DS4420 Project Report

By Dennis Ho, Avril Mauro, Heidi Eren

I. Introduction and Literature Review

Across online platforms and social media, there are an overwhelming number of recipes and variations of recipes available for users to use. Navigating through these recipes can often be challenging since users have different preferences, health needs, and dietary restrictions that they adhere to. Due to indecisiveness or limited time in meal preparation, users may also resort to convenient and unhealthy food options, leading to unenjoyable and monotonous meals. To ease this selection process and accommodate one's preferences, our project proposes a personalized recipe recommendation system that leverages machine learning to predict a user's likely rating for a recipe they have not rated before and recommend new recipes that align with their historical preferences and dietary patterns.

Our engine utilizes multilayer perceptron and collaborative filtering through a hybrid approach to focus on a user's previously made ratings to make predictions of their ratings on new recipes, and perform recommendations accordingly. For new users, predictions will be made based on choices made by users with similar preferences as well as the ingredients in the recipes. With this hybrid approach, the food recommendation system serves to be more accurate and effective in considering both user preferences and intrinsic features of the recipes themselves.

One project that was done before that aimed to address the same problem of personalized recipe recommendation is the "Plates4U" project, as they hope to use their work to simplify meal selection through suggesting recipes tailored to different users' preferences. Unlike our hybrid approach of using both collaborative filtering and multilayer perceptron to try and predict user ratings for different recipes and recommend recipes based on their historical behavior and dietary patterns, Plates4U leveraged a content-based filtering model using cosine similarity on ingredient vectors. The system that the team has developed is able to generate recommendations based on ingredients provided by users and applies basic filtering by cuisine and cook time. However, when user input is limited, there is also a backup system that recommends top-rated recipes. While this project also has the same goal of enhancing each user's experience when they are trying to discover new recipes, their approach lacked deeper personalization and did not incorporate user ratings or behavioral history into their recommendation process like our system. In addition to that, they communicated how their model is less scalable and efficient for dynamic or large-scale deployment. Either way, this project definitely provides a good comparison to ours and demonstrates a more lightweight approach that is focused on immediate usability of the tool without requiring user accounts.

Another project that tried to solve the same problem as us is the "Machine Learning Based Food Recipe Recommendation System" by Vivek et al. Their work is also about tailoring recipe suggestions depending on user preferences, however, their methodology takes a more traditional route when compared to ours. Unlike our hybrid approach that combines collaborative filtering with MLP to "learn" patterns from both user behavior and recipe characteristics, their system primarily relied on similarity-based collaborative filtering. They calculated user and item similarities using measures such as Euclidean distance and Pearson correlation, and log-likelihood, then generated recipe recommendations through a fixed-size or threshold-based neighborhood. While their approach is more computationally efficient and may be easier to interpret, it doesn't leverage the representational power of neural networks or incorporate recipe-level features (e.g. ingredients or dietary tags). It also doesn't address the cold-start problem as effectively and directly as our approach, which is when recommendations are difficult to generate for new users or items due to a lack of prior data. Despite that being the case, their work offers a solid benchmark for classical collaborative filtering techniques and helps highlight how our model emphasizes deeper personalization and flexibility.

Recommendation systems that utilize this hybrid approach are relatively rare. Most systems leverage collaborative filtering as it focuses on the preferences and behaviors of similar users or items to make predictions on what a user would like. One example of this from Kaggle leverages collaborative filtering to make restaurant recommendations based on customer ratings of the food and service. Through user-based collaborative filtering, the model finds users with similar ratings and recommends restaurants based on what these similar users liked. The project utilizes probabilistic matrix factorization to create embedding matrices for users and items respectively. The dot product of user and item embeddings results in restaurant predictions. The model is then trained to minimize the loss between actual and predicted ratings with MSE. Another project compares item-item collaborative filtering and user-user collaborative filtering to create a food recipe recommendation system based on similar users or items and their existing interactions. With matrix factorization and baseline estimation, the model creates a recommendation across multiple rounds to learn from user interaction feedback.

