Recipe Explorer Final Report

Machine Learning 5990
April 28, 2024
Maya Griffith, Alaina Birney, Nick Bosley, Nate Gaylinn

Introduction

In an era where varieties of healthy and unhealthy foods alike are available in abundance, individuals can face significant challenges in making informed and healthy dietary decisions. This issue is compounded by the diverse nutritional needs and preferences that vary with each person's body type, activity level, and specific health goals. The complexity and time-consuming nature of understanding and aligning dietary choices with personal health requirements highlight a need for an efficient solution. This paper proposes the development of "Recipe Explorer," a tool that uses machine learning to help by providing users with recipes that satisfy their nutritional needs and preferences, with enough flexibility to adapt to what they're craving at the moment.

We plan to build this tool to be able to recommend relevant recipes to the user based on preferences and nutritional needs and let the user explore related recipes. The purpose is to help users make healthy meal choices without forcing them to follow a rigid menu. We hope to make good recommendations on the first try, but also make it easy for the user to pivot to something that fits their appetite, pantry, and wallet better if they so choose.

The concept of "Recipe Explorer" is situated within the broader context of nutritional science and machine learning. By analyzing and building upon existing work in the field, including but not limited to meal recommendation systems and analyzing nutrition data, this project aims to contribute a practical and innovative solution to the challenge of personalized dietary planning.

Problem Definition and Algorithm

Task Definition

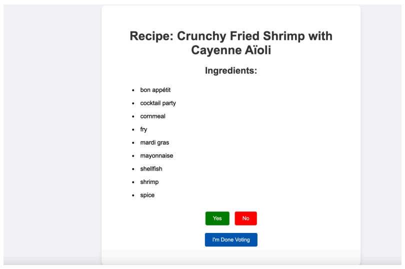
Through "Recipe Explorer", we aspire to empower individuals to make healthier, more informed food choices, thereby fostering a more mindful and health-centric approach to eating. Many other meal-planning apps do not take user preference into account or are rigid in their recipe recommendations. There is a necessity for a flexible, machine-learning-driven tool in the context of existing dietary planning solutions.

The main output of the model is to recommend recipes to a user. The user will use an online macronutrient calculator to select goals for themselves, and the app will make recommendations that fit those goals. As the user interacts with the app by voting on recipes, we integrate that into the ML model to improve its recommendations.

The following features have been implemented:

- Gather user preference data from recipe likes / dislikes (Fig. 1)

- Recommend recipes by dietary fit and user preference model (Fig. 2)
- Visualize and explore related recipes through an interactive graph (Fig. 3)
- A web-based frontend integration to make this tool more user-friendly

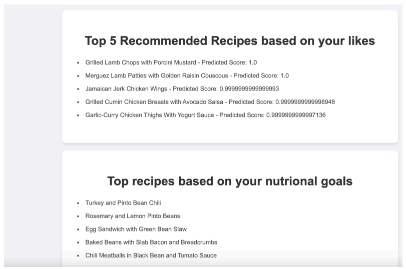


Recipe: Crunchy Fried Shrimp with Cayenne Aïoli

Ingredients:

- bon appétit
- cocktail party
- cornmeal
- fry
- mardi gras
- mayonnaise
- shellfish
- shrimp
- spice

Figure 1



Top 5 Recommended Recipes based on your likes

- Grilled Lamb Chops with Porcini Mustard - Predicted Score: 1.0
- Merguez Lamb Patties with Golden Raisin Couscous - Predicted Score: 1.0
- Jamaican Jerk Chicken Wings - Predicted Score: 0.9999999999999993
- Grilled Cumin Chicken Breasts with Avocado Salsa - Predicted Score: 0.9999999999998948
- Garlic-Curry Chicken Thighs With Yogurt Sauce - Predicted Score: 0.9999999999997136

Top recipes based on your nutrional goals

- Turkey and Pinto Bean Chili
- Rosemary and Lemon Pinto Beans
- Egg Sandwich with Green Bean Slaw
- Baked Beans with Slab Bacon and Breadcrumbs
- Chili Meatballs in Black Bean and Tomato Sauce

Figure 2

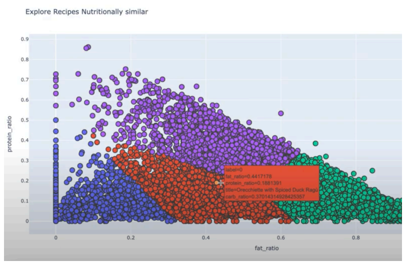


Figure 3

Screenshots of the final product:our ML algorithms integrated into a web-based application to visualize and recommend related recipes

The user interface, as depicted in Figures 1-3 above as well as this video demo, represents the culmination of our efforts to bridge advanced machine learning techniques with user-centric design. Developed using Flask, this web-based application facilitates a seamless interaction between the user and our "Recipe Explorer" system. Users can easily input their dietary preferences, view recommended recipes, and provide feedback through recipe likes or dislikes directly within the interface.This direct engagement not only enriches the dataset with valuable user preference data but also continuously improves the accuracy of our recipe recommendations through machine learning adaptation.

Dataset

We decided on the Epicurious dataset, which is publicly available on Kaggle, to provide the backing for our app. The dataset has over 20,000 recipes, offering a rich source of culinary knowledge. The contents of this dataset is well-formatted and labeled. Key fields include recipe names, ingredients, preparation steps, nutritional values (calories, protein, fat, sodium), user ratings, and various tags categorizing the recipes (e.g., dessert, quick & easy, healthy). The user ratings in the dataset provide a basis for understanding which recipes are well-received on the Epicurious website. We can use this to tailor recommendations not only based on nutritional needs but also on likely user satisfaction by allowing users to vote for their like or dislike of each recipe, changing the probability the algorithm will recommend similar recipes to the user. The various tags (like cuisine type, meal course, and dietary restrictions) help in fine-tuning the recipe recommendations according to user preferences. Given the vast number of recipes, data cleaning was essential to ensure consistency, especially in ingredient names and measurements. We used Python, along with libraries like Pandas for data manipulation and Scikit-learn for machine learning to do the job.

In our post-processing, we eliminated data that seemed erroneous or inappropriate. For instance, we attempted to normalize title formatting, remove cocktail recipes, drop duplicate recipes / columns, and drop recipes with calorie counts well outside the normal range.

As an example, here's a random entry from the dataset:

Column name	Value (columns with value of 0.0 elided)

title	Wild Rice and Chicken or Turkey Salad with Tarragon
rating	4.375
calories	565.0
protein	40.0
fat	27.0
sodium	963.0
bon appétit	1.0
herb	1.0
poultry	1.0
rice	1.0
salad	1.0
turkey	1.0

Table 1. A single recipe from our dataset. All columns with the value 0.0 have been omitted.

