Yelp Recommendation System

DeadPool: Dajeong Jeon & Jonggoo Kang

Abstract

We utilize techniques and principles of recommendation systems to build a predictive model of how people would rate restaurants where they have not gone yet. We compare the performances of KNNWithMeans, SVD, and Hybrid Recommender. To evaluate the performances of the models, we use the Root Mean Square Error (RMSE), precision, and recall metrics.

1. Introduction

Yelp provides the restaurant lists and people can find the restaurant by certain conditions such as distances or prices. People can also read and write restaurant reviews, which contain stars (from 1-5) and a text review. It helps people to make their decision easily by reading individual stars and reviews. Moreover, this rating system can be used for the recommender system based on their rating histories. The recommender system would predict the ratings of the restaurants that have not been rated by the consumer. Then it would provide the restaurants lists that the consumer might like. This project aims to build and compare the recommender systems. It finds the best recommender system with the optimal parameters in the end.

2. Data

The Yelp dataset we use comes from Kaggle [1]. This data includes four separate json file such as reviews, businesses, users, and check-ins for the are in Pheonix, AZ. Each file has the size of 229,907, 11,537, 8,282, and 43,873, respectively. Since the reviews data has the largest, we decided to choose reviews for our dataset for the project. Regarding the details of the reviews, the data has 8 instances: business_id, date, review_id, starts, text, type, user_id, and vote.

We started to transform the json file to a data frame and save it as csv file. While transforming, we extracted user_id, business_id, and stars and renamed them to user, item, and rate, respectively. In

general, the data has 229,907 rows. There are 45,981 many unique users and 11,537 many unique restaurants. The rates are in the range of 1 to 5.

Although the data has no missing values, we see that there is a cold star problem. Some users only have a small number of reviews. We believe that the number of reviews less than 10 is not represented for determining the preferences as see in Figure (1). Therefore, we filter out the user with number of reviews less than 10, and then we left with 134,031 data.

3. Evaluation Methodology

We choose to evaluate our model with Root Mean Squared Error (RMSE), Precision, and Recall.

3-1. RMSE

We choose to evaluate the models using the RMSE with 5-fold cross validations to measure the accuracy. The significance of the RMSE is that it tends to disproportionately penalize large errors because of the squared term within the summation [2]. The equation for the RMSE is as follows [2]:

$$RMSE = \sqrt{\frac{\sum_{(u,j)\in E} e_{uj}^2}{|E|}}$$

where, $e_{uj} = r_{uj} - \tilde{r}_{uj}$

$E =$ a user-item index pair of the form (u,j) corresoponding to a position in the ratings matrix

3-2. Precision

Precision is called as positive predictive value and the fraction of relevant instances among the retrieved instances. It is relevant to the query:

$$Precision = \frac{True\ positive}{True\ positive + False\ positive}$$

3-3. Recall

Recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. It is also referred to as the true positive rate or sensitivity. The equation for recall is as follows:

$$Recall = \frac{True\ positive}{True\ positive + False\ negative}$$

4. Algorithms

in this analysis, we choose three algorithms to compare: KnnWithMeans, SVD, and a Hybrid of the two.

4-1. KNNWithMeans

K-Nearest Neighbors (KNN) algorithm is good to use figure out the problem with analogous preferences. Especially, we choose to use KNNWithMeans because it weights the rates with means. Thus, we can predict the ratings that people likely rate averagely. We utilize an item-based model to check the k nearest restaurants rated by users. Therefore, their ratings can be used to predict the ratings on the restaurants to users. The equation of KNNWithMeans is as follows [2]:

$$\tilde{r}_{uj} = u_i + \frac{\sum_{j \in N_u^k(i)} sim(i,j) * (r_{uj} - u_j)}{\sum_{j \in N_u^k(i)} sim(i,j)}$$

where, $\tilde{r}_{uj}$ = Predicted rating

u_i = Average rating by item *i*

$N_u^k(i)$ = Set of k nearest items which have given ratings by users

$sim(i,j)$ = Similarity score of *i* and *j*

We perform a grid search to find hyper parameters of *'k': list(range(100,220,20)), 'sim_options': {'name': ['cosine','pearson'], 'user_based': [False]}}*. Therefore, the result shows that the best parameters is *KNNWithMeans(k=200, sim_options={'name': 'cosine', 'user_based': False}).*

