CSE258 Report: Rating Prediction for Recipes

Xinyi Liang
UC San Diego
La Jolla, USA

Jiawei Lian
UC San Diego
La Jolla, USA

Abstract

This study explores predicting user ratings for recipes on the "Food.com Recipes and Interactions" dataset, which includes over 180K recipes and 700K reviews. We embarked on a comprehensive analysis involving exploratory data analysis, feature engineering, and the application of various predictive models. Our feature set integrates detailed recipe information and user-specific details, with rigorous preprocessing for text-based data, including sentiment analysis using advanced techniques like BERT. The predictive task, treated as a classification problem, employs linear regression, naive Bayes, collaborative filtering, neural networks, and latent factor models, each evaluated for accuracy in classifying recipes into discrete rating categories. The analysis reveals the nuances of user preferences and the complexity of user-recipe interactions. The findings underscore the efficacy of diverse modeling approaches in understanding user ratings, providing valuable insights for enhancing recommendation systems and user experience in online culinary domains.

Keywords: Predictive Modeling, User Ratings, Data Analysis, Linear Regression, Latent Factor Model, Collaborative Filtering, Naive Bayes Classifier, Sentiment Analysis

1 Introduction

In the realm of online culinary platforms, understanding and predicting user preferences for recipes play a pivotal role in enhancing user experience and recommendation systems. Our project delves into the intriguing task of predicting user ratings on recipes, leveraging a rich set of features encompassing both recipe information and user-specific details. Here, we used Food.com Recipes and Interactions dataset to do predictive task.

The central objective of our work is to decipher the intricate relationship between user preferences and recipe characteristics in the context of rating prediction. In the vast landscape of culinary exploration, users rely on ratings to guide their choices, making accurate predictions an invaluable asset for personalized recommendations and an enhanced culinary journey.

Our approach to this task have predominantly treated it as a classification problem, given the discrete nature of ratings on a 0-5 scale. Various models have been employed, including linear regression, naive Bayes classifier, collaborative filtering, neural networks, and latent factor models. These models have been effective in capturing different aspects of the user-recipe interaction, providing a foundation for our exploration.

Distinguishing our work is the meticulous integration of diverse features encompassing both recipe and user-centric information. Our feature set comprises detailed recipe information such as ingredients, nutritional content, description, submission date, contributor details, number of steps, and preparation time, alongside user-specific details including previous ratings, review text, and rating dates.

For text-based information, a rigorous preprocessing phase involves data cleaning by removing punctuation and standardizing capitalization. Further, sentiment analysis is conducted using both traditional techniques like TF-IDF and cutting-edge methods like BERT [1], providing nuanced insights into the emotional context of user reviews.

While existing models have primarily focused on generating predictions based on the inherent qualities of recipes, our approach goes beyond by intricately intertwining user preferences with the recipe's intrinsic attributes. This comprehensive fusion aims to not only predict ratings but also to understand the nuanced interplay between user sentiment, recipe features, and the evolving landscape of culinary exploration.

In this paper, we present a detailed exploration of our methodology, including the feature engineering process, the selection of models, and the integration of sentiment analysis techniques. Through empirical evaluation on a diverse dataset, we showcase the effectiveness of our approach in providing accurate and personalized recipe ratings.

2 Exploratory Data Analysis

The "Food.com Recipes and Interactions" dataset is a comprehensive dataset related to recipes and user interactions from the Food.com platform. It consists of 180K+ recipes and 700K+ recipe reviews covering 18 years of user interactions and uploads on Food.com (formerly GeniusKitchen). The interaction data include [user id, recipe id, date of interaction, rating, review text]. The recipe data include [recipe name, recipe id, minutes to prepare recipe, contributor id, submitted date, tags for recipe, nutrition information, steps number, text for step, description]. To dig down into these two dataset to motivate the design of our model, first we left join the interaction data and recipe data according to the recipe id.

Table 1. Variables with Missing Values

name	description	review
1	23510	169

2.1 Missing Values

We list the variables with missing values in Table 1. We can see that the only column with a significant amount of missing values is the description. There are also 165 interactions that involve a rating with no review. Therefore, it is not wise to use description to predict the rating.

2.2 Feature Engineering

2.2.1 Time Components. We split the submission date of recipe and reviews into year and month variables.

2.2.2 Nutrition Values. The original nutrition values is a vector of size 7. Each element of these vectors represents calories, total fat, sugar, sodium, protein, saturated fat, and carbohydrates, respectively. We separate the vector to extract these elements into their own columns.

