# Recipedia

# Nutrition Regressor

CMPUT 466 Course Project

Due: April 13, 2020

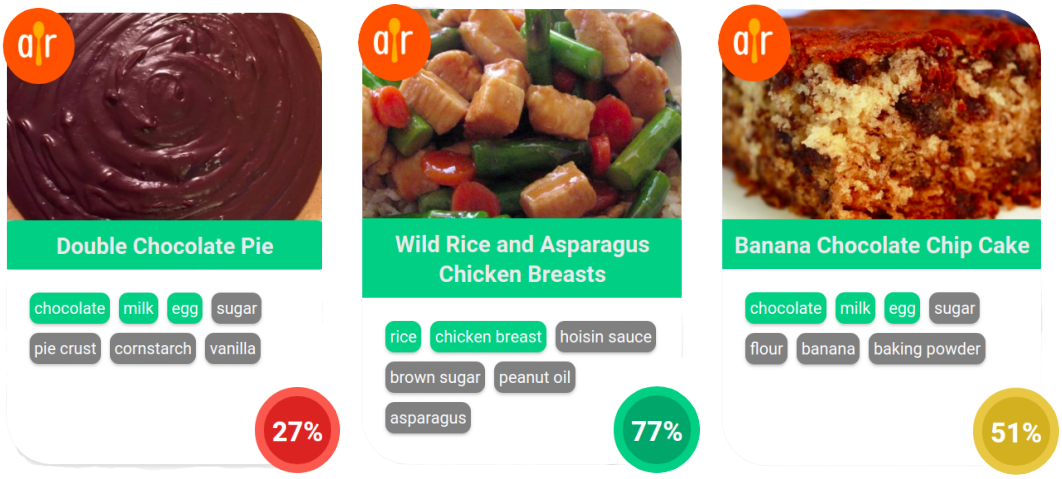
James Hryniw, 1431912

# Problem Domain

For the Software Engineering capstone project (ECE 493), my group and I (three members) created a web application called Recipedia. You can access the completed website at recipedia.ca.

Recipedia is a recipe search engine, but uses ingredients as the basis of the search query instead of names or cuisines. The motivation is that clients can look to their pantry for inspiration, and build a potential list of recipes by iteratively adding more ingredients that they do have. A full solution should allow a user to

- maximize the use of ingredients they already have;
- explore recipes quickly using different ingredient combinations;
- **indicate the relative healthiness of each recipe.**




For the CMPUT 466 course project, we are focusing only on the last component, forming an algorithm to indicate the relative healthiness of each recipe. The goal is to assign a score from 0 to 100% to each recipe, and display it at the bottom of each recipe result card in a colored circle. When comparing two recipes, a higher score should, in general, indicate a healthier recipe. I decided to undertake this challenge, alone, and apply machine learning techniques to predict a health score for each recipe. Taking on this mini-project allowed me to both add value to my capstone project and apply machine learning to a real world problem at the same time.

# Problem Formulation

Health scores can be modelled as a straightforward regression problem where the recipe metadata, nutritional content and ingredients extracted from the recipe webpage are the features and a health score, between 0 and 100 is the target label.

Recipe data is potentially extracted from various recipe websites across the internet. However, to simplify the scope of the project, all recipe data has been extracted from one website -- allrecipes.com. From this data, 23 relevant features were extracted. These features can be categorized into three groups:

- Rate - Features related to the rating or popularity of the recipe. The intuition for including features in this group is that unhealthier recipes tend to be rated higher.
- Nutritional Info (NInfo): Features related to the nutritional content of the recipe
- Ingredients: List of parsed ingredients contained in the recipe

In Table 1 below, each of the 23 extracted features is associated with one of these groups paired with a description.

| **Feature** | **Group** | **Description** |
|---|---|---|
| rating | Rate | The number of stars for the recipe, floating point average from 0 to 5 |
| reviews | Rate | Total number of ratings for the recipe |
| made_it_count | Rate | Number of people who reported making the recipe |
| calories | NInfo | Number of calories |
| total_fat | NInfo | Total fat, g |
| saturated_fat | NInfo | Saturated fat, g |
| cholesterol | NInfo | Cholesterol, mg |
| sodium | NInfo | Sodium, mg |
| potassium | NInfo | Potassium, mg |
| carbs | NInfo | Total carbohydrates, g |

| | | |
|---|---|---|
| dietary_fiber | NInfo | Fiber, g |
| protein | NInfo | Protein, g |
| sugars | NInfo | Sugars, g |
| vitamin_a | NInfo | Vitamin A, IU |
| vitamin_c | NInfo | Vitamin C, mg |
| calcium | NInfo | Calcium, mg |
| iron | NInfo | Iron, mg |
| thiamin | NInfo | Thiamin, mg |
| niacin | NInfo | Niacin, mg |
| vitamin_b6 | NInfo | Vitamin B6, mg |
| magnesium | NInfo | Magnesium, mg |
| folate | NInfo | Folate, mcg |
| ingredients | Ingredients | A list of parsed ingredients for the recipe. Ingredients are extracted from a fixed list of supported ingredients (e.g milk, peanut butter etc.) |

