Predicting and Analyzing Recipe Data

Chaim Nechamkin, Xiaolin Li, Adalie Palma

Introduction

For this project, we will create models to predict ratings of recipes from Food.com. We will also analyze the recipes to discover how well these recipes can be grouped by keyword and what information from a recipe is helpful to predict a recipe's rating. A recipe's rating is a signal of how popular a recipe is, so understanding what kinds of recipes are rated higher, which can be analyzed using keywords, will give us insight into which recipes are more popular.

Using logistic regression, tree based models and neural networks, we found that we could predict positive and negative ratings with 73% accuracy. Using KMeans and agglomerative clustering, we found that recipe keywords are not a defining characteristic of a recipe's rating.

Related Work

Classification of Foods based on Ingredients (Guria et al.)[2]

A study that uses representations of recipes to predict the cuisine. They created representations from the list of ingredients, using Mutual Information and Tf-idf, among others, to create vectors for use in classification. In this project, we will represent recipes with embeddings of the recipe instructions and other columns if needed, not just the ingredients. We will not be doing classification, instead we will predict the quality of the recipe, using the rating as a proxy for quality.

Generating Personalized Recipes from Historical User Preferences (Majumder et al.)[3]

This work is a design for a recipe generator with one marked improvement. They took the reviews that a user had written into account when generating the recipes. This led to a marked improvement in the generated recipes. This is the only source that we have found that tried to incorporate the quality of a recipe into account for modeling. However, they only used a rating for that specific user, but in our project we will be using the rating to predict the average rating, not just what a specific user might rate it.

Recipe1M+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images (Marín et al.)[4]

This work is about a dataset of over 1 million recipes. What is relevant for our project from this work is that they spend a lot of time on images of recipes, creating embeddings from images and even creating a recipe generator that uses an image as input. We saw in this work that the mages of online recipes are very connected to the recipe itself

Data Source

The datasets were collected from Food.com and can be downloaded from Kaggle[1], and they are open source. The recipes dataset contains 522,517 recipes from 312 different categories and reviews dataset contains 1,401,982 reviews from 271,907 different users. Each file comes in two versions, csv and parquet. (fig. 1)

Figure 1

Dataset	Size	Shape	Description
recipes.csv recipes.parquet	704.2 MB 178.7 MB	522517, 28	It includes information of each recipe like name, description, keywords, cooking times, servings, ingredients, nutrition, instructions, and aggregated rating.
reviews.csv reviews.parquet	496.1 MB 173.8 MB	1401982, 8	Each row represents a unique review and rating from a reader on the recipe.

The ratings are heavily imbalanced (fig. 2):

Figure 2

Rating	Null	1	1.5	2	2.5	3	3.5	4	4.5	5
Count	253,223	1,677	76	2,049	673	9,166	3,978	42,829	34,330	174,516

Data Preprocessing

In the recipes dataset, we convert the string data into numerical data. For example, we converted the time from PT24H into 1440 minutes with filling nulls as 0. We also decide to convert the aggregated rating into binary values to address the heavy imbalance present in the data. We decided to label ratings 4 and above as positive and less than 4 as negative. convert the missing value and value that is less than 4 as 0 and value from 4 to 5 as 1. The recipes with missing values for rating are the ones that have not been rated yet. We decided that they should be considered as negative reviews. This is based on Meijerink and Schoenmakers[5], as they show that people are adverse to writing negative reviews. Also, Shi et al.[6] point out that lack of rating or review can many times be an indication of lack of interest. While they were referring to recommendations, a similar logic applies to understanding if a recipe is good or bad, as people have a general understanding of what makes a recipe good or bad, and if it doesn't look promising they will probably not try it. In addition, when trying to rate a recipe, a major part of the rating is its appeal, and if it is not appealing enough to try it or rate it, that may indicate that it is not a good recipe.

Feature Engineering

We created different features for each model.

1. Logistic Regression:

We created summary statistics of the non-numeric columns, such as number of steps, images, ingredients and keywords, and length, in words, of name and description. We then log-transformed all of the features and scaled them using StandardScaler.

2. XGB and Decision Tree classifier:

We expanded the ‘Keywords’ column into 6 useful columns; Cuisine, Free of, High in, Low in, Is Healthy and Duration, and filling missing values with 0 or specific values as needed. We used both textual and numerical characteristics in our models. Aspects like "Calories," "Duration," "ReviewCount," and "DescriptionWordCount" were among the numerical characteristics. These features gave each recipe numerical data, including nutritional value, cooking time, and popularity determined by the quantity of reviews.

We also included text characteristics that were taken from the names, classifications, and descriptions of recipes. Qualitative details about the recipes, such as ingredients, preparation techniques, and cultural connotations, were recorded by these textual characteristics, which included "RecipeInstructions," "Cuisine," "Keywords," and "RecipeCategory." Our algorithms were able to take into account a wide variety of elements impacting the overall rating of recipes by using both text and numerical characteristics, which resulted in more reliable forecasts. We performed tokenization, punctuation and stopword removal, and stemming as part of our preprocessing steps for the text features. The preprocessed text data was then concatenated into strings in order to mix it with numerical characteristics.

