YELP RECOMMENDER SYSTEM

J Component Project Report for

CSE3019 – Data Mining

Bachelor of Technology

In

ECE with Specialization in Internet of Things and Sensors

By

Rajdeep Debgupta (16BIS0083)

Varun Agarwal (16BIS0002)

Vipul Jindal (16BIS0018)

Anirudh Agarwal (16BIS0179)

Under the guidance of

PROF. BALAMURUGAN R

School of Electronics Engineering

Vellore Institute of Technology, Vellore – 63014

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Contents

Github: https://github.com/passkey04/Yelp-Recommender-System

1 Need (Business Understanding)

1.1 Business Problem

Yelp has a huge potential to generate an additional revenue stream by using the vast amount of data it has. Based on the ratings given by Yelp users to various restaurants, and since a single user does not rate all the restaurants, we can predict the rating that a user would have given to an unrated restaurant using collaborative filtering techniques. These insights would be beneficial for any restaurant to identify customers that potentially like their restaurant. Hence, Yelp can provide the restaurants with the list of the users that are highly likely to visit their restaurants in return for a fee. Moreover, Yelp can also provide a user with a list of restaurants as a recommendation. If the user finds the recommendations accurate, there is a high chance that the user will increasingly use Yelp and rate new restaurants on it. This would again be beneficial for Yelp as it increases Yelp's usage among the users. There is a positive feedback loop as more users will lead to more advertising revenues, more user information, more insights, and eventually more revenue from selling the customer insights to the restaurant.

1.2 An approach to solve the business problem

A recommendation engine that predicts the rating given by the user who previously didn't visited a particular restaurant would be able to address the business problem mentioned above.

A recommendation engine will help Yelp to learn the user's likes and dislikes. Based on the likes and dislikes, the user will be recommended a restaurant. The insights of user's likes and dislikes can be shared with the restaurant so that the restaurant can target appropriate users.

A recommendation engine aims to fill the sparse matrix with rows being the unique users, column being the unique restaurants and the value in the (i,j) th position being the rating given by the ith user to the jt^{h} restaurant. The sparse matrix is called the ratings matrix.

The methods to build the recommendation engine broadly falls in two categories: *Implicit latent feature* learning based models (*Unsupervised*) and *explicit latent featur*e learning based models (*Supervised*).

Latent features are those features that characterizes a user or a restaurant.

1.2.1 Implicit latent feature learning based models (Unsupervised)

To fill the ratings matrix, there are traditional matrix-factorization based methods like singular value decomposition and non-matrix-factorization methods like ratings prediction based on the cosine similarity between restaurants (or users). In this methods, the models implicitly learns the latent features of the users and restaurants and use them to predict the unseen ratings.

1.2.1.1 Singular Value Decomposition

The singular value decomposition helps to get a low-rank approximation of the ratings matrix. This lowrank approximated matrix is not sparse like the ratings matrix and predicts the previously unseen ratings that might be given by a user to a restaurant.

1.2.1.2 Cosine Similarity Based Prediction

Another traditional approach to predict unseen ratings for a restaurant is by comparing the restaurant (the user) to other similar restaurants (users) in the data set and inferring the ratings based on the ratings given the similar restaurants (users).

We estimate the similarity between the restaurants (users) using the cosine similarity. Some modifications to this approach is to correct the ratings matrix for the restaurant (user) biases before finding the similar restaurants (users).

In this project, the singular value decomposition and cosine-similarity based ratings predictions would act as the baseline models.

1.2.2 Explicit latent features learning based models (Supervised)

After implementing the baseline models, the models like Alternating Least Squares based model and Stochastic Gradient Descent model, where we explicitly learn the latent features are implemented.

1.2.2.1 Latent Features Learning using Alternating Least Squares (ALS) Method

In the ALS method, each user is represented by a k-dimensional feature vector where each dimension represents an attribute of the user which is latent. Similarly, each restaurant is also represented using a k-dimensional vector containing k latent features describing the restaurant. These features are learnt by the model and are parameters of the model. Hence, the data instance will be a randomly initialized k-dimensional vector representing the user x_i, a randomly initialized k-dimensional vector representing the restaurant yj, the rating given by the user to the restaurant r_{ij}. The target variable is r_{ij}. The function that predicts rij from xi, yj is a simple dot product between the feature vectors. The loss function will be a mean square loss with L2 regularization on xi, yj since they are the parameters of our model. Given this setup, we can find all xi's and yj's and fill the matrix as a typical supervised learning problem.

