Yelp Data Recommendation System

Prasham Sheth (pds2136), Shreyas Jadhav(sj3006), Manas Dresswala (mad2306) and Anjani Prasad Atluri (aa4462)

INDEX

1. Introduction

In today's world, where data has become the new oil, recommendation systems has taken a huge leap in terms of research and real time implementation.

With the growing technology industry, the amount of competition between these companies to retain users has grown tremendously. Everyone is behind personalizing their experiences based on the user so that the user actually comes back to their side. In today's day and age, the number of applications and websites has increased tremendously. Especially in the area of restaurants. In the US, Yelp has become very famous for finding any kind of restaurant for breakfast, lunch or dinner. In does not stop there, we can even find pubs and cafes depending on our taste and interest.

In this task, we try and build a recommendation system for our users on Yelp using the Yelp Dataset from the Yelp Challenge 2019.

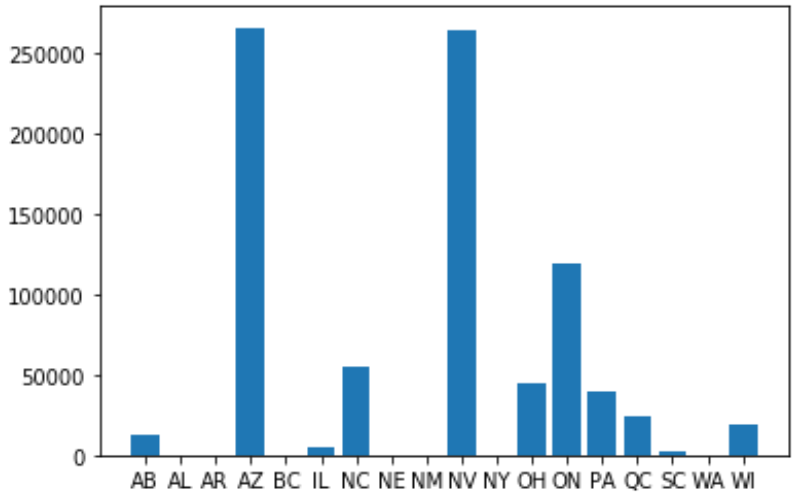
2. Data Analysis

After doing some exploratory data analysis on our data, we found the following -

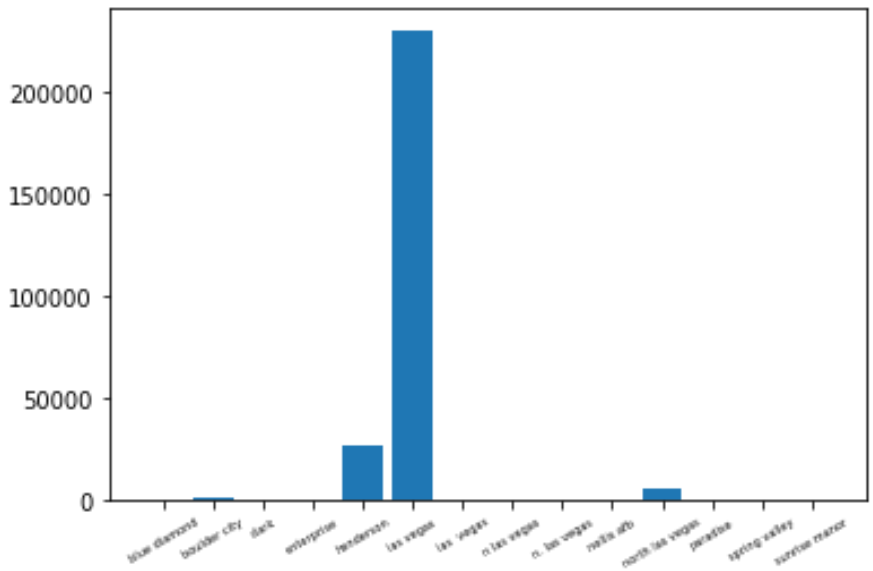
1. There are a lot of users that have rated less than 5 businesses. So we remove them and only keep the active users in the dataset.

2. Another interesting fact we found was that there are over businesses with over 1200+ categories. It was not possible to build a common recommendation system for all of them as users might interact with different businesses differently. So we decided to take only those businesses with category = Restaurant and/or Food.

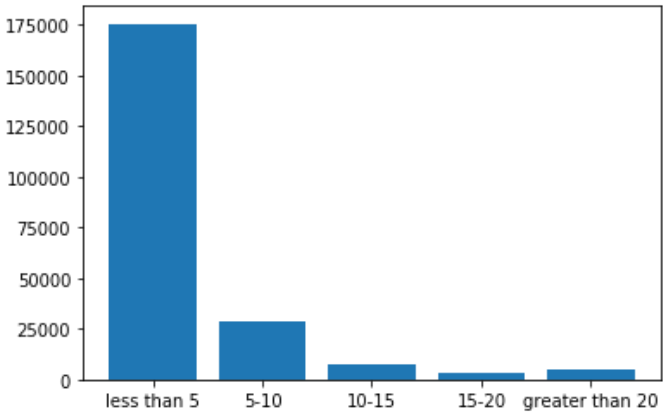
3. Next, we see that the businesses are from all over the country. When we plotted a graph of the number of ratings from each state, we see that most of the ratings are from Arizona and Nevada.



4. So we take the businesses only from Nevada. Next. we see from which city are most of the ratings coming. The below graph tells us that most of the ratings are coming from - Las Vegas.



