Restaurant Recommender Systems

Team

1. Abhinav Pathak
2. Abhishek Devarkonda
3. Nishanth Dheeram
4. Surya Narayana Kalipattapu

Table of Contents

1. Business Case

Recommender systems are needed to recommend the items that may be of interest to a user. Today recommendation systems are being used by numerous online businesses like Netflix, Amazon and Spotify. Recommendation to users has led to significant increase in profits and retention of customers for these firms and hence many other online businesses have started to realize the importance of using recommendation systems and started to implement in their business.

We have selected the Yelp dataset to implement recommender systems to recommend restaurants to users.

About Yelp

Yelp is the best way to find great local businesses. People use Yelp to search for everything from the city's tastiest burger to the most renowned cardiologist.

Yelp is an online platform that publish crowd-sourced reviews about local businesses, as well as the online reservation service Yelp Reservations and online food-delivery service Eat24. The company also trains small businesses in how to respond to reviews, hosts social events for reviewers, and provides data about businesses, including health inspection scores.

BUSINESS PROBLEM:

In the Yelp system, users vote for certain restaurants, giving them grade from 1 to 5 stars and also write a review. These ratings and reviews are then used by restaurants to improve their services in order to satisfy customers.

- Restaurants pay yelp to provide more and more data of ratings and reviews because it is a great way to understand what people think about the restaurants. By improving their services, they are the one who profit eventually and yelps makes profit by providing useful data.
- Also, users visit the restaurants recommended by yelp. Yelp helps in stimulating restaurant business and get a share of profit
- Yelp also earns through advertising because of huge customer base and high number of visitors on their yelp website

Therefore, it is very important for yelp:

- To generate more and more useful data to get the right feedback from users which can be provided to restaurants
- To recommend appropriate restaurants to users which they like to help in customer retention. It keeps user interested because his efforts in finding more about other restaurants of his liking has reduced

Yelp Dataset Challenge provides a large number of user, business and review data which can be used for a variety of machine learning applications. Our project is aiming to create a restaurant recommendation system using yelp data. We plan to find hidden correlations among users and restaurants, and then recommend new restaurants to users with similar interests. The second motivation is to identify what these interests are. We want to know what business the user favors more, so that we implemented the business recommendation system, which predicts the users rating on a certain business using users past experience (ratings on similar business) and other peoples rating on this business. The application of this project can extend the use of yelp to a social networking level, which allows users to find new restaurants and experience certain business together.

In our project, we try to make a model that can predict the evaluation of other businesses, for which the user did not vote and recommend him other restaurants which he is most likely to votes.

2. Data Understanding and Preparation.

The data was available to us in the JSON format and we used python to parse through the data and convert that into a data frame. We choose to only analyze the data on one type of business (restaurant) in one city in our project, but the application may extend to larger scopes. We only chose one city, Las Vegas, because we wanted to keep our data size small and we did not want to recommend a user with restaurant from different city.

Some users only have a small number of reviews (< 10) in the dataset, which is not enough for determining a user's preference of certain type of restaurants.

We have approximately 2 million users and 4,800 restaurants in the Las Vegas

Later, we also try to 1 use the user data with number of reviews greater than 50, 100, 150 and 200 to compare the results. Then we split the data into 70% training set and 30% test set to use for hold-out cross validation later

1. Restaurant Data:

- This file had a complex JSON structure and we used Pandas library functions to extract columns from the JSON Object.

```
####  The json object looks like this :
business
{
    'type': 'business',
    'business_id': (encrypted business id),
    'name': (business name),
    'neighborhoods': [(hood names)],
    'full_address': (localized address),
    'city': (city),
    'state': (state),
    'latitude': latitude,
    'longitude': longitude,
    'stars': (star rating, rounded to half-stars),
    'review_count': review count,
    'categories': [(localized category names)]
    'open': True / False (corresponds to closed, not business hours),
    'hours': {
        (day_of_week): {
            'open': (HH:MM),
            'close': (HH:MM)
        },
        ...
    },
    'attributes': {
        (attribute_name): (),
        ...
    },
```

```
In [68]: restaurants_LV_sf.head()
```

Out[68]:

JNArUYI2aVBHNNi6hXcG8g	Rigos Taco	21	3.5	65	1	none
eIhJZqHCEKMmzuPOOyqF9w	Mulligans Bar & Grill	10	4.0	28	1	full_bar
n1f9rWJzdGX982bFvZPBHg	Carl's Jr Restaurants	7	2.5	39	1	none
i7HvZ1yTTIaR6bfa_3jGEw	KFC	8	2.0	6	1	none
NkmE03gbKjSdqDM3SJsJmw	Church's Chicken	3	1.5	3	1	none
KBP-RXCaDQwfMP_-TLwuFg	Sonic Drive-In	37	3.0	218	1	none
hvVckRvOa814ry8q8WGwOg	Burger King	4	3.5	37	1	none
THEsYolDkHaJofZLs6ufWg	Jack In the Box	12	3.0	74	1	none
rG_8t33D2ADcbBqcXqwV0Q	Viva Zapatas Mexican Restaurant & Cantina ...	443	4.0	990	1	full_bar
_OyxUcWkSTRvRKFyEE7zbA	Tortas El Rey	11	3.0	21	1	none

Ambience	Delivery	Good For	Parking	Price Range	Take-out	Takes Reservations	Waiter Service	Noise Level
1.0	0	1.0	1.0	1.0	1.0	0	0	average
1.0	0	0.0	1.0	2.0	1.0	1	1	average
nan	0	0.0	0.0	1.0	1.0	0	0	average
nan	0	nan	nan	1.0	1.0	0	0	very_loud
nan	0	0.0	0.0	1.0	1.0	0	0	average
1.0	0	1.0	1.0	1.0	1.0	0	1	average
0.0	0	0.0	0.0	1.0	1.0	0	0	average
0.0	0	0.0	nan	1.0	1.0	0	0	quiet
1.0	0	1.0	1.0	2.0	1.0	1	1	average
1.0	0	1.0	1.0	1.0	1.0	0	0	average

Outdoor Seating
1
0
0
0
0

2. User Data:

```
In [28]: user.head(2)
```

Out[28]:

	average_stars	compliments	elite	fans	friends	name	review_count	type	user_id	votes	yelping
0	4.14	{u'profile': 8, u'cute': 15, u'funny': 11, u'p...	[2005, 2006]	69	[rpOyqD_893cqmDAtJLbdog, 4U9kSBLuBDU391x6bxU-Y...	Russel	108	user	18kPq7GPye-YQ3LyKyAZPw	{u'funny': 167, u'useful': 282, u'cool': 246}	2004-10
1	3.67	{u'profile': 117, u'cute': 204, u'funny': 594,...	[2005, 2006, 2007, 2008, 2009, 2010, 2011, 201...	1345	[18kPq7GPye-YQ3LyKyAZPw, 4U9kSBLuBDU391x6bxU-Y...	Jeremy	1292	user	rpOyqD_893cqmDAtJLbdog	{u'funny': 8399, u'useful': 15242, u'cool': 12...	2004-10

```
In [ ]: user
{
    'type': 'user',
    'user_id': (encrypted user id),
    'name': (first name),
    'review_count': (review count),
    'average_stars': (floating point average, like 4.31),
    'votes': {(vote type): (count)},
    'friends': [(friend user_ids)],
    'elite': [(years_elite)],
    'yelping_since': (date, formatted like '2012-03'),
    'compliments': {
        (compliment_type): (num_compliments_of_this_type),
        ...
    },
    'fans': (num_fans),
```

3. Review Data:

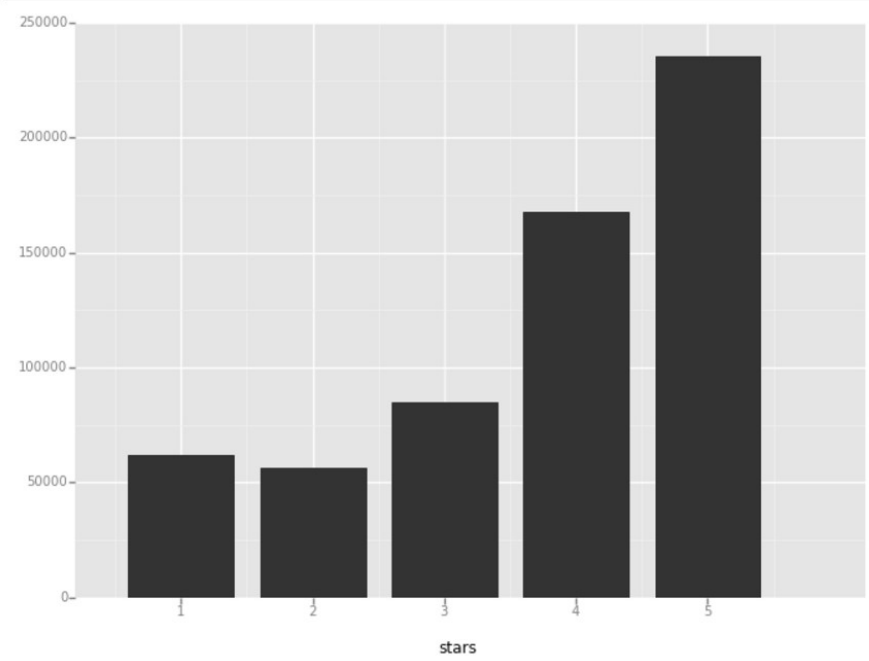
This is the user -item interaction data.

```
In [23]: # Take a sneak peak into the data
         review_LV_norm.head(2)
```

Out[23]:

	business_id	user_id	stars	name	avg_stars_user	user_normalized_stars	avg_stars_items	item_normalized_
0	JNArUYI2aVBHNNi6hXcG8g	f5vgLcoKpFcTvD4IOUxcTQ	2	Rigos Taco	3.355372	-1.355372	3.529412	-1.529412
1	JNArUYI2aVBHNNi6hXcG8g	MWw9-Eo1AQuJpdTbZ13vlw	5	Rigos Taco	4.000000	1.000000	3.529412	1.470588

The distribution of reviews by all users



3. Collaborative Filtering Model

It is an algorithm that finds similar users/items and provides recommendations to users.

It is of 2 types depending on the process flow

- **User – User collaborative filtering:** Recommends items based on ratings by similar users
- **Item – Item collaborative filtering**: Recommends items based on ratings given to similar users

1. Requirements:

Both the types of algorithms require user-item ratings matrix. Which means user's ratings for the items.

2. Uses of the model:

Now-a-days, it is very easy to gather user ratings on the items/services they have used / consumed. A simple click to rate their experience makes it convenient to the user to provide this information. This abundant information and the relatively easy collection of it makes this model powerful.

No additional information about the users and items are needed. This makes the model cost effective avoiding experts opinion to rate item features and knowing user preferences. We can also provide personalized recommendations.

