A simple food recommendation system based on k-means clustering and knn regression

Introduction

I cook every meal for myself by paying attention to their details. For example, I use a small scale at home to weigh my ingredients and see how much protein, carbohydrate or fiber they have based on their nutrition label. Therefore, In this study, I use two models, one unsupervised and one supervised, k means and knn regression, to help me find my optimal foods. The goal of clustering is to see if the algorithm can find similar foods and cluster them for me together, so I can have a nice overview of my food choices. The goal of knn is to build a simple food recommendation system for myself given a target value to the algorithm.The libraries that I used are Pandas, and numpy for data manipulation, Matplotlib, seaborn, and plotly for visualization. I wrote both algorithms from scratch and compared my results with sklearn.The results match perfectly.

Dataset

I found a data set on kaggle that has 8789 data points (names of different foods), and 76 variables, such as the amount of macro nutrients such as carbohydrate, protein and micronutrients such as vitamins, minerals. The dataset is very messy and needs to be cleaned and prepared for my machine learning algorithms.
For data cleaning, First, I read the data and understand how the data is. This is an first step:

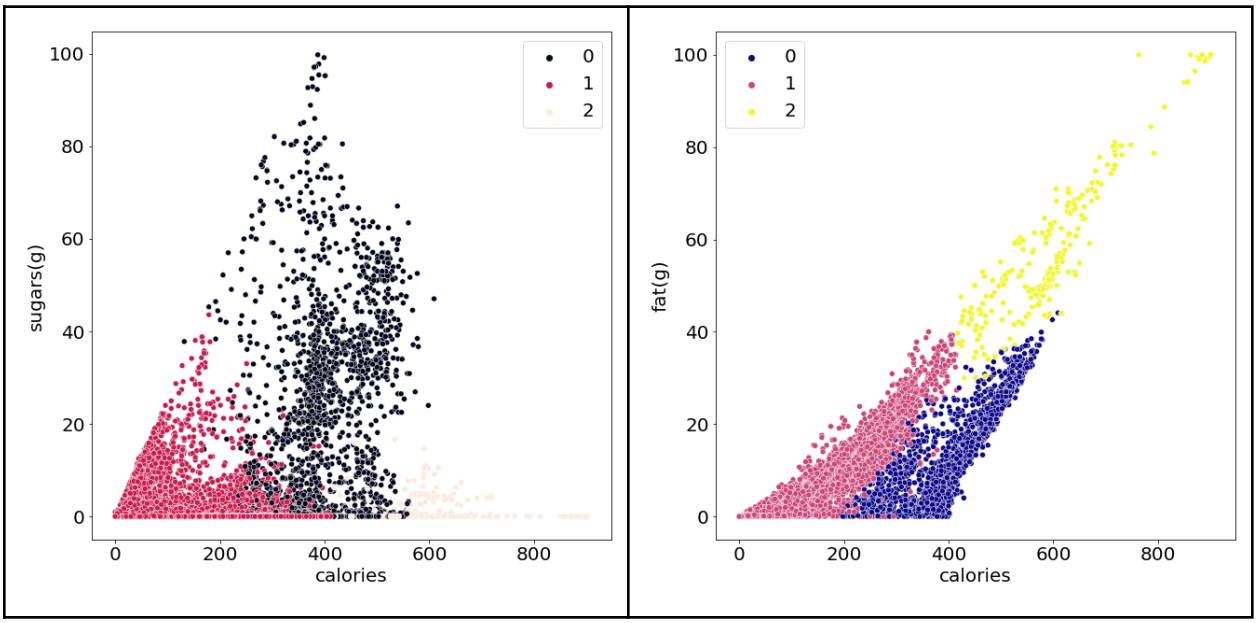
	name	serving_size	calories	total_fat	saturated_fat	cholesterol	sodium	choline	folate	folic_acid	...	fat	saturated_fatty_acids	monounsaturated_fatty_acids	polyunsa
0	Cornstarch	100 g	381	0.1g	NaN	0	9.00 mg	0.4 mg	0.00 mcg	0.00 mcg	...	0.05 g	0.009 g	0.016 g	
1	Nuts, pecans	100 g	691	72g	6.2g	0	0.00 mg	40.5 mg	22.00 mcg	0.00 mcg	...	71.97 g	6.180 g	40.801 g	
2	Eggplant, raw	100 g	25	0.2g	NaN	0	2.00 mg	6.9 mg	22.00 mcg	0.00 mcg	...	0.18 g	0.034 g	0.016 g	

