Recipes Ratings Classification

Jose Luis Estrada, Jimmy Nguyen, and Ashutosh Singh

Team 3

ADS-504

Shiley-Marcos School of Engineering

University of San Diego

Table of Contents

Define Purpose

The purpose of this report is to compare different machine learning models that predict the rating of each customer that uses the recipes from Food.com. A variety of numerical and categorical attributes are considered in this project. Yet, some of these variables could be noisy data. The outcome of this exercise will prove which variables can work to make an accurate prediction.

Background

Food.com is a website that allows users to share and copy kitchen recipes. Formerly known as Genius Kitchen, the company collected data covering 18 years of user interactions and uploads. The datasets, acquired from Kaggle Machine Learning and Data Science Community website, includes 180K+ recipes and 700K+ recipe reviews. Additionally, the recipe reviews dataset include five attributes and the recipes dataset includes 12 attributes. These datasets were used on a research project called: Generating Personalized Recipes from Historical User Preference, which aims to generate personalized recipes from incomplete input specifications and user histories.

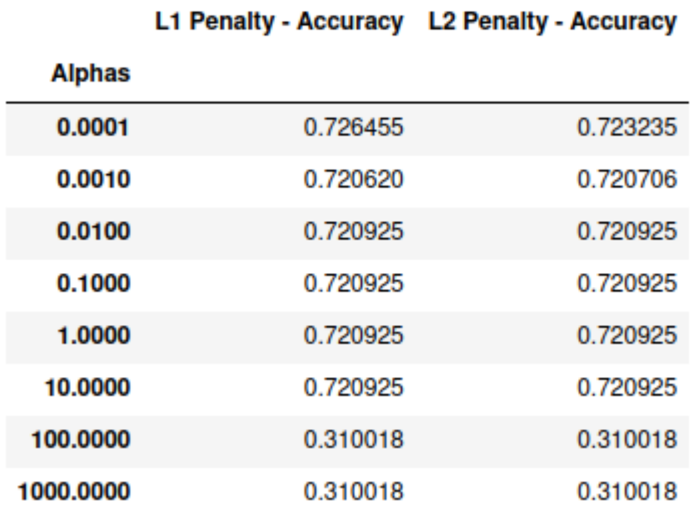
Baseline Model - Data Preprocessing

For the initial modeling phase, the first step was to retrieve all the reviews from each recipe in this data set. Then the next task is to determine the outcome, such as predicting the average rating of each recipe as a regression problem. On the other hand, text analysis and feature selections can be applied on each review as a multi-class classification problem. The decided outcome was the latter. This data set originally contained 169 missing values in the *review* column. Since the number of reviews are over 1,000,000 rows, dropping missing values is acceptable. The next step is to gather all the text data in the *review* column and perform a statistical measure called *term frequency-inverse document frequency* in order to evaluate the relevance of a word in a review in a collection of reviews. However, due to this new data transformation, the data now suffers from the curse of dimensionality with over 100,000 features. Also, this includes removing stop words in the English language. So a way to combat this was through feature selection using the *SelectKBest* function from *scikit-learn* package. This function simply scores each word and removes all irrelevant words except the highest scoring words. In this case, the hyper-parameter *K* was chosen to be 20. The motive was to find the top 20 relevant words. An example of these words were "awful", "disgusting", or "bland".

Baseline Models - Regularized Linear Classifiers

The baseline model of choice is using a regularized linear classifier with stochastic gradient descent, while estimating the gradient of the log loss function at each sample at a time. As so, these regularized linear models would then be logistic regression. These models are trained by varying between the absolute norm *L1* and squared euclidean norm *L2 as a* regularizer to control overfitting. These penalties will be trained in conjunction with fine-tuning for the optimal learning rate, or alpha. The steps for searching for the optimal hyper-parameters include a pipeline to fit the stochastic gradient descent classifier with the log loss function, iterating between the *L1* and *L2* penalty over a list of alphas in the range of 0.0001 to 1000 with a step size of 0.1. The choice for performance metric is accuracy because this is a multi-class classification problem. Due to accuracy being a misleading performance metric, it is better to use *K-folds cross validation* as a better approximation of accuracy instead of a random training and test-set split. The results for these linear models are presented in the following table below.

