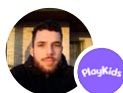




# Recommender systems with collaborative filters



Gabriel Ghellere · [Follow](#)

Published in PlayKids Tech Blog

7 min read · Sep 10, 2019



Listen



Share



Image from Pixabay

## Intro

A recommender system (or recommendation system) is a very important tool for a data scientist, because such systems are widely used by small and large companies for a variety of purposes. Their main mission is to predict the “rating” or “preference” of a user given an item. Amazon, for example, uses recommendation systems to recommend books for their customers based on previous purchases.

YouTube and Netflix use recommendation systems for recommending new content to their customers.

## **Recommender systems goals**

Recommender systems promise a lot of benefits to business, such as:

- Increased conversion rate;
- Higher profit margin;
- The user will like your platform better;
- Improves User Experience;
- Increases your LTV and decreases Churn rate.

## **Common approaches**

There are several ways to build a recommendation system, using complex machine learning algorithms or just basic math, the most popular approaches being collaborative filters and content-based filter.

### **Collaborative filters**

In this class of algorithms, for each user an item is indicated based on another user with “similar behavior”. For example, Alice and Bob are two people with similar interests in TV shows. Alice recently watched and liked Game of Thrones, but Bob has not watched it yet. The recommendation system will automatically recommend Game of Thrones to Bob, since his interests are similar to Alice’s.

### **Content-based filters**

When the company knows its products in detail, it is possible to implement a content-based filter system. To implement it, it is necessary to classify all products with tags or similar properties. For example, if I watch the Supernatural TV show, it contains the ‘terror’ and ‘comedy’ tags, so the algorithm will recommend similar tagged content, like the TV show Lucifer.

## **End-to-end collaborative filter**

Some steps are required to implement a basic collaborative filter system:

- Get similarity between users or objects;
- Understand the mathematical concept of similarity;

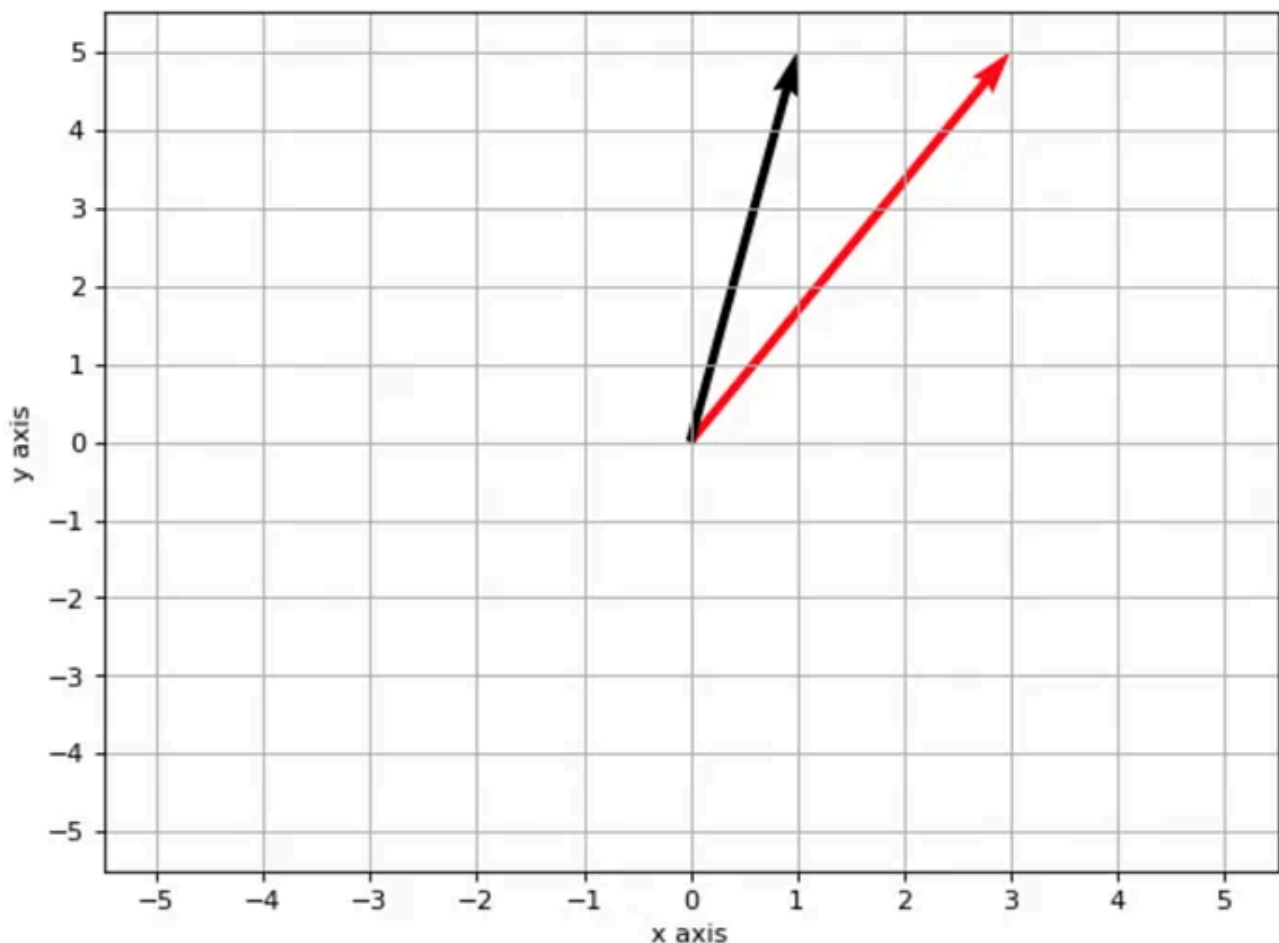
- Implement the recommender model.

