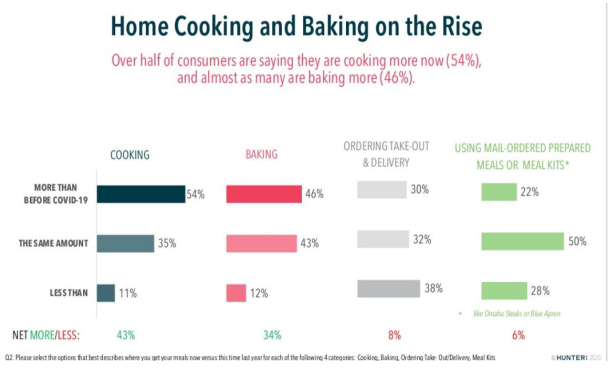
Rating Prediction for Recipes
Assignment 2 of CSE 158, Fall 2022

Sahil Dadhwal [A17222307]

Abstract*—The increasing quantity of online reviews regarding products and services has provided an important reference for people considering purchasing the product. Thus, it is imperative to implement a rating prediction to recommend which products will receive the best reviews in the market. In this project, I will conduct the rating prediction for the reviews of recipes. First, I will conduct an exploratory analysis of the data to view and report the basic statistics and any interesting findings. Next, based off of the results from the exploratory analysis, I will implement several models to identify what the most important feature is. I will purse the feature that seems to provide the best model. By using training, validation, and test sets, I tune the model parameters to maximize the performance compared to alternatives.

I. INTRODUCTION

There has recently been a vast increase in demand for home style cooking recipes. Due to the COVID-19 pandemic, government issued lockdowns have allowed people to work remotely from home which gives plenty of leeway for developing new hobbies. One hobby that seems to be commonly trending throughout most homes is cooking.



The increase in home cooking has cause a significant growth in demand for online recipe sites as people are choosing to purchase popular recipes. A rating prediction thus can further help these cooking recipe sites by recommending products that might get higher ratings.

There are a plentiful amount of resources to gather datasets in regards to users, the recipes they cooked, and their reviews. For example, the famous worldwide company, Warner Bros., overlooks the Food.com site. For this site, the average star ratings of the users that make the recipes is the primary factor that people use to decide if they would like to use that recipe. Typically for these types of companies, their revenue stems from positive reviews, thus it is imperative to construct a recommendation system that promotes the recipes that the user will most likely enjoy and leave a positive review.

In this assignment, I explore the datasets RAW_recipes and RAW_interactions, which contains over 200,000 recipes and over 1 million user interactions with the recipes, in order to predict the ratings of recipes using several different models in which I will determine the optimal model choice. Additionally, I will use a dataset that consists of a list of words that typically correlate to positive reviews. I also use the datasets interactions_train, interactions_validation, and interactions_test to maximize the performance by tuning the model parameters.

The first model is using the bag of words approach except with the ingredients. The rating of the recipes would be predicted using

the ingredients that make up the recipe itself. Logically speaking, the idea behind this was inspired by the phrase "less is more" when it comes to ingredients, so this model will test that theory on the given dataset. One issue that typically gets run into with the bag of words approach is that if the words in the bag of words have specific meaning together, then when they are split into the dictionary, the meaning gets lost. For example, if the text used to make the bag of words contained sarcasm, the bag of words would not register it. Luckily, we are working with ingredients, so this issue does not arise. The only issue that could potentially alter our results are related words. For example, if "Swiss cheese" is seen as a different ingredient than just "cheese". One way to combat this is by the use of stemming, also known as merging different inflections of words.

