# پیاده سازی سیستم‌های پیشنهاد دهنده[1]

*گزارش نهایی پروژه درس مبانی فناوری اطلاعات*


**اعضای گروه :**

**بهار برومند**

**سیدعلی امیرشاهکرمی**

**علی راد**

**غزاله زهتاب[2]**

**استاد:**

**دکتر محمد حسین منشئی**

**خانم معصومه زارع**

بهاروتابستان1399

گروه8

دانشگاه صنعتی اصفهان


[1] Recommender Systems
[2] Team leader

**مقدمه** 3
**مراحل** 3
**فاز اول: مقایسه پکیج ها** 4
**فاز دوم: پیاده سازی** 5
مجموعه داده‌ها 5
1.رستوران 5
2.لباس 5
3.کتاب 5
پکیج ها 6
پکیج :Case recommender 6
معرفی: 6
الگوریتم های پکیج: 6
طرز کار پکیج: 7
مجموعه داده رستوران: 7
مجموعه داده لباس: 9
مجموعه داده کتاب: 10
پکیج :Apache Mahout 11
معرفی: 11
الگوریتم های این پکیج: 11
داده مدل در : Apache Mahout 12
معیار شباهت و معیار همسایگی در :Apache Mahout 12
ارزیابی سیستم پیشنهاد دهنده پیاده سازی شده: 13
مجموعه داده رستوران: 13
مجموعه داده لباس: 15
مجموعه داده کتاب: 16
پکیج :LightFM 17
معرفی: 17
مدل های استفاده شده در این پروژه: 18
مجموعه داده رستوران: 19
دقت مدل نهایی و ماتریس های ساخته شده: 19
اجرا نهایی و پیشنهادات برای مشتری :U1011 19
مجموعه داده لباس: 20
مقایسه adagrad و adadelta در :warp loss 20
مقایسه BPR و :WARP 21
مقایسه K_OS WARP و :WARP 21

دقت مدل نهایی و ماتریس های ساخته شده: 22
اجرا نهایی و پیشنهادات برای مشتری 999231: 22
مجموعه داده کتاب: 22
مقایسه adagrad و adadelta در warp loss: 23
مقایسه BPR و WARP : 24
مقایسه K_OS WARP و WARP: 25
دقت مدل نهایی و ماتریس های ساخته شده: 25
اجرا نهایی و پیشنهادات برای مشتری 178622 با آستانه‌ی نویسنده کتاب: 26
اجرا نهایی و پیشنهادات برای مشتری 178622 با آستانه‌ی rating کتاب بیشتر از 5: 26
پکیج Surprise Recommender: 27
اهداف طراحی: 27
الگوریتم های قابل استفاده: 27
پکیج های SciKit: 28
تکنیک پیاده سازی سیستم توصیه گر در این انجین: 28

**نتیجه** **36**

**لینک کدهای پکیج هریک از اعضای تیم** **37**
علی راد(Case Recommender) 37
بهار برومند(Apache Mahout) 37
غزاله زمتاب(LightFM) 37
سیدعلی امیرشاهکرمی(Surprise Recommender) 37

**منابع** **37**

# مقدمه

با توجه به، محبوبیت زیاد سیستم های پیشنهاد دهنده در طی سال های اخیر، پکیج های متفاوتی با زبان های برنامه نویسی، الگوریتم ها و ویژگی های متفاوت ارائه شده است. در این پروژه به بررسی سیستم های پیشنهاد دهنده، الگوریتم های آن ها، پکیج های متفاوت و ویژگی های آن ها و در نهایت پیاده سازی سیستم پیشنهاد دهنده و اجرای آن بر روی مجموعه داده های متفاوت پرداخته شده است.

در این گزارش ابتدا مراحل پروژه توضیح داده شده است؛ سپس توضیح کلی در رابطه با رویکرد تیم در فاز اول پروژه ارائه شده است و در نهایت به جزئیات فاز دوم، بررسی مجموعه داده ها، پکیج های مخصوص این فاز و نحوه ی پیاده سازی سیستم پیشنهاد دهنده با هریک از پکیج ها پرداخته شده است.

# مراحل

پروژه ی مربوط به گروه ما به دو فاز تقسیم می شد:

- در فاز اول به توضیحات کلی در مورد چندین پکیج پرداختیم و ویژگی ها و الگوریتم های آن ها را به صورت کلی توضیح دادیم.
- در فاز دوم چهار پکیج از بین پکیج های فاز اول را انتخاب کردیم و به بررسی بیشتر این پکیج ها پرداختیم. در این فاز تمرکز اصلی ما بر روی پیاده سازی یک سیستم پیشنهاد دهنده با استفاده از هر یک از این پکیج ها و اجرای آن ها بر روی مجموعه داده های مورد نظر بود.

# فاز اول: مقایسه پکیج ها

در فاز اول پروژه ابتدا درباره سیستم های توصیه گر مطالبی را جمع آوری کردیم و با توجه به منابع معرفی شده به مطالعه کتاب سیستم های توصیه گر کاربردی نوشته آقای کیم فالک پرداختیم. این کتاب که اولین ویرایش آن در سال 2019 منتشر شده به آموزش مفاهیم و روش های پیاده سازی سیستم های توصیه گر می پردازد. ما ابتدا بخش هایی از کتاب را با هدف آشنایی بیشتر با سیستم های توصیه گر مطالعه کردیم.سپس در فاز اول پروژه گزارشی از 8 پکیج متفاوت تهیه کردیم که در آن توضیحاتی در مورد ویژگی هر یک از پکیج ها، الگوریتم های مورد استفاده ی آن ها و کاربرد آن ها ارائه دادیم. پکیج های بررسی شده در فاز اول موارد زیر هستند:

- پکیج The Universal Recommender
- پکیج PredictionIO
- پکیج HapiGER
- پکیج Apache Mahout
- پکیج Seldon
- پکیج Oryx
- پکیج LensKit
- پکیج LightFM
- پکیج Case Recommender
- پکیج Surprise

همچنین در قسمت بعدی فاز یک به توضیح در مورد الگوریتم های مهم مورد استفاده در پکیج های بالا پرداختیم که موارد زیر هستند:

- الگوریتم Collaborative Filtering
- الگوریتم Content-Based Filtering
- الگوریتم Correlated Cross-Occurrence

سپس سعی کردیم با بررسی ویژگی های مهم این پکیج ها، آنها را از جنبه های مختلف مانند کاربرد و دقت با یکدیگر مقایسه کنیم. با بررسی های انجام شده و مقایسه پکیج ها تصمیم گرفتیم چهار پکیج معروف و مهم را انتخاب کنیم. بنابراین پکیج های lightFM و Apache Mahout و Case Recommender و Surprise را برای استفاده در فاز دوم انتخاب کردیم.

# فاز دوم: پیاده سازی

## مجموعه داده‌ها[3]

### 1.رستوران[4]

این مجموعه با هدف پیشنهاد دادن رستوران به مشتری جدید در این پروژه استفاده شده است. فایل های مورد استفاده از این مجموعه داده 3 فایل با پسوند csv است. فایل اصلی شامل اطلاعات رستوران ها و مشتری ها و 3 نوع rate است که rate که ما در این پروژه استفاده کردیم rate نهایی است که یک مشتری به رستوران داده است. دو فایل دیگر که در بعضی از پکیج ها از آن استفاده شده یکی شامل اطلاعات رستوران ها و دیگری پروفایل مشتری ها است. هدف نهایی پروژه برای این مجموعه داده گرفتن چندین پیشنهاد برای مشتری U1011 است. در بین همه مجموعه داده ها این مجموعه با توجه به تراکم و اطلاعات کافی و قابل قبولی که دارد بهترین مجموعه ما محسوب می‌شود.

### 2.لباس[5]

هدف از این مجموعه داده در این پروژه پیشنهاد دادن لباس به یک مشتری و در صورت امکان سایز پیشنهادی برای آن لباس و آن مشتری است تا بتوانیم مرجوعی های کالا را کاهش دهیم. این مجموعه داده حاوی دو فایل مجزا با پسوند json از دو سایت اجاره لباس است. در این پروژه از اطلاعات جمع آوری شده از سایتی استفاده شده که اطلاعات کامل تری از جمله لباس، مشتری، سایز مشتری و ارزیابی او قرار دارد.
این مجموعه شامل اطلاعات بسیار زیادی است(سطرهای زیاد) که انجام و تحلیل داده را برای لپ تاپ ناممکن می‌کند. برای حل این مشکل 10000 سطر از این داده را به صورت تصادفی انتخاب کردیم و همه از این فایل جدید برای پیشنهاد دادن استفاده کرده ایم. نکته حائز اهمیت در این کار این است که انتخاب کردن داده ها میتواند از دقت سیستم توصیه گر را کم کند و به زمان بیشتری برای یادگیری نیاز داشته باشد.

### 3.کتاب

این مجموعه با هدف پیشنهاد دادن کتاب جدید به یک مشتری از قبل حاضر در سیستم است. اطلاعاتی که در این مجموعه داده در فایل نهایی مورد استفاده از در کنار هم قرار دادن چندین فایل به دست آمده و به دلیل این که حجم داده به نسبت نوع آزمایشی و سیستم موجود زیاد است ، 10000 سطر از اطلاعات به صورت تصادفی برای پیشبرد پروژه استفاده شده است. این مجموعه داده شامل کاربر، کتاب ، نویسنده کتاب و میزان رضایت کاربر از کتاب در یک بازه 1 تا 10 است.

---

[3] datasets
[4] https://www.kaggle.com/uciml/restaurant-data-with-consumer-ratings/metadata
[5] https://www.kaggle.com/rmisra/clothing-fit-dataset-for-size-recommendation/metadata

# پکیج ها

## پکیج Case recommender:

### معرفی:

این پکیج یک فریم ورک تحت پایتون است که معروف ترین الگوریتم های Recommendation را دارد و شامل ابزار هایی برای ساختن یک Recommender System سفارشی با الگوریتم های پکیج است.

