Learning to Cook: An Exploration of Recipe Data

Travis Arffa (tarffa), Rachel Lim (rachelim), Jake Rachleff (jakerach)

***Abstract*— Using recipe data scraped from the internet, this project successfully implemented both unsupervised and supervised machine learning techniques in order to understand the nature and composition of online recipes, as well as to predict recipe ratings based solely on the content of their posting. This paper is divided into several sections. The INTRODUCTION explains the goals of the project, and the DATASET AND FEATURES section details the dataset used. Since the objective is two-fold and multiple models were tried, the METHODS, RESULTS, AND DISCUSSIONS are split up according to the question addressed and model used.**

I. INTRODUCTION

This project's aim was to explore recipe data and use machine learning techniques to understand the structures and patterns behind good cooking, given that food is such an essential part of life. Using data acquired from Epicurious.com, this project aims to answer two main questions: (1) Assuming no prior knowledge of food and recipes, what can be learned about natural groupings and structure of recipes? (2) How well can the quality of a recipe be predicted?

Practically speaking, the input was a collection of recipes scraped from Epicurious.com. To address the first question of finding ideal clusterings of recipes based on ingredients used in each recipe, the unsupervised learning technique of K-means clustering was used. This technique, using sparse ingredient feature vectors, succeeded in isolating specific types of food and granting insight into prevalence of different ingredients in specific types of cuisine. It also revealed interesting commonalities between cuisines, which are detailed in the discussion portion of the LEARNING CUISINE TYPES section.

To address the second question, supervised learning strategies including linear regression, Naive Bayes, and Random Forest were used to predict a recipe score based on input features such as recipe length, number of ingredients, type of ingredients, and nutritional value. These techniques illuminate which aspects of the online recipe posting served as the best predictors of ratings. The less-complex regression models failed to account for much of the variance in the rating data, while the random forest technique proved to yield more accurate predictions on the test set. Naive Bayes and Random Forest both avoided overfit as well as over-emphasis on ingredients which happened to be more prevalent in the training set. The various methods, models, and results are discussed in the RATINGS PREDICTION section of this paper.

Approaching the subject of food and recipes from multiple angles allowed an increased amount of utility to be gleaned from the data acquired. Furthermore, this project was able to explore and answer interesting inquiries into the occurrence of recipe ingredients, how recipes are correlated with each other, and what makes recipes differ in quality across a wide spectrum of food and drink options.

II. RELATED WORK

In the past, attempts have been made to classify dishes within a given cuisine type [1]. This approach proved useful in the subset of Indian cuisines - our project plans to generalize to all types of cuisines. In addition, previous reports in CS229 sought to classify recipes based on known labels [2]. By performing unsupervised learning instead of supervised learning, this project aims to discover what cuisine types should be instead of how they are already classified. Therefore, it builds on the research of past CS229 students in discovering structure in recipes.

Furthermore, previous CS229 projects sought to find the optimal amounts of specific ingredients for best recipes [3], and replace ingredients in recipes without losing taste [4]. Again, this project takes a much wider view on recipe prediction, looking to predict the quality of a recipe on the whole instead of the quality different amounts of ingredients used.

III. DATASET AND FEATURES

A set of 29,662 recipes was scraped from *Epicurious.com*, an online food resource for home cooks, each with some number of ratings and reviews given by users of the site. For each recipe, its name, list of ingredients, preparation steps, nutritional information if available, tags, and average user 'would make it again' rating (ranging from 0-100) were collected.

Further processing was done on the raw scraped data. From the list of ingredients, a dictionary of the 355 most common ingredients occurring in at least 120 recipes was hand curated, then each recipe's ingredient list was filtered with these words so that ingredients such as '1/2 teaspoon ground cardamom' were reduced to simple ingredient features such as 'cardamom'. In the end, the ingredients for each recipe were represented in an $\mathbb{R}^{355}$ binary vector, where the element in index i is 1 if ingredient i is present in the recipe, and 0 if absent. Quantities of the ingredients used were not taken into account. This constituted what we term the 'ingredient features'. Each recipe also had a list of

tags, which were classifications given to the recipe by the initial chefs. In the same manner as with simple ingredient features, tags occurring at least 15 times were converted into a $\mathbb{R}^{400}$ binary vector for each recipe.

