CF Project
Restaurant Recommendation

Abhinav Saurabh (MT20127 MTech CSE)
Ramachandran T.N (2017084 BTech CSE)

Introduction

- Used Yelp Dataset
- 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.

- 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 jth 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 feature learning based models (Supervised).
- Latent features are those features that characterizes a user or a restaurant.

Data Loading

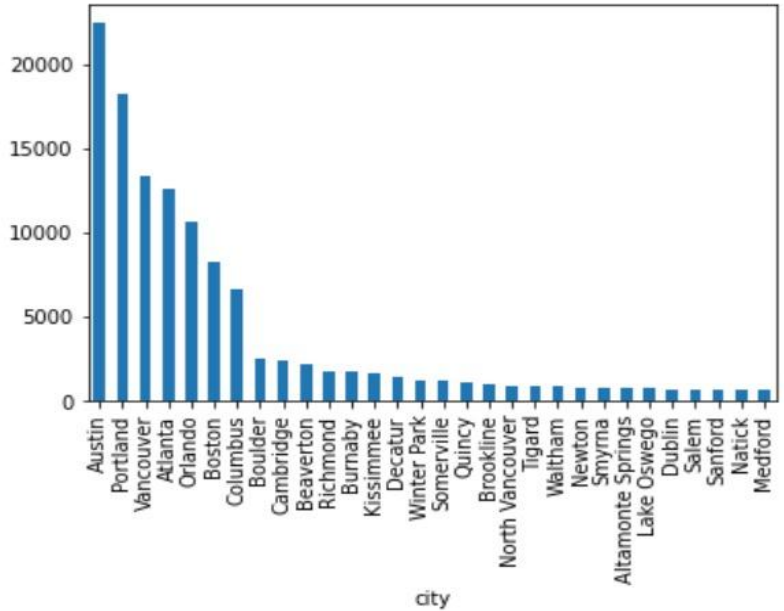
Loading the review data file and business file

	business_id	name	city	state	stars	review_count	categories	latitude	longitude	is_open	postal_code
0	6iYb2HFDywm3zjuRg0shjw	Oskar Blues Taproom	Boulder	CO	4.0	86	Gastropubs, Food, Beer Gardens, Restaurants, B...	40.017544	-105.283348	1	80302
1	tCbdrRPZA0oiIYSmHG3J0w	Flying Elephants at PDX	Portland	OR	4.0	126	Salad, Soup, Sandwiches, Delis, Restaurants, C...	45.588906	-122.593331	1	97218
2	bvN78flM8NLprQ1a1y5dRg	The Reclaimory	Portland	OR	4.5	13	Antiques, Fashion, Used, Vintage & Consignment...	45.511907	-122.613693	1	97214
3	oaepsyvc0J17qwi8cfrOWg	Great Clips	Orange City	FL	3.0	8	Beauty & Spas, Hair Salons	28.914482	-81.295979	1	32763
4	PE9uqAjdw0E4-8mjGl3wVA	Crossfit Terminus	Atlanta	GA	4.0	14	Gyms, Active Life, Interval Training Gyms, Fit...	33.747027	-84.353424	1	30316

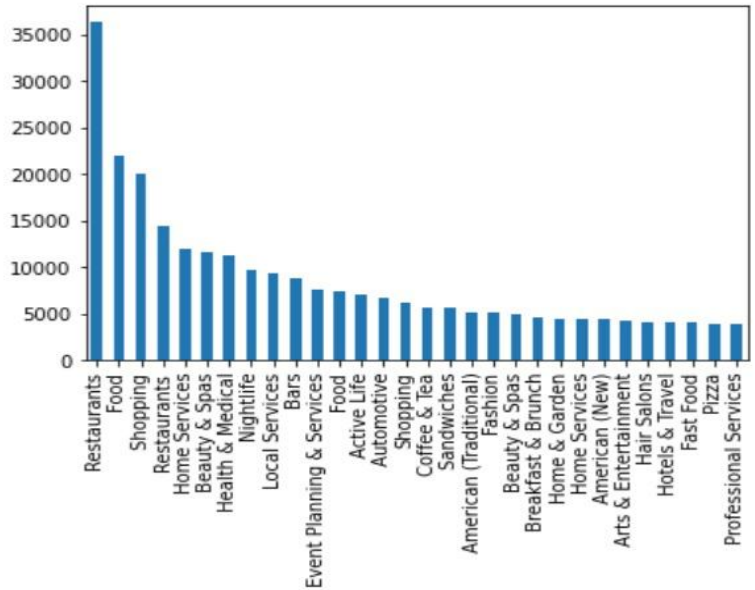
The ‘business’ file has the details about each business. Yelp has a lot of businesses of different categories like restaurants, medical care services, auto care services, etc. listed on its website. The reviews present in the ‘review’ file contains reviews for all the categories. But, the recommendation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