3 rows × 76 columns

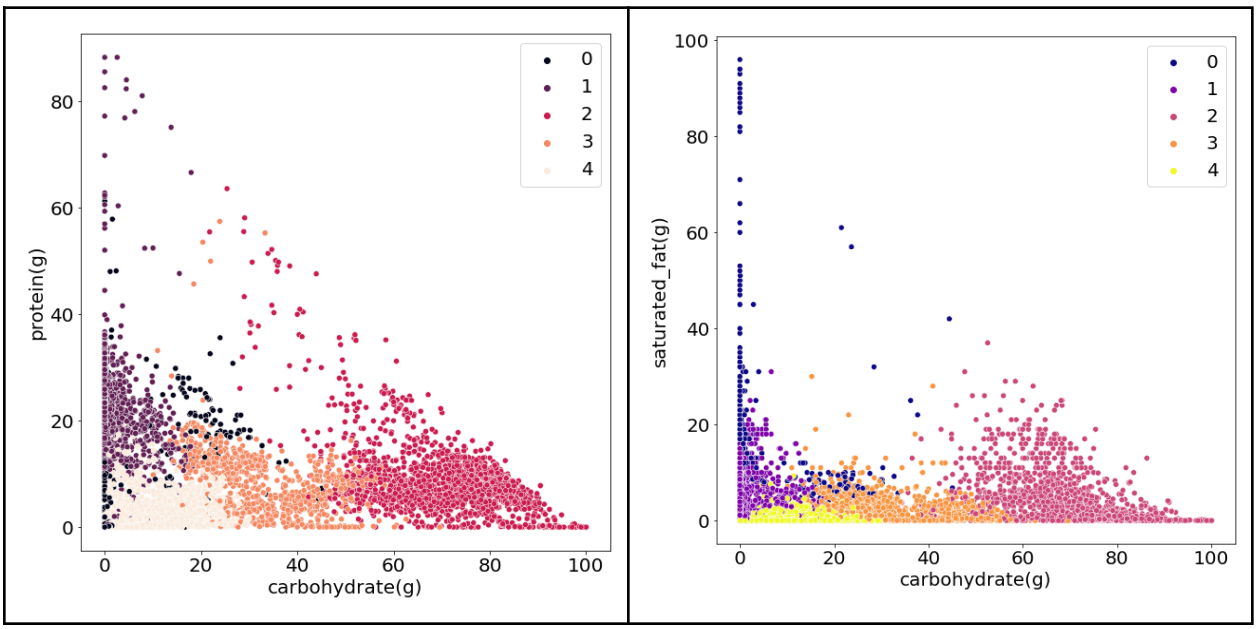
We can see that there are columns containing missing values such as NaN. Also, we can see that columns have different measurements such as gram(g), miligram(mg), and microgram(mcg) which I decide to convert all of them to gram(g). I also changed all my features except the “name” feature to numerical columns. At the end I selected some features of choices 18 out of 76 because I just wanted to see the features that are important to me.
The end result of my data that I would explore is:

	name	calories	total_fat(g)	saturated_fat(g)	cholesterol(g)	sodium(g)	vitamin_a(g)	vitamin_c(g)	vitamin_d(g)	vitamin_e(g)	magnesium(g)	protein(g)	carbohydrate(g)
0	Cornstarch	381	0.1	0.0	0.000	0.009	0.0	0.0000	0.0	0.00000	0.003	0.26	91.27
1	Nuts, pecans	691	72.0	6.2	0.000	0.000	56.0	0.0011	0.0	0.00140	0.121	9.17	13.86
2	Eggplant, raw	25	0.2	0.0	0.000	0.002	23.0	0.0022	0.0	0.00030	0.014	0.98	5.88
3	Teff, uncooked	367	2.4	0.4	0.000	0.012	9.0	0.0000	0.0	0.00008	0.184	13.30	73.13
4	Sherbet, orange	144	2.0	1.2	0.001	0.046	46.0	0.0023	0.0	0.00001	0.008	1.10	30.40
...	...	...	...	...	...	...	...	...	...	...	...	...	...
8784	Beef, raw, all grades, trimmed to 0" fat, sepa...	125	3.5	1.4	0.062	0.054	11.0	0.0000	1.0	0.00023	0.012	23.45	0.00
8785	Lamb, cooked, separable lean only, composite o...	206	8.9	3.9	0.109	0.050	0.0	0.0000	0.0	0.00019	0.022	29.59	0.00

