Matrix Factorization and Regression Models for Predicting User Ratings of California Businesses

Abstract
In this paper, we investigate the performance of various models on predicting user ratings given to businesses in California. We compare matrix factorization and standard regression models. After fine-tuning the models and evaluating the prediction error, we identify the best performing models for our dataset.

Introduction
Nowadays, an individual's preference is heavily exploited by companies and internet services, making recommendation systems a crucial area of study. To create a successful recommendation model, a large amount of review data are collected for model training. These data can be categorized into two types, implicit and explicit data. Both types of data have value and have been studied to build better recommendation models. Our interest is to use explicit user feedback on California businesses, explicit metadata regarding the businesses, and implicit review data. We use the Google Local dataset to find the best predictive model that characterizes and predicts a user's rating for a business.

Related Works
Probabilistic Matrix Factorization
This model was invented by Salakhutdinov and Mnih in 2006. That year, Netflix posed a prediction challenge for user ratings of movies. Netflix provided a movie dataset with 100 million ratings out of 8.5 billion entries. Simon Funk used the Probabilistic Matrix Factorization model in his singular value decomposition approach and won 3rd place. It had the advantage of scaling linearly with the number of observations and generating accurate predictions on a large and imbalanced dataset with high sparsity. In Salakhutdinov and Mnih's paper [1], they showed that by linearly combining the predictions of multiple Probabilistic Matrix Factorization models with the ones of Restricted Boltzmann Machine models, they were able to get an error rate of 0.8861. This is 7% better than Netflix's own system.

TransRec
In 2017, He, Kang, and McAuley developed TransRec [2]. It aimed to model the third-order interactions between a user's observed items and the user's preferred future item for large-scale sequential prediction. They modeled users as translation vectors acting on a sequence of items. TransRec was compared with these baseline models: PopRec, Bayesian Personalized Ranking (BPR-MF), Factorized Markov Chain (FMC), Factorized Personalized Markov Chain (FPMC), Personalized Ranking Metric Embedding (PRME), and Hierarchical Representation Model (HRM). TransRec outperforms the baseline models in all cases because it worked effectively on a dataset with high sparsity. It improved the best baseline prediction on the Google Local dataset by 0.6%.

TransFM
This model was proposed by Pasricha and McAuley in 2018 [3]. It combined translation and metric-based approaches for sequential recommendation with Factorization Machines. They used squared Euclidean distance to measure how strong features are connected to each other. It has the advantage of Factorization Machines in terms of linear computing time, making it easy to work on large-scale sequential

recommendation datasets. Plus, it possessed the flexibility of incorporating additional content information. It was intended to be a general-purpose model which could outperform other models with minimal feature engineering. They tested their model on businesses from six U.S. states from the Google Local dataset (in total 11,453,845 reviews, 4,567,431 users, and 3,116,785 local businesses). Their results showed that TransFM without incorporating any features did not have a better prediction accuracy in terms of Area Under Curve (AUC) than all the baseline models (PopRec, BPR-MF, FMC, FPMC, PRME, HRM, TransRec, and CatCos). By incorporating features, TransFM had the highest AUC (mean: 0.7986), which was 2.6% better than the best baseline. The authors claim this is because the translation component of TransFM effectively models both content and collaborative feature interactions.

Motivation

In this paper, we use the same Google Local dataset as previously mentioned. By reading these related works, we know that many successful and popular Collaborative Filtering recommendation systems are based on Matrix Factorization. By factoring an interaction matrix into two low rank matrices, Matrix Factorization learns the latent factors and infers unobserved user preferences of the item efficiently on a large dataset. Due to this computing efficiency, we choose to use this same approach for our task. It will reduce training time for a high-dimensional dataset like Google Local.