## Measuring similarity

The concept of similarity is a complement to distance. The distance between two points can be easily obtained using the Euclidean distance, by calculating the length of a straight line connecting two points. When we talk about similarity, instead of finding out how far two points are, our goal is to find out “how close” the points are. The similarity value is usually expressed as a number between 0 and 1, where 0 means a low similarity, and 1 a high similarity.

## Cosine similarity

With two vectors  $U$  and  $V$ , where  $U=(3, 5)$  and  $V=(1, 5)$ , the distance between  $U$  and  $V$  can be easily calculated by using Euclidean distance. But how *similar* are these vectors? A common way to measure similarity is called *cosine similarity*.

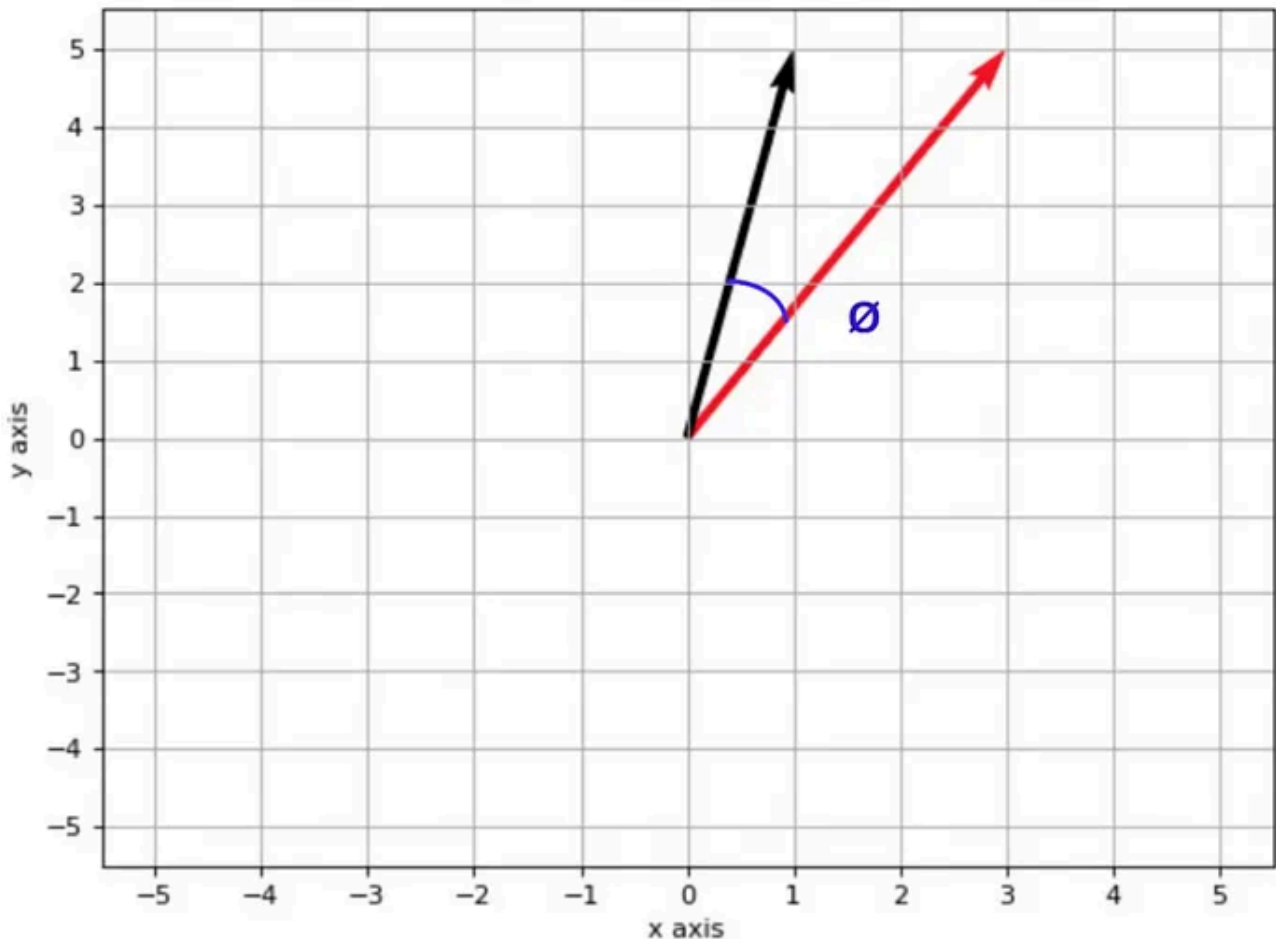


U (1,5) red and V (3,5) black. Image from author

To measure similarity, we need to calculate the angle between the vectors. The smaller the angle, the greater the similarity. To verify, just remember the

properties of the cosine function:

- $\cos(0^\circ) = 1$
- $\cos(90^\circ) = 0$
- $\cos(180^\circ) = -1$



Theta angle between the vectors. In this case  $19.65^\circ$ , similarity = 0.69

That is, using cosine similarity, when the angle between the vectors approaches  $0^\circ$ , the vectors are very similar (parallel with the same direction), so the similarity is 1. When the angle reaches  $90^\circ$ , the vectors are orthogonal and the similarity is  $0^\circ$ . Finally, with an angle of  $180^\circ$ , the vectors are opposite with similarity -1. Considering only non-negative vectors (in the first quadrant of the Cartesian plane) the similarity varies only between 0 and 1.

With this concept in mind, the similarity can be calculated with the following equation:

$$similarity = \cos(\theta) = \frac{A \cdot B}{||A|| ||B||}$$

Similarity Equation

This formula exists in every Linear Algebra book and should be familiar to engineering, physics and mathematics students.

### Collaborative filter algorithm

The next step after learning how to calculate similarity, is to implement the concept in a collaborative filter algorithm. The process uses similarity between users of a fictional platform.

Defining the problem:

- Our problem is movie recommendation. Given an  $[i, j]$  matrix, consider the rating of  $i$  users for  $j$  movies with the user rating represented by an integer ranging from 1 to 5. If the movie was not rated, its rating is 0;
- For each user, the goal is to recommend a series of movies they have not watched (rating = 0);
- For each movie  $j$  and user  $i$  who did not watch the movie, we try to find a user from a group  $U$  that is similar to user  $i$  who has watched the movie  $j$ ;
- For each similar user  $u$  get the rating for the movie  $j$  and multiply by the cosine similarity of the user  $i$  with  $u$ . Dividing the result by the number of users in the  $U$  set, we have the weighted average rating for the movie  $j$ ;
- Now sort the movies by the average obtained for each user and use this value to estimate the user's rating of the movie, then recommend the first  $N$  movies in the ranking for each user.

This technique is so similar to the KNN algorithm that, with some adaptations, it can be used to solve the recommendation problem (which will be explained later).

For example: imagine that I did not watch the movie Forrest Gump, but I watched and rated many other movies. All you have to do is to find a group of users who

watched movies that I also watched, and also watched Forrest Gump.

In this example, let's suppose that we find 2 users with preferences similar to mine. Let's assume that the first one has a 90% similarity to me and has given 5 stars to the movie, and the second one has an 80% similarity to me and has given 3 stars to the movie. To estimate my rating, the weighted average  $(0.9 \times 5 + 0.8 \times 3) / 2 = 3.45$  is calculated.

## Surprise

Surprise is a Python library for building and analysing recommender systems. There is also a built-in MovieLens Dataset with 100,000 movie ratings, from 1,000 users and 1,700 movies. The main purpose of Surprise is to give their users the perfect control over their experiments, provide several ready-to-use prediction algorithms - such as SVD - neighbourhood methods, similarity measures and tools to analyse and compare algorithm performance.

Let's explore Surprise.

Load the MovieLens 100k dataset and import it in Python:

```
1  from surprise import Dataset, evaluate
2  from surprise import KNNBasic
3
4  data = Dataset.load_builtin("ml-100k")
5  trainingSet = data.build_full_trainset()
```

surprise.py hosted with ❤ by GitHub

[view raw](#)

The next step is to select an estimator and similarity measure. In this implementation, the idea is to use a collaborative item-item filtering approach. The estimator selected is KnnBasic with cosine similarity:

```
1  knn = KNNBasic(sim_options={
2      'name': 'cosine',
3      'user_based': False })
4  knn.train(trainingSet)
```

surprise\_train.py hosted with ❤ by GitHub

[view raw](#)

The method *build\_anti\_testset* allows us to make recommendations to new users, finding all the user-movie pairs in the training set where the user has not watched the movie, and creates a “test suite” from these entries. From its result, we sort and retrieve the top 3 recommendations for each user:

```
1 testSet = trainingSet.build_anti_testset()
2 predictions = knn.test(testSet)
3
4 from collections import defaultdict
5
6 def get_top3_recommendations(predictions, topN = 3):
7
8     top_recs = defaultdict(list)
9     for uid, iid, true_r, est, _ in predictions:
10         top_recs[uid].append((iid, est))
11
12     for uid, user_ratings in top_recs.items():
13         user_ratings.sort(key = lambda x: x[1], reverse = True)
14         top_recs[uid] = user_ratings[:topN]
15
16     return top_recs
```

surprise\_predict.py hosted with ❤ by GitHub

[view raw](#)

