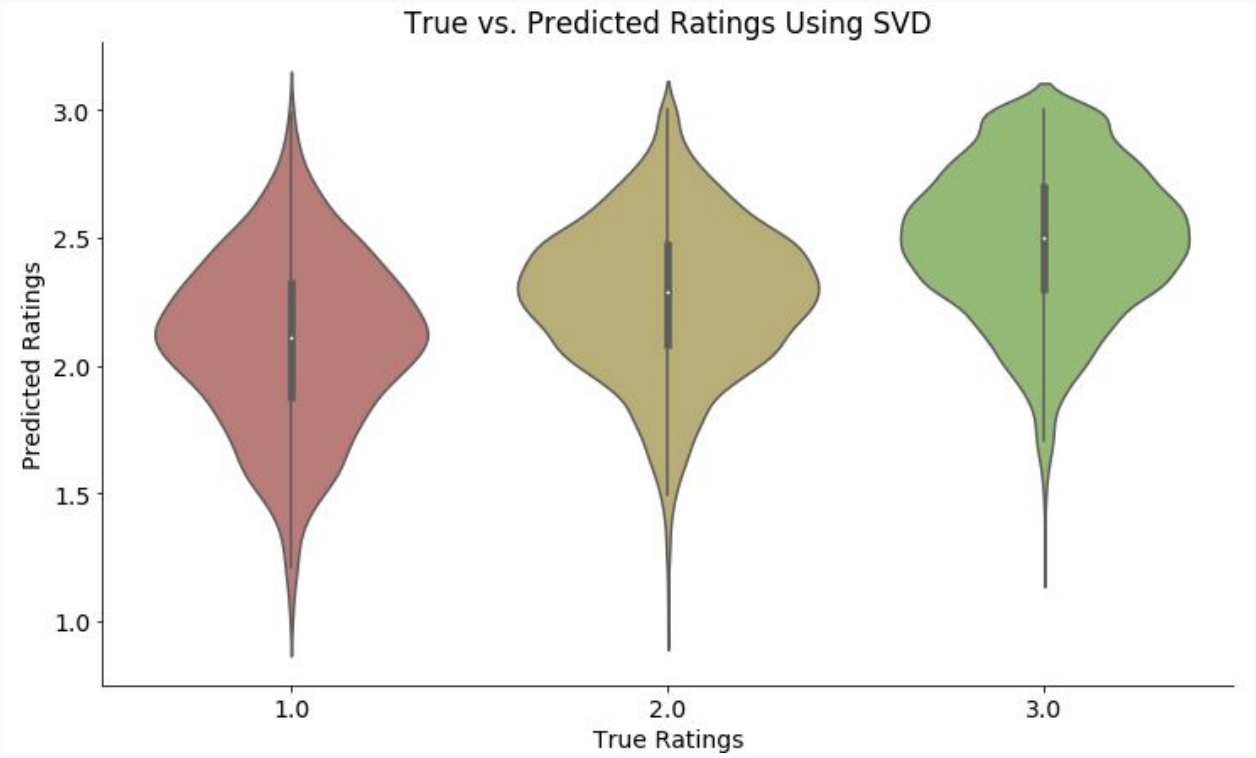
This algorithm is an improved version of the SVD algorithm that Simon Funk popularized in the million dollar Netflix competition that also takes into

account implicit ratings (yj). Using stochastic gradient descent (SGD), parameters are learned using the regularized squared error objective.

Choosing Filters

[ML Notebook]

After choosing the SVD++ algorithm, I tested the accuracy of all four different filtered datasets. The dataset which filtered out any route names occurring less than 6 times performed the most accurate predictions. Thus, it was chosen to be the dataset I trained on.



- All of the dataframes displayed discrepancies with the 1 star ratings(This is to be expected due to the inherent skewed positive ratings). Also, the