این فریمورک انواع مختلف شیوه های پیاده سازی Item Recommendation , Rating Prediction است و همچنین دارای ابزار هایی برای اعتبار سنجی نتیجه الگوریتم ها میباشد.

برای Item Recommendation که موضوع اصلی پروژه است , این پکیج مجموعه داده را به عنوان ورودی دریافت میکند و پس از انجام Train و Prediction یک خروجی در قالب فایل شامل تمام یوزر ها و ۱۰ آیتم پیشنهاد شده به آن یوزر را به ترتیب نمایش میدهد که شرح آن مفصل تر در زیر گفته میشود.

### الگوریتم های پکیج:

الگوریتم های مربوط به Item Recommendation:

- الگوریتم BPRMF
- الگوریتم ItemKNN
- الگوریتم Item Attribute KNN
- الگوریتم UserKNN
- الگوریتم User Attribute KNN
- الگوریتم( Group-based(Clustering-based algorithm
- الگوریتم( Paco Recommender (Co-Clustering-based algorithm
- الگوریتم Most Popular
- الگوریتم Random
- الگوریتم Content Based

الگوریتم های مربوط به Rating Prediction:

- روش( Matrix Factorization (with and without baseline
- روش Non-negative Matrix Factorization
- روش SVD
- روش ItemKNN

- روش Item Attribute KNN
- روش UserKNN
- روش User Attribute KNN
- روش( Item NSVD1 (with and without Batch
- روش( User NSVD1 (with and without Batch
- روش Most Popular
- روش Random
- روش gSVD++
- روش Item-MSMF
- روش CoRec (E )

الگوریتم های مربوط به Clustering:

- الگوریتم PaCo: EntroPy Anomalies in Co-Clustering
- الگوریتم K-medoids

## طرز کار پکیج:

این پکیج یک مجموعه داده به عنوان ورودی میگیرد که باید شامل حداقل سه سطر به ترتیب باشد:
User_id,item_id,feedback

این فیدبک میتواند همان rating باشد که یوزر به آیتم داده است.پس از آن با اجرای تابع compute از کلاس الگوریتم مورد نظر که ورودی آن مجموعه داده بالا میباشد، پکیج ابتدا بر روی داده ها عمل training را انجام میدهد و سپس به ازای همه یوزر های تعریف شده و آیتم های تعریف شده که حداقل یک بار در مجموعه داده آمده باشد ، عمل زیر را انجام میدهد:

- اگر Item Recommendation مدنظر باشد ، به ترتیب برای هر یوزر ، ۱۰ آیتم از بیشترین اولویت تا کمترین اولویت را مشخص میکند و در فایل خروجی که به عنوان ورودی به کلاس الگوریتم دادیم ذخیره میکند.
- اگر Rating Prediction مدنظر باشد ، باید به عنوان ورودی یک فایل test به کلاس بدهیم که در آن بازخورد مورد نظر هر یوزر برای آیتم مورد نظر که میخواهیم توسط پکیج محاسبه شود به همان ترتیب ستون های مجموعه داده قرار دارد و در فایل خروجی برای یوزر و آیتم های داده شده ، یک بازخورد پیش بینی میکند.

## مجموعه داده رستوران:

فایل data.dat به صورت زیر است :

```
1009,132834,2
1084,135085,2
1094,132830,0
1107,132660,2
1072,135088,0
1057,135080,1
1045,135051,2
1033,135030,2
1033,132572,1
1126,135108,2
1027,135062,1
1073,132951,0
1099,135066,1
1002,132921,2
```

که در آن به ترتیب یوزر ، آیتم و rating رو با ',' جدا کردیم ، الگوریتم رو به صورت زیر اجرا میکنیم که در آن فایل خروجی فایل item_out.csv می باشد.

```
knn = ItemKNN('data.dat', sep=",", output_file="item_out.csv").compute()
```

و خروجی که در این فایل ذخیره میشود به صورت زیر می باشد:

```
1001,135063,0.826938
1002,132834,1.718788
1002,135025,1.548526
1002,132922,1.438199
1002,135051,1.409027
1002,135027,1.295143
1002,132754,1.268060
1002,135060,1.208082
1002,135030,1.180720
1002,135038,1.155795
1002,135079,1.037619
1003,135085,1.734597
1003,132834,1.675409
1003,135064,1.338052
```

در این خروجی به ترتیب یوزر ، آیتم و یک مقیاس به عنوان اولویت قرار داده میشود که مقیاس ها به صورت نزولی مرتب میشوند و میتوان از آن به عنوان آیتم پیشنهادی به هر یوزر استفاده کرد.

و به عنوان خروجی Recommendation برای مجموعه داده رستوران در این پکیج تصویر زیر را داریم :

```
train data:: 138 users and 130 items (1161 interactions) | sparsity:: 93.53%
training_time:: 0.013187 sec
prediction_time:: 0.144704 sec


1011
Carreton de Flautas y Migadas
TACOS EL GUERO
Little Cesarz
Hamburguesas La perica
carnitas mata calle Emilio Portes Gil
la perica hamburguesa
TACOS CORRECAMINOS
puesto de gorditas
cafe ambar
Cafe Chaires

Process finished with exit code 0
```

در تصویر بالا ابتدا یک گزارش کلی از تعداد یوزر ها و آیتم ها ، تایم train و prediction قرار دارد.

با تابعی که ورودی آن آیدی یوزر است ، آیتم های پیشنهادی یوزر(در اینجا ۱۰۱۱) رو از لیست خروجی استخراج می کنیم و چاپ می کنیم.

## مجموعه داده لباس:

در این مجموعه داده مانند مجموعه داده رستوران، ورودی ها و خروجی الگوریتم را آماده میکنیم و برای مثال برای یوزر ۴۵۴۳۱۳ کد را اجرا میکنیم (تمامی مراحل مانند مجموعه داده رستوران است).

خروجی زیر را مشاهده میکنیم که هر کدام id یک لباس است که به کاربر پیشنهاد شده است:

```
454313
123373
123793
124204
124553
125424
125465
125564
126335
127081
127495

Process finished with exit code 0
```

دقت شود به دلیل زمان بردن زیاد training و prediction از نتایج خروجی قبلی الگوریتم برای پیشنهاد دادن لباس استفاده شده است.

بخشی از خروجی الگوریتم را در زیر مشاهده میکنید:

```
105608,782079,0.178458
105608,2157942,0.178458
105608,2904743,0.103033
105608,2647046,0.073453
105608,538143,0.066838
105608,1463543,0.055659
105608,403038,0.049456
```

## مجموعه داده کتاب:

همانند دو مجموعه داده قبل، ورودی ها و فایل خروجی را به کلاس الگوریتم می دهیم و مثلا برای یوزر 202000 خروجی را در زیر مشاهده می کنیم :

```
202000
Nigel Mansell My Autobiography
Acts of Betrayal
Miss Wyoming Uk Edition
One for My Baby
Mythology (Collins Gem)
Mythology (Collins Gem)
Postcards
Information
Cocaine Nights
253
Swimmer

Process finished with exit code 0
```

دقت شود مانند مجموعه داده قبلی به دلیل تعداد زیاد یوزر ها و آیتم ها و زمان بردن تایم train و prediction از این عمل مجدد صرف نظر شده (تابع compute الگوریتم کامنت شده) و از نتایج خروجی قبلی الگوریتم استفاده می شود.

## پکیج Apache Mahout:

### معرفی:

این پکیج یک کتابخانه ی زبان جاوا است. با استفاده از این پکیج تکنیک های متفاوتی را میتوان پیاده سازی کرد مانند Recommendation Classification،Clustering و ... که در این پروژه تمرکز اصلی ما بر روی مبحث سیستم های پیشنهاد دهنده است.

پکیج Apache Mahout با توجه به Scalable بودن آن محبوبیت و معروفیت بسیاری دارد و به همین دلیل به بررسی آن پرداختم.

در این پکیج با دریافت رفتار کاربران و استفاده از الگوریتم های مناسب، کالایی

مناسبی به کاربر پیشنهاد می دهیم.

ماهوت از داده هایی مانند clicks ، rating برای محاسبه ی پیشنهاد مناسب استفاده میکند.

برای هر مجموعه داده مراحل زیر باید انجام شود:

- داده ی خام باید به داده مدل مناسب و قابل فهم برای Apache Mahout تبدیل شود.
- معیاری که بر اساس آن شباهت بین کاربران یا کالا ها تعیین میشود، باید مشخص شود.
- با استفاده از الگوریتم ها و method های این پکیج، پیشنهاد مناسب محاسبه شود.
- روش و نحوه ی محاسبه ی پیشنهادات و در واقع سیستم پیشنهاد دهنده ی پیاده سازی شده ارزیابی شود.

### الگوریتم های این پکیج:

در این پکیج برای سیستم پیشنهاد دهنده میتوان از الگوریتم های متفاوت از جمله Collaborative Filtering در دو حالت، User-based و Item-based استفاده کرد.استفاده از حالت User-based به این معناست که الگوریتم با استفاده از شباهت بین کاربران کالای مورد نظر را پیدا میکند و در حالت Item-based شباهت بین کالاها محاسبه می شود. معمولا کالا ها تغییراتی سریع ندارند و می توان این شباهت میان کالاها را به صورت offline محاسبه کرد در صورتی که user ها ذات dynamic دارند. به همین علت scale کردن الگوریتم در حالت User-based سخت تر است.

همچنین در این پکیج امکان استفاده از الگوریتم SlopeOne Recommenders نیز وجود دارد. نکته ی مهم در مورد این پکیج این است که الگوریتم های Content Based در این پکیج به صورت مستقیم وجود ندارند و برای پیاده سازی آن ها باید از روش های دیگری استفاده کرد.

الگوریتم استفاده شده در این پروژه Collaborative Filtering از نوع User-Based است.

## داده مدل در Apache Mahout :

همانطور که در قسمت قبل اشاره شد، با استفاده از method های مخصوص به این پکیج، دادههای خام را باید به data model های قابل فهم برای Apache Mahout تبدیل کنیم تا الگوریتم های مورد نظر بر روی آن ها اجرا شود.