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

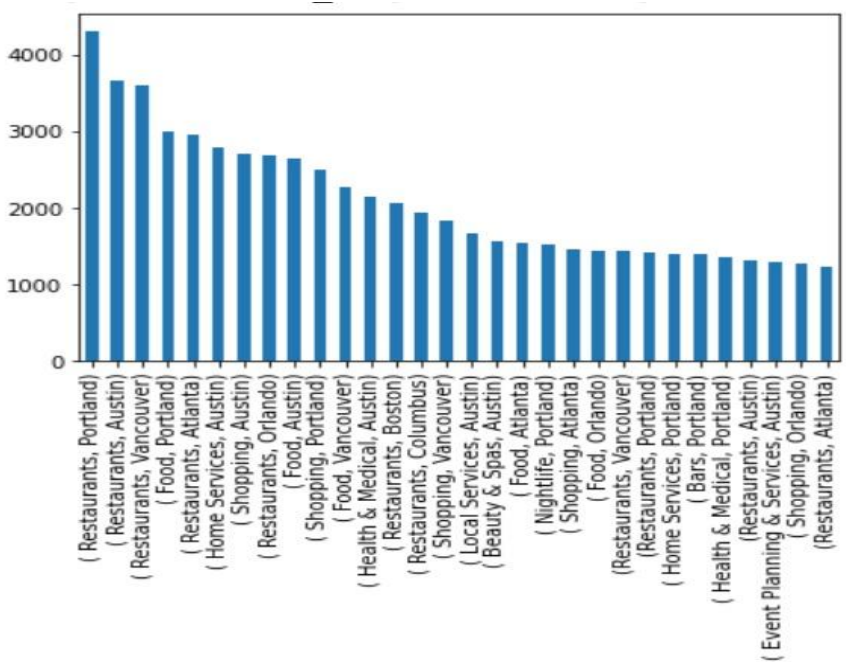
No of businesses in each city



Different type of bussiness and their count



Business and City Ordered pairs and their count



It can be seen that most of the businesses listed on the Yelp are Restaurants, Shopping places and Recreational Centers. For this given data set (obtained from Yelp's Website), top cities with maximum businesses are Portland, Austin, and Vancuover.

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
- 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.
- The top Business and City for which highest number of reviews are obtained

600000
500000
400000
300000
200000
100000
0
(Restaurants, Austin)
(Restaurants, Portland)
(Restaurants, Atlanta)
(Restaurants, Boston)
(Restaurants, Orlando)
(Food, Austin)
(Food, Portland)
(Restaurants, Vancouver)
(Nightlife, Portland)
(Nightlife, Austin)
(Bars, Portland)
(Bars, Austin)
(Bars, Boston)
(Nightlife, Boston)
(Nightlife, Atlanta)
(Bars, Atlanta)
(Restaurants, Austin)
(Food, Atlanta)
(Food, Boston)
(Restaurants, Columbus)
(Restaurants, Portland)
(Breakfast & Brunch, Portland)
(Restaurants, Atlanta)
(Breakfast & Brunch, Austin)
(Nightlife, Orlando)
(Restaurants, Boston)
(American (New), Austin)
(Food, Vancouver)
(Bars, Orlando)
(Food, Orlando)

Sparsity of the data

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.

	Restaurants in Austin	Restaurants in Portland
SHAPE	(9666, 3621)	(8968, 4273)
SPARSITY	99.33968534126829%	99.46274378485492%
#non zero entry (training)	192454	170008
#non zero entry (validation)	19328	17934
#non zero entry (testing)	19332	17936

MODELS

- SVD (Singular Value Decomposition)
- Cosine Similarity based model
- ALS based model (Alternating Least Squares)
- SGD based model (Stochastic Gradient Descent)

Singular Value Decomposition(SVD)

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.

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.

Cosine Similarity Based Prediction

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.

Alternating Least Squares (ALS) Method

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

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 xi, a randomly initialized k-dimensional vector representing the restaurant yj, the rating given by the user to the restaurant rij. The target variable is rij. 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.

Stochastic Gradient Descent

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.

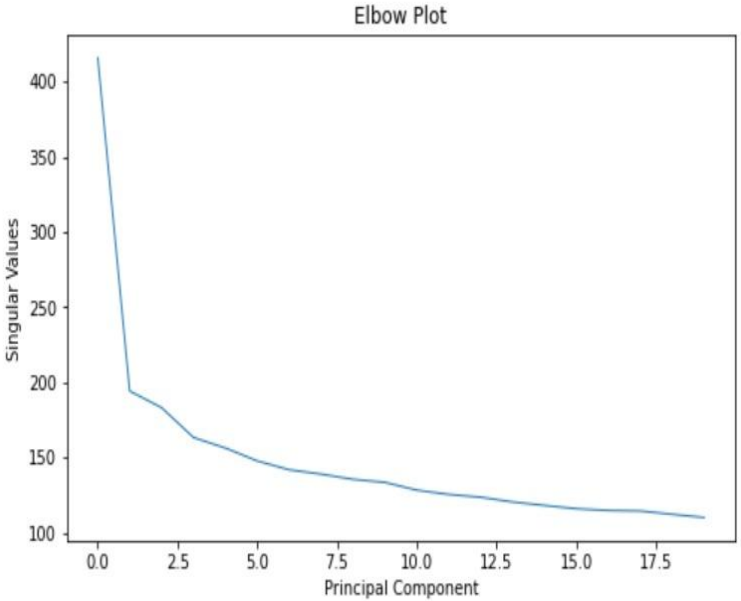
So to ensure that the latent feature for each user and restaurant is free from such biases, it is learned as a separate parameter. Essentially, the final rating given by a user i to restaurant j is broken into four components: a global bias (average of all the available ratings), a user bias, a restaurant 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.

Model Building and Performance Evaluation

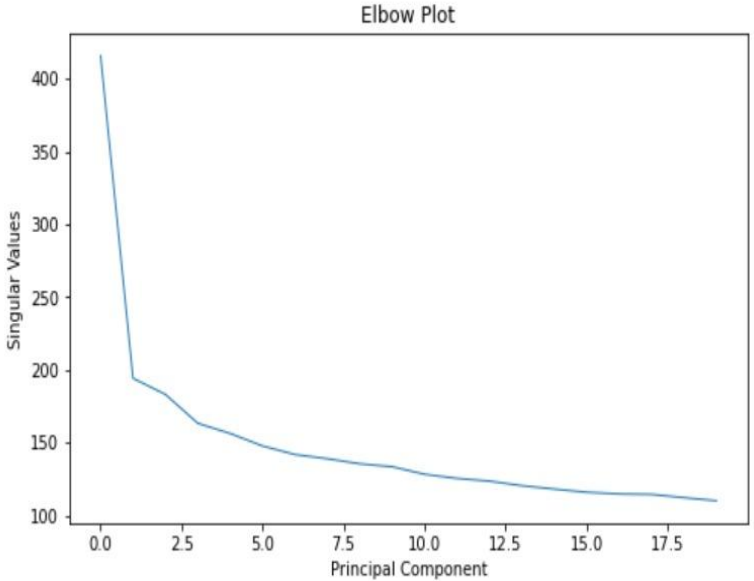
Singular Value Decomposition

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.