For the second model, I will train a simple predictor that estimates the rating from the number of times a word that exists in the dataset of positive words is used in the review. One downside maybe that the dataset contains positive words that do not exist in any of the reviews in the entire recipes database. To counteract this, I will create a set with every word of all the reviews and get rid of any words in the positive words database that does not exist in any of the reviews.

$$\text{star rating} \simeq \theta_0 + \theta_1 \times [\text{number of [positive\_words]}]$$

For the third model, will try a more naïve approach and train a model that predicts a rating based off of the length and the number of exclamation marks in the review. I am expecting this model to do poorly, the reason I am running this model as predictor is so that I can compare the results with model 2. Model 2 uses a similar method as Model 3, as in Model 2 the rating is predicted by the feature which is the number of times each individual word in the positive words dataset is used in the review whereas in Model 3, the rating is simply predicted by the length of the review and the exclamation point count.

$$\text{star rating} \simeq \theta_0 + \theta_1 \times [\text{length}] + \theta_2 \times [\text{number of '!' characters}]$$

In this report, I predict the overall rating for the recipes in the dataset RAW_recipes. I will use the mean squared error to evaluate the performance of my model. I will also implement pipelining such that my model chooses the optimal lambda value for the validation pipeline. To obtain the optimal mean squared error, I split my dataset into 3 sets. A training set, a validation set, and a test set in order to confirm the model's accuracy. For Model 1 and Model 2, we are testing diverse features and implementing two different methods of approach. For Model 3, it is closely correlated to Model 2 and expected to be worse. Model 3 is used a naïve and simplistic model used for comparative purposes.

II. RELATED WORKS

In this ever-so expanding world of science and technology, is seems as though researches have proposed diverse methods and implementations for predicting the ratings of an item given their various features. All of the strategies and methods seem to all also have fair justifications as to why it is the best one to use.

For study [1], this researcher discussed different models and approaches they made for creating a rating predictor for beer. I based my report off of the formatting of their study.

For study [2], this contains data and charts of the influxes of the growth of interest in home cooking since the start of the COVID-19 pandemic.

For the dataset contained in source [3], the creator published a public database of a list of words which signify positive reviews.

An additional source used is that of a Stanford student researcher [4] who studied the same database as me and created their own recommender system to determine the ratings. The methods and models they used vary from mine, but their research is noteworthy and valid.

For equations and simple terms and definitions I referred to the [5] Personalized Machine Learning textbook.

Lastly, in source [6], this contains the main database for my research. It is a Food.com Recipes and Interactions database from Kaggle and it provided me with the bulk of the data used to create my models.

III. DATA ANALYSIS

From the dataset, there are over 200,000 recipes and over 1 million user reviews. Due to compiler time, I had to truncate the majority of the user reviews to 5,000 as that is the maximum my device can handle. The first 4,000 elements are reserved for the training set, the next 500 elements are reserved for the validation set, and the final 500 elements are reserved for the testing set.

Firstly, locally stored the ratings for each user and recipe pair from the RAW_interactions dataset. I then stored the ratings into the main dataset, as shown in the following image as a ratings column.

Similarly, using the recipe id's gathered from the RAW_interactions dataset, I locally store the ingredients for the recipe corresponding to the recipe id's by using the content of the RAW_recipes dataset. Then, I store the ingredients of each recipe into the main dataset, as shown in the following image as an ingredient's column.

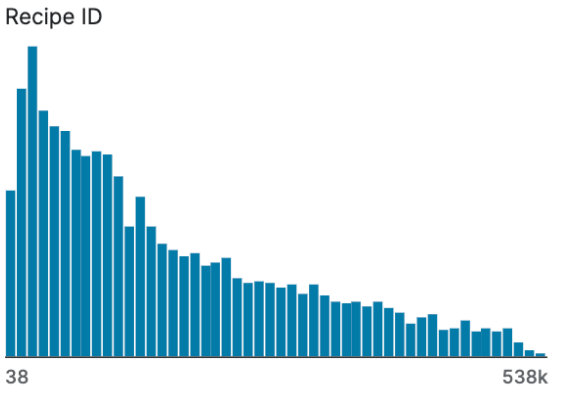
	user_id	recipe_id	date	rating	u	i	ingredients	ratings
0	2046	4684	2000-02-25	5.0	22095	44367	[flank steaks, honey, chipotle chiles in adobo...	5
1	2046	517	2000-02-25	5.0	22095	87844	[lamb, hamburger, eggs, milk, quick oats, oil,...	5
2	1773	7435	2000-03-13	5.0	24732	138181	[onions, carrots, corned beef brisket, malt vi...	5
3	1773	278	2000-03-13	4.0	24732	93054	[frozen spinach, eggs, onion, cream cheese, pe...	4
4	2046	3431	2000-04-07	5.0	22095	101723	[butter, leeks, fresh parsley, tomatoes, montr...	5
...	...	...	...	...	...	...	...	...

