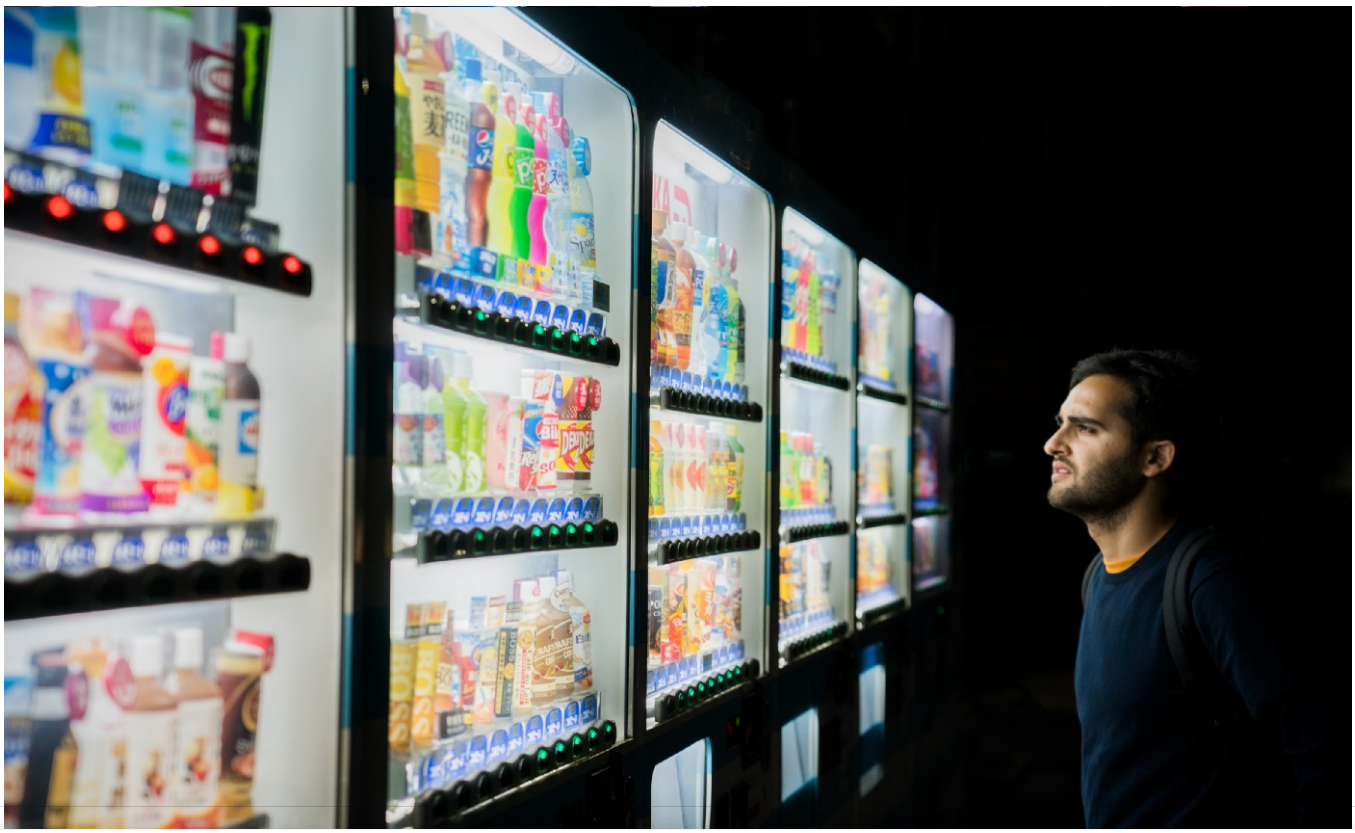
Published in Towards Data Science

Cornellius Yudha Wijaya Follow

Dec 11, 2021 · 5 min read ★ · Listen

Top 3 Python Packages to Learn the Recommendation System

Learn from these packages to up-skill your recommender system knowledge



A recommendation system is a data science problem that predicts what the user or customer wants based on the historical data. There are two common ways for recommendation systems to work — Collaborative Filtering and Content-Based Filtering.

Do you feel familiar with the terms above? Maybe yes, maybe not. In any case, this article aims to understand better the recommendation system using these top three Python packages. What are these packages? Let's get into it!

1. Surprise

Surprise is an open-source Python package for building a recommendation system based on the rating data. The name SurPRISE is an abbreviation for the *Simple Python RecommendatIon System Engine*. The package provides all the necessary tools for building the recommendation system — from loading the dataset, choosing the prediction algorithm, and evaluating the model.