I started with
Number of clusters = 3. It gives me an array of ([2137, 6265, 387]). I inspected the results, looked at one of the clusters to see if they made sense. For example the third cluster, I saw all the food that was chosen to this cluster has high calorie and high fat like Oil and Nuts.
Because I have high dimensional data I use TSNE to convert the dimensionality of data to 2D dimensional. After that, I picked two features to make a scatter plot of those two and colored points based on their cluster to see if they are effective on clustering or not.
The one interesting one that I found were calories and sugar and fat and calories.



Here, we can see that there are foods that have high calorie, but zero sugar. Those foods are the oil and nuts which make sense. And in the same cluster we can see that they have high fat and high calories.
K= 5:



Machine Learning Experiments

Feature Engineering:

I created a new feature called healthy fat which is the sum of polyunsaturated fats and monounsaturated fats. These fats are known to be good for health. I experimented with many different combinations of selecting features from the original 76 features. At last, I am reporting my results for my final attempt where I chose 6 features that I care about: Calories, Cholesterol, Protein, Carbohydrates, Fiber, and healthy fat.

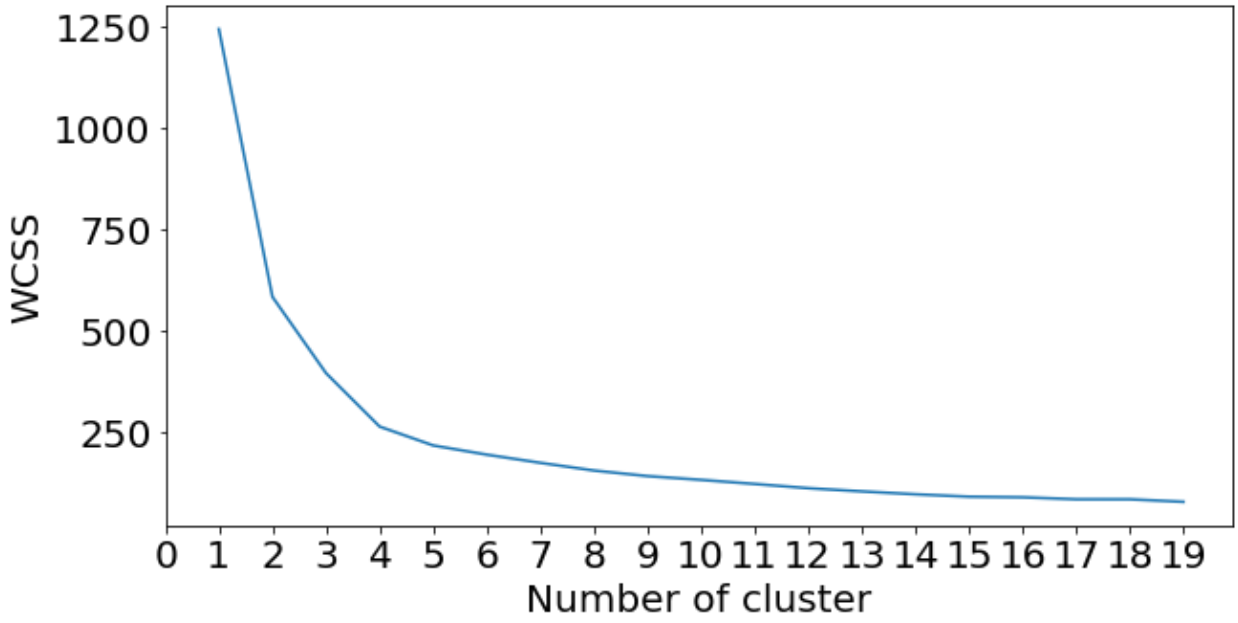
Feature scaling:

I found that feature scaling is important before feeding the data to k means algorithm. The results make much more sense when I do feature scaling at first. This makes sense because kmeans is getting the similarities based on euclidean distance. For example , chlorohistol and calories have very different scales which can affect the results disproportionately. I used minmax normalization and scale each feature to be in range of (0,1).