داده ی خام اولیه می تواند از نوع فایل، پایگاه داده های ریلیشنال مانند MySql و یا غیر ریلیشنال مانند MongoDB و همچنین داده های Generic باشد.

داده ی خام اولیه استفاده شده در این پروژه از نوع فایل و در قالب CSV است که با استفاده از کلاس datamodel در Apache Mahout به مدل قابل فهم برای پکیج تبدیل شده است.

## معیار شباهت و معیار همسایگی در Apache Mahout:

معیار شباهت تعیین کننده ی روشی است که با استفاده از آن میزان شباهت بین دو کاربر یا کالا را می سنجیم. در این پکیج از روش های متفاوتی می توان شباهت بین کاربران(یا کالا ها) را به دست آورد و برای مقایسه استفاده کرد. از جمله ی آن ها می توان به موارد زیر اشاره کرد:

- روش Pearson Correlation
- روش CityBlock Similarity
- روش Euclidean Distance
- روش Tanimoto Coefficient
- روش LogLikelihood Similarity

در واقع روش های بالا از قبل درون این پکیج پیاده سازی شده اند و با استفاده از کلاس ها و Method های مخصوص می توان یکی از این روش ها را انتخاب کرد و شباهت بین کاربران را بر آن اساس به دست آورد.

در این پروژه دو روش Pearson Correlation و CityBlock Similarity برای مجموعه داده ها تست شده است.

برای معیار همسایگی دو روش از پیش پیاده سازی شده در این پکیج وجود دارد:

- روش Nearest N users: پس از محاسبه ی شباهت هر کاربر با کاربر مورد نظر(با استفاده از معیار شباهت) N کاربر شبیه تر به کاربر مورد نظر ما به عنوان همسایه ی آن شناخته می شوند.
- روش Thresholds : پس از محاسبه ی شباهت هر کاربر با کاربر مورد نظر(با استفاده از معیار شباهت)، کاربرانی که شباهتشان از یک حدی[6] بیشتر است در حوزه ی همسایگان کاربر اصلی شناخته می شوند.

[6] threshold

هر دو معیار همسایگی در این پروژه استفاده شده است که در ادامه به بررسی آن می پردازم.

## ارزیابی[7] سیستم پیشنهاد دهنده پیاده سازی شده:

یکی از مهم ترین مراحل پیاده سازی یک سیستم پیشنهاد دهنده، ارزیابی آن سیستم است.در پکیج Apache Mahout کلاس های مخصوصی برای ارزیابی سیستم پیاده شده وجود دارد. روش های متفاوتی برای ارزیابی سیستم استفاده می شود، که دو روش Mean Average Error و Root Mean Square Error روش های ماهوت برای ارزیابی سیستم هستند.

در این پروژه برای ارزیابی سیستم از روش Mean Average Error استفاده شده است.

## مجموعه داده رستوران[8]:

برای استفاده از این مجموعه داده ابتدا باید تغییراتی بر روی فایل دانلود شده از سایت kaggle اعمال میکردم که این تغییرات را با استفاده از کد پایتون و کتابخانه ی pandas انجام دادم. در واقع باید ستون ها و داده های درون آن ها را برای Mahout قابل فهم میکردم که در تبدیل آن به datamodel مناسب دچار مشکل نشود.

در این مجموعه داده برای مشتری با شماره U1011 از معیار شباهت CityBlock Similarity و مرز همسایگی[9] 0.1 استفاده کردم.

```
//known as Manhattan distance
CityBlockSimilarity similarity = new CityBlockSimilarity(model);
//Computes a neighborhood consisting of all users whose similarity to the given user meets or e
// Similarity is defined by the given UserSimilarity.
UserNeighborhood neighborhood = new ThresholdUserNeighborhood( threshold: 0.1,similarity, model);
```

ده رستوران پیشنهاد داده شده برای این رستوران به همراه پیش بینی امتیاز برای هر یک از آن ها در شکل زیر قرار داده شده است.

[7] evaluation
[8] restaurant data set
[9] threshold

```
Terminal: Local
RecommendedItem[item:134986, value:2.0]
RecommendedItem[item:132866, value:2.0]
RecommendedItem[item:134975, value:2.0]
RecommendedItem[item:132584, value:2.0]
RecommendedItem[item:132768, value:1.7451868]
RecommendedItem[item:132608, value:1.735849]
RecommendedItem[item:132733, value:1.7068063]
RecommendedItem[item:132767, value:1.6923077]
```

حال این سوال به ذهن میرسد که اگر معیار شباهت دیگری انتخاب کنیم و یا معیار همسایگی را متفاوت کنیم سیستم پیشنهاد دهنده ی بهتری خواهیم داشت؟

برای پاسخ به این سوال، برای این مجموعه داده دو روش متفاوت پیاده سازی سیستم پیشنهاد دهنده را ارزیابی کردم و با استفاده از کلاس های evaluation مخصوص Apache Mahout این دو روش را مقایسه کردم.

برای این کار از روش Mean Average Error استفاده کردم. در این روش برای ارزیابی، مجموعه داده مورد نظر را به دو قسمت train و test تقسیم میکند.

ابتدا برای سیستم پیشنهاد دهنده معیار شباهت CityBlock Similarity و معیار همسایگی Thresholds قرار دادم و برای ارزیابی 70 درصد داده را به عنوان train در نظر گرفتم. درصد به دست آمده به صورت زیر بود.

```
20/07/20 09:31:25 INFO eval.AbstractDifferenceRecommenderEvaluator: Evaluation result: 0.7722061894371011
0.7722061894371011
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.891 s
```

سپس معیار شباهت را تغییر داده از Pearson Correlation استفاده کردم. همچنین معیار همسایگی Nearest N users را به جای Thresholds استفاده کردم. در این حالت نیز برای ارزیابی 70 درصد داده را به عنوان train در نظر گرفتم. درصد به دست آمده به صورت زیر است.

```
20/07/20 09:41:27 INFO eval.AbstractDifferenceRecommenderEvaluator: Evaluation result: 0.8922378819593242
0.8922378819593242
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.922 s
```

مقایسه ی دو درصد بالا نشان میدهد با استفاده از روش های متفاوت و ترکیب آن ها بسته به داده ی مورد نظر نتایج متفاوتی حاصل میشود. پس برای هر نوع مجموعه داده و هر نوع سیستم باید روش های متفاوت ارزیابی شوند تا بهترین روش مشخص و استفاده شود.

## مجموعه داده لباس:

برای استفاده از این مجموعه داده نیز مانند قسمت قبل دادهی دانلود شده نیاز به تغییراتی داشت که با pandas این تغییرات را اعمال کردم.

برای کاربر با شماره ی 907115 با استفاده از روش توضیح داده شده در قسمت قبل این پکیج، پیشنهادات زیر را محاسبه کرده است:

```
Terminal: Local
RecommendedItem[item:1141067, value:10.0]
RecommendedItem[item:1486261, value:10.0]
RecommendedItem[item:877767, value:10.0]
RecommendedItem[item:2826127, value:10.0]
RecommendedItem[item:1427750, value:10.0]
RecommendedItem[item:2077141, value:10.0]
RecommendedItem[item:1415952, value:10.0]
RecommendedItem[item:1871026, value:10.0]
```

همچنین برای کاربر با شماره ی 999231 نیز با استفاده از همین روش پیشنهادات زیر را محاسبه کرده است:

```
Terminal: Local +
RecommendedItem[item:1497935, value:10.0]
RecommendedItem[item:2077141, value:10.0]
RecommendedItem[item:2520466, value:10.0]
RecommendedItem[item:1432504, value:10.0]
RecommendedItem[item:871172, value:10.0]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
TODO  Terminal
```

## مجموعه داده کتاب:

مانند دو قسمت قبل برای این مجموعه داده هم تغییرات مورد نظر را اعمال کردم و در نهایت برای کاربر با شماره ی 178622 با استفاده از معیار شباهت CityBlock Similarity و حد همسایگی 0.1 (به این معنا که کاربرانی که شباهتشان با کاربر مورد نظر مساوی یا بیشتر 10 درصد است به عنوان همسایه معرفی می شوند) پکیج، پیشنهادات زیر را داده است :

```
Terminal: Local +
RecommendedItem[item:60256672, value:10.0]
RecommendedItem[item:60508302, value:10.0]
RecommendedItem[item:373709110, value:10.0]
RecommendedItem[item:373790651, value:10.0]
RecommendedItem[item:64471063, value:10.0]
RecommendedItem[item:671027573, value:10.0]
RecommendedItem[item:446671002, value:10.0]
RecommendedItem[item:140620125, value:10.0]
TODO  Terminal
```

## پکیج LightFM:

### معرفی:

یک پیاده سازی به زبان پایتون از چند الگوریتم توصیه دهنده برای بازخوردهای ضمنی[10] و صریح[11] است. این فریم ورک این امکان را برای ما فراهم می‌کند تا بتوانیم از ویژگی های[12] کاربر و کالا برای سیستم توصیه گر استفاده کنیم و نتیجه را بهبود دهیم.

الگوریتم اصلی مورد استفاده در این فریم ورک model base از زیر مجموعه ی فیلترسازی مبتنی بر همکاری[13] است که در کلاس اصلی lightfm انجام می‌شود. برای این که بتوان از این پکیج استفاده کرد باید داده را در قالب ماتریس هایی تعریف کرد و در واقع داده را برای به کار بردن در این پکیج آماده کرد. تصویر زیر نمونه قطعه کد از این تبدیل است:

```
f = open('rating_final_U1011.json', )
ff = open('userprofile.json', )
df = open(r'geoplaces2.json')
data_User = json.load(ff)
data_item = json.load(df)
data = json.load(f)
dataset = Dataset()
dataset.fit((x['userID'] for x in data),
            (x['placeID'] for x in data), (x['budget'] for x in data_User), (x['price'] for x in data_item))
(interactions, weights) = dataset.build_interactions(((x['userID'], x['placeID']) for x in data))
print(repr(interactions))
user_interactions = dataset.build_user_features((x['userID'], [x['budget']]) for x in data_User)
print(repr(user_interactions))
item_interactions = dataset.build_item_features((x['placeID'], [x['price']]) for x in data_item)
print(repr(item_interactions))
```

برای به کار بردن مدل باید مدل را بر روی داده fit کرد. پیشنهاد دادن لیست جدید ویژگی مهم در سیستم‌های توصیه گر محسوب میشود. و در پیشنهاد دادن به همه مجموعه داده ها در اجراهای متفاوت شاهد نتایج جدید نیز هستیم.