While Matrix Factorization has advantages, it is limited in the case of dealing with implicit feedback. It is very common that explicit feedback like user ratings are missing for a large dataset like Google Local. In this case, Matrix Factorization does not work effectively. Instead, Factorization Machines and models based on this idea are widely used to account for this implicit feedback limitation by incorporating arbitrary contextual auxiliary features with user and item interactions. We see Factorization Machines are not only used as a technique to build baseline models in the above related work, but also popular in industry. For instance, LightFM, a python library for Factorization Machines, maintained by an e-commerce company Lyst, has proven its effectiveness in recommendation systems. Its documentation page shows the prediction performance of Factorization Machine models with Weighted Approximate-Rank Wise (WARP) loss and Bayesian Personalized Ranking (BPR) loss over the MovieLens data is quite significant (AUC score for WARP: above 0.9; AUC score for BPR: above 0.85). We assume Factorization Machines should be a good choice for our task.

Dataset

Our data consists of business reviews from Google Local (i.e. Google Maps). One set contains data about the businesses such as name, price, and address, while the other set contains data about the reviews such as rating, reviewer name, and review time. Both datasets were collected as research for Julian McAuley's lab [2, 3]. Due to constraints of computing power and time, we limited our data to only reviews from businesses in California. This resulted in about 134K observations; We then further filtered our data to only keep reviews from users that had rated at least five businesses, resulting in about 62K observations. The purpose of this was to focus on creating a model that would benefit active users.

There are 4460 users in our dataset where the average number of ratings given per user is about 14. The distribution of this value is shown in Figure 1. Although the median number of ratings is 8, we see that there are a significant number of outliers that reach until the 500s. On the other hand, we have 36,444 businesses corresponding to these reviews. The average number of ratings given per business is 1.71, and

the distribution can be seen in Figure 2. As with the user ratings, the median per business is quite low at one but outliers reach until the 50s. As mentioned before, we limited our scope to reviews from California businesses and the locations can be seen in Figure 3.

Figure 1:

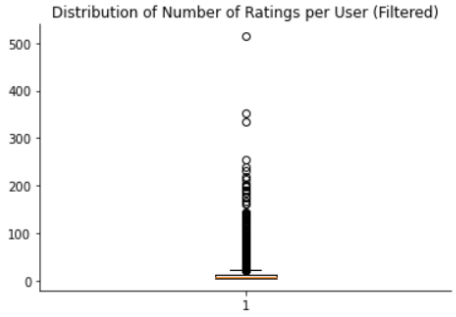


Figure 2:

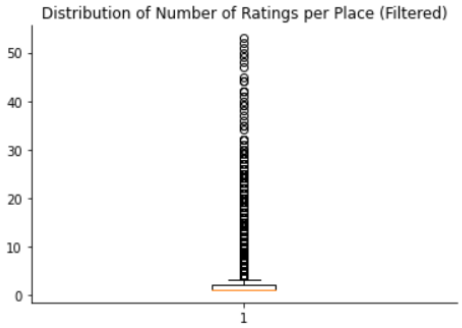
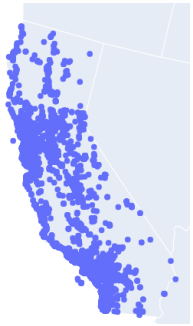
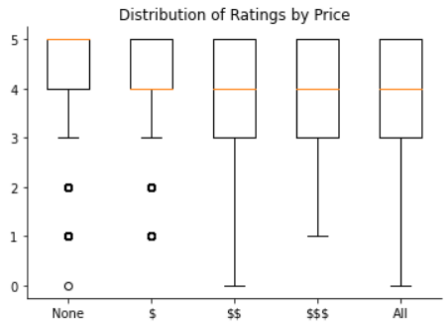


Figure 3:



Each review is assigned to at least one category defining the type of business. The top three categories are restaurant, Asian restaurant, and American restaurant. The bottom three categories are neonatal physician, abrasives supplier, and liquors-retail. Furthermore, each business is assigned a price marking. Looking at the distribution of ratings by price [Figure 4], it is apparent that while the median remains at around four across price markings (except for cases with no price marking), we see that the bottom half of ratings are much lower for more expensive businesses.

Figure 4:



Lastly, we look at how ratings shift with time. Overall, we see that the average rating starts high in 1990, increases from 2002 to 2007, sharply decreases in 2008, and then starts to rise again in the following years [Figure 5]. On a monthly scale, we observe that summer months tend to have higher ratings than the rest of the year, and the months January, April, and November have particularly lower ratings on average [Figure 6]. Finally, on an hourly scale, we see a huge spike in average rating at noon and a dip shortly after at 2PM, while the rest of the day the ratings hover at around the same range.

