NUTRITIONAL CLUSTERING OF THE FOOD ITEMS

What We Did: We worked with a dataset that includes nutritional information for various foods. This data helps us understand the healthiness of different food items based on things like calories, carbs, and vitamins.

Cleaning the Data: We removed unnecessary columns that didn't help in clustering (like food names) and fixed any missing data by filling in averages for each nutrient.

Normalizing the Data: We adjusted the data so that all the features (like calories and fat) are on the same scale. This is important because some numbers can be much larger than others, which could skew the results.

Finding Clusters: We used a method called the Elbow Method to figure out how many groups (or clusters) of foods there should be. We found that 5 clusters was a good number based on how the data was organized.

Clustering Foods: We applied K-Means clustering to group the food items into these clusters. Each food item was assigned to a cluster based on its nutritional features.

Visualizing the Results: We created plots to show how the food items grouped together. One plot compared two features (calcium and calories), and another looked at multiple features at once.

What We Learned: The clusters give us a clearer picture of how foods relate to one another based on nutrition. For instance, some clusters might contain high-calorie foods, while others might include healthier options.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
data = pd.read_csv('/content/foodstruct_nutritional_facts.csv')
data.head()
```

	Food Name	Category Name	Calcium	Calories	Carbs	Cholesterol	Copper	Fats	Fiber	Folate	...	Vitamin D	Vitamin E	Vitamin K	Omega-3 - ALA	Eico
0	Acerola	Fruits	0.012	32.0	7.7	0.0	0.00009	0.30	1.1	0.000014	...	NaN	NaN	NaN	NaN	
1	Apple	Fruits	0.006	52.0	14.0	0.0	0.00003	0.17	2.4	0.000003	...	0.0	0.00018	0.000002	NaN	
2	Apricot	Fruits	0.013	48.0	11.0	0.0	0.00008	0.39	2.0	0.000009	...	0.0	0.00089	0.000003	NaN	
3	Dried fruit	Fruits	0.055	241.0	63.0	0.0	0.00034	0.51	7.3	0.000010	...	0.0	0.00430	0.000003	NaN	
4	Avocado	Fruits	0.012	160.0	8.5	0.0	0.00019	15.00	6.7	0.000081	...	0.0	0.00210	0.000021	0.11	

5 rows × 59 columns

```
data.info()
```

```
3   Calories                          1174 non-null   float64
4   Carbs                             1174 non-null   float64
5   Cholesterol                       1119 non-null   float64
6   Copper                            1094 non-null   float64
7   Fats                              1174 non-null   float64
8   Fiber                             1076 non-null   float64
9   Folate                            1071 non-null   float64
10  Iron                              1153 non-null   float64
11  Magnesium                         1113 non-null   float64
```

```
 20  Protein                             1174 non-null   float64
 21  Saturated Fat                       1093 non-null   float64
 22  Selenium                            1020 non-null   float64
 23  Sodium                              1153 non-null   float64
 24  Trans Fat                           634 non-null    float64
 25  Vitamin A (IU)                      1118 non-null   float64
 26  Vitamin A RAE                       1056 non-null   float64
 27  Vitamin B1                          1115 non-null   float64
 28  Vitamin B12                         1083 non-null   float64
 29  Vitamin B2                          1116 non-null   float64
 30  Vitamin B3                          1115 non-null   float64
 31  Vitamin B5                          975 non-null    float64
 32  Vitamin B6                          1091 non-null   float64
 33  Vitamin C                           1124 non-null   float64
 34  Zinc                                1108 non-null   float64
 35  Choline                             732 non-null    float64
 36  Fructose                            302 non-null    float64
 37  Histidine                           709 non-null    float64
 38  Isoleucine                          713 non-null    float64
 39  Leucine                             713 non-null    float64
 40  Lysine                              721 non-null    float64
 41  Manganese                           1012 non-null   float64
 42  Methionine                          718 non-null    float64
 43  Phenylalanine                       710 non-null    float64
 44  Starch                              199 non-null    float64
 45  Sugar                               874 non-null    float64
 46  Threonine                           712 non-null    float64
 47  Tryptophan                          710 non-null    float64
 48  Valine                              713 non-null    float64
 49  Vitamin D                           829 non-null    float64
 50  Vitamin E                           816 non-null    float64
 51  Vitamin K                           791 non-null    float64
 52  Omega-3 - ALA                       176 non-null    float64
 53  Omega-6 - Eicosadienoic acid        265 non-null    float64
 54  Omega-6 - Gamma-linoleic acid       170 non-null    float64
 55  Omega-3 - Eicosatrienoic acid       114 non-null    float64
 56  Omega-6 - Dihomo-gamma-linoleic acid  119 non-null    float64
 57  Omega-6 - Linoleic acid             141 non-null    float64
 58  Omega-6 - Arachidonic acid          1 non-null      float64
dtypes: float64(57), object(2)
memory usage: 541.3+ KB
```

