Building a Clustering Model for Customer Segmentation

Import software libraries and load the dataset

In [1]:
```
import sys                                  # Read system parameters.
import os                                   # Interact with the operating system.
import numpy as np                          # Work with multi-dimensional arrays and matrices.
import pandas as pd                         # Manipulate and analyze data.
import matplotlib as mpl                    # Create 2D charts.
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import yellowbrick                          # Visualize elbow and silhouette plots.
import sklearn                              # Perform data mining and analysis.
import seaborn as sns

# Summarize software libraries used in this project.
print('Libraries used in this project:')
print('- Python {}'.format(sys.version))
print('- NumPy {}'.format(np.__version__))
print('- pandas {}'.format(pd.__version__))
print('- Matplotlib {}'.format(mpl.__version__))
print('- Yellowbrick {}'.format(yellowbrick.__version__))
print('- scikit-learn {}\n'.format(sklearn.__version__))

# Load the dataset.
#PROJECT_ROOT_DIR = "."
#DATA_PATH = os.path.join(PROJECT_ROOT_DIR, "wholesale_customers_data")
#print('Data files in this project:', os.listdir(DATA_PATH))
#data_raw_file = os.path.join(DATA_PATH, 'wholesale_customers_data.csv')
df = pd.read_csv('wholesale_customers_data.csv')
print('Loaded {} records'.format(len(df)))

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

```
Libraries used in this project:
- Python 3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
- NumPy 1.19.2
- pandas 1.1.3
- Matplotlib 3.3.2
- Yellowbrick 1.2
- scikit-learn 0.23.2

Loaded 440 records
```

Get acquainted with the dataset

In [2]:
```
# View data types and see if there are missing entries.
df.info()