5. Next we divide the users into 5 different bins based on the number of ratings from each user - users with less than 5 ratings, between 5 and 10 ratings, between 10 and 15 ratings, between 15 and 20 ratings and greater than 20 ratings. The below graph tells us that most of the users here also have rated less than 5 businesses.



6. Thus, we remove the users that have rated less than 5 users. We need to do this again as we are trying to predict the last rating of a user, so we need to have data of a user in training and testing as well.

3. Business Rule

Let's take a minute here to understand how we are going ahead with our recommendation task and what business rules are we applying here.

The main task here is to predict the last rating that the user gave to a particular business. To go about predicting the rating, we are using the previous user data that has been given to us. We are not only using the ratings that the user gave to multiple businesses but also using the other features about users and businesses given to us.

To try out something more interesting we also used the reviews as well as images given to businesses.

Before we jump to understand how we used the last predicted rating to recommend we will define our first business rule.

If a user has rated a business with rating greater than or equal to 2.5, then we say that the user liked the particular business, or else we take it that he disliked the business.

Now, if the user has rated a business with a rating greater than or equal to 2.5, and we predict the rating to be greater than 2.5 then we say that this a true positive. Similarly we are defining the accuracy metrics like precision and recall. More about these in the evaluation metrics section.

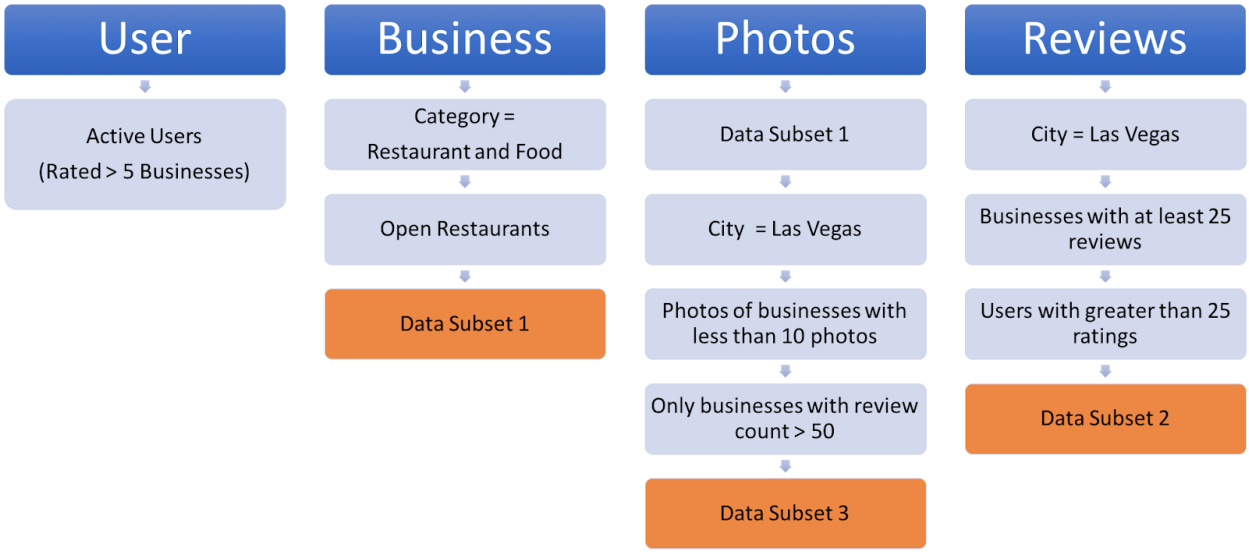
In the recommendation task we are recommending businesses that are similar to the last business the user has rated of the user liked the last business he rated, otherwise we are giving him top 5 recommendations through ALS based model.

4. Data Sampling

Before we jump into the implementation of these systems, we would like to discuss how we have sampled our data.

We have mentioned most of our sampling in the data analysis section, but here we will explain the different samples used in our models.

We had to take three different subsets in our case as each of the model had their own requirements. Due to time and memory constraints we had to use a smaller subset while running our model which was business reviews and an even smaller dataset for the model which uses business photos.



As we can see in the above flowchart we have created three subsets for our project. We did this so that we could explore the different models in depth.
The first subset, that is Data Subset 1, has the most number of users and we have only run Factorization Machine model on it.
Next, we created a model to capture the essence of reviews given to businesses by the users. That was Data Subset 2.
Lastly, we wanted to try something different and see what happens, so we used data from images to find similar businesses based on the images. For this we needed to have some photos for each business, because of which we created Data Subset 3.

5. Recommendation Systems

Now let's talk about how we are predicting the user's last rating. As mentioned above we are using the following information given to us to go about the prediction task -

1. Baseline -
    a. **ALS** using **Spark Machine Learning library**
    b. **BaselineOnly** using **surprise**
2. Similarity in Businesses using features like Categories - **Factorization Machine**
3. Reviews given by users to different businesses - **Review based Recommendation using NLP**
4. Images of the businesses - **Images based Recommendation using Image Embeddings**

This is an overall explanation of what will be doing for the prediction. Next, we will go ahead and explain each of the models where we will explain why used these models and how we implemented them.

6. Evaluation Metrics

As we promised above, before understanding anything else, lets see how we have evaluated the accuracy of our systems.

To know if we have built a good recommendation system, we first created a baseline and noted its accuracy.