```
data.describe()
```

	Calcium	Calories	Carbs	Cholesterol	Copper	Fats	Fiber	Folate	Iron	Magnesium	...
count	1149.000000	1174.000000	1174.000000	1119.000000	1094.000000	1174.000000	1076.000000	1071.000000	1153.000000	1113.000000	...
mean	0.099660	224.412266	25.049940	0.030845	0.000217	10.541371	3.123885	0.000053	0.002870	0.041921	...
std	0.264301	185.838852	27.222293	0.085214	0.000565	18.179044	6.383304	0.000135	0.007132	0.068221	...
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
25%	0.012000	72.000000	3.600000	0.000000	0.000050	0.380000	0.000000	0.000005	0.000380	0.011000	...
50%	0.031000	179.000000	13.000000	0.000000	0.000100	3.000000	1.300000	0.000017	0.001000	0.021000	...
75%	0.089000	341.750000	44.750000	0.033500	0.000200	14.000000	3.200000	0.000053	0.002500	0.041000	...
max	5.876000	902.000000	100.000000	1.085000	0.014000	100.000000	70.000000	0.002340	0.124000	0.770000	...

8 rows × 57 columns

```
data.isnull().sum()
```

	0
Food Name	0
Category Name	0
Calcium	25
Calories	0
Carbs	0
Cholesterol	55
Copper	80
Fats	0
Fiber	98
Folate	103
Iron	21
Magnesium	61
Monounsaturated Fat	110
Net carbs	1
Omega-3 - DHA	272
Omega-3 - DPA	279
Omega-3 - EPA	271
Phosphorus	48
Polyunsaturated fat	110
Potassium	45
Protein	0
Saturated Fat	81
Selenium	154
Sodium	21
Trans Fat	540
Vitamin A (IU)	56
Vitamin A RAE	118
Vitamin B1	59
Vitamin B12	91
Vitamin B2	58
Vitamin B3	59
Vitamin B5	199
Vitamin B6	83
Vitamin C	50
Zinc	66
Choline	442
Fructose	872
Histidine	465
Isoleucine	461
Leucine	461
Lysine	453
Manganese	162
Methionine	456
Phenylalanine	464
Starch	975
Sugar	300

Threonine	462
Tryptophan	464
Valine	461
Vitamin D	345
Vitamin E	358
Vitamin K	383
Omega-3 - ALA	998
Omega-6 - Eicosadienoic acid	909
Omega-6 - Gamma-linoleic acid	1004
Omega-3 - Eicosatrienoic acid	1060
Omega-6 - Dihomo-gamma-linoleic acid	1055
Omega-6 - Linoleic acid	1033
Omega-6 - Arachidonic acid	1173

dtype: int64

