


# Quick Guide to Build a Recommendation Engine in Python

MACHINE LEARNING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/MACHINE-LEARNING/) PYTHON (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/PYTHON-2/)






## Introduction

## TOP ANALYTICS VIDHYA USERS

| Rank | Name |
|---|---|
| 1 |  vopani (https://datahack.analyticsvidh/user/profile/Rohan Rao) |
| 2 |  SRK (https://datahack.analyticsvidh/user/profile/SRK) |
| 3 |  Aayushmnit (https://datahack.analyticsvidh/user/profile/aayushmnit) |
| 4 |  binga (https://datahack.analyticsvidh/user/profile/binga) |
| 5 |  Nalin Pasricha (https://datahack.analyticsvidh/user/profile/Nalin) |

More Rankings (http://datahack.analyticsvidhya.com/users)

*This could help you in building your first project!*

Be it a fresher or an experienced professional in data science, doing voluntary projects always adds to one's candidature. My sole reason behind writing this article is to get your started with recommendation systems so that you can build one. If you struggle to get open data, write to me in comments.

Recommendation engines are nothing but an automated form of a "shop counter guy". You ask him for a product. Not only he shows that product, but also the related ones which you could buy. They are well trained in cross selling and up selling. So, does our recommendation engines.

The ability of these engines to recommend personalized content, based on past behavior is incredible. It brings customer delight and gives them a reason to keep returning to the website.

In this post, I will cover the fundamentals of creating a recommendation system using GraphLab (https://www.analyticsvidhya.com/blog/2015/12/started-graphlab-python/) in Python. We will get some intuition into how recommendation work and create basic popularity model and a collaborative filtering model.

## Topics Covered

1. Type of Recommendation Engines
2. The MovieLens DataSet
3. A simple popularity model
4. A Collaborative Filtering Model



## POPULAR POSTS

- A Complete Tutorial to Learn Data Science with Python from Scratch (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/)
- Essentials of Machine Learning Algorithms (with Python and R Codes) (https://www.analyticsvidhya.com/blog/2015/08/common-machine-learning-algorithms/)

5. Evaluating Recommendation Engines

Before moving forward, I would like to extend my sincere gratitude to the **Coursera's Machine Learning Specialization (https://www.coursera.org/specializations/machine-learning) by University of Washington**. This course has been instrumental in my understanding of the concepts and this post is an illustration of my learnings from the same.

# 1. Type of Recommendation Engines

Before taking a look at the different types of recommendation engines, lets take a step back and see if we can make some intuitive recommendations. Consider the following cases:

## Case 1: Recommend the most popular items

A simple approach could be to recommend the items which are liked by most number of users. This is a blazing fast and dirty approach and thus has a major drawback. The things is, there is **no personalization** involved with this approach.

Basically the most popular items would be same for each user since popularity is defined on the entire user pool. So everybody will see the same results. It sounds like, 'a website recommends you to buy microwave just because it's been liked by other users and doesn't care if you are even interested in buying or not'.

Surprisingly, such approach still works in places like news portals. Whenever you login to say bbcnews, you'll see a column of "Popular News" which is subdivided into sections and the most read articles of each sections are displayed. This approach can work in this case because:

- There is division by section so user can look at the section of his interest.
- At a time there are only a few hot topics and there is a high chance that a user wants to read the news which is being read by most others

## Case 2: Using a classifier to make recommendation

We already know lots of **classification algorithms**. Let's see how we can use the same technique to make recommendations. Classifiers are parametric solutions so we just need to define some parameters (features) of the user and the item. The outcome can be 1 if the user likes it or 0 otherwise. This might work out in some cases because of following advantages:

- Incorporates personalization
- It can work even if the user's past history is short or not available

But has some major drawbacks as well because of which it is not used much in practice:

- The features might actually not be available or even if they are, they may not be sufficient to make a good classifier
- As the number of users and items grow, making a good classifier will become exponentially difficult

## Case 3: Recommendation Algorithms

A Complete Tutorial on Tree Based Modeling from Scratch (in R & Python) (https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/)

7 Types of Regression Techniques you should know! (https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/)

6 Easy Steps to Learn Naive Bayes Algorithm (with code in Python) (https://www.analyticsvidhya.com/blog/2015/09/naive-bayes-explained/)

Understanding Support Vector Machine algorithm from examples (along with code) (https://www.analyticsvidhya.com/blog/2015/10/understaing-support-vector-machine-example-code/)

22 must watch talks on Python for Deep Learning, Machine Learning & Data Science (from PyData 2017, Amsterdam) (https://www.analyticsvidhya.com/blog/2017/05/pydata-amsterdam-2017-machine-learning-deep-learning-data-science/)

A Complete Tutorial on Time Series Modeling in R (https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/)

RECENT POSTS

(https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-

Now lets come to the special class of algorithms which are tailor-made for solving the recommendation problem. There are typically two types of algorithms – Content Based and Collaborative Filtering. You should refer to our previous article (https://www.analyticsvidhya.com/blog/2016/03/exploring-building-banks-recommendation-system/) to get a complete sense of how they work. I'll give a short recap here.