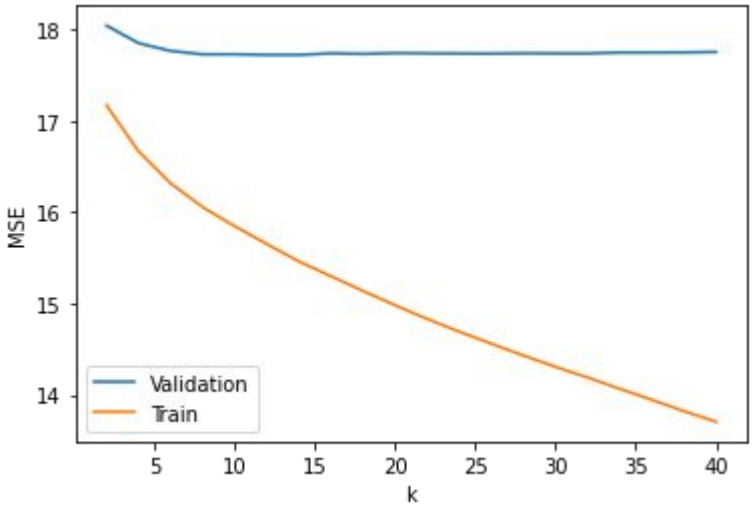
For Restaurants in Austin

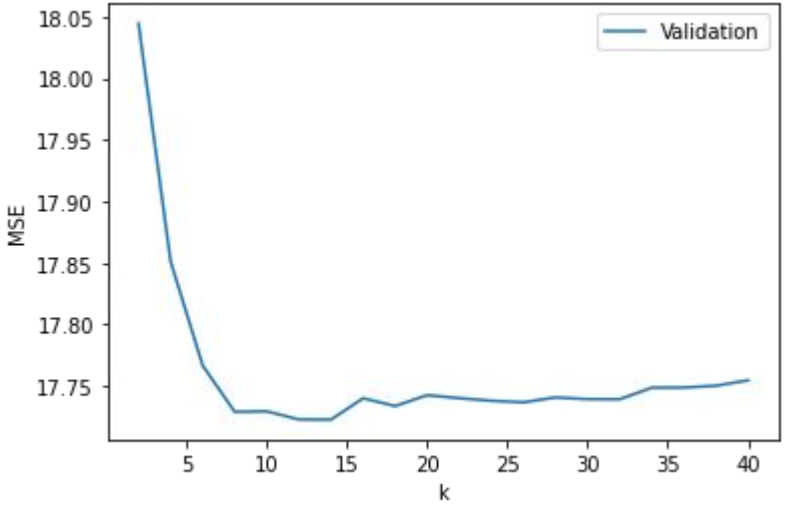


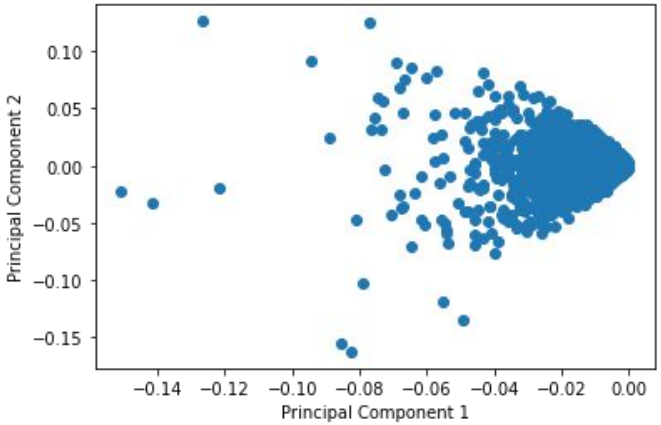
For Restaurant in Portland



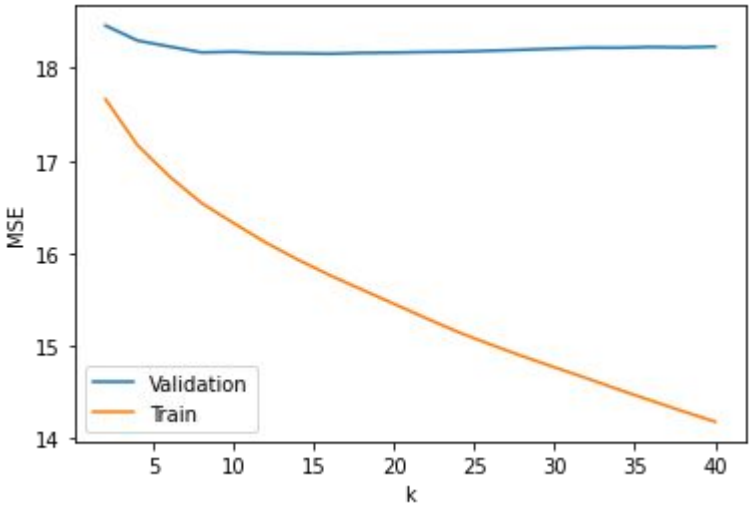
For Restaurants in Austin