```
data.shape
```

```
(1174, 59)
```

```
data.duplicated().sum()
```

```
0
```

```
x = data.drop(columns=['Food Name','Category Name'])
x
```

	Calcium	Calories	Carbs	Cholesterol	Copper	Fats	Fiber	Folate	Iron	Magnesium	...	Vitamin D	Vitamin E	Vitamin K	Om
0	0.012	32.0	7.7	0.000	0.00009	0.30	1.1	0.000014	0.00020	0.018	...	NaN	NaN	NaN	
1	0.006	52.0	14.0	0.000	0.00003	0.17	2.4	0.000003	0.00012	0.005	...	0.000000e+00	0.00018	2.200000e-06	
2	0.013	48.0	11.0	0.000	0.00008	0.39	2.0	0.000009	0.00039	0.010	...	0.000000e+00	0.00089	3.300000e-06	
3	0.055	241.0	63.0	0.000	0.00034	0.51	7.3	0.000010	0.00270	0.032	...	0.000000e+00	0.00430	3.100000e-06	
4	0.012	160.0	8.5	0.000	0.00019	15.00	6.7	0.000081	0.00055	0.029	...	0.000000e+00	0.00210	2.100000e-05	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1169	0.012	293.0	27.0	0.059	0.00005	14.00	0.7	0.000008	0.00110	0.022	...	1.000000e-07	0.00130	2.700000e-05	
1170	0.055	331.0	82.0	0.000	0.00190	0.10	3.9	0.000000	0.00180	0.002	...	0.000000e+00	0.00051	3.800000e-06	
1171	0.733	379.0	73.0	0.000	0.00033	4.40	7.5	0.000043	0.04800	0.100	...	0.000000e+00	0.00370	1.500000e-06	
1172	0.020	426.0	74.0	0.008	0.00015	9.70	2.5	0.000087	0.00060	0.014	...	0.000000e+00	0.00240	9.000000e-07	
1173	0.007	47.0	11.0	0.000	0.00002	0.10	0.3	0.000000	0.00035	0.003	...	0.000000e+00	0.00002	2.000000e-07	

1174 rows × 57 columns

Suggested code may be subject to a license |