1. **Content based algorithms:**
   - **Idea:** If you like an item then you will also like a "similar" item
   - Based on similarity of the items being recommended
   - It generally works well when its easy to determine the context/properties of each item. For instance when we are recommending the same kind of item like a movie recommendation or song recommendation.
2. **Collaborative filtering algorithms:**
   - **Idea:** If a person A likes item 1, 2, 3 and B like 2,3,4 then they have similar interests and A should like item 4 and B should like item 1.
   - This algorithm is entirely based on the past behavior and not on the context. This makes it one of the most commonly used algorithm as it is not dependent on any additional information.
   - For instance: product recommendations by e-commerce player like Amazon and merchant recommendations by banks like American Express.
   - Further, there are several types of collaborative filtering algorithms :
     1. **User-User Collaborative filtering:** Here we find look alike customers (based on similarity) and offer products which first customer's look alike has chosen in past. This algorithm is very effective but takes a lot of time and resources. It requires to compute every customer pair information which takes time. Therefore, for big base platforms, this algorithm is hard to implement without a very strong parallelizable system.
     2. **Item-Item Collaborative filtering:** It is quite similar to previous algorithm, but instead of finding customer look alike, we try finding item look alike. Once we have item look alike matrix, we can easily recommend alike items to customer who have purchased any item from the store. This algorithm is far less resource consuming than user-user collaborative filtering. Hence, for a new customer the algorithm takes far lesser time than user-user collaborate as we don't need all similarity scores between customers. And with fixed number of products, product-product look alike matrix is fixed over time.
     3. **Other simpler algorithms:** There are other approaches like market basket analysis (https://www.analyticsvidhya.com/blog/2014/08/visualizing-market-basket-analysis/), which generally do not have high predictive power than the algorithms described above.

## 2. The MovieLens DataSet

We will be using the MovieLens dataset for this purpose. It has been collected by the GroupLens Research Project at the University of Minnesota. MovieLens 100K dataset can be downloaded from here (http://grouplens.org/datasets/movielens/100k/). It consists of:

- **100,000 ratings** (1-5) from 943 users on 1682 movies.
- Each user has rated **at least 20 movies.**
- Simple demographic info for the users (age, gender, occupation, zip)
- Genre information of movies

Lets load this data into Python. There are many files in the **ml-100k.zip** file which we can use. Lets load the three most importance files to get a sense of the data. I also recommend you to read the *readme* document which gives a lot of information about the difference files.

the-art-of-fine-tuning-a-pre-trained-model/)

Transfer learning & The art of using Pre-trained Models in Deep Learning (https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/)

DISHASHREE GUPTA , JUNE 1, 2017

(https://www.analyticsvidhya.com/blog/2017/06/building-trust-in-machine-learning-models/)

Building Trust in Machine Learning Models (using LIME in Python) (https://www.analyticsvidhya.com/blog/2017/06/building-trust-in-machine-learning-models/)

GUEST BLOG , JUNE 1, 2017

(https://www.analyticsvidhya.com/blog/2017/05/neural-network-from-scratch-in-python-and-r/)

Understanding and coding Neural Networks From Scratch in Python and R (https://www.analyticsvidhya.com/blog/2017/05/neural-network-from-scratch-in-python-and-r/)

SUNIL RAY , MAY 29, 2017

(https://www.analyticsvidhya.com/blog/2017/05/launching-analytics-industry-report-2017-trends-and-salaries-in-india/)

Launching Analytics Industry Report 2017 – Trends and Salaries in India (https://www.analyticsvidhya.com/blog/2017/05/launching-analytics-industry-report-2017-trends-and-salaries-in-india/)

KUNAL JAIN , MAY 26, 2017

```
import pandas as pd

# pass in column names for each CSV and read them using pandas.
# Column names available in the readme file

#Reading users file:
u_cols = ['user_id', 'age', 'sex', 'occupation', 'zip_code']
users = pd.read_csv('ml-100k/u.user', sep='|', names=u_cols,
 encoding='latin-1')

#Reading ratings file:
r_cols = ['user_id', 'movie_id', 'rating', 'unix_timestamp']
ratings = pd.read_csv('ml-100k/u.data', sep='\t', names=r_cols,
 encoding='latin-1')

#Reading items file:
i_cols = ['movie id', 'movie title' ,'release date','video release date', 'I
MDb URL', 'unknown', 'Action', 'Adventure',
 'Animation', 'Children\'s', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fan
tasy',
 'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller
', 'War', 'Western']
items = pd.read_csv('ml-100k/u.item', sep='|', names=i_cols,
 encoding='latin-1')
```

Now lets take a peak into the content of each file to understand them better.