3. Neural Network:

We used most of the features from the logistic regression model as summary features. We augmented them with encodings created from the name, description, recipe steps, and images columns. We also used the ingredient and keywords columns.

4. Topic Modeling:

We used the ‘RecipeCategory’ and ‘Keywords’ columns as features for topic modeling. We performed tokenization, punctuation and stopword removal as part of preprocessing then vectorized the tokens using TF-IDF.

5. Clustering:

We used a mix of numeric and text features for clustering. The summary statistics created for the Logistic Regression model were the numeric features while encodings created from the name and keywords columns were the text features.

Please see Appendix A for a full list of features created and used for each model.

Supervised Learning

Methods

Evaluation Method

We chose accuracy as our main metric of evaluation (fig. 3). We are using the natural thresholds which are .5 for all of our models except for the neural network which is 0. Our labels are fairly balanced (48% positive) so simple accuracy is a good measure of how good the models are. While recall and precision are good metrics for many classification problems, we did not use them in our final comparison because they are situation dependent, requiring one to specify which type of error, type 1 or type 2 is the worst one for a specific application.

Logistic Regression

Logistic regression is a simple classifier that we used to create a baseline. In keeping with that idea, we used only simple features that we dubbed “summary” features. These features describe the text features in a basic way. We also used the number of images, the cuisine, and times for making the recipe. We left out the remaining feature for our more thorough modeling using more complicated models, which we then compared to this baseline.

XGBoost and Decision Tree classifier

XGBoost classifier is a high performance and efficient algorithm to handle large amounts of data with many features. It provides built-in regularization techniques to prevent overfitting. It helps us to control the model complexity and improve generalization to unseen data. Decision tree classifier is another useful algorithm that can handle imbalance data. Similar to XGBoost, both are capable of capturing non-linear relationships between features and the target variable.

First, we are comparing the performance of two models: XGBoost and Decision Tree. To ensure a fair comparison, we are utilizing 5-fold cross-validation and report the mean accuracy along with the standard deviation for each model. Additionally, we will also report precision, recall, and F1-score to gain insights into the performance of our models across different aspects such as true positives, false positives, and false negatives. (Appendix B)

Based on the overall summary of results, we observe that XGBoost outperforms Decision Tree in terms of mean accuracy. Additionally, XGBoost exhibits lower variability as indicated by its lower standard deviation. Then we evaluate the XGBoost model with different parameters such as n_estimators, max_depth, and learning_rate to pick the best model. We observed a higher accuracy score with higher n_estimators, but it led to longer training times. Although the adjustment on hyperparameters slightly improved the model performance (around 0.3%). But overall, the accuracy rate is still too low with 60% accuracy.

The reason why TF-IDF is not getting better results in this approach is because it relies heavily on the information content of the text data. If the text features in the dataset are not

sufficiently informative or relevant to the target variable (AggregatedRating), the model may struggle to extract meaningful patterns and make accurate predictions. It's possible that the text features such as RecipeInstructions, Cuisine, Keywords, and RecipeCategory do not adequately capture the factors that influence the aggregated rating of recipes. Another possibility is that the TF-IDF model is overwhelmed by the sheer number of features after vectorization. In this case, the model may suffer from the curse of dimensionality, where the high-dimensional feature space makes it difficult for the model to discern meaningful patterns from noise. Additionally, having too many features can lead to overfitting, where the model learns to memorize the training data instead of generalizing well to unseen data.

Neural network

The dataset contains five text features and one image feature. We attempted to use these features in our predictions by creating embeddings and encodings of the text and encoding of the images. The best way to use encodings is in neural networks, as that is the natural way of using encodings. Also, encodings are high dimensional, and we used four different encodings in this project. Two of the features were simple lists of items, the keywords and ingredients features. Since they are just a bag of items, with no structure, the best way of using them is to create embeddings from them as in a word2vec model. For the other text data, description and steps, we used a BERT transformer model to convert them into encodings, as the structure for these columns means a lot, as they are basically paragraphs of text. In the case of the names it was also decided to use the transformer model, as it can help encode a more precise meaning of the name.

For the images, there are too many to download all of them, so we used the first image from each recipe, as we assume that the first image is meant to be the most representative of the recipe. Also, since most of the negative recipes are ones without any reviews, it seems that one factor in the rating is the first image, as people ‘judge a recipe by its cover’ i.e. this is one's first impression of the recipe and probably plays a large factor in whether the recipe is ever tried, and if a recipe is never tried, it will not be rated. We used sentenceTransformers library to create the encodings because this library is one actually meant to create encodings, as opposed to using models created for other uses which would have to be repurposed and fine-tuned. We also chose the sentenceTransformers library because it also can be used to create encodings images. We did try other transformers such as Distillbert and Deberta, but saw no improvement. We tried several convolutional networks for images as well, but they also had no advantage so we stuck with sentenceTransformers which also helped streamline our workflow.