While these collaborative filtering approaches are effective for basic recommendation systems, the model can perform better by predicting on more complex preferences such as nonlinear ones. A hybrid approach with MLP can process both the embeddings and learn arbitrary interactions between their features to learn richer patterns. One paper titled "A Deep Neural Networks Model for Restaurant Recommendation Systems in Thailand" by Saelim et al. integrates deep collaborative filtering with multi-layer perceptrons to create restaurant recommendations. The deep collaborative filtering utilizes user-item interactions as latent factors to predict user preferences. The MLPs integrate term frequency-inverse document frequency (TF-IDF) techniques to process texts from reviews as meaningful features. This model leverages this hybrid approach to generate a more accurate ranked list of restaurant recommendations based on user preferences and supplementary restaurant information. Hence, in addition to its ability to learn additional features including user demographics and recipe metadata, our proposed ensemble model can deliver more accurate and personalized recipe recommendations in capturing both linear and nonlinear interactions through a combination of the best-performing models that a collaborative filtering or MLP approach would not be able to do alone.

II. ML Methodology and Data Collection

In order to build an effective and personalized recipe recommendation system, it is crucial that our dataset successfully reflect the diverse range of recipes, user preferences, and the nuanced behaviors that different users may exhibit when they rate recipes online. Our data was sourced from a publicly available Kaggle dataset containing over 180,000 recipes and approximately 700,000 user reviews from Food.com, including 18 years of user interactions/ratings. The dataset we selected contained roughly 1.1 million records of interaction data which provides a rich but noisy foundation for us to build the predictive models we envision. We used Python to handle all of our data preprocessing for the models.

As we review our dataset, a major challenge identified was the extremely left-skewed distribution of recipe ratings. The recipe ratings ranged from 0 to 5, however, the majority of those ratings were 5-star ratings. The mean rating for all of the reviews in the dataset was 4.41, and more than 70% of the recipe ratings (55K+) were valued at a 5. This skewed distribution proposed a serious problem for both our collaborative filtering and multilayer perceptron models, as it will inhibit those models' ability to distinguish between truly high-quality recipes and average or poor ones. For example, if there is a dataset where most recipe ratings represent only one class (e.g., "good"), models built to predict recipe ratings will overfit or become biased towards predicting the dominant label, which will reduce the predictive power and generalizability of those models.

To address the issue of potential overfitting and unfair predictions, we decided to rebalance the dataset through a stratified sampling process which focused on improving the distribution of ratings across the full 1–5 scale. This allowed us to take 1,939 ratings from each rating class. To further improve the effectiveness of our dataset, we extracted the stratified sample from only the latest three years (2015-2018). We see this approach to have multiple benefits. For one, it provided a more even spread of ratings, which reduces class imbalance and improves model sensitivity to rating differences. Secondly, it will allow our recommendation/prediction system to focus on more recent user behavior and recipe trends as opposed to older ones, which are likely to better reflect modern culinary preferences and ingredient availability.

Within our dataset, we also decided to exclude the records from users ratings when those users have given less than two ratings. This will ensure that each user in the sample has enough data to form a pattern of preferences, useful for collaborative filtering. We believe that this filtering is critical for collaborative filtering in particular, which relies heavily on user-item interaction matrices. Users with only a single rating offer limited insight into their preferences and will likely add noise to similarity computations between users and items. Our valid user count was 430 and the final dataset had 1,739 total ratings.

We then divided our clean dataset into an 80-20 train-test split and stratified by rating to preserve the balanced distribution during evaluation. We believe that this structure will allow us to assess the model's ability to predict across the full spectrum of ratings rather than being biased toward the previously dominant high scores. After the data is split, we continue with further data cleaning and feature engineering to ensure that our features are aligned with the learning task. For MLP, we needed to encode our categorical variables "tags" and "ingredients" which were lists for each recipe. To do so, we unnest those columns and one-hot encoded the most common entries. For tags, we filtered them to include only those that appeared in at least 25% of the data (roughly 50,000 recipes), as that will allow us to retain meaningful metadata while reducing sparsity and overfitting risks. At the same time, we performed the same process with the ingredients but retained those with over 1,000 appearances across recipes to encode the most informative features while maintaining computational efficiency.