4-2. Singular Value Decomposition (SVD)

We choose to use Singular Value Decomposition algorithm by assuming that there are some factors that majorly contribute in determining ratings. That is, SVD transforms users and items to a shared latent space. Therefore, the relationship between the user and item is computed by the inner product of their vectors. The equation of SVD is as follows:

$$\tilde{r}_{uj} = \bar{u}_i * \bar{u}_i + \mu_i$$

keeps only the most significant factors to project the data to a lower dimensional space. To find the best parameters, we try to find them with the condition of *param_grid = {'n_factors':list(range(100,500,100)), 'n_epochs': list(range(10,50,10)), 'lr_all': [0.003,0.004,0.005,0.006,0.007], 'reg_all':[0.03,0.04,0.05,0.06,0.07]}.* Therefore, the result of the grid search is *SVD(n_factors=100, n_epochs=40, lr_all=0.003, reg_all=0.07)*

4-3. Hybrid Recommender

Hybrid recommender systems is designed to use the algorithmic power of various recommender systems to make robust inferences. For example, results from different algorithms are combined into a single unified score by computing the weighted aggregates of the scores from individual ensemble components. The methodology for weighting the components may use regression models. This system design can be formalized as follows:

$$Pred(u,i) = PredR(u,i) + (1-a) * PredS(u,i)$$

where, $R\ and\ S =$ Two Recommendation Modules

The hybrid model we use is to combination of KNNWithMeans and SVD. Therefore, *WeightedHybrid([algo1, algo2])*.

5. Results

The models were built with the data set of size 134,031. The RMSE was calculated by using RMSE with 5-fold cross validations. The results are shown as follows:

Algorithm	RMSE with 5-fold cross valudation	Precision	Recall
KNNWithMeans	1.0723	0.6971	0.6876
SVD	1.0209	0.6983	0.7199
Hybrid	1.1185	0.6805	0.8753

As seen in the graph above, SVD performs the best with RMSE score 1.0209 out of three algorithms. To validate with the best model, we looked at the RMSE, precision and recall values. The SVD model had the lowest residual mean squared error values compared to the other models according to Figure (2). It also had the highest precision (0.6983) and decent recall (0.71994) values as seen in Figure (3), and precision and recall scores are shown as graph in Figure (4), and Figure (5)

One of our major challenges are selecting the cutoff points to solve the cold star problem. Due to the cold star problem, we pare down the dataset by deleting ratings that have been rated less than 10 times. In this process, it is not clear to decide how many ratings we should filter since there is no standard cutoff point.

6. Conclusion

We used the Yelp rating data to build the recommender systems and compare three different algorithms, which were KNNWithMeans, SVD, and Hybrid. Grid Search was used to optimize the best parameters for each algorithm. With the optimal parameters, three algorithms were compared on RMSE, Precision and Recall. In conclusion, SVD had the best results; it showed higher performance measures on RMSE and precision. Therefore, we concluded that SVD is the best algorithm for this Yelp rating data.

There are some limitations working on this project. One of the limitation is the magnitude of the dataset; it would be nice to be able to use all files that we have so we might get a better output. When we compare three algorithms, SVD shows the best result regardless of the size of the data and the different parameters. In the future, we could try different cutoff points that would be larger than 10. Additionally, we could use different algorithms for Hybrid recommender, and compare the result if we might have a different output.

REFERENCES

[1] RecSys Challenge 2013: Yelp business rating prediction. https://www.kaggle.com/c/yelp-recsys-2013/data. Accessed: 2018-05-01.

[2] Aggarwal, Charu C. *Recommender Systems: The Textbook*. Springer, 2016.