Let's install the package to learn more about the recommendation system.

```
pip install scikit-surprise
```

After installing the package, let's try to build a recommendation system based on the tutorial package. We did this to see if the package had been properly installed or not.

```
from surprise import SVD
from surprise import Dataset
from surprise.model_selection import cross_validate

# Load the movielens-100k dataset (download it if needed).
data = Dataset.load_builtin('ml-100k')

# Use the famous SVD algorithm.
algo = SVD()
```



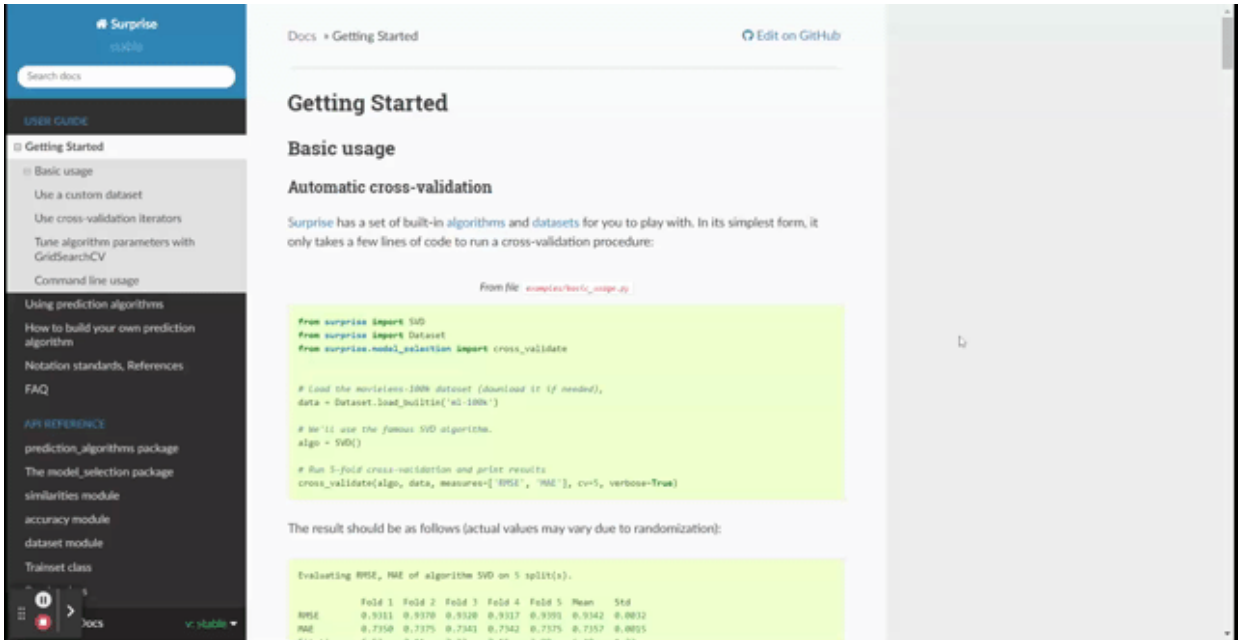
```
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5,
verbose=True)
```

```
                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9405  0.9353  0.9297  0.9349  0.9352  0.9351  0.0034
MAE (testset)     0.7393  0.7383  0.7308  0.7364  0.7400  0.7370  0.0033
Fit time          6.95    6.73    6.72    6.77    6.74    6.78    0.09
Test time         0.12    0.12    0.11    0.11    0.11    0.12    0.00

{'test_rmse': array([0.94051092, 0.93532844, 0.92965265, 0.93487384, 0.93516802]),
 'test_mae': array([0.73930083, 0.73834337, 0.73082354, 0.7363844 , 0.74004212]),
 'fit_time': (6.9523396492004395,
  6.7286131381988525,
  6.722811222076416,
  6.7664713859558105,
  6.742457389831543),
 'test_time': (0.12399792671203613,
  0.11500048637390137,
```


If you can see the output above, you are good to go. The code above analyzes the dataset by rating, and using the SVD algorithm; we create the recommendation system. The metrics we evaluated are RMSE and MAE with 5-fold as an evaluation method.

If we want to understand further how the algorithm and evaluation work, we could visit the Surprise documentation. The developer has stated that they strongly emphasized the documentation to explain every detail of the algorithm. That is why we would explore documentation for our material learning.



GIF by Author

If we look at the Getting Started part, the documentation is full of learning material on developing machine learning prediction for recommendation systems and evaluating it.

If we look at the prediction algorithm section, we will get a more detailed part about the estimation using baseline estimates and similarity measures. This part is pretty well written and lets you understand the basic algorithm used in the recommendation system.

Surprise provides a bunch of built-in algorithms. All algorithms derive from the `AlgoBase` base class, where are implemented some key methods (e.g. `predict`, `fit` and `test`). The list and details of the available prediction algorithms can be found in the `prediction_algorithms` package documentation.