In addition to the dataset from Kaggle, we also voted on recipes to train our user preference model. This second dataset consists of 'likes' and 'dislikes' for the recipes in the original dataset, submitted by the project team. We each viewed random recipes and cast votes based on our real preferences. This data is represented by the title of the recipe, and a single value representing the vote: 0.0 for dislike, 1.0 for like.

Algorithm Definition

This project uses different ML algorithms for different features:

1. Recommend a selection of recipes based on dietary fit:
    a. We experimented with two algorithms for this purpose:
        i. K-Means Clustering
            1. Sub-divide our recipes into clusters of related recipes.
            2. Use silhouette score to determine an optimal number of clusters.
            3. Do feature selection to fit the clusters to user preferences. We experimented with macronutrient ratios (relative amounts of calories, carbs, fats, and proteins), raw macronutrient values, and ingredient and theme tags from the full dataset.
            4. Use K-Means to assign recipes to clusters according to their macronutrient ratios
            5. Assign the user to a cluster by passing their preferences into the same K-Means classifier as above.
            6. Return all recipes belonging to the user's cluster.
        ii. Random Tree 'Forest'
            1. Extract a selection of tags from the recipes, to use as labels.
            2. Train a decision tree classifier for each individual tag, using the raw macronutrient values (calories, carbs, fats, proteins, and sodium) as features.

            3. Collect user inputs for each of the feature values.
            4. Run each tree classifier on the inputs, determining which tags apply and which do not.
            5. Using the compiled list of positive/negative tags, return the recipes with matching positive/negative tags.

2. Predict whether a user will like a recipe based on likes / dislikes
    a. Collect user feedback in the form of a list of tuples of recipe title (string) and vote (1.0 for like, 0.0 for dislike).
    b. We experimented with two algorithms for this purpose as well:
        i. Logistic Regression
            1. Perform Logistic Regression on the portion of our dataset with voted items, to identify the most relevant recipe features and train a classifier.
            2. Tune regularization parameters to optimize performance.
            3. Use the Logistic Regression classifier to predict votes for unvoted items in the dataset, in order to predict better recommendations for the user.
        ii. Support Vector Machine (SVM)
            1. Utilize random forest for feature selection, including features that have importance over a defined threshold when training the SVM in the next step.
            2. Apply the SVM algorithm to the subset of our dataset that contains voted items. This will help to identify the most relevant features in recipes and develop a predictive model.
            3. Utilize the trained SVM model to make predictions on unvoted items within the dataset in order to make accurate recommendations for the recipes that a user will like.
3. Visualize recipe graph
    a. Use t-Distributed Stochastic Neighbor Embedding (t-SNE) to project the dataset into two dimensions
    b. Use the K-Means clustering algorithm from (1) above to color the data points representing recipes, highlighting which cluster the user belongs to, and what recipes are in that / other clusters.

Experimental Evaluation

Methodology

Our hypothesis is that the Recipe Explorer tool can provide personalized recipe recommendations that align closely with individual users' nutritional goals and personal preferences.

To evaluate the performance of our Logistic Regression and SVM user preference models, we had to combine the user preference data with the full recipe dataset (see above). We constructed three separate dataframes, one for just the user preference data to use as targets, one for just the recipe features, and a third representing both, which is computed using a simple join. To represent user preferences, we used one column to hold the user_id and another to hold their vote (0.0 for dislike, or 1.0 for like). We initially utilized LOGOCV for evaluations. Under this methodology, groups were based on the user_id within the combined dataframe, which ensured the model would be trained on data from all users except one, then tested on that excluded user. This LOGOCV methodology essentially represented the model being trained on the preferences of various users, then tested on a new user. However, to better mirror the real-world application and enhance our understanding of the model's predictive performance on individual users, we transitioned to using K-Fold CV. This tests the model's ability to learn a single user's preferences and apply them to new, unseen recipes instead of testing the model's ability to learn multiple user's preferences and apply them to a new user. In this K-Fold CV approach, the combined dataframe for each user is independently divided into "K" folds. K is set to 3 as we have a relatively small dataset of indicated preferences for each user (200 - 300 samples). Then, the model is trained on K-1 of these folds and tested on the remaining fold. This process is repeated for each user, then results are aggregated to gain an assessment of the average performance of the model for each individual user.

Average accuracy over cross-validation folds was calculated to serve as a simple scalar metric for model evaluation when comparing performance with future iterations of this model. Because our data regarding user preferences was balanced (each user submitted an equal number of likes and dislikes, either 100 or 150 of each), accuracy served as a valuable metric for estimating the proportion of true results over all cases. Additionally, we calculated the average F1 score over cross-validation folds to provide a more robust metric for evaluating the accuracy of predictions of recipes that a user will like.

We plotted the receiver operating characteristic (ROC) curve and calculated the resulting area under the curve (AUC) to provide an additional measure of performance. Through visualization of the ROC curve and associated AUC calculation, we were able to assess the model's performance in terms of true positive rate and false positive rate across a range of decision thresholds. This provided a more in-depth assessment of the model's performance than average accuracy or F1 score alone.

We calculated and visualized the confusion matrix for the Logistic Regression and SVM user preference models to provide visual representations of the models' performance in terms of actual versus predicted classifications. These confusion matrices provide more details on the models' performance because they help one to understand what types of errors are most prevalent in each model (false positives or false negatives).

Lastly, we created the plot of train versus test error for the Logistic Regression and SVM user preference models to help gain a visual understanding of the tradeoff between bias and variance within each model. Error was computed as 1-accuracy, which served as a valuable metric due to our balanced dataset. If both train and test error were shown to be high through this plot, we could assume that the model suffers from high bias. If train error was shown to be significantly lower than test error through this plot, we could assume that the model suffers from high variance. The results of this plot helped to guide the next steps for improvement of our model.

To evaluate the performance of the K-Means recommendation model, we calculated silhouette scores for various numbers of clusters (3-16) and created a 2D visualization of the clusters after reducing the dataset dimensionality using t-SNE. Prior to reducing data dimensionality with t-SNE, we standardized the data with Sklearn's StandardScaler to account for t-SNE's sensitivity to feature scale. We chose t-SNE over PCA for dimensionality reduction because the relationships between features in our dataset are nonlinear. Dimensionality reduction was necessary to reduce the data dimensions from three to two in order for the visualization to be easier to interpret.

The data used for evaluation was a dataframe consisting of the ratios of carbs, fat, and protein for each recipe. These ratios are necessary for the K-Means recommendation algorithm because recipes are clustered based on the ratios of carbs, fat, and protein within a recipe. For example, recipes with high values for protein and carb ratios and a low ratio for fat ratio would be appropriate to recommend to a user whose macros communicate a desire for a high protein, and high carb, but low fat recipes.