Additionally, for each recipe, several real-valued features were extracted that were used in the ratings prediction segment of our project. These include length of recipe name, number of steps, number of ingredients, nutritional information (per serving fiber, polyunsaturated fat, sodium, carbohydrates, monounsaturated fat, calories, fat, saturated fat, cholesterol, protein).

For the first part of the project, clustering recipes, all ingredient and tag binary vectors for the 29,662 recipes were used. For the second part of our project, ratings prediction, only the features for recipes with greater than 15 ratings were used to minimize noise in the ratings. The data was divided into a training set of size 8,352 (80% of our data) and test set of size 2,088 (20% of our data) for a total of 10,440 training examples. The specific features used for each model will be discussed later in the METHODS parts of the next sections.

IV. LEARNING CUISINE TYPES

A. Methods

Since the model needed to be resistant to the bias of previous conceptions about food groups, unsupervised learning was a natural choice to address the goal of learning natural groupings of food. Specifically, K-means clustering was performed on binary ingredient vectors representing each of the 30,000 recipes scraped from Epicurious.com. In K-means, the following objective value is minimized, where there are m training examples, and k clusters each centered around a point $\mu_c, 1 \leq c \leq k$:

$$J(c, \mu) = \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2 \quad (1)$$

In order to find the optimal cluster size and learn the "best" grouping of food, K-means was run with values of k between 2 and 30. In addition to finding absolute error for each value of k, both the top ingredients by both volume and percentage, and the top tags by both volume and percentage were examined for each cluster. By inspecting the makeup of each cluster, the makeup of the dishes in each naturally occurring cluster were able to be determined. For K-means clustering, total error monotonically decreases with the number of clusters, and a kink in the total error vs k graph represents the most natural clustering of data. Therefore, the best possible clustering of ingredients into cuisine types is expected to occur at the kink.

B. Results

The total clustering error of K-means vs cluster size is plotted in Fig 1. Though total error starts to decrease at a decreasing rate between $k = 3$ and $k = 5$, there is no distinct kink. Instead, the absolute error decreases relatively smoothly over time, implying that there is not an ideal clustering of recipes.

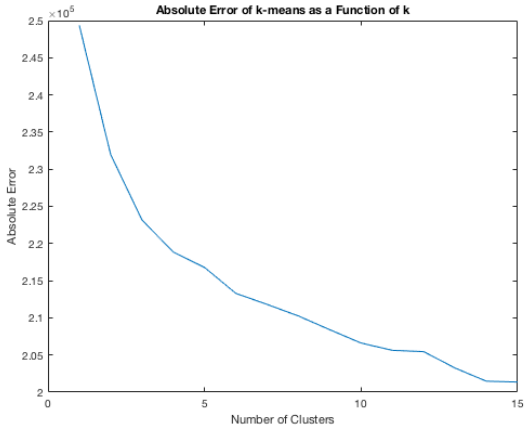

Fig. 1. Total clustering error of K-means vs number of clusters

C. Discussion

Though no single ideal grouping of cuisine types was found, the clustering algorithm did present a peculiar qualitative insight into the nature of cuisines. Manually examining the user-given tags corresponding to recipes of each cluster for different values of k, it is observed that when k is increased, a new cuisine type is 'discovered'. To see the significance of this discovery, please examine the two graphs presented of clusters of binary ingredient vectors run through principal component analysis and projected into two dimensions.

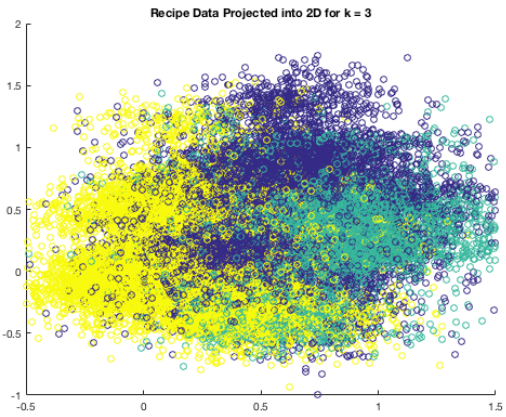

Fig. 2. Three clusters identified as Cooked Meals (Yellow), Desserts (Purple), and Drinks (Green)

For example, for $k = 3$ (see Fig 2), the most frequent tags by percentage of the groupings above were split into "italian american, mediterranean, african", "cake, dessert", and "punch, cocktail, alcoholic". This corresponded to a breakdown of categories, respectively, as Cooked Food, Desserts, and Drinks.