APPENDIX

Figure (1)

```
user['rate'].value_counts()
```

```
1       22829
2        7432
3        3813
4        2443
5        1712
6        1183
7         936
8         668
9         572
10        427
11        368
12        339
13        280
14        217
16        185
15        178
18        160
17        136
20        116
19        115
22        111
21        105
24         82
23         78
27         60
26         59
28         57
25         54
30         47
32         45
      ...
```

Figure(2)

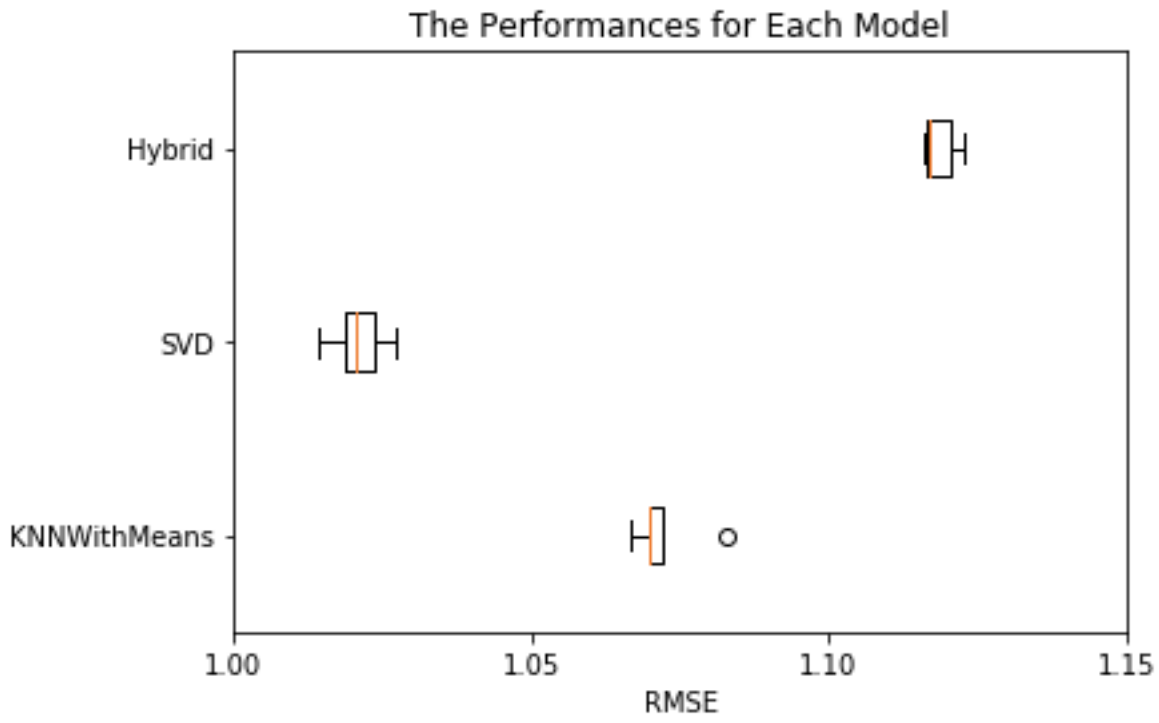

Figure (3)