Every algorithm is part of the global Surprise namespace, so you only need to import their names from the Surprise package, for example:

```
from surprise import KNNBasic
algo = KNNBasic()
```

Some of these algorithms may use baseline estimates, some may use a similarity measure. We will here review how to configure the way baselines and similarities are computed.

Baselines estimates configuration

Note

This section only applies to algorithms (or similarity measures) that try to minimize the following regularized squared error (or equivalent):

$$\sum_{r_{ui} \in R_{train}} \left(r_{ui} - (\mu + b_u + b_i)\right)^2 + \lambda \left(b_u^2 + b_i^2\right) .$$

Image by Author

The documentation also gives you an excellent selection for learning material; I suggest you could start from the following section to learn more about the model and the evaluation:

- prediction_algorithms package
- The model_selection package
- similarities module
- accuracy module
- dataset module

surprise.similarities.cosine()

Compute the cosine similarity between all pairs of users (or items).

Only **common** users (or items) are taken into account. The cosine similarity is defined as:

$$\text{cosine\_sim}(u, v) = \frac{\sum_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum_{i \in I_{uv}} r_{vi}^2}}$$

or

$$\text{cosine\_sim}(i, j) = \frac{\sum_{u \in U_{ij}} r_{ui} \cdot r_{uj}}{\sqrt{\sum_{u \in U_{ij}} r_{ui}^2} \cdot \sqrt{\sum_{u \in U_{ij}} r_{uj}^2}}$$

depending on the `user_based` field of `sim_options` (see Similarity measure configuration).

For details on cosine similarity, see on Wikipedia.

Image by Author

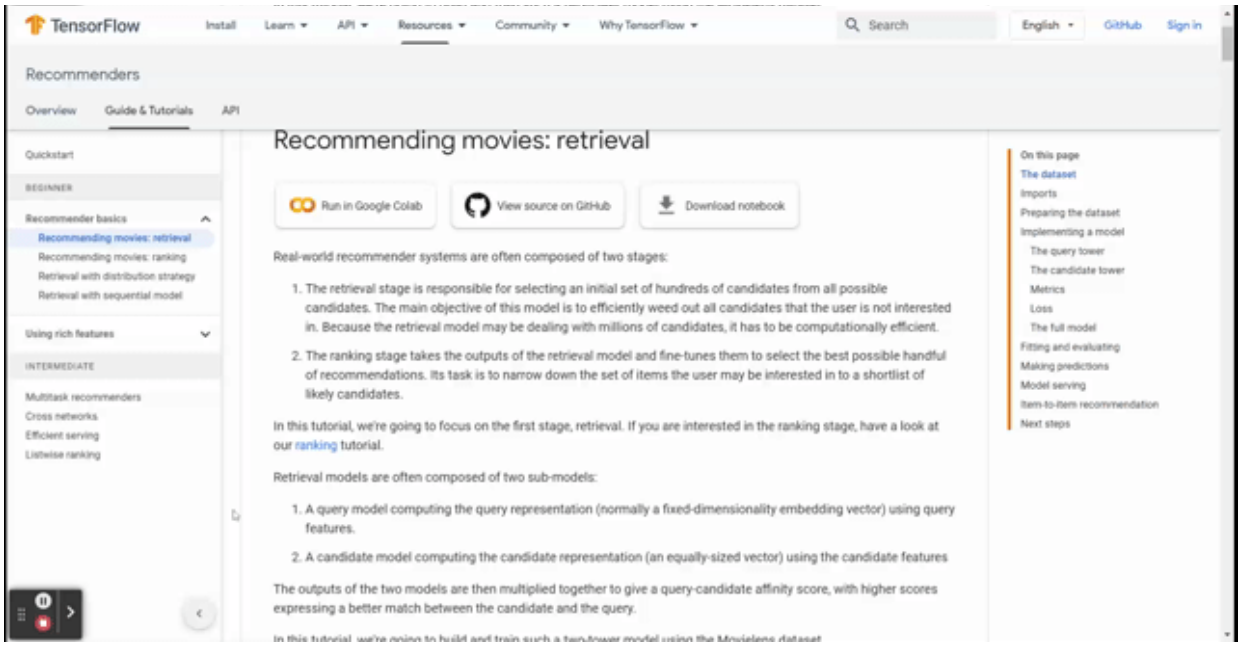
2. TensorFlow Recommenders

The TensorFlow framework contains a library to build the recommendation system called TensorFlow Recommenders. Like the other package, the TensorFlow Recommenders contains dataset examples, recommender algorithms, model evaluations, and deployment.

To Install the package, you need to run the following code.

```
pip install tensorflow-recommenders
```

TensorFlow Recommenders would allow us to build a recommendation system based only



GIF by Author

The documentation is structured from how you load the data, select the algorithm, and evaluate it.

Not only does the documentation contain the how-to-work for the beginner, but it also gives you tips on related content such as Feature Preprocessing.