2.2.3 Text Length. Extract review and description length as new variable to see whether the length of them will impact the rating.

2.3 Data Visualization and Analysis

2.3.1 Rating. Our task is to predict the rating according to the user and recipe information. Therefore, it is important to know the distribution of rating.

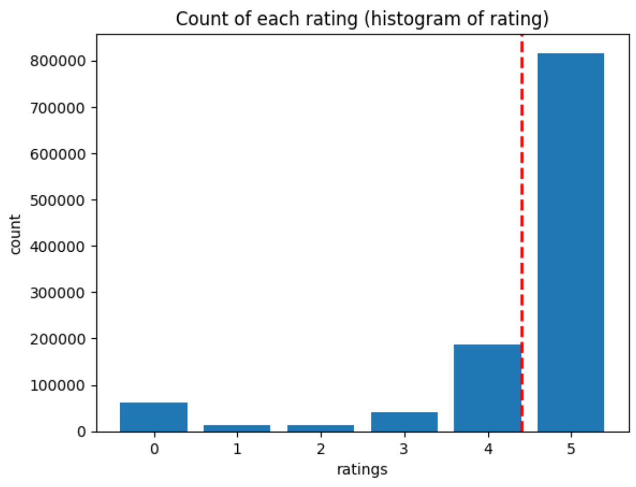


Figure 1. Distribution of Recipe Ratings

Figure 1 shows that the dataset might be bad at predicting reviews with a rating score of 0,1,2,3 and would be better at predicting reviews with a rating score of 4 or 5. This is because the dataset has a lot of reviews with a rating score of 5 than the others. The mean of the rating is **4.41**.

2.3.2 Minutes. From Figure 2, we can easily see there are few outliers in recipe minutes. The outliers are (id: 261647, name: no bake granola balls, minutes: 2147483647) and (id: 447963, name: how to preserve a husband, minutes: 1051200). We eliminate these two outliers from our dataset.

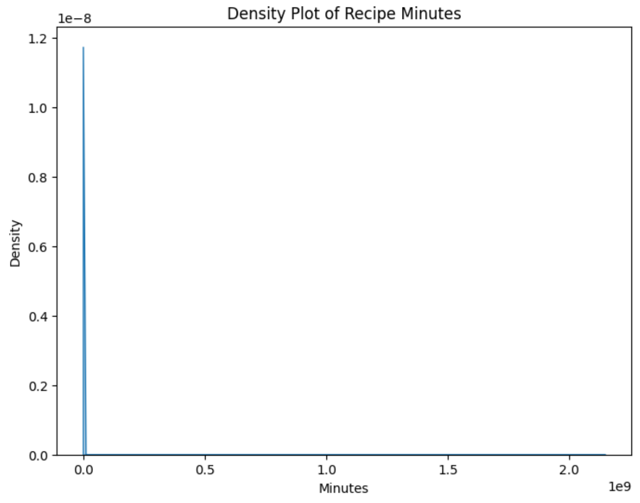


Figure 2. Distribution of Recipe Preparation Time (min)

2.4 Feature Selection

To check what features have impact on rating, we draw the correlation matrix of some features.

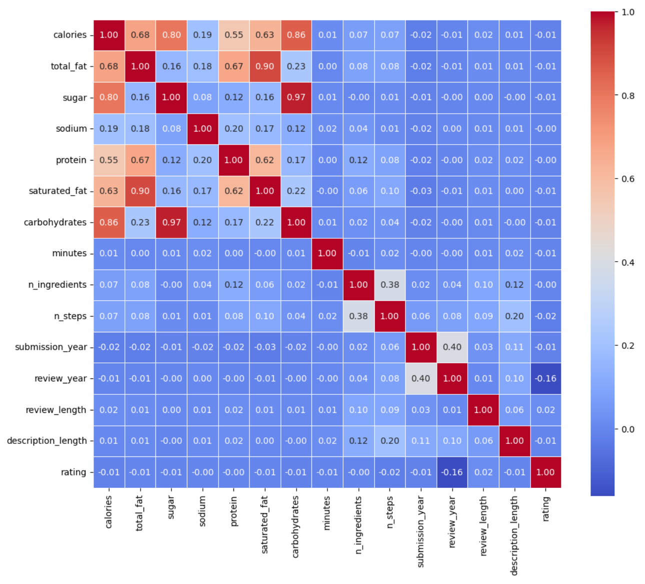


Figure 3. Correlation Plot between Features

The one negative relationship is rating and review year, interestingly, which means that reviews have become slightly more negative over time.