# View first 10 records.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440 entries, 0 to 439
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Fresh             440 non-null    int64
 1   Milk              440 non-null    int64
 2   Grocery           440 non-null    int64
 3   Frozen            440 non-null    int64
 4   Detergents_Paper  440 non-null    int64
 5   Deli              440 non-null    int64
dtypes: int64(6)
memory usage: 20.8 KB
```

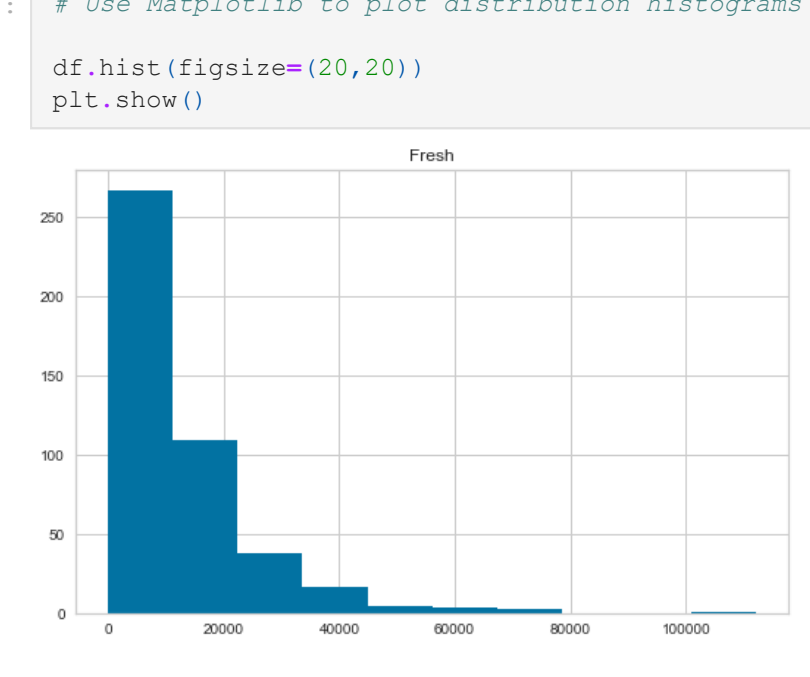
In [3]:
```
df.head(10)
```

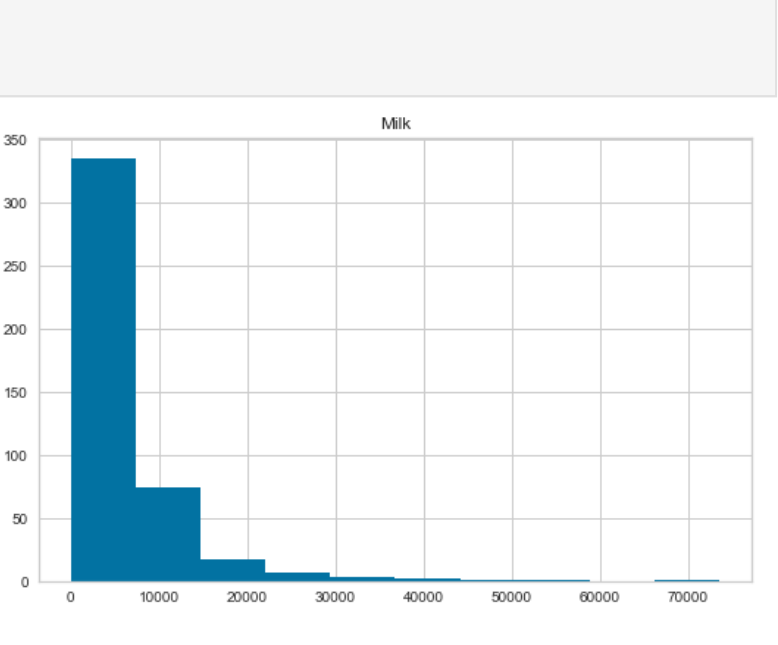
Out[3]:

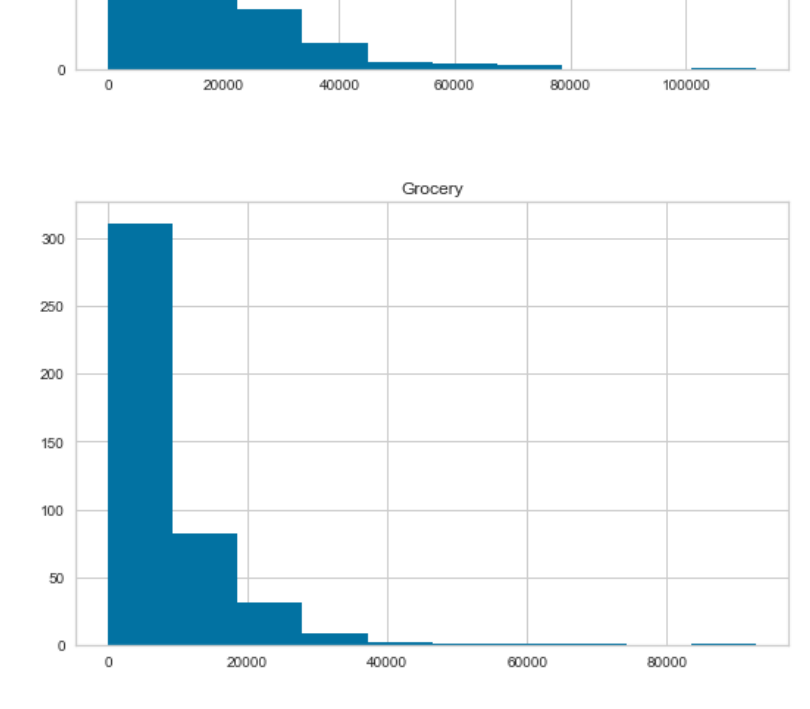
	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Deli
0	12669	9656	7561	214	2674	1338
1	7057	9810	9568	1762	3293	1776
2	6353	8808	7684	2405	3516	7844
3	13265	1196	4221	6404	507	1788
4	22615	5410	7198	3915	1777	5185
5	9413	8259	5126	666	1795	1451
6	12126	3199	6975	480	3140	545
7	7579	4956	9426	1669	3321	2566
8	5963	3648	6192	425	1716	750
9	6006	11093	18881	1159	7425	2098

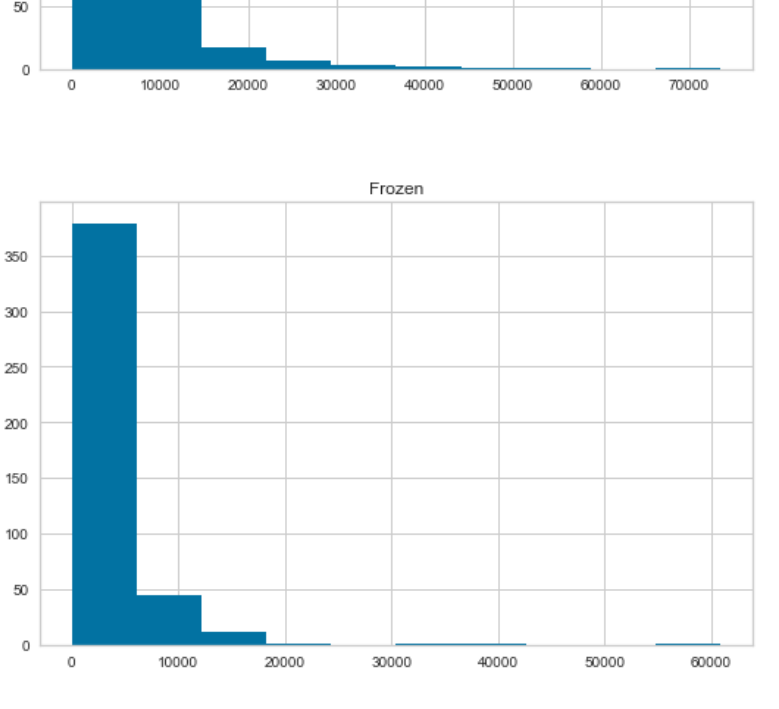
Examine the distribution of various features

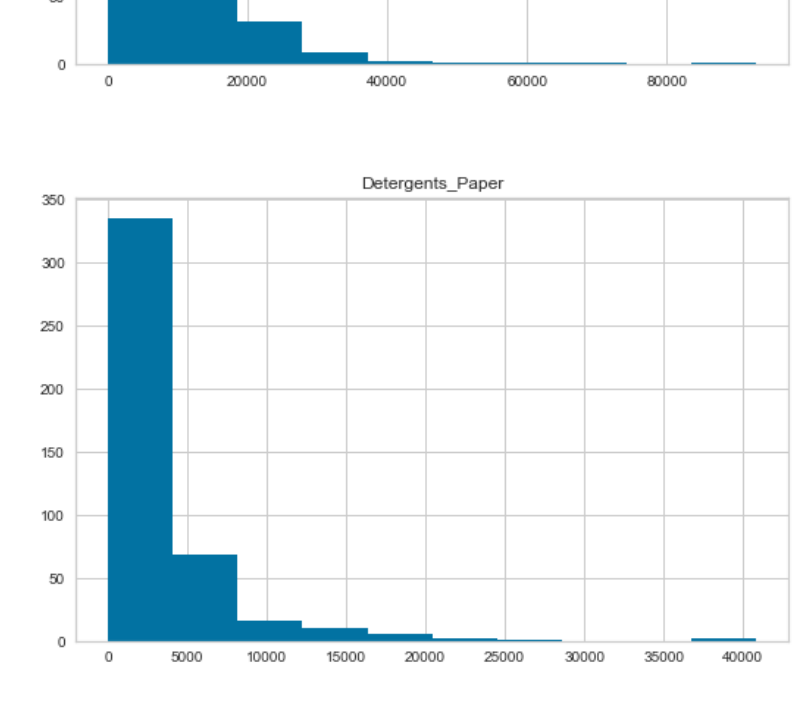
In [4]:
```
# Use Matplotlib to plot distribution histograms for all features.