1.2.2.2 Latent Features Learning using Stochastic Gradient Descent (SGD) model

In the SGD model, the setting is similar to ALS where the latent feature for each user and restaurant is learned. In addition to ALS, there is an additional parameter that is learned for each user and restaurant. For each user and restaurant, a bias term is also learned. The intuition behind learning this bias term is many a times some user or restaurant tend to give / receive higher ratings on average as compared to others. So to ensure that the latent feature for each user and restaurant is free from such biases, it is learned as a separate parameter. Essentially, the final rating given by a user i to restaurant j is broken into four components: a global bias (average of all the available ratings), a user bias, a restaurant bias and a component dependent on the interaction between user i and restaurant j. To learn the parameters of the model (all the biases and latent feature vectors), stochastic gradient descent is used. Hence, it is called SGD model.

1.2.3 Converting it into a Regression/Classification Problem

Using the latent features learned for each user and restaurant, the matrix completion problem can be converted to a regression or a classification problem that can be dealt with using powerful techniques like Random Forest Regressor, etc. In this project, we have tried with Random Forest Regressor. .

1.2.3.1 Random Forest Regressor Based Model

Random forest regressor tries to learn the non-linear dependency between the user latent vectors and restaurant latent vectors. The target variable is the rating given by user i to restaurant j. It essentially boils down to a regression problem.

1.2.4 Evaluation Metric

The evaluation metric used for comparing the different models is the mean square error (MSE). The mean square error is easy to interpret. The square root of MSE will give the error the recommendation engine makes while predicting an unknown rating. The lower MSE means the model has correctly learned the latent features that characterizes the user and restaurant which can be used by Yelp to generate insights and recommends new restaurants to users and new users to restaurants.

The model selection is based on the best out of sample MSE obtained for a model. The reasons for improvement in the MSE between different models are analyzed and documented in this report.

1.2.5 Scope of the Project

Since the recommendation engine will be used for the recommending restaurants to the users, it doesn't make sense to build a single recommendation engine for all the restaurants across cities. It would be highly unlikely that a user would travel to just visit a restaurant (unless the user is an avid traveler). Hence, mini-recommendation engine needs to be built for each city. For this project, top two cities with highest number of reviews will be selected for building the recommendation engine.

2 Dataset

The 'review.json' and 'business.json' dataset is being used to conduct the analysis, build a recommender system and address the business problem.

2.1 Source

The 'review.json' and 'business.json' dataset were obtained from the Yelp's website (https://www.yelp.com/dataset).

2.2 Description and Exploratory Analysis

The 'review' dataset has following columns: 'review_id', 'user_id', 'business_id', 'stars', 'date', 'text', 'useful', 'funny', and 'cool'. Below is the snippet of the data:

	user_id	review_id	text	votes.cool	business_id	votes.funny	stars	date	type	votes.useful
0	Xqd0DzHaiyRqVH3WRG7hzg	15SdjuK7DmYqUAj6rjGowg	dr. goldberg offers everything i look for in a...	1	vcNAWiLM4dR7D2nwwJ7nCA	0	5	2007-05-17	review	2
1	H1kH6QZV7Le4zqTRNxoZow	RF6UnRTtG7tWMcrO2GEoAg	Unfortunately, the frustration of being Dr. Go...	0	vcNAWiLM4dR7D2nwwJ7nCA	0	2	2010-03-22	review	2
2	zvJCcrpm2yOZrxKffwGQLA	-TsVN230RCkLYKBeLsuz7A	Dr. Goldberg has been my doctor for years and ...	1	vcNAWiLM4dR7D2nwwJ7nCA	0	4	2012-02-14	review	1
3	KBLW4wJA_fwoWmMhiHRVOA	dNocEAyUucjT371NNND41Q	Been going to Dr. Goldberg for over 10 years. ...	0	vcNAWiLM4dR7D2nwwJ7nCA	0	4	2012-03-02	review	0
4	zvJCcrpm2yOZrxKffwGQLA	ebcN2aqmNUuYNoyvQErgnA	Got a letter in the mail last week that said D...	1	vcNAWiLM4dR7D2nwwJ7nCA	0	4	2012-05-15	review	2

Figure 1: Snapshot of the 'review' data

'business_id', 'review_id' and 'stars' are the required column from this dataset.

The 'business' file has the details about each business. Yelp has a lot of businesses of different categories like restaurants, medical care services, auto care services, etc. listed on its website. The reviews present in the 'review' file contains reviews for all the categories. But, the recommendation engine is to be built only for the restaurants. Hence, the 'business' file is used to filter the unwanted reviews from 'review' file.