We created a neural network model with word embeddings using the EmbeddingBag module from pytorch. There are two separate EmbeddingBags created, one for the ingredients and one for the keywords. We combined the output from the embeddings with the summary features and all of the encodings to create one large tensor which we fed into a multi layer perceptron.

We used a standard pytorch workflow. The training was done on a P100 gpu. The optimizer used is AdamW, the current state of the art optimizer as it converges faster than other methods. We used a rather large batch size of 1024 because the data is extremely noisy, so we needed enough samples per batch, otherwise the models would fail to converge.

For hyperparameter tuning we utilized a grid search over the size of the hidden layers, number of hidden layers, and dimension of the embeddings. We also adjusted the learning rate based on the results and convergence, but we did not use grid search as it is too expensive to run a grid search over four parameters.

Creating encodings for the text and image features took about 1.2 hours and downloading the images took over six hours. Creating the encodings took a long time as well as being a major use of resources, as the encodings were created using a gpu. One major tradeoff in this model was the choice to use only the first image of the images. If we would have used all of the images, downloading them all would have taken over 24 hours and used too much memory. We may have gotten better results if we did fine tuning on preexisting encoding models, however initial results were worse, and used too much resources.

Supervised Evaluation

In figure 3 we show the mean accuracy and standard deviation of the accuracy from our cross validation of our models. In interest of completeness we show the confusion matrices for our threshold (fig. 4).

The neural network is the best model, but only by .017. Its standard deviation is also much less than the other models, so it is more consistent and less influenced by the split of the data. While the score for the neural network is within one standard deviation of the logistic regression model, the logistic regression model is much less consistent. We will use the neural network as our best model, but will caution that since it is not much better than the others, in many instances it may be a good idea to use a simpler model.

Figure 3

Model	Mean Accuracy	Standard Deviation of Accuracy
Logistic Regression	0.711985	0.025384
XGBoost	0.601814	0.001843
Neural Network	0.729215	0.001828

Figure 4

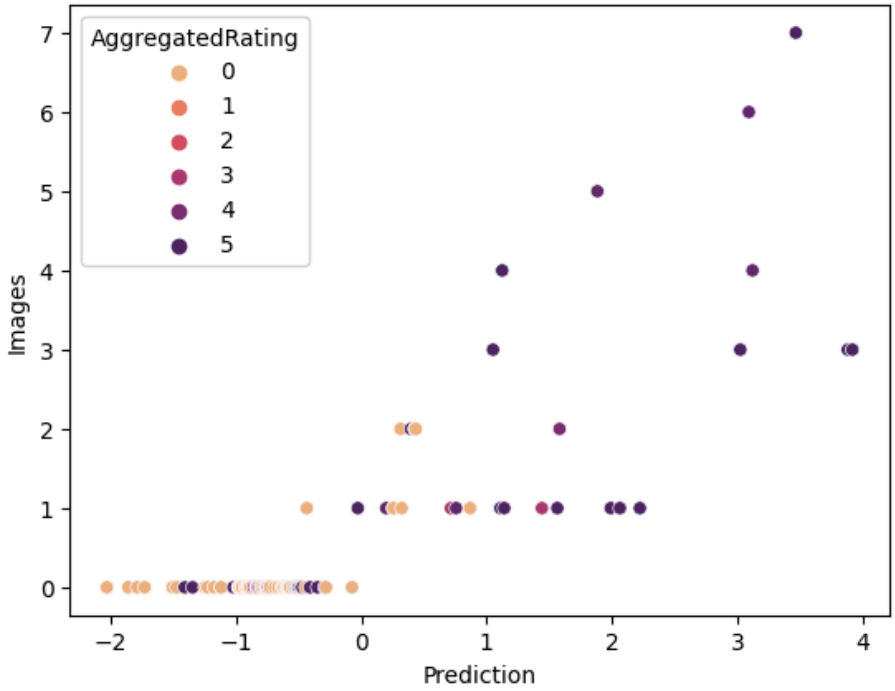
Confusion Matrices	True Label	Prediction

		Negative	Positive
Logistic Regression	Negative	45%	7%
	Positive	22%	26%
XGBoost	Negative	37%	15%
	Positive	25%	23%
Neural Network	Negative	45%	7%
	Positive	20%	28%

