Cooking Up Recommendations: A Latent Factor Model for Predicting Recipe Ratings

Jack Weston
University of California, San Diego
Jweston@ucsd.edu

Chase Peterson
University of California, San Diego
cepeters@ucsd.edu

Cole Carter
University of California, San Diego
Cscarter@ucsd.edu

Nathaniel Greenberg
University of California, San Diego
ngreenberg@ucsd.edu

Abstract

In the past few years, the culinary domain has experienced significant growth in user-generated content, including recipe-sharing platforms. Accurately predicting recipe ratings can enhance user experience by providing better recommendations and improving platform engagement. This paper explores a latent factor model tailored for recipe datasets to predict ratings effectively. Experimental results demonstrate the model's capacity to outperform baseline approaches, highlighting its potential for practical applications in the food-tech industry.

Introduction

Online recipe-sharing platforms have revolutionized how people discover, share, and rate recipes. These platforms rely heavily on user-generated content, including ratings and reviews, which serve as critical ways to determine recipe quality and relevance.

Traditional recommendation systems, such as collaborative filtering, have shown promise in domains like e-commerce and streaming services but often fail to capture the nuanced preferences unique to culinary applications.

This paper introduces a latent factor model designed to predict recipe ratings by integrating user ratings of recipes. Our model aims to bridge the gap between recommendation systems and the needs of recipe-sharing platforms. Through evaluation on a publicly available dataset, we demonstrate the effectiveness of our approach in capturing user preferences and improving rating prediction accuracy.

Part 1: EDA

	Dataset	Column Name	Data Type	Unique Values	Missing Values
0	PP_recipes	id	int64	178265	0
1	PP_recipes	i	int64	178265	0
2	PP_recipes	name_tokens	object	176694	0
3	PP_recipes	ingredient_tokens	object	177699	0
4	PP_recipes	steps_tokens	object	178091	0
5	PP_recipes	techniques	object	41760	0
6	PP_recipes	calorie_level	int64	3	0
7	PP_recipes	ingredient_ids	object	177524	0
8	PP_users	u	int64	25076	0
9	PP_users	techniques	object	24609	0
10	PP_users	items	object	25066	0
11	PP_users	n_items	int64	576	0
12	PP_users	ratings	object	10754	0
13	PP_users	n_ratings	int64	576	0
14	RAW_interactions	user_id	int64	226570	0
15	RAW_interactions	recipe_id	int64	231637	0
16	RAW_interactions	date	object	6396	0
17	RAW_interactions	rating	int64	6	0
18	RAW_interactions	review	object	1125282	169
19	RAW_recipes	name	object	230185	1
20	RAW_recipes	id	int64	231637	0
21	RAW_recipes	minutes	int64	888	0
22	RAW_recipes	contributor_id	int64	27926	0
23	RAW_recipes	submitted	object	5090	0
24	RAW_recipes	tags	object	209115	0
25	RAW_recipes	nutrition	object	229318	0
26	RAW_recipes	n_steps	int64	94	0
27	RAW_recipes	steps	object	231074	0
28	RAW_recipes	description	object	222668	4979
29	RAW_recipes	ingredients	object	230475	0
30	RAW_recipes	n_ingredients	int64	41	0

Overview:

Unique Recipes, Users, and Reviews:

There are 178,265 unique recipes with information like ingredients, nutrition, and cooking time. There are 25,076 unique users, each having their own user data from Food.com. There are 2,316,370 reviews totaling 18 years, including user ratings and textual feedback.

Initial Observations:

Time Range of Reviews:

The review dataset spans several years, showing a steady growth of reviews over time. The earliest review was posted in 2011, while the most recent review appeared in 2024.

Average and Median Cooking Times:

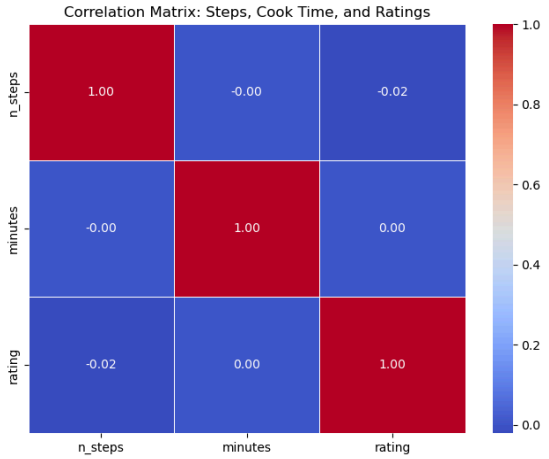
The average cooking time across recipes is around 45 minutes, while the median is 30 minutes.