3. Procedure:

The process flow for User – User collaborative filtering model includes

- Find the most similar users to the user to whom we would like to recommend using any relevant distance formula
- Based on the item ratings from similar users, find the probable rating value for the items that the specific user has not rated.
- Find the items with highest predicted ratings and suggest them to the user
- Our aim is to provide user with recommendations of items he has not rated. For this, we use the user item rating matrix, find the users that are most similar to the user in concern. This is done by calculating the similarity matrix for all the users in the database. For this there are multiple distance formulae that can be used.
- Jacard distance: It is good only when we have implicit feedback of the restaurant (whether people rated them or not), i.e., when we don't need to take into consideration the number of stars received by a restaurant.
- Cosine similarity: It calculates the similarity using the cosine distance formula with each item as a dimension. It does not consider the user bias involved in ratings given.
- Pearson correlation: It readjusts the mean of ratings by the user and then calculates the cosine similarity on these adjusted ratings.

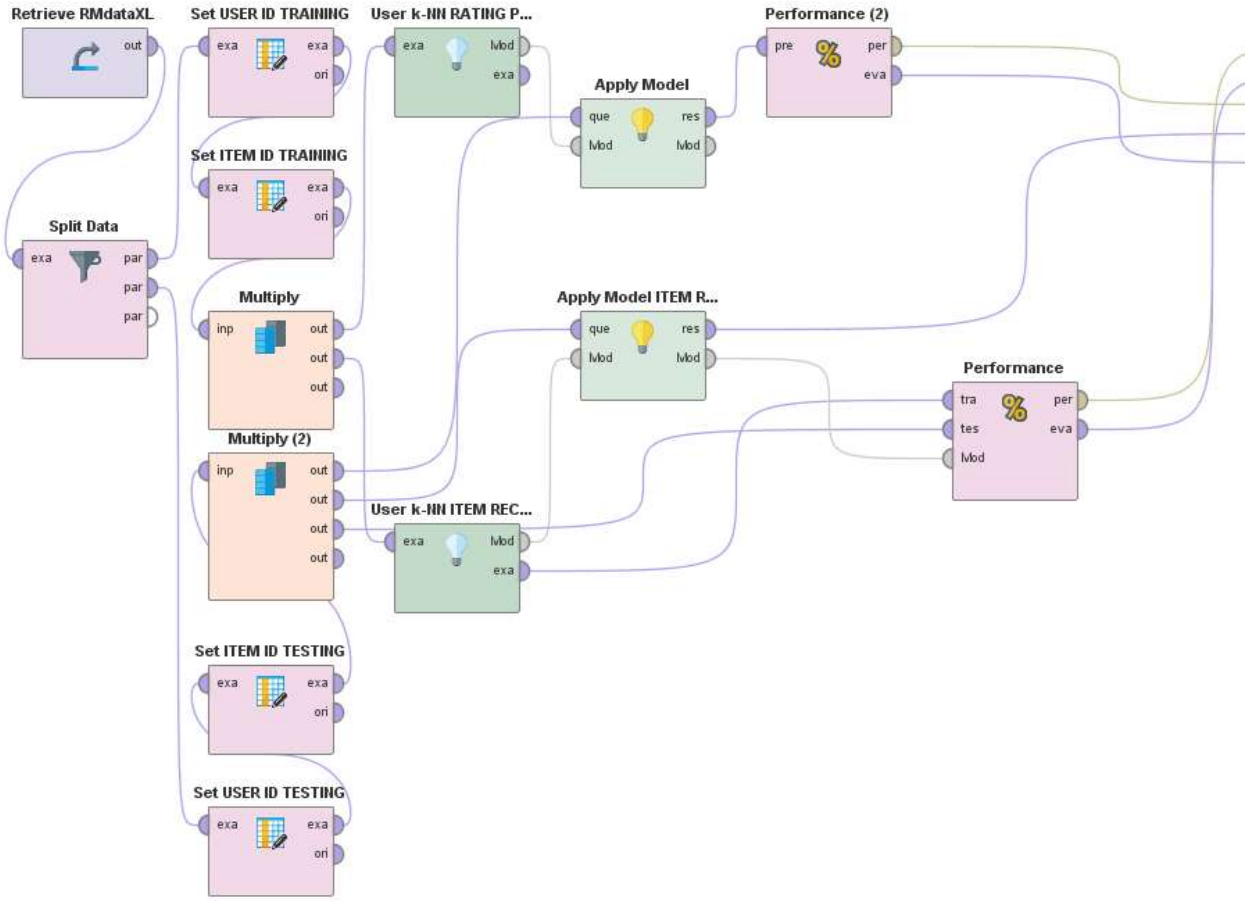
It is seen that Pearson correlation works well for the User – User collaborative filtering algorithm. This is because the formula makes sure that the user bias is taken care by adjusting the mean rating of the user.

Then for the user in concern, we identify the 'k' most similar users who have rated an item that is not rated by the user.

Then using the ratings of the identified similar users, average rating, weighted average rating, standard Group lens approach etc are used to predict the rating of the concerned user for the item.

Now the predicted ratings of the user are ranked and the top ‘n’ items are recommended to the user.

Overall, the ratings of similar users are used to provide recommendations.



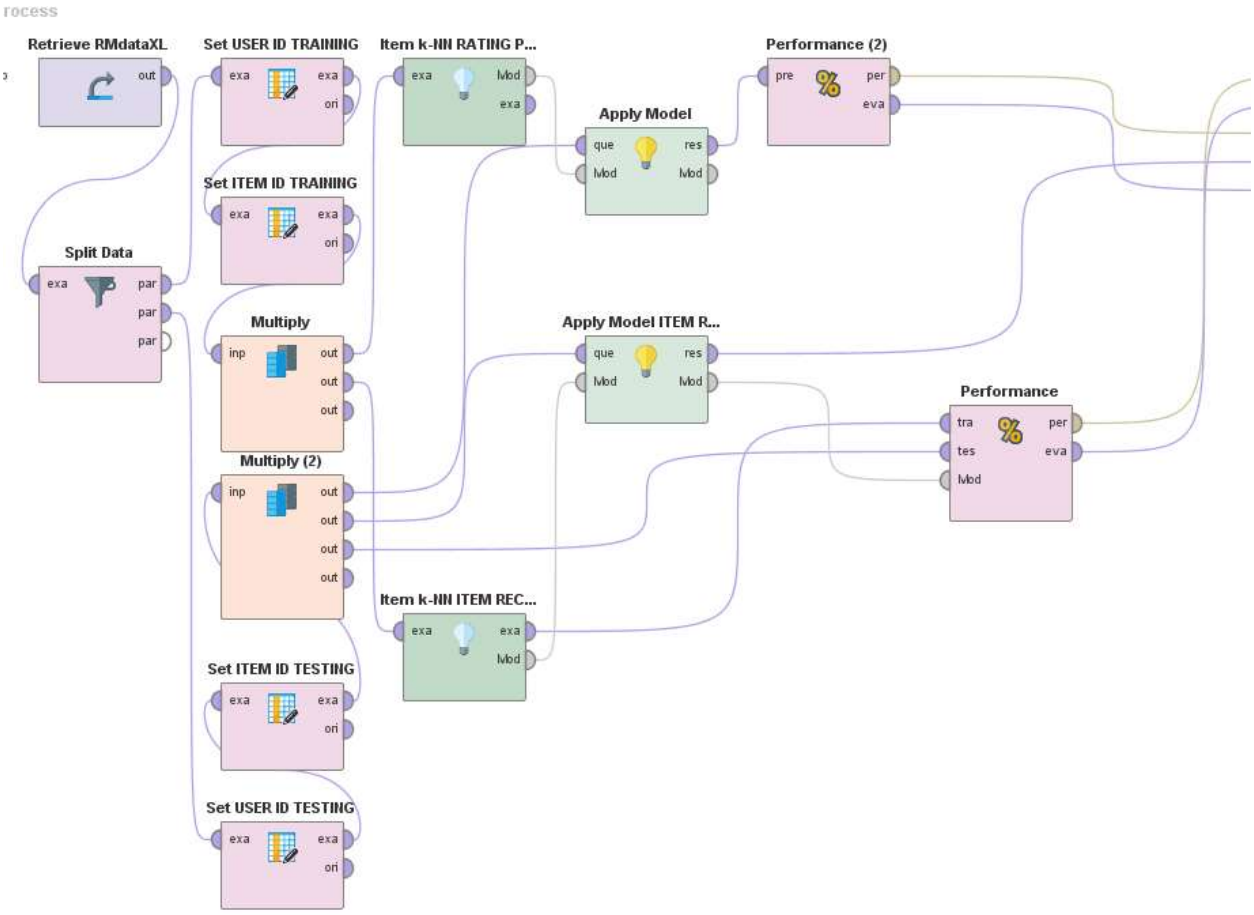
For the Item – Item collaborative filtering model, the similar procedure is being followed. The only difference here is that similarity between the items are identified instead of similar users. Hence the process is as follows:

Find the similarity between various items that are available. For this the same above mentioned 3 distance calculations Jacard, Cosine similarity, Pearson correlation can be used. It is observed that Cosine similarity works well in this case.

Then identify the items not rated by the users and find its ‘k’ most similar items for which the user has provided ratings earlier.

Then with the user ratings of the identified similar items, use average rating, weighted average rating, standard Group lens approach etc to predict the user rating for the items. These items are ranked in descending order and the top ‘n’ items are recommended to the user.

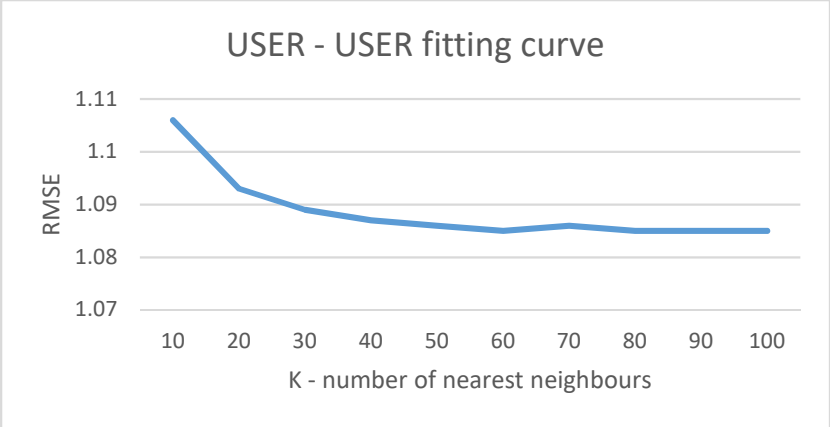
Overall, the ratings of the user are used to provide recommendations to himself.