برای هر مجموعه داده باید مراحل زیر را انجام داد:

- داده را به خوبی تحلیل کرد و آن را فهمید.
- مدل های مختلف را طراحی کرد .

[10] Implicit feedback
[11] Explicit feedback
[12] features
[13] collaborative filtering

- داده را به دو قسمت test و train تقسیم کرد.
- مدل های طراحی شده را بر روی train , test مقایسه کنیم و مدل بهتر را انتخاب کنیم.
- دقت مدل را اندازه گرفت.
- بعد از نوشتن توابع مربوطه لیست پیشنهادی مورد نظر را تولید کرد.

در ادامه به هر کدام از این موارد پرداخته خواهد شد.

## مدل های استفاده شده در این پروژه:

سه loss function اصلی و دو مدل برنامه نرخ یادگیری تعریف شده در lightfm[14] در این پروژه مقایسه شده اند. باید توجه کرد که مدل های یادگیری این پکیج با در نظر گرفتن نتایج مثبت و منفی تصمیم گیری میکنند.

برنامه ها نرخ یادگیری:

1. adagrad
2. adadelta

loss functions:

1. BPR
2. WARP
3. k_OS WARP

در تصویر زیر یک نمونه مدل را میبینیم:

```
alpha = 1e-3
epochs = 70

model = LightFM(no_components=30,
                loss='warp',
                learning_schedule='adadelta',
                user_alpha=alpha,
                item_alpha=alpha)
```

[14] learning rate schedules

## مجموعه داده رستوران[15]:

در این مجوعه از برای پیشنهاد دادن چون اطلاعات رستوران هایی که کاربر U1011 قبلا رفته بود کاملا از داده ها حذف کردیم از پروفایل کاربر و رستوران ها برای پیشنهاد دادن بهتر استفاده کردم اطلاعات حذف شده از داده اصلی جهت سنجش بهتر به این صورت است :

| | userID | name | rating |
|---|---|---|---|
| 1 | U1011 | churchs | 1 |
| 2 | U1011 | tortas hawai | 2 |
| 3 | U1011 | tacos de la estacion | 1 |

در نهایت موارد درگیر در تصمیم‌گیری به این صورت است:

budget از پروفایل مشتری، price از پروفایل رستوران و rate نهایی که مشتری ها به رستوران داده اند.

مدل استفاده شده نیز برای این داده WARP است که با توجه به الگوریتم و سنجش های انجام شده مدل بهتری است.

### دقت مدل نهایی و ماتریس های ساخته شده:

```
<138x131 sparse matrix of type '<class 'numpy.int32'>'
    with 1159 stored elements in COOrdinate format>
<138x142 sparse matrix of type '<class 'numpy.float32'>'
    with 276 stored elements in Compressed Sparse Row format>
<131x134 sparse matrix of type '<class 'numpy.float32'>'
    with 261 stored elements in Compressed Sparse Row format>
Precision: train 0.18
AUC: train 0.61

Process finished with exit code 0
```

### اجرا نهایی و پیشنهادات برای مشتری U1011:

پس از منطبق کردن داده در توابع مربوط در lightfm که منجر به تشکیل ماتریس هایی میشود که مشتری و رستوران و پروفایل هایشان را درگیر میکند . در نهایت با اجرای کد و وارد کردن مشتری هدف(U1011) لیستی تحویل میگیریم که با معیار اصلی rating انتخاب شده اند. در عکس زیر دو نمونه از این اجرا ها را میبینیم:

[15] restaurant data set

```
Known Likes:

 Recommended Items:
1- Cafe Chaires
2- palomo tec
3- don burguers
4- churchs
5- Chilis Cuernavaca

Process finished with exit code 0
```

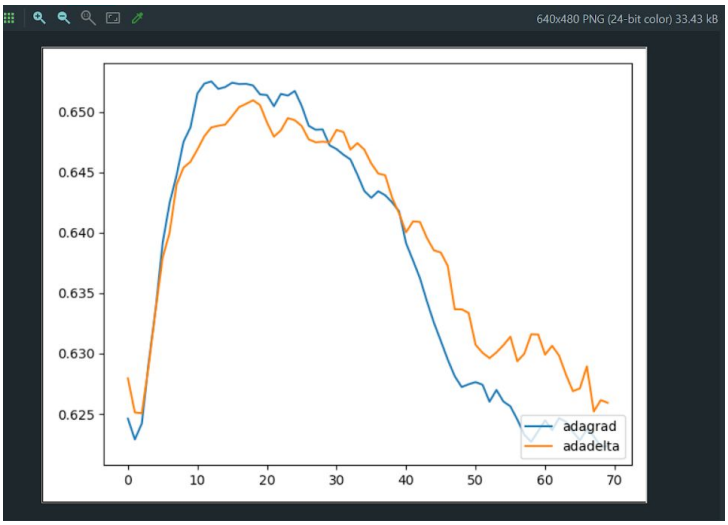
```
collabrative_restaurant ×

 Recommended Items:
1- churchs
2- cafe ambar
3- Cafe Chaires
4- Chilis Cuernavaca
5- pizza clasica
```

همان طور که میبینیم در دو نمونه اجرا رستوران church که از رستوران هایی بوده که مشتری U1011 رفته است در لیست قرار دارد. و رستوران های جدید نیز هستند. در این اجرا پیشنهادات جدید از سوی سیستم پیشنهاد دهنده نیز داده شده است.

## مجموعه داده لباس:

در این مجموعه داده به یک مشتری لباس و سایز آن را پیشنهاد می‌دهیم از مشخصات لباس از سایز و از وضعیت کاربر از rating و وضعیت fit بودن یا نبودن برای طراحی مدل استفاده شده است. داده را به دوقسمت test و train تقسیم کرده (20درصد تست) و loss های متفاوت با learning های متفاوت را بر روی آن امتحان کردم. ابتدا با سنجش مدل های یادگیری شروع کردم در ابتدا دو learning schedule مختلف را با هم مقایسه کردم و نتایج زیر حاصل شده است:[16]

### مقایسه adagrad و adadelta در warp loss:[17]

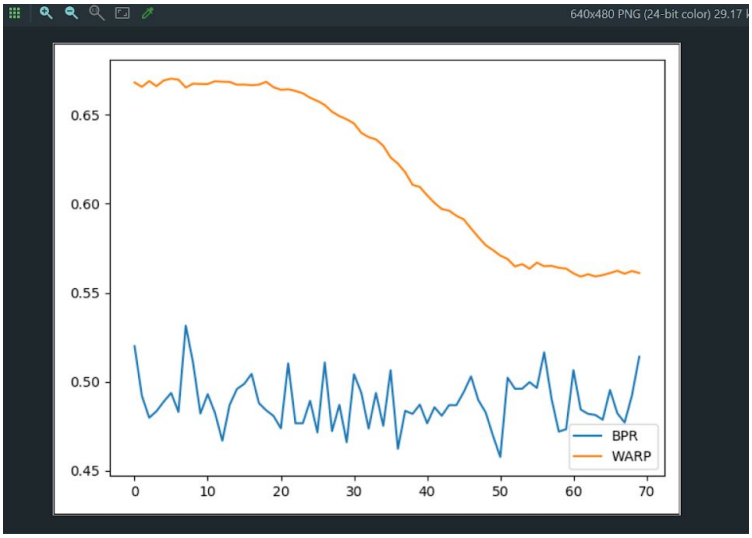



همانطور که در نمودار بالا نشان داده شده است. adadelta عملکرد بهتری از adagrad دارد . به طور معمول چون

[16] تست ها نسبت به فیلم تحویل داده شده مجدد انجام شده اند و تفاوت هایی دارند.
[17] این تست در فیلم ضبط شده نتیجه ای متفاوت داشت و در آن اجرا adagrad عملکرد بهتری داشت.

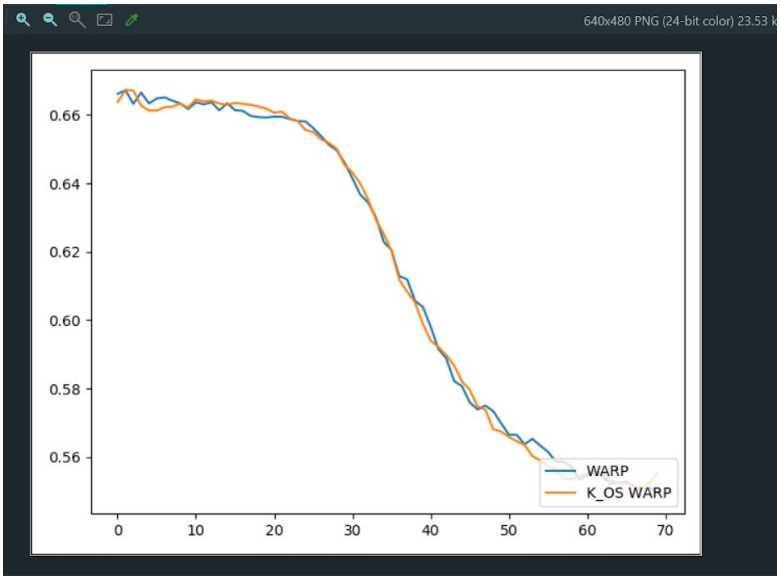
adadelta یک نمونه گسترش یافته از adagrad می‌باشد و معمولا با حذف یکنواختی ها و داده ها مهاجم به نتیجه بهتری دست پیدا میکند. گاهی نیز این حذف ها ممکن است نتیجه معکوس داشته باشد و نتیجه adagrad بهتر شود.

