ILS-Z: 534 SEARCH (Fall 2016)
Yelp Dataset Challenge
Final Project Report

Introduction

The objective of this project is to perform two challenging tasks on Yelp dataset. The first task is about predicting the Category of the restaurant using reviews and tips. The second task is to observe, analyze and model the user behavior and recommend the restaurants they would most likely visit in future based on his/her cuisine preferences.

Yelp Dataset

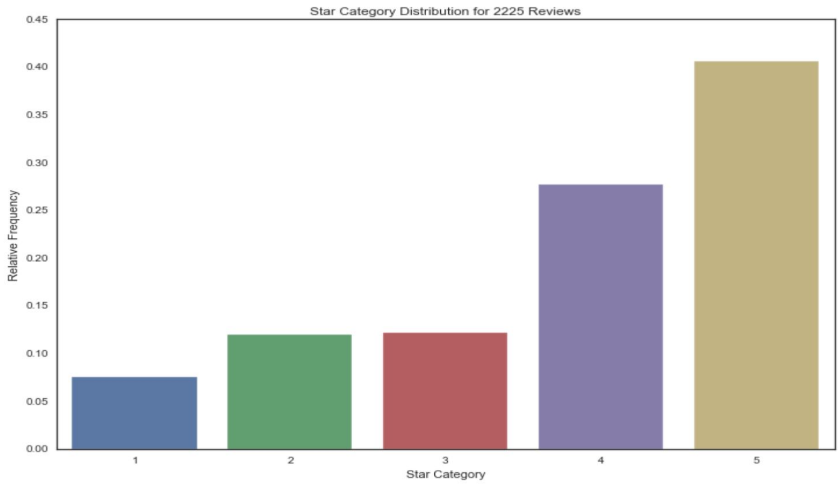
Exploratory Data Analysis

To help visualize and gain insights on the data, we have performed in-depth exploratory analysis and tried to understand the data. Yelp has made a subset of its data publicly available in the context of Yelp Dataset Challenge. Data is provided for users, tips, reviews and businesses. The data which is provided in Json format is huge with a little under two million data records.

The following figures give an insight about the entire data and the way we have split it for testing and validation purposes.

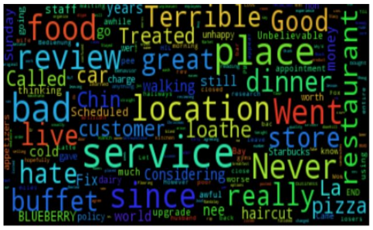
- Train: 1,940,708 records
- Test: 523,438 records
- Tuning: 523,438 records

In order to create a dataset with similar distribution from the test data, we have adopted stratified sampling for validation and tuning, and joined the Reviews, Tips and Business JSON files using Apache Spark.

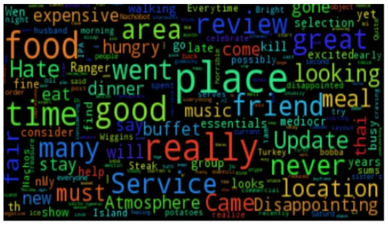


We analyzed the ratings or stars given by the users and observed that majority of the users have given positive ratings. The reviews with corresponding stars as Five or Four are observed to have a higher relative frequency than Three, Two and One stars.

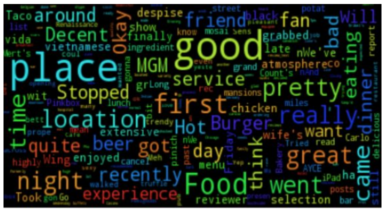
Word clouds were further constructed to observe and get a high level understanding about the text reviews. It was noted that reviews with One or Two stars had a majority of negative words which is justifiable with the corresponding stars of a review. While the reviews with Four and Five stars obviously had positive words, the reviews with a Three star rating too were observed to have positive words which has later helped us in giving weights accordingly, while working on task two.



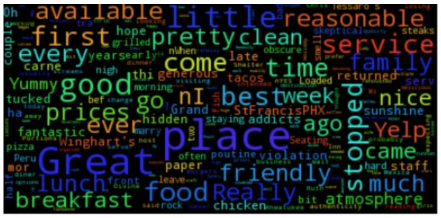
★ Word Cloud



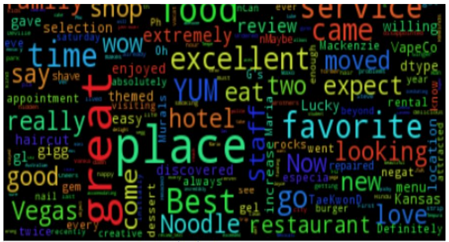
★ ★ Word Cloud



★★★ Word Cloud



★★★★ Word Cloud



★★★★★ Word Cloud

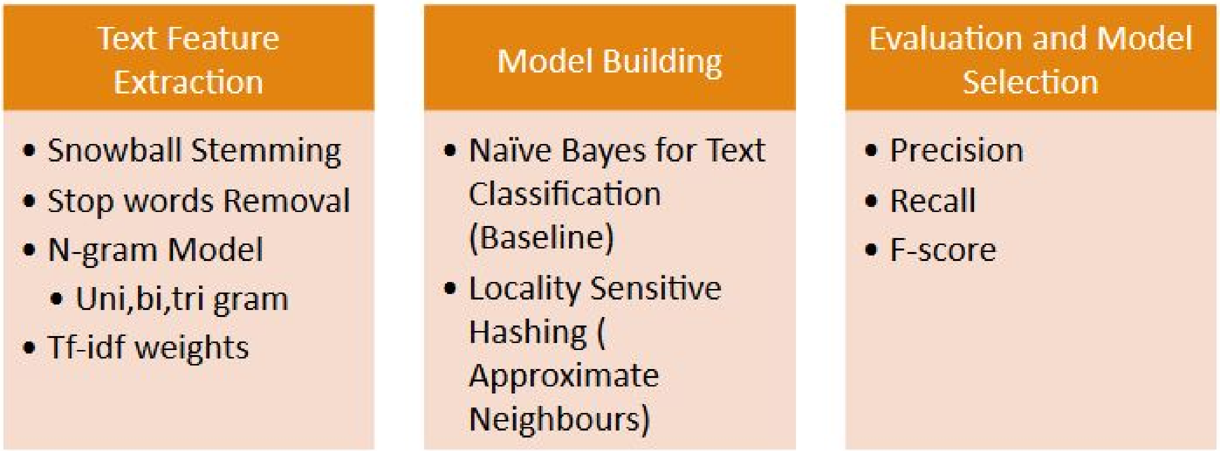
Task 1

“Predicting the category of the restaurant using reviews and tips”

Data Preparation

The data for this task is from the “Yelp Dataset”. The files of interest are business.json (File containing information about Businesses), Tip.json (Information about tips user had left for the business) and reviews.json(The reviews themselves). These files are joined in memory using a spark map-reduce framework and the processed files are saved in blocks of about 12 MB each. There were about 118 of these, constituting a little more than 1.4 GB of data. The reason for fragmenting a single larger file into smaller pieces is that these smaller pieces can be processed parallely using a map-reduce framework. Also since most of this data is textual data, the processing of one file can be independant of each other. Thus we this framework provisions itself for efficient parallel processing framework later.