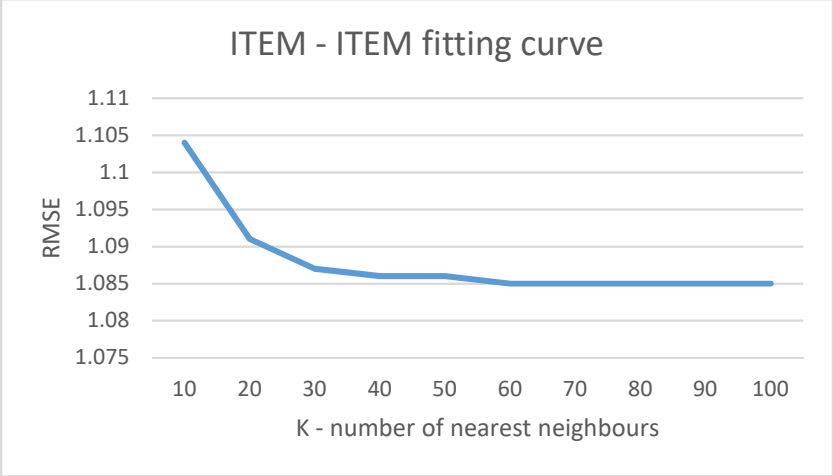


4. Model evaluation:

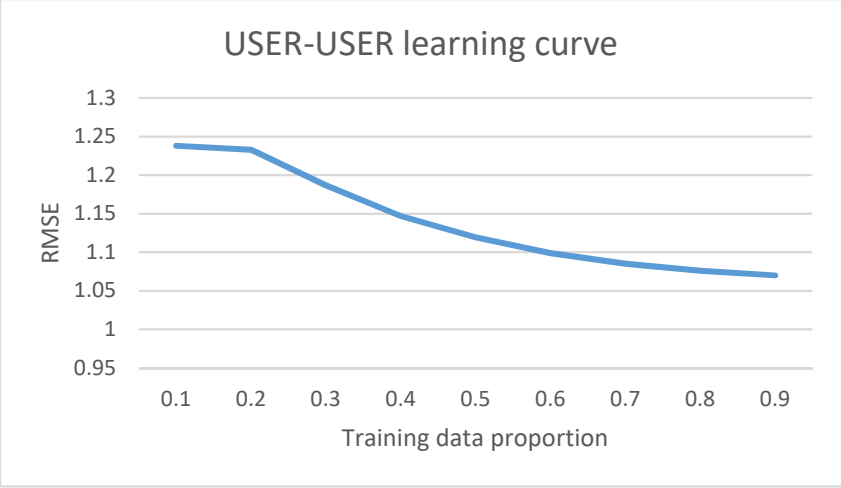
Since the prediction of ratings are numeric values, RMSE is considered a good metric for measuring the accuracy of the model. The model building involves use of RMSE to decide the appropriate value of nearest neighbors that maximizes the prediction accuracy. Hence a fitting curve is plot to decide the best k value.

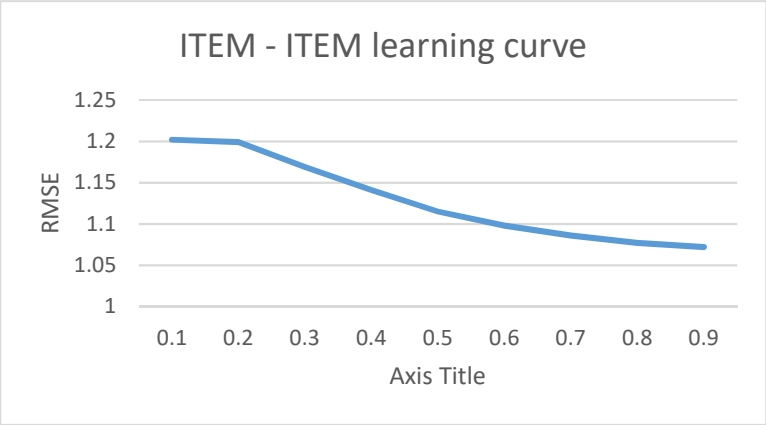
In addition to this there is an option to alter the regularization parameters of the model learning that could impact the model result. We have seen that the default values are the best in our case.





From the trials we have plotted the fitting curve and have selected the best value of K that offers the least RMSE.

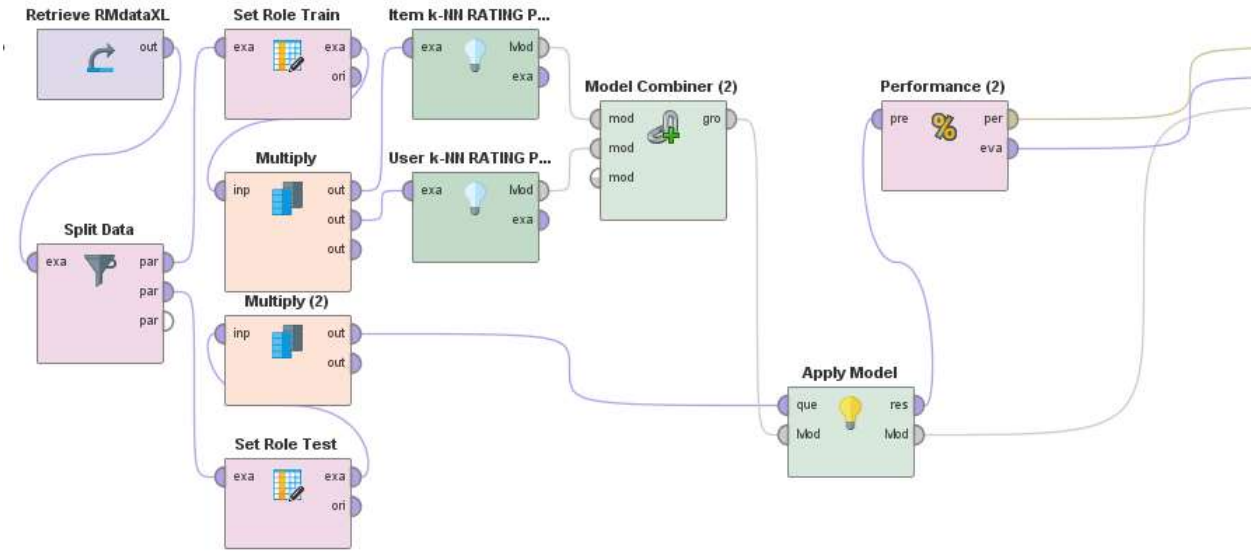




The learning curves are just a representation of how the model learns based on the amount of training data that is provided to it. As the proportion of the training data increases, we see that the RMSE on the testing data decreases.

5. Attempts to improve and other recommendations:

There are multiple ways of improving the model. For example, trying to build a hybrid model based on USER-USER and ITEM-ITEM models is an option. We have tried building a hybrid model and this has come out to be the best model with a slight improvement over the individual ones with an RMSE of 1.068



The other ways of improving a model are using other types of hybrid methodologies like Bagging, Bootsting, cross validation. These could not be performed in the project due to time and equipment constraints.

6. Final Results:

Under collaborative filtering we have tried USER-USER, ITEM-ITEM and Hybrid methodologies. The best model in each are:

- USER-USER:
  - RMSE = 1.085
  - 60 nearest neighbors
  - Pearson Correlation
  - Regularization User: 10; Regularization Item: 5; Shrinkage: 10
- ITEM-ITEM:
  - RMSE = 1.085
  - 60 nearest neighbors
  - Cosine similarity
  - Regularization User: 10; Regularization Item: 5; Shrinkage: 10
- Hybrid:
  - RMSE = 1.069
  - Equal weight of above 2 models

7. Limitations:

- It requires a good database of user ratings to provide good recommendations
- User must rate sufficient number of items to improve the effectiveness and customizability of the model
- No other relevant external features can be added into the algorithm to improve its accuracy
- Could reduce the diversity of recommendations provided

4. Content Based Recommendation:

1. Basic Idea:

Content based model ranks an item according to its similarity to other items observed for the user in question.

2. Why Content Based?

- User Independence: Collaborative filtering needs other user's rating to find the similarity between the users and then give the suggestion. Instead, content-based method only has to analyze the restaurants and user profile for recommendation.
- Transparency: Collaborative method gives the recommendation because some unknown users have the same taste like the user in question, but content-based method recommend you the items based on what features.
- No cold start: opposite to collaborative filtering, new restaurants can be suggested before being rated by a substantial number of users. We just would need Restaurant attributes discussed below.

More importantly, we would want to recommend to a user similar restaurants that he has been to before. People usually have a preference to the restaurant type. For example,

- Few people like places which serves alcohol and has good ambience, while others would prefer a quiet and romantic place. Similar points can be made for preference for availability of parking, outdoor seating arrangements etc.
- Young people mostly prefer fast food restaurant
- People between age of 30-50 prefer family restaurants

Basically, we are taking advantage of people's affinity towards certain attribute of a restaurant

3. Implementation in python:

- To run this model, we need two datasets
  1) *User Item interaction data*: This is obtained from the review table available in the yelp dataset. The review table contains ratings given by the users for all the restaurants

```
observation_data_lv[['business_id','user_id','stars_x']].head(4)
```

	business_id	user_id	stars_x
0	UEUQS4z7s-DRzQjky92KYw	ay9H1RpjbBkaiXGxfh7LaA	4
1	UEUQS4z7s-DRzQjky92KYw	Z9S2pUACe0izz5y27H-IOA	4
2	UEUQS4z7s-DRzQjky92KYw	t2CJ8v_BvhxrC291ZWnRSg	3
3	UEUQS4z7s-DRzQjky92KYw	6IHcb7PC6qiYR9qEsBieVQ	5

  2) *Item attribute data*: This is obtained from the business data table. Below is a snapshot of few columns for 2 restaurants

```
item_data_lv.head(2)
```

Unnamed: 0	Unnamed: 0.1	business_id	open	review_count	stars	Acc_credit_card	Alcohol	Ambience
0	5	--jFTZmywe7StuZ2hEjxyA	1	8	3.5	1	none	0
1	17	-1B-DEGkLE1kDj5ENAF2NQ	1	18	4.0	1	full_bar	1

Delivery	Parking	Price_Range	Take_out	Takes_Reservations	Waiter_Service	Noise_Level	Good_For	Outdoor_Sea
0	0.0	1.0	1	0	0	average	1	1
0	0.0	2.0	1	0	1	average	1	1

- We have used GraphLab module to build recommendation system.

```
import graphlab as gl
```

- Graphlab requires datasets in SFrame format. Therefore, first convert both of the datasets – item data and user-item interaction data to SFrames.

```
observation_data_lv_sf =  gl.SFrame(observation_data_lv)
item_data_lv_sf =  gl.SFrame(item_data_lv)
```

- Before, creating an instance of recommendation system, we split the data into test and training.

```
train, test = gl.recommender.util.random_split_by_user(observation_data_lv_sf,item_id = "business_id", user_id = "user_id",
                                                       max_num_users=10000)
```

- Then, we train the model.

```
m = gl.recommender.item_content_recommender.create(item_data_lv_sf,observation_data = train, target= "stars_x",
                                                   item_id="business_id",user_id = "user_id",
                                                   max_item_neighborhood_size=100)
```

Model first computes the similarity between items using the content of each item. The similarity score between two items is calculated by first computing the similarity between the item data for each column, then taking a weighted average of the per-column similarities to get the final similarity.

- Finally, we test our model using RMSE criteria. Results are discussed in 'Evaluation' section

```
m.evaluate_precision_recall(test)
```

- The recommendations are generated according to the average similarity of a candidate item to all the items in a user's set of rated items.