2.5 Text Analysis

2.5.1 Ingredients. We extract every ingredient from each recipes' ingredients vector and get the 10 most common ingredient.

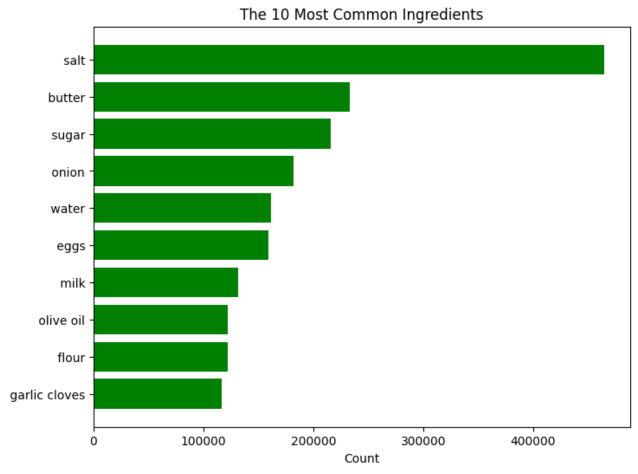


Figure 4. 10 Most Common Ingredients

However, just like "the" in sentence, it is common to see these ingredients whether in bad recipe or popular recipe. Therefore, we extracted ingredients by focusing on the high-rating recipe and low-rating recipe.

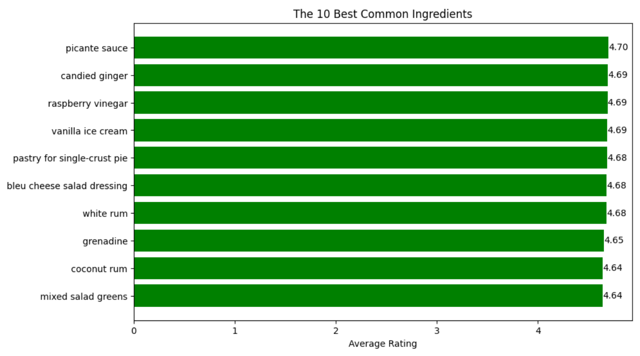


Figure 5. 10 Best Common Ingredients

In the following model building, we will use best common ingredients and worst common ingredients to fit the model.

2.5.2 Exclamatory Reviews. As is known to all, people like using exclamation marks (like "!") to express their love, in which they may give high rating to this recipe. Therefore, we extract exclamation marks and draw the number of exclamation marks versus rating to see their relationship. The correlation is **0.15**.

2.6 Word Clouds

Here are the word clouds of recipes' ingredients and names:

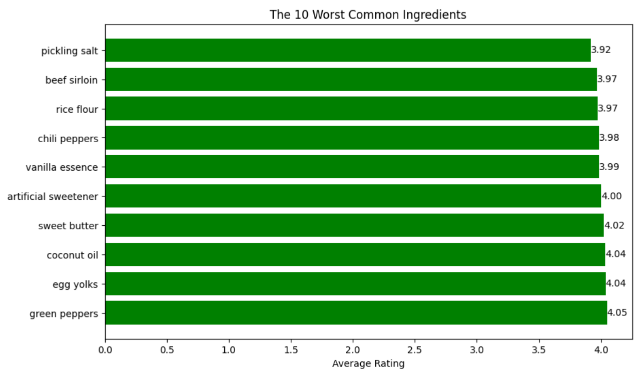


Figure 6. 10 Worst Common Ingredients

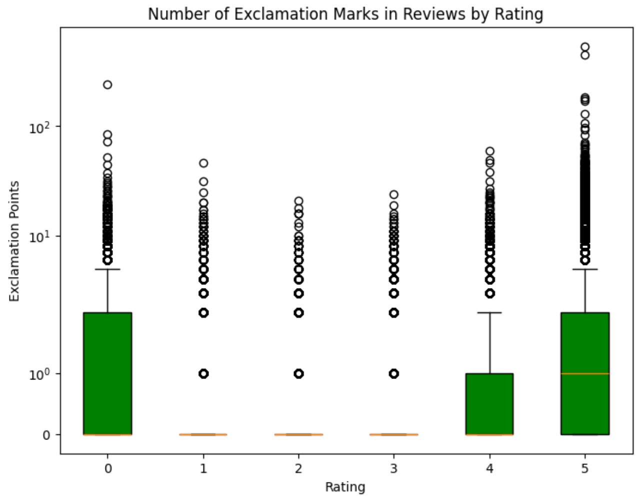


Figure 7. Relationship between Exclamation Marks and Rating

3 Predictive Task

3.1 Task Description

The predictive task is to classify recipes into different rating categories based on user and recipe information. Instead of predicting the exact numerical rating, we aim to categorize recipes into discrete classes, which are specific rating values (e.g., 1-star, 2-star, ..., 5-star).

3.2 Evaluation Metric

The evaluation metric for the classification task will be **accuracy**. Since the ratings are discrete, we want to assess how well the model correctly predicts the rating category for each recipe. We split the dataset into training and test part. We use test part to give the final evaluation of the model. Also, we use cross-validation (GridSearchCV) for parameters tuning.

The formula for calculating accuracy is given as:

$$\text{Accuracy} = \frac{\sum(\mathbb{1}\{(y_{\text{pred}}) = (y_{\text{test}})\})}{\text{len}(y_{\text{test}})}$$

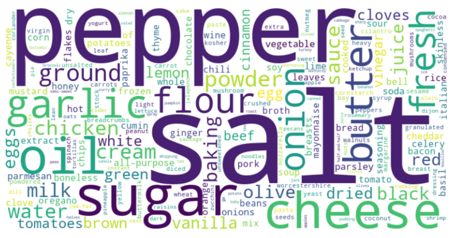


Figure 8. Ingredients Word Clouds

salad
chicken
chocolate
sauce
cheese
cake
soup
bread
cookies
easy
cream
rice
pie
beef
potato
pork
casserole
baked
lemon
pasta
bacon
shrimp
butter
sweet
apple
roasted
green
potatoes
orange
muffins
pot
bean
corn
tomato
crock
spinach
peanut
style
stuffed
grilled
fried
garlic
italian
low
turkey
pizza
zucchini
cinnamon
banana
beans
honey
strawberry
chili
sausage
dip
creamy

Figure 9. Recipe Names Word Clouds

where:

- y_{pred} represents the rounded predictions.
- y_{test} represents the actual test labels.
- $\mathbb{1}\{\}$ denotes the indicator function, which is 1 when the condition inside is true, and 0 otherwise.
- The sum is taken over all the test samples, and the total is divided by the number of test samples to get the accuracy.

3.3 Validating Model Predictions

To assess the validity of the model's predictions, we analyze the confusion matrix to understand the distribution of correct and incorrect predictions across rating categories.

4 Model

4.1 Majority Class Baseline

A simple baseline is predicting the majority class (most common rating) for all recipes. This provides a baseline to beat and helps evaluate the model's performance

The accuracy of the baseline model is 0.166.

4.2 Linear Regression

4.2.1 Model Description. We selected the linear regression model, specifically Ridge regression, as the primary method for predicting users' ratings of recipes. This decision was based on the following considerations: Firstly, linear regression is simple and easy to understand, providing intuitive explanations for the impact of features. Secondly, we assume a linear relationship between users' ratings and certain features of the recipes and users. Finally, by introducing L2 regularization, Ridge regression helps us address the issue of multicollinearity among features, thereby reducing the risk of overfitting. The model's predicted results were rounded off to adapt to the practical application of the rating system.

4.2.2 Model Optimization Strategy.

1. **Feature engineering**: We standardize numerical features to ensure the model does not favor features with larger values. Categorical features were encoded using one-hot encoding. We utilized the Hugging Face's `transformers` library for user comments on a particular recipe. We created a sentiment analysis pipeline by invoking the `pipeline` function and specifying `sentiment-analysis`. This pipeline uses a BERT-based model for text sentiment analysis. The output sentiment and confidence were used as features.
   Regarding feature selection, we calculated the correlation between features and the target variable, removing features with low correlation to the target. From Figure 10, we see that sentiments, each user's average rating, and each recipe's average rating are correlated with the rating.
2. **Hyperparameter Tuning**: We employed grid search to determine the optimal hyperparameter λ for Ridge regression. Using cross-validation, we evaluated the model's performance under different λ values, selecting the best regularization strength to balance bias and variance.
   The Ridge regression model can be represented as:

$$\text{Minimize}\left(\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2\right)$$

   where:
   - y_i is the response variable for the i^{th} observation.
   - β_0 is the intercept of the model.
   - β_j is the coefficient for the j^{th} explanatory variable.
   - x_{ij} is the j^{th} explanatory variable for the i^{th} observation.
   - λ is the regularization parameter, controlling the magnitude of the coefficients.
   - p is the number of explanatory variables.
   - n is the number of observations.