Data Pipeline



Text Feature Extraction

- Snowball Stemming
- Stop words Removal
- N-gram Model
  - Uni,bi,tri gram
- Tf-idf weights

Model Building

- Naïve Bayes for Text Classification (Baseline)
- Locality Sensitive Hashing (Approximate Neighbours)

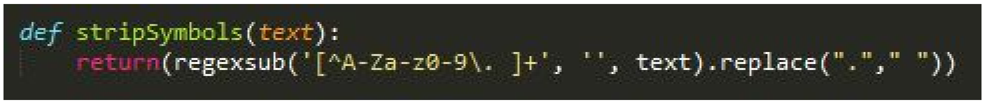
Evaluation and Model Selection

- Precision
- Recall
- F-score

Fig - 1 : Blocked representation of the data pipeline

The reviews are plain text that are user written and hence has spelling mistakes, emoticons, numbers, URL links etc. The first part of the data processing was to handle these and standardize them.

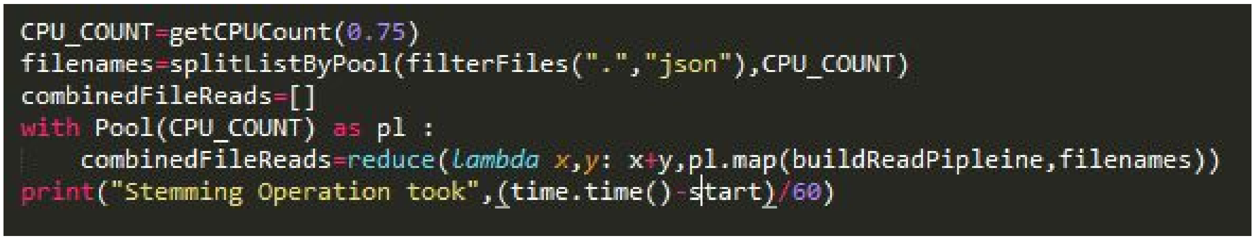
A Regex expression was used to clear the smileys, http URLs and words that aren't part of the English language. We chose to replace "." by " " (space) because of the semantic meaning they offer to the passage.



```
def stripSymbols(text):
    return(regexsub('[^A-Za-z0-9\. ]+', '', text).replace("."," "))
```

In order to extract features from text, Snowball Stemmer and Stopwords from NLTK library was used for stemming and a reference corpus for stop words.

Since the data files are split into smaller manageable files now, and we could leverage multiple processors in the CPU for handling text processing. We use Multiprocessing python library which uses a map-reduce framework for assigning a function to mapper instance and reduces the result into one single instance as given by the reduce function.



```
CPU_COUNT=getCPUCount(0.75)
filenames=splitListByPool(filterFiles(".","json"),CPU_COUNT)
combinedFileReads=[]
with Pool(CPU_COUNT) as pl :
    combinedFileReads=reduce(lambda x,y: x+y,pl.map(buildReadPipleine,filenames))
print("Stemming Operation took",(time.time()-start)/60)
```

We deployed our code in hulk.soic.indiana.edu which as about 54 cores, we leverage 75% of the cores to spawn child process to develop the data pipleine(Stemming, Stopword removal, Symbols handline etc) and return a clean processed string for further analysis. It was seen that the parallel processing decreases the processing time to about 6 minutes compared to the original one hour plus processing time.

Also we have routines to create ngram of the words. We generate features from unigram, bigram and trigram word distributions.

Task -1, Baseline Algorithm : Naive Bayes

Categorizing a restaurant is a multi label classification problem. We considered many other discriminative methods like SVM and logistic regression. However we quickly realized these methods wouldn't work because the data had about 1368 categories, so SVM and logistic regression and boosting methods like AdaBoost has to be an ensemble of 1368 "One Vs All" classifiers. Also since we have large data points, quadratic optimization of SVM becomes very slow and takes lot of time when the data points are larger than 20,000. So we needed a generative model which could do a multi-label classification and Naive Bayes stood out because the data processing was just counting. The training and validation and Test was split as 60%, 20% and 20%. This was stratified spilt to keep the distribution as same as test dataset. Moreover 60% of the feature vectors were unpacked one per a category to reproduce the multi class distribution problem. I.e <Feature><C1,C2> was unpacked as <Feature><C1> and <Feature><C2>

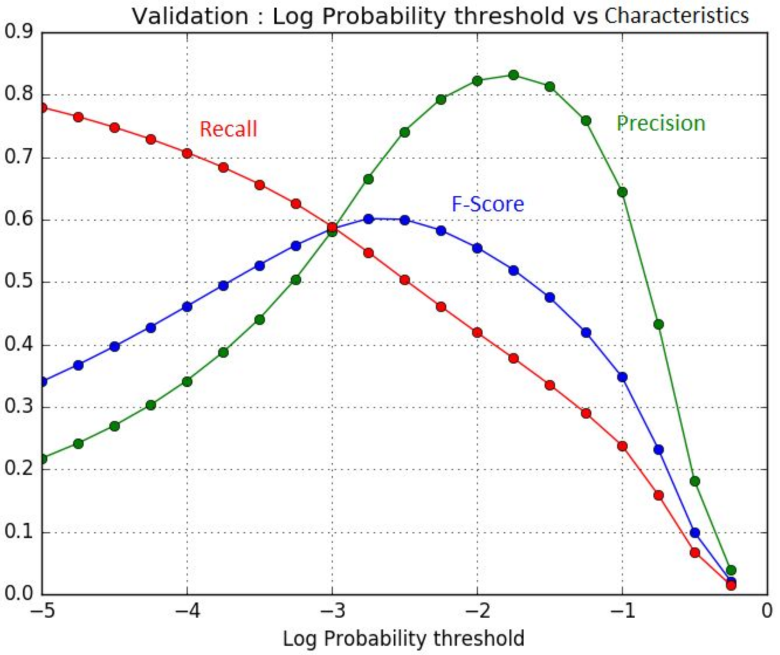
Proposed Modifications to Naive Bayes Method

Usually class with maximum estimated probability is returned as the most probable class. However since this is a multi class problem, we intend to modify the original algorithm by returning all classes with a probability greater than the threshold.

This threshold function is learnt using the validation dataset. The method goes to predict log probabilities for all class, and validation dataset was used to find the threshold.