The two methods we used to predict recipe ratings were Multi-Layer Perceptron (MLP) and Item-Item Collaborative Filtering (CF). In our CF, we used a manual cosine similarity function to calculate similarity scores, and we enhanced the model by introducing bias features. This was done by taking the global average rating of the full dataset and it from the average rating of each user. Next, we added that bias value back to the global average to create a baseline for each user. That baseline was used in the latter half of a 70-30 blend in tandem with the CF prediction. This proved to be helpful in indicating aspects of user behavior during the training. For example, there may be users that rate everything highly or everything poorly, and incorporating bias allows us to adequately capture those preferences on a user-by-user basis. Another enhancement we made was time-decay with a 2-year half-life, reducing the weight of ratings by half every two years out, giving more importance to recent ratings. We also explored using ensemble methods to combine the CF with other models that may be better suited for our data, such as Singular Value Decomposition (SVD), which is a matrix factorization technique that decomposes a matrix into three matrices (U, Σ, V^T), effectively identifying latent factors that capture the underlying patterns in the data. We tried both a simple weighted average ensemble method and an advanced Random Forest ensemble. In the simple ensemble, we found the optimal weight by iteratively testing values from 0 to 1 (in 0.05 increments) and selecting the one that produces the lowest RMSE when blending the two prediction methods. The Random Forest ensemble enhanced prediction accuracy by learning the optimal way to combine CF and SVD predictions along with additional features such as user bias (difference between a user's average rating and global average), prediction disagreement (absolute difference between CF and SVD predictions), confidence score (weight representing how many ratings the user had already done), and recency (normalized time-based feature where recent ratings have values closer to 1). The model discovers complex, non-linear relationships between these features and actual ratings, which allows it to make better predictions than a simple weighted average, especially for users or items with particular characteristics.

We also implemented a Multi-Layer Perceptron (MLP) model using the Keras library in R to predict recipe ratings. The MLP architecture combined user and recipe embeddings as well as additional recipe-specific meta features such as preparation time, number of ingredients, and steps. These inputs were passed through fully connected hidden layers with ReLU activation and dropout for regularization, followed by a linear output layer. To optimize the performance of the model, we utilized a Random Grid Search which conducts a random hyperparameter search across various embedding dimensions, hidden layer sizes, dropout rates, learning rates, and batch sizes. After training the model, it was evaluated using standard regression metrics such as mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE), and R^2. To further optimize model performance, we utilized ensemble modeling which averages the predictions from the top-five performing MLP models equally (for simple ensemble modeling) and weighted (for weighted ensemble modeling). The side-by-side plots compare the training and validation loss of the model, predicted vs actual ratings, and model performance across the standard regression metrics.

III. ML Results and Interpretation

METHOD	R2	RMSE	RANK
Item-Item CF	0.1713	1.1072	7
Singular Value Decomposition	0.2563	1.0489	6
Simple Ensemble (CF/SVD)	0.2776	1.0338	5
Random Forest Ensemble (CF/SVD)	0.8832	0.4158	1
Single Model MLP	0.4741	0.8391	3
Simple Ensemble MLP	0.4317	0.7463	4
Weighted Ensemble MLP	0.4928	0.8241	2

Collaborative Filtering

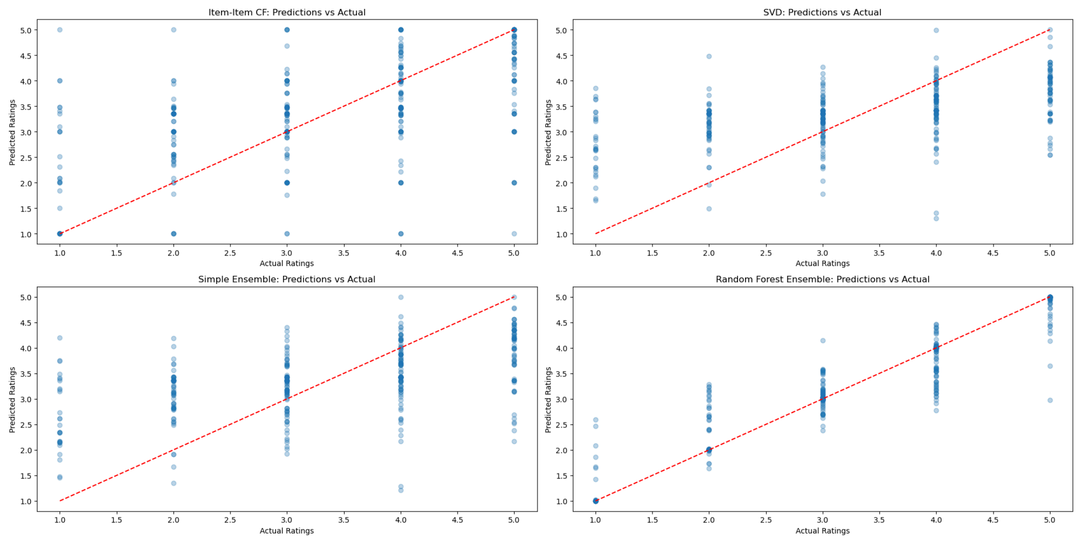
Item-Item CF called out to us immediately over User-User given the nature of our data which contained more recipes (200k+) than users (25k). We hypothesized that this method would be more effective at handling sparsity and demonstrated greater stability in capturing item preferences compared to user preferences. With 15 neighbors and the time-decay and bias enhancements, it achieved an R^2 of 0.1713 and RMSE of 1.1072. This indicated mediocre performance to start as the model explains only about 17% of rating variance and predictions are off by more than 1 class on average. This was our fundamental starting block for the CF, but we had a long way to go to improve our model accuracy.