The 'business' file have several attributes about the businesses listed on Yelp. Some of them are listed below:

```
business_id, categories, city, hours_Friday, hours_Monday, is_open, latitude,
longitude, name, neighborhood, etc.
```

Just to get some idea of the reviews, business categories, and city, few histograms are plotted.

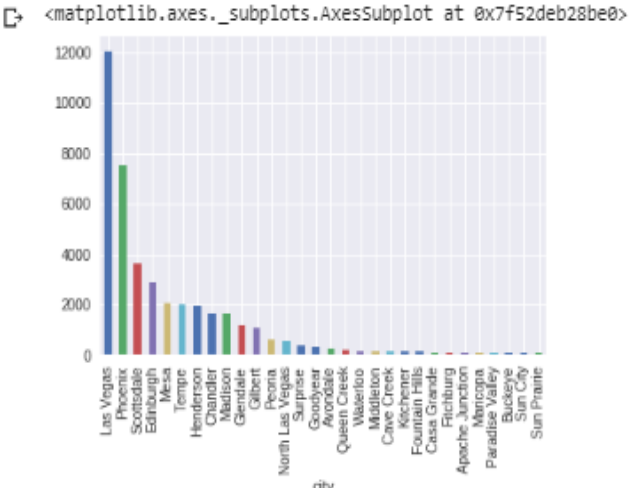


Figure 2: Number of Businesses in each city ('business' file)

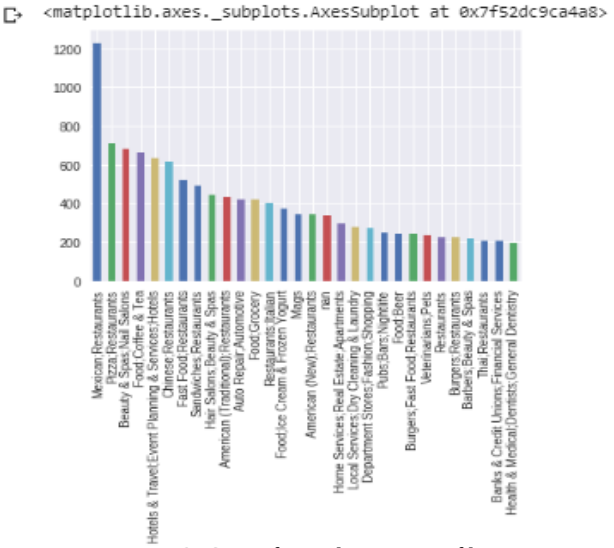


Figure 3: Count for each category of business (in 'business' file)

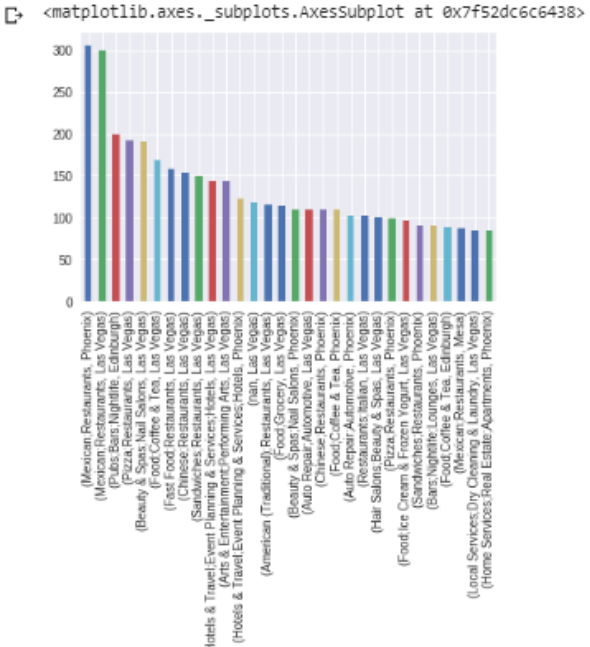


Figure 4: Count of ordered pair of business category and city in the 'business' file