For the K-Means recommendation model, we calculated the silhouette score and plotted the score for various numbers of clusters for two reasons. First, it was necessary to produce a metric such as this to determine the optimal number of clusters for our model in a quantitative way; the number of clusters associated with the silhouette score closest to one was considered the most optimal. Second, the silhouette score associated with the number of clusters used in our model serves as an indication of model performance. A silhouette score is an indication of how similar a point is to its own cluster compared to other clusters. The score can range from -1 to 1, where a higher value is more desirable, indicating that clusters are well-separated and clearly defined. We chose to use the silhouette score to arrive at the optimal number of clusters instead of the elbow method because it makes the decision regarding the optimal number of clusters more meaningful and clear. Additionally, although calculating the silhouette score is more computationally expensive than the elbow method, we can afford it because the size of our dataset is manageable.

Additionally, we created a visualization of the clusters in order to have a way to visually assess the separation of clusters in our K-Means recommendation model. A visualization of this kind could help to understand why the silhouette score for the optimal number of clusters is not closer to 1 through showing if clusters are overlapping or if points are not assigned to the correct clusters.

To evaluate the performance of the Decision Tree Classifier ‘forest’, we employed a much simpler method. Taking inspiration from the analysis of our SVM and Logistic Regression bias and variance, we created a similar train versus test error plot for the ‘forest’, however we used a slightly different method for determining the accuracy. As we trained the classifiers, using various train-test splits, we recorded the score of each classifier - measuring the approximate accuracy of each individual tag label. Afterwards, we multiplied all of the scores together to get an estimation of the accuracy, which stems from an assumption that each tag is completely independent from all of the others. The assumption is likely not true, however it gave us a baseline to work with. As before, the results of this plot gave a benchmark for further improvements to the model, primarily taking the form of early stopping.

As each additional tag in the forest diminished the accuracy somewhat, a few select tags were chosen for their predictive power, or how they improve the accuracy of their corresponding trees. These tags are limited to common ingredients, like chicken, fish, cod, salmon, beef, beef rib, bean, pasta, lettuce, spinach, and bell pepper.

We are currently presenting these results through graphs and written communication of values for average accuracy and average F1 score. The graphs can help one to gain an idea of how the models are performing visually while the average accuracy and average F1 scores can be used for easy comparison with average accuracy and average F1 score for future iterations of this model once created. By including the average accuracy, average F1 scores, ROC curve, AUC-ROC calculation, and confusion matrix visualization, we are able to provide a detailed evaluation of the models.

Results

Our baseline system includes the K-Means model to recommend recipes based on dietary fit and the Logistic Regression model to predict whether a user will like a recipe based on likes / dislikes.

K-Means

The K-Means algorithm was used to subdivide recipes into clusters associated with macronutrient ratios, so we can identify a segment of the recipe space suitable for each user. To start, we used silhouette score to find an ideal number of clusters (4), and used t-SNE to visualize the results of running K-Means with that hyperparameter (see figures 4 and 5).

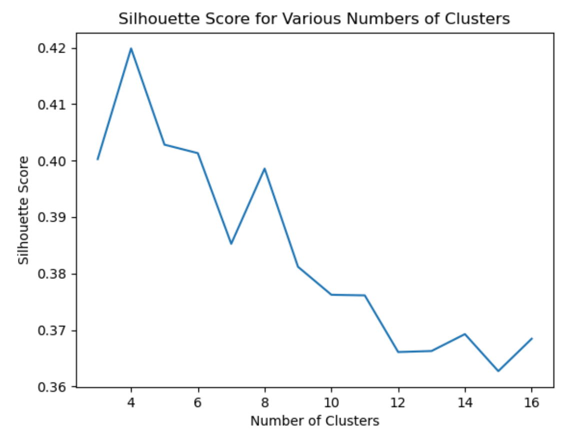


Figure 4. Silhouette scores for various numbers of clusters ranging from 3 to 16 for the K-Means recipe recommendation model.

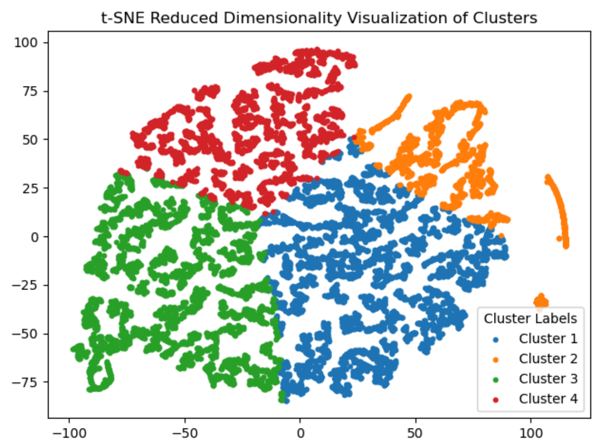


Figure 5. t-SNE reduced dimensionality visualization of clusters for the K-Means recipe recommendation model.

Given the low silhouette score and poor separation between clusters, we explored using additional features from the dataset to train our model. We compared results in four conditions:

1. Just computed macronutrient ratios (as used for figures 4 and 5 above)
2. Raw macronutrient features
3. All recipe features, including raw macronutrient features, ingredients, and tags
4. All recipe features and computed macronutrient ratios

For each condition, we evaluated silhouette scores for a range of cluster sizes:

Condition	Cluster Size					
	2	3	4	5	6	7
1: Macro ratios	0.461	0.401	0.419	0.403	0.401	0.385
2: Macro features	0.929	0.887	0.710	0.666	0.605	0.591
3: All features	0.983	0.940	0.921	0.729	0.663	0.583
4: All features + ratios	0.983	0.940	0.922	0.729	0.674	0.576

Table 2. Silhouette scores for different conditions (feature sets) and cluster sizes

We then visualized the results of K-Means for all clusters with a silhouette score within 90% of the best score for each condition (highlighted in yellow in table 2 above). Figures 6-9 below show sample visualizations from conditions 1-4 with the maximum number of clusters considered. The Visualizations generated with smaller numbers of clusters look very similar.