- Users

```
print users.shape
users.head()
```

```
(943, 5)
```

| | user_id | age | sex | occupation | zip_code |
|---|---|---|---|---|---|
| 0 | 1 | 24 | M | technician | 85711 |
| 1 | 2 | 53 | F | other | 94043 |
| 2 | 3 | 23 | M | writer | 32067 |
| 3 | 4 | 24 | M | technician | 43537 |
| 4 | 5 | 33 | F | other | 15213 |

(https://www.analyticsvidhya.com/wp-content/uploads/2016/06/1-users.png)

This reconfirms that there are 943 users and we have 5 features for each namely their unique ID, age, gender, occupation and the zip code they are living in.

- Ratings

```
print ratings.shape
ratings.head()
```

```
(100000, 4)
```

| | user_id | movie_id | rating | unix_timestamp |
|---|---|---|---|---|
| 0 | 196 | 242 | 3 | 881250949 |
| 1 | 186 | 302 | 3 | 891717742 |
| 2 | 22 | 377 | 1 | 878887116 |
| 3 | 244 | 51 | 2 | 880606923 |
| 4 | 166 | 346 | 1 | 886397596 |

(https://www.analyticsvidhya.com/wp-content/uploads/2016/06/2.-ratings.png)

This confirms that there are 100K ratings for different user and movie combinations. Also notice that each rating has a timestamp associated with it.

- Items

```
print items.shape
items.head()
```

```
(1682, 24)
```

| | movie id | movie title | release date | video release date | IMDb URL | unknown | Action | Adventure | Animation | Children's | ... | Fantasy | Film-Noir | Horror | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | Toy Story (1995) | 01-Jan-1995 | NaN | http://us.imdb.com/M/title-exact?Toy%20Story%2... | 0 | 0 | 0 | 1 | 1 | ... | 0 | 0 | 0 | 0 |
| 1 | 2 | GoldenEye (1995) | 01-Jan-1995 | NaN | http://us.imdb.com/M/title-exact?GoldenEye%20(... | 0 | 1 | 1 | 0 | 0 | ... | 0 | 0 | 0 | 0 |
| 2 | 3 | Four Rooms (1995) | 01-Jan-1995 | NaN | http://us.imdb.com/M/title-exact?Four%20Rooms%... | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 |
| 3 | 4 | Get Shorty (1995) | 01-Jan-1995 | NaN | http://us.imdb.com/M/title-exact?Get%20Shorty%... | 0 | 1 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 |
| 4 | 5 | Copycat (1995) | 01-Jan-1995 | NaN | http://us.imdb.com/M/title-exact?Copycat%20(1995) | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 |

(https://www.analyticsvidhya.com/wp-content/uploads/2016/06/3.-items.png)This dataset contains attributes of the 1682 movies. There are 24 columns out of which 19 specify the genre of a particular movie. The last 19 columns are for each genre and a value of 1 denotes movie belongs to that genre and 0 otherwise.

Now we have to divide the ratings data set into test and train data for making models. Luckily GroupLens provides pre-divided data wherein the test data has 10 ratings for each user, i.e. 9430 rows in total. Lets load that:

```
r_cols = ['user_id', 'movie_id', 'rating', 'unix_timestamp']
ratings_base = pd.read_csv('ml-100k/ua.base', sep='\t', names=r_cols, encodi
ng='latin-1')
ratings_test = pd.read_csv('ml-100k/ua.test', sep='\t', names=r_cols, encodi
ng='latin-1')
ratings_base.shape, ratings_test.shape
```

```
Output: ((90570, 4), (9430, 4))
```

Since we'll be using GraphLab, lets convert these in SFrames.

```
import graphlab
train_data = graphlab.SFrame(ratings_base)
test_data = graphlab.SFrame(ratings_test)
```

We can use this data for training and testing. Now that we have gathered all the data available. Note that here we have user behaviour as well as attributes of the users and movies. So we can make content based as well as collaborative filtering algorithms.

# 3. A Simple Popularity Model

Lets start with making a popularity based model, i.e. the one where **all the users have same recommendation** based on the most popular choices. We'll use the graphlab recommender functions popularity_recommender for this.

We can train a recommendation as:

```
popularity_model = graphlab.popularity_recommender.create(train_data, user_i
d='user_id', item_id='movie_id', target='rating')
```

Arguments:

- **train_data**: the SFrame which contains the required data
- **user_id**: the column name which represents each user ID
- **item_id**: the column name which represents each item to be recommended
- **target:** the column name representing scores/ratings given by the user

Lets use this model to make top 5 recommendations for first 5 users and see what comes out:

```
#Get recommendations for first 5 users and print them
#users = range(1,6) specifies user ID of first 5 users
#k=5 specifies top 5 recommendations to be given
popularity_recomm = popularity_model.recommend(users=range(1,6),k=5)
popularity_recomm.print_rows(num_rows=25)
```

