CS 221

An Artificial Intelligence Approach for Recipe Evaluation

Jorge Cordero
Dept. of Electrical Engineering
Graduate School of Business
Stanford University
icoen@stanford.edu

Ivy Wang
Dept. of Computer Science
Stanford University
wangivy@stanford.edu

1 Introduction

The goal of this project is to determine the rating of a dish based on its recipe. We will be doing this using the Kaggle dataset 'Epicurious - Recipes with Rating and Nutrition'[1].

The input of our algorithm will be the full JSON format text of the recipe ingredients and the output of our algorithm will be a real number rating between 0 and 5. For example, on the recipe for Lentil, Apple, and Turkey Wrap, the input would be the JSON format text of the ingredients:

```
INPUT:
    ingredients:[] 15 items
    0:4 cups low-sodium vegetable or chicken stock
    1:1 cup dried brown lentils
    2:1/2 cup dried French green lentils
    3:2 stalks celery, chopped
    4:1 large carrot, peeled and chopped
    5:1 sprig fresh thyme
    6:1 teaspoon kosher salt
    7:1 medium tomato, cored, seeded, and diced
    8:1 small Fuji apple, cored and diced
    9:1 tablespoon freshly squeezed lemon juice
    10:2 teaspoons extra-virgin olive oil
    11:Freshly ground black pepper to taste
    12:3 sheets whole-wheat lavash, cut in half crosswise, or 6 (12-inch) flour tortillas
    13:3/4 pound turkey breast, thinly sliced
    14:1/2 head Bibb lettuce
```

and the expected output should reflect the average rating the recipe received:

```
OUTPUT:
    2.5
```

2 Evaluation Metric and Baseline

For the evaluation metric, we classified predictions $h(x)$ as correct if they were within one point of the true rating y, as expressed in the equation below:

$$\text{correct}(h(x)) \iff h(x) \in [y-1, y+1]$$

The baseline model that we use is a constant function that outputs the mean ratings from the training set. Because our evaluation metric allows for a small degree of error in the prediction, our baseline performs reasonably well with an an RMSE of 1.3435 units (rating points); and based on the evaluation metric, an accuracy of 73.30%.

For the oracle, we had every member of the team look at 20 samples of the entire recipe (not just limited to the ingredients) and predict a rating on a scale from 0 to 5. Based on the evaluation metric of being within one point of the true rating, the average accuracy of a human oracle is 58.3%.

We theorize the oracle algorithm is not as good as the baseline algorithm since there are only three graders who have different backgrounds and different preferences. Furthermore, determining whether a recipe is successful is a task that humans find incredibly difficult, especially without having tried recreating the recipe. This is one of the main motivations behind using artificial intelligence to predict the success of a recipe.

3 Approach & Methods

3.1 Dataset and Data Preprocessing

We split all of our data using a 60/20/20 split for our training, validation, and test set respectively. This gives us $12,011$ samples for our training set and 4004 samples each for our validation and test sets.

Our initial dataset is a CSV file obtained directly from ‘Epicurious - Recipes with Rating and Nutrition’. The dataset contains entries representing recipes, each with an integer for the ’Rating’ label and a 0 or 1 value for each ingredient feature depending on whether or not the ingredient appeared in the recipe. For example, the recipe shown above would correspond to the following:

$$\{\ \text{‘rating’}{:}3.75,\ \text{‘carrot’}{:}1,\ \text{‘eggs’}{:}0,\ \text{‘thyme’}{:}1, ...\}$$

The initial features contain erroneous data for non-food items due to incorrect text parsing, so we have removed those labels. We have also combined the data so that features with similar labels such as ‘cookie’ and ‘cookies’ will appear as a single feature.

After seeing our algorithms perform more poorly on this dataset than our baseline, we believe that the initial dataset over-simplified the available data and does not fully capture the information needed to predict the ‘rating’. In an attempt to remedy this, we have parsed the dataset from the original JSON file containing all the recipes and expanded out features to include the following parameters: length of the recipe directions, length of the ingredients, and frequency of the ingredient occurrence within the recipe.

3.2 Models and Algorithms

3.2.1 Linear Regression

As an initial exploration of potential models, we hypothesized that a good model for our problem would be a Multivariate Linear Regression. We trained our model using Batch Gradient Descent, for a total of 9,000 iterations. After tuning the regularization parameter to prevent overfitting, our model yields an RMSE of 1.3112 units and an accuracy of 70.50%.

3.2.2 K-Nearest Neighbors

The second algorithm we examined is the K-Nearest Neighbors approach which returns the average ratings of the K closest observations from the training set. In this method, we measure closeness by taking the Euclidean distance of the set of features that have been centered and normalized.

We tuned our algorithm with respect to the number of neighbors (k) used and obtained the following performance graph where we obtain an RMSE of 1.2856 and 1.3237 units and an accuracy of 74.13% and 73.60% for $k = 5000$ and $k = 105$ respectively.