We conducted trials with alternative approaches that ultimately did not make it to our final solution. User-User CF performed poorly, yielding negative R^2 values due to the dataset's characteristics and challenges with user preference variability. This confirmed our original hypothesis of using Item-Item over User-User CF. Our next trial involved a content-based hybrid experiment incorporating recipe features like tags and ingredients underperformed pure collaborative filtering, suggesting that user rating patterns were not strongly influenced by explicit content features. Those added content features seemed to introduce noise rather than enhancing predictions. This made sense to us because there were over 500 unique tags and ingredients to choose from, and some were incredibly specific (. Even at the top 10-20 tags and ingredients, our hybrid model performed worse than the pure Item-Item CF. To avoid overfitting, we dropped the hybrid approach and explored other ways to blend our predictions to capture more signals.

Next we tried Matrix Factorization with SVD, and this demonstrated significant improvements over basic Item-Item CF, achieving an RMSE of 1.0489 and R^2 of 0.2563. Jumping from 17% to 25% of explained variance was a positive sign for us – although our predictions were still off by 1 class. We found that using 25 latent factors beyond explicit feature representations proved to be helpful in handling the sparsity problem more effectively. The model was able to capture nuanced patterns in user ratings while incorporating both user and item biases. SVD works well for our recipe data because it finds hidden patterns in how people rate recipes. We were pleased with these results and knew we could move forward with our blending technique.

Our first ensemble approach was a simple weighted average of the Item-Item CF and SVD results. Our most optimal weights were a 55-45 split CF and SVD. The performance of this ensemble did increase slightly from the SVD alone, achieving an RMSE of 1.0338 and R^2 of 0.2776. This straightforward linear combination demonstrated that the models captured complementary signals and that SVD contributed more valuable information, representing an important intermediate step in our modeling evolution. We enjoyed this approach, but were not yet satisfied with the results.

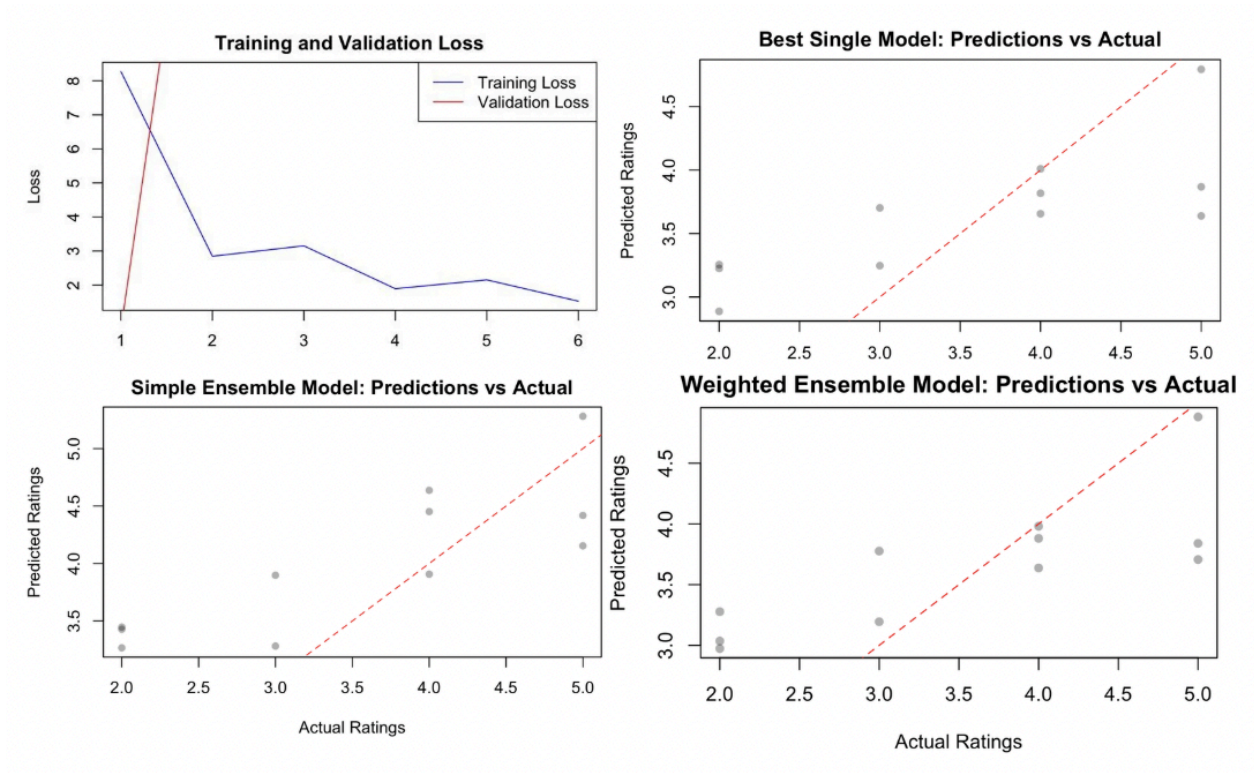
Our breakthrough came with the Random Forest ensemble approach, which dramatically improved performance with an RMSE of 0.4158 and R^2 of 0.8832. With 88% of the rating variance explained by this model, and predictions falling within less than half a class apart, we felt confident in analyzing these results. This advanced method excelled by learning non-linear relationships between model outputs, leveraging additional features beyond base predictions, and making context-aware decisions about which underlying model to trust in different contexts. You can see the results of our trials below:



FEATURE	Bias	Time	Other	Disagreement	SVD	CF
IMPORTANCE	0.945	0.018	0.018	0.009	0.007	0.003

The Random Forest model revealed to us some valuable insights on the nature of our users. It especially highlighted how user rating patterns (bias) greatly influence their actual rating, accounting for 95% of the prediction power. Some people always give high ratings while others are tougher critics, regardless of the actual recipe quality. Time factors matter a bit as well (1.8%) because people's food preferences change with seasons and over time. The discrepancy between the Item-Item CF predictions and SVD predictions also provided useful information (1.7%). To our surprise, the predictions from our sophisticated models (SVD: 0.7%, Item-Item CF: 0.3%) contributed very little compared to knowing a user's rating habits. For recipe recommendations, this means we should focus on understanding each person's rating style first, watch how their tastes change over time second, and worry less about detailed recipe characteristics like ingredients or cooking methods. Knowing if someone is a generous or strict rater tells us more than analyzing the recipes themselves.

Multi-Layer Perceptron



The four subplots above represent the results from the MLP model. The training and validation loss plot shows that the training loss decreased steadily over 6 epochs, which conveys effective learning of the model. The validation loss, however, exponentially increases, which conveys overfitting. The model fits the training data better than the validation data, which may be due to the large size of the embeddings from the dataset. The top right figure represents predictions from the best single MLP model. Majority of the points fall below the prediction line, suggesting that the model underestimates the ratings and may not fully capture user ratings. The bottom left figure represents the simple ensemble model which equally aggregates the predictions from the top five models. The points appear to be clustered to the prediction line, suggesting slightly more accurate predictions compared to the single model. However, the points have higher coverage across higher ratings of 4 to 5. This helps to reduce underprediction and improves the model fit around the prediction line. The bottom right figure represents the weighted ensemble model, which weights inversely by the validation error of the top five models. The predictions appear to be more closely aligned above and below the prediction line with less deviation vertically. With better performing models having more influence, the prediction accuracy increases significantly for the weighted ensemble model.

After running the random grid search, the best performing model was the best single model with embedding dimension of 32, dense layer 128 (dropout=0.4), then dense layer 128 (dropout=0.3), learning rate of 0.0005, epoch of 10, and batch size of 64. For the weighted ensemble model, the MSE was 0.68 and R^2 was 0.49 which resulted in the best performing model in capturing meaningful relationships between the users and the recipe ratings. This means that on average, the squared difference between the actual and predicted values was 0.68, which is relatively lower compared to the other models that penalized large errors more heavily. The R^2 was 0.49 which means the weighted ensemble model explains about 49% of the variability in recipe ratings. This is significantly higher in magnitude than the other models, conveying that meaningful patterns were captured by weighting the top performing models.