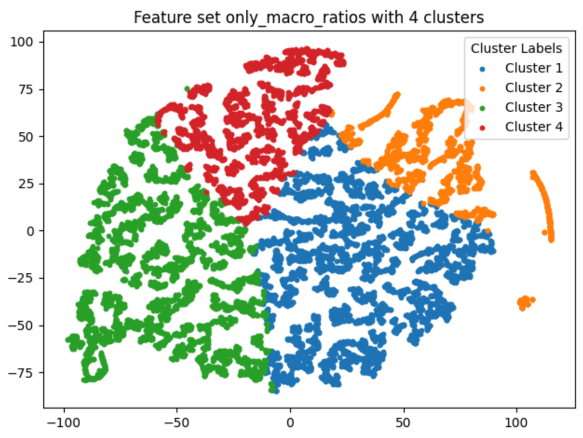


Figure 6. t-SNE projection of clusters built from macro ratios (condition 1, same as above)

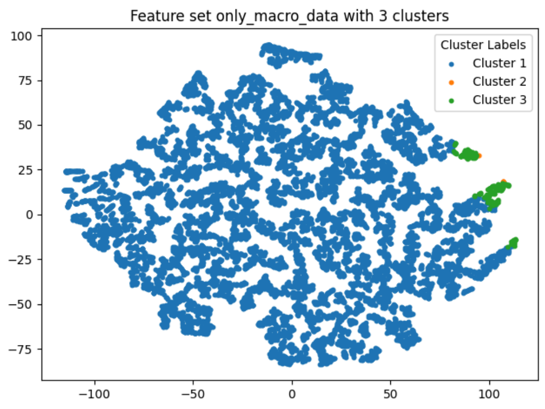


Figure 7. t-SNE projection of clusters built from macro features (condition 2)

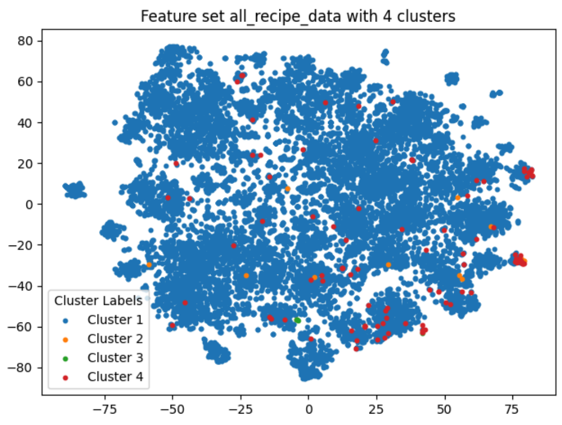


Figure 8. t-SNE projection of clusters built from all recipe features (condition 3)

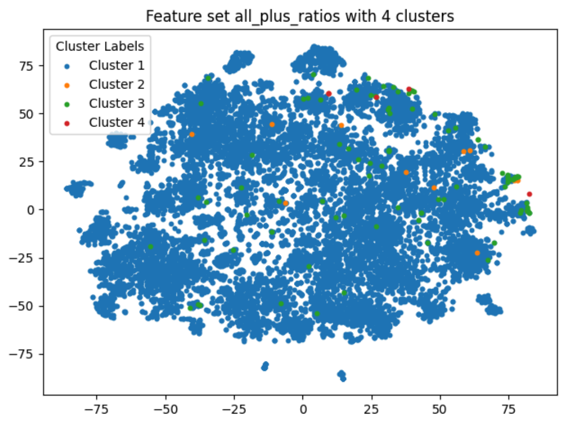


Figure 9. t-SNE projection of clusters built from all recipe features and macro ratios (condition 4)

Decision Tree Classifier Forest

The Decision Tree Classifier set was used to determine the set of selected tags associated with macronutrient and sodium values, which allows us to infer which recipe tags are most relevant from user input, in order to display recipes which match those tags. We used the train vs. test curves (see figures 10 and 11) to guide the selection of a regularization measure of the maximum number of leaf nodes allowed (75). Additionally, each of the curves generated in testing displayed the test data having minimal error at around an 80-20 train-test split.

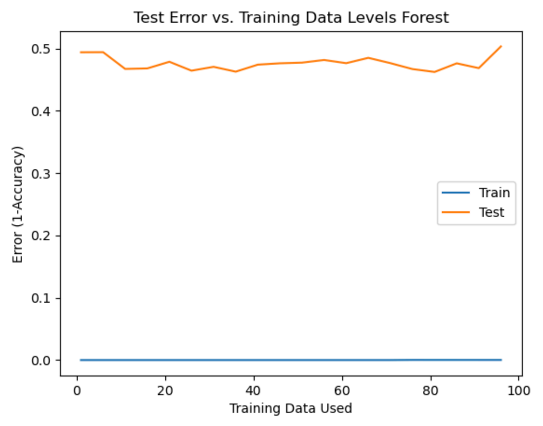


Figure 10. Train vs. Test error for decision tree forest, iteration 1.

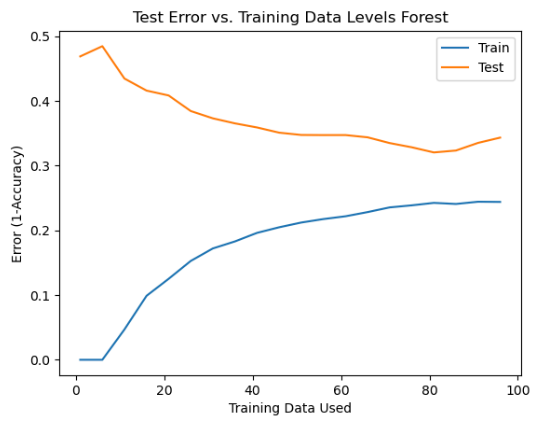


Figure 11. Train vs. Test error for decision tree forest,iteration 2.

For the second model iteration, the testing accuracy landed at around 0.6720 when using the optimal train-test split. Unfortunately, in practice, the regularization seems to make all tags much more difficult to get a positive value for any single tag, let alone multiple.

Logistic Regression

Results from evaluation of the Logistic Regression user preference model can be found below.

For the second iteration of this model, the average accuracy was found to be approximately 0.6275 while the average F1 score was found to be approximately 0.6131. We were also to achieve an ROC AUC of 0.675. ROC curves with AUC and confusion matrices for both iterations of the logistic regression model can be found in figures 12-17 below.

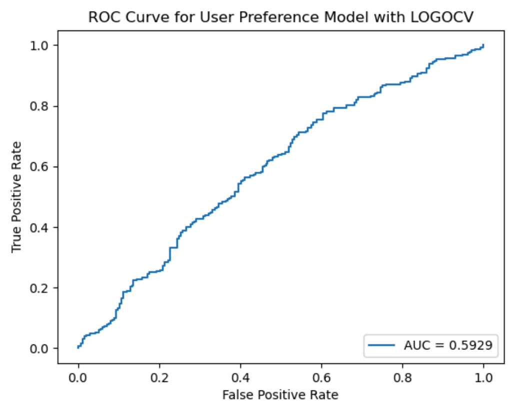


Figure 12. ROC curve for logistic regression user preference model iteration 1.
AUC = 0.5929.