EDA

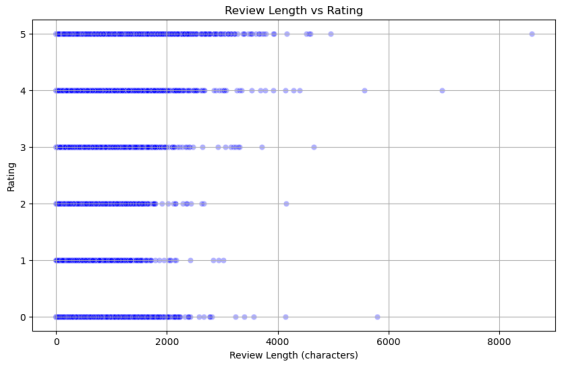
Distribution of Ratings:

The ratings distribution shows that the majority of recipes receive 3-5 stars, but there is a significant proportion of 0-2 star ratings.

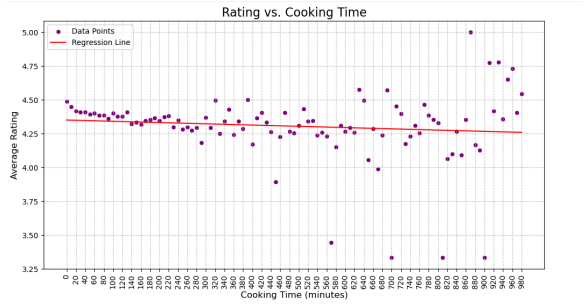
We used word clouds to look at user reviews from RAW_interactions.csv filtered by ratings. This gave us insights into the sentiment expressed by users in their reviews. For higher ratings (4-5 stars), words like "thank," "good," "added," and "used" were most commonly used, reflecting positive feedback and satisfaction. On the other hand, lower ratings (0-2 stars) featured words such as "followed," "sorry," "really," and "way," which suggest dissatisfaction or frustration with the recipe.



We then investigated the relationships between the number of steps, cooking time, and ratings using a correlation matrix. The results showed that there is almost no correlation between the number of steps and ratings (correlation value of -0.02). Similarly, the correlation between cooking time (in minutes) and ratings is essentially 0.00, indicating that these features do not significantly influence user ratings on their own.



We further examined the relationship between review length and ratings. We found that longer reviews tend to correspond with higher ratings, particularly for 4-5 star recipes. The length of reviews increases as ratings rise, with reviews for 5-star recipes sometimes reaching up to 3,000 characters. For lower ratings (0-2 stars), reviews are typically shorter, but we do see a small increase in length for 0-star reviews, where the character count ranges between 1,500 to 2,500 characters.



Finally, we looked at the relationship between cooking time and ratings. We found a somewhat negative correlation between cooking time and ratings for recipes that take less than 2 hours to prepare. The linear regression is not steep enough though for a clear relationship to be determined. We do see however as cooking time increases, the number of ratings tends to decrease.

Part 2: Predictive Task

Predicting User Rating Based on Recipe Features

Our predictive task was predicting the user rating from a scale of 0 to 5 stars for any recipe. We used recipe features such as ingredients, cooking time, and nutritional information. This aligns with our goal of developing a recommendation system that evaluates user satisfaction with recipes

Predictive Task Evaluation:

We considered using a few metrics to evaluate the model such as Mean Absolute Error (MAE), Mean Square Error (MSE), and Classification Accuracy.

However, to evaluate the model's performance we ended up using the Mean Squared Error as our primary metric, because it effectively measures the average magnitude of errors and penalizes large deviations more heavily, ensuring the model avoids significant prediction inaccuracies. Unlike metrics like Mean Absolute Error (MAE), MSE is more sensitive to outliers, which is important for accurately predicting continuous ratings.

Baselines for Comparison:

We used the following baselines for comparison.

- Global Average Baseline: Predicts the global average rating for all recipes.
- User Average Baseline: Predicts the average rating of a user
- Recipe Average Baseline: Predicts the average rating of a recipe
- Combined Average Baseline: Chooses between user or recipe averages based on the size of data samples, defaulting to

the global average when neither is available.