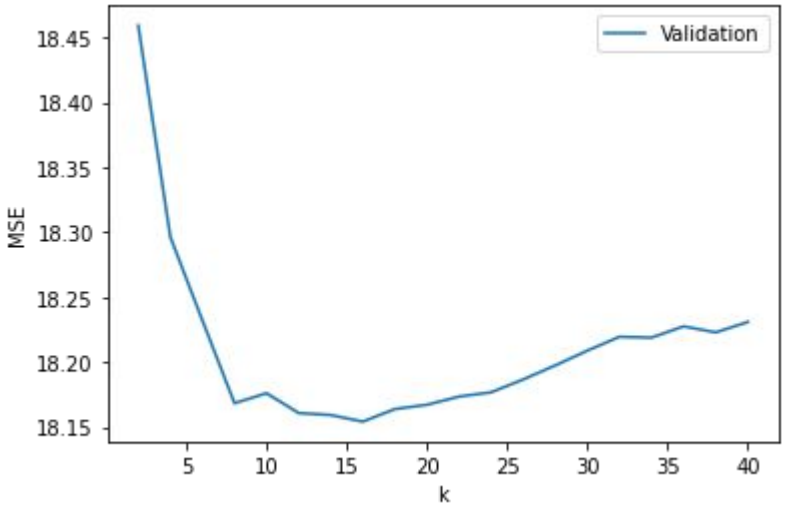




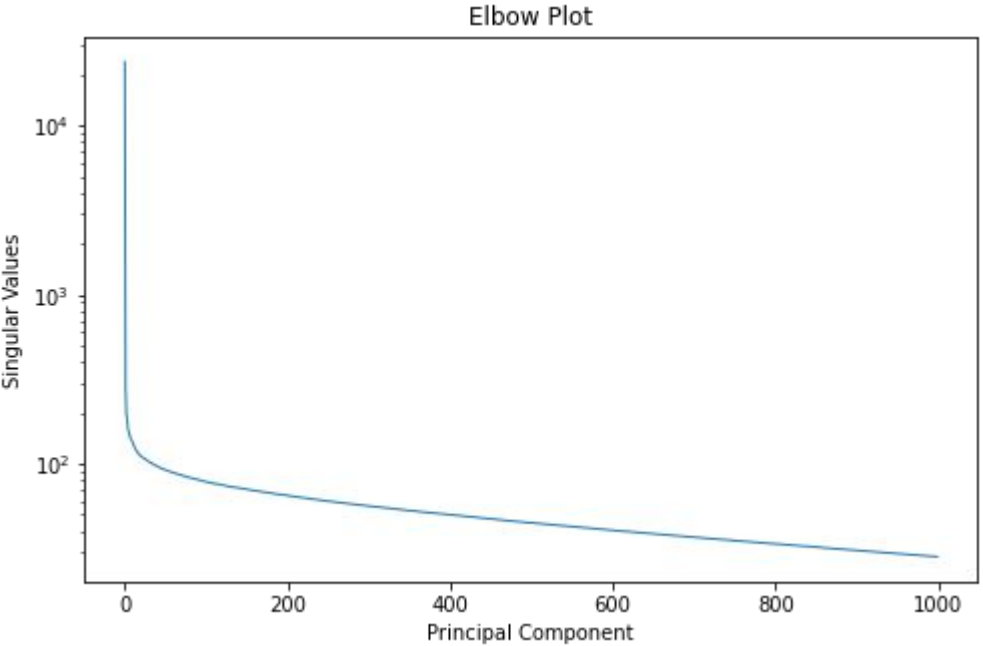


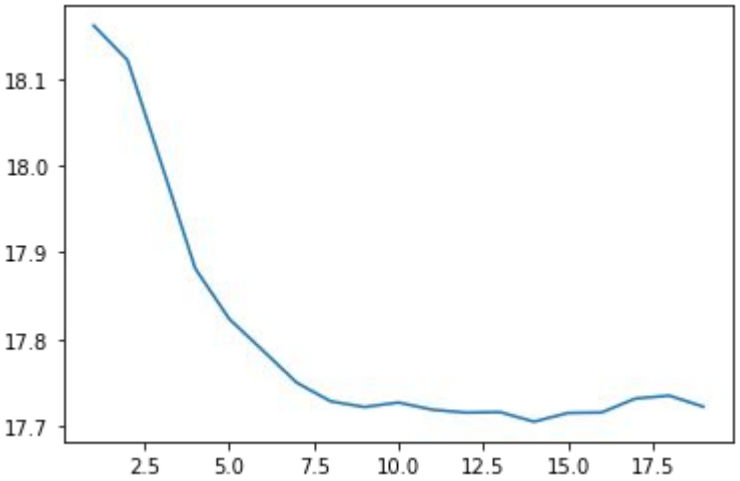
For Restaurants in Portland



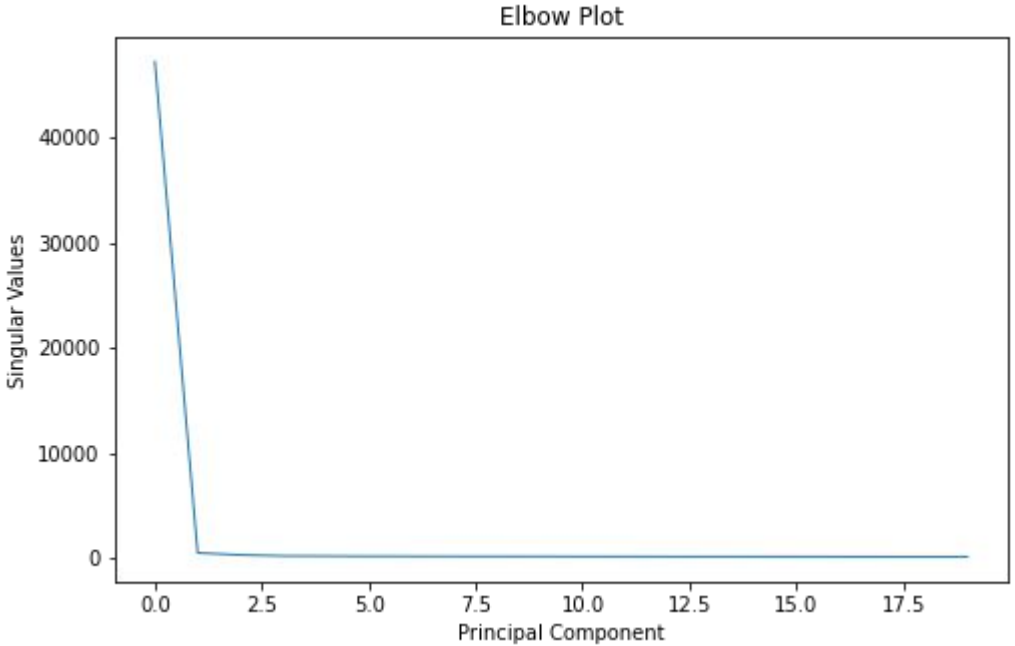


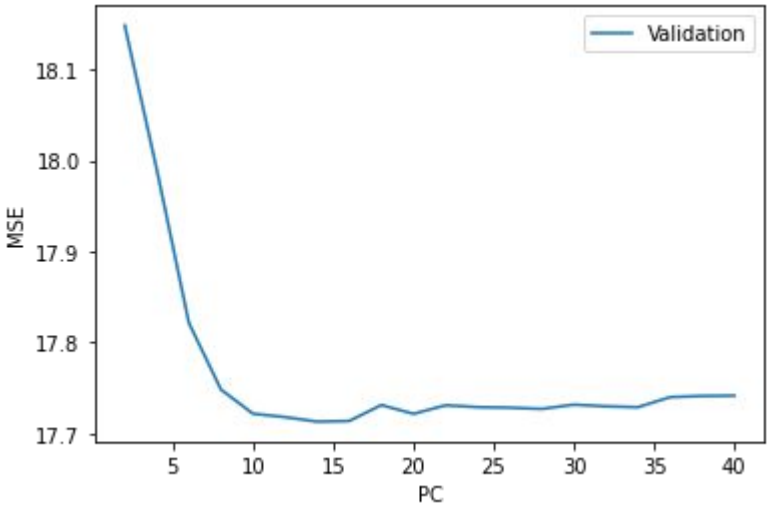
