Reverse Engineering Food Products

THE UNIVERSITY OF CHICAGO

Shivani Shah

code: https://github.com/shivanishah1/finalproject1.git

Background

My project looks at reverse engineering food product recipes, with the input as a product's nutrition label (nutrient breakdown plus ingredients list), and outputs the grams of each ingredient used (for a specific serving).

Methods

The nutrition label contains a product's ingredients and macro/micronutrient breakdown. Pairing this information with each individual ingredient's nutrient breakdown, a system of equations can be constructed (with the variables as the quantities of each ingredient). Gradient descent can be used to optimize this system (we can't directly solve it because there's no guarantee that the number of variables and equations are the same). Using the techniques/math we learned in class; I built my own optimization scheme. I explain the math and the decisions I made for the model (ex. penalty weighting, which variables to penalize, etc.) in the Jupyter Notebook(which is in the linked GitHub). Because my use of gradient descent depends on the initial inputted guess for values, I chose to send randomized inputs, perform gradient descent on each input, and output the solution with the lowest loss. The loss function provides a method to gauge the accuracy of the results, as it combines the equation results (equations evaluated using the solution variables) with the parameter results (all ingredients are >0 and are in decreasing order (as the given ingredient list on the nutrient label is in decreasing order)).

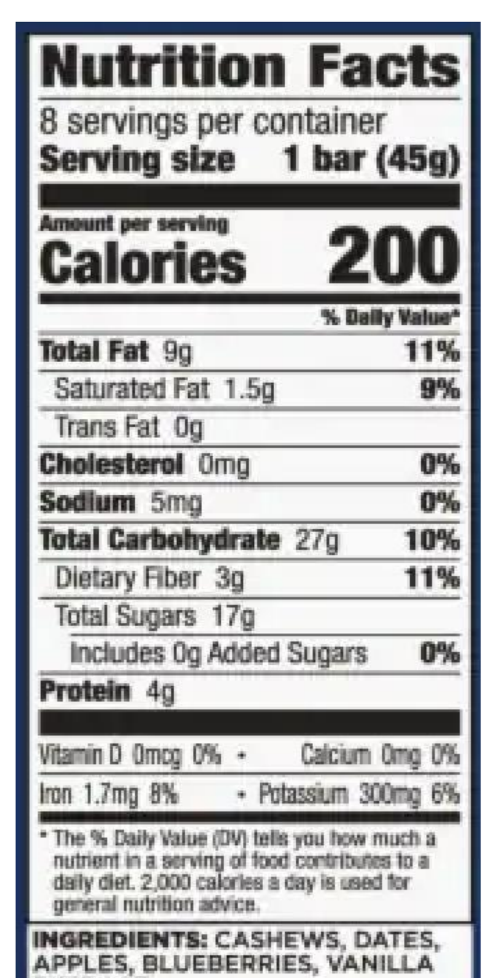
Results

Looking At Tate's Cookies:

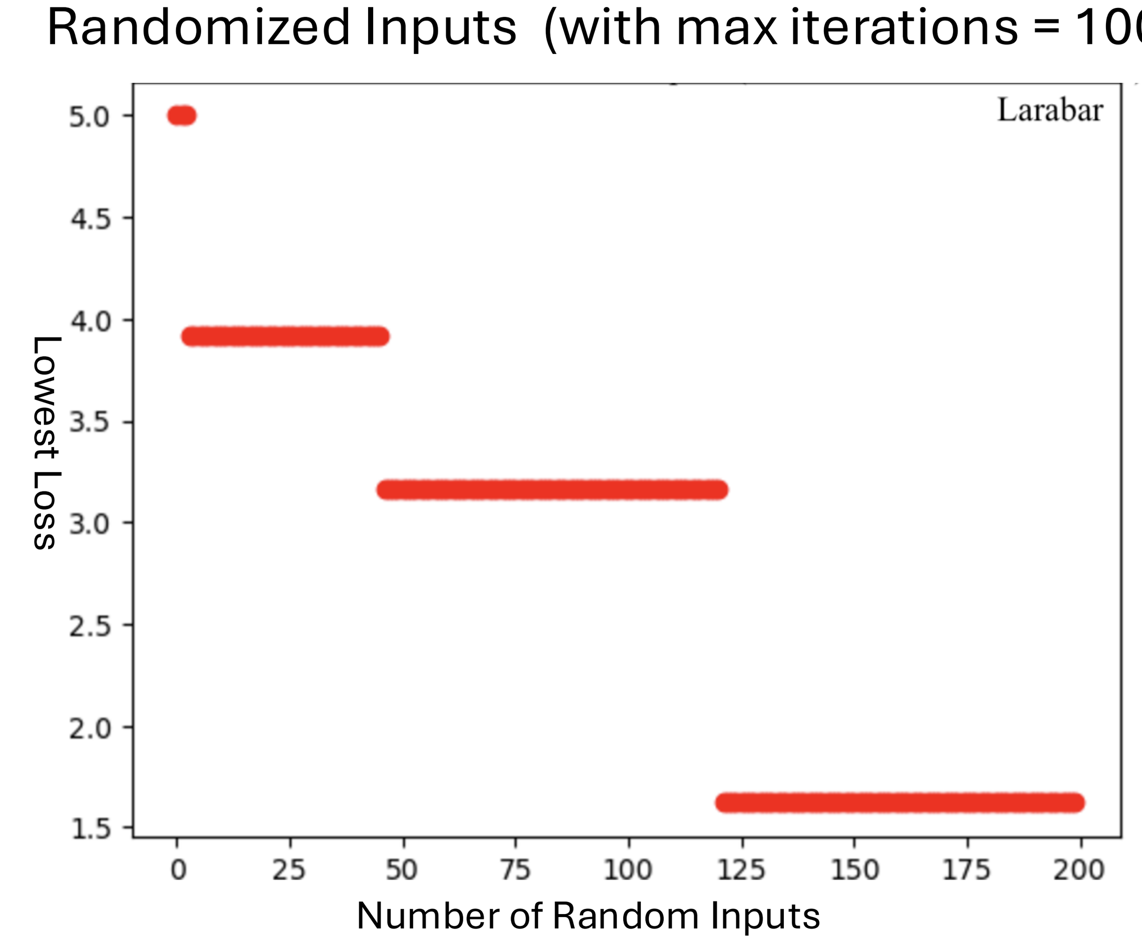
Results for a serving size of 4.5 dozens of 3in cookies:

Ingredients	Bon Appetite Copycat Recipe (g)	Predicted Results using Specific Input (g)	Predicted Results using Random Inputs (g)
Semisweet Chocolate Chips	340	389	280
All-Purpose Flour	240	276	234
Salted Butter	227	257	238
Cane Sugar	150	151	241
Brown Sugar	165	154	162
Eggs	100	109	109
Salt	5	.002	5
Baking Soda	5	.002	5
Vanilla	5	.001	5
LOSS (not measured in g)	(5g of water: not on the nutrition label)	1.15	.484

