Learning to Cook – An Exploration of Recipe Data

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

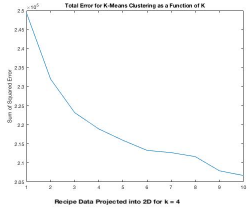
Goals

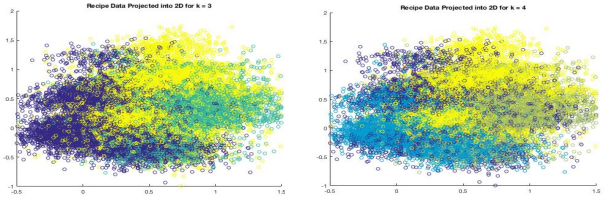
We set out to solve two problems. First, we wanted to figure out the different "types" of recipes based purely on what ingredients were included, which would allow us to understand which ingredients are prevalent in which type of cuisine. Second, we wanted to predict recipe review scores based on recipe ingredients and real valued features such as nutrition score and number of steps.

(1) Clustering Recipes

Model: We sought to define a cuisine based solely on its ingredients and no preconceived notions about cuisine itself. Thus, we found the unsupervised learning strategy of k-means clustering to be the best model for this task, which we could then verify with a recipe's tags.

Results and discussion: Varying the number of clusters, we obtained the following graph of total squared error. We see an inflection point around k=3, suggesting 3 as the optimal number of clusters.





The graphs shows clusters for k=3 and k=4. For k=3, inspecting the tags of recipes belonging to each cluster, we observe that these clusters correspond to meals, drinks and desserts. We also observe an interesting trend: as we increased the number of clusters, these recipe classes were split further into natural subclasses. When k increases from 3 to 4, the cluster corresponding to 'meals' (in purple) is split into Asian and European cuisines. For each increase of k past the kink, we still discover new cuisine types with similar flavors based on their tags and ingredients, meaning that our most informative cluster sizes were not dependent on cluster error graph's inflection.

Data/Features

Data: We scraped all recipes currently on Epicurious.com for our data set. For each recipe, we scraped its ingredients, preparation steps, nutritional information, cook time, and user ratings (ranging 0-100). We filtered out recipes with fewer than 15 ratings, and collected 10,440 recipes in total. For clustering, we used all the recipe data. For prediction, we randomly partitioned the dataset into training and test sets constituting 80% and 20% of the recipes respectively.

Features: For Naive Bayes, Clustering, and Random Forest, we used hand-curated $\mathbf{R}^{355}$ binary feature vectors of ingredients. We used simple features like number of steps, number of ingredients for linear regressions, and expanded the ingredient features for Naive Bayes as well.

(2) Recipe Rating Prediction

To learn the quality of a recipe (measured by its rating), we tried several different machine learning models, including linear regression, locally weighted linear regression, Naive Bayes, and Random Forest. We discuss the latter two models in depth.

Naive Bayes

Model: Naive Bayes is a probabilistic model for classification that assumes the occurrence of features is conditionally independent given the class variable. While this assumption does not hold in the case of recipes, it is a good baseline model for prediction.

Data: We discretized the ratings into evenly-sized buckets and performed multiclass Naive Bayes classification, experimenting with number of buckets and feature type (a $\mathbf{R}^{355}$ binary feature vector of ingredients, and a $\mathbf{R}^{126025}$ binary feature vector of paired ingredients).

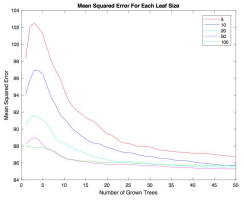
Results and discussion: Prediction works better with pairwise ingredient features. We expect this to be the case, since the conditional independence assumption does not hold for recipe ingredients, and ingredients tend to "go well together".

Mean Absolute Error		
Number of buckets	**10**	**20**
Basic ingredient features	8.41	9.42
Pairwise ingredient features	7.78	8.84

Random Forest

Model: Random Forest uses randomized samples of data to fit several smaller regression trees. It outputs a prediction that is the average output of each tree. Randomization reduces correlation between trees, and the use of multiple trees counteracts overfit. We chose RF due to the large number of ingredients and potential overfit to ingredients that occur frequently in the train data.

	Full	**Reduced**
Min. Leaf Size	50	5
Num. Trees	100	10
Num. Predictors	355	15
Sub-Sample Size	200	200
Abs Test Error	6.07	6.08



Results: RF had an average absolute test error around 6.08, and MSE of 86.5. RF outperformed other regression techniques that we attempted, including linear regression and locally-weighted linear regression.

Discussion: The test errors for each model indicate that additional features and tree complexity did not yield more accurate predictions. Fifteen predictors were chosen for the reduced model based on the *Out-of-Bag Variable Importance* parameter, which equals the average difference between tree outputs that included the feature, and those that did not. The sample size was kept constant to account for the sparsity of the feature vectors.

Future Research

The next step for our project would be to auto-generate recipes using the clustered tags while striving to maximize ratings. This would represent a combination of the supervised and unsupervised techniques currently presented, as well as additional modeling to account for varying amounts of each ingredient. Another interesting avenue of research would be to look at how the ingredients and amounts of each ingredient correspond to nutritional value.