SVD with Bias Correction



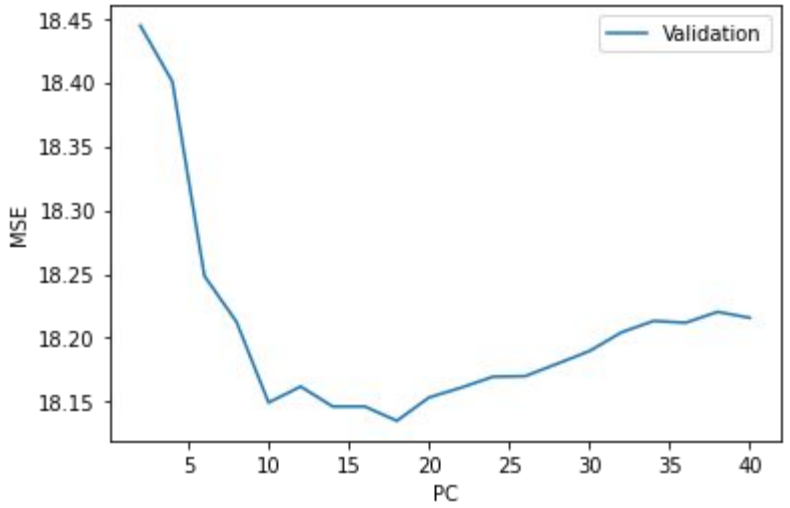


For Restaurants in Austin





For Restaurants in Portland



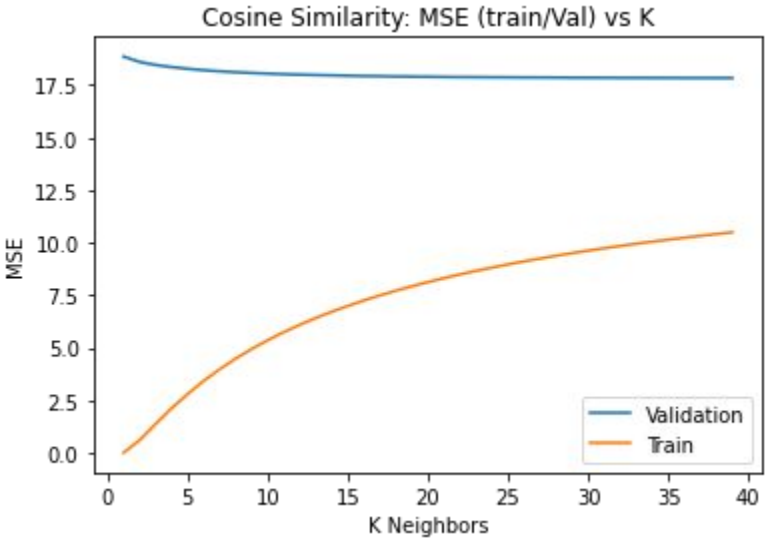
Cosine Similarity with correction for bias