To see the output, just map the movie ID with their name to each user. In the surprise Github dataset you can find this code and others.

```
1 import os, io
2
3 def read_item_names():
4     """Read the u.item file from MovieLens 100-k dataset and returns a
5     mapping to convert raw ids into movie names.
6     """
7
8     file_name = (os.path.expanduser('~') +
9                 '/.surprise_data/ml-100k/ml-100k/u.item')
10    rid_to_name = {}
11    with io.open(file_name, 'r', encoding='ISO-8859-1') as f:
12        for line in f:
13            line = line.split('|')
14            rid_to_name[line[0]] = line[1]
15
16    return rid_to_name
```

surprise\_map.py hosted with ❤ by GitHub

[view raw](#)



## The result:

```
1 196 ['Very Natural Thing, A (1974)', 'Walk in the Sun, A (1945)', 'War at Home, The (1996)']
2 186 ['Mamma Roma (1962)', 'Conspiracy Theory (1997)', 'Toy Story (1995)']
3 22 ['Entertaining Angels: The Dorothy Day Story (1996)', 'King of New York (1990)', 'Usual Susp
4 244 ['Other Voices, Other Rooms (1997)', 'Big Bang Theory, The (1994)', 'Godfather, The (1972)'
5 166 ['Mamma Roma (1962)', 'Delta of Venus (1994)', 'Carmen Miranda: Bananas Is My Business (199
6 298 ['North by Northwest (1959)', 'Pinocchio (1940)', 'Amadeus (1984)']
7 115 ['2001: A Space Odyssey (1968)', 'Clockwork Orange, A (1971)', 'Three Colors: White (1994)'
8 253 ['Entertaining Angels: The Dorothy Day Story (1996)', 'Michael (1996)', 'Empire Strikes Bac
9 305 ['Lone Star (1996)', 'African Queen, The (1951)', 'My Left Foot (1989)']
10 6 ['Bullets Over Broadway (1994)', 'Rosewood (1997)', 'Rear Window (1954)']
11 62 ['Fugitive, The (1993)', 'Cyclo (1995)', 'King of New York (1990)']
12 286 ['Other Voices, Other Rooms (1997)', 'Big Bang Theory, The (1994)', 'B. Monkey (1998)']
13 200 ['Usual Suspects, The (1995)', 'Batman (1989)', 'Glory (1989)']
14 210 ['Cyclo (1995)', 'Chairman of the Board (1998)', 'Death in Brunswick (1991)']
15 224 ['King of New York (1990)', 'Very Natural Thing, A (1974)', 'Walk in the Sun, A (1945)']
16 303 ['Mamma Roma (1962)', 'Entertaining Angels: The Dorothy Day Story (1996)', 'Substance of Fi
17 122 ['Convent, The (Convento, O) (1995)', 'Every Other Weekend (1990)', 'Homage (1995)']
18 194 ['L.A. Confidential (1997)', 'Reservoir Dogs (1992)', 'Chairman of the Board (1998)']
19 291 ['Other Voices, Other Rooms (1997)', 'Big Bang Theory, The (1994)', 'Chairman of the Board
20 234 ['War, The (1994)', 'They Made Me a Criminal (1939)', 'Leading Man, The (1996)']
21 119 ['Pushing Hands (1992)', 'Reluctant Debutante, The (1958)', 'Casablanca (1942)']
22 167 ['Legal Deceit (1997)', 'King of New York (1990)', 'Marlene Dietrich: Shadow and Light (199
23 299 ['Death in Brunswick (1991)', 'Chasing Amy (1997)', 'Substance of Fire, The (1996)']
24 308 ['Little Big League (1994)', 'Son in Law (1993)', 'Black Beauty (1994)']
25 95 ['Cinderella (1950)', 'Dave (1993)', 'In the Line of Fire (1993)']
26 38 ['Other Voices, Other Rooms (1997)', 'Big Bang Theory, The (1994)', 'Cyclo (1995)']
27 102 ['Mirage (1995)', 'Love Is All There Is (1996)', 'Damsel in Distress, A (1937)']
28 63 ['Aiqing wansui (1994)', 'Someone Else's America (1995)', 'Heathers (1989)']
29 160 ['Aiqing wansui (1994)', 'Entertaining Angels: The Dorothy Day Story (1996)', 'Leading Man,
30 50 ['Condition Red (1995)', 'Delta of Venus (1994)', 'Very Natural Thing, A (1974)']
```

results.txt hosted with ❤ by GitHub

[view raw](#)

As you can see, each user receives personalised recommendations based on past ratings. A weakness of collaborative filtering is called “cold start” which is characterised by the question:

*How will we make recommendations to new users?*

Netflix gives its new users a list to select some content that they like. I believe that from that moment on it makes the recommendations.



## Conclusion

In this post we saw how to implement a collaborative filter from end to end, from the beginnings of linear algebra, cosine similarity, recommendation system types to a “hello world” of the surprise library with KNN algorithm. It is highly recommended to explore the most out of the surprise library, test other estimators from the surprise lib, and the next step is to use deep learning to make our recommender algorithm, like Boltzmann Machines.