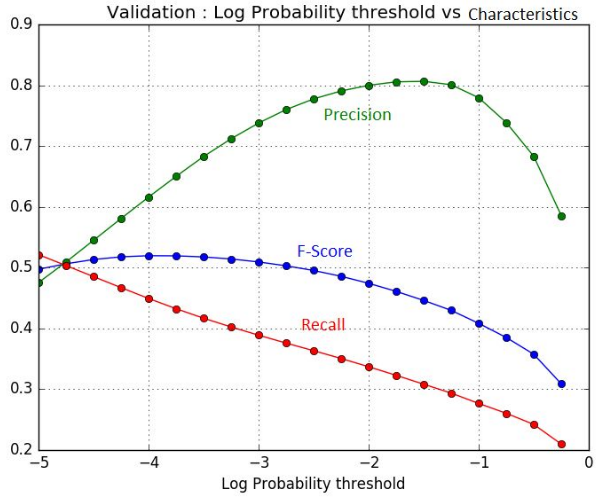
Results

Unigram Results

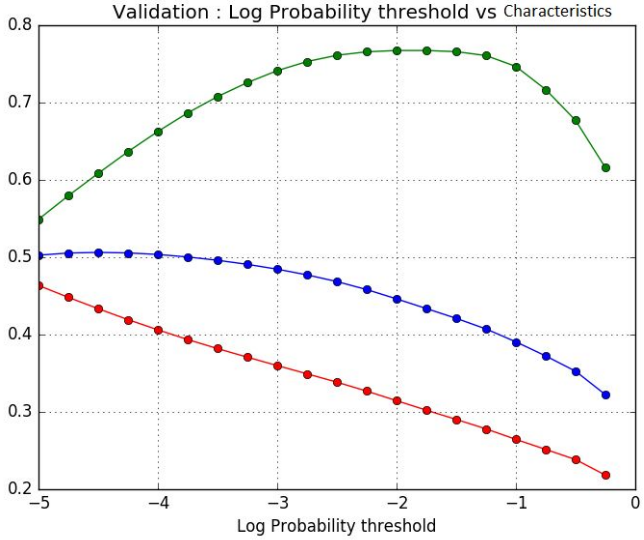


As seen on X-axis, these are log probabilities and Y-axis are characteristics (F-score, Precision and Recall). The machine learner aims to learn a right balance of f-score by varying the threshold log probability. In log probability scale, 0 is probability of event occurring at 1. It can be seen that precision and recall increases till a break even point after which precision increases and recall decreases. This is because of the fact that as the threshold decreases, lot of categories become candidate categories, and many of these might be correct. Thus, precision increases, however recall decreases. F-score is a balance between precision and recall and the highest. It wouldn’t be fair to use ranking metrics here like recall @5 because there is no inherent order between the class categories.

For bIgram and Trigram, characteristics are as follows.



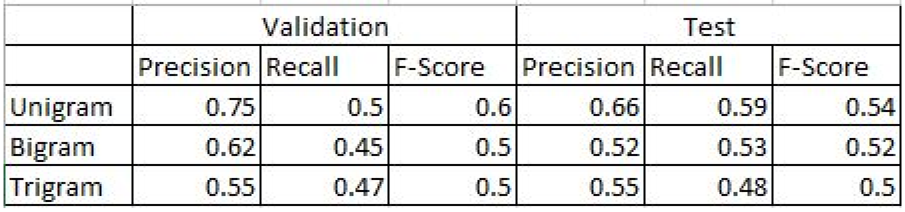
Bigram Results



Trigram Results

It can be seen that precision and recall decreases with increase in n-grams. Unigram offered the most precision and recall than the others. The f-score varied between the models as unigram > bigram > trigram. This result was surprisingly common between other teams and more focused study is required to investigate this phenomenon.

Summarized results are of the unigram, bi-gram and tri-gram are as follows.



	Validation			Test		
	Precision	Recall	F-Score	Precision	Recall	F-Score
Unigram	0.75	0.5	0.6	0.66	0.59	0.54
Bigram	0.62	0.45	0.5	0.52	0.53	0.52
Trigram	0.55	0.47	0.5	0.55	0.48	0.5

Results Discussion

From the above results, the unigram seems to have the highest score. It is unusual that the higher n-grams offer less precision and recall as the lower n-grams. With this as baseline, we also compare the result with LSH, locality sensitive hashing.

Approach 2 : Locality Sensitive Hashing

Locality sensitive hashing (LSH) is one effective technique of reducing high dimensional data. The basic idea of LSH is that if we have documents that are very similar to each other we would generate similar hash codes for those documents. LSH aims to maximize the probability of hashing for similar items. When LSH finds two similar documents it simply hashes them in a way that similar items are put in buckets with high probability. Bit sampling is one of the easiest ways in which LSH is used.

Changes made to LSH

In order for us to effectively deal with our multi label classification problem for this task we implement min hash technique along with LSH to obtain maximum efficiency. Using min hash we quickly estimate how similar two sets of classes are. Min hash compares two classes and gives out similarity values. Based on the similarity the hashing function would combine similar classes together and put them as our target class. This approach is ideal because we are looking at finding similar training classes and not the exact class labels.

Since our dataset is huge LSH was considered over other algorithms. Infact when we used nearest neighbors to estimate the nearest neighbour match we were not able to achieve similar class grouping effectively. LSH does a more efficient approach than nearest neighbors. We followed a process using LSH in which

- LSH Finds k nearest neighbours
- Generates similar hash codes
- Then we use the hashing by LSH to combine all classes of training class as predicted class

Implementation

The implementation was carried out using python's scikit learn package LSH forest. LSH forest is python's own for generating approximate nearest neighbours using local sensitive hashing forest.

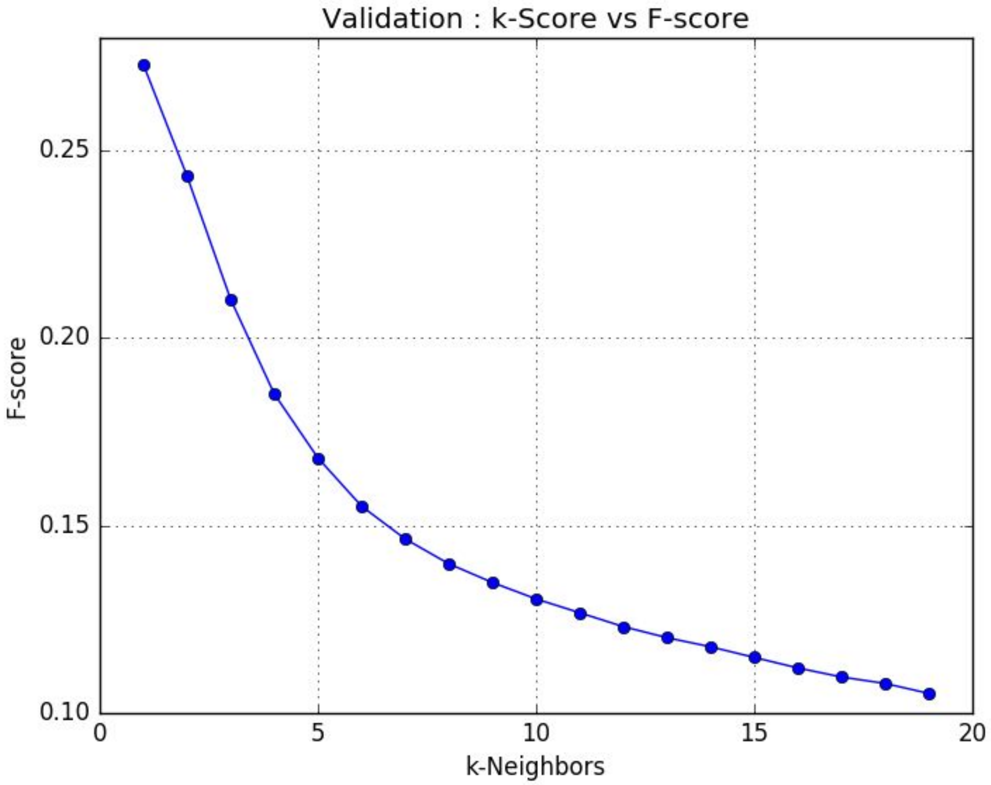
Parameters

N_estimators - Number of trees
N_candidates - Candidates evaluated per estimator
N_neighbors - Number of neighbors to be returned