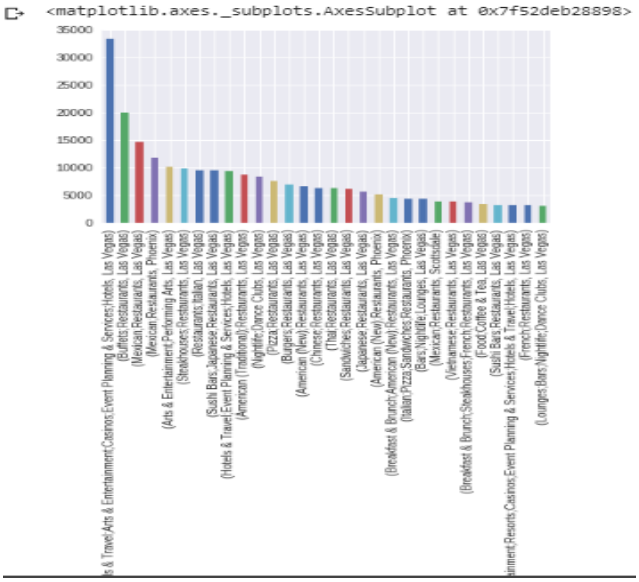


Figure 5: Review count of ordered pair of restaurant and business category in each city

As it can be seen, Las Vegas, Phoenix and Scottsdale have the highest number of unique businesses (Yelp's data set may be only a part of their complete data that they have released hence we don't see popular cities like New York, San Francisco, etc.)

Moreover, restaurant as a business category have the most reviews which is a good news as the recommendation engine would have a lot of reviews from which it can learn the user and restaurant latent features and predict the ratings.
Although the restaurants in Las Vegas have the highest number of reviews, we won't be selecting it as it would be highly probable that the reviews are given by the travelers. Since the travelers are only present in

Github: https://github.com/passkey04/Yelp-Recommender-System

Las Vegas for some time, it don't make sense to recommend them restaurants in Las Vegas once they have returned to their home cities. We would be selecting the restaurants in Phoenix and Scottsdale for building the model.

2.3 Merging and Cleaning datasets, and Creating Ratings Matrix

Merging data files: The 'business' file is merged with the 'review' file to get information about the businesses and filter out reviews obtained from non-restaurants and restaurants not from the selected city (here Phoenix and Scottsdale are the selected cities).

Cleaning the file: After merging and filtering the 'review' file, the file is cleaned to remove the Nan values. There were not many missing values in the dataset.

Filtering the file for minimum number of reviews: The dataset is also filtered to only have users who have at least rated ten restaurants. This ensures that we can divide the data set into 2 reviews per user as a test data, two reviews per user as a validation data and at least 6 reviews per user as a part of training data. *This would also ensure that the sparsity of the matrix is decreased and recommendation engine will have some signals to learn from.*

Train – Validation – Test Split: Training, testing and validation data are all matrices of same size (# of unique users x # of unique restaurants). All the three matrices are disjoint i.e. the entries of test and validation matrix are zeroed out in the train matrix. The element-wise multiplication of three matrices returns a zero matrix.

2.4 Information about the Rating matrix

Characteristics	Restaurants in Phoenix	Restaurants in Scottsdale
Shape	(773, 2129)	(424, 2048)
Sparsity	98.18 %	97.45 %

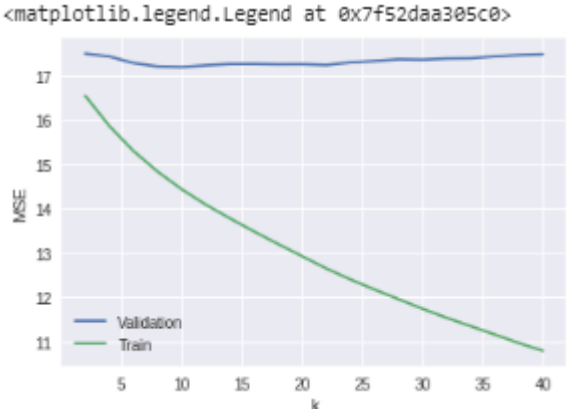
The model building and performance evaluation are done for both the matrices. But for the discussion purpose and due to space constraint, we would only discuss the outcomes for restaurants in Phoenix. The outcomes were similar for the restaurants in Scottsdale. The results for the restaurants in Scottsdale are available in the appendix.