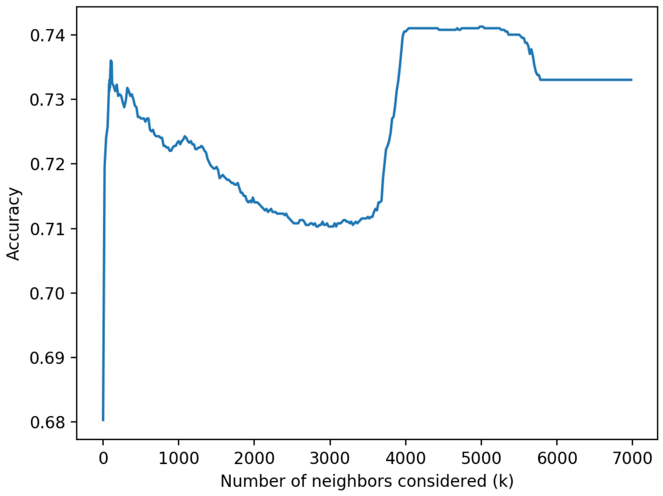

Figure 1: Tuning of KNN Algorithm

One important observation is that this performance graph is bimodal, with two very distinct peaks. This is anomalous and we will discuss the implications of this in the results section below.

3.2.3 Tree Based Methods

We also decided to implement Tree Based Methods, which use a series of decision rules to $stratify$ the feature space, referred to as nodes [2]. A single regression tree bases its prediction on the mean of the training observations to which the testing observation falls into, therefore an increase in the number of leaves results in a decrease in bias but a subsequent increase in variance. In order to counteract for this effect, we leveraged the concept of $Bootstrapping$ by implementing a Random Forest method. We decided to create 10,000 decision trees, where for each split in each of them, we randomly take into consideration only $28 \sim \sqrt{792}$ predictors. Approximately $\frac{764}{792}$ of the splits will not be considering the $strongest$ predictor, which results in a decorrelation of the trees, and a decrease in variance [2]. The resulting model gives an RMSE is 1.2641 with an accuracy of 73.10%.

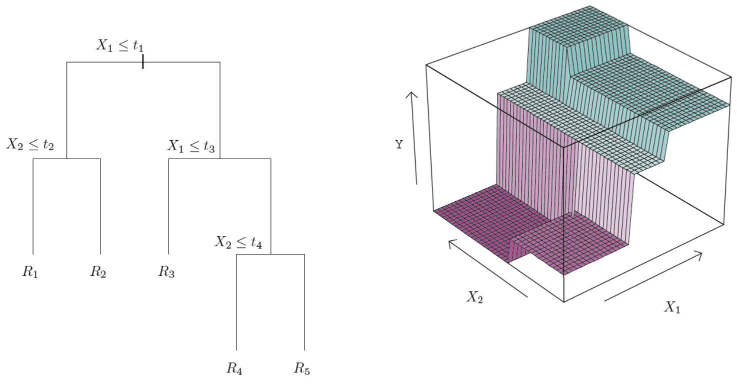

Figure 2: Stratification of the Feature Space

4 Results

Figure 3 depicts plots of the True Rating vs Predicted Rating by our Linear Regression model, Random Forest, and KNN methods, respectively. Each point represents an actual rating and predicted rating pair. Overlap of these points give a more dense appearance of points and appear darker on the plot.