df.hist(figsize=(20,20))
plt.show()
```

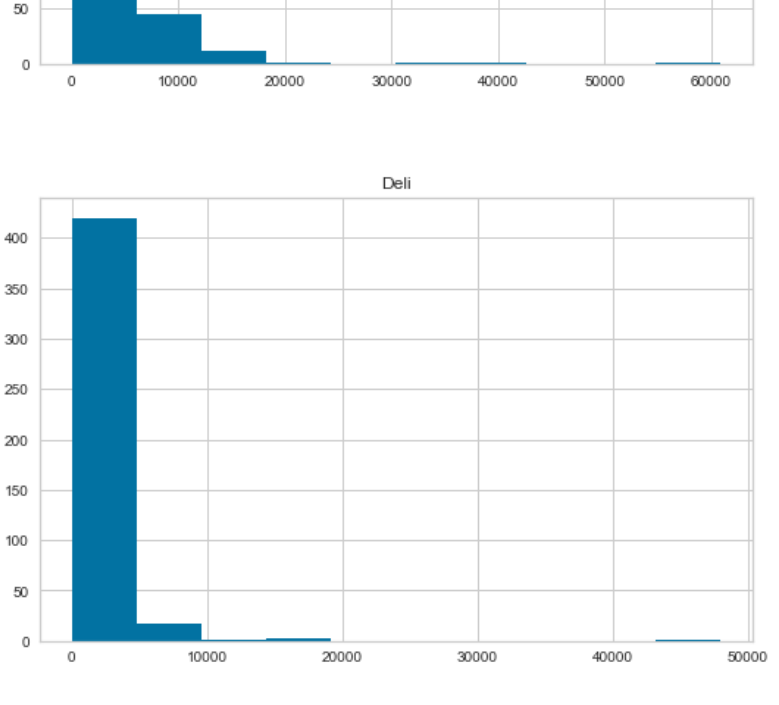












Examine a general summary of statistics

In [5]:
```
# View summary statistics (mean, standard deviation, min, max, etc.) for each feature.
df.describe()
```

Out[5]:

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Deli
count	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000
mean	12000.297727	5796.265909	7951.277273	3071.931818	2881.493182	1524.870455
std	12647.328865	7380.377175	9503.162829	4854.673333	4767.854448	2820.105937
min	3.000000	55.000000	3.000000	25.000000	3.000000	3.000000
25%	3127.750000	1533.000000	2153.000000	742.250000	256.750000	408.250000
50%	8504.000000	3627.000000	4755.500000	1526.000000	816.500000	965.500000
75%	16933.750000	7190.250000	10655.750000	3554.250000	3922.000000	1820.250000
max	112151.000000	73498.000000	92780.000000	60869.000000	40827.000000	47943.000000

Use a *k*-means model to label every row in the dataset

In [6]:
```
# Construct a k-means clustering model class object.
# Use 3 as the initial number of clusters.