| user_id | movie_id | score | rank |
|---|---|---|---|
| 1 | 1500 | 5.0 | 1 |
| 1 | 1201 | 5.0 | 2 |
| 1 | 1189 | 5.0 | 3 |
| 1 | 1122 | 5.0 | 4 |
| 1 | 814 | 5.0 | 5 |
| 2 | 1500 | 5.0 | 1 |
| 2 | 1201 | 5.0 | 2 |
| 2 | 1189 | 5.0 | 3 |
| 2 | 1122 | 5.0 | 4 |
| 2 | 814 | 5.0 | 5 |
| 3 | 1500 | 5.0 | 1 |
| 3 | 1201 | 5.0 | 2 |
| 3 | 1189 | 5.0 | 3 |
| 3 | 1122 | 5.0 | 4 |
| 3 | 814 | 5.0 | 5 |
| 4 | 1500 | 5.0 | 1 |

```
|     4     |    1201    |   5.0   |   2    |
|     4     |    1189    |   5.0   |   3    |
|     4     |    1122    |   5.0   |   4    |
|     4     |    814     |   5.0   |   5    |
|     5     |    1500    |   5.0   |   1    |
|     5     |    1201    |   5.0   |   2    |
|     5     |    1189    |   5.0   |   3    |
|     5     |    1122    |   5.0   |   4    |
|     5     |    814     |   5.0   |   5    |
+-----------+------------+---------+--------+
[25 rows x 4 columns]
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/06/4.-popularity-recomm.png)

Did you notice something? The recommendations for all users are same – 1500,1201,1189,1122,814 in the same order. This can be verified by checking the movies with highest mean recommendations in our ratings_base data set:

```
ratings_base.groupby(by='movie_id')['rating'].mean().sort_values(ascending=False).head(20)
```

```
movie_id
1500    5.000000
1293    5.000000
1122    5.000000
1189    5.000000
1656    5.000000
1201    5.000000
1599    5.000000
814     5.000000
1467    5.000000
1536    5.000000
1449    4.714286
1642    4.500000
1463    4.500000
1594    4.500000
1398    4.500000
114     4.491525
408     4.480769
169     4.476636
318     4.475836
483     4.459821
Name: rating, dtype: float64
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/06/5.-mean-ratings.png)

This confirms that all the recommended movies have an average rating of 5, i.e. all the users who watched the movie gave a top rating. Thus we can see that our popularity system works as expected. But it is good enough? We'll analyze it in detail later.

# 4. A Collaborative Filtering Model

Lets start by understanding the basics of a collaborative filtering algorithm. The core idea works in 2 steps:

1. Find similar items by using a similarity metric
2. For a user, recommend the items most similar to the items (s)he already likes

To give you a high level overview, this is done by making an **item-item matrix** in which we keep a record of the pair of items which were rated together.

In this case, an item is a movie. Once we have the matrix, we use it to determine the best recommendations for a user based on the movies he has already rated. Note that there a few more things to take care in actual implementation which would require deeper mathematical introspection, which I'll skip for now.

I would just like to mention that there are 3 types of item similarity metrics supported by graphlab. These are:

1. **Jaccard Similarity:**
    - Similarity is based on the number of users which have rated item A and B divided by the number of users who have rated either A or B
    - It is typically used where we don't have a numeric rating but just a boolean value like a product being bought or an add being clicked
2. **Cosine Similarity:**
    - Similarity is the cosine of the angle between the 2 vectors of the item vectors of A and B
    - Closer the vectors, smaller will be the angle and larger the cosine
3. **Pearson Similarity**
    - Similarity is the pearson coefficient between the two vectors.

Lets create a model based on item similarity as follow:

```
#Train Model
item_sim_model = graphlab.item_similarity_recommender.create(train_data, use
r_id='user_id', item_id='movie_id', target='rating', similarity_type='pearso
n')

#Make Recommendations:
item_sim_recomm = item_sim_model.recommend(users=range(1,6),k=5)
item_sim_recomm.print_rows(num_rows=25)
```

```
+---------+----------+----------------+------+
| user_id | movie_id |     score      | rank |
+---------+----------+----------------+------+
|    1    |   1463   | 5.33134491184  |  1   |
|    1    |   1639   | 5.19047619048  |  2   |
|    1    |   1449   | 5.13030574673  |  3   |
|    1    |   1201   |      5.0       |  4   |
|    1    |   1189   |      5.0       |  5   |
|    2    |   1449   | 5.60893773837  |  1   |
|    2    |   1642   | 5.40357227269  |  2   |
|    2    |   1594   | 5.40357227269  |  3   |
|    2    |   709    | 5.31676089125  |  4   |
|    2    |   657    | 5.25555989075  |  5   |
|    3    |   157    | 5.61820784799  |  1   |
|    3    |   189    | 5.52914160744  |  2   |
|    3    |   607    | 5.42510822511  |  3   |
|    3    |   1142   | 5.31481481481  |  4   |
|    3    |   482    | 5.30634023854  |  5   |
|    4    |   1007   | 6.16067653277  |  1   |
|    4    |   251    | 6.07219460669  |  2   |
|    4    |    14    | 6.04743083004  |  3   |
|    4    |   811    |  6.0320855615  |  4   |
|    4    |   608    | 6.02194357367  |  5   |
|    5    |   114    | 5.54641910543  |  1   |
```

```
|    5     |   1137    |  5.5319284802  |  2   |
|    5     |   960     | 5.42959001783  |  3   |
|    5     |   1251    | 5.33817829457  |  4   |
|    5     |   1158    |     5.3125     |  5   |
+----------+-----------+----------------+------+
[25 rows x 4 columns]
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/06/6.-similarity-model-1.png)

Here we can see that the recommendations are different for each user. So, personalization exists. But how good is this model? We need some means of evaluating a recommendation engine. Lets focus on that in the next section.

## 5. Evaluating Recommendation Engines

For evaluating recommendation engines, we can use the concept of precision-recall. You must be familiar with this in terms of classification and the idea is very similar. Let me define them in terms of recommendations.

- **Recall:**
  - What ratio of items that a user likes were actually recommended.
  - If a user likes say 5 items and the recommendation decided to show 3 of them, then the recall is 0.6
- **Precision**
  - Out of all the recommended items, how many the user actually liked?
  - If 5 items were recommended to the user out of which he liked say 4 of them, then precision is 0.8

Now if we think about recall, how can we maximize it? If we simply recommend all the items, they will definitely cover the items which the user likes. So we have 100% recall! But think about precision for a second. If we recommend say 1000 items and user like only say 10 of them then precision is 0.1%. This is really low. Our aim is to maximize both precision and recall.

An idea recommender system is the one which only recommends the items which user likes. So in this case precision=recall=1. This is an optimal recommender and we should try and get as close as possible.

Lets compare both the models we have built till now based on precision-recall characteristics:

```
model_performance = graphlab.compare(test_data, [popularity_model, item_sim_model])
graphlab.show_comparison(model_performance,[popularity_model, item_sim_model])
```

```
PROGRESS: Evaluate model M0

Precision and recall summary statistics by cutoff
+--------+--------------------+--------------------+
| cutoff |   mean_precision   |    mean_recall     |
+--------+--------------------+--------------------+
|   1    |        0.0         |        0.0         |
|   2    | 0.000530222693531  | 0.000106044538706  |
|   3    | 0.000353481795688  | 0.000106044538706  |
|   4    | 0.000265111346766  | 0.000106044538706  |
```

```
|    5     | 0.000212089077413 | 0.000106044538706 |
|    6     | 0.000176740897844 | 0.000106044538706 |
|    7     | 0.000151492198152 | 0.000106044538706 |
|    8     | 0.000132555673383 | 0.000106044538706 |
|    9     | 0.000117827265229 | 0.000106044538706 |
|    10    | 0.000212089077413 | 0.000212089077413 |
+--------+-------------------+-------------------+
[10 rows x 3 columns]

PROGRESS: Evaluate model M1

Precision and recall summary statistics by cutoff
+--------+-------------------+-------------------+
| cutoff |  mean_precision   |    mean_recall    |
+--------+-------------------+-------------------+
|    1     | 0.0084835630965   | 0.00084835630965  |
|    2     | 0.00742311770944  | 0.00148462354189  |
|    3     | 0.00777659950513  | 0.00233297985154  |
|    4     | 0.00715800636267  | 0.00286320254507  |
|    5     | 0.00615058324496  | 0.00307529162248  |
|    6     | 0.00618593142453  | 0.00371155885472  |
|    7     | 0.00605968792607  | 0.00424178154825  |
|    8     | 0.00596500530223  | 0.00477200424178  |
|    9     | 0.00612701779192  | 0.00551431601273  |
|    10    | 0.00583244962884  | 0.00583244962884  |
+--------+-------------------+-------------------+
[10 rows x 3 columns]
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/06/7.-evaluate.png)

Here we can make 2 very quick observations:

1. The item similarity model is definitely better than the popularity model (by atleast 10x)
2. On an absolute level, even the item similarity model appears to have a poor performance. It is far from being a useful recommendation system.

There is a big scope of improvement here. But I leave it up to you to figure out how to improve this further. I would like to give a couple of tips:

1. Try leveraging the additional context information which we have
2. Explore more sophisticated algorithms like matrix factorization

In the end, I would like to mention that along with GraphLab, you can also use some other open source python packages like the following:

- **Crab (http://muricoca.github.io/crab/)**.
- Surprise (https://github.com/NicolasHug/Surprise)
- Python Recsys (https://github.com/ocelma/python-recsys)
- MRec (https://github.com/Mendeley/mrec)

# End Notes

In this article, we traversed through the process of making a basic recommendation engine in Python using GrpahLab. We started by understanding the fundamentals of recommendations. Then we went on to load the MovieLens 100K data set for the purpose of experimentation.

Subsequently we made a first model as a simple popularity model in which the most popular movies were recommended for each user. Since this lacked personalization, we made another model based on collaborative filtering and

observed the impact of personalization.

Finally, we discussed precision-recall as evaluation metrics for recommendation systems and on comparison found the collaborative filtering model to be more than 10x better than the popularity model.

Did you like reading this article ? Do share your experience / suggestions in the comments section below.