*numbers may not add up to 100% due to rounding

Failure Analysis

Figure 5

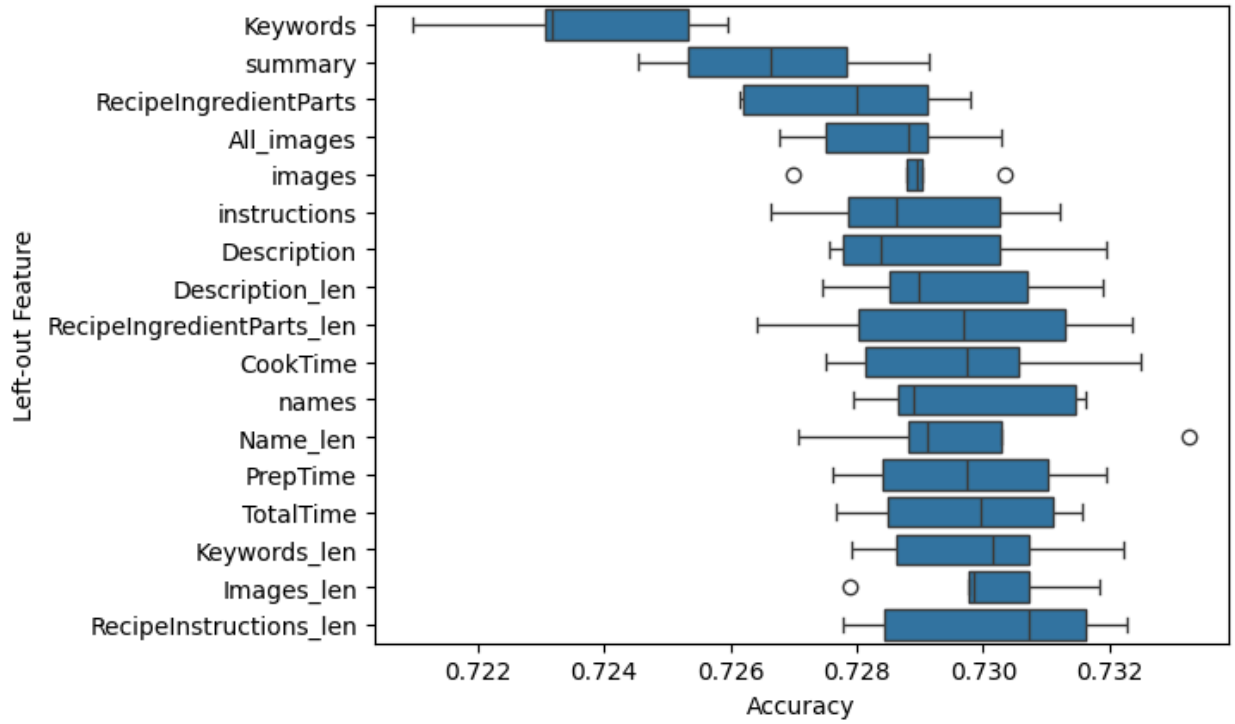


One form of failure in the results is that we get a very high rate of type II errors, that is, our models consistently predict false negatives almost as often as they predict true positives. One possible cause is that many (about 30%) positive reviews state the reviewer purposely changed the recipe. This means that the rating given is not the rating for the recipe that is in our data, rather a very similar recipe. It is possible that our models are predicting the rating of the recipe if it were actually followed, and that the correct label is actually negative, but the rating is really for a different recipe, for which the correct label is positive.

Another form of failure obvious in our results is that the prediction is very dependent on the number of images (fig. 5). Recipes with no images have a very low chance of being predicted above zero, the threshold, and almost all of the recipes with more than one image are predicted above zero. However, as noted in our feature ablation analysis, the number of images is actually the least significant feature. It seems that the number of images has the same causes as the rating, which does not sound intuitive. Since the number of images should not be correlated with the taste of the recipe, this is a clear indication that taste is not a significant factor in the rating.

Feature Ablation

Figure 6



*Features ordered by mean accuracy

We did feature ablation using a leave-one-out strategy (fig. 6). We also tried leaving out all of the summary features. We also tried leaving out both image related features, the number of images and the encodings of the images, because, as we noted in our failure analysis, the model consistently gives low ratings to recipes with no images and high ratings to those with images. We used 5-fold cross validation, ensuring that the same set of folds used for each feature.We can see that the keywords feature is the one that makes the most difference, but even that is minute; about .3% from the closest and .5% from the average The number of images has one of the smallest effects, and the image encodings, while having one of the largest effects, is still insignificant.

Unsupervised Learning

Methods

An initial challenge was dealing with the size and class imbalance of our dataset. The limited hardware available to us made it difficult to process our full dataset. Meanwhile, approximately half of the recipes in our dataset didn't have a rating, and the majority of the recipes with a rating were rated 5 stars, with only a couple thousand rated 1 or 2 stars. Both of these challenges were addressed by taking a subset of our dataset to perform our analysis on, rather than using the full dataset. We worked with all 1k of the recipes with a 1 star rating and took a 2k sample of the recipes with a 5 star rating, resulting in a more balanced dataset that was small enough to easily process on our hardware.

To explore the recipes dataset, we attempted to find clusters and topics for recipes of different ratings. Our hypothesis was that recipes of same ratings would have similar attributes while recipes of different ratings would have different attributes from each other. If our hypothesis was correct, a good clustering would cluster recipes of the same rating together while clustering recipes of different ratings in separate clusters. Similarly, a good topic model would find distinct topics for recipes of different ratings.

We approached topic modeling first. For topic modeling, we used Latent Dirichlet Analysis (LDA) and Non-negative Matrix Factorization (NMF) to identify three topics for each subset of recipes with a specific rating. We chose LDA because it will learn from the recipes dataset a mixture of topics, where each topic is a probability distribution over words. This allows us to generate a group of words that is likely to represent the common ideas, or topics, that represent each rating group in the recipes dataset. We chose NMF because it uses matrix factorization to learn a weight for each word being associated with a topic. This allows us to identify the most likely words associated with each topic.