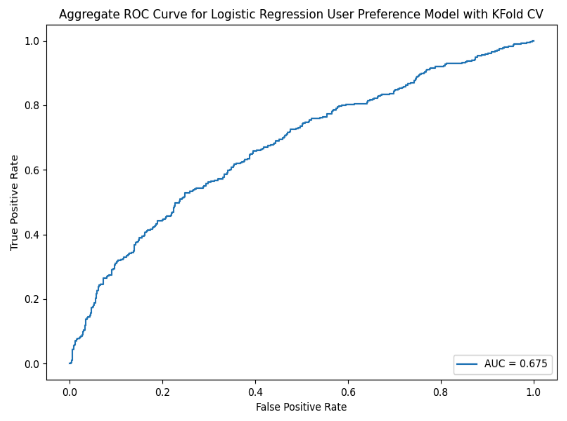


Figure 13. ROC curve for logistic regression user preference model iteration 2.
AUC = 0.675.

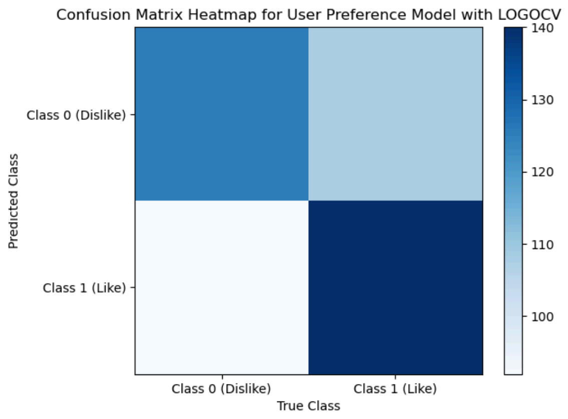


Figure 14. Confusion matrix heatmap for logistic regression user preference model iteration 1.
Values are as follows: [[125 108]
[92 140]]

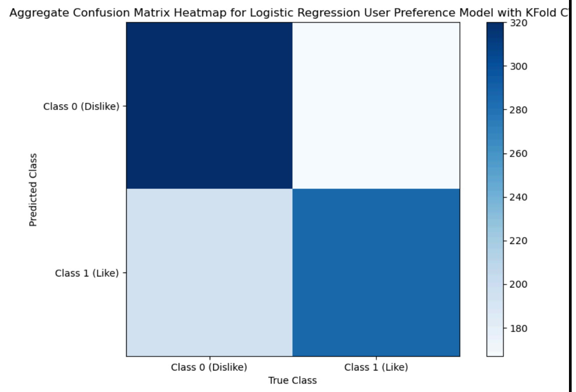


Figure 15. Confusion matrix heatmap for logistic regression user preference model iteration 2.
Values are as follows: [[320 167]
[193 286]]

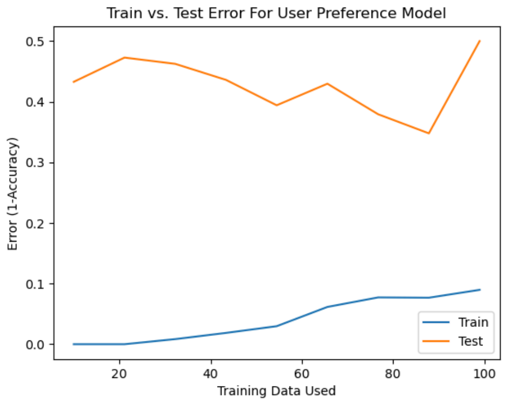


Figure 16. Train vs. Test error for logistic regression user preference model iteration 1.

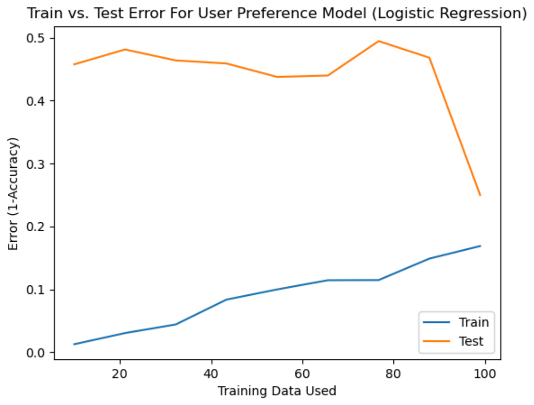


Figure 17. Train vs. Test error for logistic regression user preference model iteration 2.

Using the area under the ROC curve (ROC AUC) as our primary performance metric, we experimented with using Ridge or Lasso regularization, and did a sweep over the regularization coefficient C (20 logistically-spaced values between 0.01 and 10.0). See results in table 3 below. From this we determined the ideal hyperparameter settings are to use Ridge regularization, with C≈1.129. This is very close to what we used to generate our initial results (Ridge regularization, C=1.0).

C =	0.010	0.014	0.021	0.030	0.043	0.062	0.089	0.127	0.183	0.264
Lasso	0.538	0.541	0.541	0.544	0.541	0.536	0.534	0.535	0.541	0.558
Ridge	0.557	0.565	0.574	0.585	0.597	0.610	0.622	0.633	0.643	0.650

C =	0.379	0.546	0.785	1.129	1.624	2.336	3.360	4.833	6.952	10.0
Lasso	0.585	0.610	0.632	0.641	0.645	0.646	0.642	0.641	0.639	0.636
Ridge	0.655	0.656	0.658	0.658	0.658	0.658	0.657	0.656	0.654	0.654

Table 3. ROC AUC scores for different regularizers and coefficient values

Support Vector Machine

Results for the SVM user preference model can be found below.

The average accuracy was found to be approximately 0.6221 while the average F1 score was found to be approximately 0.5929. The ROC curve, confusion matrix, and plot of train versus test error for the SVM user preference model can be found in figures 18-20 below.

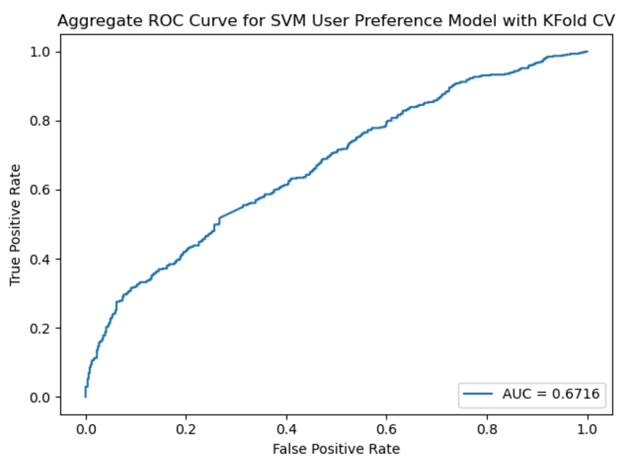


Figure 18. ROC curve for SVM user preference model. AUC = 0.6716.