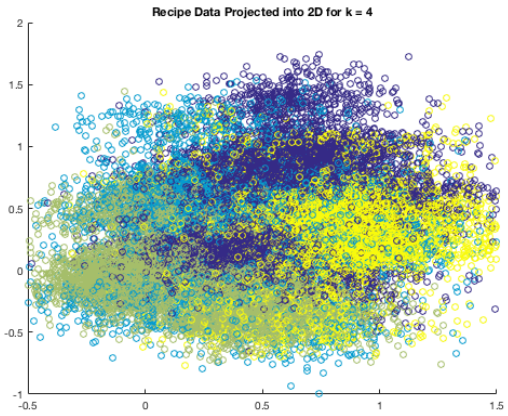

Fig. 3. Four clusters identified as European Based Meals (Green), Asian Based Meals (Teal), Desserts (Purple), and Drinks (Yellow)

When k is increased from 3 to 4 (see Fig 3), notice that instead of finding an entirely new grouping, the formerly yellow cluster splits into two, producing groups with top tags "stir-fry, wok, chinese, vietnamese, korean" and "mediterranean, italian american, greek".

This change in clusters is representative of the trend in increasing k values. For each increase of clusters past $k = 3$, one of the 3 major groups gets rearranged, but the general structure of having Meal groups, Desserts groups, and Drinks groups stays the same. This intuitively makes sense, since those three groups are extremely different in ingredient makeup. However, inside of those groups, there are myriad ways to distinguish between meal types. This phenomenon explains the lack of a strong kink in the graph of error as a function of k.

As evidence to the above arguments, it is observed that when k increases from 4 to 5, four clusters have relatively similar top tags, but the fifth now has top tags "middle eastern, african" and only rearranges the Meals group. Further rearrangements include splitting Desserts into Baked Goods and Frozen Desserts from 6 to 7, and drinks into Fruity Drinks and Non Fruity Drinks from 10 to 11.

Even though K-means is a non hierarchical algorithm, its results displayed hierarchical tendencies. Below the main hierarchy of Meals/Desserts/Drinks, by looking at when one of those clusters "splits" into different food groups, the new cluster's inherent differences can be assessed. Since Asian food and European food split immediately within meals, they are quite different and unique. However, since heavy meat based American food, and German/Scandanavian food do not split into separate clusters until $k = 15$, it can be inferred that the cuisines are similar.

V. RATINGS PREDICTION

In this section, our project aimed to create a model that utilized scraped and extracted information from the recipe postings, and could determine the user score for an online recipe with reasonable accuracy. Prediction methods included linear and locally-Weighted Regression, Naive Bayes, and Random Forest. Average absolute error was used as the optimal metric of test error since it is less sensitive to outliers and better handles the noisiness of the averaged ratings and the sparsity of features.

(I) Regression

A. Methods

The first attempt at ratings prediction used the numeric features provided by the scraper, which included nutritional score and recipe length, in order to determine which recipes internet users preferred. Both linear regression and locally-weighted linear regression with exponential weights and a bandwidth parameter of 5 were run.

B. Results

Overall, these methods did not accurately predict user ratings. The mean absolute test error and mean squared error for linear regression were 6.94 and 92.84, respectively. Using locally-weighted regression, errors worsened to an absolute error of 7.76, and a mean squared error of 163.66.

C. Discussion

Locally-weighted linear regression is susceptible to outliers and overfitting, which may account for its poor performance compared to standard linear regression. Given the variance in the number of ratings per recipe, as well as individual user biases, the number of outliers is expected to be significant. Furthermore, recipe ratings are likely not linearly related to the features chosen, and the small set of real-valued features used mean there is high bias in this model.

(II) Naive Bayes

A. Methods

The next attempt at ratings prediction used Naive Bayes, a simple probabilistic model that applies Bayes' theorem with strong independence assumptions between the features. Specifically, it assumes that features are conditionally independent given the class variable. While the assumption does not hold in the case of recipes, since ingredient features are in reality not independent (some ingredients, after all, tend to 'go together'), it serves as a good baseline model for prediction.