## References

- [1] C. Mathieu A. Das and D. Ricketts. Maximizing profit using recommender systems. WWW, 2010.
- [2] H. Anton. Elementary Linear Algebra. John Wiley Sons, Incorporated, 1993.
- [3] Z. Gantner H. S. C. Newell B. P. Knijnenburg, M. C. Willemsen. Explaining the user experience of recommender systems, user modeling and user-adapted interaction,. 2012.
- [4] T. Keenan. What is content-based filtering?  
<https://www.upwork.com/hiring/data/what-is-content-based-filtering/>. Last accessed 18 August 2019.
- [5] S. Luo. Introduction to recommender system.  
<https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26>, 2018. Last accessed 18 August 2019.
- [6] M. Nadeem. How youtube recommends videos.  
<https://towardsdatascience.com/how-youtube-recommends-videos-b6e003a5ab2f>, 2018. Last accessed 18 August 2019.
- [7] Netflix. How the Netflix recommendation system works.  
<https://help.netflix.com/pt/node/100639>. Last accessed 18 August 2019.
- [8] R. Sentance. How to boost conversion rates by 32%: a recommendation engine case study. <https://econsultancy.com/retail-case-study-personalisation-recommendation/>, 2018. Last accessed 18 August 2019.
- [9] Surprise. Surprise overview. <http://surpriselib.com/>. Last accessed 18 August 2019.

[10] Z. Tabaei and M. Fathian. Using customer lifetime value model for product recommendation: An electronic retailing case study. IJEEEE, 2012.

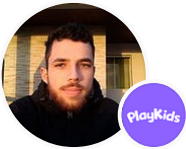
Data Science

Machine Learning

Recommender Systems

Collaborative Filtering

Content Based



Follow

## Written by Gabriel Ghellere

50 Followers · Writer for PlayKids Tech Blog

Intern Data Scientist at Playkids

---

### More from Gabriel Ghellere and PlayKids Tech Blog

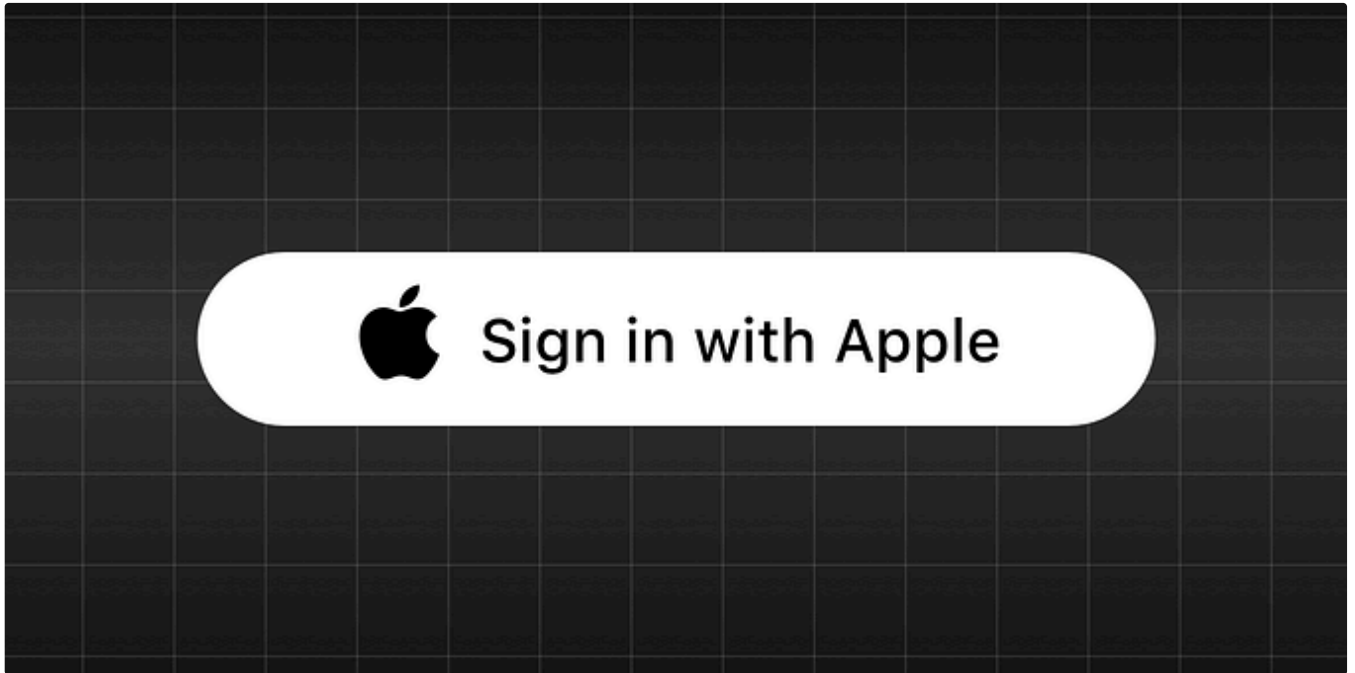



## Sistemas de recomendação com filtros colaborativos

Um dos tópicos de estudo que sempre me chamou atenção na área de Data Science são os sistemas de recomendação, tais sistemas são...

