

بهترین روش برای توسعه‌ی یک Recommender System، ابتدا توسعه‌ی متدهای تست و ارزیابی متریک هاست. در این گزارش، چگونگی ارزیابی سیستم پیشنهاد دهنده و توسعه‌ی تست‌ها را با هم مرور می‌کنیم.

شماره‌ی ۱ گزارش پیشرفت پروژه

پالایش مشارکتی سیستم پیشنهاد دهنده | ۱۴ فروردین ۱۴۰۰

فائزه سادات سعیدی نژاد

یکی از مهمترین مراحل ساخت یک سیستم پیشنهاد دهنده (و احتمالا هر سیستم و برنامه‌ی دیگری) نوشتن تست‌های آن بر اساس متریک‌های ارزیابی‌اش است. در ادامه متریک‌های ارزیابی سیستم پیشنهاد دهنده را نام برده و توضیح مختصری می‌دهم.

۱. پوشش‌دهی^۱: درصد پیشنهادهای امکان‌پذیر که سیستم پیشنهاد دهنده می‌تواند ارائه دهد.
۲. گوناگونی^۲: مقدار گستردگی سیستم پیشنهاد دهنده برای پیشنهاد دادن آیتم‌ها گوناگون، البته باید توجه داشته باشیم که آیتم‌ها را خیلی رندوم هم نباید پیشنهاد دهد.
۳. تازگی^۳: تازگی و محبوبیت آیتم‌های پیشنهادی توسط سیستم پیشنهاد دهنده.
۴. اعتماد کاربر^۴: این متریک که ارتباط زیادی با محبوبیت و تازگی دارد، به این معناست که آیتم‌هایی باید به کاربر پیشنهاد داده شود که به چشمش آشنا است و خیلی آیتم‌هایی نیست که کاربر تا به حال ندیده است، برای جلب کردن اعتماد کاربر. توجه داشته باشید که اگر novelty درست پیاده‌سازی شده باشد، بین آیتم‌هایی که به چشم کاربر آشنا هستند و آیتم‌هایی که تازگی دارند باید مصالحه^۵ وجود داشته باشد.
۵. ریزش^۶: هرچند وقت یک‌بار آیتم‌های پیشنهادی تغییر می‌کنند؛ آیا با هربار اکشن کاربر (امتیاز دادن، خرید کردن و...) سریعا آیتم‌های پیشنهادی تغییر می‌کنند؟
۶. واکنشی^۷: چقدر سریع آیتم‌های پیشنهادی می‌توانند بر اساس تغییر رفتار کاربر، تغییر کنند.

روش‌های تست سیستم پیشنهاد دهنده:

یکی از روش‌های تست که زیاد کاربردی ندارد این که با استفاده از نظرسنجی، مستقیما از کاربر بپرسیم که آیتم‌های پیشنهاد داده شده را دوست دارد یا خیر (Perceived Quality). روش رایج‌تر A/B Testing است. برای استفاده از این روش باید Top-N recommender را آماده کرد که به دو روش می‌توان این کار را انجام داد (فهمیدن علاقه‌مندی‌های کاربر): صریح^۸ و ضمنی^۹ (در مقاله‌ی اصلی در رابطه با این موضوع بیشتر توضیح خواهم داد).

از نمونه تست‌های دیگر می‌توان به متریک دقت^{۱۰} اشاره کرد که به دو صورت می‌توان آن را اندازه‌گیری کرد:

۱. خطای میانگین مطلق^{۱۱}: همانطور که از اسمش مشخص است، برای اندازه‌گیری‌اش، میانگین تفاضل نمره‌ای که سیستم پیشنهاد دهنده فکر می‌کند کاربر به یک آیتمی می‌دهد از نمره‌ای است که در حقیقت کاربر به آن آیتم داده است. فرمولش به صورت زیر است:

¹ Coverage

² Diversity

³ Novelty

⁴ User trust

⁵ Trade-off

⁶ Churn

⁷ Responsiveness

⁸ Explicit

⁹ Implicit

¹⁰ Accuracy metric

¹¹ Mean absolute error (MAE)

$$\frac{\sum_{i=1}^n |y_i - x_i|}{n}$$

مثال:

predicted rating	actual rating	error
5	3	2
4	1	3
5	4	1
1	1	0

$$MAE = (2+3+1+0)/4 = 1.5$$

۲. خطای جذر میانگین مربعات^{۱۲}: تفاوت میان مقدار پیش‌بینی شده توسط سیستم پیشنهاد دهنده و مقدار واقعی می‌باشد. RMSD یک ابزار خوبی است برای مقایسه خطاهای پیش‌بینی توسط یک مجموعه داده‌است و برای مقایسه چند مجموعه داده کاربرد ندارد. فرمولش بصورت زیر است:

$$\sqrt{\frac{\sum_{i=1}^n (y_i - x_i)^2}{n}}$$

مثال:

predicted rating	actual rating	error ²
5	3	4
4	1	9
5	4	1
1	1	0

$$RMSE = \sqrt{(4 + 9 + 1 + 0)/4} = 1.87$$

توجه کنید که هر دو خطا هستند، پس هر چه مقدارشان کمتر باشد، بهتر است.

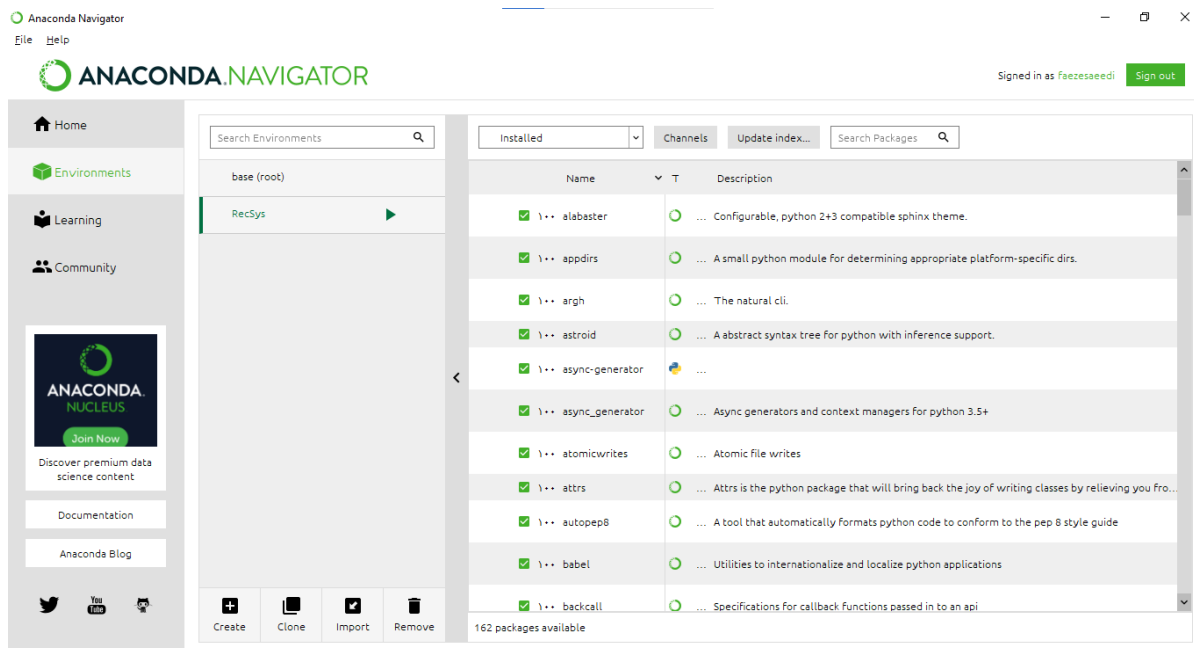
از روش‌های دیگر می‌توان به Hit rate اشاره کرد که به چندگونه می‌توان آن را انجام داد، اعتبار سنجی یک طرفه^{۱۳}، average rating hit rate، cumulative hit rate، reciprocal hit rate.