   The regularization term $\lambda\sum_{j=1}^{p}\beta_j^2$ aims to prevent overfitting by constraining the size of the coefficients.
3. **Cross Validation**: We also implemented cross-validation to ensure a more reliable assessment of the model's performance, safeguarding against overfitting and confirming that the model generalizes well to unseen data.

After optimizing the model on the training set and predicting the data in the test set, we got an accuracy of about **0.72**.

4.2.3 Challenges Encountered. During the model development process, we encountered several challenges. The

Table 2. Linear Regression Model Parameters

Parameter	
sentiment	0.48517008
user average rating	0.7218734
recipe average rating	0.81221282
λ	10
Metric	
Accuracy	0.72

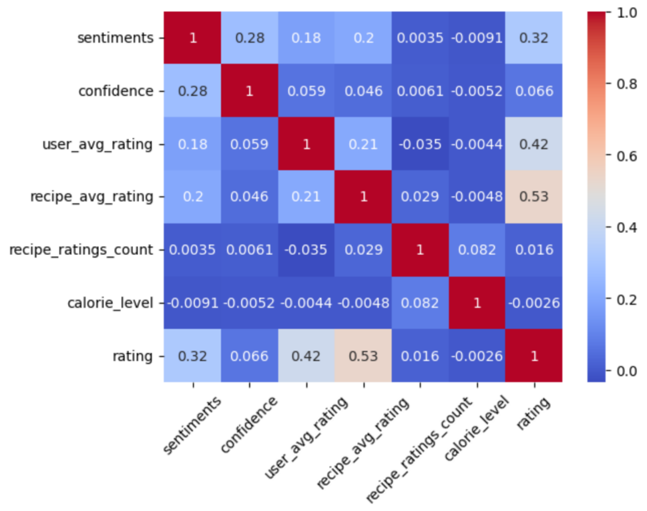


Figure 10. The correlation between features and the rating

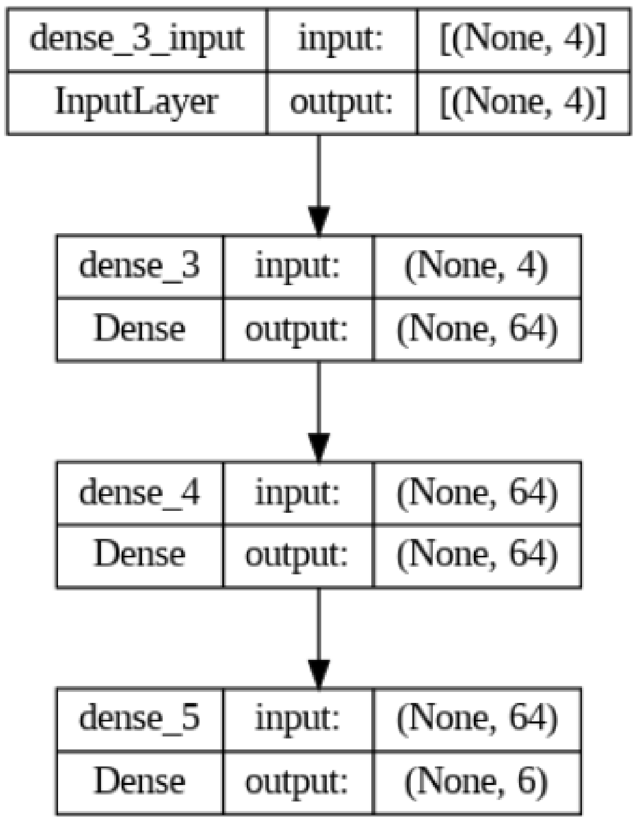


Figure 11. The architecture of the neural network

initial linear regression model exhibited a degree of overfitting. By introducing Ridge regression and adjusting the regularization strength, we were able to mitigate this issue to some extent.