- User's preference for certain features of restaurants is estimated by predicting ratings for each restaurant

4. Discovering and Using features of Reviews:

Text generated by user reviews could have some information.

- From Review table, we obtained all the restaurant reviews given by all the customers.
- A Restaurant is reviewed by many customers. We combined all the reviews in a single cell for each restaurant
- Unfortunately, the review text obtained did not have readily available information giving features. Therefore, we cleaned the text and brought it into more informative format. We took the following steps:
    - Removed all the punctuation and numbers from text using regular expressions.
    - Stemmed all the words so that words like 'eat' and 'eating' are not treated differently
    - Eliminated all the stop words which are common to most of the documents and does not give any information
    - For the remaining words, computed the TFIDF score for each word in a review.

5. Following methods were used to improve the RMSE:

- Using TFIDF metrics for each restaurant: On yelp website, Users write reviews for each restaurants describing their experience at the restaurant. They write about quality of food, ambience, time taken to serve food, hygiene, parking and various other things
    - Cutting off the size of the word metrics by term frequency to capture the most important words. Top 120 words were used.
    - N-grams: They are more descriptive of a text than single words.
    - Using lexical knowledge of the restaurant: We chose only those few words that are relevant to the context of restaurants and removed other words manually. Following words were removed: 'this', 'this place', 'things', 'since', 'says' 'said', 'look', 'has', 'did', 'didn't', 'done', 'don't', because', 'any', 'asks', 'very', 'went', 'was very'.
- After calculating TFIDF, Vector Space Model is used to determine which items are closer to each other. It calculates the proximity based on the angle between the vectors
- Cosine of angle is used because value of cosine will increase with decreasing value of the angle which signifies more similarity
- Used the restaurants attribute like:

- Accepts credit card yes/no
- Serves alcohol yes/no
- Ambience good/bad
- Parking yes/no
- Price Range High/Medium/Low
- Delivers food yes/no
- Noise level: High/Medium/low
- Waite Service Good/Bad
- Outdoor seating yes/no

6. Evaluation:

Root Mean Square Error (RMSE) is used to evaluate the models. This is a good criteria of evaluation because our target variable is Ratings which is continuous

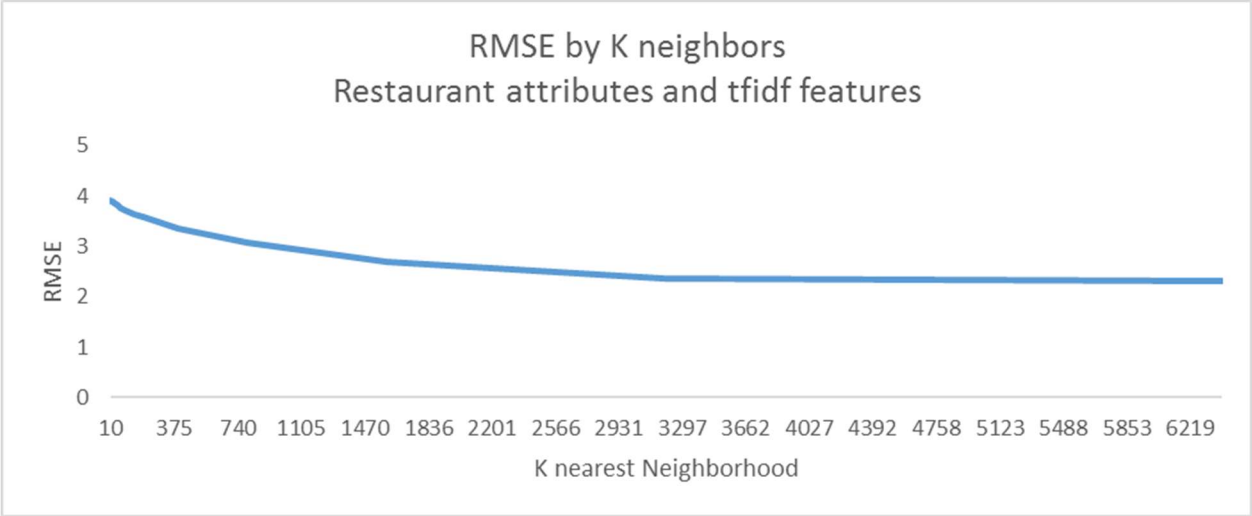
- Recommendation system created using different values of 'Item neighborhoods' ranging from 5 to 10 (see figure below). For each item, we hold this many similar items to use when aggregating models for predictions. Decreasing this value decreases the memory required by the model and decrease the time required to generate recommendations, but it may also decrease recommendation accuracy
- Below is one iteration of RMSE:

```
train, test = gl.recommender.util.random_split_by_user(observation_data_lv_sf,item_id = "business_id", user_id = "user_id", max_nu
m = gl.recommender.item_content_recommender.create(item_lv_wo_tfidf_sf,observation_data = train, target= "stars_x",item_id="busine
m.evaluate_rmse(test, target = 'stars_x')
```

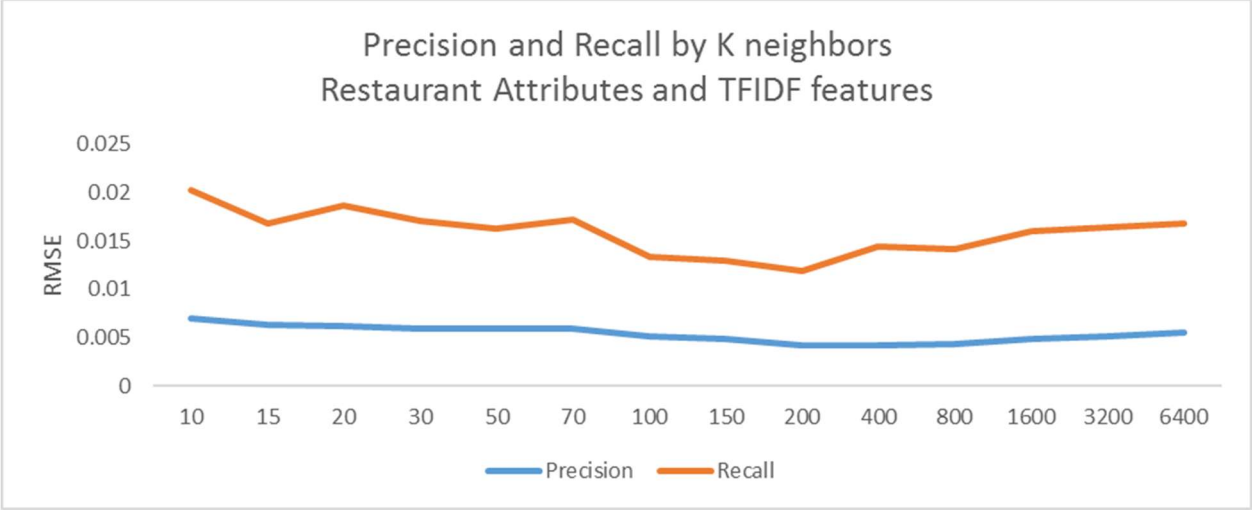
```
Data:
+------------------------+-------+----------------+
|        user_id         | count |      rmse      |
+------------------------+-------+----------------+
| -UdfeFtBW1AnS535073MGA |   2   | 2.68416089293  |
| 6wmjOWQ5n4zEdpokD06gdw |   1   | 2.59347328544  |
| B_3nQiMH0hkDOJ_i7RZ0pg |   1   | 4.47367259502  |
| WYaR96mI3LkE151qY4BTMg |   1   | 3.47899838448  |
| pPxx259yE48IRZ7mrOCTvQ |   1   | 2.19156694412  |
| CP3ZQ7Sg_-MFjP53LFjjpA |   1   | 4.58233002901  |
| buegj3RydlucLj_ZqIdhOQ |   3   | 0.513806277668 |
| _69GhRKU9qc9_a16BkjItQ |   1   | 0.18248796463  |
| UasvQNFRK5_cnlP9M3VndQ |   1   | 2.55946368217  |
| ucFbQYSnUO3fcNwI7nKZfQ |   1   | 4.51776607037  |
+------------------------+-------+----------------+
[3381 rows x 3 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.,
'rmse_overall': 2.3717204705664625}
```

- In order to improve the model, we trained and tested the model for different values of K nearest neighborhoods and also using feature selection:

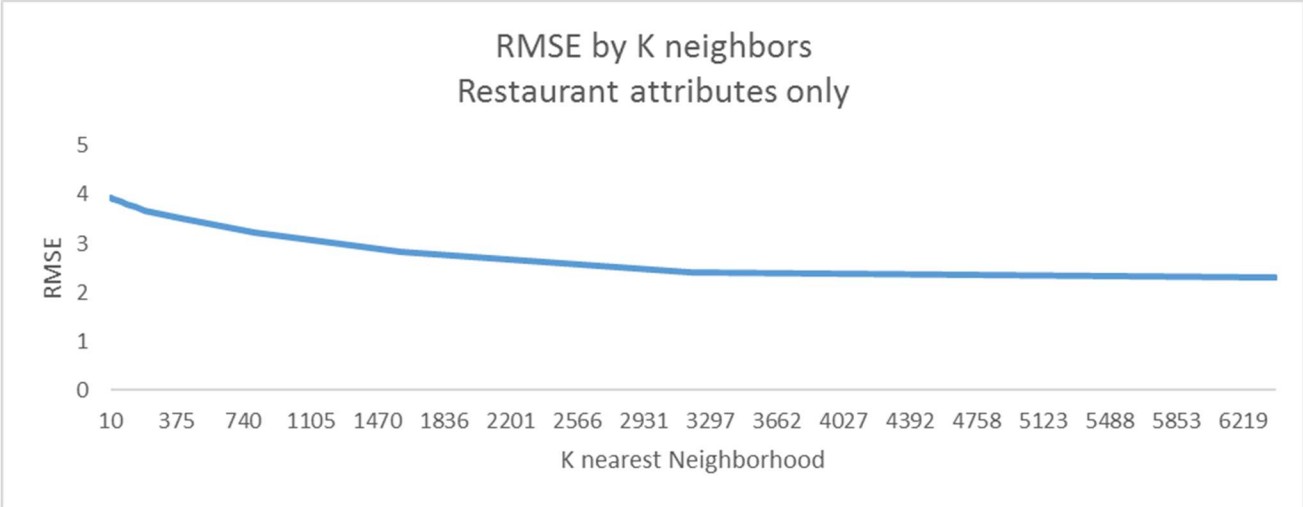
CASE 1: Taking all the attributes of restaurants and TFIDF features