```
x.fillna(x.mean(), inplace=True)
x
```

	Calcium	Calories	Carbs	Cholesterol	Copper	Fats	Fiber	Folate	Iron	Magnesium	...	Vitamin D	Vitamin E	Vitamin K	C
0	0.012	32.0	7.7	0.000	0.00009	0.30	1.1	0.000014	0.00020	0.018	...	7.752714e-07	0.001692	2.857535e-05	0.
1	0.006	52.0	14.0	0.000	0.00003	0.17	2.4	0.000003	0.00012	0.005	...	0.000000e+00	0.000180	2.200000e-06	0.
2	0.013	48.0	11.0	0.000	0.00008	0.39	2.0	0.000009	0.00039	0.010	...	0.000000e+00	0.000890	3.300000e-06	0.
3	0.055	241.0	63.0	0.000	0.00034	0.51	7.3	0.000010	0.00270	0.032	...	0.000000e+00	0.004300	3.100000e-06	0.
4	0.012	160.0	8.5	0.000	0.00019	15.00	6.7	0.000081	0.00055	0.029	...	0.000000e+00	0.002100	2.100000e-05	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1169	0.012	293.0	27.0	0.059	0.00005	14.00	0.7	0.000008	0.00110	0.022	...	1.000000e-07	0.001300	2.700000e-05	0.
1170	0.055	331.0	82.0	0.000	0.00190	0.10	3.9	0.000000	0.00180	0.002	...	0.000000e+00	0.000510	3.800000e-06	0.
1171	0.733	379.0	73.0	0.000	0.00033	4.40	7.5	0.000043	0.04800	0.100	...	0.000000e+00	0.003700	1.500000e-06	0.
1172	0.020	426.0	74.0	0.008	0.00015	9.70	2.5	0.000087	0.00060	0.014	...	0.000000e+00	0.002400	9.000000e-07	0.
1173	0.007	47.0	11.0	0.000	0.00002	0.10	0.3	0.000000	0.00035	0.003	...	0.000000e+00	0.000020	2.000000e-07	0.

1174 rows × 57 columns

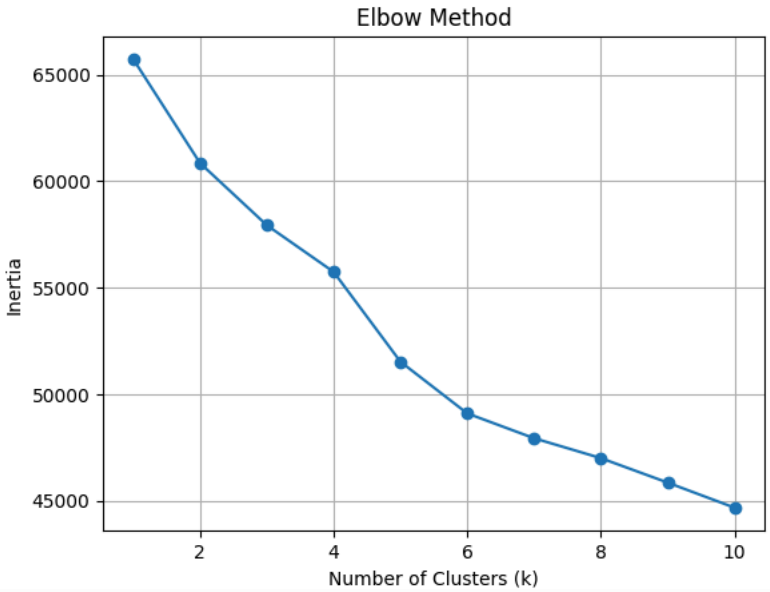
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
```

```
from sklearn.cluster import KMeans

inertia = []
k_values = range(1, 11)  # Test for 1 to 10 clusters
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(x_scaled)
    inertia.append(kmeans.inertia_)
```

```
plt.plot(k_values, inertia, marker='o')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method')
plt.grid(True)
plt.show()
```