Each of these baselines provided a progressive approach to predict ratings. We used all of these as a framework to evaluate the latent factor model. We considered predicting the average rating as a strong baseline for our model because it provided a simple benchmark. If the model couldn't outperform this baseline, it would suggest that the features and parameters we used were not effective.

Model Implementation:

To prepare the dataset for our predictive task, we performed several preprocessing steps. We extracted user_id, recipe_id, and rating then split the data into training and testing sets, with 90% used for training the model and 10% for testing its performance. To make the data work with the model we created mapping to convert user_id and recipe_id into numerical indices. Additionally, we made a few dictionaries containing helpful information such as the list of recipes rated by each user, the users who rated each recipe, and the ratings themselves.

In terms of features, the model directly uses user and recipe IDs, along with the averages calculated from the data. This includes the global average rating, the average rating for each user, and the average rating for each recipe. On top of these features, the model also learns latent factor embeddings which is referred to as γ_U for users and γ_R for recipes. These embeddings show the relationship of how users tend to interact with certain types of recipes.

Validity of Model Predictions:

To validate the model's predictions, the Mean Squared Error (MSE) on the test set was computed. This metric would tell us how close the model's predictions are to the actual ratings. Then we will compare the latent factor model's MSE against the several baseline methods we had which were the global average, user averages, and recipe averages. Throughout the process, iterative optimization techniques, including stochastic sampling, were used to refine the model's parameters and improve its overall performance.

Part 3: Model

The Latent Factor Model (LFM) was chosen due to its ability to capture complex interactions between users and recipes through latent features. It predicts ratings using the formula:

$$Prediction = \alpha + \beta_U[u] + \beta_R[r] + \gamma_U[u]\gamma_R[r]$$

α: $Global\ average\ rating$

β_U, β_R: $User\ and\ recipe\ biases$

γ_U, γ_R: $Latent\ factors\ for\ users\ and\ recipes$

We optimized the model using the Adam optimizer with a learning rate of 0.1. A regularization term (weighted by λ=0.00001) prevents overfitting by penalizing large parameter values.

During the development and training of our model, we encountered several issues. One of the key issues was scalability. With the dataset containing over 700,000 reviews and interactions, managing memory efficiently was critical. The large size of the data made it impractical to load everything into memory at once and process the entire dataset. To address this we employed sampling techniques, randomly selecting smaller subsets of the data for training at each iteration. This approach still allowed the model to learn effectively from the full dataset over multiple iterations.

Another significant challenge we had was overfitting. In our case, many users and recipes

had limited interaction data, making it easy for the model to memorize specific patterns rather than generalizing to unseen data. To mitigate this we applied regularization, adding penalty terms to the loss function to discourage overly complex parameter values. This really helped our model strike a balance between fitting the training data and maintaining generalizability on the test set.

Lastly, the biggest hurdle we encountered was convergence. Training the model with latent factors required fine-tuning the parameters over many iterations. The way we solved this was by using iterative updates over 100 training epochs. This allowed the model to refine its parameters step by step, without overshooting optimal values. Although it took longer to run it was critical in ensuring that the model could handle the large-scale data while maintaining robust predictive performance.

The other models we considered were basic collaborative filtering approaches such as using only user and recipe averages. As well as simple linear regression models that did not leverage latent features.

The unsuccessful attempts reveal that the simpler models failed to capture nuanced user-recipe interactions. Our LFM outperformed these baselines with significant reduction in MSE. The strengths of our model is it handles large datasets effectively and learns hidden user and recipe attributes that improves the predictions. However, a weakness is the LFM requires careful tuning of latent factors and their regularization parameter. After retrieving the results of our model, we learned that we overfitted as our Test MSE was higher than our training MSE. This was a result of overtuning our hyperparameters to the training dataset. We tried to use a higher level approach and test all of the hyperparameters using grid search.

We ran the model for only 10 iterations, but with more time, it would have been better to complete the full 50 iterations we originally planned. It would also have been interesting to experiment further with different values for the learning rate and lambda. Due to resource constraints, we were only able to explore a limited range of parameters, and additional tuning could have improved the model's performance on the test set.

Running just 10 iterations was a limitation, as we began to encounter memory issues on the cluster where the Latent Factor Model (LFM) was implemented. This process highlighted the importance of carefully managing resources and avoiding overfitting during techniques like grid search, ensuring that models remain generalizable to unseen data.