For the Naive Bayes model, ratings were discretized into buckets, each capturing the same fraction of training examples. Then multiclass Naive Bayes classification was performed, where each 'class' corresponds to a range of predicted ratings. Suppose there exists a new recipe r for which a prediction will be run. Let k be the number of buckets, N be the number of possible ingredients, and n_i be the number of ingredients for recipe r. Let x be the binary ingredient features of the new recipe (so, for the collected

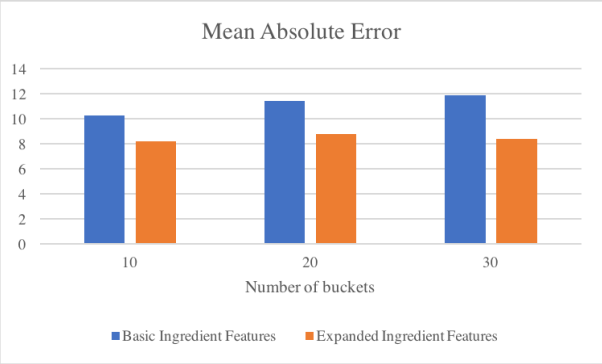

Fig. 4. Mean absolute error for Naive Bayes with basic ingredient features vs expanded ingredient features, and varying number of ratings buckets

simple ingredient features, $x \in \mathbb{R}^{355}$). The bucket the recipe belongs to is chosen by:

$$\begin{aligned} b &= \arg \max_{b=1,\cdots,k} p(y=b|x) \\ &= \arg \max_{b=1,\cdots,k} \frac{p(x|y=b)p(y=b)}{p(x)} \\ &= \arg \max_{b=1,\cdots,k} \frac{(\prod_{i=1}^{n_r} p(x_i|y=b))p(y=b)}{\sum_{b'=1}^{k} (\prod_{i=1}^{n_r} p(x_i|y=b'))p(y=b')} \end{aligned}$$

Thus, whichever class has the highest posterior probability is picked. This model estimates $p(x_i|y)$ with a binomial distribution, based on the counts of ingredients in recipes that have been seen already, with Laplace smoothing. Supposing there are m recipes as training data,

$$\begin{aligned} p(q|y=b) &= \phi_{q|y=b} \\ &= \frac{\sum_{i=1}^{m} 1\{x_q^{(i)} = 1 \wedge y^{(i)} = b\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = b\} + 2} \end{aligned}$$

Additionally, due to the naivete of the conditional independence assumption for recipes, an expanded ingredients feature set was experimented with, where pairwise ingredients were used instead of simple ingredients. This was expected to better model dependencies between ingredients. Hence, when running Naive Bayes with the expanded feature set, each feature vector has 355^2 binary features, i.e. $x \in \mathbb{R}^{126025}$. For both sets of experiments, the the feature vectors are expected to be sparse, since the recipes scraped use, on average, 10.7553 ingredients.

B. Results

By experimenting with the number of buckets ($k = 10, 20, 30$), it was found that regardless of the number of buckets the ratings are discretized into, the expanded ingredients feature set performs significantly better than the basic ingredients feature set, as can be seen from Fig. 4.

Discretizing the labels into 10 buckets, a mean absolute error of 10.248433 is obtained using the basic ingredient features, and 8.198488 using the expanded ingredient features. This constitutes a reduction in error of about 20%.

C. Discussion

Overall, Naive Bayes was a poor predictor of recipe ratings, performing worse than the other prediction models used in terms of both mean squared error and mean absolute error. This is likely because the conditional independence assumption the model is built on does not hold in the case of recipes. However, using the expanded pairwise feature set yielded a marked improvement over the simple ingredient feature set used, since pairwise ingredient features captures dependencies between ingredients better. This agrees with one's intuition, that the 'pairing' of ingredients, rather than just the ingredients independently, has a stronger impact on how good the food tastes, and thus how well rated the recipe is.

(III) Random Forest

A. Methods

The Random Forest technique uses randomized samples of data to fit a multitude of smaller regression trees instead of one large tree. It outputs a prediction that is the average output of each smaller tree. The randomization serves to reduce correlation between trees, while the overall method reduces overfit to the training data that is common in many large trees. This model was chosen due to the large number of ingredients, as well as the potential for overfit to specific ingredients that happen to be more prevalent in the training data.