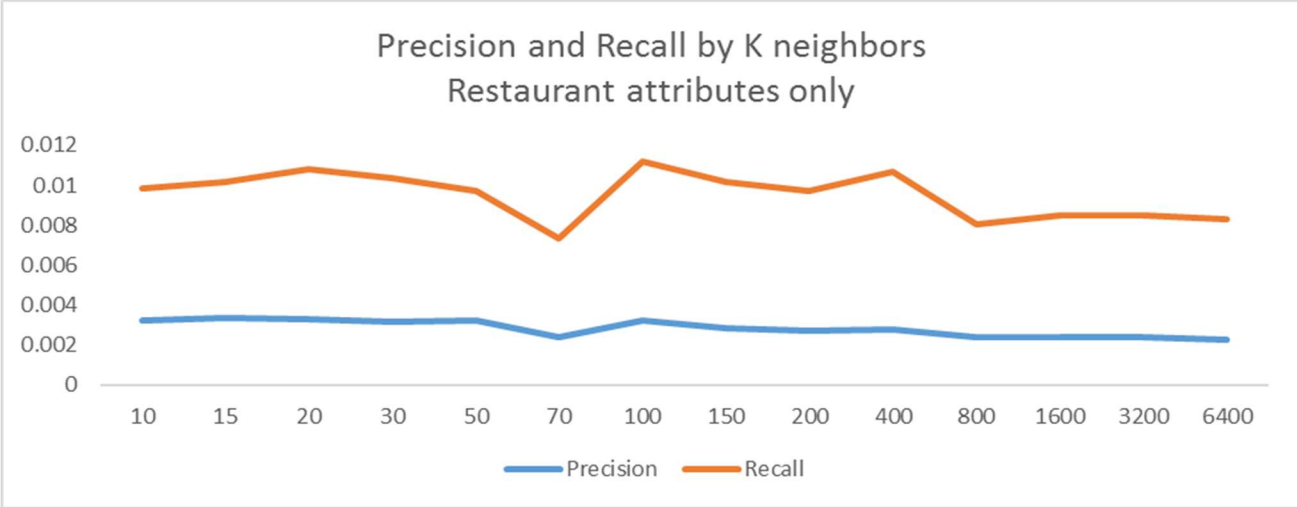


This shows that the RMSE value decreases for a K value of 2,500. Then it saturates and becomes constant.

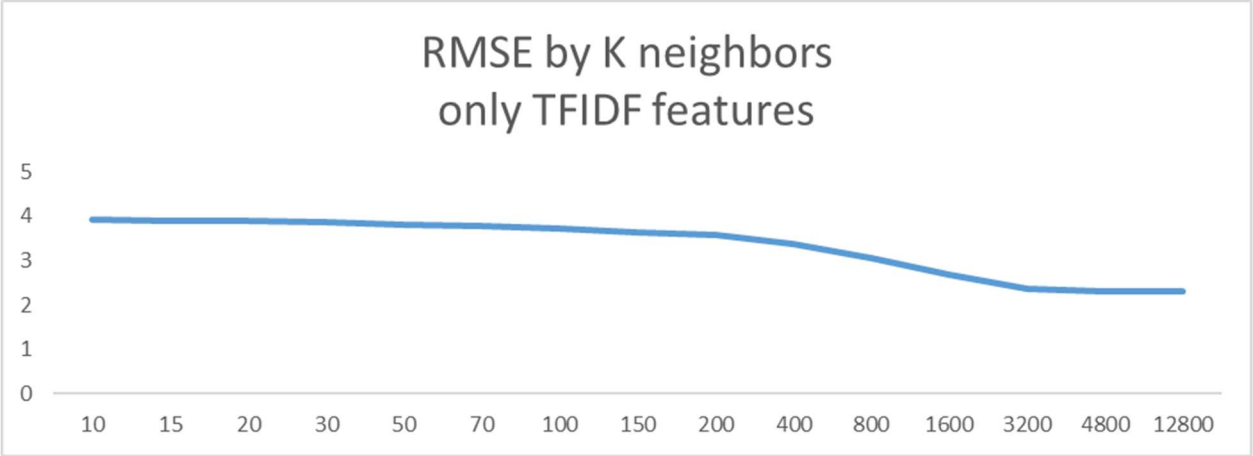


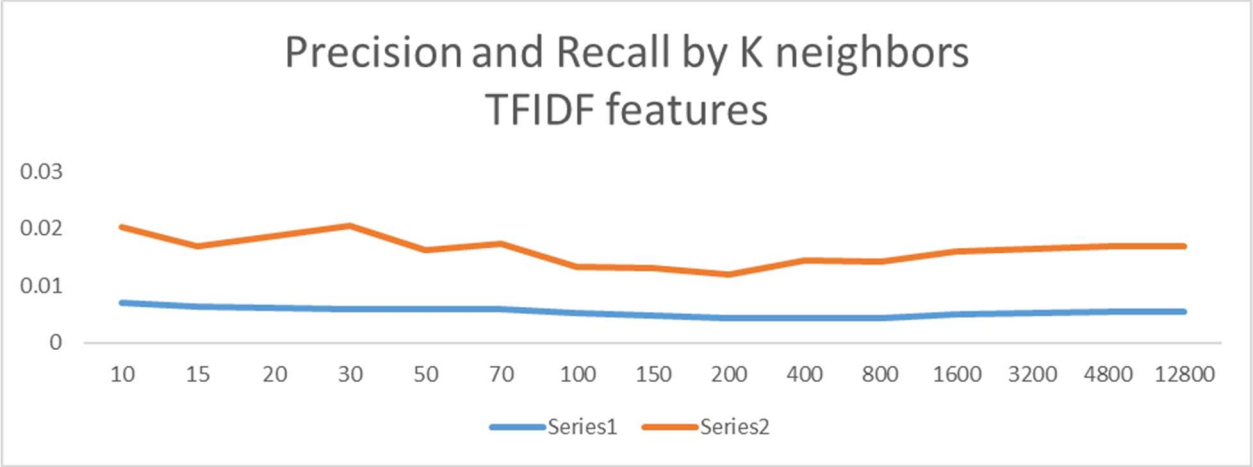
CASE 2: Taking only restaurant attributes





Case 3: Taking only TFIDF features





7. Results and Summary:

- The best RMSE obtained is 2.28
- After trying out different, the best RMSE obtained was with all the restaurant features, which includes restaurant attributes like parking, ambience, service, delivery etc., with TFIDF features.
- The best model was obtained at K value of 2,600 which essentially means that 2,600 nearest items are considered and averaged to get the
- We get the list of item recommendation for each user, ranked based on score (ratings) that he would recommend.

8. Limitations:

- Semantic meaning of the words is not considered. Negative or positive context of a word is considered to be same
  For example: user1 writes: “This is a delicious vegetarian restaurant” and user2 writes: “Non vegetarian food tastes bad”. In such a case the word “vegetarian” is being used in different context but our model considers it to be same
- We are assuming that content/attribute of each item contains enough information to discriminate them.
- If user has not reviewed/liked many restaurants, then we cannot build a solid profile for that user, which could lead to inaccurate recommendations

9. What we could have done?

- Given more weightage to user ratings of a restaurant whose review had more useful, funny and cool votes
- Item data transform: Feature engineering transformer that transforms the item data into a form in which calculating distances between feature is meaningful

5. Matrix Factorization method

1. About the Model

As the name clearly suggests, matrix factorization method is about breaking down the user item interactions which is the data-frame ‘**review_LV_norm’** or in algebra terms factorize the user-item interaction matrix. The aim is to find out two or more matrices which interact or multiply to form the original matrix.

EXB 1 : Snapshot of the data-frame (Restaurant_recommender_systems.html Sec 2.3)

```
# Take a sneak peak into the data
review_LV_norm.head(2)
```

	business_id	user_id	stars	name	avg_stars_user	user_normalized_stars	avg_stars_items	item_normalized_
0	JNArUYI2aVBHNNi6hXcG8g	f5vgLcoKpFcTvD4lOUxcTQ	2	Rigos Taco	3.355372	-1.355372	3.529412	-1.529412
1	JNArUYI2aVBHNNi6hXcG8g	MWw9-Eo1AQuJpdTbZ13vlw	5	Rigos Taco	4.000000	1.000000	3.529412	1.470588

Given that each user’s have rated some restaurants (items in our case), we want to predict how the users would rate the restaurants that they have not yet rated. This will help us generate recommendation based on how high a user would rate the restaurant such that we can make

recommendations to the users. In this case, all the information we have about the existing ratings can be represented in a matrix.

Let us take a toy example to understand this more clearly. Let us assume that we have 5 users and 10 restaurants in a town. The rating a user can give to each restaurant are integers ranging from 1 to 5.

	D1	D2	D3	D4
U1	5	3	-	1
U2	4	-	-	1
U3	1	1	-	5
U4	1	-	-	4
U5	-	1	5	4

The task of predicting the missing ratings can be considered as the solution to this problem, taking an assumption that the behavior of each user is consistent across the restaurants they have already rated.

The intuition behind using matrix factorization is that we want to observe some latent features which are hidden and drive the user preference for a specific restaurant. Features like the quality of food, cleanliness, speedy delivery and customer-service can be some latent features which are not directly observed in the data but drive the rating given by the user.

Hence, if we can discover these latent features, we will be able to profile the taste of the users towards certain aspects of a restaurant he considers important while giving ratings to the restaurant. The idea is that the features associated with the user would match the features associated with the restaurant for a high rating.

2. How the algorithm works

Our task, then, is to find two matrices $\mathbf{P}$ (a $|U| \times K$ matrix) and $\mathbf{Q}$ (a $|D| \times K$ matrix) such that their product approximates $\mathbf{R}$:

$$\mathbf{R} \approx \mathbf{P} \times \mathbf{Q}^T = \hat{\mathbf{R}}$$

Rhat is the predicted matrix which is the product of two matrices **P** and $\mathbf{Q}^T$. The algorithm uses gradient descent technique to find the global minimum and tries to optimize the loss function which is the RMSE.

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2$$

3. Why this model

The advantage of factorizing the interaction data into two or more matrices is to discover latent features underlying the interactions between two different kinds of entities. We can also consider more than two kinds of matrices like one for user, one for item and one which expresses the timing of the interactions to make more accurate predictions (tensor factorization, a more complex approach).

4. Assumptions of the model

While discovering the latent features, we also assume that the number of latent features would be less than the number of users and the number of items. This is because it would not be reasonable to assume that each user is associated with a unique feature and if this is the case there would be no point in making recommendations, because each of these users would not be interested in the items rated by other users. The same argument applies to the items.

5. How to implement this model in python.

- For implementing this we first take the user- item interaction data 'review_LV_norm' as shown in exhibit EXB 1.
- Now we can use the train test technique or cross validation technique to train and test the model.
- We then use scikit learn functions or cross_validation implementation in graphlab
- Once we have the training set ready we can use the matrix factorization implementation of graph lab to run the matrix factorization algorithm. The code for this is in exhibit EXB 2
- The graphlab implementation allows us to select the number of latent features and the regularization parameters which are used to control the co-efficient of the linear model which is used to estimate the missing value of a user item rating based on the product of two factorized matrices.