For both LDA and NMF approaches, we relied solely on the text attributes in our recipes dataset. We used a TF-IDF vectorizer to vectorize the text attributes then used these vectors as the input for LDA and NMF to learn from.

Our initial hypothesis on which text attributes to use was that the Keywords and RecipeCategory attributes were good summaries of a recipe's content, which would make them useful to identify the common topics in our dataset. However, we quickly realized that there were several keywords and recipe categories that were prominent in our dataset, leading to many topics that were similar. To verify that certain words were dominating the dataset, we created a wordcloud for the keywords and a wordcloud for the recipe categories. These wordclouds were created using all of the recipes in our dataset, excluding 17k recipes that did not have any keywords. (All recipes had a category.)

From these wordclouds, we identified a total of 15 keywords, such as Hours, Low, Mins, Easy, Healthy, to consider as stop words, or words to exclude from our TF-IDF features for the

LDA and NMF models. However, we noted that there are still certain keywords that show up frequently for all recipes, regardless of the rating, such as Friendly, Kid, and Free. After using these keywords as stop words, in addition to default English stop words, to preprocess the Keywords and RecipeCategory for our topic models, the resulting topics had more variety.

Our objective for clustering is to identify a clustering that groups recipes of the same rating in the same cluster. Because of the huge class imbalance in the number of recipes in our dataset for each rating, we attempted to cluster a subset of recipes with a rating of 5 against all recipes with a rating of less than or equal to 2. If our model was ideal, it would be able to cluster the recipes with a rating of 5 into one cluster, and all other recipes into another cluster, resulting in 2 clusters.

To find clusters for the recipes, we used KMeans and Agglomerative clustering. We chose KMeans clustering because we already know the number of ideal clusters. Additionally, KMeans clustering doesn't assume a tree structure in the data like agglomerative clustering does. We chose agglomerative clustering because it allows for more complex cluster shapes and is more robust to outliers than KMeans clustering.

We used T-SNE visualizations to help guide feature selection for clustering. As mentioned above, we used a subset of our full dataset so that there was an approximately equal number of recipes with a rating of 5 and recipes with a rating below 2. These were the two groups we attempted to cluster into. Because our objective was to find 2 clusters, we used T-SNE on the extracted features to identify 2 components, then plotted these components, with binary color coding based on their true rating group: 5 or not.

We ended up using a selection of existing numerical attributes from the recipes dataset and combining these with embeddings of text attributes to create our features. The existing numerical attributes were columns like total time for a recipe or the number of reviews. We used the columns Name and Keywords for the text attributes. Leveraging our previous work, we reused the stop words we used in topic modeling and filtered these words out from the Keywords column before creating the features we used for clustering. After filtering out stop words, we got embeddings of the text attributes using an open source Word2Vec model and Sentence Transformer model.

Unsupervised Evaluation

For topic modeling, LDA and NMF had comparable performance and identified similar topics. As mentioned above, several keywords and categories were filtered out from the words used for topic modeling. Wordclouds, as seen in (fig. 7) and Appendix C were created to visualize the most common words in the dataset, allowing us to easily identify which words to select as stop words and filter them out for topic modeling.

A selection of topics produced by our models can be seen in (fig. 8). A full list of topics can be found in Appendix D.

Figure 7

Figure 8

Rating	LDA Topics	NMF Topics
5	Asian, Rice, Berries Breads, Chicken, Poultry High, Pork, Breakfast, Brunch	Chicken, Poultry, Breast Large groups, Cookie, Brownie Kid friendly, European
1	Cookie, Brownie, Free Friendly, Kid, European Breads, Chicken Poultry	Cookie, Brownie, Bar Quick Breads, Breakfast Poultry, Chicken breast

For topic modeling, we used coherence as our evaluation metric. The coherence of a topic was measured by the average cosine similarity between Word2Vec embeddings of the words that make up a topic. The coherence of a topic model was measured by the average coherence of the model's identified topics. Coherence allows us to measure whether the topic model is able to identify semantically reasonable topics or whether it is merely grouping random words together as a "topic". A higher value of coherence indicates more semantically meaningful topics, which is desirable. The coherence of each topic model is listed in (fig. 9).

Figure 9

Rating	LDA Coherence	NMF Coherence
5	0.1700	0.1394
4	0.1455	0.1251
3	0.1328	0.1389
2	0.1908	0.1434
1	0.1458	0.1444
No rating	0.1609	0.1652
AVG	0.1576	0.1427

Across all ratings, LDA and NMF had similar coherence scores, always within 0.05 of each other. The coherence scores for LDA have a slight improvement over NMF for the majority of the ratings, although NMF achieved better coherence scores for recipes with a rating of 3 and recipes with no rating.