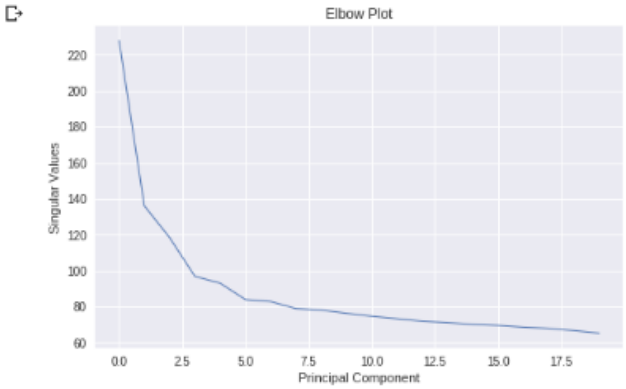
3 Model Building and Performance Evaluation

3.1 Implicit latent features learning based models (Baseline Models)

3.1.1 Singular Value Decomposition

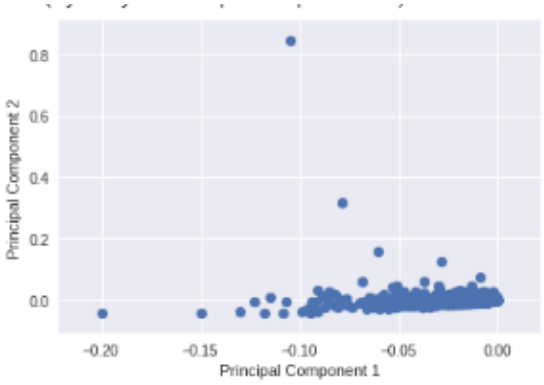
One of the popular methods of doing matrix completion is based on Singular Value Decomposition. The underlying motivation for doing this method is that *we expect our matrix to exhibit a low rank structure* (as there'll only be some handful of (often abstract) features which determine how a user rate a restaurant and how a restaurant is rated by a user - this can be thought of as taste of users or type of restaurants). From the spectrum (Elbow Plot) of the matrix, it's quite clear that most of the power is captured in the first few singular values and we expect to detect the latent features and get a better reconstruction by using the first k singular vectors. Monitoring the validation error, we find that we get the best possible results at k=2. Moreover, we obtain a huge difference between the validation and train MSE at k=2 i.e. diff (MSE) ~ 3.





Since the matrix is very spase we fill all the Nan values with zero before taking an SVD, the reconstruction is implicitly trying to construct back the matrix with a lot of zero entries instead of actually filling in those missing values. Also, due to the extreme low rank of the matrix, even at k=2, we are able to capture most of the zeros, thus, making SVD based matrix incompetent.

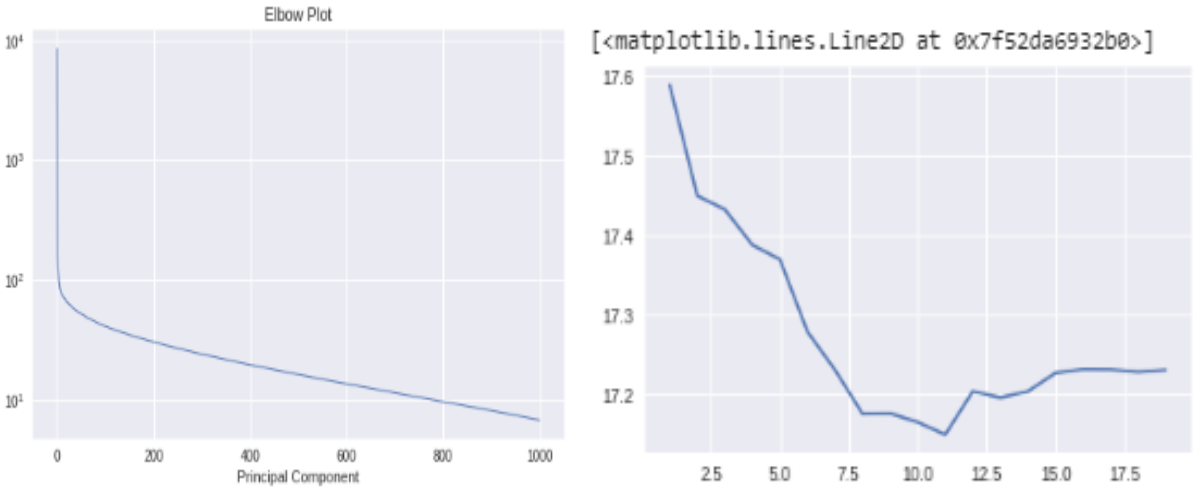
On visualizing the first two components of user and restaurant features, we again find that there isn't any (observable) structure in these components.



3.1.1.1 SVD with bias correction:

We obtain similar results on decomposing the matrix into $\boldsymbol{X} = \boldsymbol{u} \times \mathbf{1}^T + \mathbf{1} \times \boldsymbol{r} + \boldsymbol{X}'$, where u and r, are user and restaurant bias vectors and X' is the matrix which captures variational information, and performing a matrix completion based on SVD on X'. The best validation MSE we could obtained was SVD (with Bias Correction) for k to around 11.