Results

1. **Evaluation 1**

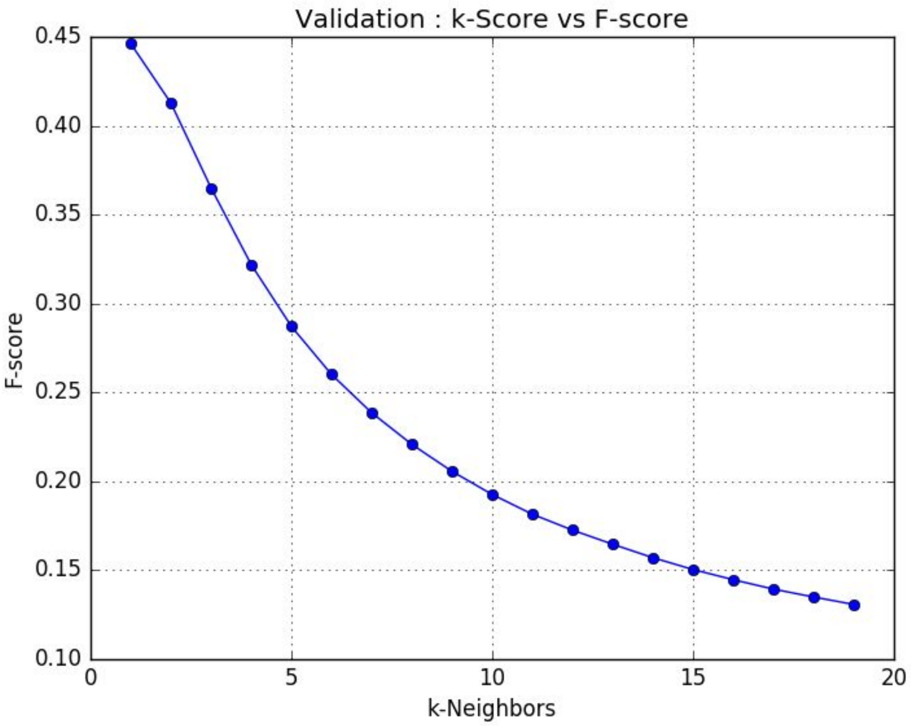
Set
N_estimators - 1
N_candidates - 10



The F-score when we set the parameters to the above values is 0.27 for 1-NN and it decreases as we increase to K neighbors. The evaluation was worser than our Naive bayes baseline and hence this result was sidelined and considered below par.

2. Evaluation 2

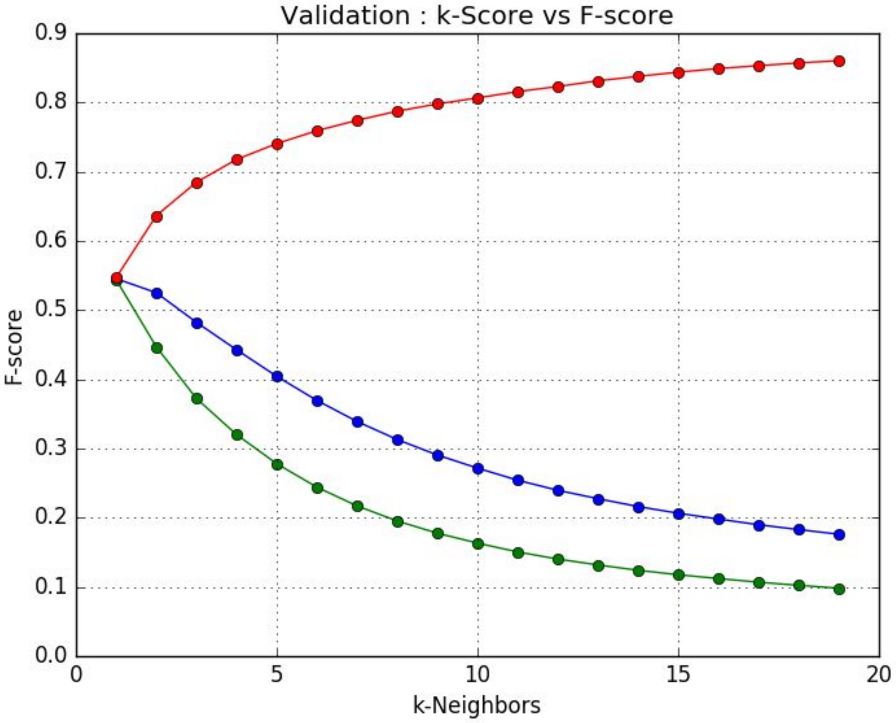
Set
N_estimators - 10
N_candidates - 10



The parameters were then varied and the number of estimators were now increased to 10 while the candidates per estimator was still kept at 10. The F-score 1-NN for this trial was 0.45 and it decreased as we progressed to K neighbors. The evaluation was better than our previous trial but was worser than our Naive bayes baseline. This showed that the varying the number of estimators comprehensively improved our results.

3. Evaluation 3

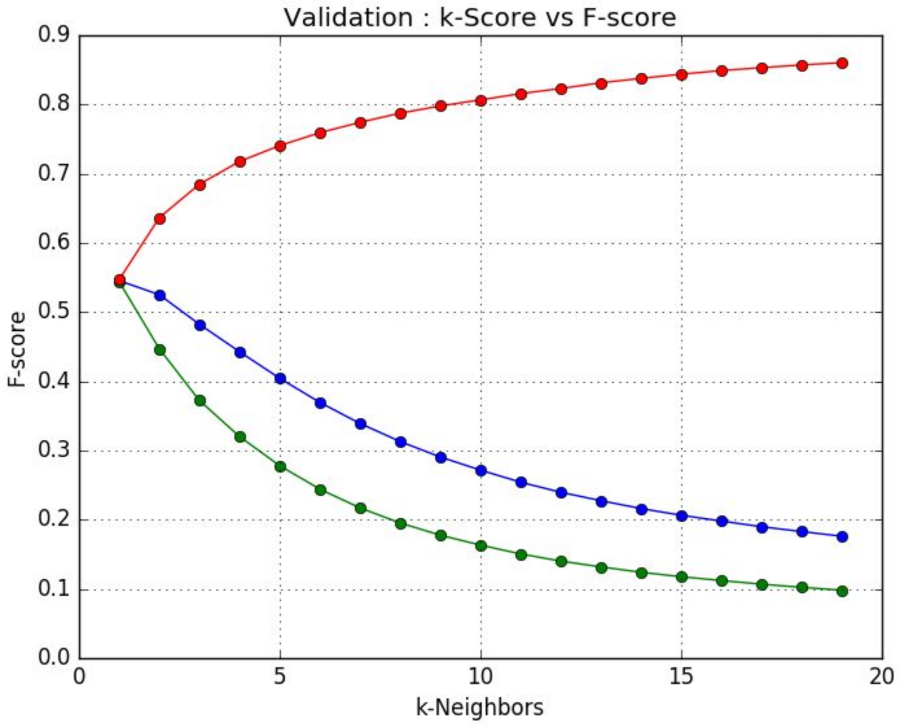
Set
N_estimators - 10
N_candidates - 50



The parameters were varied again and the number of estimators were kept as before while the number of candidates were now increased to 50. The F-score for 1-NN showed substantial progress as the score was at 0.55 for this trial. The evaluation was better than our previous trial and also better that our Naive bayes baseline. This showed that LSH had the capacity to outperform our baseline.