After we append the 2 new columns of ingredients and rating, we will also do a similar process for the review. This part is specifically used for Model 2 and Model 3, as we will need quick access to the user reviews.

So, using the reviews gathered from the RAW_interactions dataset, I locally store the them as they correspond to the user and recipe pair in the main dataset. Then, I store the reviews of each user and recipe pair into the main dataset, as shown in the following image as the new review column.

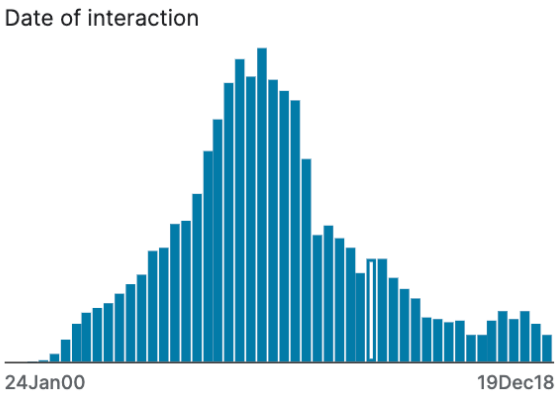
	user_id	recipe_id	date	rating	u	i	ingredients	ratings	review
0	2046	4684	2000-02-25	5.0	22095	44367	[flank steaks, honey, chipotle chiles in adobo...	5	this is absolutely delicious. i even served i...
1	2046	517	2000-02-25	5.0	22095	87844	[lamb, hamburger, eggs, milk, quick oats, oil,...	5	thought this was terrific!
2	1773	7435	2000-03-13	5.0	24732	138181	[onions, carrots, corned beef brisket, malt vi...	5	easily the best i have ever had. juicy flavor...
3	1773	278	2000-03-13	4.0	24732	93054	[frozen spinach, eggs, onion, cream cheese, pe...	4	a little greasy, but a huge hit with the guests.
4	2046	3431	2000-04-07	5.0	22095	101723	[butter, leeks, fresh parsley, tomatoes, montr...	5	leeks on a pizza?! it was really delicious. ...
...	...	...	...	...	...	...	...	...	...

In regards to the ratings prediction, we can observe that number of users that tried each recipes. From this basic statistic, on average, users tend to try the recipes with the id's of around 130,000. Perhaps this can help strengthen our model as we make a predictor that has a feature which uses how many users made the recipe play a factor on the rating prediction.



Another statistic to keep in mind is the date the recipes were interacted with. As shown in the image below, the season that the users tend to interact with the recipes more

peaks in summer. Specifically, the mean month is May. From this, we can also consider the season being a feature as to predict what the user might rate the recipe. If it is summertime, perhaps the user would rate the winter recipes poorly, or if it is winter, the user might rate the summer recipes poorly. Building a recommender system based off of the seasons does not seem to far-fetched of an idea and it is often implemented in the real world.



IV. PREDICTION MODEL

Before applying any of the models described, I first must pre-process the dataset. Once that is completed, I will demonstrate the prediction models described to predict the ratings of the recipes.

A. Evaluation

I choose to use mean squared error (MSE) to be the evaluator of our model's performance. A desired MSE value is expected to be low.

The Mean Squared Error between a model fθ(X) and a set of labels y is defined as:

$$\text{MSE}(y, f_\theta(X)) = \frac{1}{|y|}\sum_{i=1}^{|y|}(f_\theta(x_i) - y_i)^2,$$

In other words, the average squared difference between the model's predictions and the labels.

A. Regression

For our models, I will use various regression models to predict the rating. To name a few, I will look at linear regression and also regularized regression using ridge. I will be pipelining as well to ensure that my recommender system is calculating the optimal lambda value each time.