4.2.4 Other Considered Models.

In addition to linear regression, we considered neural networks to analyze the non-linear relationships.

The neural network receives three features: sentiment, confidence, user average rating, and recipe average rating. The network has two hidden layers with 64 neurons each, and the activation function ReLU is used. As it's a classification problem, the output layer has six neurons (corresponding to ratings 0-5), using a softmax activation function.

The model achieves an accuracy of **0.78**.

4.2.5 Unsuccessful Attempts.

Throughout the model development process, we experimented with various variants of linear regression, including polynomial regression. However, these methods either led to excessive complexity or did not significantly improve prediction accuracy. Also, we tried extracting features from ingredients using PCA for feature engineering. However, the components outputted from PCA do not show a high correlation score.

4.3 TF-IDF and Naive Bayes Classifier

4.3.1 Model Description.

The Naive Bayes algorithm is a supervised machine learning algorithm based on the Bayes' theorem. It is a probabilistic classifier that is often used in NLP tasks like sentiment analysis (identifying a text corpus' emotional or sentimental tone or opinion). Here, the sentiment analysis is the hidden layer of our model, and the final output is rating 0-5 (the final class).

1. **Data cleaning**: We clean the review text using a simple texthero call, so that our feature representations (bag of words -> tf-idf) is not filled with bad vectors. The new review text would exclude punctuation, numbers, single chars, and multiple spaces from each essay.
2. **Text Embeddings**: We also tokenize the text using a bag of words -> **TF-IDF** objects. Then, this object will be used to create text embeddings of the review text, and feed them into Gaussian Naive Bayes Model.

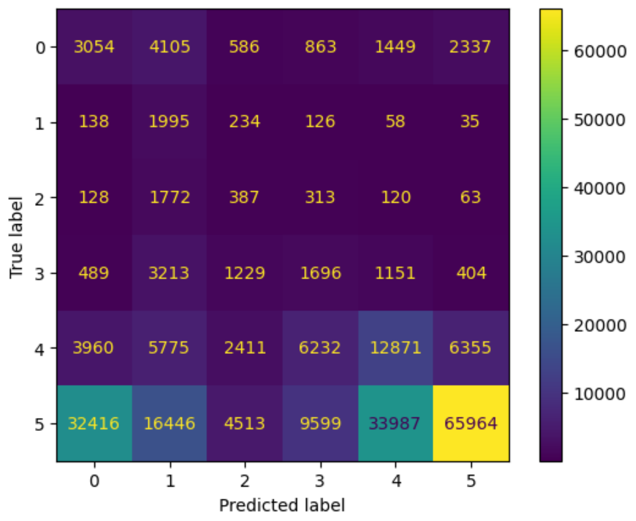
4.3.2 Model Optimization Strategy.

The TF-IDF based Gaussian Naive Bayes Classifier can be written in a mathematical way:

$$\hat{y} = \underset{k\in\{1,2,..K\}}{\arg\max}\ p(C_k)\prod_{i=1}^{n} p(x_i|C_k)$$
$$p(x_i = v|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}} \quad (1)$$
$$x_i = TF(word_i, doc)\cdot IDF(word_i, doc)$$

After optimizing the model in the training set, we got an accuracy of **0.64** on the test set.

4.3.3 Challenges Encountered. As expected in 2.3.1, the Naive Bayes Model predicted better on the test data that was labeled as 5 rather than the the out 4 numbers. Its also interesting to note that not all review text had a actual rating score indicative of their review text because the rating score is optional on food.com. For example, take a look at some of review text with ratings of 0 that said positive words indicative of a 4 or 5 rating score. This explains why the model would predict some reviews having a rating score of 0, when the review text actually had a rating score of 5.



True \ Predicted	0	1	2	3	4	5
0	3054	4105	586	863	1449	2337
1	138	1995	234	126	58	35
2	128	1772	387	313	120	63
3	489	3213	1229	1696	1151	404
4	3960	5775	2411	6232	12871	6355
5	32416	16446	4513	9599	33987	65964

Figure 12. Confusion Matrix of Naive Bayes Model

4.4 Collaborative Filtering

4.4.1 Model Description. Collaborative filtering is a technique used in recommendation systems to predict or recommend items (such as movies, products, or recipes) to users based on the preferences and behaviors of other users. The idea behind collaborative filtering is to leverage the collective wisdom of a group of users to make personalized recommendations for an individual user.Here, we use item-based collaborative filtering as our baseline model.In item-based collaborative filtering, the system recommends items that are similar to the ones the target user has liked or interacted with in the past. The similarity between items, we first try Jaccard Similarity.

$$rating(u,i) = \frac{\bar{R}_i + \sum_{j\in I_u\setminus\{i\}}(R_{u,j}-\bar{R}_j)\cdot Sim(i,j)}{\sum_{j\in I_u\setminus\{i\}} Sim(i,j)}$$

where i is the recipe id.