**مقایسه BPR و WARP:**




WARP نتیجه بهتری دارد. در واقع هر دو مدل از 3 تایی rating مثبت و منفی و کاربر استفاده میکنند اما BPR این 3 تایی ها را به صورت تصادفی انتخاب می‌کند ولی WARP تصادفی نیست و با در نظر گرفتن فاکتور هایی به صورت هوشمندانه 3 تایی ها را انتخاب می‌کند و معمولا عملکرد بهتری دارد.

**مقایسه K_OS WARP و WARP:**



به دلیل تشابه دو روش نتایج تقریبا منطبق هستند.

در نهایت برای پیاده سازی مدل این داده ست از WARP و adadelta استفاده می‌کنیم که عملکرد بهتری داشته اند.

**دقت مدل نهایی و ماتریس های ساخته شده:**

```
<8943x3074 sparse matrix of type '<class 'numpy.int32'>'
    with 10000 stored elements in COOrdinate format>
<8943x8946 sparse matrix of type '<class 'numpy.float32'>'
    with 18176 stored elements in Compressed Sparse Row format>
<3074x3119 sparse matrix of type '<class 'numpy.float32'>'
    with 9668 stored elements in Compressed Sparse Row format>
Precision: train 0.01, test 0.01.
AUC: train 0.86, test 0.72.

Process finished with exit code 0
```

**اجرا نهایی و پیشنهادات برای مشتری 999231:**

```
Known Likes:
1- 2725433 : 8
2- 1192323 : 4

 Recommended Items:
1- 147594 : 24
2- 131533 : 12
3- 405907 : 8
4- 155381 : 8
5- 167112 : 12
```

```
Known Likes:
1- 2725433 : 8
2- 1192323 : 4

 Recommended Items:
1- 147594 : 24
2- 131533 : 12
3- 159395 : 14
4- 405907 : 8
5- 155381 : 8
```

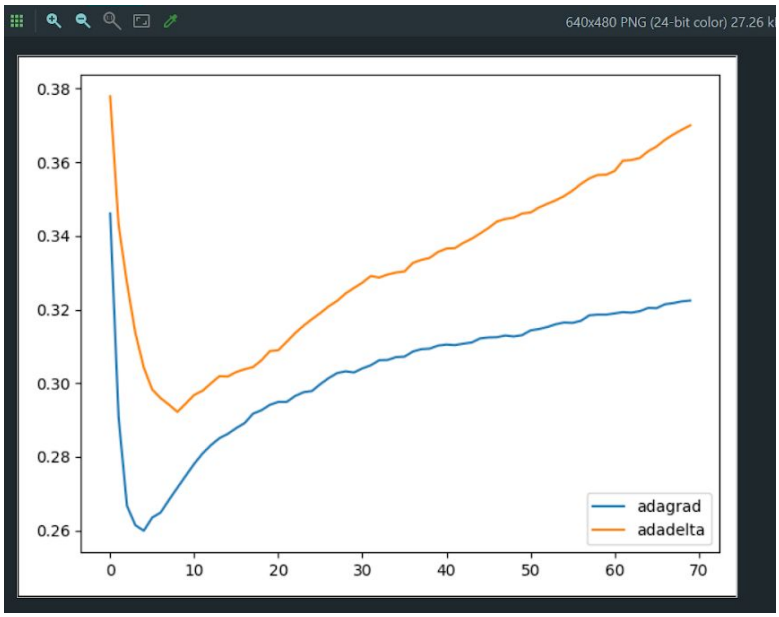
لیست نمایش داده شده به مشتری در دو اجرا متفاوت نشان میدهد که این مشتری لباس 147594 را احتمالا دوست دارد و سایز پیشنهادی به او سایز 24 است.

## مجموعه داده کتاب:

با توجه به اینکه هدف از این مجموعه داده پیشنهاد دادن کتاب به کاربر است برای طراحی مدل از ماتریس کتاب و نویسنده

و ماتریس کاربر و rating استفاده کرده ام در واقع یک بار با قرار دادن نویسنده در ماتریس ویژگی های کتاب و یک بار با قرار دادن rating در این ماتریس اجرا گرفتم. بعد از تقسیم داده به test و train به صورت 20 درصد برای تست، جهت سنجش بهترین loss و یادگیری با توجه به نتایج زیر تصمیم گیری کردم:

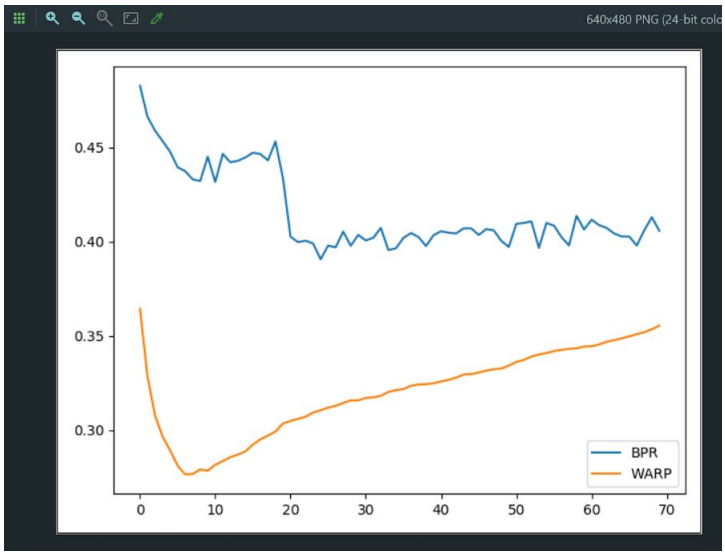
**مقایسه adagrad و adadelta در warp loss:**



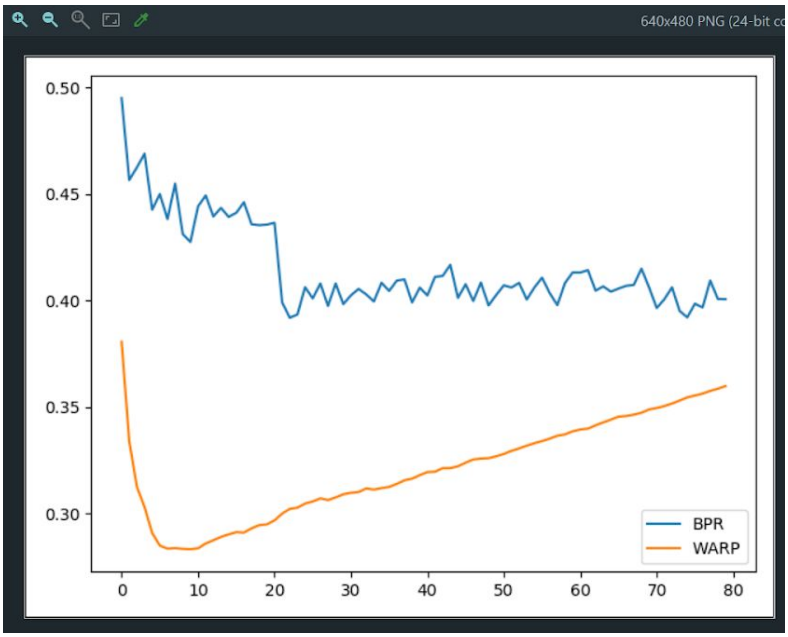

نتیجه می‌شود که adadelta عملکرد بهتری نسبت به adagrad دارد. از نظر من دلیل این که هیچ کدام نتیجه خیلی خوبی تا این جا نداشته اند (یعنی 0.38 بیشترین مقدار بوده است) میتواند به دلیل تنوع داده باشد. در واقع همان طور که در توضیحات دادهها بیان شد این 10000 سطر داده از مجموعه داده بزرگتر به صورت رندوم انتخاب شده اند پس یادگیری و گرفتن نتیجه از آن ممکن است بیشتر زمان ببرد؛ میبینیم که شیب نمودار adadelta صعودی و زیاد است و به احتمال زیاد در دوره[18] های بعدی به نتایج بهتری می رسد و همچنین اختلاف آن نیز با adagrad بیشتر می‌شود.

[18]epoch

**مقایسه BPR و WARP :**

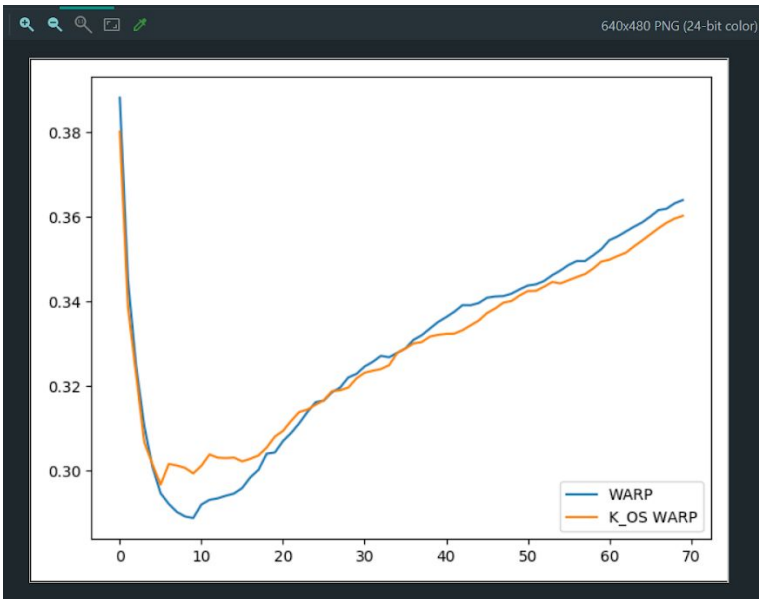



نتیجه BPR بهتر بوده است. میتوان نتیجه گرفت انتخاب های رندوم در این داده تا دوره‌ی 70 نتیجه بهتری نسبت به WARP داشته است، اما با پیشبرد زمان BPR در حال ثابت شدن است و WARP در با شیب مثبت در حال پیشروی است و پیش بینی می‌شود WARP اگر زمان برای یادگیری داشته باشد نتیجه بهتری در دوره های بعد بگیرد در شکل زیر epoch را روی 80[19] قرار داده ام در حالت قبل روی 70 بود میبینیم که اختلاف دو نمودار کم شده است. در نهایت برای پیشنهاد اخر چون از دوره 70 استفاده کردم loss را BPR قرار داده ام.