Kmeans

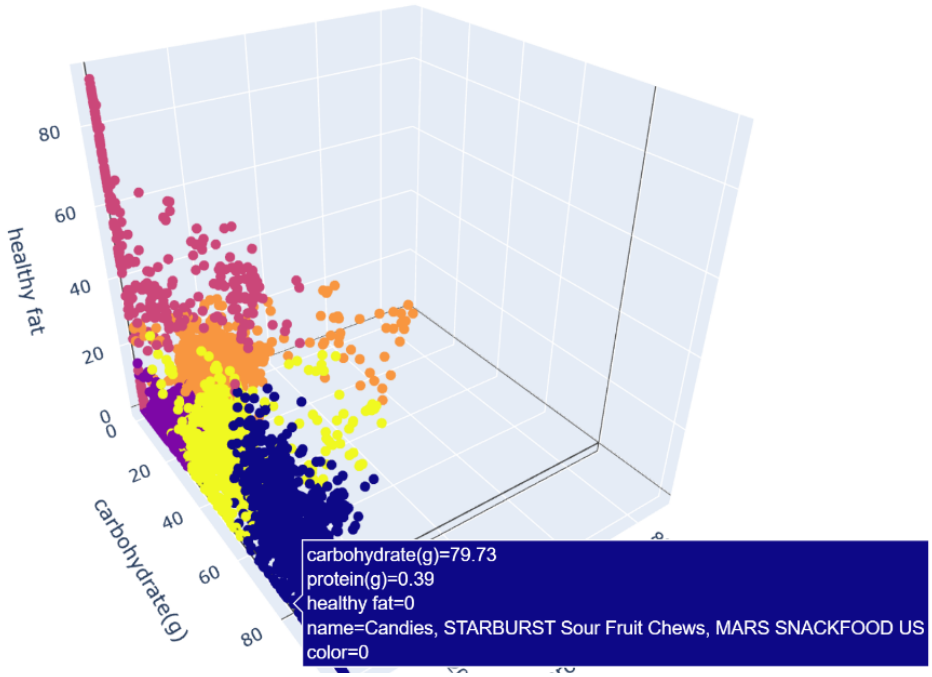
K value for Kmeans

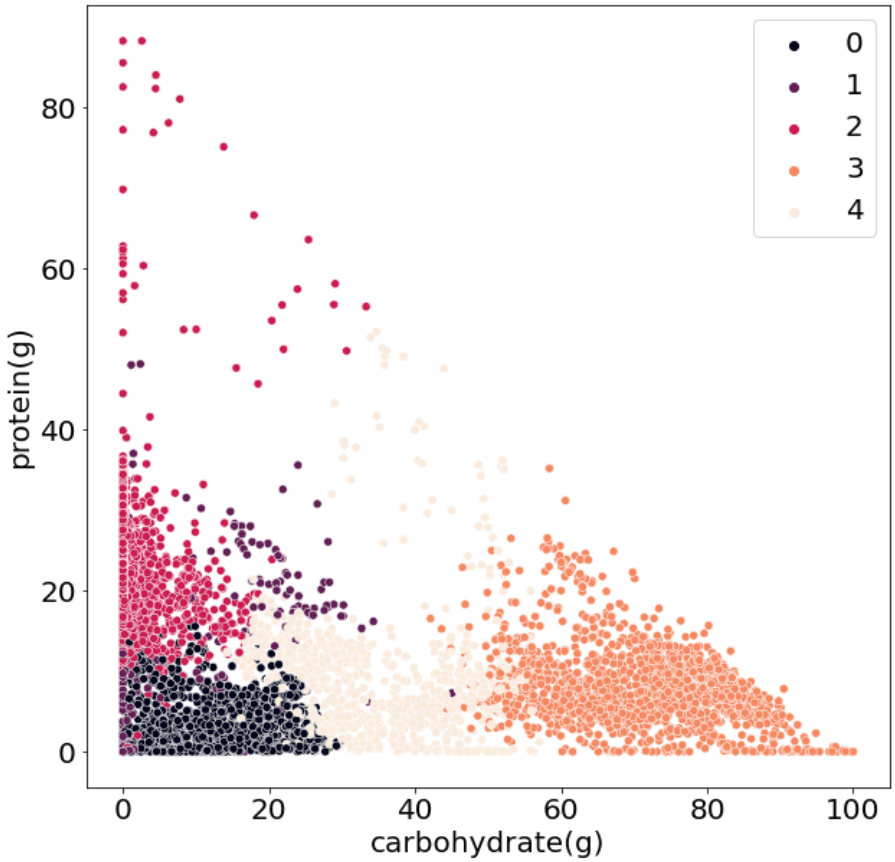
In order to find the best k-value, I used different values for k ranging from 1 to 20 and I got this plot. Based on this plot, I chose k =4 where the elbow forms. The y axis is the sum of squared distances of samples to their closest cluster center known as WCSS. Of course, more clusters means less variability within the data of a cluster, so the optimal value should be where the elbow forms.