7 min read · Apr 15, 2019

 122  3



 Fellipe Callegas in PlayKids Tech Blog

## Implementing Sign in with Apple on the server-side


A guide for integration

6 min read · Jan 17, 2020

 232  5





 Charles Barros in PlayKids Tech Blog

## MatCap—Render & Art pipeline optimization for mobile devices

Mobile graphics optimization and why should you care about it.

8 min read · Nov 29, 2019



114



2



 Gabriel Ghellere in Introdução ao jupyter, pandas e pokemons.

## Como usar Data Science para estudar pokemons!

1 min read · Aug 15, 2018



64



1



---

See all from Gabriel Ghellere

See all from PlayKids Tech Blog

---

**Recommended from Medium**





Umair Iftikhar

## Part 3: Exploring Content-Based Filtering in Recommendation Systems

In the previous parts of our recommendation system series, we covered the fundamentals of item-item collaborative filtering and how to...

3 min read · Oct 26, 2023



65



Angel Das in Towards Data Science

# Exploring Recommendation Systems: Review of Matrix Factorization & Deep Learning Models

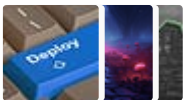
Summary of Recommender Systems (Alternate Least Square, LightFM, Matrix Factorization with Neural Networks, and Neural Collaborative...

★ · 9 min read · Nov 10, 2022

👤 210 🗨



## Lists



### Predictive Modeling w/ Python

20 stories · 1057 saves



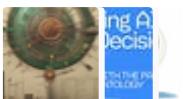
### Practical Guides to Machine Learning

10 stories · 1264 saves



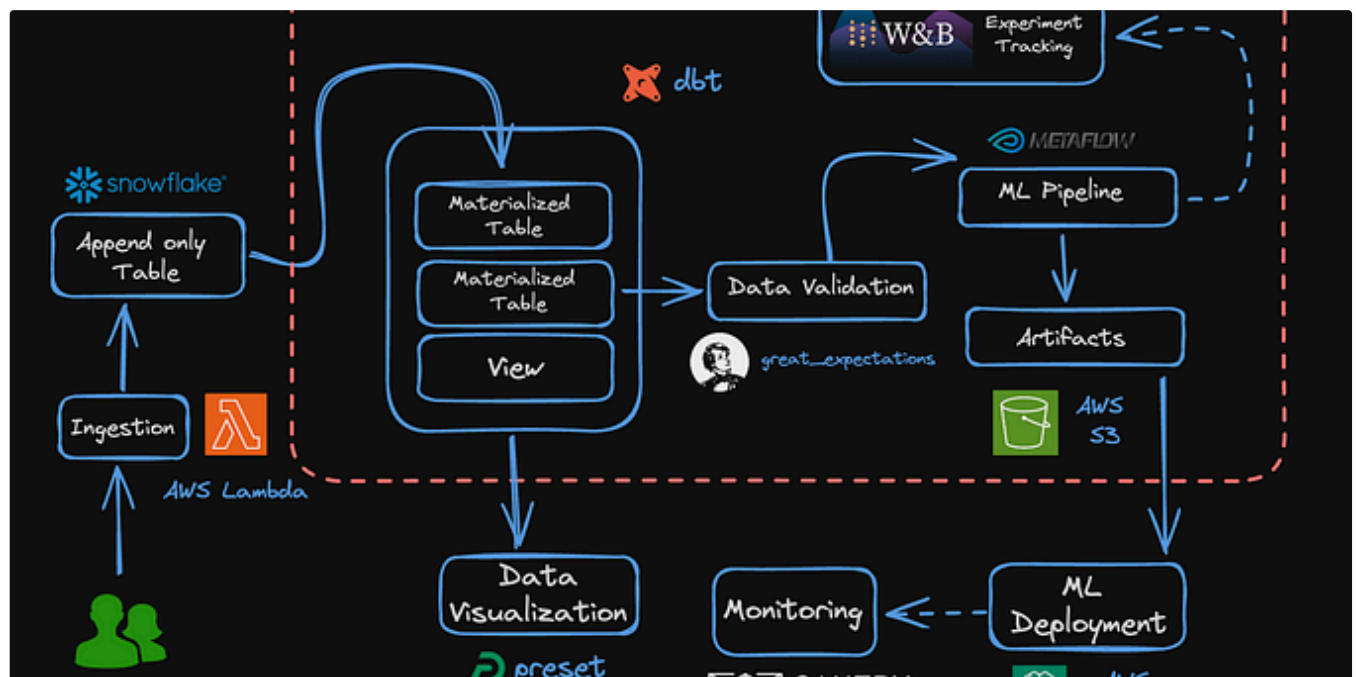
### Natural Language Processing

1344 stories · 827 saves



### data science and AI

40 stories · 122 saves



👤 Zain ul Abideen

## One-Stop Guide for Production Recommendation Systems

Candidate Generation, SSR for CG, ANN Search, Ranking, Evaluation Metrics, and Architectural paradigm of Real-time Recsys