4. Evaluation 4

Set
N_estimators - 30
N_candidates - 500



The parameters were varied again and this time we increased both the estimators and candidates more than the previous time. The F-score for 1-NN showed an amazing progress of upto 0.67 for 1-NN for this trial with the precision and recall at 0.60 and 0.76 respectively. This was the best result that we obtained from our run using LSH

Summary

At the end of task 1

- Our baseline model Naive bayes gave us a F-score of 0.59
- The LSH model with 30 LSH trees and 500 candidates outperformed our Naive Bayes model and gave us a F-score of 0.67
- The LSH model is the best amongst the models that we took for our evaluation.

Task 2

Introduction

The proposed research question for Task-2 is,

“Modeling user behaviour and recommending the restaurants they would visit in the future based on cuisine”

As we know, for any business to offer satisfactory services to its customers, it is very important to understand the needs of the user. Most of the times, this is not quite explicit. So, it comes over to the business to determine the needs and behaviour of the users.

To solve this challenge, the Yelp Dataset is used to get information about the users and the underlying trends in their behaviour. The Yelp Dataset as we know is very challenging to deal with not only because of the way it is structured but mostly because of its sheer size.

To summarize, the data from the Yelp challenge is extracted, filtered to suit the requirement, employed to develop a predictive model, the results are visualized in the form of graphs and a comparative approach is used to produce and show the results.

Methodology

The main steps in the employed methodology can be described as follows:

- Data Extraction
- Data Preprocessing
- Data Modelling
- Data Analysis
- Data Visualization

Data Extraction

As mentioned, the primary dataset used for analysis is the Yelp Dataset from the official challenge. There are multiple associated data files as part of the entire dataset. All of the data is is publicly available and can be downloaded from the official website. The data is in the form of a flat file which can be converted to corresponding JSON files after some amount of processing. After carefully extracting all of the data, it was filtered to suit the requirements of the proposed task.

For this task, there are two main aspects of interest - User Data and Behaviour Data. The User data is self explanatory in the sense that it contains information about the User. And, by

Behaviour Data, we mean the data associated with the restaurants that the user has previously visited and likewise. After the filtering, about 1363240 records were used to build the model. Apart from this, based on the restaurant visited, cuisines of interest for each user was determined from a pool of 110 cuisines.

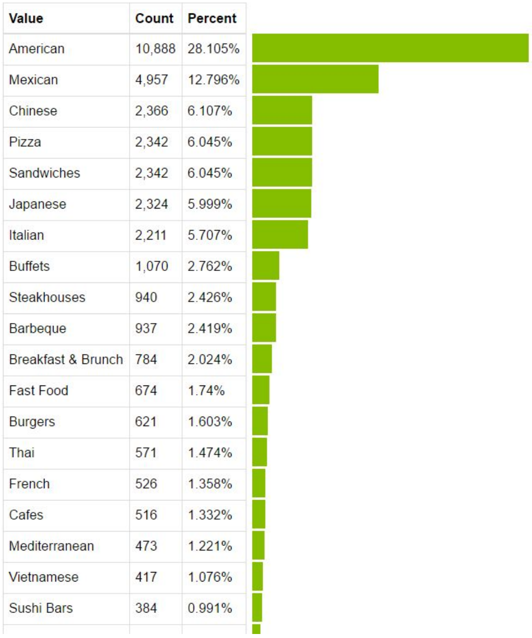
Data Preprocessing

Once the data was extracted and filtered out to suit the requirements of the research question, a lot of preprocessing needed to be done on this data. For instance, every time a user visited a restaurant, a date was recorded to denote the visit. However, more than the exact date, the week in which he/she visited a particular cuisine is of interested. So, the data associated with the dates and corresponding cuisine data is used to find out the “cuisine-of-the-week” for every user. A new column is added to the dataset called “Cuisine-Combo” as part of the feature engineering process to hold the same.

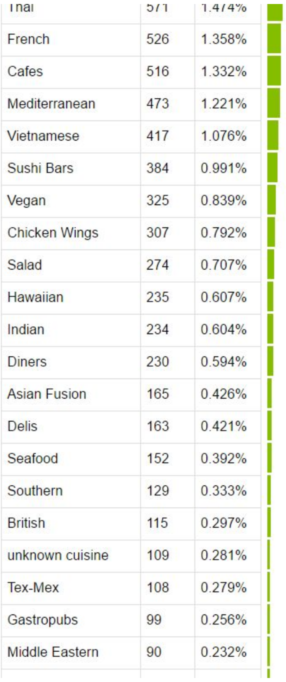
Based on this, the model is further developed to make predictions and provide recommendations. Finally, a subset of the data (read dataset with fewer features) is used to develop the model. In this filtered dataset, we have information associated with the User ID of every user, Business ID of the restaurant, “Cuisine-Combo” and some other features is used.

Data Modelling and Analysis

Before starting the data modelling process, it is essential to look at the distribution of the data. And, after some analysis, it was found that, American, Mexican and Chinese are the most frequently visited cuisines which is evident from the following results obtained.



Value	Count	Percent
American	10,888	28.105%
Mexican	4,957	12.796%
Chinese	2,366	6.107%
Pizza	2,342	6.045%
Sandwiches	2,342	6.045%
Japanese	2,324	5.999%
Italian	2,211	5.707%
Buffets	1,070	2.762%
Steakhouses	940	2.426%
Barbeque	937	2.419%
Breakfast & Brunch	784	2.024%
Fast Food	674	1.74%
Burgers	621	1.603%
Thai	571	1.474%
French	526	1.358%
Cafes	516	1.332%
Mediterranean	473	1.221%
Vietnamese	417	1.076%
Sushi Bars	384	0.991%



Thai	571	1.474%
French	526	1.358%
Cafes	516	1.332%
Mediterranean	473	1.221%
Vietnamese	417	1.076%
Sushi Bars	384	0.991%
Vegan	325	0.839%
Chicken Wings	307	0.792%
Salad	274	0.707%
Hawaiian	235	0.607%
Indian	234	0.604%
Diners	230	0.594%
Asian Fusion	165	0.426%
Delis	163	0.421%
Seafood	152	0.392%
Southern	129	0.333%
British	115	0.297%
unknown cuisine	109	0.281%
Tex-Mex	108	0.279%
Gastropubs	99	0.256%
Middle Eastern	90	0.232%

Key recommenders used for the prediction and recommendation tasks:

- Matrix Factorization Recommender
- Item Based Similarity Recommender
- Popularity Recommender

Matrix Factorization is one of the basic and very effective ways to develop recommender systems. A factorization recommender learns latent factors for each user and item and uses them to make predictions. Whereas, Item Based Similarity Recommender is a type of Collaborative Filtering approach which is based on the similarity between items calculated over the user's preferences with those items. Finally, Popularity model ranks every item based on overall popularity.