For clustering, we used T-SNE visualizations to help with feature selection. We used T-SNE because the manifold learning allows us to have 2-dimensional data to easily plot for visualizations yet still retains local distances between neighbors. If a set of features, when visualized into 2 components using T-SNE, results in distinct and separated groups, the set of features would be good for clustering. We visualized multiple combinations of attributes from our recipes dataset using T-SNE, however none resulted in clearly clustering the recipes by rating, as seen in (fig. 10).

Figure 10

When using only embeddings of the textual features, there was no visible separation into 2 components and the recipes of different ratings are all mapped in approximately the same area. This result was the same for embeddings from both Word2Vec and Sentence Transformer models.

When using only numerical features, the resulting T-SNE visualization shows some areas where there is separation between the differently colored dots, representing recipes of different ratings. However, the majority of the dots still overlap and there is no clear distinction between clusters.

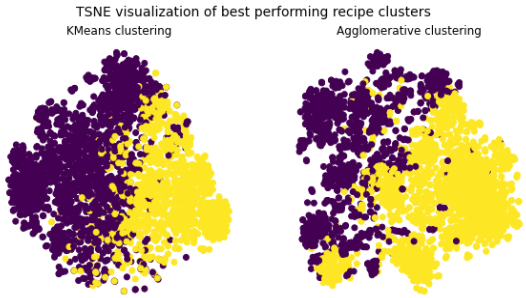
When using textual features and pre-existing numerical features, there was no clear visual improvement compared to using only numerical features. The shape of the visualized components varied depending on whether we used Word2Vec or Sentence Transformer embeddings, but there was still no clear distinction between clusters. This suggested that the numerical features are the dominant determining factors of the clusters when using both textual and numerical features. This also suggested that recipes cannot be easily clustered by rating using the available features in the dataset.

These observations were further backed up by the metrics we used to evaluate the clusters: v-measure and adjusted Rand Index (ARI). V-measure was chosen to evaluate how well each cluster only has recipes of the same rating and that all recipes of the same rating are assigned to the same cluster. ARI was chosen to evaluate how well the clustering model is able to put recipes of the same rating in the same cluster and recipes of different ratings in different clusters. For both metrics, higher values are better. Additionally, since we knew our dataset had approximately equal number of recipes per cluster, we calculated the ratio of the sizes of the 2 clusters, where an ideal clustering would result in a cluster size ratio of 1. According to the metrics displayed in (fig. 11) for the clusters identified in (fig. 12), none of our clustering models were able to identify cluster labels that accurately clustered recipes by rating.

Figure 11

Clustering Model	**Cluster Size Ratio**	**Adjusted Rand Index**	**V-measure**
KMeans	0.5984	0.0022	0.0232
Agglomerative, with Euclidean distance	0.1643	0.0080	0.0013

Figure 12



KMeans clustering achieved a marginally better v-measure of 0.02 while agglomerative clustering achieved a better ARI of 0.008. However, both of these values are close to 0, indicating that these clustering models were not much better than random labeling.

Sensitivity Analysis

In the evaluation metrics listed above, the KMeans clustering performs better than the Agglomerative clustering. Agglomerative clustering achieves more balanced cluster sizes, as seen with a higher cluster size ratio, but KMeans clustering is marginally, though not significantly, better in achieving complete or pure clusters, as seen with a higher v-measure and

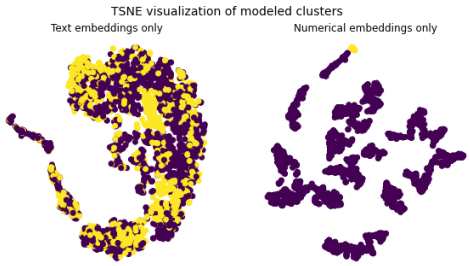
ARI. The results listed above are for using Sentence Transformer text embeddings for the KMeans clustering and Word2Vec text embeddings for the Agglomerative clustering.

The choice of features has a significant impact on the performance of KMeans clustering, as seen by the metrics in (fig. 13). Despite the T-SNE visualization showing more separation and clustering of recipes when using numerical features, KMeans clustering has better performance when not using numerical features.

Figure 13

Features Used	**Cluster Size Ratio**	**Adjusted Rand Index**	**V measure**
W2Vec Text features	0.5589	0.0022	0.0001
ST Text features	*0.5984*	*0.0538*	*0.0232*
W2Vec Text features + Numerical features	0.0080	-0.0002	3.4709e-6
ST Text features + Numerical features	0.0078	-0.0002	3.6736e-6
Numerical features	0.0078	-0.0002	3.6736e-6

Figure 14



When numerical features are used, KMeans clustering tends to group the majority of the data points into a single cluster, with the second cluster being as small as 12 recipes, seen in the top right of (fig. 14). Since we know in our dataset that the number of recipes in each rating group (5 versus not 5) is approximately equal, clustering the majority of recipes into a single cluster clearly has a negative impact on v-measure and ARI since the clusters will have low purity.

Discussion

The main surprise we experienced during this project was that the use of textual data, such as recipe category or keywords, wasn't as useful in segregating recipes of different ratings as we initially thought. As seen in both the supervised and unsupervised approaches, using features that were considered "metadata" or summary features such as the number of images or prep time, was much more important in obtaining clear results than using textual data. This implies that there is no defining factor for why a recipe of a certain category or using a specific keyword is given a high or low rating.