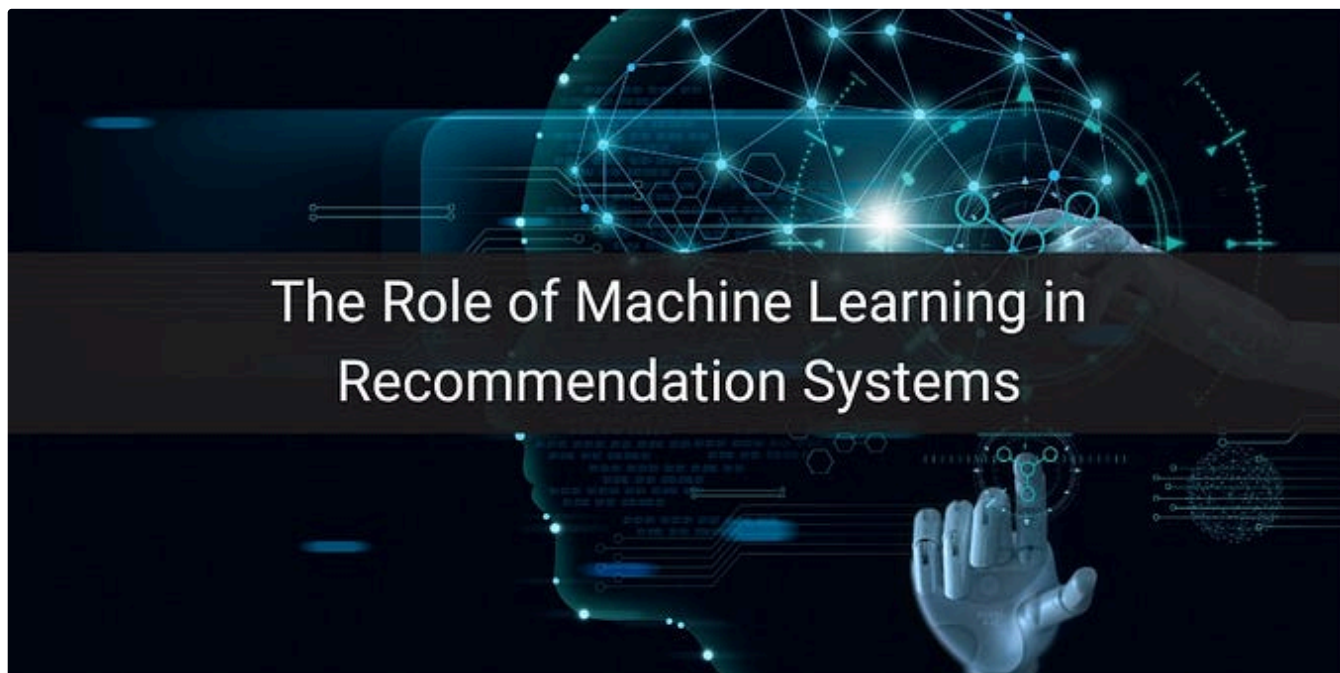
24 min read · Nov 21, 2023



271



2



## The Role of Machine Learning in Recommendation Systems



Om Belorkar

### The Magic of Machine Learning in Recommendation Systems

What is a recommendation system? Analogy.