EXB 3 : Snapshot of matrix factorization model (Restaurant_recommender_systems.html Sec 3.5.1)

```
In [161]: fctr_model = gl.recommender.factorization_recommender.create(observation_data = review_LV_norm_sf_train,\
                                                    user_id='user_id',item_id='business_id',target ='stars',num_factors=12,\
                                                     regularization=0.00000001 ,linear_regularization= 0.00000001,verbose= False)
          fctr_model.evaluate(review_LV_norm_sf_test)
```

6. How to evaluate the model.

- There are two ways we can evaluate our model by split validation and cross validation.
- Cross validation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds.
- Split Validation works by splitting given dataset into training data and test data so that we can train out models on the training data and validate our model against unseen test data to measure the performance.
- In this case we have used a simple split validation to observe the performance of the model to save time and iterate over parameters which can improve the model performance.
- The metrics of interest are RMSE, precision and recall.
- RMSE is the root mean square error which is the loss function which the algorithm tries to minimize. This metric is calculated by taking the difference of the predicted values from the actual value and taking a square of that and then the mean of all such errors for all the predicted values and then taking the square root .
- The use of RMSE is very common and it makes an excellent general purpose error metric for numerical predictions.
- Compared to the similar Mean Absolute Error, RMSE amplifies and severely punishes large errors.]

$$\text{RMSE}= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\mathbf{yi}-\mathbf{y\hat{}i})\mathbf{2}}$$

- Precision and Recall In pattern recognition and information retrieval with binary classification, precision (also called positive predictive value) is the fraction of retrieved instances that are relevant, while recall (also known as sensitivity) is the fraction of

relevant instances that are retrieved. Both precision and recall are therefore based on an understanding and measure of relevance[Wikipedia].

7. How to improve the model performance.

- Number of latent factors can be changed to improve the model performance.
- L2 Regularization we can also change the regularization for the interaction terms.
- Linear regularization is changing the L2 regularization term linear term of the user item interaction equation.
- Below is the implementation of the model improvement procedure in python where we iterated serval values of Number of latent features and the regularization terms to get the best RMSE.

EXB 3: Snapshot of the matrix factorization model with multiple iterations (Restaurant_recommender_systems.html Sec 3.5.1)

```
In [135]: for num_face in np.arange(10,20,2):
              fctr_model = gl.recommender.factorization_recommender.create(observation_data = review_LV_norm_sf_train,\
                                                    user_id='user_id',item_id='business_id',target ='stars',num_factors=num_face,verbos
          e= False)
              print (fctr_model.evaluate(review_LV_norm_sf_test))
```

8. Model Performance.

- The below table shows the RMSE, precision and recall while iterating through various parameters.

EXB 4: Table 1 and Table 2 with performance over multiple iterations.

Table 1	INPUTS Parameters				Evaluation Parameters		
Models	CV	L2 Regularization on Factors	L2 Regularization on Linear Coefficients	Number of latent factors.	RMSE	Precision at 10	Recall at 10
1	5	0.00000001	1E-10	10	1.3070	0.004315	0.027743
2	5	0.00000001	1E-10	12	1.2952	0.006504	0.043089
3	5	0.00000001	1E-10	14	1.2954	0.005776	0.039113
4	5	0.00000001	1E-10	16	1.3008	0.006165	0.041791
5	5	0.00000001	1E-10	18	1.3099	0.006192	0.041918
6	5	0.00001	1E-10	12	1.2946	0.0050	0.0322
7	5	0.000001	1E-10	12	1.2968	0.0053	0.0355
8	5	0.0000001	1E-10	12	1.2945	0.0058	0.0379

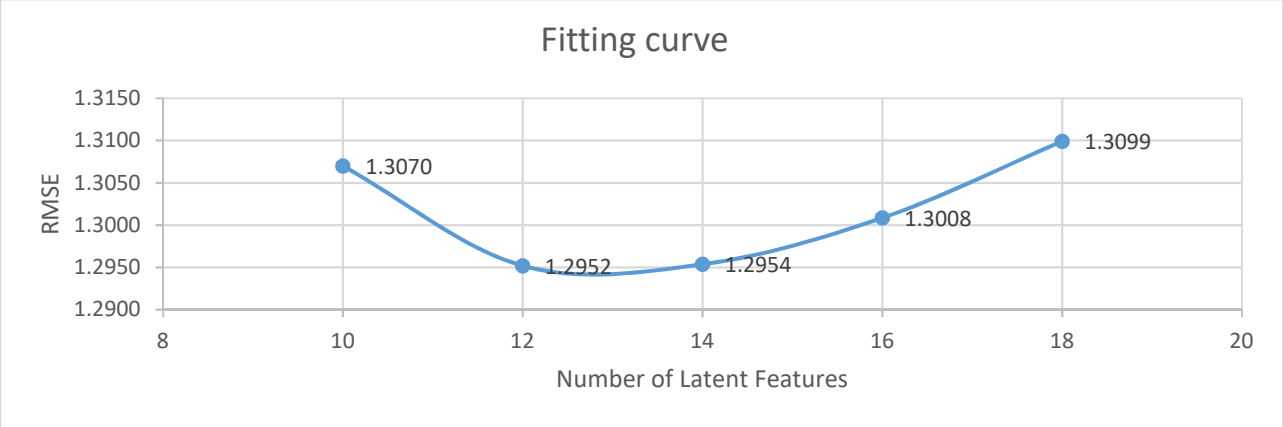
9	5	0.0000001	0.000000001	12	1.2948	0.0069	0.0460
10	5	0.0000001	0.00000001	12	1.2949	0.0060	0.0401

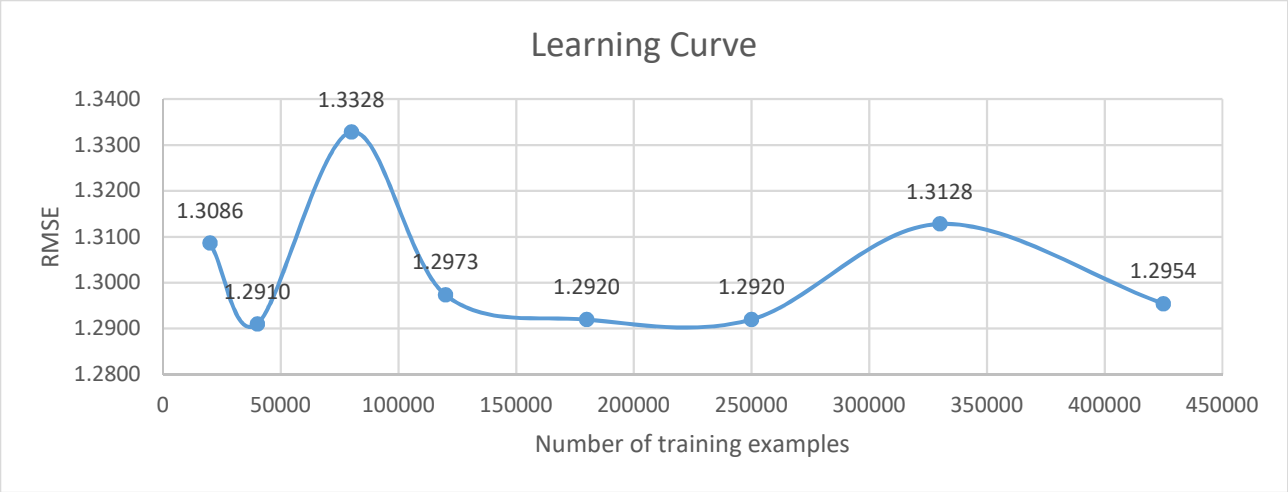
Table 2

Models	No of training examples	RMSE	Precision at 10	Recall at 10
3	20000	1.3086	0.004595605	0.031478
4	40000	1.2910	0.005598614	0.037394
5	80000	1.3328	0.007533712	0.051023
6	120000	1.2973	0.006734344	0.04394
7	180000	1.2920	0.007540804	0.051168
8	250000	1.2920	0.007283467	0.049863
9	330000	1.3128	0.00386817	0.026697
10	424981	1.2954	0.006504362	0.043089

- The fitting and the learning curves for this model are below. We see that the fitting curve has the least RMSE when the number of latent features are 12. Similarly we see that the learning curves perform pretty consistent with the total dataset as well.

EXB 5: Figure 1 and Figure 2 are fitting and learning curves respectively.





9. Limitations

- The latent features the algorithm assumes may not be valid for the data.
- The algorithm is resource intensive.
- The algorithm tries to minimize the RMSE by starting with a random value for the two factor matrices and applies gradient descent to arrive at the best results. Hence the algorithm may just reach the local minima.
- If the step size is too large the algorithm may miss the global minimum and never yield an optimal result.
- Cold start problem.

10. Results

- We have tried to keep the code file as an HTML file as that becomes too large when we create a pdf or word format and hence we are providing the reference to certain parts of the code which will help us keep the document crisp and save some space.

EXB 6: The results from the matrix factorization model.(Restaurant_recommender_systems.html Sec 3.5.1)

6. Matrix Factorization method with Factorization machines (regular and ranking)

1. About the Model

- The algorithm of this model is same as the matrix factorization model, however this model additionally learns latent factors for each user and item and uses them to make rating predictions. This includes both standard matrix factorization as well as factorization machines models

- Here for this model we are providing the user information along with user attributes as one of the side information.
- The difference between regular and ranking is that regular uses the data from user-item interactions and ranking builds a model when we do not have the values for user item interactions.

EXB 7: User data.(Restaurant_recommender_systems.html Sec 2.4)

```
In [28]: user.head(2)
```

Out[28]:

	average_stars	compliments	elite	fans	friends	name	review_count	type	user_id	votes	yelping_
0	4.14	{u'profile': 8, u'cute': 15, u'funny': 11, u'p...	[2005, 2006]	69	[rpOyqD_893cqmDAtJLbdog, 4U9kSBLuBDU391x6bxU-Y...	Russel	108	user	18kPq7GPye-YQ3LyKyAZPw	{u'funny': 167, u'useful': 282, u'cool': 246}	2004-10
1	3.67	{u'profile': 117, u'cute': 204, u'funny': 594,...	[2005, 2006, 2007, 2008, 2009, 2010, 2011, 201...	1345	[18kPq7GPye-YQ3LyKyAZPw, 4U9kSBLuBDU391x6bxU-Y...	Jeremy	1292	user	rpOyqD_893cqmDAtJLbdog	{u'funny': 8399, u'useful': 15242, u'cool': 12...	2004-10

- We are also providing the restaurant attributes in the side information for this model.