Figure 5:

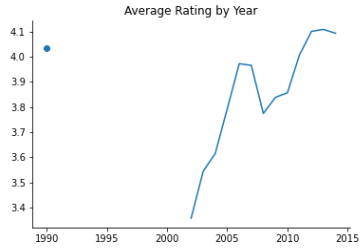


Figure 6:

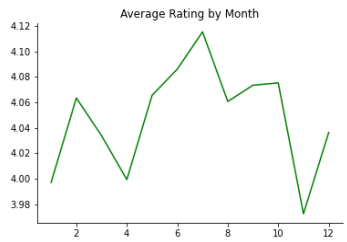
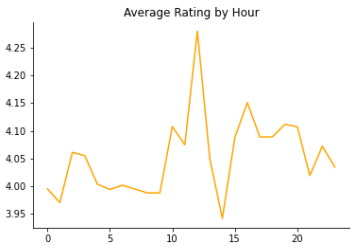


Figure 7:



To comply with the previous articles studying this dataset, we split the data in the same manner. The test set contains every user's last review, the validation set contains every user's second to last review, and the train set contains the rest of the reviews, containing 4460, 4460, and 53,418 observations respectively.

Models

The motivation for picking various models is to identify the prediction method that best fits our dataset's characteristics. The model must also be general enough to not overfit our predictive task. We use a Mean Value predictor as a baseline. We start with recommender system models including Latent Factor which is commonly used for predicting ratings as it is simple to understand and requires little training. We also attempt FastFM and Probabilistic FM (PFM) as they are both extensions to the simple Latent Factor model. For instance, with FastFM we can incorporate not only user-business interactions, but also features such as review time and business price. Next, we move on to standard regression models including Linear regression, Random Forest, KNN, and SVR.

We evaluate the models' performance with mean square error (MSE). For the recommender system models, we also accounted for the regularization term in our loss function, which helps us to pick parameters with low MSE and model complexity, thus, reducing the probability of overfitting the model. We utilize grid search with our training and validation data on all the models to identify the best parameters and evaluate the final model with our test data.

We use only user and business IDs as input data for the following models: Latent Factor, PFM, and FastFM. We use user and business IDs, business category, year, month, hour, price, user mean rating, and place mean rating for FastFM with features. Finally, we use year, month, hour, price, user mean rating, and place mean rating for the following models: Linear Regression, Random Forest, KNN, and SVR.

Mean Value Predictor:

To establish a baseline for our models, we used a simple Mean Value predictor. It predicts review ratings as the average of all the ratings of the dataset. Such a technique is a reasonable approach when facing uncertainties of having new user/business entries, it is often used in other models like Latent Factor Model. By using the Mean Value predictor, we received a baseline MSE of 1.279, which is surprisingly low considering the rating ranges from 0 to 5.

Latent Factor Model (Bias only):

The Latent Factor Model utilizes matrix factorization to find weighted low-rank approximations. We assume that there exists a low dimensional representation of users and businesses, and their relation to rating can be predicted using the matrix. The model also introduces the regularization term lambda to lower complexity which reduces overfitting. The equation is shown in Eq 1, where α is the global average, βu is how much the user tends to rate above the mean, and βb is how much this business tends to receive higher ratings than others. After fine-tuning some of the parameters, we reached an MSE of 1.129 on the Latent Factor Model.

$$\text{Eq 1. } f(u, b) = \alpha + \beta u + \beta b$$

Complete Latent Factor Model:

Just like the Latent Factor Model with bias only, the complete Latent Factor Model contains additional terms γu and γb, shown in Eq 2. γu describes the user's preferences using a low-dimensional vector, γb describes the business' properties using a low-dimensional vector. Training with the complete Latent Factor Model shows MSE lower than 0.9, whereas testing the complete Latent Factor Model with our test dataset produces an MSE of 1.117, this indicates some level of overfitting, which may expose a weakness in our model.