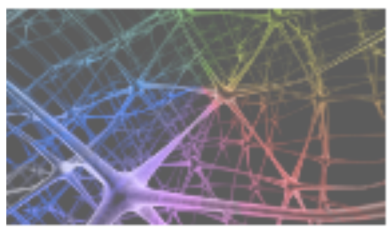
## You can test your skills and knowledge. Check out Live Competitions (http://datahack.analyticsvidhya.com/contest/all) and compete with best Data Scientists from all over the world.



## RELATED

(https://www.analyticsvidhya.com/blog/2014/04/training-recommendation-tutorials-pycon-2014-usa/)
Training recommendation (Tutorials from PyCon 2014 - USA) and contest update (https://www.analyticsvidhya.com/blog/2014/04/training-recommendation-tutorials-pycon-2014-usa/)
April 24, 2014
In "Big data"

(https://www.analyticsvidhya.com/blog/2016/03/exploring-building-banks-recommendation-system/)
Exploring Recommendation System (with an implementation model in R) (https://www.analyticsvidhya.com/blog/2016/03/exploring-building-banks-recommendation-system/)
March 24, 2016
In "Business Analytics"



(https://www.analyticsvidhya.com/blog/2017/02/5-deep-learning-applications-beginner-python/)
5 More Deep Learning Applications a beginner can build in minutes (using Python) (https://www.analyticsvidhya.com/blog/2017/02/5-deep-learning-applications-beginner-python/)
February 22, 2017
In "Deep Learning"

TAGS: COLLABORATIVE FILTERING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/COLLABORATIVE-FILTERING/), CONTENT BASED RECOMMENDATION ENGINES (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/CONTENT-BASED-RECOMMENDATION-ENGINES/), GRAPHLAB IN PYTHON (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/GRAPHLAB-IN-PYTHON/), GRAPHLAB TUTORIAL (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/GRAPHLAB-TUTORIAL/), ITEM ITEM COLLABORATIVE FILTERING

(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/ITEM-ITEM-COLLABORATIVE-FILTERING/), RECOMMENDATION ENGINES (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/RECOMMENDATION-ENGINES/), RECOMMENDATION WITH GRAPHLAB (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/RECOMMENDATION-WITH-GRAPHLAB/)

Previous Article
Director / VP (Analytics) – Gurgaon (8 – 14 Years of Experience) (https://www.analyticsvidhya.com/blog/2016/05/director-vp-analytics-gurgaon-8-14-years-experience/)

Next Article
Getting Started with Big Data Integration using HDFS and DMX-h (https://www.analyticsvidhya.com/blog/2016/06/started-big-data-integration-hdfs-dmexpress/)

(https://www.analyticsvidhya.com/blog/author/aarshay/)

Author

## Aarshay Jain (https://www.analyticsvidhya.com/blog/author/aarshay/)

Aarshay is a ML enthusiast, pursuing MS in Data Science at Columbia University, graduating in Dec 2017. He is currently exploring the various ML techniques and writes articles for AV to share his knowledge with the community.

(mailto:aarshayjain@gmail.com)

(https://in.linkedin.com/in/aarshayjain)

(https://github.com/aarshayj) (aarshay)

This is article is quiet old now and you might not get a prompt response from the author. We would request you to post this comment on Analytics Vidhya **Discussion portal** (https://discuss.analyticsvidhya.com/) to get your queries resolved.

## 32 COMMENTS

**Vinay Jain says:**
JUNE 2, 2016 AT 5:18 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111735)
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111735#RESPOND)

I am completly new to the field of data science. I have started courses of Machine Learning.
Can you please suggest me how to proceed or should I consider some other options. I plan to proceed further in the field of AI. Please suggest how should i carry on or begin my journey. ?

**Aarshay Jain says:**
JUNE 2, 2016 AT 5:49 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111737)
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111737#RESPOND)

Thanks for reaching out. I'm sorry but there is no fixed answer to your question and this thread is probably not the right place to answer. I

recommend reading similar discussions on http://discuss.analyticsvidhya.com (http://discuss.analyticsvidhya.com) and you can start a new thread as well.

You can also check out the learning paths on our website if you're interested in a particular tool.

Hope this helps.

---

**r4sp4 says:**

REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111885#RESPOND)

JUNE 5, 2016 AT 1:14 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111885)

In this blog, data is already divided into train and test. But ,how to divide data into Train and Test? On what basis to make this decision?

---

**Aarshay Jain says:**

REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111918#RESPOND)

JUNE 6, 2016 AT 9:02 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111918)

There can be various considerations. One of them can be simply based on time, i.e. initial part as train then test. But you can face a cold start problem here, i.e. there might be some users in test which are not in train.
Another option is to keep a particular number of entries of a user in the test file. This approach has been followed here. But in real case scenarios, the time approach should be used because it is more practical.

---

**Friggits says:**

REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111736#RESPOND)

JUNE 2, 2016 AT 5:30 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111736)

Good article , very educative

---

**Aarshay Jain says:**

REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111739#RESPOND)

JUNE 2, 2016 AT 6:06 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111739)

Thanks you!