You must be wondering, how we have defined accuracy here.
We have used to evaluation metrics to define the accuracy -

1. RMSE (Root mean squared error)
2. Precision and Recall

Precision can be defined as the number of true positives divided by the number of true positives plus the number of false positives. We have defined true positives as follows, if the user has actually given the last business a rating of 0.25 or above and if the model predicts the rating to be 2.5 or above, then this is considered as true positive (as the predicted rating of the model and the actual rating are in agreement that the user would have given this business an above average score). The true negatives are the cases in which the user has actually given a rating of 2.5 or lower to a business and if our models predict the rating to be 2.5 or lower. The case would be considered a False Negative if the actual rating of the user was above 2.5 but our model predicts a value of 2.5 or lower, and similarly a case would be False Positive if our model predicts a rating of 2.5 or higher when the user has actually given the business a 2.5 or lower rating. Recall is defined as the number of true positives plus the number of true positives plus false negatives.

7. Coverage

We are defining a metric called **catalogue coverage** for the recommendations we are providing. The catalogue coverage is essentially what ratio of our overall unique businesses in the dataset are getting given as recommendations to the users.

8. Implementation

8.1. Baseline

To understand if our model is actually working better and the work we did would make a difference in the business, we compare our models to two baselines.

We compare our models using two baselines, they are mentioned below -

8.1.1. Biased Baseline

The first baseline is a biased baseline that was calculated using the user and item bias in our data. We used the **surprise** package to calculate this baseline.

In the surprise package we used the **BaselineOnly** API that take into account the user bias and the item bias of our data.

8.1.2. ALS

We also tried the Collaborative Filtering model **ALS** (Alternating Least Squares) package using the **Spark** Machine Learning library as our second baseline.

8.2. Factorization Machine (FM)

Most of the recommendation problems present have a dataset that is formed by a collection of (user,item,rating) tuple which is the starting point of most of the collaborative filtering algorithms. However, the yelp data set apart from consisting of the aforementioned tuples consist of a plethora of meta-data regarding the users and the businesses and this can be used to give better predictions. To incorporate this meta-data we decided to use Factorization Machine(FM).

Factorization machines provide a great advantage when used with a feature rich dataset. FM is especially useful in this case as it introduces higher order interactions that are affected by categorical data and the data we are working on consists of a lot of meta-data that's categorical. This also means that the model goes beyond co-occurrences in order to find stronger relationships between latent representation of each feature.

In our analysis, category feature (meta-data for business) is comprises a string of categories (Thai, Italian, Acai Bowls, etc) associated with the business. These provide useful information about the business and therefore we harness them into our model as embeddings in order to get better predictions. Each unique category mentioned in the string of all the business present in the dataset is taken as a column of the dataset and encoded to be used as an input to the model.

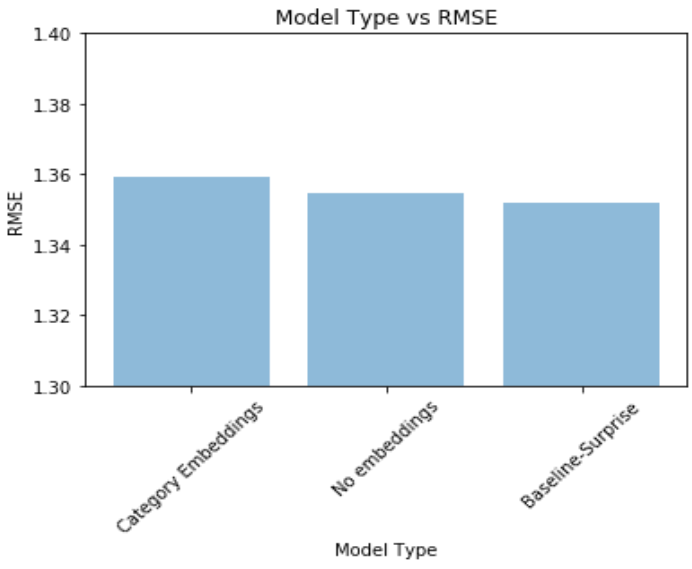
We implement our FM model on two Data Subsets i.e Data Subset 1 and Data Subset 2

For Data Subset 1 and Data Subset 2, we built two models each for the two subsets i.e one with existing features of the user and business data without categories as embeddings and one with existing features of the user and business data without categories as embeddings and see how the two models' accuracies compares against each other as well as against the baseline for their respective subsets.

We use FM models to perform a regression task. We split the both data subsets into training and testing set. Here testing set comprises of the data of all unique users and the last restaurant they rated. After this, we take the rating feature as the response variable and all others as the independent variables and perform regression using FM. The ratings predicted by the model are then compared with the true ratings in the test set and then accuracy is calculated in terms of Root Mean Squared Error(RMSE).

Data Subset 1:

Data Subset 1 consists of data for active users (defined as users that rated more than 5 times) for restaurants and food places that are open and currently functional in all the states along with user information (compliment counts, review counts, average rating) as well as business information(attributes, categories, location, hours).



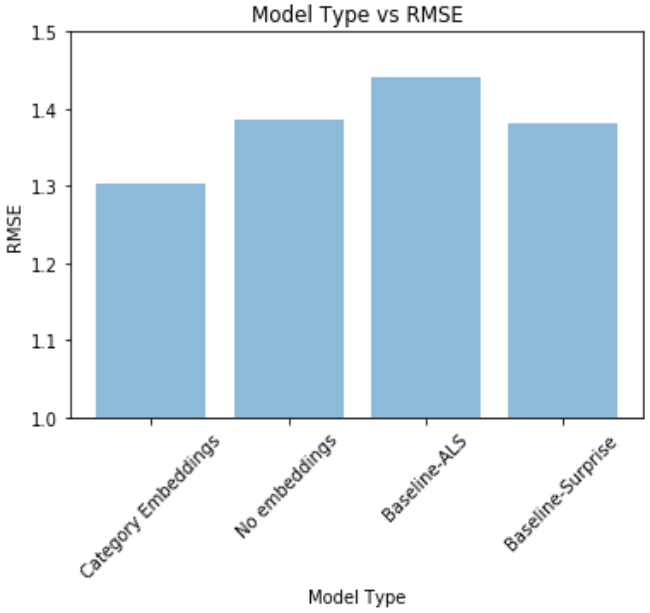
Fig(8.2.1). Comparisons for RMSE for Data Subset 1

The RMSE for the FM with category embeddings is slightly greater than both the one with no embeddings as well as the baseline model(implemented using Surprise package)

Now we implement FM model on Data Subset 2, which is also used for Review Based Recommendation in order to implement an Ensemble model with that model.