When computing similarities return the item's average rating if no similar items exist (i.e., if the denominator is zero), or the global average rating if that item hasn't been seen before.

The baseline model's accuracy is around 0.675

4.4.2 Improvement Strategy I. From 2.4, we see that rating is negatively related with review date. This implies that ratings from a longer time ago are relatively less important for the current rating.Therefore, we introduce a decay function.

$$rating(u,i) = \frac{\bar{R}_i + \sum_{j\in I_u\setminus\{i\}}(R_{u,j}-\bar{R}_j)\cdot Sim(i,j)\cdot f(t_{u,j})}{\sum_{j\in I_u\setminus\{i\}} Sim(i,j)\cdot f(t_{u,j})}$$

where i is the recipe id.

Here, the decay function we use is (c is a controllable parameter):

$$f(t_{u,j}) = e^{-ct}$$

We select best parameter using Cross Validation. The best c is 0.01. **The optimized model's accuracy is around 0.693**, which confirms the importance of review date.

4.4.3 Improvement Strategy II. In 4.3, we have transformed review text into TF-IDF vector. Based on Improvement Strategy I, We apply consine similarity on these vector to calculate the similarity of two reviews, which are related with the recipe.

$$Sim(i,j) = cosine\_sim(review(u,i), review(u,j))$$

where review(u,i) has been transformed into tf-idf vector.

The improved model's accuracy is around 0.708, which confirms the sentiment impact of reviews on ratings.

4.4.4 Challenges Encountered. Collaborative filtering can face challenges in scenarios where the user-item interaction matrix is sparse, meaning that users have not interacted with many items or have provided ratings for only a few items. As the number of users and items grows, the computational complexity of finding similar users or items can increase.

4.5 Latent Factor Model

4.5.1 Model Description. The Latent Factor Model (LFM), specifically Singular Value Decomposition (SVD), was used since it dives deep into users' preferences and the features of recipes. Given the typically sparse nature of user-recipe interaction data, LFM is particularly adept at extracting meaningful insights from limited user feedback. Our model, implemented using the `surprise` library, predicts recipe ratings by uncovering latent factors influencing user preferences and recipe features.

The Latent Factor Model using Singular Value Decomposition can be represented as follows:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{2}$$

The objective function to be minimized, which includes the regularization terms, is:

$$\min \left(\sum_{(u,i)\in R} (r_{ui} - \hat{r}_{ui})^2 + \lambda_{bu} b_u^2 + \lambda_{bi} b_i^2 + \lambda_{pu} ||p_u||^2 + \lambda_{qi} ||q_i||^2 \right)$$

where:

- $\hat{r}_{ui}$ is the predicted rating of user u for item i.
- r_{ui} is the actual rating of user u for item i.
- μ is the global average rating.
- b_u is the user bias for user u.
- b_i is the item bias for item i.
- q_i is the latent factor vector for item i.
- p_u is the latent factor vector for user u.
- λ_{bu}, λ_{bi}, λ_{pu}, and λ_{qi} are the regularization parameters for user bias, item bias, user latent factors, and item latent factors, respectively.
- $|| \cdot ||^2$ denotes the squared Euclidean norm.

4.5.2 Optimization Strategy.

1. **Hyperparameter Tuning**: We employed `GridSearchCV` for systematic hyperparameter optimization, focusing on the optimal settings for the number of latent factors, learning rate, and regularization parameters.
2. **Cross Validation**: We also implemented cross-validation to ensure a more reliable assessment of the model's performance, safeguarding against overfitting and confirming that the model generalizes well to unseen data.