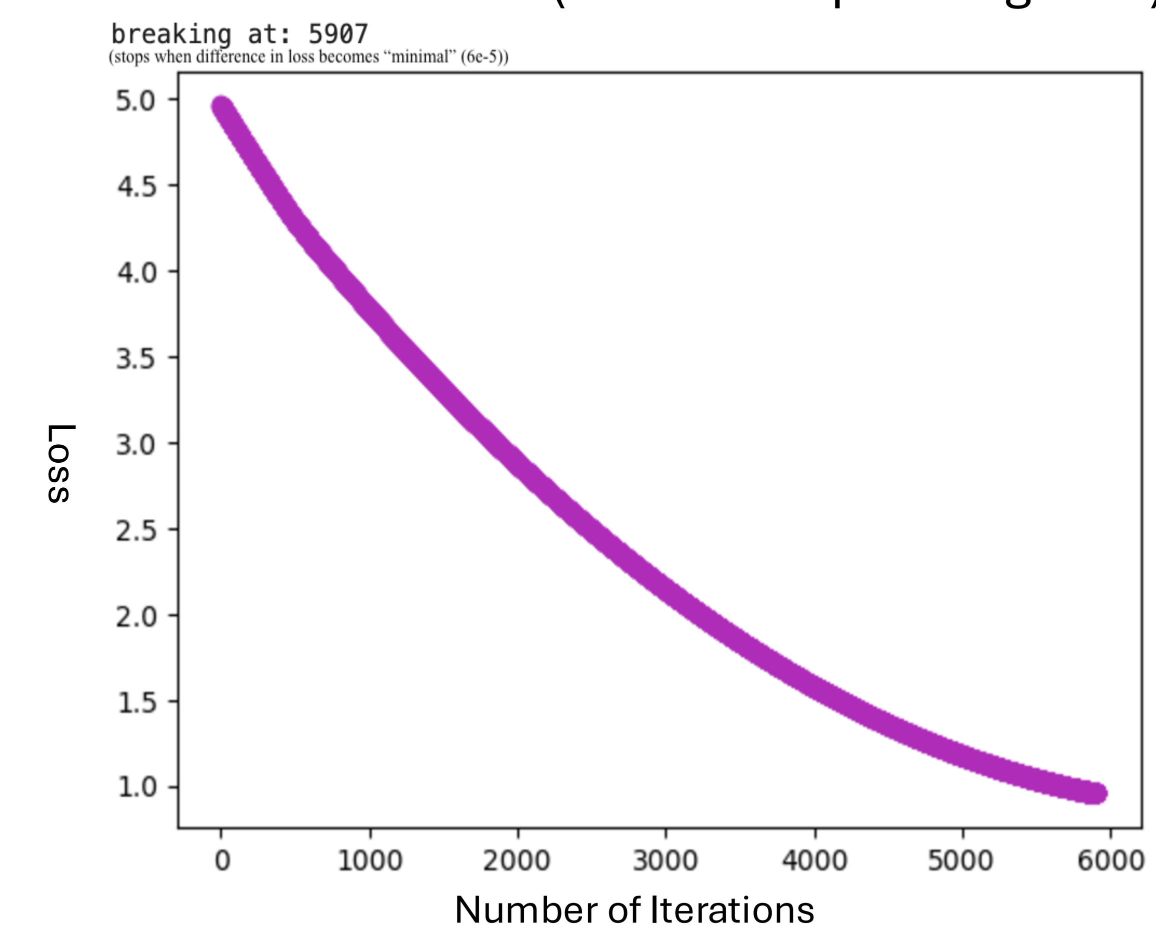
Predictions for Larabar Blueberry Muffin Bar: using heavy method (only looking at "heavy" ingredients (no vanilla extract))