Table 1 - Collected Features

Note that all the features, with the exception of the ingredients, are numeric and can be used as input features in a straightforward way. Since the ingredients are extracted from a fixed list (of length 893), we can add each ingredient as a feature where that feature is 1 if the ingredient is included and 0 if it is not.

We introduce specific **augmented features** to help the models perform better. Augmented features, explained in Table 2, include

| **Feature** | **Group** | **Definition** |
|---|---|---|
| cal_protein | Augmented | The percentage of calories coming from protein. Defined as<br>protein * 4.0 / total_calories. |
| cal_fat | Augmented | The percentage of calories coming from fat. Defined |

| | | |
|---|---|---|
| | | as<br>total_fat * 9.0 / total_calories. |
| cal_carbs | Augmented | The percentage of calories coming from carbs. Defined as<br>carbs * 4.0 / total_calories. |

Table 2 - Augmented Features

For the output score, we took our results by manually cross-referencing the recipe data we had collected against nutritionrank.com which offers a commercial nutrition scoring service with the same scoring scheme we are targeting. Many, but not all the recipes we collected were found on this website. By manually attaching the nutrition rank score to our collected data, we are essentially trying to approximate their closed-source commercial recipe ranking algorithm using machine learning.

Since this is not a famous dataset, and was manually cross-referenced, the number of data samples is relatively low and contained a few transcription errors. In total, the dataset contains **690 labelled samples**.

# Approach

We define the hypothesis class for this problem as the set of all ordinary least square (OLS) models (with L1 or L2 regularization), Theil-sen estimators, and neural networks with up to two hidden layers. We will additionally consider a trivial regression model that predicts the mean of the training samples as a baseline. For all four models, predictions will be in the range of 0 to 100.

**Features**

In addition to tuning model-specific hyperparameters, since the feature list is not predetermined I also needed to try different combinations of features. In particular, this means including or discluding the Rate, Ingredient or Augmented feature groups.

**Definition of Risk**

The risk of the model on the training and validation sets will be the average absolute error of the data samples in the training set. Since y is normalized, we multiply the risk by the standard deviation of the nutrition score so that the output is equal to the expected absolute nutriscore point difference for any prediction.

$$R = \frac{\hat{\sigma}_y}{M} \sum_{i=1}^{M} |\hat{y} - y|$$

**Measure of Success**

The goal of training the nutriscore model is to minimize the risk on the validation set such that a good score on the test set can be achieved. What this means, is that we will use the model and hyperparameters that yielded the lowest **validation** score and report the results on the test set, whether they went up or down from the previous iteration. Of course, decreasing the risk on the validation set should hopefully correlate to minimizing the risk on the test set. That way, it's as if we only see the test set once.

There is no threshold for success on this model as we are not comparing it with an existing model.

# Results

**Optimal Feature Set**

After some experimentation, it was found that including the nutrition information and augmented features, but not the rating or ingredients yielded the best validation results. Note you will not find the code with ingredient features implemented in my submission, as the amount of additional processing and code overhead was too extensive to justify keeping just so that the mediocre results could be reproduced. If you are interested, you can find it on the *ingredient-features* branch at https://github.com/recipedia-uofa/nutrition-regressor/tree/ingredient-features.

In Table 3 is a comparison of the validation performance for different feature sets on an OLS model. The best performing combination is bolded, and was used for the remaining training of the models.

| Feature Set | Validation Risk |
|---|---|
| NInfo | 9.504170798611378 |
| NInfo + Rate | 9.559796656280657 |
| **NInfo + Augmented** | **9.459243994315843** |
| NInfo + Rate + Augmented | 9.493474011813323 |

| | |
|---|---|
| NInfo + Rate + Augmented + Ingredients | 37.46889322586937 (Ouch.) |

Table 3 - Feature Selection

**kfold-Analysis / Outlier Analysis**

Since the data was manually labelled and with few samples, I wanted to do some tests to filter out errors in the dataset. To do this, k-fold cross-validation was conducted on the dataset with an OLS model, which reports the validation performance of every fold. The trick is that if each fold in the dataset is iid, then the validation performance should be close to the same for each fold. If there is a large variation, then that means that either there are not enough samples in the dataset to make it iid or there are large outliers in the dataset.