Kmeans food recommender

First of all, I use 3d and 2d plots, to gain a bit of understanding of different clusters in my dataset. For example, you see that blue cluster down there, this is the area for me to avoid, where it is all carbohydrates and low fats and protein. I used the names to hover on the data points, so I can see which foods are in which clusters. This blue cluster is all about candies.


healthy fat
carbohydrate(g)
carbohydrate(g)=79.73
protein(g)=0.39
healthy fat=0
name=Candies, STARBURST Sour Fruit Chews, MARS SNACKFOOD US
color=0


0
1
2
3
4
protein(g)
carbohydrate(g)

Moreover, I ask my kmeans recommender to return the names of the top foods similar to my test food. For this matter, I chose k-values to be higher and disregarded the elbow plot. The reason is that I wanted very fine clustering. I recently heard about the benefits of chia seed, which is one of my favorites with a relatively high fiber, protein and healthy fat. I made a new data point for chia seed. I searched on the internet to find the nutrition facts of a chia seed, and I entered the values. Then I used my kmeans algorithm to see which cluster it is related to. First I chose 20 cluster and it predict that chia seed would relate to cluster #11. I printed all food that is in this cluster, to see what other food is like chia seed with similar beneficiary. I found 114 foods. Some of them are:

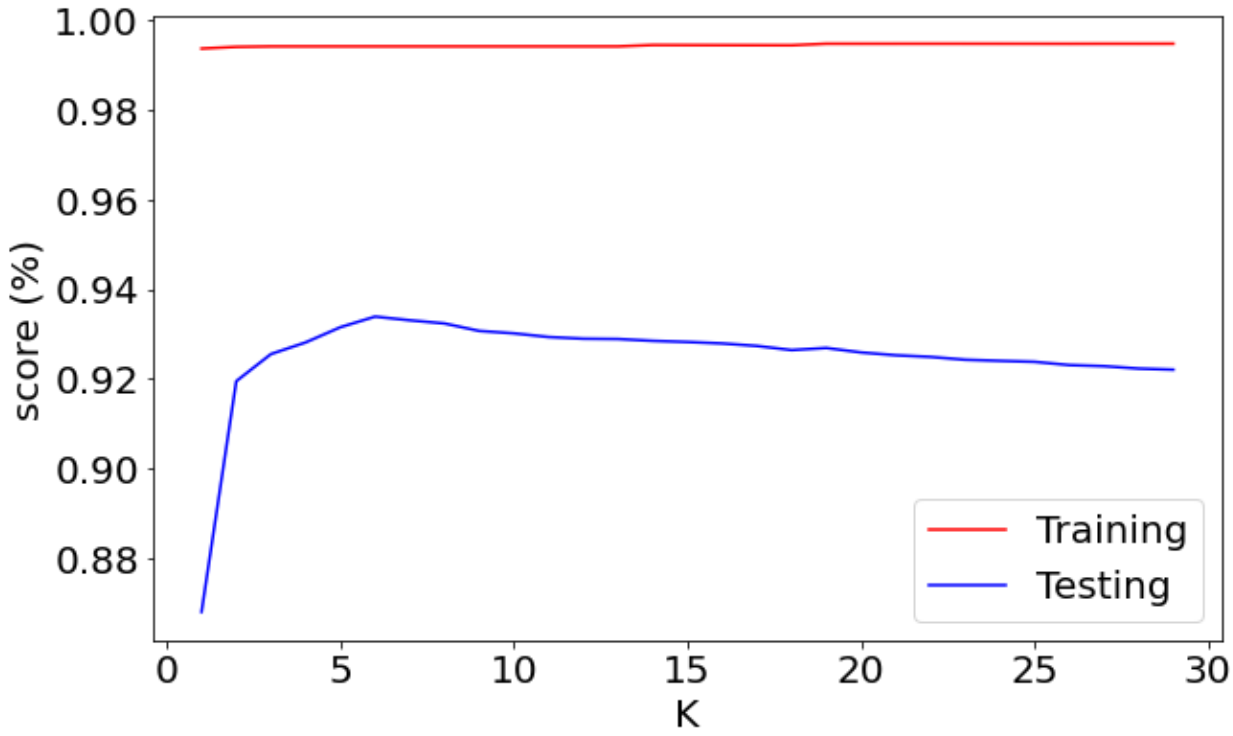
‘Amaranth grain’, 'Apricots', 'Arrowroot flour', 'Beans','Bulgur, 'Celery flakes', 'Cereals', 'Chickpea flour (besan)', 'Couscous','Beverages', 'Bread',,'Lentils', 'Mothbeans', 'Peas', 'Pectin', 'Pepeao', 'Peppers', 'Quinoa','Rye grain', 'Seaweed', 'Seeds', 'Wild rice', 'Winged beans', 'Wocas', 'Yardlong beans', 'Yellow rice’.

K-nearest neighbor regression

Using the same data frame this time I called it df_data3. In df_data3 I dropped the column name using .drop(columns=['name']). Because KNN is supervised, I need a target vector. In this exemple I picked “calorie” as y, and the rest of them as x. Then I splitted the data to train and test. For this problem, I decided to choose 0.2 for the proportion of the dataset to include in the train split. After that I got the training set with shape 7031x6 and test shape by 1758x6, where 6 is the number of features. We use the same minmax normalization for our features as we did for kmeans.