[19] به دلیل این که این فرایند زمان و cpu زیادی نیاز دارد برای بیشتر از 80 عملا ممکن نبود با لپ تاپ امتحان کنم.

**مقایسه K_OS WARP و WARP:**




در این مقایسه با وجود نزدیکی که این دو حالت با یکدیگر دارند اما WARP نتیجه بهتری داشته است.

**دقت مدل نهایی و ماتریس های ساخته شده:**

پس از طراحی مدل با bpr و adadelta ماتریس ها و دقت به صورت زیر است:

```
<6017x8300 sparse matrix of type '<class 'numpy.int32'>'
    with 10000 stored elements in COOrdinate format>
<8300x13572 sparse matrix of type '<class 'numpy.float32'>'
    with 16600 stored elements in Compressed Sparse Row format>
Precision: train 0.00, test 0.00.
AUC: train 0.51, test 0.53.

Process finished with exit code 0
```

می‌توان نتیجه گرفت این داده در کل نتیجه‌ای دقیق و مناسب در ای دوره ارائه نخواهد داد.

**اجرا نهایی و پیشنهادات برای مشتری 178622 با آستانه[20]ی نویسنده کتاب:**

```
Known Likes:
1- God Wants You to Be an Entrepreneur : Rickey Singleton

 Recommended Items:
1- The Diary of a Young Girl: The Definitive Edition : Anne Frank
2- Crooked Little Heart : Anne Lamott
3- The Conspiracy Club : JONATHAN KELLERMAN
4- Ella Enchanted (Trophy Newbery) : Gail Carson Levine
5- Crewel World (Needlecraft Mysteries) : Monica Ferris

Process finished with exit code 0
```

**اجرا نهایی و پیشنهادات برای مشتری 178622 با آستانه‌ی rating کتاب بیشتر از 5:**

```
Known Likes:
1- God Wants You to Be an Entrepreneur : Rickey Singleton

 Recommended Items:
1- This Is Graceanne's Book : A Novel : P. L. Whitney
2- Gossip Girl #1 : A Novel by Cecily von Ziegesar (Gossip Girl) : Cecily Von Ziegesar
3- On the Road (Essential.penguin S.) : Jack Kerouac
4- The Case of the Class Clown (Jigsaw Jones Mystery, 12) : James Preller
5- Lamb : The Gospel According to Biff, Christ's Childhood Pal : Christopher Moore

Process finished with exit code 0
```

[20] threshold

## پکیج Surprise Recommender:

### معرفی:

توصیه گر surprise در واقع یک scikit است. البته از آن به عنوان یک انجین نیز یاد می شود. در واقع surprise یک ابزار برای تحلیل و ساخت سیستم های توصیه گر است.

با استفاده از این انجین می توانیم دیتاستی که ریتینگ های آن به صورت صریح مشخص شده است را تحلیل و پیشنهادهای مناسب را به کاربر ارائه دهیم اما اگر دیتاست فاقد ریتینگ باشد یا ریتینگ های آن به صورت ضمنی بیان شده است توسط این انجین قابل تحلیل نیست. به عبارت دیگر این انجین قابلیت پردازش و تحلیل داده های content-based را ندارد.

### اهداف طراحی:

صاحبان کسب و کارها یا کاربران در زمینه های مختلف داده هایی دارند که به تحلیل درست روی آن ها و نتیجه گیری از این تحلیل ها برای تصمیم گیری های درست نیاز دارند. انجین surprise به کاربران این اجازه را می دهد تا بر روی داده های خود تست های لازم را انجام دهند، داده ها با فرمت های مختلف را به راحتی با استفاده از الگوریتم های مختلف پردازش کنند و نتایج مورد نظر خود را به روش های مختلف دریافت کنند. هدف اصلی طراحی این انجین، هندل کردن دیتاست های مختلف بوده است.

### الگوریتم های قابل استفاده:

این انجین علاوه بر این که امکان استفاده از الگوریتم های آماده مانند الگوریتم های baseline ، روش های همسایگی، الگوریتم های matrix factorization-based همچون SVD و PMF را فراهم کرده است، این امکان را به کاربر می دهد تا الگوریتم مورد نظر خود را در این انجین پیاده سازی کند. علاوه بر این ها similarity measure های متنوعی مانند cosine و MSD و Pearson نیز در این انجین پیاده سازی شده اند و قابل استفاده هستند.

این انجین ابزارهایی را برای ارزیابی، تحلیل و مقایسه عملکرد الگوریتم های مختلف ارائه می کند. به طور مثال روش های cross-validation با استفاده از iterator های CV به راحتی قابل اجرا هستند. همچنین قابلیت جستجوی کامل روی مجموعه ای از پارامترها نیز در این انجین پیاده سازی شده است.

## پکیج های SciKit:

همانطور که اشاره کردیم surprise یک SciKit است. SciKit ها یک سری پکیج های اضافه شدنی برای SciPy هستند. SciPy در واقع یک اکوسیستم از نرم افزارهای متن باز در زمینه ریاضیات، علوم پایه و مهندسی بر پایه پایتون می باشد. پکیج های SciKit به صورت کاملا مستقل و جدا از نسخه اصلی SciPy توسعه و میزبانی می شوند و همه آنها تحت گواهینامه های OSI-approved هستند.

پکیج *NumPy* کتابخانه *SciPy* پکیج *SymPy* پکیج *pandas* کتابخانه *Matplotlib* کنسول *IPython*

## تکنیک پیاده سازی سیستم توصیه گر در این انجین:

در این جا ما برای تحلیل دیتاست خود با استفاده از انجین surprise از الگوریتم SVD استفاده می کنیم. این الگوریتم از الگوریتم های تکنیک collaborative filtering است و به طور دقیق تر جزو تکنیک های Model-based می باشد. همچنین در یک تقسیم بندی دیگر الگوریتم SVD را می توانیم از الگوریتم های تکنیک های matrix factorization-based نیز بدانیم.

برای تحلیل داده های یک مجموعه داده با استفاده از این انجین کافیست ابتدا داده ها را به صورت مرتب در یک فایل data ذخیره کنیم یا به صورت فریم هایی از دیتا به انجین بدهیم.

rating_final.data

```
U1077,135085,2
U1077,135038,2
U1077,132825,2
U1077,135060,1
U1068,135104,1
U1068,132740,0
U1068,132663,1
U1068,132732,0
U1068,132630,1
U1067,132584,2
U1067,132733,1
U1067,132732,1
U1067,132630,1
U1067,135104,0
U1067,132560,1
U1103,132584,1
U1103,132732,0
U1103,132630,1
U1103,132613,2
U1103,132667,1
U1103,135104,1
U1103,132663,1
U1103,132733,2
U1107,132660,2
U1107,132584,2
U1107,132733,2
U1044,135088,2
U1044,132583,1
```

در این جا برای داده های رستوران را به صورت یک فایل .data به انجین داده ایم. این مجموعه داده دارای سه ستون است که به ترتیب شناسه کاربر، شناسه رستوران و امتیازی است که کاربر به رستوران داده است. برای خواندن فایل rating_final.data توسط انجین باید ستون های آن را مشخص کنیم. این کار را در خط 52 با مقدار دهی line_format انجام داده ایم. حال داده ها را از مجموعه داده در متغیر data بارگذاری می کنیم و سپس یک مجموعه به عنوان مجموعه train از روی این داده ها می سازیم. الگوریتم SVD که استفاده می کنیم این مجموعه train را می گیرد و تحلیل های خود را روی آن انجام می دهد و به عبارت دیگر یاد می گیرد. پس از اینکه پردازش الگوریتم بر روی مجموعه train به اتمام رسید با استفاده از همان مجموعه train یک مجموعه به نام مجموعه test از داده هایی که در مجموعه train نیستند ساخته می شود. مثلا اگر ده رستوران داشته باشیم و یک کاربر به دو رستوران امتیاز داده باشد، دو ردیف داده در مجموعه train دارد اما در مجموعه test به ازای تمام رستوران هایی که کاربر به آن ها امتیاز نداده

است یعنی 8 ردیف داده وجود دارد که البته مقدار امتیاز کاربر به آن ها مشخص نیست و وظیفه الگوریتم SVD در اینجا تعیین مقدار امتیازی است که این کاربر به هر کدام از این رستوران ها می دهد.

main.py | custom_test | custom_dataset | custom_train | test_accuracy.py

```
# path to dataset file
file_path = os.path.expanduser(
    'rating_final.data')
# As we're loading a custom dataset, we need to define a reader. In the
# restaurant dataset, each line has the following format:
# 'user item rating', separated by ',' characters.
reader = Reader(line_format='user item rating', sep=',')

data = Dataset.load_from_file(file_path, reader=reader)
trainset = data.build_full_trainset()
algo = SVD()
algo.fit(trainset)

# Then predict ratings for all pairs (u, i) that are NOT in the training set.
testset = trainset.build_anti_testset()
predictions = algo.test(testset)

top_n = get_top_n(predictions, n=130)
```

بنابراین با اجرای تابع test در الگوریتم SVD بر روی مجموعه test و با توجه به این که الگوریتم قبلا روی مجموعه train تحلیل های خود را انجام داده است لیستی از پیش بینی هایی که این الگوریتم انجام داده است بدست می آید. حال با دادن این پیش بینی ها به تابعی که خودمان تعریف کرده ایم یعنی تابع get_top_n تعداد n پیشنهاد اولی که بهتر است به هر کاربر داده شود را دریافت می کنیم. این تابع پیش بینی های الگوریتم را به ازای هر کاربر مرتب می کند و سپس بسته به ورودی تابع n عدد رستورانی را که پیش بینی می کند هر کاربر بیشترین امتیاز را به آنها بدهد به عنوان خروجی برمی گرداند.