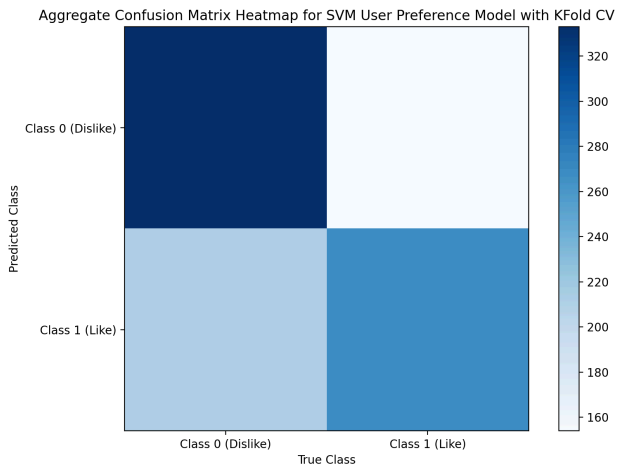


Figure 19. Confusion matrix heatmap for SVM user preference model. Values are as follows: [[333 154]
[212 267]]

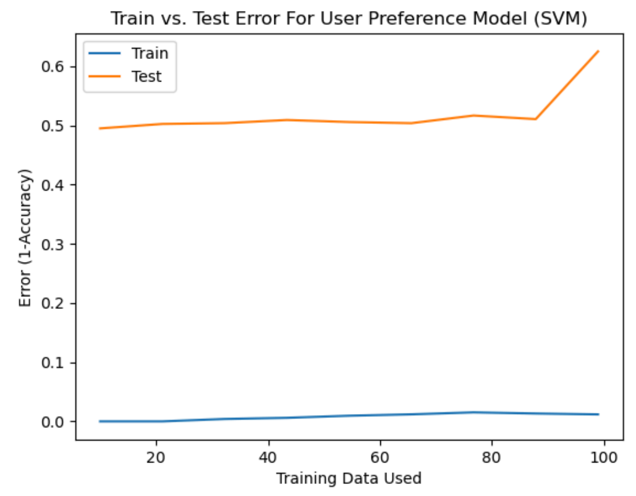
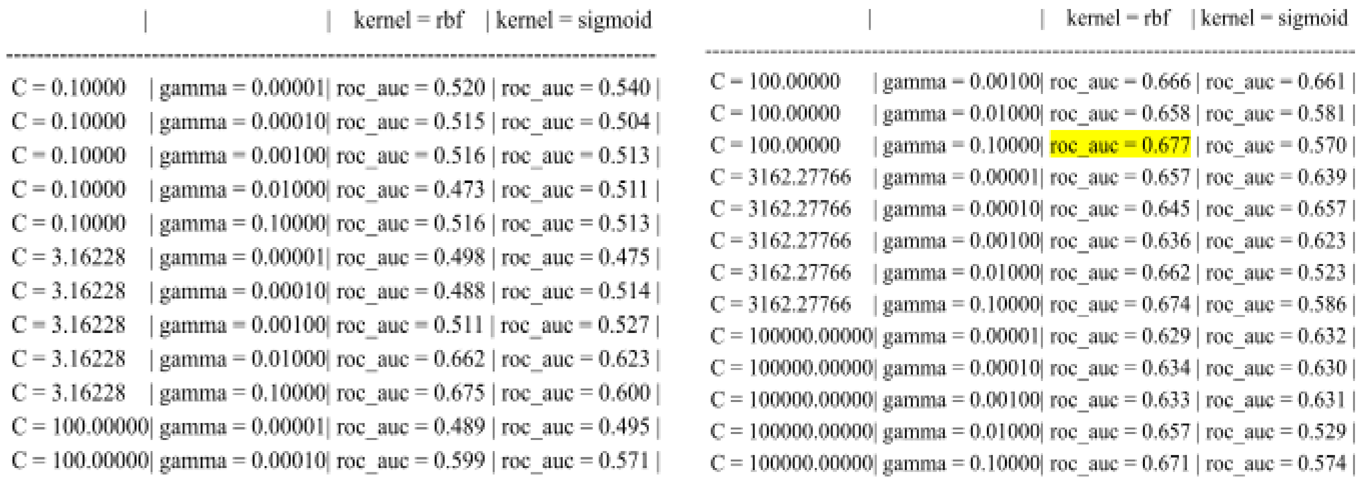


Figure 20. Train vs. Test error for the SVM user preference model.

Using the ROC AUC as our primary performance metric, we experimented with different values of C and gamma as well as different kernels. Initially, regarding kernels, the hyperparameter sweep included the radial basis function (RBF), the polynomial function, and the sigmoid function. However, we found that using the polynomial kernel never resulted in the highest ROC AUC over many runs of the hyperparameter sweep, so it was ultimately dropped from the sweep so that we could experiment with more values of C and gamma without necessitating an extremely long runtime. Ultimately the hyperparameter sweep included the radial basis and sigmoid functions as kernels, 5 values of C logistically spaced between 0.1 and 100,000, and 5 values of gamma logistically spaced between 0.00001 and 0.1. The range for C and gamma was chosen in this way because we found that the ROC AUC was consistently higher when high values for C and low values for gamma were used, so we made the range more extreme in this way to see if the ROC AUC could be improved further. Intuitively, this makes sense as a high value for C would imply less flexibility in the model's margin, and thus, more correct classifications at the risk of overfitting while a low value for gamma would result in a smoother decision boundary at the risk of overfitting. Together, a high value for C and a low value for gamma could strike a balance between overfitting and underfitting. From this, we determined that the ideal hyperparameters are to use RBF as a kernel, a value of 100 for C, and a value of 0.1 for gamma.



```
               |                    |  kernel = rbf   | kernel = sigmoid
--------------------------------------------------------------------------------
C = 0.10000    | gamma = 0.00001| roc_auc = 0.520 | roc_auc = 0.540 |
C = 0.10000    | gamma = 0.00010| roc_auc = 0.515 | roc_auc = 0.504 |
C = 0.10000    | gamma = 0.00100| roc_auc = 0.516 | roc_auc = 0.513 |
C = 0.10000    | gamma = 0.01000| roc_auc = 0.473 | roc_auc = 0.511 |
C = 0.10000    | gamma = 0.10000| roc_auc = 0.516 | roc_auc = 0.513 |
C = 3.16228    | gamma = 0.00001| roc_auc = 0.498 | roc_auc = 0.475 |
C = 3.16228    | gamma = 0.00010| roc_auc = 0.488 | roc_auc = 0.514 |
C = 3.16228    | gamma = 0.00100| roc_auc = 0.511 | roc_auc = 0.527 |
C = 3.16228    | gamma = 0.01000| roc_auc = 0.662 | roc_auc = 0.623 |
C = 3.16228    | gamma = 0.10000| roc_auc = 0.675 | roc_auc = 0.600 |
C = 100.00000| gamma = 0.00001| roc_auc = 0.489 | roc_auc = 0.495 |
C = 100.00000| gamma = 0.00010| roc_auc = 0.599 | roc_auc = 0.571 |
```

```
               |                    |  kernel = rbf   | kernel = sigmoid
--------------------------------------------------------------------------------
C = 100.00000      | gamma = 0.00100| roc_auc = 0.666 | roc_auc = 0.661 |
C = 100.00000      | gamma = 0.01000| roc_auc = 0.658 | roc_auc = 0.581 |
C = 100.00000      | gamma = 0.10000| roc_auc = 0.677 | roc_auc = 0.570 |
C = 3162.27766     | gamma = 0.00001| roc_auc = 0.657 | roc_auc = 0.639 |
C = 3162.27766     | gamma = 0.00010| roc_auc = 0.645 | roc_auc = 0.657 |
C = 3162.27766     | gamma = 0.00100| roc_auc = 0.636 | roc_auc = 0.623 |
C = 3162.27766     | gamma = 0.01000| roc_auc = 0.662 | roc_auc = 0.523 |
C = 3162.27766     | gamma = 0.10000| roc_auc = 0.674 | roc_auc = 0.586 |
C = 100000.00000| gamma = 0.00001| roc_auc = 0.629 | roc_auc = 0.632 |
C = 100000.00000| gamma = 0.00010| roc_auc = 0.634 | roc_auc = 0.630 |
C = 100000.00000| gamma = 0.00100| roc_auc = 0.633 | roc_auc = 0.631 |
C = 100000.00000| gamma = 0.01000| roc_auc = 0.657 | roc_auc = 0.529 |
C = 100000.00000| gamma = 0.10000| roc_auc = 0.671 | roc_auc = 0.574 |
```