Github: https://github.com/passkey04/Yelp-Recommender-System



3.1.2 Cosine similarity based prediction with bias correction

A natural way to think about the prediction problem is to assume that the rating r = f(u,v), where u and v are the feature vectors for users and restaurants. One would expect that the rating will be higher if a user's interests matches with the type of restaurant he/she visits, implying that the rating can be thought of as a weighted combination of the user-restaurant "similarity" measure. We formulate

$$\hat{r}_{ui} = \bar{r_u} + \frac{\sum_{u'} sim(u,u')(r_{u'i} - \bar{r_{u'}})}{\sum_{u'} |sim(u,u')|}$$

We used the restaurant based similarity so that we could tackle the cold-start problem for the new users.

The best out of sample error achieved by the model after hyperparameter tuning was MSE = 14.64 for k=39 which is comparable to that of SVD. We realize that cosine similarity is not a good measure to use at such high dimensions and combined with this, the high sparsity in our data doesn't give the model enough signal to detect patterns.

3.2 Explicit latent features learning based models

3.2.1 Learning Latent Features using Alternating Least Squares

Another popular method of matrix factorization is to assume that the matrix factorizes into X,Y and then

solve for them.

$$\hat{r}_{ui} = \mathbf{x}_u^\intercal \cdot \mathbf{y}_i = \sum_k x_{uk} y_{ki} \qquad L = \sum_{u,i \in S} (r_{ui} - \mathbf{x}_u^\intercal \cdot \mathbf{y}_i)^2 + \lambda_x \sum_u \|\mathbf{x}_u\|^2 + \lambda_y \sum_u \|\mathbf{y}_i\|^2$$

$$\frac{\partial L}{\partial \mathbf{x}_u} = -2 \sum_i (r_{ui} - \mathbf{x}_u^\intercal \cdot \mathbf{y}_i) \mathbf{y}_i^\intercal + 2\lambda_x \mathbf{x}_u^\intercal$$

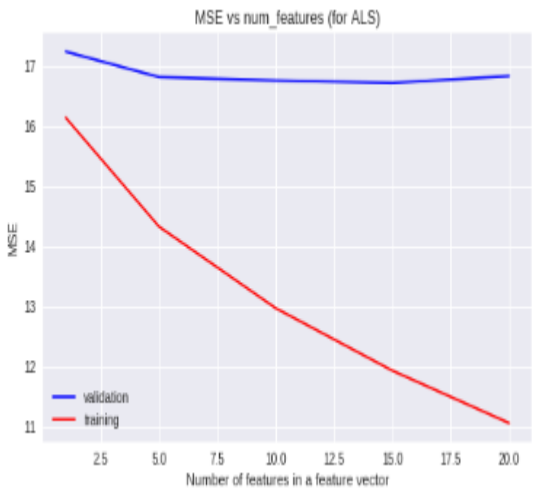
$$0 = -(\mathbf{r}_u - \mathbf{x}_u^\intercal Y^\intercal) Y + \lambda_x \mathbf{x}_u^\intercal$$

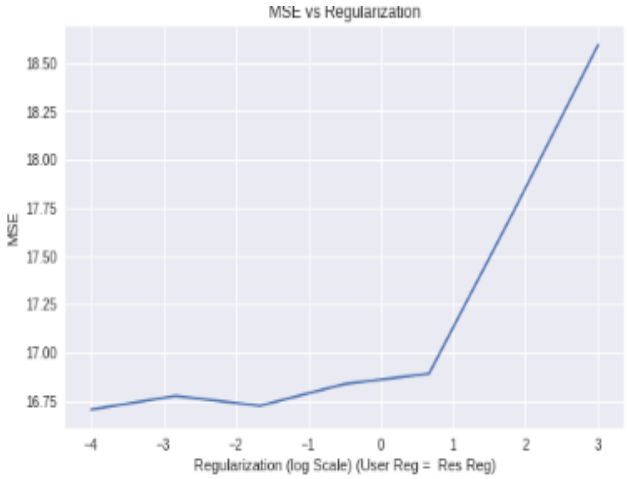
$$\mathbf{x}_u^\intercal (Y^\intercal Y + \lambda_x I) = \mathbf{r}_u Y$$

$$\mathbf{x}_u^\intercal = \mathbf{r}_u Y (Y^\intercal Y + \lambda_x I)^{-1}$$

We get the best out of sample error for feature number = 10 and user_reg = res_reg = 0.001.
Best MSE (for ALS) = `17.230543322467152`