Data Subset 2:

Data Subset 2 consists of data for active users (defined as users that rated more than 25 times) for active restaurants (rated more than 25 times) in only Las Vegas city along with user information (compliment counts, review counts, average rating) as well as business information(attributes, categories, location, hours).



Fig(8.2.2). Comparisons of RMSE for Data Subset 2

We can infer the following:

- Our FM model with categories as embeddings performs substantially better than both the baselines (ALS as well as Surprise) as well as our FM model without categories as embeddings.
- Our FM model with categories without embeddings performs better than the ALS model but performs slightly worse than the Surprise baseline model.

8.3. Review based Recommendation using NLP

In the given dataset we have an interesting field that represents the reviews given by a user for the particular business. The inspiration to use these reviews to make a recommendation system was to use Content Based systems to generate personalized user content by understanding the similarity between the contents of different types of items or users.

With the theme of content-based recommendation in mind we further thought of collecting all the reviews given by a particular user and apply Natural Language Processing over all of them and convert each of those into a latent space (a vector represents a user). Each vector can be thought of as representing a user in the latent space and the information we can get from the vector is the overall liking of the user. This follows due to the fact that we have collected all the reviews given by a certain user and then represented it into the latent space and hence, by this we have a preference (if not exactly the preference at least the understanding of the user's preferences).

We do a similar thing with all the businesses listed and also represent them into a latent space of vectors. Here each vector represents the business and hence would be carrying the information about the peculiar things the business is being talked about by the different users who have visited the place. This can be thought of as the key points the business is famous for and can be used to be matched with users who have preferences for similar things.

This can be thought of as getting similarity between user's likings and businesses' offerings and using those to generate recommendations. To put it words "by all the reviews of the users we know user X likes or talks of the particular things a lot. From the reviews of businesses, we know business Y is being talked about for some particular things. If it turns out that the lists are similar for X and Y, we say that user X would like to visit place Y." For example: Consider a situation when the word "pizza" is very frequent both in the reviews for a restaurant, and also in the reviews of a user. The intuition behind this could be that everybody is talking about the pizzas of this restaurant, and this user happens to care about the pizza a lot, therefore they can be somehow matched up. At this point, we really cannot tell whether this user would like this restaurant (it could be that everybody is actually complaining about the pizzas), but still, there is some relationship between this user and this restaurant. At the regression step, we still use locally weighted linear regression and hence can then tell whether user would actually like the place or not by using the predicted value of the ratings.

We thus, concatenate all the reviews given by a user as well concatenate all the reviews for a particular business. To get a vector into the latent space we use the Bag of words model and use the TF-IDF vectorizer to get the vector representation. After getting the vectors for each user and business we create a rating matrix kind of structure which has similarity as its value for the corresponding cell between user and business. The matrix has users as its rows. Each row is representing the preference vector for the user as it can be thought of as representation of

weight for each of the businesses. We follow it by using regression to predict the ratings for each of the businesses and use the predicted ratings to generate recommendations for users. The similarity values are the cosine similarity computed between the vectors of user and a business.

TF-IDF: TF-IDF stands for term frequency-inverse document frequency, and the TF-IDF weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the TF-IDF weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. (“http://www.tfidf.com/”) We used the TF-IDF vectorizer of the scikit-learn to generate the TF-IDF vectors.

For this particular task we used the businesses listed in Las Vegas only. We restricted ourselves to exploring the users who have at least reviewed 25 times and also the businesses that have at least 25 reviews for them. This was done as the number of reviews that are required to gauge the behaviour of users and the peculiar things about a business requires more reviews to perform better. The same thing can be done with a smaller number of reviews as well but it creates 2 problems:

1. The smaller number of reviews don’t actually help you understand the behaviour of user’s having reviewed only a few times and the same follows for the businesses.

2. Another problem we encountered was that, keeping the threshold number less than 25, increased the number of users and businesses to be compared. As we are using pairwise similarity computation, it exhausted the computing capabilities and couldn’t be computed on a single device (A case when we are computing a pairwise similarity computation between 1 million things could lead to Petabytes of memory requirements; creating a 10^6 x 10^6 matrix in memory means 10^{12} floats. A python float takes 16 bytes so we end up with $16*10^{12}$ bytes, which translates to ~4PT (four petabytes) of RAM.) To avoid this situation, we need to parallelize the computation of similarities, but as we were keen on seeing the proof of the concept, we restricted ourselves to limiting the subset for exploration.

We got a training RMSE of 1.40 while the testing RMSE was 1.55. The test set used for computing this value of RMSE was the one having the last rating for each user. When split randomly to create a test set, we got training RMSE of 1.41 and testing RMSE of 1.40. This shows that the model, in general, was able to learn the pattern but wasn’t able to generalize itself to particularly predict the last ratings b a user. Hence it is advisable to use it as a recommendation engine providing top-k recommendations as it would then neutralize the thing for not being able to predict a particular value of rating.