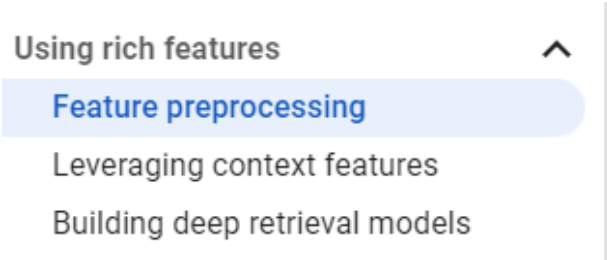


Image by Author

If you want to look at a more advanced use case, you could check out the Documentation for Intermediate cases. It shows you content such as Multitask recommenders, Cross networks, etc.

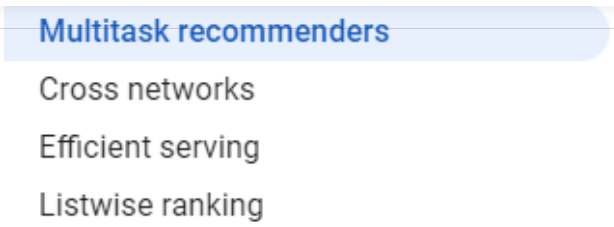


Image by Author

If you want to have the source code or contribute to the open-source, you could visit the GitHub page.

3. Recmetrics

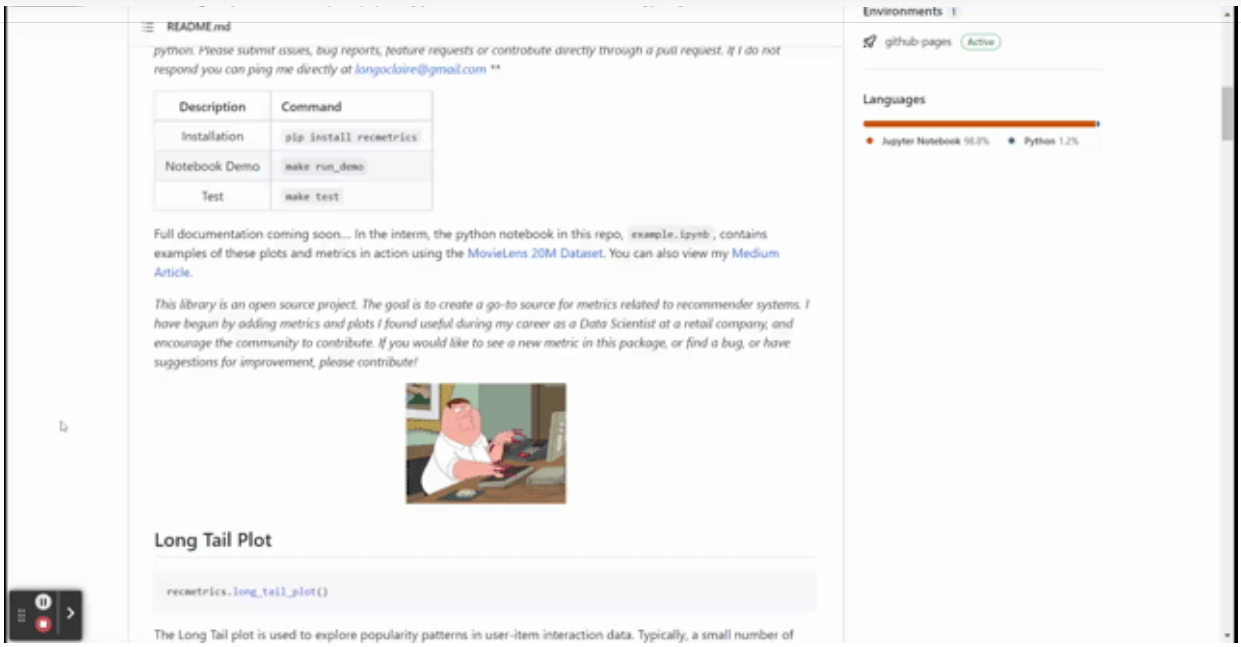
Learning about the recommendation system algorithm would not be complete without the evaluation metrics. The previous article I mentioned has taught us some basic recommendation evaluation metrics, but a Python package focuses on the metrics — Recmetrics.

To install the package, you only need to run the following code.

```
pip install recmetrics
```

The package contains many evaluation metrics for the recommendation system, such as:

- Long Tail Plot
- Coverage
- Novelty
- Personalize



GIF by Author

If you want to explore the package with an example dataset, you could explore the example notebook provided in the package.

Conclusion

A recommendation system is a data science problem to predict what the user or customers want based on the historical data. Learning recommendation system could be better with Python Package to accompany your studies. The package that I recommended are:

1. Surprise
2. TensorFlow Recommendation
3. Recmterics

I hope it helps!