ALS produces similar performances to that of other baseline models which wasn't surprising given the nature of the data





3.2.2. Learning Latent Features through Stocastic Gradient Descent

From the performance of baseline models and ALS, we realized that the high sparsity of our matrix is forcing the baseline models to predict the missing ratings to be zero in even with small number of latent variables for user/restaurant features. This also implies that the underlying matrix is very low rank. We formulate a similar decomposition to that of ALS(and cosine similarity) but with some bias terms all of which learned through gradient descent by minimizing a regularized cost function which only looks at error between the actual non-zero rating and predicted rating, instead of the entire matrix. We thought this was a reasonable model to try as the number of parameters we've to learn will be small due to the low rank

Github: https://github.com/passkey04/Yelp-Recommender-System

structure of matrix.

$$\hat{r}_{ui} = \mu + b_u + b_i + \mathbf{x}_u^\intercal \cdot \mathbf{y}_i$$

$$L = \sum_{u,i} (r_{ui} - (\mu + b_u + b_i + \mathbf{x}_u^\intercal \cdot \mathbf{y}_i))^2 + \lambda_{xb} \sum_u \|b_u\|^2 + \lambda_{yb} \sum_i \|b_i\|^2 + \lambda_{xf} \sum_u \|\mathbf{x}_u\|^2 + \lambda_{yf} \sum_u \|\mathbf{y}_i\|^2$$

$$\frac{\partial L}{\partial b_u} = 2(r_{ui} - (\mu + b_u + b_i + \mathbf{x}_u^\intercal \cdot \mathbf{y}_i))(-1) + 2\lambda_{xb} b_u$$

$$\frac{\partial L}{\partial b_u} = 2(e_{ui})(-1) + 2\lambda_{xb} b_u$$

$$\frac{\partial L}{\partial b_u} = -e_{ui} + \lambda_{xb} b_u$$

$$b_u \leftarrow b_u + \eta (e_{ui} - \lambda_{xb} b_u)$$

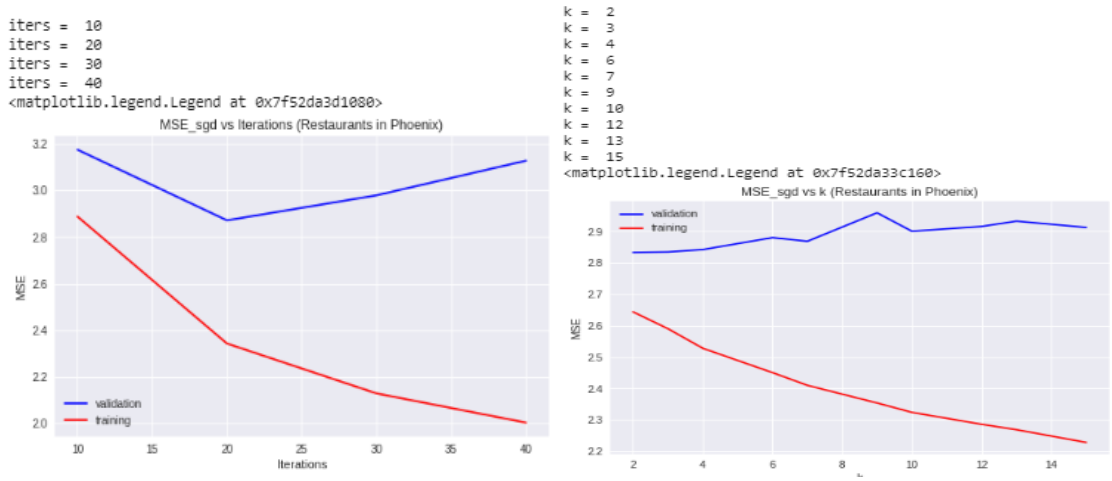
$$b_i \leftarrow b_i + \eta (e_{ui} - \lambda_{yb} b_i)$$

$$\mathbf{x}_u \leftarrow \mathbf{x}_u + \eta (e_{ui} \mathbf{y}_i - \lambda_{xf} \mathbf{x}_u)$$

$$\mathbf{y}_i \leftarrow \mathbf{y}_i + \eta (e_{ui} \mathbf{x}_u - \lambda_{yf} \mathbf{y}_i)$$

On tuning for the optimal k, we found that we get the best out of sample error at k=2.Over fitting for higher values of k is predictable given the sparsity of matrix. We can also observe that in the spectrum of singular values of the full matrix, we observe inflection points at k=2 and k=7 both of which minimizes the validation error for our method based on SGD as well. In addition to incorporating the bias terms that we learn, we believe that looking at the error of only the actual non-zero values is one of the contributing factors for the good performance we get using this model.