Eq 2. $f(u, b) = \alpha + \beta u + \beta b + \gamma u * \gamma b$

Probabilistic FM:
Probabilistic FM estimates the best factorization for the rating matrix. The idea is that a user's preferences are modeled by linearly combining other feature vectors using user-specific coefficients. Training PFM amounts to finding the best rank approximation to the rating matrix. We set the model learning rate to 0.01, the number of epochs to 10, and the number of factors to 20. Running the model gives us an MSE of 7.339.

FastFM:
Like the idea of the latent factor model, factorization machines can express many different latent factor models and are often used for collaborative filtering tasks. We use the FastFM library's ALS regression solver to predict user ratings. When predicting with only user and business IDs in our sparse matrix, we get a prediction MSE of 2.037. With additional features (business categories, year, month, hour, and price) included in the sparse matrix, we get a prediction MSE of 1.950.

Linear Regression:
The linear regression model describes the relationship between a dependent variable and one or more independent variables. A simple model was introduced to make preliminary prediction tasks, yet it produces surprisingly good outcomes. Fitting the model with our dataset generates function coefficients which are used to make predictions. The resulting MSE is 0.233 on training data, and 0.227 on test data.

Random Forest:
Random forest model ensembles multiple decision trees into a final decision. It can be used in regression or classification tasks, which in our case, we use as a regression task. We build 100 decision tree estimators with a depth of 10 to find the best split point candidates and the validation MSE is 0.203. A grid search is done to find the optimal estimator count and depth, resulting in a lower test MSE of 0.200.

KNN
The KNN model is popularly used in classification problems, but here we use it as a regression model. We choose the neighbor size to be 5 and let the regressor map feature data to high dimensional space. The predicted rating MSE yields 0.278 at first, but with grid search optimization on the number of neighbors, it lowers down to 0.262 on the test set.

SVR
We also use the Support Vector Regression model which uses the principle of SVM for regression problems. The SVR finds a hyperplane in multidimensional feature space that has the maximum number of data points within the decision boundary line. We test our training dataset with the linear kernel and

radial basis function kernel to compare their performance. With respective validation MSEs of 0.241 and 0.220 for the two kernels, we use the latter kernel to evaluate on test data which gives us an MSE of 0.216.

Evaluation and Results

All the models' results can be seen in Table 1. Out of the recommender system models, the complete latent factor performed best. However, all of the standard regression models significantly outperformed any of the recommender system models, with Random Forest performing the best overall. During our feature engineering, we attempted to use various combinations of features. We discovered that there were too many categories (over 2000) for it to be a good predictor, so we left that out. With the features year, month, and user mean rating, we were bottlenecking at an MSE around 0.8. After adding hour, price, and place mean rating, our MSE dropped down to around 0.2. The significance in adding these features agrees with our initial data exploration, as we saw huge fluctuations when plotting rating versus hour and price indicating some sort of correlation.

Table 1:

	Mean Value	**Latent Factor (Bias only)**	**Complete Latent Factor**	**PFM**	**FastFM**	**Linear Reg.**	**Random Forest**	**KNN**	**SVR**
MSE	1.279	1.129	1.117	7.339	1.950	0.227	0.200	0.262	0.216

Conclusion

From the models we tested, standard regression models perform the best in predicting a user's rating on California businesses with the Google Local dataset. In particular, Random Forest achieves the lowest MSE of 0.200, whereas matrix factorization based recommendation models show little to none improvement over the baseline model.

Future work could use a greater portion of the dataset, not limiting to California. Moreover, location data was not incorporated into the features of any of our models due to data quality and time constraints. We predict that this could be an important feature, especially when applying to a larger scale of locations.

References

[1] Salakhutdinov, Ruslan, and Andriy Mnih. *Probabilistic Matrix Factorization* . NIPS Proceedings, 2008, https://papers.nips.cc/paper/2007/file/d7322ed717dedf1eb4e6e52a37ea7bcd-Paper.pdf.

[2] He, Ruining, et al. "Translation-Based Recommendation." *ArXiv.org*, 8 July 2017, https://arxiv.org/abs/1707.02410.

[3] Pasricha, Rajiv, and Julian McAuley. "Translation-Based Factorization Machines for Sequential Recommendation." *ACM Conferences*, 1 Sept. 2018, https://dl.acm.org/doi/10.1145/3240323.3240356.