(1) Linear Regression

One of the simplest ways to determine a relationship between the features X and labels y would be through their linear relationship.

$$y = X\theta.$$

(2) Logistic Regression

Logistic regression can also be applied to my models, as logistic regression associates positive values of x with positive labels of y and negative values of x with negative labels of y.
That can be applied to my model which uses a list of positive terms to determine which reviews will correspond to a positive rating. It would be beneficial because not only will there be positive values associated for the positive words that were found, but also negative values for the reviews that had negative words associated with them.

$$\frac{1}{|y|}\sum_{i=1}^{|y|}\underbrace{\delta(y_i = 0)\delta(x_i \cdot \theta \le 0)}_{\text{label is negative and prediction is negative}} + \overbrace{\delta(y_i = 1)\delta(x_i \cdot \theta > 0)}^{\text{label is positive and prediction is positive}}$$

V. EVALUATION

In this section of the report, I will evaluate and compare the performances of the 3 different models. Whether the models accurately or inaccurately predict the data is irrelevant, the reasoning behind the great or terrible accuracies will be discussed and explained in the conclusion.

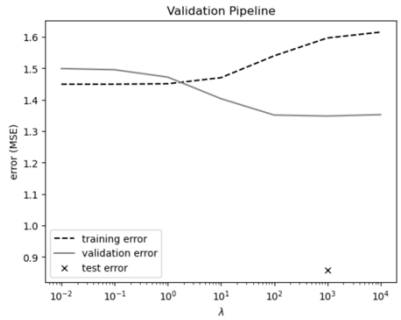
A. Regression

Model 1a: *Bag of Ingredients*

For this model, we used the bag of words approach to create the recommender system. The performance of this model is shown in the figure below. This model did not perform as expected.

I pipelined to calculate the optimal lambda value, but even then, the optimal MSE value is roughly 1.3. So the predictor for Model 1a is inaccurate.

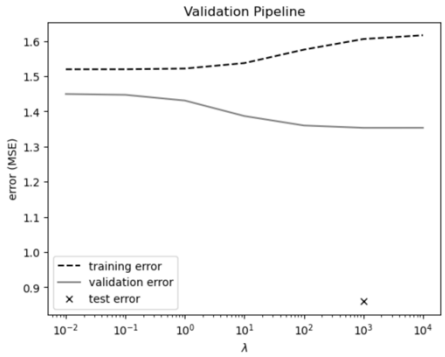
```
l = 0.01, validation MSE = 1.498915852929805
l = 0.1, validation MSE = 1.4952852430447277
l = 1, validation MSE = 1.471770443695989
l = 10, validation MSE = 1.4031366690193852
l = 100, validation MSE = 1.3510362865907537
l = 1000, validation MSE = 1.3480169976537397
l = 10000, validation MSE = 1.3524189732804106
```



Model 1b: *Bag of Ingredients(stemming)*

For this model, we used the bag of words approach to create the recommender system. This model is closely related to Model 1a, except with the implementation of stemming. The performance of this model is shown in the figure below. This model also did not perform as expected. Similarly as Model 1a, I pipelined to calculate the optimal lambda value, but even then, the optimal MSE value is roughly 1.3 which is almost the same as Model 1a. So the predictor for Model 1b is inaccurate.

```
l = 0.01, validation MSE = 1.4491815965227033
l = 0.1, validation MSE = 1.4468676562818437
l = 1, validation MSE = 1.4306097431824087
l = 10, validation MSE = 1.3865982089258952
l = 100, validation MSE = 1.3597148702377106
l = 1000, validation MSE = 1.3530743720092104
l = 10000, validation MSE = 1.35313839429604
```



Model 2: *Positive Words*

For this model, we use a dataset that contains words in reviews that correlate to positive star reviews and use that to make our recommender system. Unfortunately, the positive words dataset is far to large to be implemented on my local device. However, a teaching assistant recommended I still mention this non-completable model, as it has the potential to be a great recommender system.

Just by looking through random components of the data manually, it seems as though this Model would have been really accurate, as most of the high reviews contained positive words, and the negative reviews did not contain positive words. Again, this does not prove for certain that it would have made a good recommender system there could be a lot of outlier cases I missed.