B. Results

Both a full model, and a reduced model were built with details shown below. A minimum leaf size of 50 was also chosen based on the graph below:

TABLE I
Comparison of Full and Reduced Random Forest Models

	Full	Reduced
Minimum Leaf Size	50	5
Number of Trees	100	10
Number of Predictors	355	15
Subsample Size	200	200
Mean Absolute Test Error	6.07	6.08

The test error for each model was nearly identical, indicating that the additional features as well as tree complexity in the full model did not produce more accurate predictions. The fifteen predictors were chosen for the reduced model based on the Out-of-Bag Variable Importance parameter, which is roughly the average absolute difference between tree outputs that included the feature, and those that did not. Further, the sub-sample size was kept constant (and relatively high) for both the full and reduced models to account for the sparsity of the feature vectors.

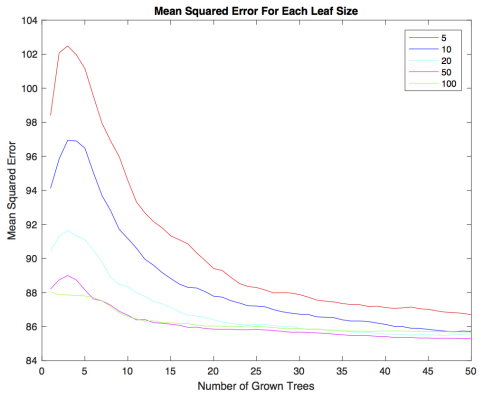

Fig. 5. Error based on Minimum Leaf Sizes of 5, 10, 20, 50, and 100

C. Discussion

The Random Forest procedure produced reasonable predictions, with average absolute error around 6.08, and mean squared error of 86.5. Random Forest outperformed all other regression techniques attempted. One fault of the Random Forest predictions was that for each test recipe, the predicted rating tended to remain close to the mean of the training ratings, which was roughly 87. This was particularly true when the number of trees or tree-width was large, because the weights for each feature (ingredient) remained small. This was counteracted by using more features and examples for both the full and reduced model trees, as well as using the fillProximities function to account for outliers and clustering of data.

Through use of the Variable Importance parameter, we were also able to determine the fifteen most influential ingredients on ratings. These included sugar, salt, chocolate, and broccoli, with the latter suggesting lower ratings in most of its corresponding regression trees.

Overall: Comparison of Recipe Ratings Prediction Techniques

TABLE II
Comparison of Test Error for each Method

Method Used	**Mean Absolute Error**	Mean Squared Error
Linear Regression	6.94	92.84
Locally Weighted Linear Regression	7.76	163.66
Naive Bayes ($k = 10$)	10.25	184.98
Naive Bayes with Expanded Features ($k = 10$)	8.20	131.35
Random Forest Full	6.07	85.81
Random Forest Reduced	6.08	86.50

Overall, Random Forest performed the best of all our ratings prediction techniques, and Naive Bayes the worst. The latter makes overly simplistic assumptions regarding the conditional independence of ingredient features, whereas the former is able to elucidate most accurately the features that sets a recipe apart from others. These conditional dependence assumptions are mitigated in the pairwise Naive Bayes implementation, thus explaining the drop in error. Furthermore, the comparative success of Linear Regression against Naive Bayes suggests that predictive power could lie within nutritional data - a set of features that was not taken into account for Naive Bayes.

VI. Conclusions and Future Work

Both arms of this project provided interesting results. The first goal, learning about cuisine structure, provided more promising results. The K-Means clustering technique shed light onto an underlying hierarchical structure of cuisines based on ingredients. Though the expressed goal of finding the ideal clustering of cuisine types was not reached, this project uncovered a much different, and more useful explanation of how the interaction of cuisine types works. In terms of prediction methods, more naive attempts at regression and classification faltered due to their inability to counteract overfit, massive differences in sparsity of ingredients, and flawed model assumptions. The reduced-form Random Forest Model was best able to address the challenges in the data by injecting randomness into the training data, and making fewer assumptions about linearity and conditional independence. Future research into the network of dependence of ingredients is necessary. With a better understanding of the interaction between ingredients, more apt models can be chosen to predict ratings.

References

[1] A. Jain, R. NK, G. Bagler, Spices form the basis of food pairing in Indian cuisine. Indian Institute of Technology Jodhpur, February 2015.
[2] J. Naik, V. Polamreddi, Cuisine Classification and Recipe Generation. CS 229, November 2015.
[3] D. Safreno, Y. Deng, The Recipe Learner. CS 229, November 2013.
[4] A. Agarwal, D. Jachowski, S.D. Wu, RoboChef: Automatic Recipe Generation. CS 229, November 2009.