Table 4. ROC AUC score for different kernels and values of C and gamma.

Discussion

Our hypothesis of “Recipe Explorer tool can provide personalized recipe recommendations that align closely with individual users' nutritional goals and personal preferences,” is weakly supported, but even with fine tuning our algorithms have significant room for improvement. For example, the accuracy on the user preference model is greater than .5 (random), but only by a small amount (our best accuracy was 0.6275). Since the model performs better than random, we can conclude that our tool provides personalized recommendations, but not ones we can be highly confident in. It’s unclear what the threshold for “good enough” would be in this case, without evaluation with actual users.

Our results can be explained in terms of the algorithms and data we used. Our dataset is moderately sized, with many sparse categorical features, and a fair amount of low-quality data that we had to filter out. We are able to produce a model that recommends recipes to a user better than a random algorithm, partly due to our data and the cleaning process. Our attempt to use K-Means to cluster data resulted in a low silhouette score, indicating the clusters were not very meaningful, even after experiments with feature selection. This is also somewhat supported by the performance of the decision tree forest.

As shown in figure 4, preliminary silhouette score results revealed that four clusters were associated with the highest silhouette score within the range tested (3-16). This indicates that four clusters is optimal for the K-Means recommendation model. However, the silhouette score found for this model with four clusters was only .42, and as shown in figure 5 our clusters don’t seem to correlate with the structure of our data, and have very poor separation. We thought adding more features might help, but as shown in Table 2 and Figures 6-9, the results were not much better. By adding more features, we significantly improved silhouette scores, but at the cost of having dramatically unbalanced clusters. In all other cases, 98% of recipes ended up within a single cluster, which is unhelpful for our purposes. This suggests the high silhouette scores come from isolating a handful of outliers from the rest of the dataset. This is a common result of using sparse categorical features (like the ingredients and tags in our dataset), whose values can’t meaningfully be averaged, in the K-Means algorithm, which is based on averages. This suggests K-Means might be an inappropriate way to segment our dataset, or that there is no clean way to segment our recipe space into discrete, meaningful clusters with similar macronutrient values.

We had similar difficulty clustering recipes using a decision tree classifier forest. Our first attempt (see figure 10) achieved accuracy barely above random chance. By adding a high degree of regularization, we were able to get higher test accuracy, though our maximum testing accuracy was lower (see figure 11). It seems likely that the mixed impact of regularization is in part a consequence of the sparse nature of the data—there are far more recipes without a given tag than there are with that tag, probably leading to a tendency to overrepresent false negative tag assignments. This is evident in practice, where there is a great difficulty in getting the forest of classifiers to recommend even a single tag—which additionally reiterates the difficulty found by K-Means, that the data is not easily separable. This comes from the fact that the tag-having and tag-missing recipes are deeply intermingled in their macronutrient values, with any given tag appearing in many different contexts.

As shown in figure 13, the ROC AUC for the second iteration of the user preference model was found to be 0.675. This represents improvement from the first iteration of the model, where the ROC AUC was found to be 0.5929. Additionally, the average accuracy over folds as well as the average F1 score improved from the first to the second iteration; we achieved an average accuracy over folds of approximately 0.6275 and an average F1 score of approximately 0.6131 in the second iteration, while we achieved an average accuracy across subjects of approximately 0.5682 and an average F1 score of approximately 0.5810 in the first iteration. The performance increase can be attributed to acquiring more data (thus, increasing the model's ability to generalize), adjusting the evaluation methodology to K-Fold from LOGOCV as described above, and determining optimal regularization hyperparameters from the hyperparameter sweep. The confusion matrices for each iteration of the logistic regression user preference mode (figures 14 and 15) provide additional insight into what types of errors the model is making. Interestingly, the main source of error changed from to false positives in the second iteration, while the main source of error had been false negatives in the first iteration. It is unclear exactly why the types of errors the model is making changed in this way, it is possible that this change is due to differences in the recipes that were liked or disliked when creating more data.

As shown in the results section for the SVM user preference model (see figures 18-20), the performance of the SVM was quite similar to that of the second iteration of the logistic regression user preference model with an ROC AUC of 0.6716. The main source of error for this model was also false positives and the average accuracy over folds was similar to the logistic regression user preference model at 0.6221. The average F1 score for the SVM user preference model was slightly worse than the F1 score for the logistic regression user preference model, as it was found to be approximately 0.5929. This implies that the SVM user preference model is slightly worse at predicting the positive class than the logistic regression user preference model although, as stated, performance is quite similar overall. It was a bit surprising that we were not able to achieve better results with the SVM. This surprise is because we performed more rigorous feature selection and thought that the SVM may be better suited for this problem than the logistic regression model. Initially, we attempted to use principal component analysis (PCA) for feature selection, but this resulted in worse metrics overall than choosing features with variance greater than 0 as was done for the logistic regression user preference model. This result was likely due to nonlinearity present in the data. Because PCA was unsuccessful, we then tried to use a random forest to determine which features to use based on feature importance. We experimented with different thresholds of feature importance to use to determine which features to include and found that the highest accuracy and ROC AUC were achieved when a threshold of 0 was used.