Initially, this yielded

[**0.25735498** 0.52669603 **0.41158381** 0.59890805 0.58229935]

which has some obvious problems in the first and third fold. So, to further my search for the culprit I created an outlier analysis test. This test trains and tests an OLS model **on the same data**. Finding the largest errors in the training set should point out the most problematic samples since OLS is susceptible to outliers. This yielded a couple samples where I had accidentally labelled the sample by the number of calories in the recipe instead of the nutriscore (a *much* lower number)! Removing the top 10 worst performing samples yields a much more reasonable k-fold cross validation result.

[0.64362631 0.63993017 0.62505006 0.5822936  0.60630005]

**Model Selection / Hyperparameter Tuning**

For model selection, we considered a total of five model types with different training objectives. Additionally, we consider the trivial estimator, which simply predicts the mean of the training samples. For each model, shown in Table 4, we consider a set of hyperparameters to tune.

| **Model Type** | **Description** | **Hyperparameter Set** |
|---|---|---|
| Trivial Estimator | Predicts the mean of the training samples | N/A |
| OLS | | N/A, normalization is only potential |

| | | |
|---|---|---|
| | | hyperparameter but the data is already normalized |
| Ridge | OLS + L2 regularization | max_iter: [None, 100, 1000]<br><br># Controls strength of L2 regularization<br>alpha: [0.01, 0.3, 1, 3, 10] |
| Elastic Net | OLS + L1 and L2 regularization | # Controls portion of L1 regularization. 1.0 means full L1 regularization<br>l1_ratio: [1.0, 0.7, 0.5, 0.3, 0.1, 0.01, 0.001]<br><br># Controls strength of regularization<br>alpha: [0.001, 0.01, 0.1] |
| Theil-Sen Regressor | Linear regression robust to outliers. | N/A, normalization is only potential hyperparameter but the data is already normalized |
| Neural Network Regressor | Neural network with up to two hidden layers. Last layer has an identity activation function and the training objective is an MSE loss function with L2 regularization. | # Solve with adam solver, newton's method or stochastic gradient descent<br>solver: [adam, lbfgs, sgd]<br><br># Activation function in the hidden layer<br>activation': [relu, identity]<br><br># FEATURES = 22<br>hidden_layer_sizes: [(100,), (len(FEATURES),), (len(FEATURES*2),), (100,100), (len(FEATURES*2),len(FEATURES*2))],<br><br># L2 Penalty<br>alpha: [0.0001, 0.001, 0.01, 0.1] |

Table 4 - Models and hyperparameters considered

Given that there are limited data samples, I used k-hold validation, reporting the average validation score over k folds of the training data. A custom coordinate descent algorithm was used to optimize the hyperparameters on this validation score. The results of this optimization are displayed in Table 5.

| **Model Type** | **Optimal Hyperparameters** | **Validation Risk** |
|---|---|---|
| Trivial Estimator | default | 16.816565147378 |
| OLS | default | 9.459243994315843 |
| Ridge | max_iter: None (auto)<br>alpha: 0.01 | 9.456763772222752 |
| Elastic Net | l1_ratio: 0.001<br>alpha: 0.001 | 9.478138604019149 |
| Theil-Sen Regression | default | 10.337839937315657 |
| **Neural Network** | **solver: sgd**<br>**activation: relu**<br><br>**# Two hidden layers, 100 nodes each**<br>**hidden_layer_sizes: (100, 100)** | **7.618993571809913** |

Table 5 - Model selection

**Test Performance**

The neural network, bolded in Table 6 above, had the best validation performance. We therefore selected this model to report the test performance. The test set has a total of 138 samples, kept separate from the training and validation sets.

| **Model Type** | **Parameters** | **Test Risk** |
|---|---|---|
| Neural Network | random_state: 628<br>max_iter: 1000<br>solver: sgd<br>activation: relu<br>alpha: 0.0001<br>hidden_layer_sizes: (100, 100) | **7.104656565791909** |

Table 6 - Test Performance

Therefore, the expectation of the model is to have a 7.10 point absolute difference between the predicted nutriscore and the “true” nutriscore.