13 min read · Oct 30, 2023

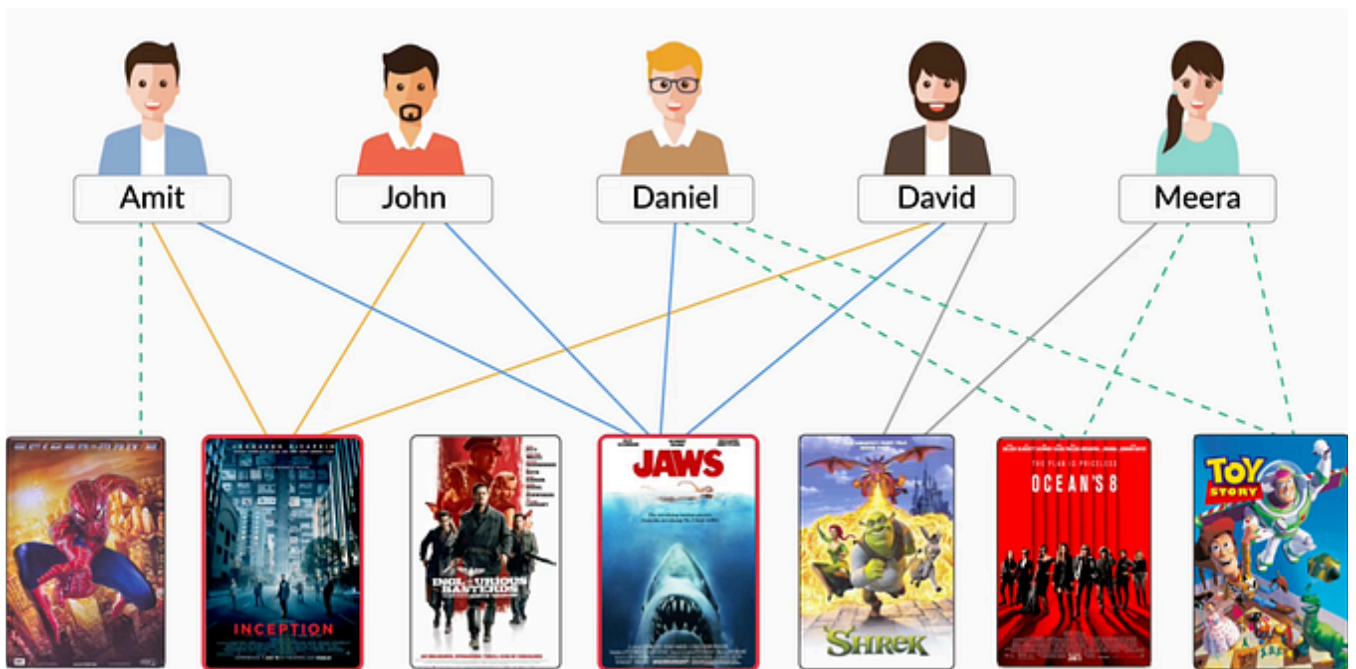


1.2K



5





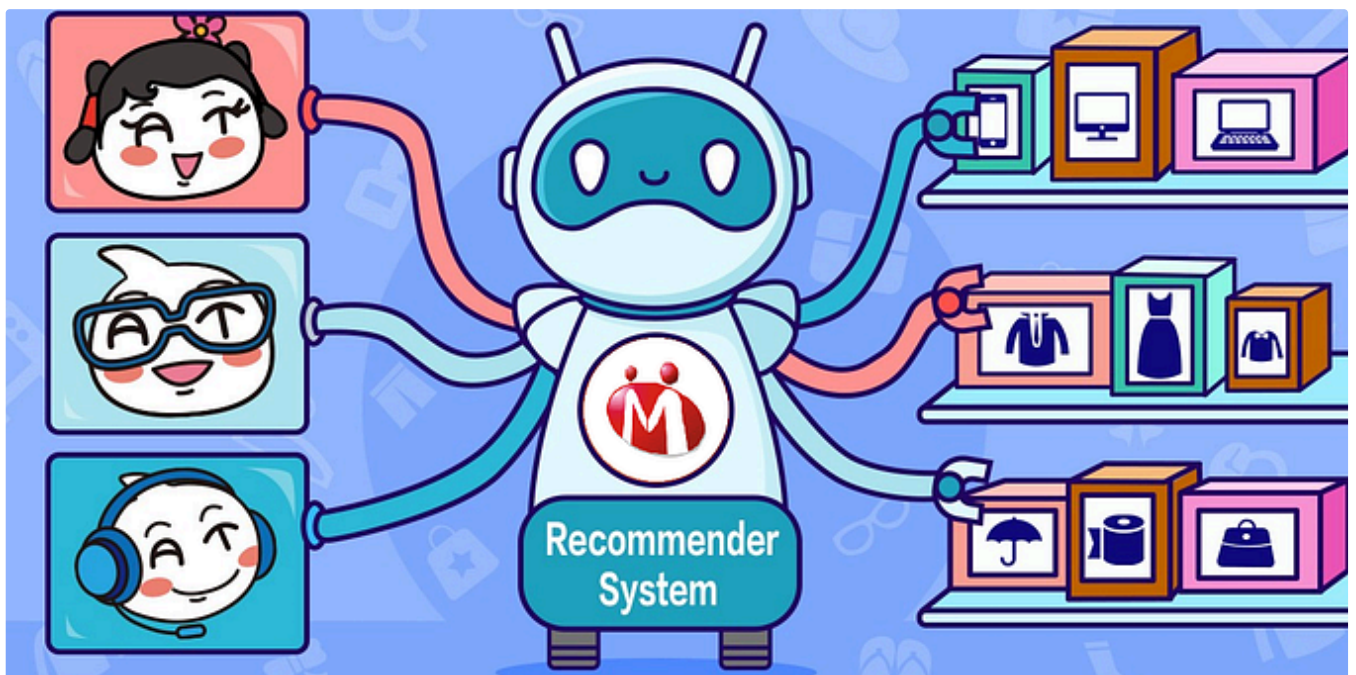
😊 Sharan Harsoor in AI Mind

## Recommendation Systems: An Overview

A recommendation system, also known as a recommender system or engine, is a type of software application or algorithm designed to provide...

25 min read · Nov 13, 2023

👏 118



👤 Merve ÖZKAN

## Recommender Systems

💡 Recommender systems are systems that work by recommending products or services to users using some techniques.

2 min read · Feb 22, 2024



See more recommendations