The MLP model demonstrates the ability to capture nonlinear relationships between user and recipe ratings. Although the base model is decent, it performs underfitting with higher ratings. With ensemble models, the performance of the predictions improves greatly since the embedding layers and dense transformations in the top performing models capture latent user-recipe interaction and nonlinear effects of recipe features. Hence, the weighted ensemble performs the best balance in error reduction and fit for this data.

IV. Conclusions and Future Work

Our project has the goal of building a personalized recipe recommendation system. Through experimentation with multiple models, including collaborative filtering, singular value decomposition, multilayer perceptron, and ensemble methods, we have found that hybrid approaches significantly outperformed any single-model strategies. To be more specific, the Random Forest Ensemble that combines CF and SVD predictions was able to achieve the highest performance score, with an R^2 of 0.8832 and RMSE of 0.4158. This model and approach benefited from integrating nonlinear relationships between predictors and was able to contextually weight the model outputs for each user.

Our best-performing model leveraged additional features that were not initially included in the dataset, including user bias, rating recency, prediction disagreement, and rating consistency. All of those additional features helped overcome some of the challenges commonly seen in sparse and highly skewed datasets. We also found that user bias, a metric that describes how a user typically rates recipes, was the most influential factor that affects their ratings for specific recipes. That highlighted the importance of accounting for individual tendencies when making similar recommendation systems.

We also took additional steps to meet the key assumptions of the models we are using. For collaborative filtering and ensemble methods, some crucial assumptions are that users we are incorporating into our model have multiple interactions and that the data covers a distributed range of rating values. We effectively addressed these assumptions by filtering out users with fewer than two ratings and using stratified sampling to balance the highly skewed distribution of ratings. Next, we applied a time-decay function to reflect the assumption that user preferences will change over time. These strategies have helped us meet the assumptions around user behavior and data structure. On the other hand, our MLP model was observed to underperform. We believe that this is partly because neural networks assume richer and denser datasets with more metadata and consistent patterns to learn from. And because our dataset was relatively sparse and lacked detailed contextual features, this assumption wasn't fully met which eventually led to weaker performance from the deep learning model.

We observed a few limitations that our models face when being leveraged as a recipe recommendation system. Even though the MLP was flexible, it didn't do a great job capturing the full range of user preferences and using embeddings for user and recipe IDs wasn't enough to reflect individual rating habits. It also didn't directly account for user bias, and the structure of the model was a bit too rigid to handle the sparse data and complex patterns in our dataset. Additionally, the use of Mean Squared Error as the loss function may not have been ideal for predicting user ratings, as it treats the values on the 0–5 scale as continuous and equally spaced. This overlooks the fact that these user ratings are ordinal, and that the difference between values isn't always "meaningful", which can cause the model to overly penalize small errors even when they have little practical significance. On the other end, Item-Item Collaborative Filtering had a better accuracy score but had trouble with cold-start scenarios and sometimes made similar predictions for users with very different tastes. Moving forward, we would like to account for other dimensions/features like seasonality and user profiles into our analysis, explore different model structures and loss functions, and test the system in real time to see how it performs with actual users.

Works Cited

(PDF) machine learning based food recipe recommendation system, January 2023. https://www.researchgate.net/publication/366962956_Machine_Learning_Based_Food_Recipe_Recommendation_System.

Fong, Anthony, Alex Pham, and Zachary Nguyen. n.d. "Plates4U: A Collaborative-Filtering Approach to Recipe Recommenders." https://dsc-capstone.org/projects-2020-2021/reports/project_43.pdf.

Princeashburton. "Collaborative Filtering on Restaurants." Kaggle, October 12, 2018. https://www.kaggle.com/code/princeashburton/collaborative-filtering-on-restaurants.

vipashaaV321. "Collaborative-Food-Recipe-Recommendation-System/Readme.Md at Main · VIPASHAAV321/Collaborative-Food-Recipe-Recommendation-System." https://github.com/vipashaaV321/Collaborative-Food-Recipe-Recommendation-System/blob/main/README.md.

Saelim, Apisara, and Boonserm Kijsirikul. "A Deep Neural Networks Model for Restaurant Recommendation Systems in Thailand." Proceedings of the 2022 14th International Conference on Machine Learning and Computing, Guangzhou, China, Association for Computing Machinery, 2022.