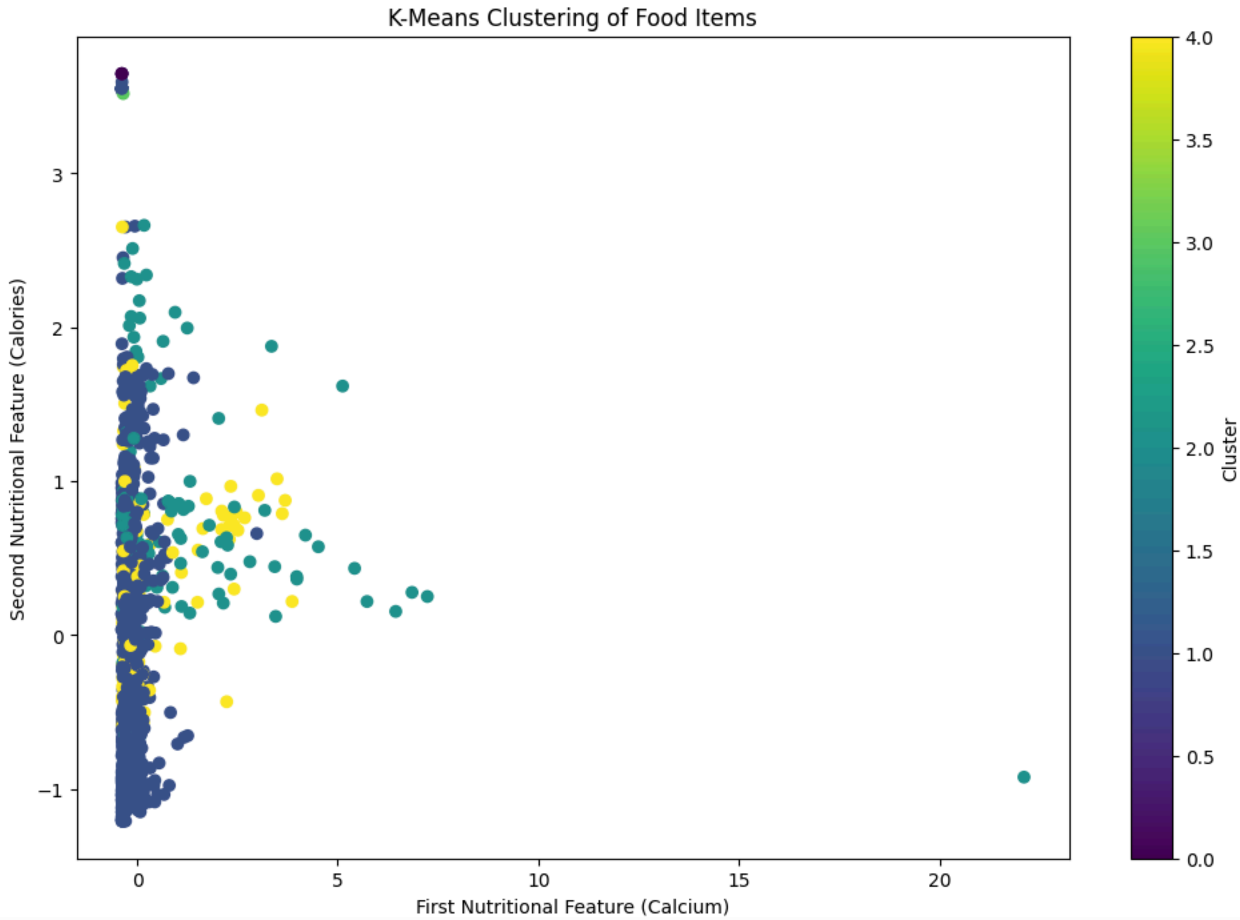

```
optimal_k = 5  # Change based on Elbow Method results
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
data['Cluster'] = kmeans.fit_predict(x_scaled)


print(data[['Food Name', 'Cluster']].head())
```

```
     Food Name  Cluster
0      Acerola        1
1        Apple        1
2      Apricot        1
3  Dried fruit        1
4      Avocado        1
```

Start coding or generate with AI.

```
plt.figure(figsize=(12, 8))
plt.scatter(x_scaled[:, 0], x_scaled[:, 1], c=data['Cluster'], cmap='viridis', marker='o')
plt.title('K-Means Clustering of Food Items')
plt.xlabel('First Nutritional Feature (Calcium)')
plt.ylabel('Second Nutritional Feature (Calories)')
plt.colorbar(label='Cluster')
plt.show()
```



Start coding or generate with AI.

The analysis helped us group food items based on their nutritional profiles, making it easier to understand which foods are similar.