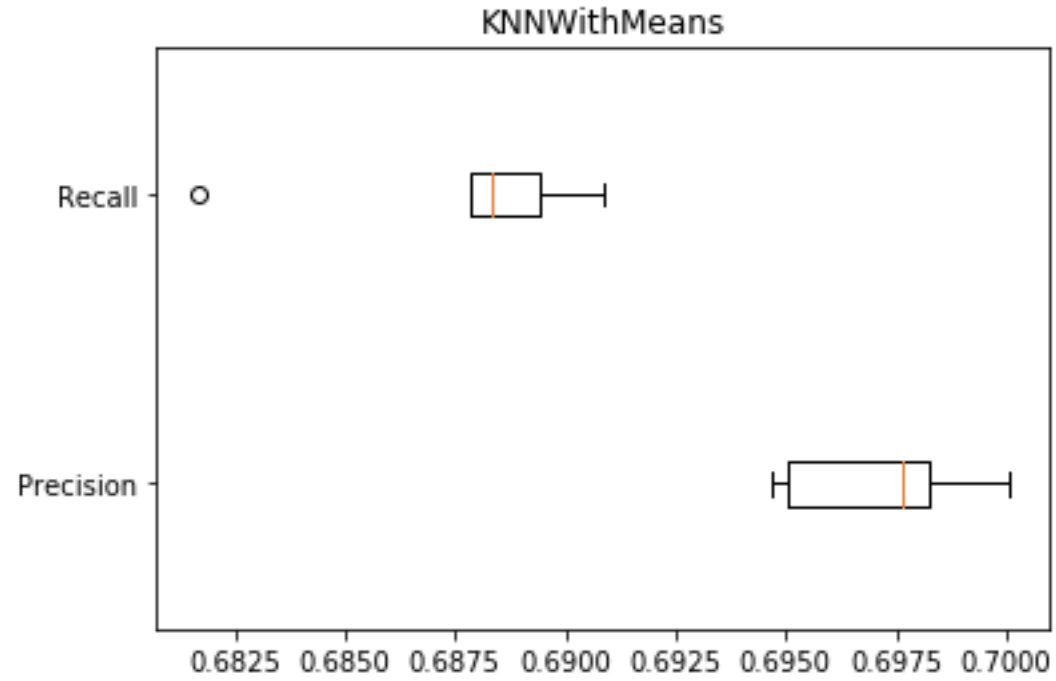

Figure (4)

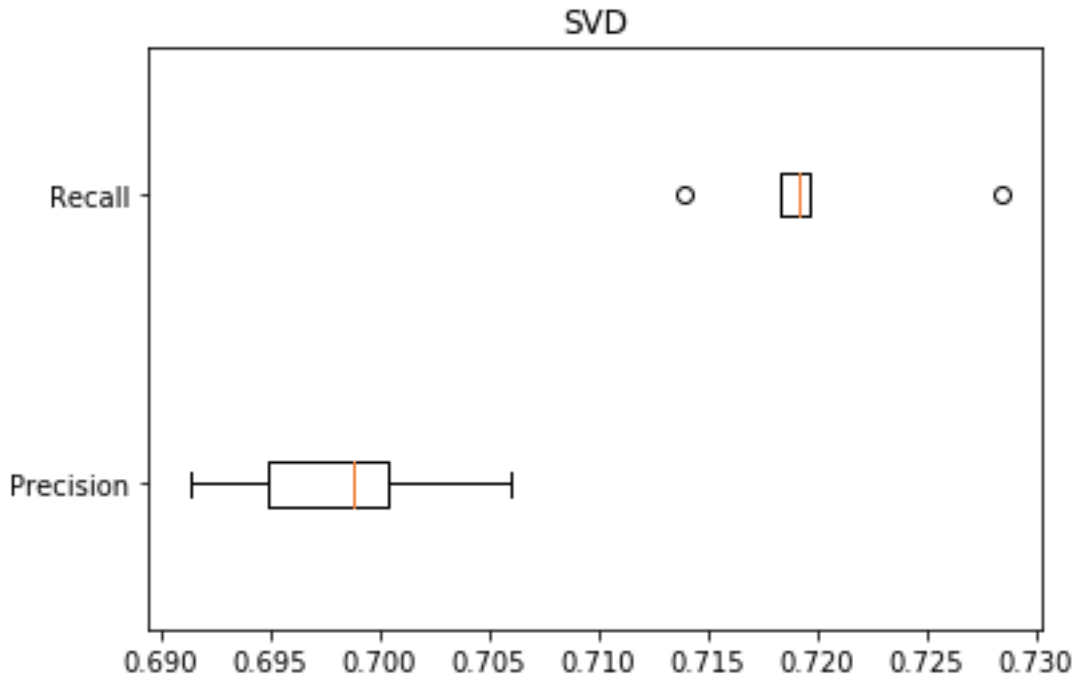

Figure (5)