**How did you turn your text into features and why?**

First, I read the JSON file into a dataframe using Pandas library. Then, I converted the values under the ‘ingredients’ column from row of a list into a row of strings using the ‘apply’ and ‘lambda function’. I dropped two columns from the ‘df\_yummly’ data frame which were created while I was experimenting to convert the lists into strings, using ‘drop’ function from pandas DataFrame. Then, I split the main data into test and train sets which the train was 80 percent and the test was 20 percent using train\_test\_split library from sklearn. I transformed these strings under the ingredient’s column using Tfidf Vectorizer where for the train set, I applied ‘fit\_transform’ function and for the test set I only used ‘transform’ function. I used Tfidf vectorizer because it would allow me to do both fit and transform

**What classifiers/clustering methods did you choose and why?**

For clustering, I used k-means clustering. The reason was that k-means clustering has been used widely so it was relatively easier to understand. I used Silhouette score to determine the best number of clusters. Though the score for two clusters was slightly higher than the score for nine clusters, I chose to use nine clusters because I thought that using nine clusters would help the data points to be more organized. Then I passed the vectorized train data into my k-means model. To obtain the labels from the clusters, I used the ‘predict’ function from the k-means model where the vectorized train data was passed as a parameter.

For classifiers, I used logistic regression. Though I was not sure if the dataset would be linearly separable, I found it easier to implement. I imported ‘Logistic Regression’ library from sklearn.linear\_model. I used the ‘random\_state’ parameter because it would set seeds. I made the seed number 30 which could be the reason it is taking a very long time to run. I passed two arguments into the regression model which were the vectorized train data and the y\_train data I had from the train and test splitting.

The logistic regression model had run for nearly 3 hours before I decided to terminate it and try another classification algorithm, such as kNN. However, terminating the model took very long so I proceeded with KNN. I chose KNN because it would take relatively a smaller time than regression and it would also give the closest neighbors. However, at the end I wrote codes on how I would have proceeded if the regression model had completed running.

The vectorized train data set and the y\_train were passed into the kNN model where I had initialized with neighbors’ number of 7. I assigned the number on my own and I did not use any method to determine the accurate number of neighbors.

I had run into errors when I was trying to fit my input of three strings/ingredients into the k-means model because the model had nine clusters while my input was only three. I was hoping to pass the fitted input from kmeans into the kNN model, but it did not work out. The other error occurred when I was trying to pass the k-means predicted from my input into the kNN model because the kNN model would only allow a 2D array, not a 1D array.

I was able to pass the vectorized input array of ingredients into the ‘predict’ function from the kNN model. It returned an array of numbers that looked like labels which could be cuisine ID’s that were neighbor to the ingredients I had used as input. However, I was not able to use the cluster labels from the k-means model, which I wish I had managed to.

**Describe functions/code**

For the user input, I created a variable with a list of strings where ingredients were the strings. Then, I used Tfidf vectorizer to pass the train data into its ‘fit’ function only. Then I passed the input variable into ‘transform’ function from the model where the train data was fitted in. Then I used the ‘predict’ function from the k-means model with nine clusters to generate labels for the transformed input.

<https://stackoverflow.com/questions/42825655/using-predict-on-new-text-with-kmeans-sklearn>

<https://towardsdatascience.com/kmeans-clustering-for-classification-74b992405d0a>

<http://theprofessionalspoint.blogspot.com/2019/03/advantages-and-disadvantages-of.html>