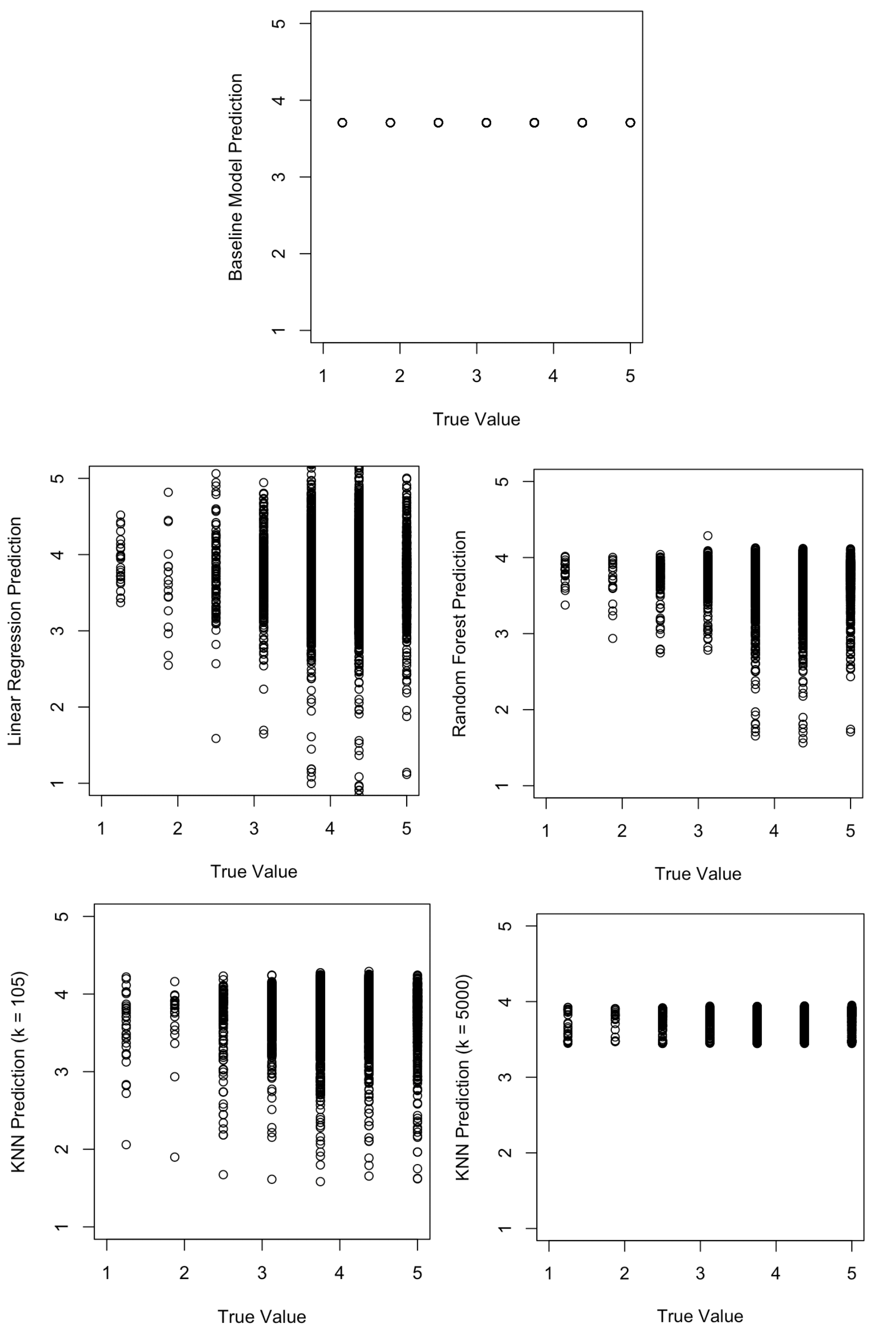

Figure 3: True Ratings vs Predicted Ratings

The RMSE and the accuracies of the ratings are shown in Table 1 below.

Despite the RMSEs of our models being slightly lower than compared to the baseline RMSE, the accuracy for most of our models have failed to significantly increase and some have actually decreased. Furthermore, as shown earlier in Figure 1, we observe the anomalous behavior in our KNN model where the accuracy suddenly improves when the number of neighbors is large. In particular, the KNN predictions shown in Figure 3 show that the predictions made for KNN with $k = 5000$ are very similar to the predictions of the baseline. This makes sense, as taking the average rating of 5000 nearest neighbors out of $12,011$ training observations will give a result that is very close to the average rating for all training observations.

	Model				
	Baseline	**Linear Regression**	**Random Forests**	**KNN** $(k = 105)$	**KNN** $(k = 5000)$
RMSE	1.3435	1.3112	1.2641	1.2856	1.3237
Accuracy	0.7330	0.7050	0.7310	0.7332	0.7412

Table 1: Performance of each model with respect to RMSE and accuracy.

The fact that prediction models that give outputs similar to the baseline perform better strongly suggests that our features do not sufficiently capture enough information for our models to use. This is supported by how the RMSE decreases when $k = 150$ compared to when $k = 5000$, which means that the predictions are closer to the actual ratings when $k = 150$, despite accuracy decreasing. Our hypothesis is that the information given by the features do not contribute enough information to make an good prediction, so simply picking a conservative baseline prediction fares better.

5 Next Steps

As of this moment, the features that are being used do not substantially contribute to prediction accuracy due to them being sparse in high dimensions. Thus, our models are suffering from `The Curse of Dimensionality`.

One potential solution to this problem is to reduce sparsity and contextualize the ingredients using word embeddings. There have been previous attempts to do this with food features specifically, both of which involve word2vec methods applied to co-occurences of ingredients. The two approaches that we plan on trying consist of creating word embeddings of the ingredients using word2vec applied to the co-occurence of ingredients from our recipe dataset[3], and using the co-occurence of the ingredients based on their flavor profiles[4]. This will capture context information between how various ingredients are related. We will then generate an embedding of each recipe by taking the weighted mean of the embeddings of its constituent ingredients. This will allow our recipes to be directly comparable according to a similarity metric; e.g. we could perform KNN to predict the recipe ratings based on the closeness of the queried recipe with respect to labeled recipes.

References

[1] H. Darwood, “Epicurious - recipes with rating and nutrition,” 2017.

[2] T. H. R. T. Gareth James, Daniela Witten, *An Introduction to Statistical Learning*. 2013.

[3] J. Altosaar, “food2vec - augmented cooking with machine intelligence,” 2017.

[4] Y.-Y. Ahn, S. E. Ahnert, J. P. Bagrow, and A.-L. Barabási, “Flavor network and the principles of food pairing,” *Scientific reports*, vol. 1, p. 196, 2011.