Table 3. Latent Factor Model Parameters and Metrics

Parameter	
Number of Factors (`n_factors`)	5
Learning Rate (`lr_all`)	0.001
User Bias Regularization (`reg_bu`)	0.001
Item Bias Regularization (`reg_bi`)	0.1
User Latent Factors Regularization (`reg_pu`)	0.1
Item Latent Factors Regularization (`reg_qi`)	0.1
Metric	
Accuracy	0.68

After optimization, the model gives an accuracy of **0.68**.

4.5.3 Challenges Encountered. Initially, handling large datasets was a significant challenge. However, this issue was mitigated by the efficient implementation of SVD in the `surprise` library. Concerns regarding model overfitting, especially with a large number of latent factors, were addressed through careful hyperparameter tuning and cross-validation.

5 Literature

One other research that Food.com dataset is *Generating Personalized Recipes from Historical User Preferences* [3]. They combine two important tasks from natural language processing and recommender systems: data-to-text generation [2] and personalized recommendation [4]. The personalized recipe generation involves expanding upon a recipe name and incomplete ingredient information to produce comprehensive, natural-text instructions aligned with the user's historical preferences. Their approach incorporates both technique- and recipe-level representations derived from the user's previously consumed recipes.They employ an attention fusion layer to merge these 'user-aware' representations, guiding the generation of recipe text.

From their research report, we realized the significance of ingredients and historical preferences for users. Moreover, an attention fusion layer can be employed to comprehend some text related to users and recipes. Unlike this paper, we do not focus on generating recipes but rather on understanding their preferences, attempting to infer their degree of liking towards ratings.

6 Results and Conclusions

Based on the comprehensive dataset analysis, the neural network model with sentiment analysis emerged as the most effective, achieving an accuracy of 0.78. This outperformed other models like the Ridge Regression (0.72), Collaborative Filtering (0.71), Latent Factor Model (0.68), and Naive Bayes Classifier (0.64). All of these models outperformed the baseline model, the Majority Class Baseline model, which had the lowest accuracy (0.166).

Table 4. Model Performance Comparison

Model	**Accuracy**
Neural Network	0.78
Ridge Regression	0.72
Collaborative Filtering	0.71
Latent Factor Model	0.68
Naive Bayes Classifier	0.64
Majority Class Baseline	0.166

In terms of feature representations, from the correlation graph, we can see that sentiments, each user's average rating, and each recipe's average rating are correlated with the rating. From the model results, these features also proved highly effective. In contrast, features like calories, n_steps, do not contribute to the ratings.

Although the neural network model performs the best, the parameters are difficult to interpret. The parameters of the linear regression model are in Table 2. For the parameters of the linear regression model, the coefficients indicate the magnitude and direction of the influence of the feature on the rating. The coefficient for sentiment is 0.49, suggesting a moderate positive effect. The user average rating, with a coefficient of 0.72, has a stronger positive impact. The recipe average rating, with the highest coefficient of 0.81, implies that it is the most influential factor, where an increase in the recipe rating leads to a substantial increase in the target prediction. This analysis reveals that historical ratings of users and recipes play a more critical role in the predictive model than sentiment analysis.

The outperformance of the neural network model might be because the neural network can learn complex patterns in the data. The relationship between the features and ratings is not linear. Incorporating sentiment analysis provided a nuanced understanding of user opinions, which is critical in rating predictions. In comparison, linear models like Ridge Regression, while robust, were limited in capturing the complexities of user preferences and textual nuances. Collaborative Filtering and Latent Factor Models, although effective in capturing user-item interactions, omit the view data, which, as mentioned, contributes to the rating. The Naive Bayes Classifier's simplistic feature independence assumption was a drawback in the context of interdependent and nuanced features like sentiment.

The neural network's success demonstrates the value of handling complex data patterns and non-linear relationships in predictive modeling. This also emphasizes the crucial role of feature engineering, cross-validation, and hyperparameter tuning in enhancing model accuracy and understanding intricate data relationships.

References

[1] Shivaji Alaparthi and Manit Mishra. 2020. Bidirectional Encoder Representations from Transformers (BERT): A sentiment analysis odyssey. *arXiv preprint arXiv:2007.01127* (2020).

[2] Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research* 61 (2018), 65–170.

[3] Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian McAuley. 2019. Generating personalized recipes from historical user preferences. *arXiv preprint arXiv:1909.00105* (2019).

[4] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K Lam, Sean M McNee, Joseph A Konstan, and John Riedl. 2002. Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th international conference on Intelligent user interfaces*. 127–134.

Received 5 December 2023