EXB 8: Restaurant data.(Restaurant_recommender_systems.html Sec 2.2)

```
In [68]: restaurants_LV_sf.head()
```

Out[68]:

JNArUYI2aVBHNNI6hXcG8g	Rigos Taco	21	3.5	65	1	none
eihJZqHCEKMmzuPOOyqF9w	Mulligans Bar & Grill	10	4.0	28	1	full_bar
n1f9rWJzdGX982bFvZPBHg	Carl's Jr Restaurants	7	2.5	39	1	none
i7HvZ1yTTIaR6bfa_3jGEw	KFC	8	2.0	6	1	none
NkmE03gbKjSdqDM3SJsJmw	Church's Chicken	3	1.5	3	1	none
KBP-RXCaDQwfMP_-TLwuFg	Sonic Drive-In	37	3.0	218	1	none
hvVckRvOa814ry8q8WGwOg	Burger King	4	3.5	37	1	none
THEsYolDkHaJofZLs6ufWg	Jack In the Box	12	3.0	74	1	none
rG_8t33D2ADcbBqcXqwV0Q	Viva Zapatas Mexican Restaurant & Cantina ...	443	4.0	990	1	full_bar
_OyxUcWkSTRvRKFyEE7zbA	Tortas El Rey	11	3.0	21	1	none

Ambience	Delivery	Good For	Parking	Price Range	Take-out	Takes Reservations	Waiter Service	Noise Level
1.0	0	1.0	1.0	1.0	1.0	0	0	average
1.0	0	0.0	1.0	2.0	1.0	1	1	average
nan	0	0.0	0.0	1.0	1.0	0	0	average
nan	0	nan	nan	1.0	1.0	0	0	very_loud
nan	0	0.0	0.0	1.0	1.0	0	0	average
1.0	0	1.0	1.0	1.0	1.0	0	1	average
0.0	0	0.0	0.0	1.0	1.0	0	0	average
0.0	0	0.0	nan	1.0	1.0	0	0	quiet
1.0	0	1.0	1.0	2.0	1.0	1	1	average
1.0	0	1.0	1.0	1.0	1.0	0	0	average

Outdoor Seating
1
0
0
0
0

- These side attributes will help the model better understand the user item relations ships by trying to model based on the user attributes and the restaurant attributes as the features for predicting the score a user may give to each restaurant.

2. How to implement this model in python.

- For implementing this we first take the user- item interaction data ‘review_LV_norm’ as shown in exhibit EXB 1.

- Now we can use the train test technique or cross validation technique to train and test the model.We then use scikit learn functions or cross_validation implementation in graphlab
- Once we have the training set ready we can use the matrix factorization implementation of graph lab to run the matrix factorization algorithm. The code for this is in exhibit EXB 2
- The graphlab implementation allows us to select the number of latent features and the regularization parameters which are used to control the co-efficient of the linear model which is used to estimate the missing value of a user item rating based on the product of two factorized matrices.

EXB 9 : Snapshot of matrix factorization model with user and restaurant side features (Restaurant_recommender_systems.html Sec 3.6.1 and 3.7.1)

```
In [72]: mtrx_fctr_model = gl.recommender.factorization_recommender.create(observation_data = review_LV_norm_sf_train,\
                                                user_id='user_id',item_id='business_id',target ='stars',num_factors=12,\
                                                user_data= user_lv_sf_mf,\
                                                 item_data = restaurants_LV_sf_mf ,verbose= False)
```

```
In [76]: rnkng_fctr_model = gl.recommender.ranking_factorization_recommender.create(observation_data = review_LV_norm_sf_train,\
                                               user_id='user_id',item_id='business_id',target ='stars',num_factors=12,\
                                                                        user_data= user_lv_sf_mf,\
                                                item_data = restaurants_LV_sf_mf,verbose= False)
```

3. How to improve the model performance.

- We have used the best matrix factorization model from the above iteration and use the side features on that.
- We have used 5 fold cross validation and taken the mean RMSE, precision and Recall.

EXB 10: Snapshot of the matrix factorization model with side features (Restaurant_recommender_systems.html Sec 3.6.3)

```
In [116]: # Evaluate the model on Cross validation
          folds = gl.cross_validation.KFold(review_LV_norm_sf, 5)
          params = dict([('user_id', 'user_id'), ('item_id', 'business_id'), \
                         ('target', 'stars'), ('user_data', user_lv_sf_mf), ('item_data', restaurants_LV_sf_mf)])

In [117]: job = gl.cross_validation.cross_val_score(folds,gl.recommender.factorization_recommender.create,params)

          [INFO] graphlab.deploy.job: Validating job.
          [INFO] graphlab.deploy.job: Creating a LocalAsync environment called 'async'.
          [INFO] graphlab.deploy.map_job: Validation complete. Job: 'Cross-Validation-Dec-04-2016-20-29-19-127000-50250985' ready for exec
          ution
          [INFO] graphlab.deploy.map_job: Job: 'Cross-Validation-Dec-04-2016-20-29-19-127000-50250985' scheduled.

In [118]: print (job.get_results())
```

4. Results

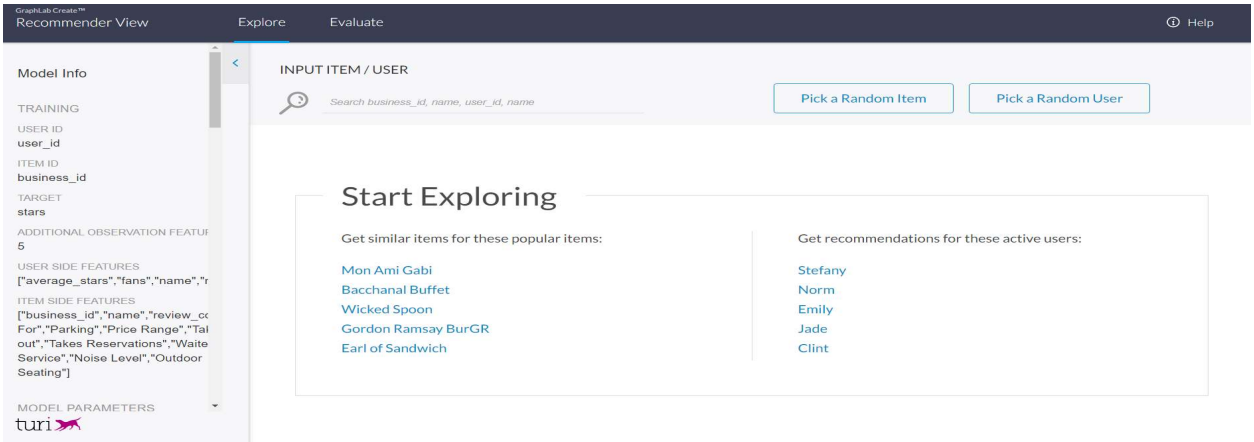
- We have tried to keep the code file as an HTML file as that becomes too large when we create a pdf or word format and hence we are providing the reference to certain parts of the code which will help us keep the document crisp and save some space.

EXB 11: The results from the matrix factorization model.(Restaurant_recommender_systems.html Sec 3.6.3)

5. Deployment

- Here we have used graphlab canvas to deploy and visualize our model.

EXB 11: Visualization of the model (Restaurant_recommender_systems.html Sec 3.6.4)



EXB 12: Visualization user recommendations (Restaurant_recommender_systems.html Sec 3.6.4)

Focus User

	name	user_id	average_stars	compliments	elite	fans	friends	review_count
	Nadine	43OUvOgkOlU\	5	{"hot":1}	[]	0	["E6mBK8VJpT91...	1

Top Recommended Items

Score	name	business_id	rank	review_count	stars	tot_checkins	Acc_credit_card	Alcohol
2.64	Bacchanal Buffet	slyHTizqAiGu12	1	5216	4	17256	1	full_bar
2.58	Jaleo	cN6qpVw6RaSr! JGGF4Tw	2	944	3.5	1927	1	full_bar
2.27	BURGER BRASSERIE	xnRljkaR6zaVTp	3	536	3.5	973	1	full_bar
2.17	Diablo's Cantina	yJDSMugu_Pom	4	914	3	3320	1	full_bar
2.07	Yonaka Modern Japanese	HpaYCM_NCaul	5	747	4.5	2314	1	beer_and_wine
2.02	Bachi Burger	t6SuvEq9PPVGz	6	946	4	3377	1	beer_and_wine

EXB 13: Visualization item-item similarity (Restaurant_recommender_systems.html Sec 3.6.4)

INPUT ITEM / USER

Search business_id, name, user_id, name

Pick a Random Item | Pick a Random User

Focus Item

	name	business_id	review_count	stars	tot_checkins	Acc_credit_card	Alcohol	Ambience
	Las Gorditas Los	OfSLGavUS3J k	6	3.5	9	1	none	0

Top Similar Items

Score	name	business_id	rank	review_count	stars	tot_checkins	Acc_credit_card	Alcohol
1.00	El Tenampa	Du3ESDGEGz6!	1	8	2.5	34	1	beer_and_wine
1.00	Quiznos Sub	8AoHXhEnpfqB(	2	5	3	7	1	none
1.00	Chow Time	jwovjTwrBAhid4	3	7	3	4	1	none
1.00	Dino's Little Italy	pNcV8xFwAxNr	4	4	2.5	12	1	none
1.00	Grand China Fine Chinese Cuisine	TUxCIJ6Jj2IEOc	5	12	3	7	1	beer_and_wine
1.00	L.A. Italian Kitchen	3ojGEFVvMY2V	6	12	3	12	1	none

INPUT ITEM / USER

Search business_id, name, user_id, name

Pick a Random Item | Pick a Random User

Focus Item

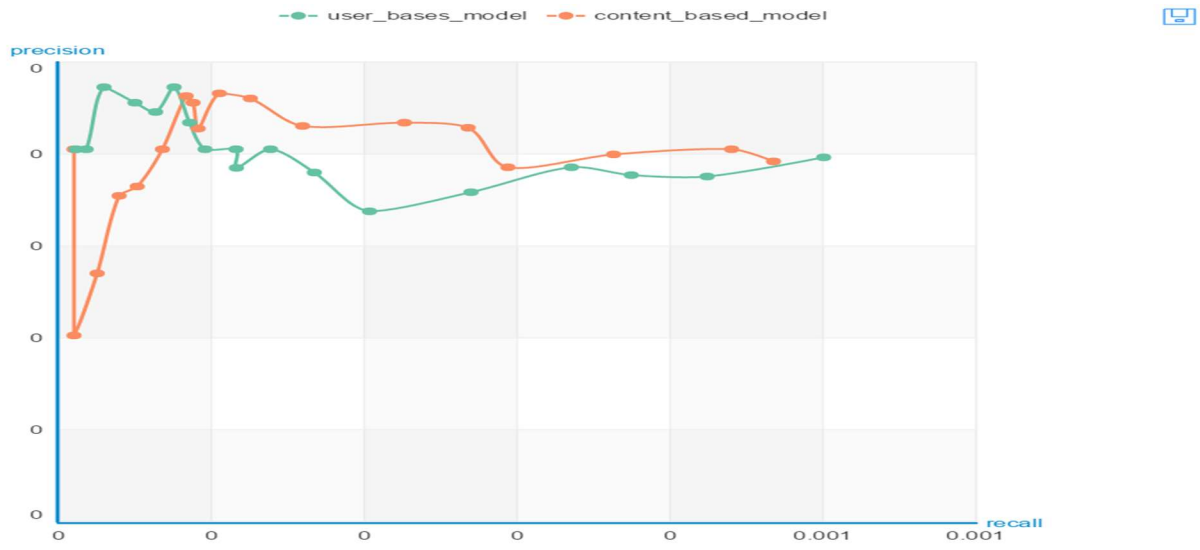
	name	business_id	review_count	stars	tot_checkins	Acc_credit_card	Alcohol	Ambience
	Mon Ami Gabi	4bEjOyTaDG24	6200	4	12778	1	full_bar	1