Experiments:

I used different values of k and measured training and testing r-squared as an accuracy metric. Below is the plot of the result for different number of Ks ranging from 1 to 30.



Again we can see that the elbow forms at k=5. The training and testing accuracy for k=5 are around 0.99 and 0.93. This implied a bit overfitting which might be reasonable for a small value of k.

KNN food recommender system

Since KNN is supervised, I choose a target vector that I care about, for example calorie. I ask my KNN algorithm to first of all return the neighbors and their respective distance, and then predict the target value, which can be any of the vectors in the dataset.

For example , this time I chose Flax seed, again one of the very beneficial seeds for health. The recommender system returned the following as 5 nearest neighbors of the flax seed.

```
['Spices, poppy seed']
['Seeds, with salt added (decorticated), toasted, sesame seed kernels']
```

```
['Seeds, without salt added (decorticated), toasted, sesame seed kernels']
['Seeds, roasted and toasted, whole, sesame seeds']
['Nuts, unblanched, honey roasted, almonds']
```

Also, it predicted its calories to be 558 in 100 grams of seed which is not far from the 534 of ground truth.

Comparison of my algorithms vs sklearn

I used sklearn kmeans and nearest neighbor regression and found out the following.

All my k means results match with sklearn's kmeans. There were some bugs in my initial k means algorithm that I fixed during this process. The WCSS distances returned by my kmeans is exactly the same as sklearn. The speed of processing for sklearn is faster than my algorithm.

For KNN, I wrote a knn regressor that takes the train test, and one data point from the test set, and it returns the mean value of the target values for the nearest neighbors of that point. It also returns the distances and neighbors. This is different from sklearns implementation, and I used sklearn's knn to get the accuracy metric on the whole test set which was very time consuming with my code.