Table 1. Baseline Models - Performance Table

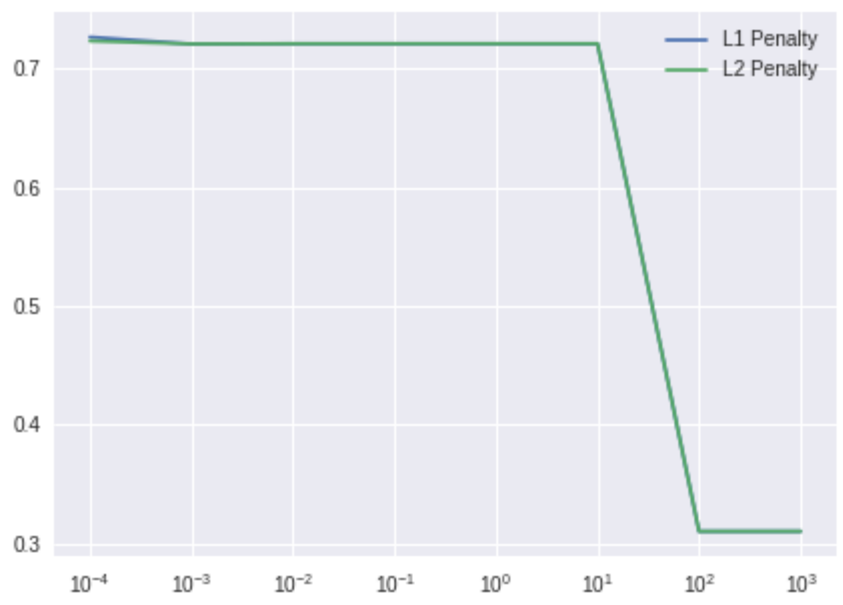


Alphas	L1 Penalty - Accuracy	L2 Penalty - Accuracy
0.0001	0.726455	0.723235
0.0010	0.720620	0.720706
0.0100	0.720925	0.720925
0.1000	0.720925	0.720925
1.0000	0.720925	0.720925
10.0000	0.720925	0.720925
100.0000	0.310018	0.310018
1000.0000	0.310018	0.310018

Note. This table displays the accuracies of each learning rate denoted as alphas between two regularizers: L1 and L2.

The deciding factor for finding the optimal hyper-parameters is one that yields the highest accuracy. The following graph below shows the accuracy performance of each penalty over a range of alphas.

Figure 1. Baseline Models - Performance Graph



Note. This graph shows a comparison of the penalties over the learning rates in the range of 0.0001 to 1000 with a step size of 0.1.

Besides the initial learning rate of 0.0001, the graph above shows that there are no other differences in performance between the choices of penalty. Thus, the selected learning rate was 0.001 and the regularizer chosen is the L1 penalty with the highest accuracy of 72.6%. Meanwhile, one can argue that the L2 penalty can be selected also with an accuracy of 72.3%.

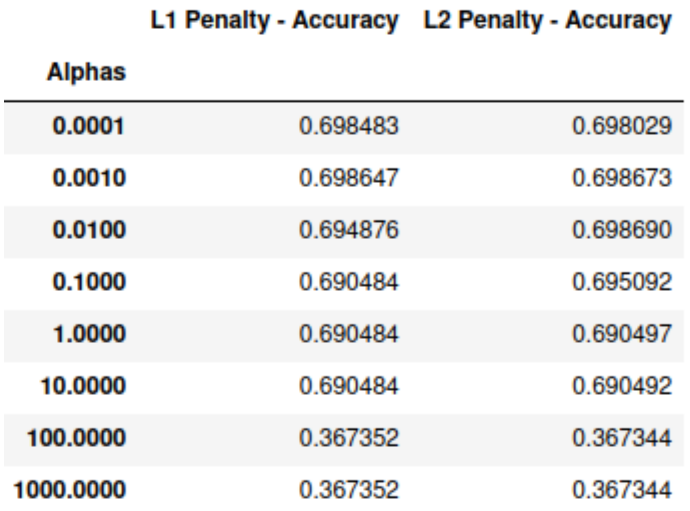
Baseline Model - Feature Engineering to Improve Performance