The above mentioned models are applied over the User ID and the Cuisine-Combo to make predictions. Initially, a smaller subset of the data was used to develop models for predictions and recommendations. However, it was extended to work with the entire dataset proving to give better results. The whole process is carried out multiple time with small tweaks in the parameters like Number of Factors, Type of Solver and Regularization value.

The type of Solver used plays a key role in getting more accurate values. The two types of Solvers used here are Alternate Least Squares (ALS) and Stochastic Gradient Descent (SGD). In SGD, we repeatedly pick some subset of the loss function to minimize one or more cells in the rating matrix and setting the parameters to better make just those 0. And, in ALS, we minimize the entire loss function at once, but, only twiddling half the parameters. That's because the optimization has an easy algebraic solution if half the parameters are fixed. So we fix half, recompute the other half, and repeat. There is no gradient in the optimization step since each optimization problem is convex and doesn't need an approximate approach.

Model-0: We first began by using user count and cuisine-combo(week_no + cuisine) as the input for our models. This turned out to be a cold start problem. The visualizations emphasize how badly the model performed. Matrix Factorization using SGD, ALS solvers with number of factors as 5 and 50 were tried. We decided to ditch the approach.

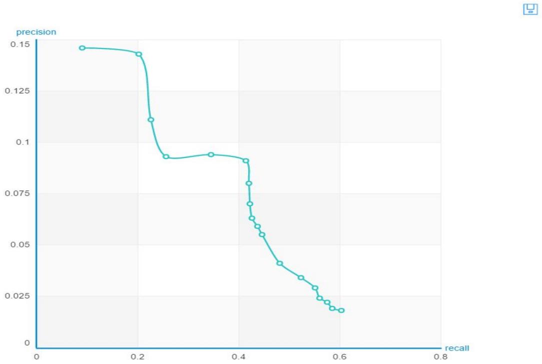
Model-1: We used just cuisine without week information and count of users to that cuisine as input to our models. We used the entire dataset and the algorithms Matrix Factorization (SGD,ALS @ 5,50 and 100) , Item Similarity and Popularity Recommender to recommend restaurants with certain cuisines to the user .

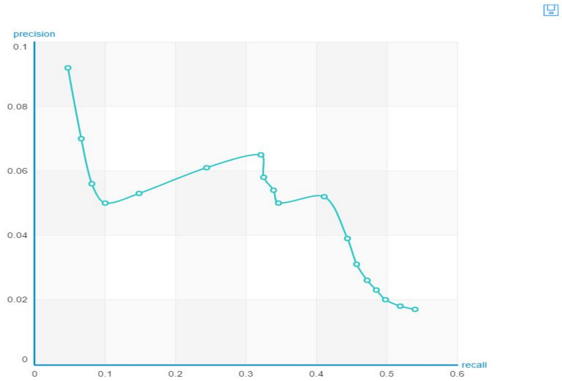
Model-2: Similar to Model -1 but we took a subset such that a user should visit a restaurant at least 3 or more times and then recommended those restaurants to the user.

Data Visualization

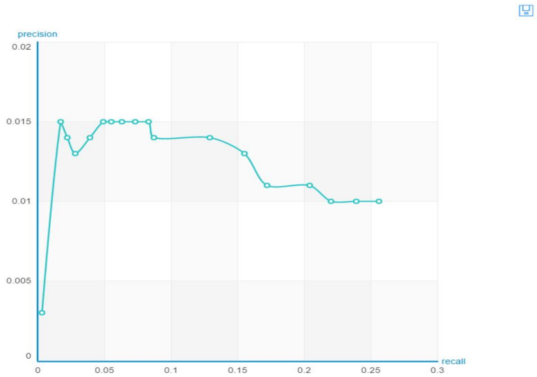
Model-0 (Considered Weekly Trends; Regularization : 1e-008)

SGD @ 5, 50



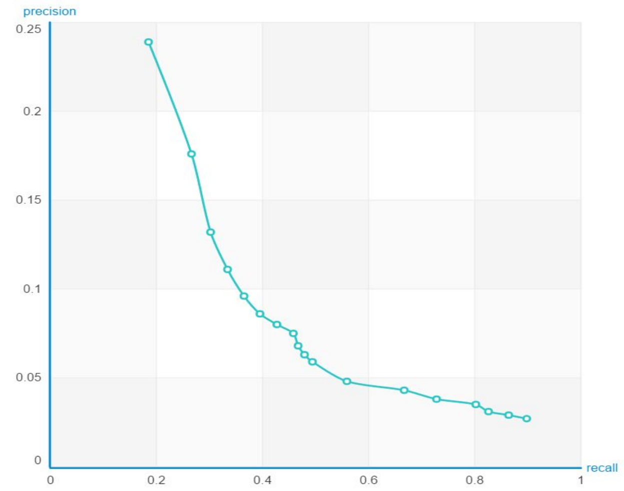


ALS @ 50

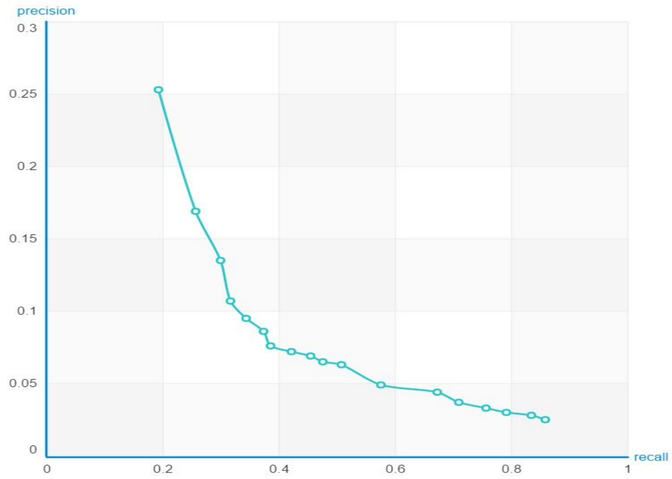


Model-1 (Complete data; Regularization: 1e-008)