From this implementation, we can say that reviews help to predict the ratings and can be used to gauge the user preferences and the offering by business and match two of those to generate match ups and hence recommendation engine to recommend a business to a user.

8.4. Image based Recommendation using Image Embeddings

We used a subset 3 of the main dataset for this model of ours. In this content based model we use the images (images labeled as food) of the businesses to find similarities between the last business we are trying to predict the score for and the previous businesses the user has rated.

We are able to do this as the dataset has a list of images for each business, and these images are labelled as either food, indoor, outdoor and menu. There is a lot of noise in the indoor and outdoor images and we can't actually say anything about the similarities in businesses just based on how the businesses look on the outside and inside. That is why we are taking the images labeled as food from here on. The images labeled as food mostly contain the images of the food that is served at these places (though there is some noise in the images, some of them contain images of people eating these foods instead of just food images). For different businesses sometimes there are exact same image (i.e. an image say i1 is present in business b1 and business b2 with different names), these kinds of images are not considered for similarity.

The intuition:

The main intuition behind the methodology we are going to discuss is that similar businesses (restaurants in our case as we have taken restaurants) serve similar food items, for example a pizza place will serve pizzas and two pizza places are similar to each other as they serve the same items. So in this procedure we are taking the similarity of the items served in the businesses as a measure for calculating the similarity of the businesses.

For generating the embeddings from the images we will be using image2vec python library, this library has two trained models based on two different architectures (Resnet-18 and Alexnet). It is basically a Convolutional Neural Network which was trained on over a million images of the ImageNet database. This network has 18 layers and can classify images into a 1000 different categories. This library uses pre-trained weights on this architecture by transfer learning and removes the last dense layers of the classifier and gives 512 embedding values for each image. These embedding would encode the content of the images in lower dimensions. Thus we can use this embedding vector to get the approximate similarity between the images.

Procedure:

1. Get all the images of the businesses in the subset and upload them to Google drive (As we are using Google colab to run our codes).

2. Use the img2vec library to generate the embeddings for all of these images and store these embeddings in a dictionary with key as the image ID and the values as the embeddings.

3. For each image belonging to the last business the user has rated, compare it with all the images of all the businesses the user has previously rated (this comparison is done by calculating the euclidean distance between the image embeddings).

4. For each image of the last rated business of the user, take its top k1 (our hyperparameter 1) closest image neighbors from all the images of the businesses the user has previously visited. Suppose there are 6 images in the last rated business of the user we will get 6*K1 image neighbors in totality for the last rated business. These will be in the form of a list of tuples. Example: [(business1,distance),(business2, dist2)] etc. We are taking the business ID as the first element in the tuple and not the image ID, we are doing this by getting the corresponding business ID of the image that is close to the current image.

5. After doing the 4rth step we will notice that there can be same business that is in the top 6 closest images for each image of the last business the user has rated. Consider this example, suppose there are 3 images that are closest to the one of the images of the last business the user has rated let us call them img1,img2,img3 and say img1 and img3 belong to the same business call it b1(this can be possible as the image we are searching against can be a pizza and both the images img1 and img3 can be images of different pizzas from a business b1) and say that img2 belongs to a business b2. Now since we are representing them as tuples with their corresponding business ID and the distance between their images, the representation for the aforementioned example would be [(b1,d1),(b2,d2),(b1,d3)]. So we will collect the business IDs b1 and take the distance as the average of d1 and d3, the reasoning here being that if a business is serving different types of dishes that are similar to the dishes the current business we are considering we will have to account for both the dishes. So the new representation for our example would be [(b1,(d1+d3)/2),(b2,d2)].

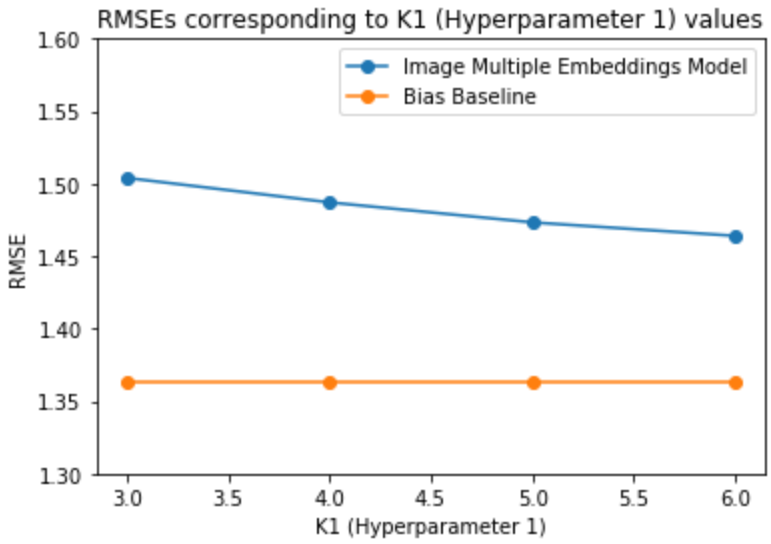
6. Then from the list of tuples obtained we will be taking top k2 (2nd hyper-parameter) closest businesses to the last business the user has rated (euclidean distance formula is used to calculate the distance). This will be stored as a dictionary with the key as the user ID and the value as a list of tuples with the 1st element in the tuple being the business ID and the second element being the distance of this business from the last business the user has rated last.