As noted in the sections on feature ablation and failure analysis, whether or not a recipe has images is the single most telling indication of a recipe's rating, but even without using

images in our model, we get almost identical results as with using images. This indicates that having images is just an indicator of a good recipe, not a cause. The most compelling explanation is that good recipes are those that the author put a lot of work into, and one part of putting work into a recipe is taking a picture, so having a picture is an indicator that effort has been put into a recipe.

As mentioned, we faced issues regarding both the dataset size and ratings imbalance. Given more time or resources, this project could be improved upon to address these issues. We could collect more data, especially recipes with lower ratings such as 1 or 2 stars, to get a balanced dataset without having to ignore the rest of the 5 star recipes. This should help us find clearer patterns for clustering.

Ethical Considerations

There are minimal ethical issues in our supervised or unsupervised approaches since our dataset is publicly available information. The names of authors of recipes and reviews are included in our dataset. These names are the online usernames that are displayed for users on Food.com. These names can potentially be personally identifiable information if users choose to use their real name as their online username. However, since we don't use this information in any of our approaches, we do not have to worry about personally identifiable information leaking into our modeling solutions. Another ethical concern is that users may choose which recipes to try or avoid depending on how machine learning algorithms are used to forecast aggregated ratings based on different recipe variables. Users may be misled or frustrated if the models show biases or errors, which might damage their faith in the platform or recipe suggestion system.

An additional ethical issue for our recipe rating predictor is ensuring that we do not predict recipes of a certain cuisine type to have a disproportionately high or low rating. This concern is due to the fact that topic modeling of reviews revealed that one of the topics for reviews with a rating of 1 was "Offensive name for minorities" (see Appendix E for more details). If our model predicted certain recipes to have a low rating due to belonging to a particular minority group, we risk perpetuating a negative bias against that minority group.

Statement of Work

Task	**Team member**
Data preprocessing and feature engineering	Adalie, Chaim and Xiaolan
Supervised learning	Chaim and Xiaolan
Unsupervised learning	Adaile and Xiaolan
Report	Adaile, Chaim and Xiaolan

Appendices

Appendix A: List of features used

Features Used

LR = Logistic Regression
NN = Neural Network
TM = Topic Modeling
CLU = Clustering

Feature	Model	Description
Name_len	LR, NN	Length, in words, of recipe name
Description_len	LR, NN	Length, in words, of recipe description
Images_len	LR, NN, CLU	Number of images
RecipeIngredientParts_len	LR	Number of ingredients
RecipeInstructions_len	LR, CLU	Number of steps
CookTime	LR, NN	Cook time in minutes
PrepTime	LR, NN	Preparation time in minutes
TotalTime	LR, NN, CLU	Total time in minutes
cuisine	LR	Flag if at least one keyword is a cuisine with at least 1,000 appearances over all recipes
Description (384 dimensions)	NN	Embeddings of the text from the Description column in the data created with the sentence transformers library with the "all-MiniLM-L12-v2" model
instructions (384 dimensions)	NN	Embeddings of the text of all of the steps from the RecipeInstructions column in the data, joined together, created with the sentence transformers library with the "all-MiniLM-L12-v2" model
Names_ST (384 dimensions)	NN, CLU	Embeddings of the text from the Name column in the data created with the sentence transformers library with the "all-MiniLM-L12-v2" model
Keywords_ST (384 dimensions)	CLU	Embeddings of the text from the Keywords column in the data created with the sentence transformers library with the "all-MiniLM-L12-v2" model

Names_W2V (300 dimensions)	CLU	Embeddings of the text from the Name column in the data created with the gensim library with the “word2vec-google-news-300” model
Keywords_W2V (300 dimensions)	CLU	Embeddings of the text from the Keywords column in the data created with the gensim library with the “word2vec-google-news-300” model
RecipeCategory	TM	String for recipe category
RecipeIngredientParts	NN	List of ingredients
Keywords	NN, TM	List of keywords

Features for Tree-Based Models

Numerical Features	**Text Features**
Calories, Duration, IsHealthy, FreeOf, HighIn, LowIn, Num_Images, DescriptionWordCount, ReviewCount, FatContent, SaturatedFatContent, CholesterolContent, SodiumContent, CarbohydrateContent, FiberContent, SugarContent, ProteinContent, RecipeServings, RecipeYield	Name, RecipeInstructions, Cuisine, Keywords, RecipeCategory

Appendix B: Additional Classifier results

XGBoost Classifier result:

```
Accuracy (XGBoost): 0.6018142846206844
Confusion Matrix (XGBoost):
 [[38627 15564]
 [26048 24265]]
Classification Report (XGBoost):
              precision    recall  f1-score   support

           0       0.60      0.71      0.65     54191
           1       0.61      0.48      0.54     50313

    accuracy                           0.60    104504
   macro avg       0.60      0.60      0.59    104504
weighted avg       0.60      0.60      0.60    104504


Accuracy (XGBoost): 0.6018142846206844
Average Precision (XGBoost): 0.6165825476378705
AUC (XGBoost): 0.6415229130517203
```