With an accuracy of 72.6%, further improvements can be possible through feature engineering to extract new information for the regularized linear models. The previous data set, *RAW_interactions.csv,* only contains information about the recipe id, the reviews each recipe received, and the ratings. However, there was another data set available called *RAW_recipes.csv,* which included information about each recipe, such as the number of steps, ingredients, or the number of minutes it took to cook a recipe. There were other columns in this data set that have Python lists of data for every row. For example, the column *nutrition* contains nutritional values and the *ingredients* column contains material needed to cook a recipe. Thus, this may be valuable and rich information about each recipe, which may or may not have a factor on how a recipe may be rated. Thus, in order to extract this information, both data sets were merged using an *inner join* based on the recipes' id column from each data set. The reasoning behind an inner join is to have information about each recipe and their reviews. Now that the data set is merged, the columns *nutrition* and *ingredients* will need to go through feature extraction by parsing each Python list in each row to convert these information into new columns. This was possible by parsing the text string from each list and then converting them to float values in new columns. These new columns are now called *calories, total fat, sugar, sodium, protein, saturated fat,* and *carbohydrates.* The final data set is a combination of these nutritional values, the number of steps, number of minutes, number of ingredients, and the previous top 20 words used in the baseline model.

Final Model - Regularized Linear Model

The final modeling steps followed the previous steps outlined in the baseline model. A range of the learning rate along with the choice of *L1* and *L2* was tuned to find the optimal hyper-parameters that yield the highest accuracy. The performance of these models are listed in the following table below.

Table 2. Final Models - Performance Table

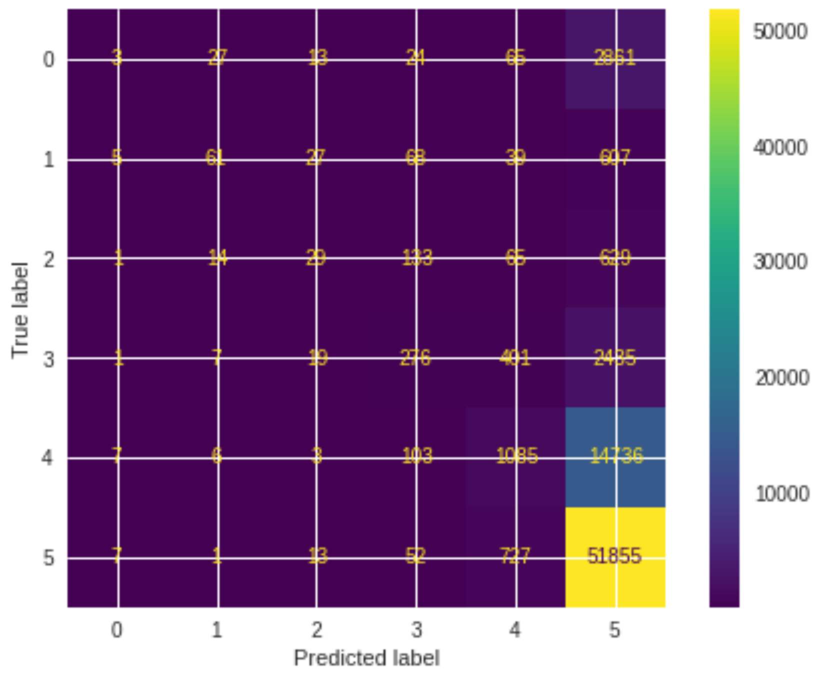


Alphas	L1 Penalty - Accuracy	L2 Penalty - Accuracy
0.0001	0.698483	0.698029
0.0010	0.698647	0.698673
0.0100	0.694876	0.698690
0.1000	0.690484	0.695092
1.0000	0.690484	0.690497
10.0000	0.690484	0.690492
100.0000	0.367352	0.367344
1000.0000	0.367352	0.367344

Note. This table shows the performance of the regularized linear models on the new merged data.