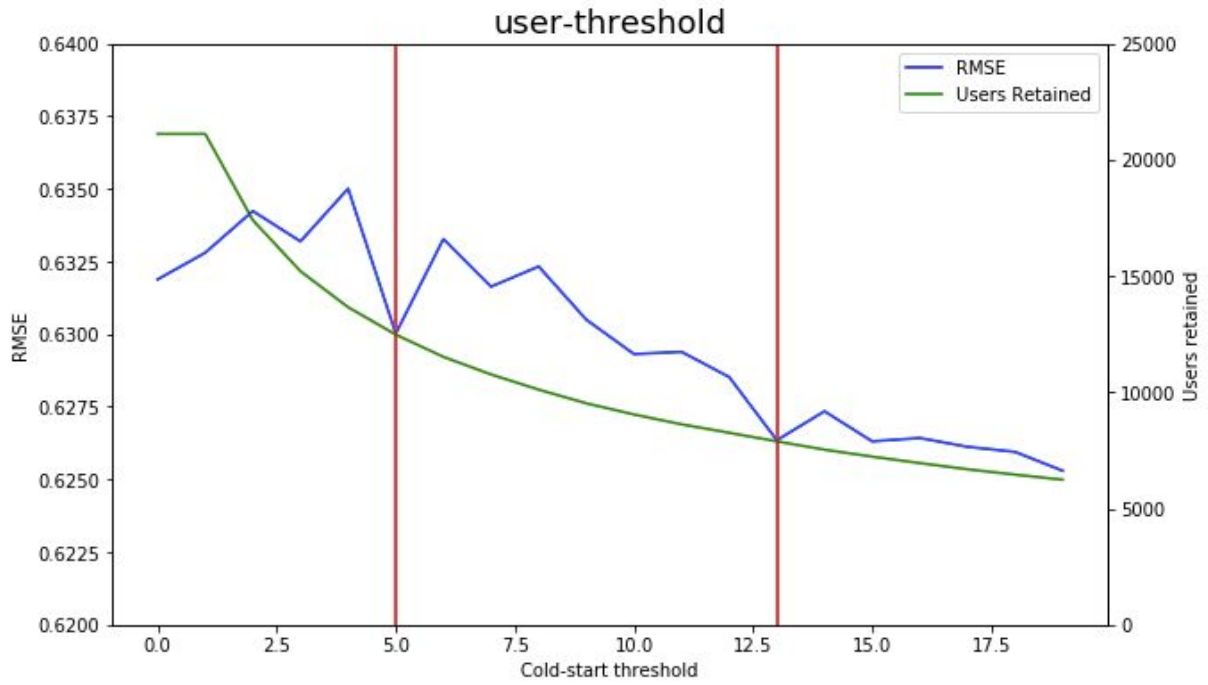
one star ratings are not imperative to this project's goal. It is more important that the 1 star ratings are different enough to be filtered out of the top ten routes recommended to users.

- Notice the 3-star rating has a fat bulge at the top of the "violin" which indicates its predicting 3-star ratings for some of the true 3-star routes. This was not as prominent in the other dataframes
- The 1-star rating also has a fatter tail than the other datasets displayed

Cold Start Threshold

ML Notebook

Coldstart Threshold: There is a problem when only using collaborative based filtering: *what to recommend to new users with very little or no prior data?* Remember, we already set our cold start threshold for the routes by choosing the dataset that filtered out any route occurring less than 6 times. Now, let investigate where to put the threshold for users.



It is my hypothesis that the initial filtering of the routes is what affected the RMSE of the users

- Increasing the user threshold to 5 would increase the RMSE by .005 & would lose approximately 40% of the data.
- Increasing the user threshold to 13 would increase the RMSE by .0075 & would lose approximately 60% of the data
- If there were a larger increase in the RMSE (>= .01) I would trade my users' data for this improvement. However, these improvements are too minuscule to give up 40%-60% of my data to train on. Instead, I voted to keep some of these outliers to help the model train, and will focus on fine tuning my parameters using gridsearch to improve the RMSE

Predictions

[Final Predictions Notebook]

In the final predictions notebook, the user can enter their user_id number and receive a list of top ten routes recommended to them:

```
Where in the world should you climb next?
Top Ten Rock Climbing Recommendations for User #12:
```

	route	user_rate_predict	avg_rating	num_users_rate	climb_type	usa_routes	usa_boulders	crag	country
1	Abstrakt	2.965450796247701	2.931	29	Rope Climb	5.13b	V11	Hylteberget	SWE
2	Hegar	2.809462278617374	2.778	27	Rope Climb	5.10d	V4/V5	Nissedal	NOR
3	Thaiboxing	2.767406704232296	2.811	37	Rope Climb	5.12b	V8	Jarlsberget	NOR
4	Vestpillaren	2.7656782513690805	2.919	86	Rope Climb	5.10c	V4	Lofoten	NOR
5	Quimera	2.745291868794412	2.900	20	Rope Climb	5.11a	V5	Pego	ESP
6	Trampoline	2.7450755717280315	2.000	1	Bouldering	5.12a	V7	Smuggs	USA
7	Magnetfinger	2.7443795704718594	2.939	49	Rope Climb	5.13a	V10	Pfalz	DEU
8	Villskudd	2.7339687745935484	2.820	128	Rope Climb	5.10a	V3	Bohuslan	SWE
9	Prismaster	2.731965630823839	2.649	114	Rope Climb	5.10a	V3	Bohuslan	SWE
10	Kachoong	2.724216995120893	2.850	40	Rope Climb	5.10d	V4/V5	Arapiles	AUS

Future Improvements

- In the future, I would love to spend more time creating a filtering system, wherein a climber could filter out the type, difficulty of climb, & country before receiving their top ten recommendation
- This recommendation system could also be improved by connecting to the 8a.nu website so that the user could input their actual online ID instead of just their user_id number
- Due to RAM constraints on google colab, I had to train a 65% sample of the original 6x dataset. Without resource limitations, I would love to train on the full dataset. Preliminary tests showed that the bigger the training size, the lower the RMSE. One test showed an increase in sample size could increase the RMSE by .03 (in contrast to the .005 improvement I received when increasing the coldstart threshold)