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

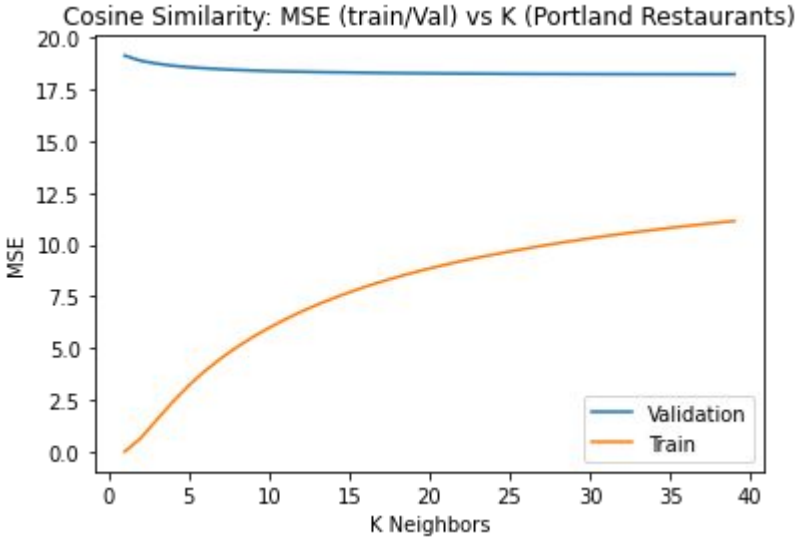
$$\hat{r}_{ui} = \bar{r_u} + \frac{\sum_{u'} sim(u,u')(r_{u'i} - \bar{r_{u'}})}{\sum_{u'} |sim(u,u')|}$$

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

For Restaurants in Austin



For Restaurants in Portland



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.

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

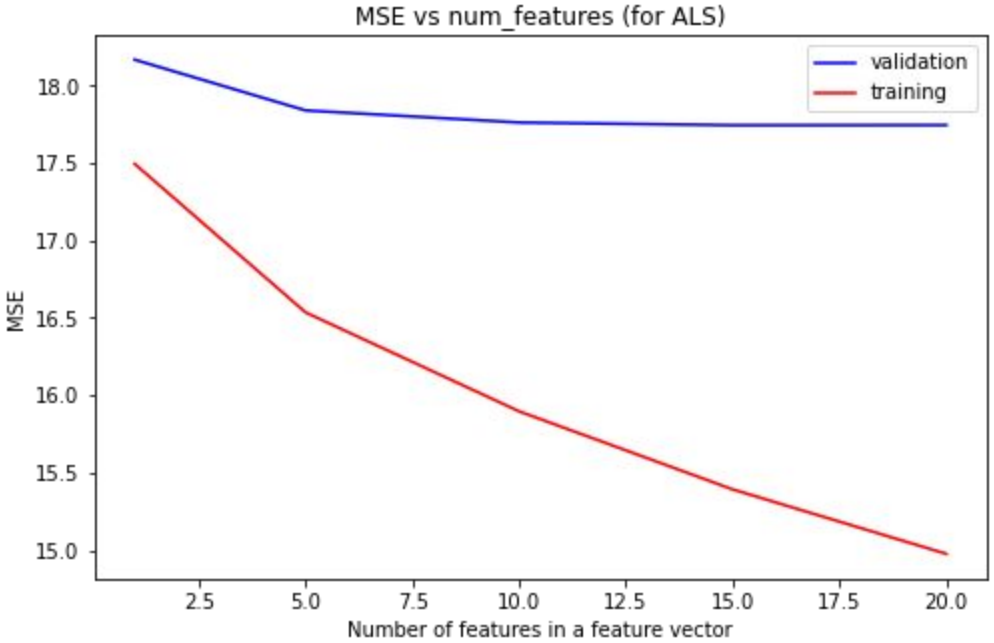
$$\frac{\partial L}{\partial \mathbf{x}_u} = -2 \sum_i (r_{ui} - \mathbf{x}_u^\intercal \cdot \mathbf{y}_i) \mathbf{y}_i^\intercal + 2\lambda_x \mathbf{x}_u^\intercal$$

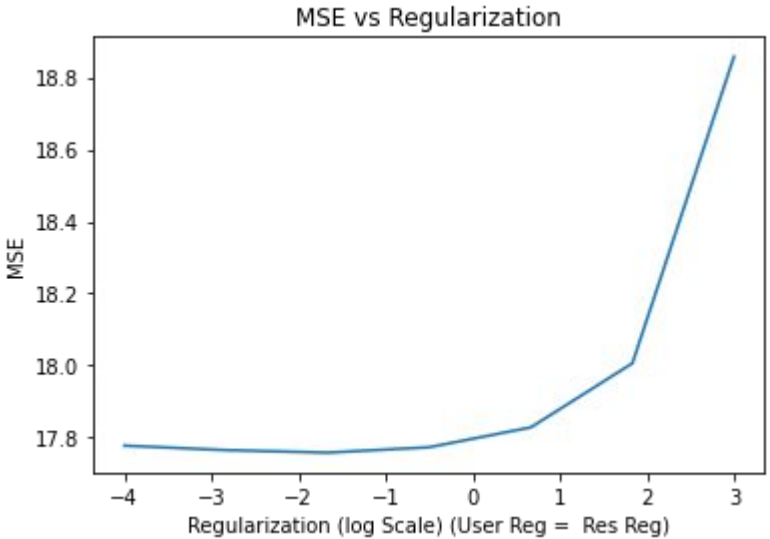
$$0 = -(\mathbf{r}_u - \mathbf{x}_u^\intercal Y^\intercal) Y + \lambda_x \mathbf{x}_u^\intercal$$