main.py | custom_test | custom_dataset | custom_train | test_accuracy.py | te

```python
def get_top_n(predictions, n=10):
    '''Return the top-N recommendation for each user from a set of predictions.

    Args:
        predictions(list of Prediction objects): The list of predictions, as
            returned by the test method of an algorithm.
        n(int): The number of recommendation to output for each user. Default
            is 10.

    Returns:
    A dict where keys are user (raw) ids and values are lists of tuples:
        [(raw item id, rating estimation), ...] of size n.
    '''

    # First map the predictions to each user.
    top_n = defaultdict(list)
    for uid, iid, true_r, est, _ in predictions:
        top_n[uid].append((iid, est))

    # Then sort the predictions for each user and retrieve the k highest ones.
    for uid, user_ratings in top_n.items():
        user_ratings.sort(key=lambda x: x[1], reverse=True)
        top_n[uid] = user_ratings[:n]

    return top_n
```

برای اینکه عملکرد الگوریتم را ارزیابی کنیم ، داده های مربوط به کاربر با شناسه U1011 را از مجموعه train حذف کردیم. این داده ها در واقع سه ردیف بودند. البته برای این که الگوریتم کاربر U1011 را در نظر بگیرد در مجموعه داده یک ردیف به صورت زیر برای کاربر U1011 اضافه کردیم.

```
U1043,132630,1
U1068,132733,1
U1068,132594,1
U1068,132660,0
U1011,,0
```

*ردیف اضافه شده برای کاربر U1011*

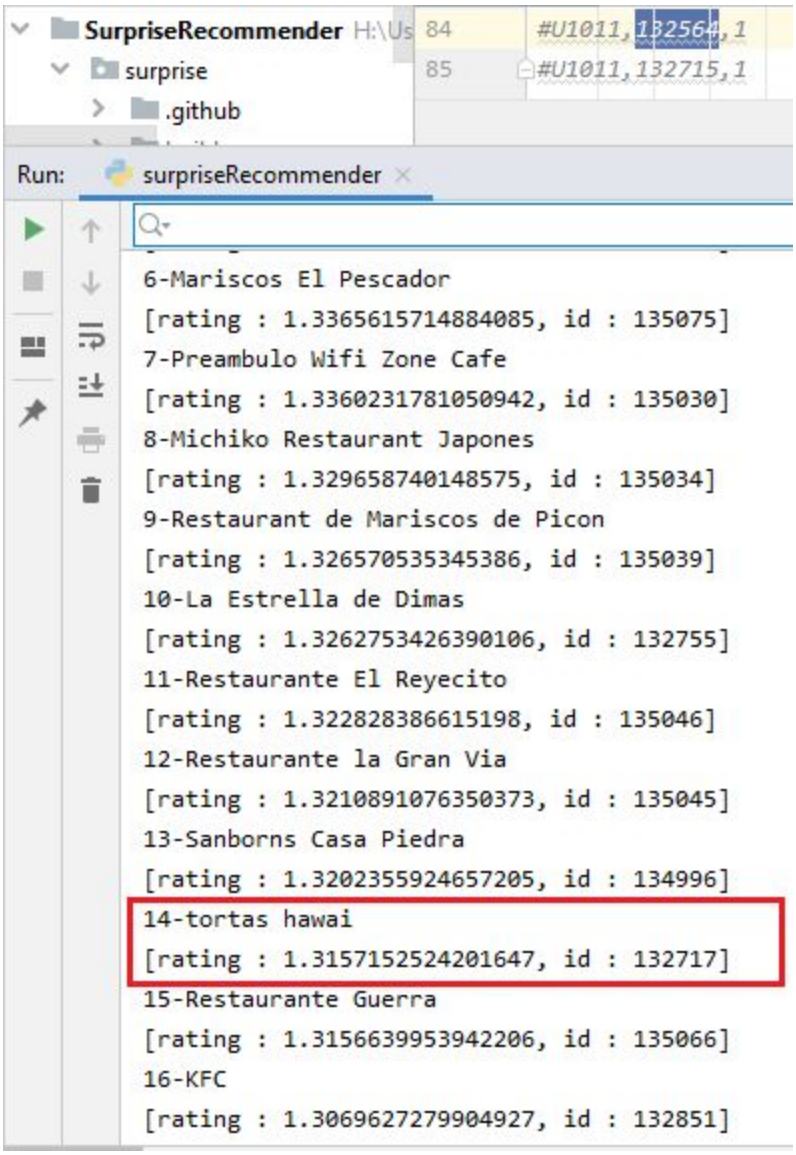
```
U1011,132717,2
U1011,132564,1
U1011,132715,1
```