---

**Hulisani says:**

REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111740#RESPOND)

JUNE 2, 2016 AT 6:18 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111740)

Thanks for sharing such an Amazing article, can I please have in pdf.

---

**Aarshay Jain says:**

REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111759#RESPOND)

JUNE 2, 2016 AT 1:35 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111759)

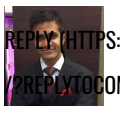
Thanks Hulisani! Unfortunately, we don't have proper pdf formats. Generally what I do is print the page and save it as a pdf. I won't look that good but mostly works.

---

**Naveed1228 says:**
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111741#RESPOND)
JUNE 2, 2016 AT 6:43 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111741)

Nice Article.How can we use this article on website using Flask or Django.I am newbie if you suggest me some resources.I will thankful to you.

---



**Aarshay Jain says:**
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111760#RESPOND)
JUNE 2, 2016 AT 1:37 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111760)

I think GraphLab has deployment facility which you can use. I'm not sure how exactly that works. Another option could be to make a RestFul API of your recommender and call it on the website.

---

**Ron Williams says:**
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111744#RESPOND)
JUNE 2, 2016 AT 7:24 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111744)

Hi, thanks for the article, which is, in itself very useful.

I just wanted to highlight what I see as a limitation of 'recommender' systems, such as google's or YouTube's. Especially YT. The nub of my objection to them is that they either recommend things I've already seen, or very similar things. In a lot of cases I've basically already moved on from that, and really don't want more of the same.

E.g. suppose I accidentally watch a 'Game of Thrones' clip; YT will ad nauseam present more of them, and most especially what irks me is that they present the exact same clip, but uploaded by a different user. grrr…

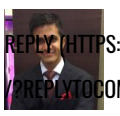
Or, say I look at 'How to Create an Amazon Affiliate Site in 25.3 seconds and make SQUILLIONS' – yep, you guessed it, I get the ones on how to do that zact-same thing in 16.3 seconds and make $500, or $344,253.45 or …

And with google – say I've been looking for products for the abovementioned Affiliate site. Google will have picked up on that and will most irritatingly present me, for literally days with those products, which I have absolutely no intention of buying.

Another annoying 'feature' is, suppose I've already bought a product. Google will assume I'm interested in such things and present me with opportunities to buy even more of the thing that I already have one of, and really, really don't need any more. I've got one. Thanks anyhow.

Recommender systems have a looong way to go, to be actually useful as marketing tools, as opposed to irritants.

---



**Aarshay Jain says:**
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111761#RESPOND)
JUNE 2, 2016 AT 1:40 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111761)

Thanks for sharing your thoughts. I agree with you totally. But I think its a good things. We can an untapped potential and this gives a perfect opportunity to explore this further and design better systems.

I think one potential reason causing this could be that Google and say Amazon talk to each other only superficially. So google might get the info that a user is searching for a product but it might not have info about whether the product is already bought. This is just

my assumption and I don't know how that system actually works. I will explore this further for sure!

**basel says:**
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111749#RESPOND)
JUNE 2, 2016 AT 9:09 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111749)

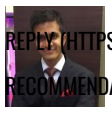
how can someone build a very big recommender system with like netflix , i mean what is the way for that?

**Aarshay Jain says:**
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111762#RESPOND)
JUNE 2, 2016 AT 1:42 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111762)

I belive GraphLab is a scalable tool. You can use it on large datasets as well. But I'm not sure whether it'll be possible on our PC or a GraphLab server is required. You can contact GraphLab directly for this.

**basel says:**
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111769#RESPOND)
JUNE 2, 2016 AT 3:23 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111769)

no i'm not talking about scalability i'm talking about real recommendation system like the one which is being used in netflix or quora ...etc of course these systems don't use simple algorithms with 1 peice of code , of course they are using them but they add to them complex new algorithms in machine learning and recommendation system's algorithms



**Aarshay Jain says:**
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111770#RESPOND)
JUNE 2, 2016 AT 3:28 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111770)

For that I recommend going through research articles. I'm pretty sure you'll find papers from netflix. Not sure about Quora..

**james says:**
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111765#RESPOND)
JUNE 2, 2016 AT 2:33 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111765)

I am a R person. Are there similar codes in R instead of Python ?

**Aarshay Jain says:**
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111766#RESPOND)
JUNE 2, 2016 AT 2:44 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111766)

Yes there are 2 more articles on recommendation engines based on R:
– http://www.analyticsvidhya.com/blog/2016/03/exploring-building-banks-recommendation-system/ (http://www.analyticsvidhya.com/blog/2016/03/exploring-building-banks-recommendation-system/)
– http://www.analyticsvidhya.com/blog/2015/10/recommendation-

engines/ (http://www.analyticsvidhya.com/blog/2015/10/recommendation-engines/)

---

**Ash says:**
JUNE 6, 2016 AT 12:01 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-111925)
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=111925#RESPOND)

Hey!

I am developing a model similar to the one in this link: http://www.salemmarafi.com/code/collaborative-filtering-with-python/ (http://www.salemmarafi.com/code/collaborative-filtering-with-python/).