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

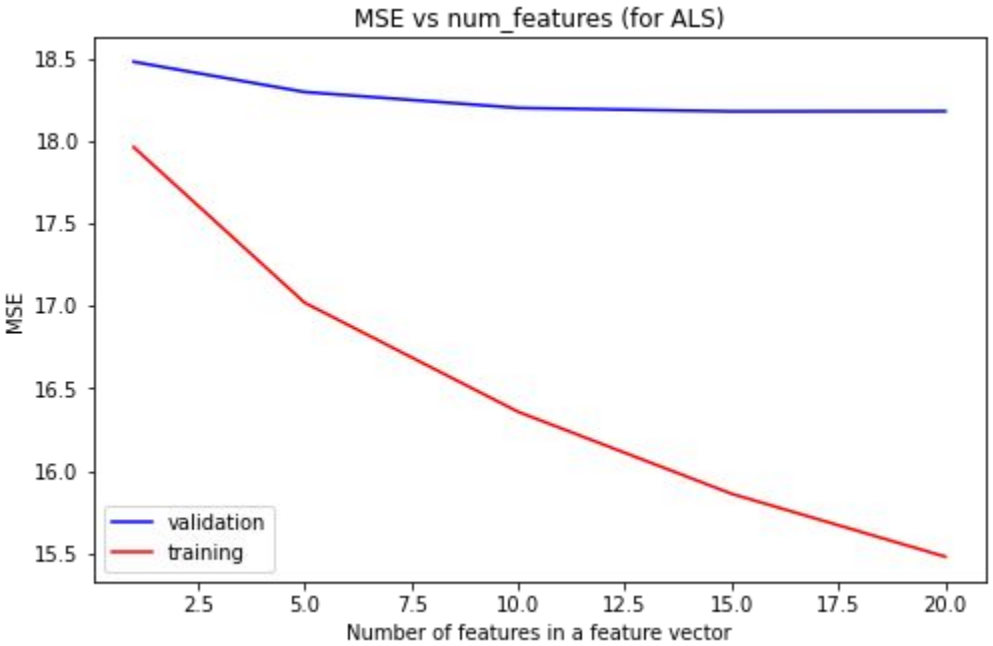
$$\mathbf{x}_u^\intercal = \mathbf{r}_u Y (Y^\intercal Y + \lambda_x I)^{-1}$$

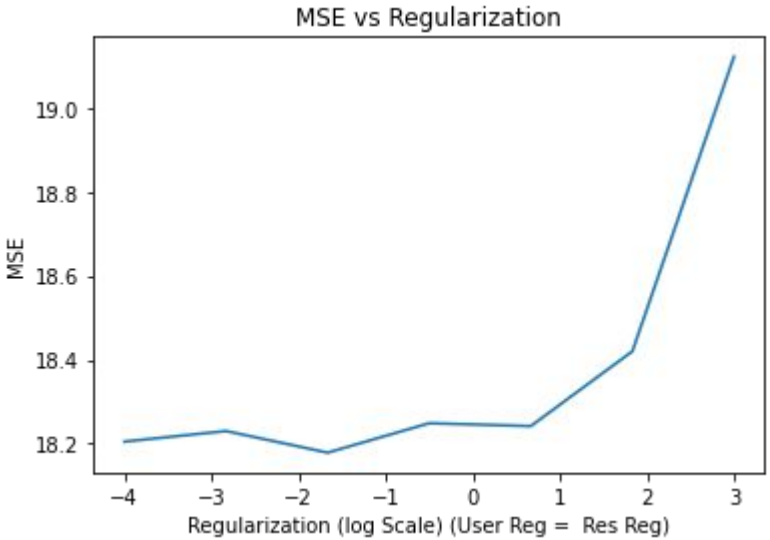
For Restaurants in Austin





For Restaurants in Portland





Learning Latent Features through Stochastic 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 structure of matrix.

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

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

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

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

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

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

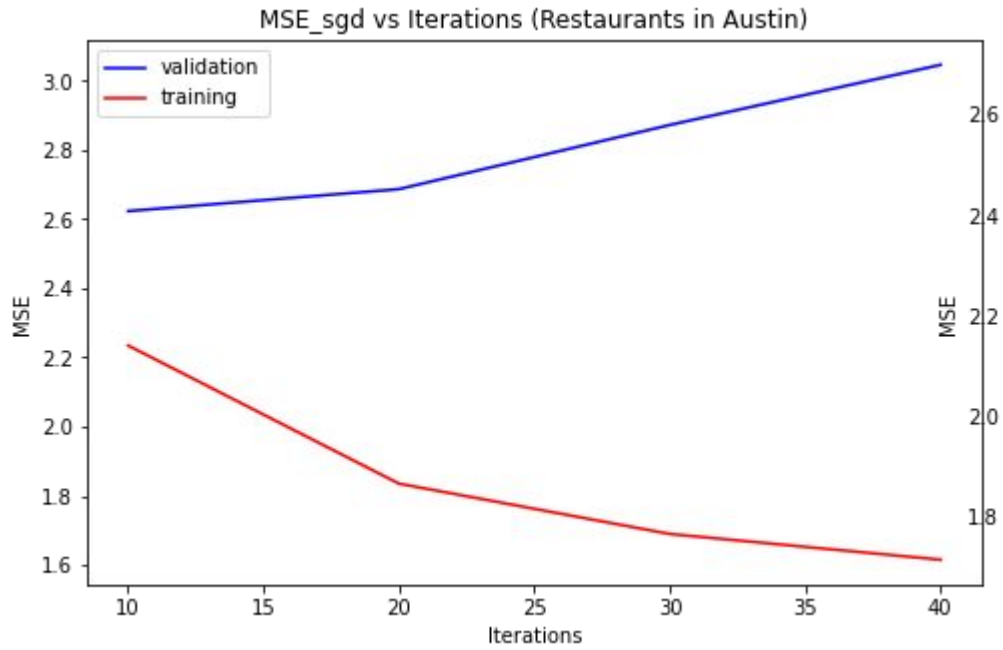
$$b_i \leftarrow b_i + \eta \left(e_{ui} - \lambda_{yb} b_i\right)$$