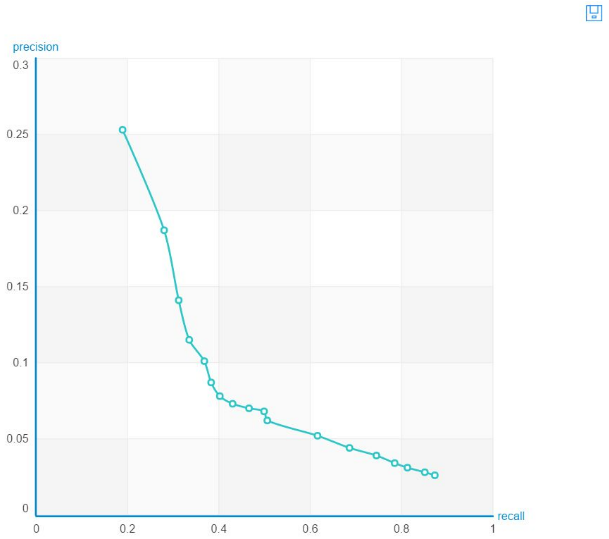
Solver - SGD
Number of Factors - 5



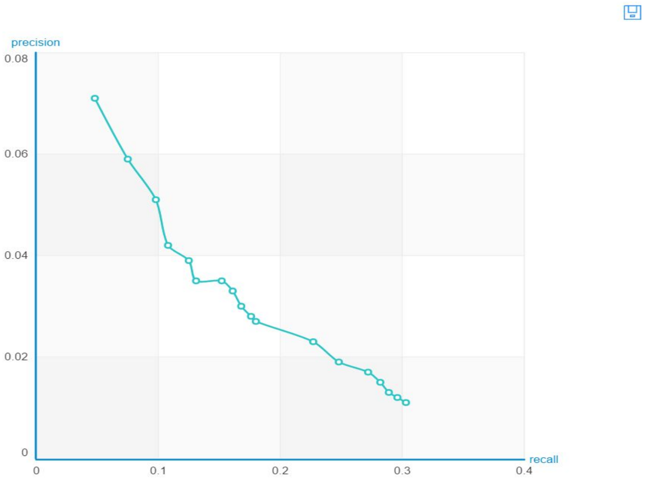
Solver - SGD
Number of Factors - 50



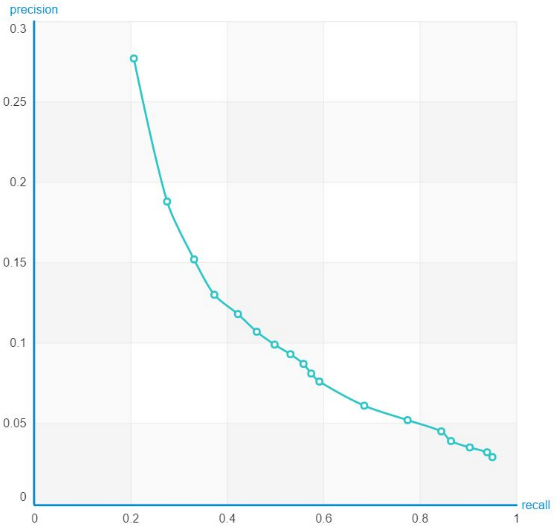
Solver - SGD
Number of Factors - 100



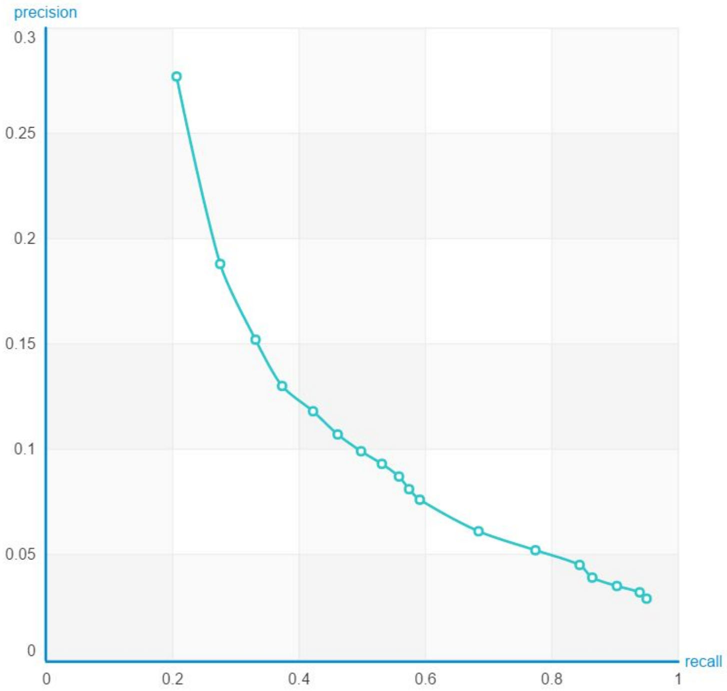
Solver - ALS
Number of Factors - 50



Popularity Recommender

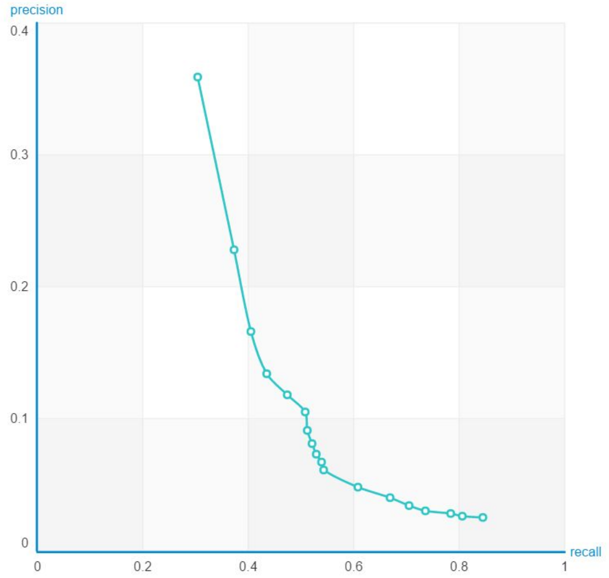


Item Similarity Recommender

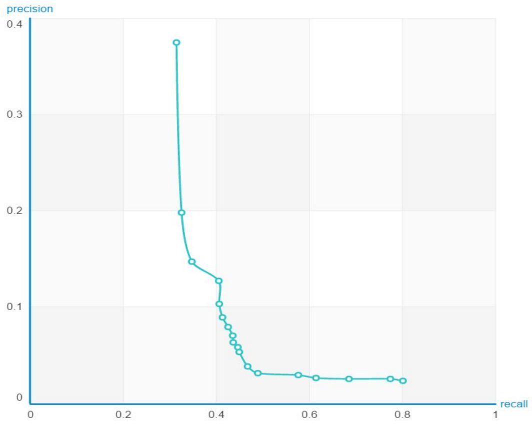


Model-2 (Filtered Data; Regularization: 1e-008)