How do you think I should evaluate this model?

Any suggestions please?

---

**Aarshay Jain says:**
JUNE 13, 2016 AT 11:40 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-112184)
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=112184#RESPOND)

have you tried the precision-recall technique which I've explained here?

---

**manish says:**
JUNE 13, 2016 AT 11:05 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-112182)
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=112182#RESPOND)

I am having Trouble while importing 'graphlab', when I import and run I am getting the following output:

————————————————————
File "", line 32, in
import graphlab as gl

ImportError: No module named 'graphlab'

————————————————————
I have googled out things but am unable to find any specific solution on this, I am using Windows and using Anaconda – Spyder !!!

---

**Aarshay Jain says:**
JUNE 13, 2016 AT 11:39 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-112183)
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=112183#RESPOND)

You need to install graphlab first. The first year license is free.

---

**manish says:**
JUNE 27, 2016 AT 10:55 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-112740)
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=112740#RESPOND)

Thanks, great article.

---

**Kaustubh Sakhalkar says:**
JUNE 14, 2016 AT 12:30 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-112212)
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=112212#RESPOND)

Hi Aarshay excellent article! helped me immensely with a project I am working

on. Keep them coming!

**Aarshay Jain says:**

REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=112215#RESPOND)

JUNE 14, 2016 AT 1:51 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-112215)

Sure stay tuned!

**Apurva says:**

REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=117858#RESPOND)

NOVEMBER 3, 2016 AT 5:30 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-117858)

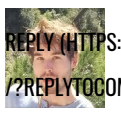
Good article Aarshay. I landed on this when exploring some other stuff.

**Aarshay Jain says:**

REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=119853#RESPOND)

DECEMBER 16, 2016 AT 5:12 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-119853)

Thanks Apurva!



**Nicolas Hug says:**

REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=118558#RESPOND)

NOVEMBER 19, 2016 AT 11:18 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-118558)

Hi! Nice article!

I saw you mentioned Crab as an alternative to GraphLab, but unfortunately crab insn't maintained anymore.

Some other Python recommender system libraries are python-recsys (https://github.com/ocelma/python-recsys (https://github.com/ocelma/python-recsys), in support mode I suppose), mrec (https://github.com/Mendeley/mrec (https://github.com/Mendeley/mrec)) and RecSys (https://github.com/Niourf/RecSys (https://github.com/Niourf/RecSys) — I am the creator of RecSys).

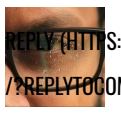


**Aarshay Jain says:**

REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=119852#RESPOND)

DECEMBER 16, 2016 AT 5:10 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-119852)

Hi Nicolas. Thanks for reaching out! I'll add this to the post.



**Lobeeb Ibrahim says:**

REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=119348#RESPOND)

DECEMBER 7, 2016 AT 10:24 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-119348)

graphlab doesn't work for Python3. Do you suggest a substitute?

**Aarshay Jain says:**

REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/?REPLYTOCOM=119851#RESPOND)

DECEMBER 16, 2016 AT 5:04 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/06/QUICK-GUIDE-BUILD-RECOMMENDATION-ENGINE-PYTHON/#COMMENT-119851)

I suggested crab as an alternative. But see the comment from Nicolas Hug for further details.

## LEAVE A REPLY

Your email address will not be published.

Comment

Name (required)

Email (required)

Website

SUBMIT COMMENT

### ANALYTICS VIDHYA

About Us (https://www.analyticsvidhya.com/about-me/)
Team (https://www.analyticsvidhya.com/about-me/team)
Volunteers (https://www.analyticsvidhya.com/av-volunteers/)
Careers (https://www.analyticsvidhya.com/about-me/career-analytics-vidhya/)
Write for us (https://www.analyticsvidhya.com/about-me/write/)
Contact (https://www.analyticsvidhya.com/contact/)

### DATA SCIENTISTS

Blog (http://analyticsvidhya.com/blog)
Discuss (https://discuss.analyticsvidhya.com)
Learning Paths (http://analyticsvidhya.com/learning-paths-data-science-business-analytics-business-intelligence-big-data/)
DataHack (https://datahack.analyticsvidhya.com)
Events (https://analyticsvidhya.com/blog/category/events/)
Jobs (https://www.analyticsvidhya.com/jobs/)

### COMPANIES

Advertise with Us (https://www.analyticsvidhya.com/contact/)
Recruit using AV (https://www.analyticsvidhya.com/jobs/company/signup/)
Hackathon (https://datahack.analyticsvidhya.com/)
DataFest-2017 (https://www.analyticsvidhya.com/datafest-2017)

### SOCIAL MEDIA

9,799 FOLLOWERS
(http://www.twitter.com/analyticsvidhya)
33,695 FOLLOWERS
(http://www.facebook.com/Analyticsvidhya)
1,878 FOLLOWERS
(https://plus.google.com/+Analyticsvidhya)
Email SUBSCRIBE
(https://feedburner.google.com/fb/a/mailverify?uri=analyticsvidhya)