The performance on the new merged data did not improve. In fact, the performance on the merged data actually dipped 3% in accuracy, if the selected model was using a learning rate of 0.0001 with either the *L1 or L2* penalty. This means that the previous data set containing only the reviews of each recipe performed slightly better without the recipe information added. Another interesting note is that these new features may have added noise and complexity to the regularized linear classifiers. Thus, a decrease in performance may be due to non-linear decision boundaries that the regularized linear classifiers are unable to learn. In order to analyze the reasoning behind this, further analysis can be done by testing the model on a 67% training and 33% test split. The final results are shown below in a confusion matrix.

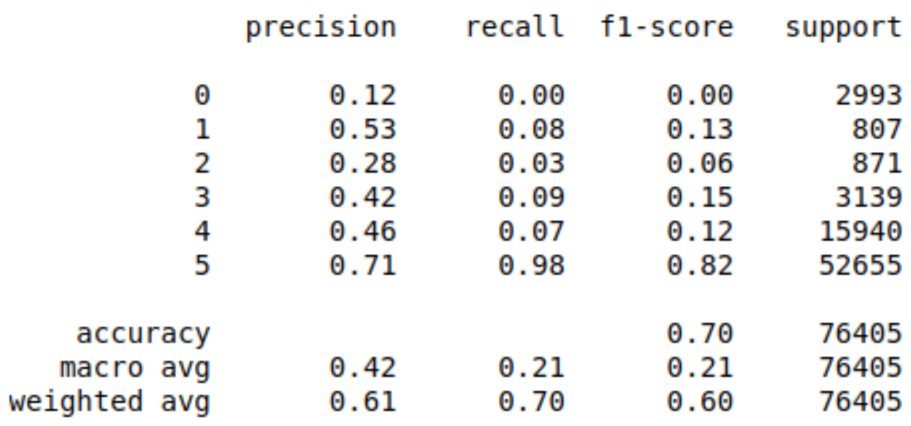
Figure 2. Final Model - Confusion Matrix



Note. This confusion matrix graph shows the number of predicted labels against the true labels.

The graph above shows that the regularized linear models are dominated by high positive ratings of class 5. This may suggest the models are more likely to classify each recipe with a high rating of 5. For example, recipes where the true labels are 0, the model only predicted 3 ratings that were actually 0. Meanwhile, the model is apparently biased because it incorrectly predicted 2,861 recipes with a rating of 5 instead. A performance report can be found below for further analysis on each classification.

Table 3. Final Model - Performance Report



```
              precision    recall  f1-score   support

           0       0.12      0.00      0.00      2993
           1       0.53      0.08      0.13       807
           2       0.28      0.03      0.06       871
           3       0.42      0.09      0.15      3139
           4       0.46      0.07      0.12     15940
           5       0.71      0.98      0.82     52655

    accuracy                           0.70     76405
   macro avg       0.42      0.21      0.21     76405
weighted avg       0.61      0.70      0.60     76405
```

Note. This table shows the performance report for each class predicted. The overall accuracy is 70% using the final model.

The goal of this project is to predict the rating of each recipe. However, with a final performance of 70% accuracy, this may not be a dependable result for deployment into any production system yet. Thus, in pursuit of improving performance, other methods are explored with different pre-processing techniques and non-linear models, such as an ensemble model and a multi-layered neural network.