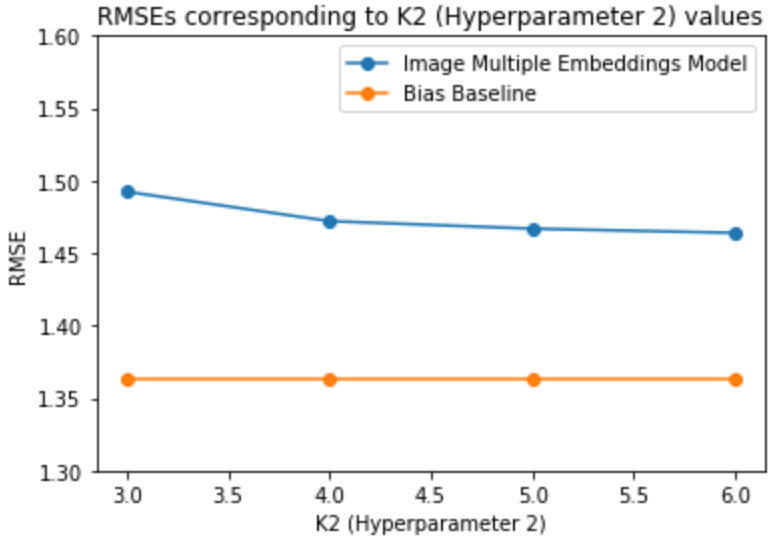
7. We then use the inverse of the distance (1/distance) as the similarity metric between the businesses and predicting the rating of the last rated business by taking the weighted average of the ratings this user gave to its k2 closest neighbors.

RMSE of this model with various combinations of hyper-parameters:

K1 (hyperparameter 1)	**K2 (hyperparameter 2)**	**RMSE**
3	3	1.522
3	6	1.5040
4	6	1.4871
5	6	1.4733
6	6	1.4640
6	3	1.4924
6	4	1.4721
6	5	1.4669

The RMSE obtained on this subset by our ALS based baseline is 1.775 and the RMSE obtained by user-item bias based baseline is 1.363.





The first plot represents the variation of the RMSE score with change in the 1st hyper-parameter value when kept the 2nd hyper-parameter as constant (set to a value of 6). The x-axis represents the value of the 1st hyper-parameter, and we can see that as the value of the hyper-parameter 1 is increasing, the RMSE is getting better (decreasing). The second plot represents the

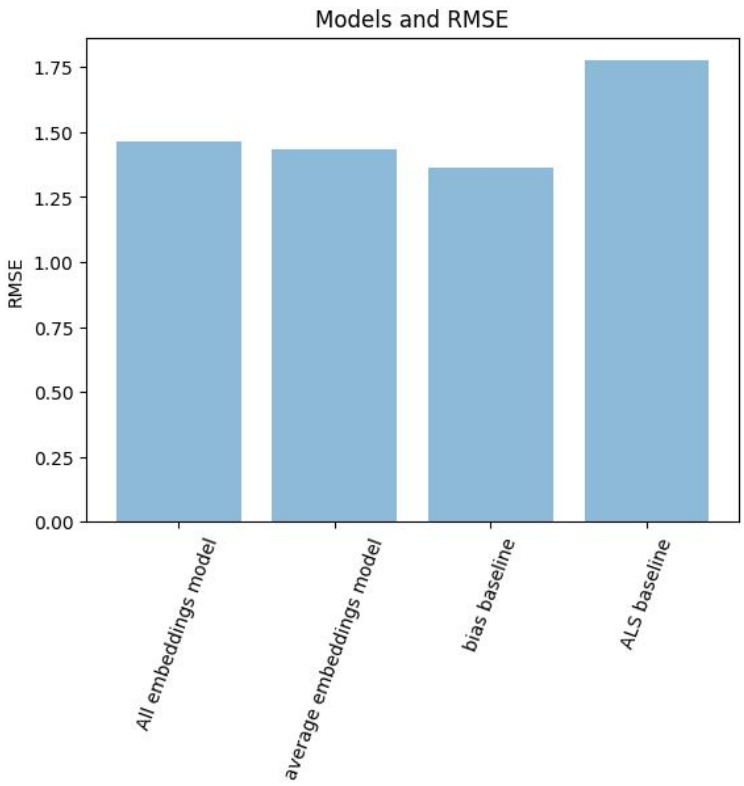
Modification 1:

Comparing the obtained RMSEs with the baselines we felt that our model could perform better, so we made a modification to our model, this time instead of each business having multiple embeddings (one per each image it has), we have averaged all the embeddings of each image of a business to one embedding vector. This gives us one embedding for each business. Now we just compare the embeddings of the last business to the embeddings of all the businesses the user had previously rated. Now instead of taking only top k2 neighbors we are taking the weighted average over all the businesses the user has previously rated. Note that this methodology doesn't make use of hyper parameters like the previous one.

The average embeddings model doesn't have hyper parameters, as we are averaging out all the embeddings of a business into one single vector of embeddings (this will leave us with one embedding vector for one business, so we don't have hyper parameter 1(k1)) and we are using all the previous businesses a user has rated to calculate the score for the last business (so there is no hyper-parameter 2).

Computing the RMSE after this modification, we have attained an RMSE of 1.4355 which is better than the RMSE of any combination of hyper parameters from the previous model. But this still couldn't beat the bias based baseline model. So we then made an ensemble model with the predicted scores from the ALS and the predicted scores from our model, and we will learn the weightage to be given to each model using regression. (More about this ensemble method in the section 8.5.2)

Model	**RMSE**
Model with multiple (all) embeddings for each business	1.4640
Model with average embedding for each business	1.4355
Bias Baseline	1.363
ALS Baseline	1.775



The above plot shows the RMSEs of the two models we have made v/s the RMSEs of the baselines we have used. It can be seen that the standalone image based models performed better than the ALS baseline but couldn't perform better than the bias baseline. This could be attributed to the noise (we have said how the label (like 'food') and the actual content of the image is not matching) in the images of the dataset and the smaller size of the images in the dataset.