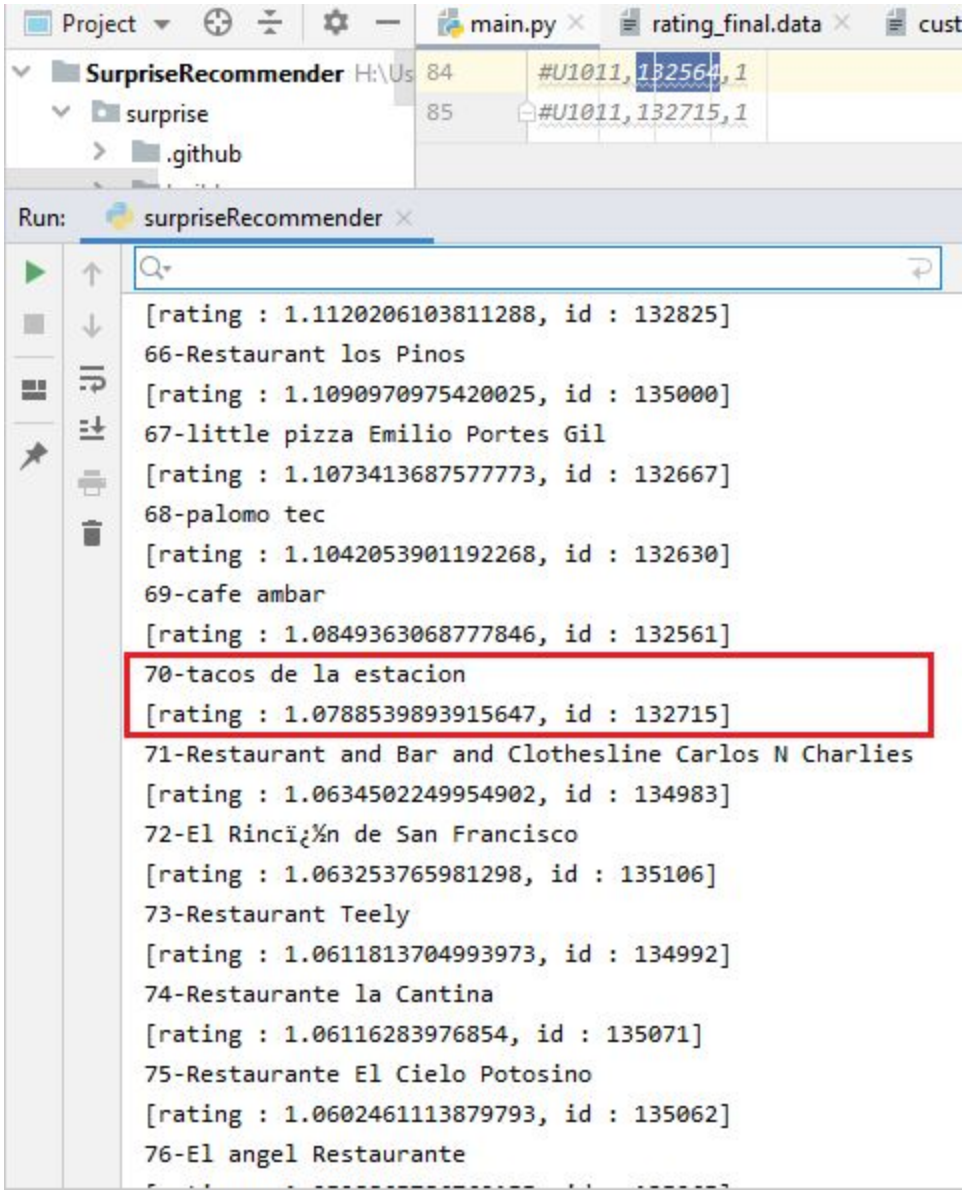
*سه ردیف حذف شده مربوط به کاربر U1011*

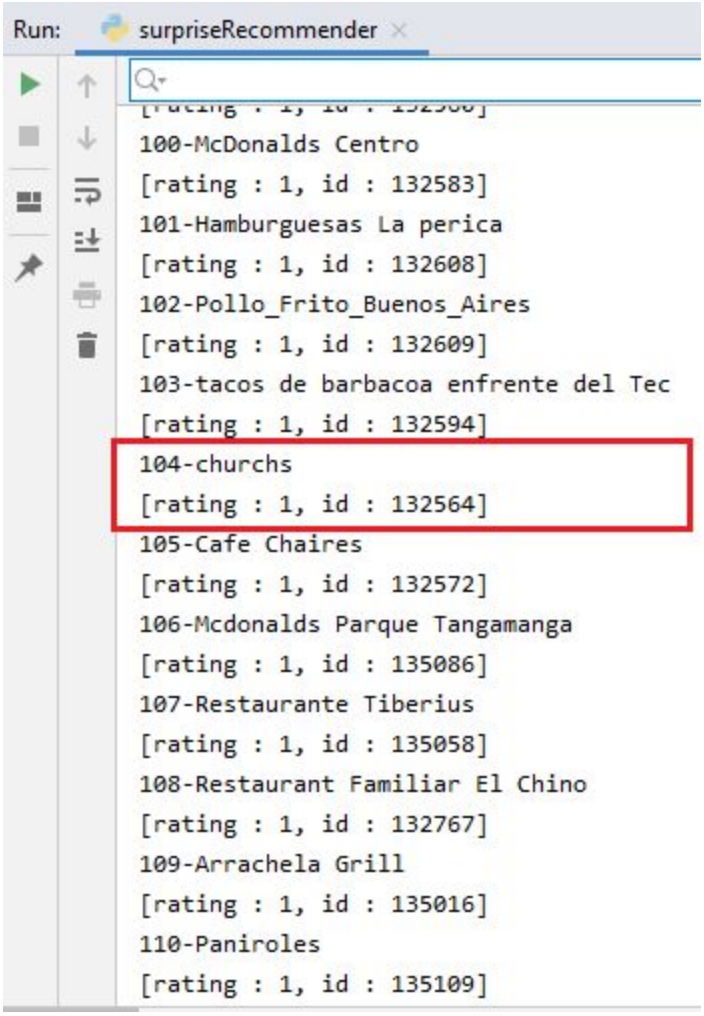
```
print("Recommended Restaurants to U1011 : ")

for uid, user_ratings in top_n.items():
    if uid == 'U1011':
        restaurant_info = [(iid, _) for (iid, _) in user_ratings]
        for i in range(0, 10):
            file_handle.seek(0)
            for row in csv_file:
                if row[0] == restaurant_info[i][0]:
                    print(str(i+1) + "-" + row[4] + "\n[rating : " + str(restaurant_info[i][1]) + ", id : " + str(restaurant_info[i][0]) +"]")
```

حال پیشنهادهایی که برای کاربر U1011 بدست آورده ایم را به ترتیب و با نام رستوران هایی که الگوریتم پیش بینی می کند بیشترین امتیاز را به آنها می دهد را چاپ می کنیم.

با توجه به مجموعه داده اولیه که سه ردیف آن را حذف کردیم، کاربر U1011 به رستوران با شناسه 132717 که نام آن "tortas hawai" می باشد امتیاز 2 را داده است. همچنین به هر دو رستوران با شناسه 132564 با نام "churchs " و 132715 با نام "tacos de la estacion" امتیاز 1 را داده است. حال با توجه به این که ما این سه ردیف را حذف کرده ایم با پیش بینی الگوریتم استفاده شده برای امتیاز کاربر به این رستوران ها و مقایسه این پیش بینی با اتفاقی که در واقعیت افتاد عملکرد الگوریتم خود را ارزیابی کنیم.


SurpriseRecommender H:\Us
surprise
.github
84 #U1011,132564,1
85 #U1011,132715,1
Run: surpriseRecommender
6-Mariscos El Pescador
[rating : 1.3365615714884085, id : 135075]
7-Preambulo Wifi Zone Cafe
[rating : 1.3360231781050942, id : 135030]
8-Michiko Restaurant Japones
[rating : 1.329658740148575, id : 135034]
9-Restaurant de Mariscos de Picon
[rating : 1.326570535345386, id : 135039]
10-La Estrella de Dimas
[rating : 1.3262753426390106, id : 132755]
11-Restaurante El Reyecito
[rating : 1.322828386615198, id : 135046]
12-Restaurante la Gran Via
[rating : 1.3210891076350373, id : 135045]
13-Sanborns Casa Piedra
[rating : 1.3202355924657205, id : 134996]
14-tortas hawai
[rating : 1.3157152524201647, id : 132717]
15-Restaurante Guerra
[rating : 1.3156639953942206, id : 135066]
16-KFC
[rating : 1.3069627279904927, id : 132851]


Project
main.py
rating_final.data
SurpriseRecommender
surprise
.github
84 #U1011,132564,1
85 #U1011,132715,1
Run:
surpriseRecommender
[rating : 1.1120206103811288, id : 132825]
66-Restaurant los Pinos
[rating : 1.1090970975420025, id : 135000]
67-little pizza Emilio Portes Gil
[rating : 1.1073413687577773, id : 132667]
68-palomo tec
[rating : 1.1042053901192268, id : 132630]
69-cafe ambar
[rating : 1.0849363068777846, id : 132561]
70-tacos de la estacion
[rating : 1.0788539893915647, id : 132715]
71-Restaurant and Bar and Clothesline Carlos N Charlies
[rating : 1.0634502249954902, id : 134983]
72-El Rincï¿½n de San Francisco
[rating : 1.063253765981298, id : 135106]
73-Restaurant Teely
[rating : 1.0611813704993973, id : 134992]
74-Restaurante la Cantina
[rating : 1.06116283976854, id : 135071]
75-Restaurante El Cielo Potosino
[rating : 1.0602461113879793, id : 135062]
76-El angel Restaurante




این تصاویر نشان دهنده پیش بینی الگوریتم برای امتیازاتی است که کاربر U1011 به رستوران های مختلف می دهد. همان طور که مشخص است کاربر به رستوران "" در واقعیت امتیاز 2 داده است و پیش بینی الگوریتم هم بدون دانستن واقعیت، حدودا امتیاز 1.32 می باشد. در این پیش بینی به نظر الگوریتم خیلی خوب عمل نکرده. البته با توجه به این که بین سه رستوران مورد نظر بیشترین امتیاز را برای رستوران "" پیش بینی کرده است عملکرد رضایت بخشی می باشد. همچنین الگوریتم برای رستوران "" حدودا امتیاز 1.08 را پیش بینی کرده که به واقعیت یعنی امتیاز 1 بسیار نزدیک است. همچنین برای رستوران "" امتیاز 1 را پیش بینی کرده است که دقیقا مطابق واقعیت است. بنابراین این الگوریتم از نظر کارایی به نظر الگوریتم قوی است البته همانطور که مشاهده کردیم همیشه نمی تواند پیش بینی دقیق و درستی داشته باشد اما به عمده پیش بینی های آن درست و قابل اعتماد است.

# نتیجه

این پروژه که در دوفاز انجام شد در فاز اول نشان داد که موتورها و پکیج های متنوع در بسترهای گوناگون وجود دارند و به برنامه‌نویس کمک می‌کنند تا با توجه به نیاز با استفاده از یک یا چند مورد از این پکیج ها به جای پیاده سازی یک سیستم پیشنهاد دهنده از صفر، از یک فریم ورک آماده استفاده کند و سریع تر نتیجه بگیرد. بعضی از این پکیج ها قابلیت این را دارند که بشود در آن ها تغییراتی داد یا الگوریتمی به آن‌ها اضافه کرد. همچنین تعداد قابل توجهی پکیج متن باز وجود دارد که در این پروژه به پیاده سازی 4 مورد از این پکیج ها پرداخته شده است.

میتوان بیان کرد که داده در یک سیستم پیشنهاد دهنده حرف اول را می‌زند. با داشتن یک مجموعه داده معتبر و مناسب میتوان پیشنهادات دقیق تری داشت. آنچه که تجربه در این پروژه به ما نشان داد حذف بخشی از داده حتی اگر به صورت تصادفی باشد ممکن است دقت سیستم را کاهش دهد. در مجموعه داده‌هایی که ما داشتیم ؛ رستوران مجموعه داده بهتری بود چون خودش هم خیلی بزرگ نبود هم مرتبط بود. مجموعه داده کتاب به دلیل پراکندگی زیاد از مجموعه داده های نامناسب برای پیشنهاد دهنده هایی که در الگوریتم نیاز به ماتریس دارند محسوب می‌شد چون با خارج کردن ماتریس از حالت پراکنده و تبدیل آن به ماتریس فشرده حافظه زیادی مصرف می‌کرد و در نهایت اجرا نمی‌شد و ما را ناچار کرد تا با انتخاب بخشی از داده به صورت تصادفی خروجی بگیریم که دقت بالایی ندارد.

برای داشتن پیشنهاد مناسب بهتر است از پکیج هایی استفاده کنیم که قابلیت استفاده از پروفایل کاربر و کالا را دارند چون ارتباط بهتری بین آیتم ها برای پیشنهاد دادن برقرار می‌کنند. پکیج lightfm این قابلیت را از بین این 4 پکیج داشت، اما این پکیج و سایر پکیج هایی که ماتریس میسازند یا از مدل بیس استفاده می‌کنند برای داده های کلان به تنهایی مناسب نیستند. از موارد دیگر پکیج surprise بود که فقط توانایی پیشنهاد دادن به صورت ضمنی را دارد و درصورت نبود rating نمیتوان از آن استفاده کرد.

پس برای انتخاب پکیج باید به نوع دیتا و این که چه چیزی قرار است پیشنهاد شود و به چه الگوریتمی نیاز داریم قطعا باید توجه کرد تا بتوان به نیاز مطرح شده پاسخ داد.

# لینک کدهای پکیج هریک از اعضای تیم

**علی راد(Case Recommender)**

https://iutbox.iut.ac.ir/index.php/s/tnzENfKdfkwWZoL

**بهار برومند(Apache Mahout)**

https://iutbox.iut.ac.ir/index.php/s/8A2xXxyCWXeeBc7

**غزاله زهتاب(LightFM)**

https://iutbox.iut.ac.ir/index.php/s/LxaFf3pWAwM3NZj

**سیدعلی امیرشاهکرمی(Surprise Recommender)**

https://iutbox.iut.ac.ir/index.php/s/6y56w3mrRwGZdKs

# منابع


https://mahout.apache.org/

https://making.lyst.com/lightfm/docs/home.html

https://pypi.org/project/CaseRecommender/

http://www2.informatik.uni-freiburg.de/~cziegler/BX/

https://surprise.readthedocs.io/en/stable/


**لینک دریافت فاز اول پروژه به صورت کامل:**

https://iutbox.iut.ac.ir/index.php/s/JrDiZPT444QswP8

**لینک دریافت فیلم پروژه:**

https://drive.google.com/file/d/1-72llRJB76_7jFVOpvBdDRNpBZgl_xKq/view