Top Similar Items

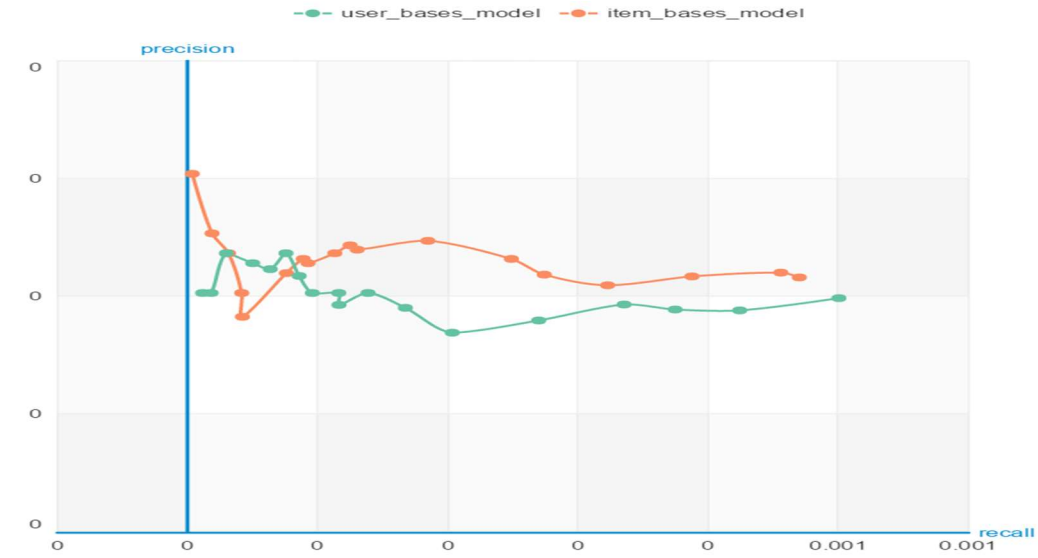
Score	name	business_id	rank	review_count	stars	tot_checkins	Acc_credit_card	Alcohol
0.97	Wicked Spoon	zt1TpTuJ6y9n5	1	4967	3.5	14302	1	full_bar
0.92	Earl of Sandwich	2e2e7WgqU1Br	2	4687	4.5	17613	1	beer_and_wine
0.90	Serendipity 3	Xhg93cMdemu5	3	3781	3	10721	1	full_bar
0.86	Taco Bell	84JrJfQPA9SLk	4	19	3	132	1	none
0.85	Carl's Jr	XQokPEF3UJU2	5	18	3	90	1	none
0.85	Gordon Ramsay BurGR	aGbjLWzcrnEx2	6	4620	4	10680	1	full_bar

6. Model comparison at various values of K

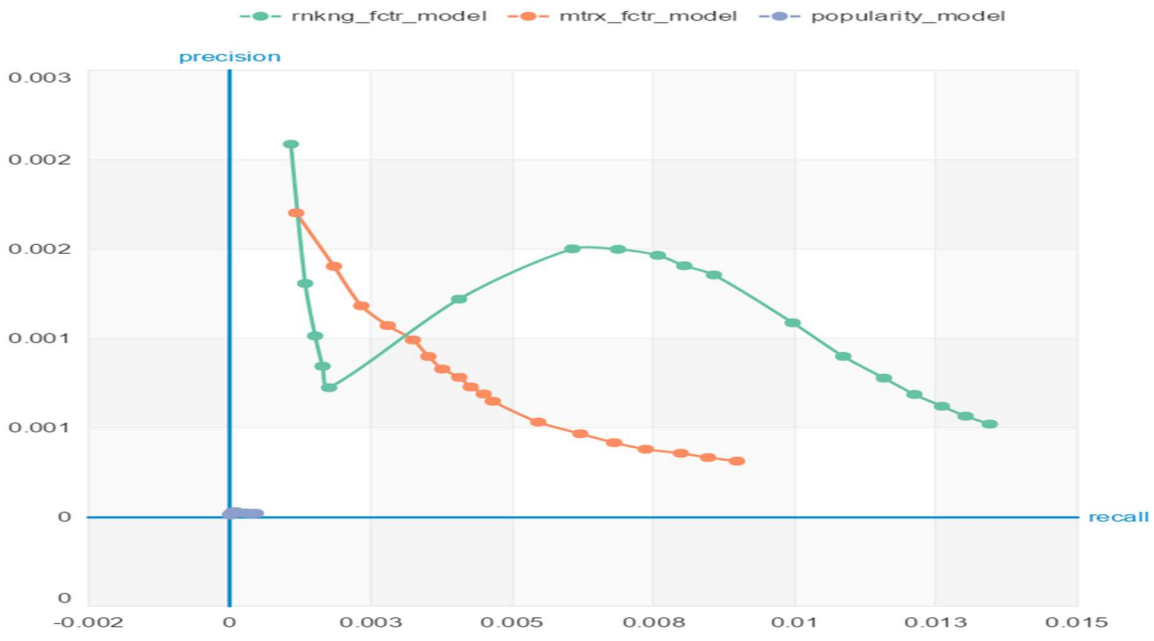
- The user based model seems to perform better than content based at lower values of K in terms of precision and recall. But the content based model performs consistently better at higher values of K.



- The item based model seems to consistently perform better than user based at all values of K.



- The ranking matrix factorization based model seems to consistently perform better than matrix factorization model at based at all values of K. And both these models out-perform the baseline popularity based model by a huge margin.



7. Trust-Aware Recommender System - Recommender Systems and the Next-Generation Web

1. About the Model

- In general, trust-Aware recommender systems aim to exploit the information from friends networks to improve the recommendation performance.
- The hope is that the accuracy of the recommendations can be increased - for instance, by taking the opinions of explicitly trusted neighbors into account instead of using peers that are chosen based only on a comparison of the rating history.[5]

2. Advantages of this model

- In particular, the goal here is also to alleviate the cold-start problem and improve on the user coverage measure - in other words, if no sufficiently large neighborhood can be determined based on co-rated items.[5]
- The opinions of trusted friends can serve as a starting point for making predictions. In addition, one conjecture when using explicit trust networks is that they help make the recommender system more robust against attacks, because desired "incoming" trust relationships to a fake profile cannot easily be injected into a recommender database.[5]

3. How to implement this model in python.

- For implementing this we first take the user- item interaction data ‘review_LV_norm’ as shown in exhibit EXB 1.
- Now we can use the train test technique or cross validation technique to train and test the model. We then use scikit learn functions or cross_validation implementation in graphlab.
- Once we have the training set ready we can use the matrix factorization implementation of graph lab to run the matrix factorization algorithm. The code for this is in exhibit EXB 2
- The graphlab implementation allows us to select the number of latent features and the regularization parameters which are used to control the co-efficient of the linear model which is used to estimate the missing value of a user item rating based on the product of two factorized matrices.

EXB 14 : Snapshot of matrix factorization model with influential users (Restaurant_recommender_systems.html Sec 4.1.1)

In [96]:
```
inf_mtrx_fctr_model = gl.recommender.factorization_recommender.create(observation_data = review_LV_norm_sf_train,\
                                                                      user_id='user_id',item_id='business_id',target ='stars',num_factors=12,\
                                                                      user_data= user_lv_sf_mf,\
                                                                       item_data = restaurants_LV_sf_mf ,verbose= False)
```

- Here we are selecting the most influential users based on the metrics like they have more than average number on metrics like fan, useful, funny and cool votes, reviews etc.
- Based on this model we will generate recommendations for the influential candidates and use their recommendations to get common recommendations from their followers (on the candidates in their friends list who are not influential)and show them with highest preference. Let us go through an example to see how it works.

EXB 15: Generating recommendations for Helen and influential user and Nicole a non-influential user and a follower/friend of Helen.(Restaurant_recommender_systems.html Sec 4.2.1).

In [151]:
```
# Let us explore the recommendations we made for the influencer Helen and the follower Nicole
print 'Nicole\'s Recommendations'
print (Nicole_recomm_rest)
print '-------------------------------------------------------------------------------------------'
print 'Helen\'s Recommendations'
print (helen_recomm_rest)
```

```
Nicole's Recommendations
2317                              El Ausente
2329                            Tortas El Rey
4724                         Church's Chicken
4769                                     KFC
6442                     Great Steak & Potato
6463                              Very Venice
6568                                     KFC
12124                    Little Caesars Pizza
14709                        DQ Grill & Chill
15910              Rockin' Taco Mexican Grill
16287                            Chicago Dogs
16982                                    KFC
17074                                    KFC
17362                        Pan Asian Express
18052                    Meat Chix And Wieners
19276                        Pan Asian Express
19701    Triple 7 Restaraunt and Microbrewery
19844                    El Canelo Restaurant
22519                          Pattys Tamales
23206                    Moe's Southwest Grill
Name: name, dtype: object
-------------------------------------------------------------------------------------------
Helen's Recommendations
2407                         Arby's
2516                            KFC
4397          Little Caesars Pizza
5422                  Feast Buffet
6351                        Subway
6456                       Denny's
7109     MORE The Buffet at Luxor
7273             Cafe at Harrah's
9727                  Feast Buffet
13013                      Quiznos
13978              Excalibur Pool
14134                           KFC
14358     South Beach Marketplace
15156                    McDonalds
16642            LA Subs & Salads
17074                           KFC
17920                 Burger King
18977              Nathan's Famous
19262               48th and Crepe
19276           Pan Asian Express
```

- Now when we take on the common restaurants of interest to both the candidates we see the following result.

EXB 16: Common recommendations for Helen and influential user and Nicole a non-influential user and a follower/friend of Helen.(Restaurant_recommender_systems.html Sec 4.2.1).

```
In [150]: # the user_id WsPgO3ycZjRp7wFh18JdMQ with name Nicole actually is friends with
          # Helen whom we consider very influential.
          common_rest = Series(list(set(Nicole_recomm_rest) & set(helen_recomm_rest)))
          common_rest

Out[150]: 0      Pan Asian Express
          1                    KFC
          2    Little Caesars Pizza
          dtype: object
```

4. Results

- We have tried to keep the code file as an HTML file as that becomes too large when we create a pdf or word format and hence we are providing the reference to certain parts of the code which will help us keep the document crisp and save some space.

7. Deployment

- We have used graphlab canvas to deploy and visualize our model.

8. What we could have done?

- Given more weightage to user ratings of a restaurant whose review had more useful, funny and cool votes
- Item data transform: Feature engineering transformer that transforms the item data into a form in which calculating distances between feature is meaningful

Ref:

1. http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/

2. https://en.wikipedia.org/wiki/Cross-validation_(statistics)
3. https://en.wikipedia.org/wiki/Precision_and_recall
4. http://www.columbia.edu/~jwp2128/Teaching/W4721/papers/ieeecomputer.pdf
5. **Recommender systems : an introduction Dietmar Jannach 1973-**
6. http://findoutyourfavorite.blogspot.com/2012/04/content-based-filtering.html
7. https://turi.com/products/create/docs/graphlab.toolkits.recommender.html
8. https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/
9. https://en.wikipedia.org/wiki/Yelp
10. http://cs229.stanford.edu/proj2015/301_report.pdf