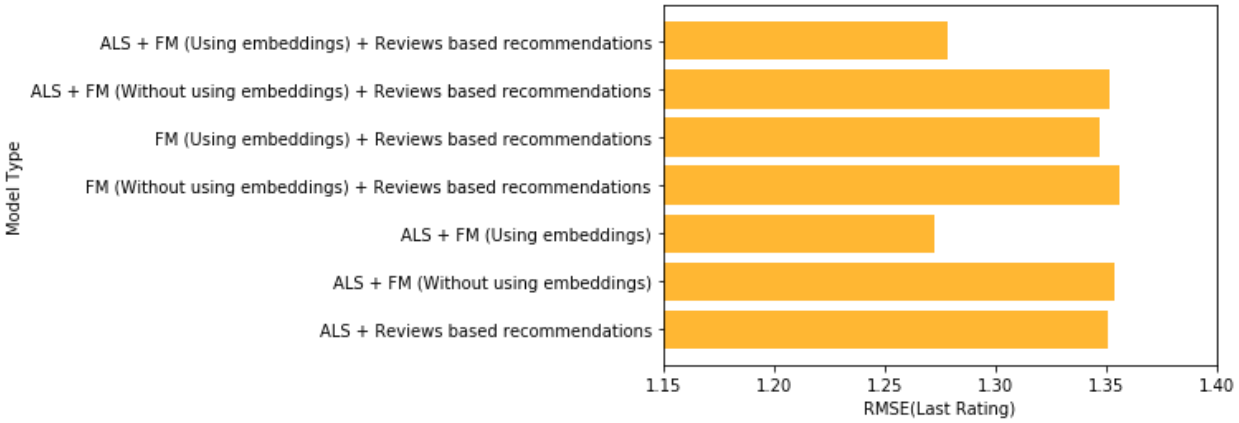
8.5. Ensemble Methods

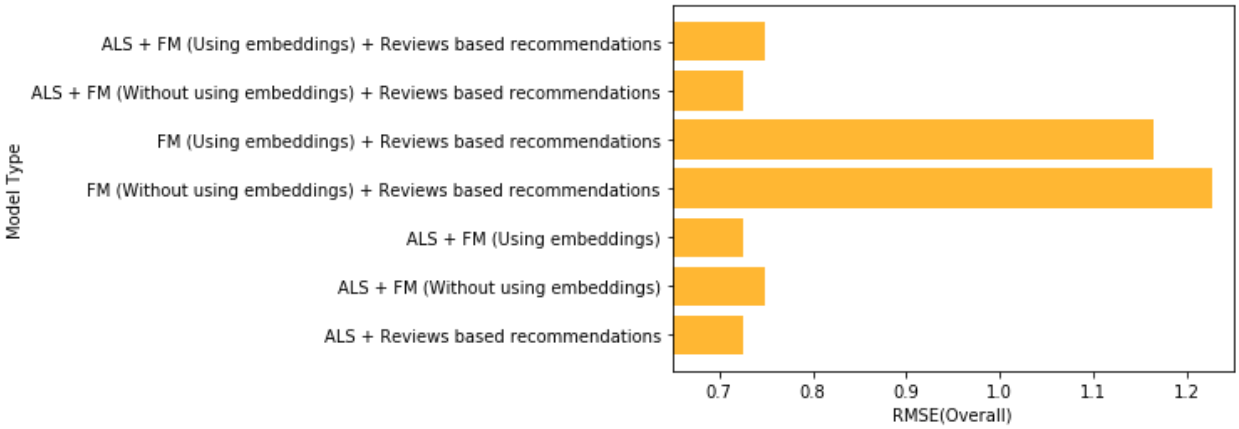
Ensemble methods are meta-algorithms that combine several machine learning techniques into one predictive model in order to decrease variance (bagging), bias (boosting) or improve predictions.

8.5.1 Ensemble using ALS, FM and Review based Recommendation

We created ensemble method using different combinations of the methods and this were the findings of each of those. Here we have shown two columns for RMSE. The first one is for the test set defined as the set of business, user pairs that the user has last rated. The second one is the one calculated all the ratings and can be thus, used to gauge the performance of the model considering all the businesses.

Model	**RMSE (Last Ratings)**	**RMSE(Overall)**
ALS + Reviews based recommendations	1.3513	0.7254
ALS + FM (Without using embeddings)	1.3541	0.7482
ALS + FM (Using embeddings)	1.2727	0.7256
FM (Without using embeddings) + Reviews based recommendations	1.3562	1.2268
FM (Using embeddings) + Reviews based recommendations	1.3470	1.1649
ALS + FM (Without using embeddings) + Reviews based recommendations	1.3516	0.7253
ALS + FM (Using embeddings) + Reviews based recommendations	1.2785	0.7482





On subset without embeddings

- For ALS **Precision** = 0.8637 **Recall** = 0.9418
- For review based **Precision** = 0.8301 **Recall** = 0.9130
- For factorization machine **Precision** = 0.8168 **Recall** = 1.0
- For ALS + FM **Precision** = 0.8670 **Recall** = 0.9329
- For ALS + Review based **Precision** = 0.8661 **Recall** = 0.9355
- For FM + Review based **Precision** = 0.8168 **Recall** = 1.0
- For ALS + Review Based + FM **Precision** = 0.8664 **Recall** = 0.9350