Tuning the hyperparameter like number of iterations and number of features in the latent feature vector, we got minimum MSE of `2.8545074083478035` for 20 iterations and number of features = 2



Github: https://github.com/passkey04/Yelp-Recommender-System

3.3.1 Random Forest Regressor

The feature vectors learned from SGD were used as the features for the regression problem. Fitting a Random Forest Regressor with max_depth = 5, we obtained a comparable MSE of 2.710045376922961 for the restaurants in Phoenix.

3.4 Ensemble of all the predictors

We formulate our final prediction as a linear combination of the predictions of our Gradient based model and random forest. The MSE obtained for the ensemble model was lower than all the individual models. The best validation MSE obtained was 2.6333817860275097 for restaurants in Phoenix.

.

3.5 Summary of all the Models

Model	Validation MSE	
	Restaurants in Phoenix	Restaurants in Scottsdale
SVD	17.19524313333571	16.68550995321838
Cosine Similarity based model	16.68550995321838	17.025866680642704
ALS based model	17.230543322467152	16.729128335518105
SGD based model	17.230543322467152	2.7363082502586065
Random Forest Regressor	2.710045376922961	2.6717940609972395
Ensemble of SGD and RF	**2.6333817860275097**	**2.5183558867860882**

Based on the analysis of all the models, we choose the Ensemble Model for both the Restaurants in Phoenix and for the Restaurants in Scottsdale as the best model for recommendation.

On testing the models on the Test Data, we got: MSE of 2.663 (for Restaurants in Phoenix) and 2.518 (for Restaurants in Scottsdale).

4 Deployment and Monitoring

4.1 Scaling Up Issues

In the real world scenario rarely recommendation engines are implemented using Python. Some big data frameworks like Spark is used that helps in parallelizing the computation. As the number of users on the platform increases, it becomes increasing difficult and slow to train the models by using the traditional methods. Spark has an in-built map reduce framework that helps in parallelizing the computation and efficiently deploys the recommendations in the production environment.

Generally, ALS and SGD based models scale up better since its computation can be parallelize, whereas SVD and Cosine Similarity based models' complexity increase by n^2 hence training these models become increasingly inefficient as the number of users and restaurants increases.

4.2 Testing and monitoring the models in real time (Future work)

Since the model develops recommendation for users, it is very difficult to test if the predictions were correct or not. Many a times some proxy is used. For example, if the recommendation engine provides recommendation of items to purchase, it is pretty straight forward that if the user purchased the item, clicked on the item, etc. it can be termed as a successful recommendation. In our case, if the user reviews the restaurants on Yelp after the model recommends it to the user, it can termed as a success. Another way to monitor the success is by aliasing with the restaurants and getting to know if a customer visited the restaurant after the recommendation.

To test between two different models, A/B testing is generally used to determine the best model. Some population is shown recommendation based on 'A' model and other population is shown recommendation based on 'B' model. The statistical difference between the model's successes are analyzed. If the difference is significant, we can term one model better over the other.

Many a times, a new user comes on board, so the model don't know much about the user. So, it cannot recommend the restaurants to the user. This is called a Cold start problem. To overcome this problem, the meta-data of user like demographic information, location, profession, etc. can be used to find similar users and based on the similarity the model can make recommendations, or the model can recommend the most popular one. In our case, since we don't have the meta-information about the users that helps in characterizing the user, we might go ahead with recommending the most popular restaurant.

4.3 Ethical Issue concerning with the Recommendation Engines

Generally, the recommendation engine also take in account the demographic information of the user. In that case, there are chances that model might become biased towards some caste, gender, profession etc. These biases were learnt from the past data. Hence, the model will only aggravate this discrimination against some set of people. Fortunately, in our case, the model is not using any user specific details. Therefore, the chances of such racial biases creeping into the model is low. Moreover, since it is just a restaurant recommendation, the recommendations made by the model might not have much social implications.

5 Code and Data

All the code and data for the project is available on the github repository with the following link:

https://github.com/passkey04/Yelp-Recommender-System

6 References

- http://blog.ethanrosenthal.com/2016/01/09/explicit-matrix-factorization-sgd-als/
- http://blog.ethanrosenthal.com/2015/11/02/intro-to-collaborative-filtering/