Part 4: Related Literature

Dataset

We used a dataset of over 700,000 user reviews of recipes, where each review includes a rating (ranging from 0 to 5). The dataset was used to train the LFM, which predicts ratings for recipes.

Related Work

"Using Tags and Latent Factors in a Food Recommender System," implements the use of matrix factorization techniques for predicting user preferences. The paper extends traditional matrix factorization by incorporating tagging information, allowing the model to capture nuances about user preferences and recipe attributes. Tags, such as "spicy" or "gluten-free," help refine recommendations by modeling user-specific and recipe-specific latent factors.

State-of-the-Art Methods

Matrix factorization is a widely used approach, especially with stochastic gradient descent (SGD) as the preferred optimization method. This method incorporates additional information, such as tags, to improve predictions.

The referenced study proves this as it achieves better predictions by integrating the tag-based latent features. Other advanced techniques, such as deep learning-based recommenders or hybrid methods combining content-based and collaborative filtering, are also examples commonly discussed in current universities such as the University of California, San Diego.

Similarities and Differences in Findings

Prior work like the "Using Tags and Latent Factors" study incorporates tagging as an additional input, while our approach solely relied on ratings to simplify the model. Though the method is effective, we were still able to achieve a high accuracy rating, showing that the trade-off between model complexity might be overshadowed by simplicity and ease of implementation.

Part 5: Results & Conclusions

The final latent factor model achieved a test MSE lower than all of our baselines with a final MSE of 1.486.

Baseline	MSE
Using Global Averages	1.60
Using User Averages	1.57
Using Recipe Averages	1.77
Using Either Average	1.53

The latent embeddings ($\gamma_U and\ \gamma_R$) proved essential for capturing nuanced relationships, allowing the model to learn complex interactions between user preferences and recipe characteristics. Additionally, global, user, and recipe averages provided reliable fallback predictions, further enhancing the model's baseline performance.
The latent factor model's components offer meaningful insights into user and recipe dynamics. User biases (β_U) revealed individual preferences, while recipe biases (β_R) reflected the general appeal and popularity of specific recipes. The learned latent factors uncovered deeper patterns, such as how user tastes align with particular recipe attributes, demonstrating the power of these embeddings in extracting subtle interactions. Key factors contributing to the model's success include the combination of latent factors with explicit averages. This improved the robustness of predictions, and the use of regularization to prevent overfitting and maintain generalizability. Overall, the latent factor model effectively predicts user satisfaction and provides valuable insights into the interplay between user behavior and recipe features, establishing its utility in recommendation systems for food-related datasets.

Our decision to use a Latent Factor Model (LFM) was based on its ability to capture hidden relationships, adapt to collaborative filtering, and handle high dimensionality. With the recipe data, we recognized that user preferences and recipe characteristics involve complex, subtle interactions that are difficult to identify with

simpler models. The latent factors in the model represent these underlying variables, such as the unique attributes of recipes and user preferences. Additionally, collaborative filtering allowed our LFM to associate similar users based on their preferences and identify patterns in recipe ratings.

Another advantage of the LFM was its ability to handle the large scale of the dataset by reducing dimensionality, making the model more efficient and scalable despite the significant number of users and recipes. While we faced limitations in hyperparameter tuning and resource constraints due to memory and compute capacity on our cluster, the basic implementation of the LFM yielded solid results and accurate predictions.

Looking ahead, this project has room for expansion, such as incorporating health-related factors like dietary restrictions or nutritional content. These additions could enhance the model's utility for communities with allergies or those aiming to manage their caloric intake. Overall, the LFM proved to be a strong choice for this dataset, and we are optimistic about building on this foundation in future iterations.

1. Harper, F. M., & Konstan, J. A. (2015). The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS), 5*(4), Article 19. https://doi.org/10.1145/2750511.2750528

2. Li, S. (2018). *Food.com Recipes and Interactions* [Data set]. Kaggle. Retrieved from https://www.kaggle.com/datasets/shuyangli94/food-com-recipes-and-user-interactions

3. Weston, J., Peterson, C., Carter, C., & Greenberg, N. (n.d.). *CSE158: Programming assignments for CSE 158/258* [Computer software]. GitHub. Retrieved December 3, 2024, from https://github.com/jak-weston/CSE158