On subset with embeddings

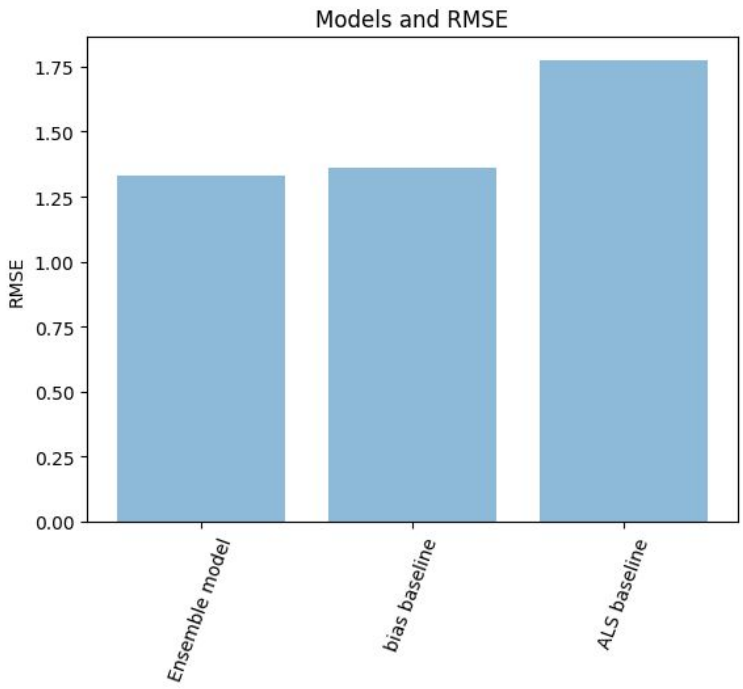
- For ALS **Precision** = 0.8607 **Recall** = 0.9516
- For review based **Precision** = 0.8102 **Recall** = 0.9609
- For factorization machine **Precision** = 0.8002 **Recall** = 1.0

- For ALS + FM **Precision** = 0.8612 **Recall** = 0.9485
- For FM + Review based **Precision** = 0.8005 **Recall** = 1.0
- For ALS + Review Based + FM **Precision** = 0.8613 **Recall** = 0.9464

8.5.2 Ensemble using ALS and Image based Recommendation

We have seen that standalone image based models could not perform better than the bias baseline so now we will use our image based model in conjunction with the ALS to get the predicted score of the last rated business. We are not however giving equal weights for the predictions for both of these models though. We will use the actual ratings the user has given to the last business and fit a regression model with one variable for the rating our model predicted and another variable for the rating ALS predicted. Then the regression model will learn the coefficients for these both variables. These coefficients will indicate what weight should be given to the predicted rating of ALS and what weight should be given to the image based model. On running regression and getting the coefficients we found that more weight was given to the image based model than ALS.

Model	**RMSE**
Image Model with Average Embeddings + ALS Ensemble	1.331
Bias Baseline	1.363
ALS Baseline	1.775



We can see that this ensemble model successfully out performed (though by a small amount) the bias based baselines. We can hence say that making ensemble models with image based models worked well in our scenario than using standalone models.

For Image based ensemble models:

- Model with multiple (all) embeddings for each business **Precision** = 0.8196 **Recall** = 0.9292
- Model with average embedding for each business **Precision** = 0.8177 **Recall** = 0.9439
- Ensemble of ALS and image based model **Precision** = 0.7964 **Recall** = 0.9988

8.6 Serving Recommendations

Since our goal was defined to predict the rating of the last business the user has been to, we will associate our recommendation task with the predictions we made for the last business.

Suppose if we predict that the user would give the last business a score of over 2.5 (>2.5) we can assume that according to our prediction the user would like the last business he's been to above average. For such users we will recommend businesses that are similar to the last business he has been to. We do this by first one hot encoding the categories of the businesses and generating embedding vectors (list with one hot encoded categories) for each business. Then we take the embeddings of the last business the user has rated and get its 5 closest neighbors based on hamming distance. We suggest these 5 businesses to that user. Now for the users that haven't liked the last business they have been to we will suggest top 5 businesses from the ALS model trained on the same subset. This way we are seeing to that all the users in the subset are getting covered.

Note: Since our ensemble of ALS + FM (Using embeddings) model performed better than all our models we are getting the predicted threshold from the predictions of this model.

Running catalogue coverage on the recommendations we provided through our model gave us a coverage of 0.6765, which says that our model is recommending **67.65%** of the unique businesses present in the set at least once.

9. Future Scope

- We thus have created a proof of concept for the working of the recommendation engine using images and reviews combined with factorization machine and Alternate Least Squares method. The POC stills needs to be modified for being able to deploy the same into an industry. For that we need to have distribution of computation so that we can perform computationally intensive tasks, which would enable us to have more amount of data into consideration.

- If we have the images labelled as the ones uploaded by the users so having user_id associated with a particular image then we can use those to do a similar task as we did for the reviews and use the images to understand the user's preferences and then use those to build the recommendation system.

- For the Reviews based prediction presently we have used Bag of Words model followed by TF-IDF Vectorizer to create the latent space vectors. We can use a RNN (recurrent neural network) to understand the reviews and their association with ratings. Once the network is trained to predict the ratings from the intermediate flattened layer of the network to get the embeddings for the reviews and use those as the representation for the vectors in latent space. (We have created 3 networks (LSTM, Convolution followed by LSTM, Glove based)for the same task which can be found in the repository).

- Further, we can also use the data for friends of the user to create a social network and use the social network based recommendation engine for creating recommendations for a particular user. This could solve cold start problem as we might be able to generate better recommendations using the data available for the friends of a particular user. The location based recommendations can also be generated using this method as a user can have multiple friends in different locations and those can be used to give recommendations to the user when he/she visits that area.