Model 3: *Review Length and Term*

For this model, I trained a simple predictor that has the two features:

(1) the length of the review

(2) the number of "!" characters

For feature 2, I also opted to test different content such as "thanks", "great", "yum", etc. However, that had minimal effect on the MSE value. The MSE value of all the different features ranged from 1.4 to 1.5.

The image below shows the MSE result of the original 2 features. As expected for this Model, it resulted in a poor MSE result.

```
MSE = residuals[0] / len(y)
MSE
```

```
1.4912253697682007
```

VI. CONCLUSION

The model with the best performance is Model 1a. Although there is still potential for improvement. I will now list some things that could have been potential issues for my models, resulting in sub-par results.

Model 1a: *Bag of Ingredients*

For this model, we used the bag of words approach to create the recommender system. Although this is the best model, it did not perform as expected. For this model to be improved, a number of implementations could be implemented. One example of something that could be changed is what is contained in the bag of words. For example, even with stemming the term “salt” and “salt and pepper” are2 separate terms, whereas in reality they can be considered the same. I think that means the data is slightly flawed as it includes multiple cases like this where 2 ingredients are list as 1. So fixing that issue could potentially help the performance of Model 1a.

Model 1b: *Bag of Ingredients(stemming)*

For this model, it has the same reasoning as Model 1a. The only difference is we use stemming. However, the problem that arose with my implementation is that I assumed stemming would truncate words like “Swiss cheese” to “cheese”, bit it does not. Instead it truncates words like “cheeses” to “cheese”. So a solution to perhaps better the accuracy for Model 1b could be to import a new stemming library which focuses of stemming words like “Swiss Cheese”. This can potentially increase the accuracy.

Model 2: *Positive Words*

For this model, we use a dataset that contains words in reviews that correlate to positive star reviews and use that to make our recommender system. As mentioned before, the positive words dataset is far to large to be implemented on my local device. A teaching assistant recommended I still mention this non-completable model, as it has the potential to be a great recommender system.
I will still recommend some possible solutions to be able to get this to run.

The reason that this Model does not work is because the positive words database is far to large, so a simple yet time intensive solution could be to remove the words in the database that do not appear in our main dataset at all. The only issue with this is that we are potentially deleting positive words that could be in the test data set. There are pros and cons to this, but it has the potential to get this Model to run!

Model 3: *Review Length and Term*

For this model it was expected to result in a poor performance result. I trained a simple predictor that has the two features of the length of the review and the number of “!” characters.
One way to potentially increase the accuracy of the model is to try to use a word for feature 2 that is most common in the positive reviews instead of an exclamation point. However, I already attempted this with terms such as “thanks”. So another approach could be to train the model to fit a polynomial function to estimate the ratings based on the exclamation point feature, and to just get rid of the length of the review feature.

We can fit polynomials from degree one to ten and then determine which polynomial gives the most optimal MSE value. That is one way we can improve this model, but again, this model was intended to be poor so it can be used as a comparative model.

VII. REFERENCES

[1] Jiao, Xun and Fang, Zhou. "Rating Prediction for Beer" https://cseweb.ucsd.edu/classes/wi15/cse255-a/reports/fa15/042.pdf

[2] Shoup, Mary Ellen. "Survey: Cooking More at Home Could Become the New Normal Post-Pandemic." *Foodnavigator*, William Reed Ltd, 15 Apr. 2020, https://www.foodnavigator-usa.com/Article/2020/04/15/Survey-Cooking-at-home-will-become-the-new-normal-post-pandemic.

[3] Ruikar, Amey. https://github.com/AmeyRuikar/sentiment-analysis/blob/master/positive%20words.csv

[4] Hensley, Sarah. "CS 229 Final Report: Recipe Rating Prediction (Natural Language)" http://cs229.stanford.edu/proj2019aut/data/assignment_308832_raw/26626258.pdf

[5] McAuley, Julian. "Personalized Machine Learning ." *Personalized Machine Learning*, https://cseweb.ucsd.edu/~jmcauley/pml/

[6] Li, Shuyang. "Food.com Recipes and Interactions." *Kaggle*, 8 Nov. 2019, https://www.kaggle.com/datasets/shuyangli94/food-com-recipes-and-user-interactions.