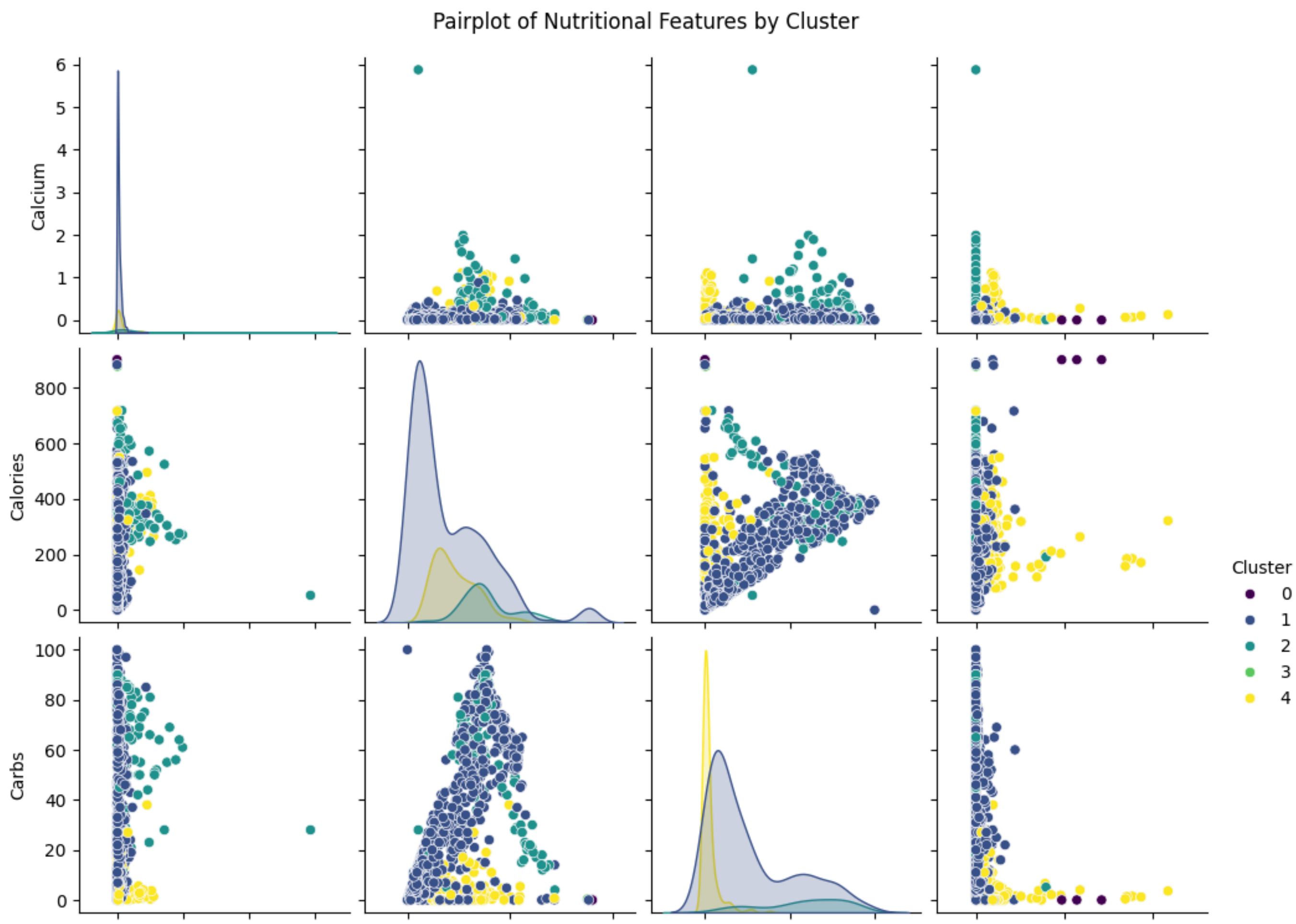
Understanding the Groups: The clusters can help people choose healthier foods by showing which items share similar nutritional characteristics. For example, you might see a group of low-calorie, high-fiber foods, which are generally healthier.

Who Can Benefit: Nutritionists and health-conscious consumers can use these insights to make better food choices. Food companies can also use this information to create products that appeal to specific health trends.

Next Steps: In the future, we could explore other clustering methods, include more features, or even use this data to predict how healthy a food item is based on its cluster. We might also dive deeper into what each cluster means for overall health.

In simple terms, this project showed us how to organize and make sense of food nutrition data, helping people understand their food choices better.

```
sns.pairplot(data, hue='Cluster', vars=x.columns[:4], palette='viridis')
plt.suptitle('Pairplot of Nutritional Features by Cluster', y=1.02)
plt.show()
```




```
sns.pairplot(data, hue='Cluster', vars=x.columns[:4], palette='viridis')
plt.suptitle('Pairplot of Nutritional Features by Cluster', y=1.02)
plt.show()
```