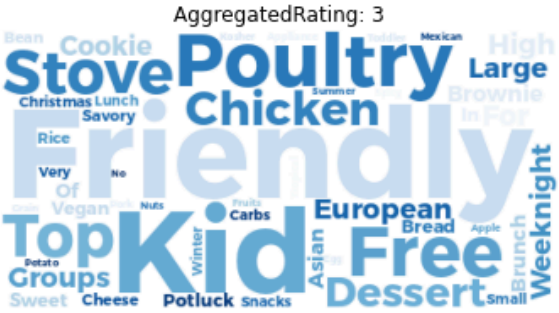
Decision Tree Classifier result:

```
Classification Report (Decision Tree Classifier):
              precision    recall  f1-score   support

           0       0.57      0.75      0.65     54191
           1       0.59      0.38      0.46     50313

    accuracy                           0.57    104504
   macro avg       0.58      0.56      0.55    104504
weighted avg       0.58      0.57      0.56    104504

Accuracy (Decision Tree): 0.5718250019138024
Average Precision (Decision Tree): 0.5491638503312264
AUC (Decision Tree): 0.5840571797374843
```

```
Confusion Matrix (Decision Tree Classifier):
 [[40760 13431]
 [31315 18998]]
```

Appendix C: Additional Wordclouds

Keywords by recipe rating



Appendix D: List of Recipe Topics

Rating	LDA Topics	NMF Topics
5	Asian, Rice, Berries Breads, Chicken, Poultry High, Pork, Breakfast, Brunch	Chicken, Poultry, Breast Large groups, Cookie, Brownie Kid friendly, European
4	Breads, Free, Beverages, Brunch Cookie, European, Stove top High, Potato, Kid friendly, Cheese	Chicken, Poultry, Breast Kid friendly, Stove top, Cheese Quick breads, Breakfast, Yeast

3	Kid friendly, Cookie, Brownie Cheese, Beans, Apple Chicken, Poultry, high carbs	Chicken, Poultry, breast, high carbs Cookie, Brownie, Large groups Quick breads, Breakfast
2	Pork, Sauces, Asian, Cheese, Rice Cookie, Friendly, Large groups Vegan, Beans, Beverages, Potato	Cookie, Brownie, Large groups Chicken, Poultry, Breast, high carbs Quick breads, Breakfast, Yeast
1	Cookie, Brownie, Free Friendly, Kid, European Breads, Chicken Poultry	Cookie, Brownie, Bar Quick Breads, Breakfast Poultry, Chicken breast
No rating	Breads, European, Brunch Free, Poultry, Chicken Meal, Pork, Large groups	Large groups, Brownie, Breads Chicken, Poultry, Breast Cookie, Brownie, Bar

Appendix E: List of Review Topics

Rating	**LDA Topics**	**NMF Topics**
5	Yum, chocolate chips, rave reviews Recipe made, great, thanks Sounds fantastic, tagged, wonderful, thanks	Used, made time, like Easy, make, quick, delicious Great recipe, thanks, sharing
3	Nothing spectacular, bourbon, good or bland flavor Spices, nutmeg, cinnamon Little flavor, time	Make sauce, recipe easy Good, pretty good, thanks Next time, add, use, try
1	Photos literally gives needed lighting Didn't like, hated, hoped, way spicy Offensive name for minorities	Recipe good, would make one time Like taste sorry, tasted nothing, like bland Way much salt, sorry

References

[1] Alvin. *Food.com - Recipes and Reviews*. *Kaggle*, https://www.kaggle.com/datasets/irkaal/foodcom-recipes-and-reviews.

[2] Guria, Sudipa, et al. "Classification of Foods based on Ingredients." *2023 International Conference on Computer Communication and Informatics (ICCCI)*, 2023. https://doi.org/10.1109/ICCCI56745.2023.10128374.

[3] Majumder, Bodhisattwa Prasad, et al. "Generating Personalized Recipes from Historical User Preferences." *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language*

Processing (EMNLP-IJCNLP), 2019, pp. 5976 - 5982. https://doi.org/10.48550/arXiv.1909.00105.

[4] Marín, Javier, et al. “Recipe1M+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images.” *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 43, no. 1, 2021, pp. 187-203, https://doi.org/10.1109/TPAMI.2019.2927476.

[5] Meijerink, Jeroen, and Emma Schoenmakers. “Why are online reviews in the sharing economy skewed toward positive ratings? Linking customer perceptions of service quality to leaving a review of an Airbnb stay.” *Journal of Tourism Futures*, vol. 7, no. 1, 2021, pp. 5-19, https://doi.org/10.1108/JTF-04-2019-0039.

[6] Shi, Lei, et al. “Selection bias mitigation in recommender system using uninteresting items based on temporal visibility.” *Expert Systems with Applications*, vol. 213, https://doi.org/10.1016/j.eswa.2022.118932.