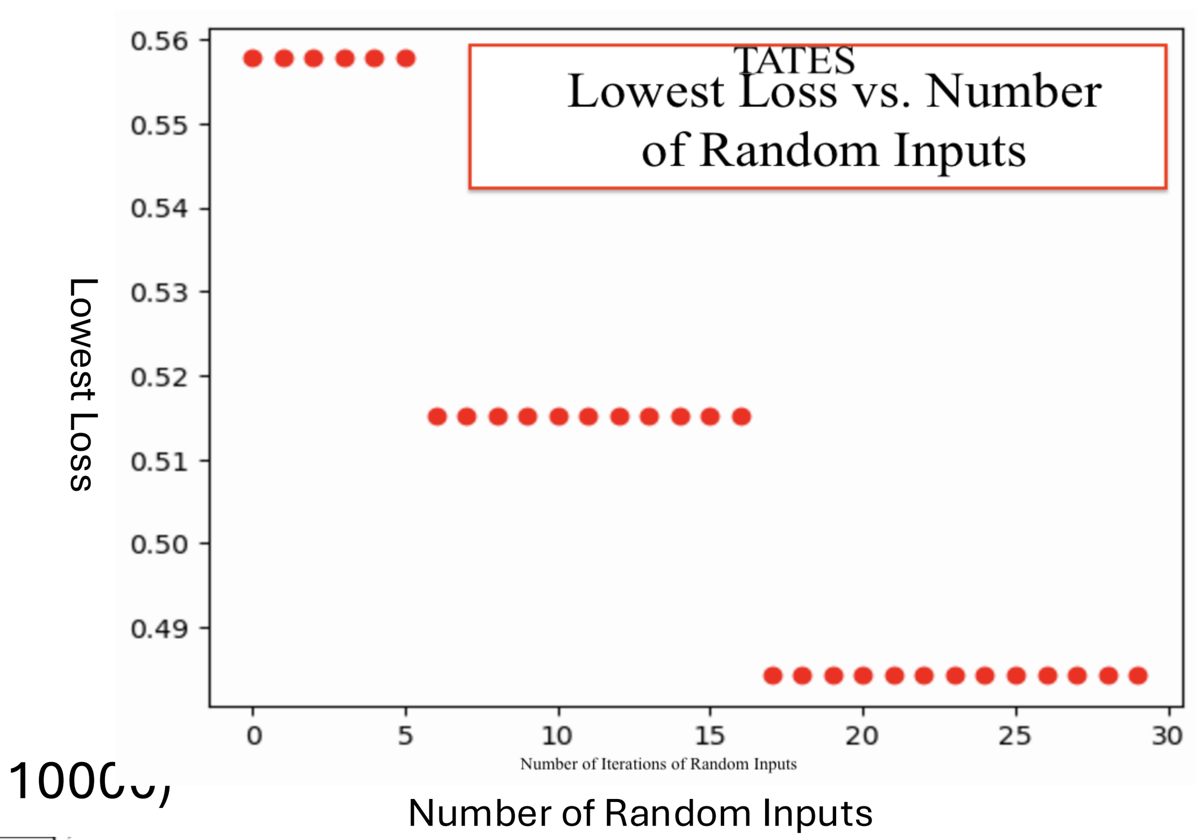


LARABAR: Lowest Loss vs. Number of Randomized Inputs (with max iterations = 10000)



TATE'S: Loss vs. Number of Iterations of Gradient Descent (with one inputted guess)





LARABAR: Lowest Loss Prediction for One Bar (45g) (loss of 1.54)

Ingredients	Amount (g)
Cashews	22
Dates	21.4
Apples	3.45
Blueberries	1.34

Limitations

The FDA has multiple rounding rules for the nutrition labels (leading to imperfect equations). In addition, nutrient breakdowns for each ingredient isn't exact because manufacturers don't release where they source their ingredients from.

Conclusion

Some ingredients (ex. vanilla extract, baking soda, food coloring, xanthan gum) have little nutritional value, barely contributing to any equation. For ingredients like these, I restricted their values. Therefore, my project works best for more simple/pure recipes that have a limited amount of extra add-ins like dyes or gums. However, the amount of these add-ins is the key secret for many manufacturers (Oreos, Goldfish, etc.), so if I combined my project with sensory analysis, I could get closer to finding these add-in amounts.

References

https://towardsdatascience.com/food-ingredient-reverse-engineering-through-gradient-descent-2a8d3880dd81 I read this article but didn't utilize any of the code in the linked GitHub/their method used built in libraries while I did not.

https://www.bonappetit.com/test-kitchen/cooking-tips/article/tate-s-cookie-recipe-taste-test (for the Tate's cookie recipe listed in the chart)

https://www.nutritionvalue.org/ (source for nutrient breakdown of ingredients)