¹² Root mean square error (RMSE)

¹³ Leave-one-out cross validation

قسمت عملی:

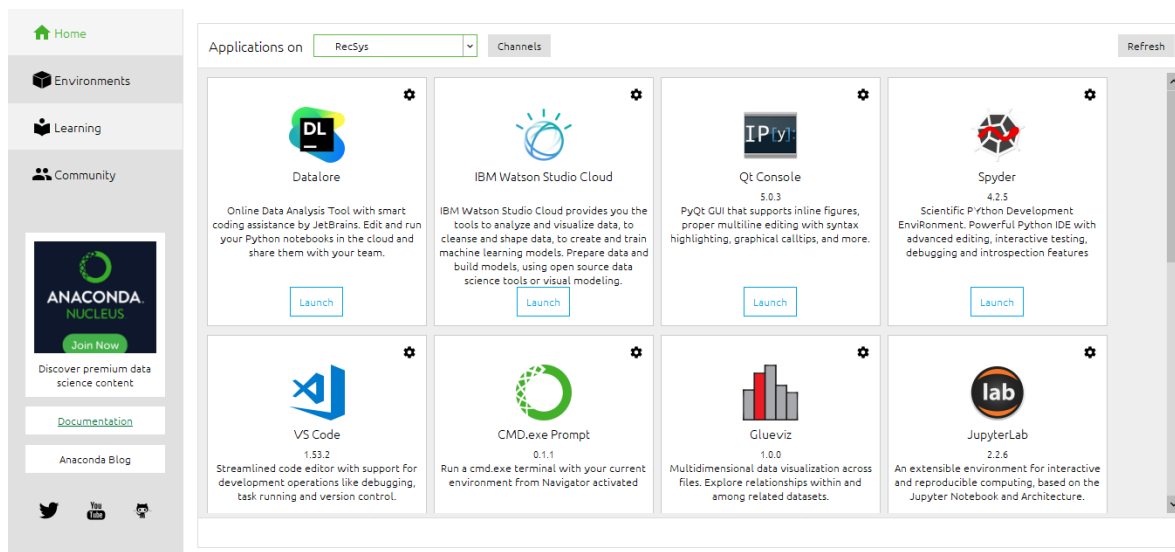
برای نوشتن تست‌ها از نرم‌افزار Spyder در Anaconda استفاده می‌کنیم. ابتدا Anaconda را باز می‌کنیم، سپس یک environment می‌سازیم به نام RecSys:



حال در ترمینال این محیط، کتابخانه‌ی Surprise را که برای استفاده از تست‌های از قبل نوشته شده‌ی سیستم پیشنهاد دهنده است، با استفاده از دستور زیر نصب می‌کنیم:

Conda install –c conda-forge scikit-surprise

سپس به صفحه‌ی Home رفته و Application on Spyder را بروی RecSys قرار داده و برنامه‌ی Spyder را باز می‌کنیم.



بعد از باز شدن Spyder کدهای Evaluating را که باز کرده و فایل TestMetrics.py را اجرا می‌کنیم، در خروجی نتیجه‌ی ارزیابی متریک‌های ذکر شده را می‌بینیم.

```
Console 1/A x
Loading movie ratings...

Computing movie popularity ranks so we can measure novelty later...

Computing item similarities so we can measure diversity later...
Estimating biases using als...
Computing the pearson_baseline similarity matrix...
Done computing similarity matrix.

Building recommendation model...

Computing recommendations...

Evaluating accuracy of model...
RMSE: 0.9033701087151801
MAE: 0.6977882196132263

Evaluating top-10 recommendations...
Computing recommendations with leave-one-out...
Predict ratings for left-out set...
Predict all missing ratings...
Compute top 10 recs per user...

Hit Rate: 0.029806259314456036

rHR (Hit Rate by Rating value):
3.5 0.017241379310344827
4.0 0.0425531914893617
4.5 0.020833333333333332
5.0 0.06802721088435375

chr (Cumulative Hit Rate, rating >= 4): 0.04960835509138381
```

```
Console 1/A x
chr (Cumulative Hit Rate, rating >= 4): 0.04960835509138381

ARHR (Average Reciprocal Hit Rank): 0.0111560570576964

Computing complete recommendations, no hold outs...

User coverage: 0.9552906110283159
Computing the pearson_baseline similarity matrix...
Done computing similarity matrix.

Diversity: 0.9665208258150911

Novelty (average popularity rank): 491.5767777960256

In [2]: |
```

در این پروژه، از داده‌ی سایت [Movie lens](#) استفاده شده. کد تست‌ها و فایل دیتاست در ادامه‌ی فایل‌های این گزارش آمده است.