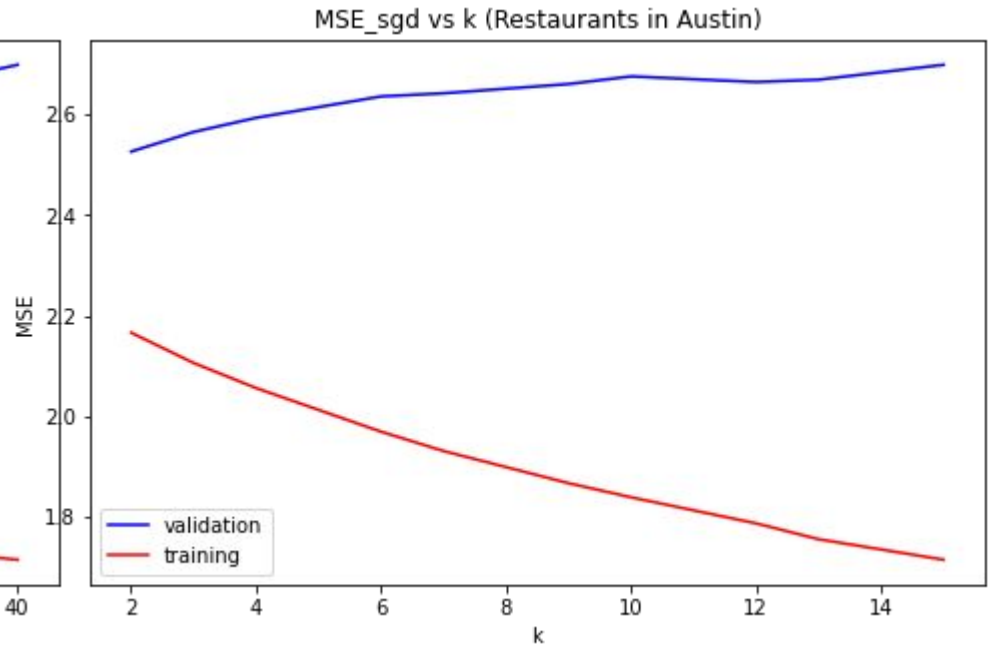
$$\mathbf{x}_u \leftarrow \mathbf{x}_u + \eta \left(e_{ui} \mathbf{y}_i - \lambda_{xf} \mathbf{x}_u\right)$$

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

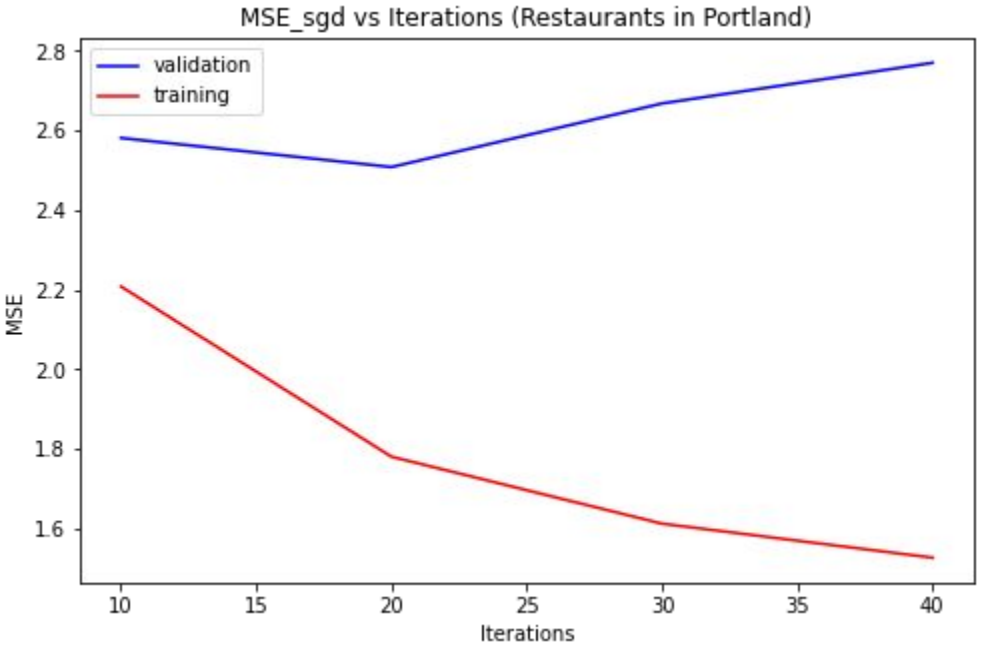
Tuning the Iters hyper-parameter

For Restaurants in Austin





For Restaurants in Portland



Results

Model	VALIDATION MSE	
	Restaurants in Austin	Restaurants in Portland
SVD	17.72956170325437	18.154128047816524
Cosine Similarity	17.837366517765673	18.216949652884058
ALS	17.720206854789584	18.205941902691375
SGD	2.5194437078503342	2.476078061616359

Future Work

- We are also implement Neural Network based model, Random Forest Regressor.

- We will also implement Ensemble of SGD, NN and RF.

- Content Based Filtering is also being implemented.

Thank You