Solver - SGD
Number of Factors - 5

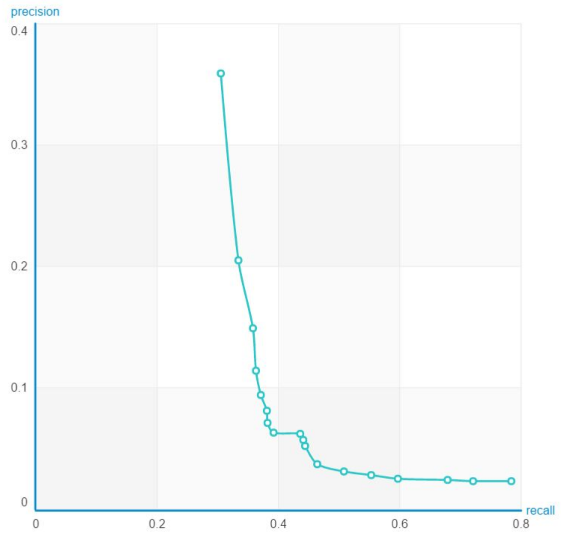


Solver - SGD
Number of Factors - 50



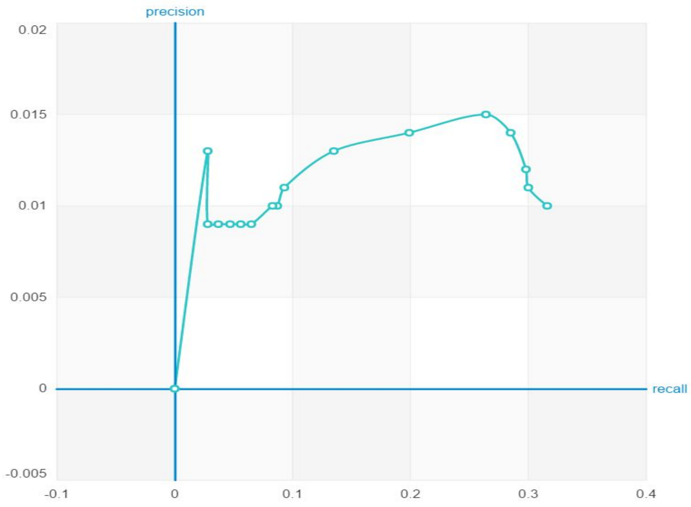
Solver - SGD

Number of Factors - 100

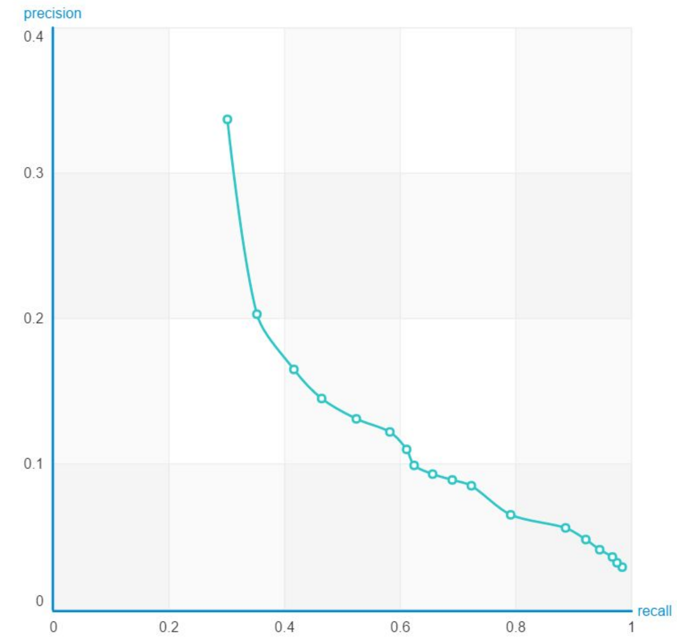


Solver - ALS

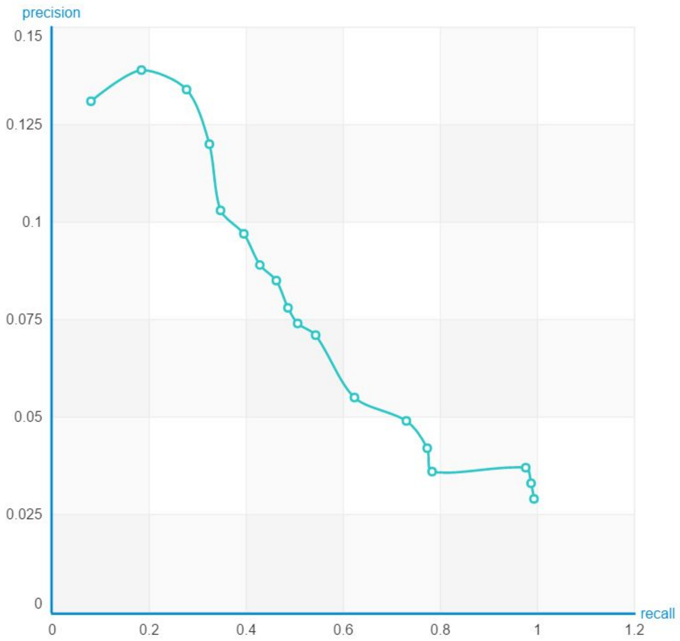
Number of Factors - 50



Popularity Recommender



Item Similarity Recommender



Results

Results obtained with all users in the entire dataset : 955734 rows

Recommender Metrics	Popularity	Item Similarit y (Jaccard)	Matrix Factorization (SGD @ 5)	Matrix Factorization (SGD @ 50)	Matrix Factorization (SGD @ 100)	Matrix Factorization (ALS @ 50)
RMSE	**1.75**	**2.21**	**2.28**	**1.14**	**0.44**	**2.12**
Precision @1	**0.27**	**0.14**	**0.24**	**0.25**	**0.005**	**0.07**
Recall @1	**0.20**	**0.09**	**0.18**	**0.19**	**0.002**	**0.04**
Precision @10	**0.08**	**0.07**	**0.05**	**0.06**	**0.01**	**0.02**
Recall @10	**0.57**	**0.49**	**0.44**	**0.47**	**0.12**	**0.17**

Results obtained with filtered users who visited at least 3 restaurants or more: 72452 rows

Recommender Metrics	Popularity	Item Similarit y (Jaccard)	Matrix Factorization (SGD @ 5)	Matrix Factorization (SGD @ 50)	Matrix Factorization (SGD @ 100)	Matrix Factorization (ALS @ 50)
RMSE	**3.26**	**5.77**	**3.97**	**2.96**	**2.87**	**4.98**
Precision @1	**0.33**	**0.13**	**0.35**	**0.35**	**0.35**	**0.0**
Recall @1	**0.30**	**0.07**	**0.30**	**0.30**	**0.30**	**0.0**
Precision @10	**0.08**	**0.07**	**0.06**	**0.05**	**0.05**	**0.01**
Recall @10	**0.68**	**0.50**	**0.53**	**0.44**	**0.44**	**0.08**

Conclusion

For task one, we have tried different models among which LSH forest with 30 trees and 500 candidates, 1 NN offers an f-score of 0.67 and has recorded as the best model among all the several models. The baseline highest f-score for this task was observed to be 0.59 obtained using Naïve Bayes model.

For task two, in which we concentrated on recommending the restaurant a user is most likely to visit in future by solely focusing on counting the number of times a user has visited a particular business:

· It is observed that Matrix Factorization using Gradient Descent Solver with 50 factors gives the best accuracy.

· If the frequency of the visit to the cuisine is greater that 3, it increases all the more.

· Using week number information along with the cuisine is a cold start problem as there is not enough information available to make accurate predictions.

· Using the top two or three ranked cuisines (cut-off @ 1,2,3) we can use locations information (latitude, longitude) of the user to predict all the business a person may visit.

Future Work

Task 2 can be extended further by including the user's location which is obtained by the latitude and longitude. Using the location information to find clusters of frequently visited places can help in accurately predicting which restaurant a user may next visit.

References

1. https://turi.com/learn/userguide/vis/visualization.html
2. https://www.yelp.com/dataset_challenge
3. https://github.com/ganesh91/Yelp-Dataset-Challenge

Team Members Contribution

Name	Contribution
Aditya Tanikanti	Project Idea, Task-2, Presentations, Documentation
Arun Sankaranarayanan	Project Idea, Task-1, Presentations, Documentation
Ganesh Nagarajan	Project Idea, Task-1, Presentations, Documentation
Keerthi Teja Nuthi	Project Idea, EDA, Presentations, Documentation
Sanjana Pukalay	Project Idea, Task-2, Presentations, Documentation