Random Forest

A Random Forest classifier algorithm is employed to perform the automatic classification of recipes in cuisines as the first step of the proposed system. Some well-established text preprocessing and feature selection methods (like TF-IDF and Bag of Words techniques) have been employed to perform automatic recipe cuisine classification. Random forest was adopted as the recognition module and the best combination of feature selection and classifier has been selected after the experiments as the resulting system.

In this project, we adopted the Random Forest Classifier (RFC) algorithm as the recognition module for our proposed multi-label recipe classification system. In RFC, a set of Decision Tree Classifiers (DTCs) is implemented as the core decision-making mechanism. The

Random Forest Classifier has used selected data using selectkbest recipe/interaction dataset to identify top 20 features for 7 different text columns. Additionally, we have used ‘recipe_id’, ‘review’ & ‘rating’ columns from the interaction dataset and 7 different columns from the review dataset and joined the recipe and interaction dataset using recipe_id and ran an RFC model on them. RFC creates a set of decision trees from randomly selected subsets of the training data to decide the final class of a given testing sample.

With the random forest model run on the imbalanced dataset, it yielded an average of 72% accuracy using 10-fold cross-validation. The overall objective of the project is to predict the rating of each recipe and give an accuracy of 72% which is above average but this model can be enhanced further by making the dataset balanced and using more advanced techniques with other models to get better outcomes. In search of better accuracy, we have performed a neural network which is the next model.

Neural Network

The last approach for this work was to test a neural network model to provide a more accuracy increase compared to the baseline model by using a more complex non-linear model. To take advantage of the complexity of the model, both datasets with interactions and recipes were merged to have one complete dataset with 17 attributes. After merging them, the nutrition attribute, an attribute with an array of seven attributes, was separated into seven new variables, including information about nutrition such as calories, sugar, saturated fats, sodium, total fat, protein, and others. This strategy will help the neural network to be trained while taking into consideration more variables.

Furthermore, changes were made to the preprocessing steps to include the variables that use text such as name, tags, steps, description, and ingredients. For these variables, they were imputed into a vectorizer and used a Kbest = 50. By acquiring the top 50 words for each variable, they were included in the final dataset that was trained with the neural network model. Thus, the dataset with independent variables concluded with 265 attributes.

The target variable was transformed using the LabelEncoded, similarly to the baseline model, to treat the rating variable as a multi-classification label. The number of instances was different for different ratings. Most of the interactions gave ratings equal to five, and the least was equal to one. To avoid memorization of frequency on the model, the labels were balanced by making them equal to 9K instances of each label.

After training and testing the neural network, the results showed an accuracy of ~18% on both the trained and test sets. These techniques provided a worse solution than the baseline model. Different hypotheses were considered to make further improvements considering the preprocessing steps mentioned before, and another test was made without balancing the labels. This strategy provided a better accuracy of ~72%, but realistically is not representative of a model that can be used with different datasets since the model is memorizing the frequency of each label.

Conclusion

In summary, the regularized linear model performs at an average accuracy of 72% with only the review text data. However, upon the assumption that feature engineering would improve the baseline performance, it was interesting to see that it decreased in accuracy by 3%. This may be due to the complications of adding noise and complexity instead of relevant information. Future improvements may include different topic modeling techniques to analyze the sentiment or the grouping of phrases that have a relationship with positive or negative ratings. Other improvements can include different feature selection techniques, such as *Principal Component Analysis* or *t-Distributed Stochastic Neighbor Embedding* on each food review.

Although we have got an accuracy of 72% with random forest, it can still be enhanced further by tuning the parameter which is the best way to enhance the accuracy of the model. Additionally, adding more variables or removing unwanted variables while making data balanced during preprocessing can further enhance the model accuracy.

It seems that some of the preprocessing steps were taking the right approach, but accuracy was very low to consider using the model with a different dataset. Additional preprocessing steps should be made to further improve the accuracy of the model using the string variables, or the string variables do not help the model at all.