Related Work

- Yum-Me: A personalized Nutrient-Based Meal Recommender System

    - Yum-Me provides a quiz-like UI to gather user food preferences then uses those results to gather nutritionally appropriate food options.
    - Many other food recommendation systems learn user preferences over time through food logging or ratings of recipes, so Yum-Me attempts to learn these preferences more rapidly through the quiz interface. Similarly, we would like to provide a quiz-based interface to gather these user preferences more rapidly compared to learning them over time. However, our quiz will differ from Yum-Me's; we would like to design a series of simple inputs where users can enter data like their goal macronutrients and types of foods they like or avoid (for example, a user could specify that they enjoy Mediterranean food and are allergic to dairy), but we do not propose to provide a series of images of food and prompt the user to select appealing images from that series as Yum-Me does.
    - Yum-Me also provides a method for food image analysis, which is not one of our goals at this time. Our model will be based on the nutritional content of the foods that are used in a recipe rather than visual analysis of photos of recipes.
    - Based on the quiz responses, Yum-Me generates meal recommendations by pulling from a database to match the user's preferences. This differs from our proposed model in its method of evaluating and ranking meals. Instead, we propose to utilize a more straightforward scalar value approach, which is likely to be more scalable to larger datasets of recipes as less computational power would be needed to rank recipes in our approach as it lacks the visual analysis component. Our model may also return recipes that are more directly aligned with the user's nutritional goals because they will be the main focus of recommendation.
- Eating Healthier: Exploring Nutrition Information for Healthier Recipe Recommendation (NutRec)
    - NutRec recommends recipes based on ingredients that the user defines. It does so through creating healthy pseudo-recipes given ingredients, then uses those pseudo-recipes to find similar, real recipes in datasets from Allrecipes and Yummly.
    - This is similar to our proposed model because nutritional values are directly incorporated into the recommendation process, but differs in various ways. Namely, we do not propose to create pseudo-recipes. Instead, we propose to use users' nutrition goals and preferences to gather actual recipes more directly through employing machine learning to find recipes that align with users' nutritional goals rather than using machine learning to find recipes that are comparable to generated pseudo-recipes.
- Healthy Recipe Recommendation using Nutrition and Ratings Models
    - Similarly to our proposed project, the system described in this paper is designed to recommend healthy and appealing recipes based on user preferences. However, this system does not allow a user to explicitly define their nutritional goals in terms of macronutrients. This is a large part of the goal for our project, as we want users to be able to customize the recommendation system to fit their needs in terms of weight goals and physical activity.
    - The system described in this paper utilizes linear regression to understand the nutritional content of a recipe given its ingredients. This differs from our proposed model because we propose to instead use dimension reduction and keyword analysis to reduce recipe relevance to a single scalar value representing the recipe's relevance to a user's goals.
    - An additional similarity between this system and our proposed model is the presence of a ratings model. Using a graph neural network, the described system models user rating scores to predict how much a user will like a given recipe. We propose to use logistic

regression for this task instead because we expect that it will be more simple, interpretable, and less computationally expensive.

Code and Dataset

This project uses the Kaggle Epicurious dataset, available here.
The source code for this project is hosted on GitHub here.
The cleaned version of our dataset (with cocktails removed, for example) can be found in our source repository here.

Conclusion

In this paper, we introduced "Recipe Explorer," a novel tool designed to assist individuals in navigating dietary choices through personalized recipe recommendations. By providing a tool that simplifies the complex decision of what to eat, and integrating it with a simple user-interface, we not only make healthy eating more accessible but also more enjoyable.

Throughout the project, we explored the integration of various machine learning techniques including K-Means clustering, decision tree forrests, logistic regression, and support vector machines to develop a system that not only suggests recipes based on dietary compatibility but also adapts to user feedback through preference modeling.

Our experimental evaluations highlighted challenges such as the need for algorithmic tuning and the limitations of certain models in capturing the nuances of user preferences. The insights gained from silhouette scores, ROC curves, and confusion matrices guided our iterative improvements, leading to refined models that better understand and predict user likes and dislikes.

Building the final web-based application allowed us to make our algorithms more accessible to the general public, the target audience that this tool is meant for. While building the app, we had to make several design choices like figuring out which algorithms should be implemented in the app. Our thorough analysis of each of the algorithms helped us decide that logistic regression and decision trees provided the most accurate results, so we have integrated those into our product.

As we move forward, we plan to enhance the accuracy and efficiency of our recommendations by incorporating more sophisticated data processing techniques and exploring alternative machine learning models, such as different types of neural networks. Additionally, we aim to expand our dataset to include a wider variety of recipes and user interactions, which will help improve the system's ability to learn and adapt to diverse dietary needs.

Bibliography

Chen, Meng, et al. “Eating Healthier: Exploring Nutrition Information for Healthier Recipe Recommendation.” Information Processing &amp; Management, vol. 57, no. 6, Elsevier BV, Nov. 2020, p. 102051. Crossref, doi:10.1016/j.ipm.2019.05.012.

F., Pedregosa. “Scikit-learn: Machine Learning in Python.” *Journal of Machine Learning Research*, vol. 12, 2011, pp. 2825-2830.

Géron, Aurélien. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd ed., O'Reilly, 2022.

Harris, Charles R., et al. "Array Programming with NumPy." Nature, vol. 585, no. 7825, Springer Science and Business Media LLC, 16 Sept. 2020, pp. 357–362. Crossref, doi:10.1038/s41586-020-2649-2.

Laing, Brian Yoshio, et al. "Effectiveness of a Smartphone Application for Weight Loss Compared With Usual Care in Overweight Primary Care Patients." Annals of Internal Medicine, vol. 161, no. 10_Supplement, American College of Physicians, 18 Nov. 2014, p. S5. Crossref, doi:10.7326/m13-3005.

Martínez-Garmendia, Josué. "Machine Learning for Product Choice Prediction." Journal of Marketing Analytics, Springer Science and Business Media LLC, 4 Apr. 2023. Crossref, doi:10.1057/s41270-023-00217-7.

Tang, Yew Siang et al. "Healthy Recipe Recommendation using Nutrition and Ratings Models." (2019).

The pandas development team. Pandas-Dev/Pandas: Pandas. v2.2.0, Zenodo, 2024, doi:10.5281/ZENODO.3509134.

Tian, Haoye, et al. "A Music Recommendation System Based on Logistic Regression and eXtreme Gradient Boosting." 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, July 2019. Crossref, doi:10.1109/ijcnn.2019.8852094.

Yang, Longqi, et al. "Yum-Me." ACM Transactions on Information Systems, vol. 36, no. 1, Association for Computing Machinery (ACM), 17 July 2017, pp. 1–31. Crossref, doi:10.1145/3072614.