# Use fresh products and milk products only for the initial training data.

# Fit the training data to the clustering object.

# Predict the cluster labels based on the training data.
```

In [7]:
```
df.head()
```

Out[7]:

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Deli
0	12669	9656	7561	214	2674	1338
1	7057	9810	9568	1762	3293	1776
2	6353	8808	7684	2405	3516	7844
3	13265	1196	4221	6404	507	1788
4	22615	5410	7198	3915	1777	5185

In [8]:
```
df2 = df[["Fresh","Milk"]]
```

In [9]:
```
df2.head()
```

Out[9]:

	Fresh	Milk
0	12669	9656
1	7057	9810
2	6353	8808
3	13265	1196
4	22615	5410

In [10]:
```
scaler = StandardScaler()
```

In [11]:
```
df2_scaled = scaler.fit_transform(df2)
```

In [12]:
```
df2_scaled
```

Out[12]:
```
array([[ 5.29331898e-02,  5.23567773e-01],
       [-3.91303970e-01,  5.44457667e-01],
       [-4.47029261e-01,  4.08537706e-01],
       [ 1.00111067e-01, -6.24019925e-01],
       [ 8.40239480e-01, -5.23964566e-01],
       [-2.04805557e-01,  3.34068589e-01],
       [ 9.95035090e-03, -3.52315651e-01],
       [-3.49981451e-01, -1.13980948e-01],
       [-4.77900913e-01, -2.91409401e-01],
       [-4.74497115e-01,  7.18043904e-01],
       [-6.83474454e-01,  5.23459952e-01],
       [-9.14279066e-02, -3.53786629e-01],
       [ 1.56049887e+00,  8.84800166e-01],
       [ 7.29576981e-01,  5.90511794e-01],
       [ 1.00156365e+00,  4.97658776e-01],
       [-1.19382738e-01, -3.45143316e-01],
       [-6.89179316e-01,  4.09622893e-01],
       [-4.83876608e-01,  6.89330097e-01],
       [ 5.22498836e-01,  7.19933704e-01],
       [-3.34970077e-01, -4.07812310e-01],
       [ 4.38987159e-01, -1.73259414e-01],
       [illegible]
       [-8.83902719e-01,  7.84148857e-01],
       [ 3.90700729e-01,  6.15484233e-02],
       [ 1.28529159e-01, -7.46239371e-01],
       [-5.13838682e-01, -5.88073029e-01],
       [-9.35672105e-01, -3.13520133e-01],
       [-3.87344066e-01, -6.48979279e-01],
       [ 3.97587482e-01, -8.91572425e-02],
       [ 1.47421656e+00,  1.91635492e-01],
       [-7.57803893e-01,  1.76442842e-01],
       [-2.48792055e-02, -4.94068505e-01],
       [-1.84461900e-01, -4.76976773e-01],
       [-7.07142720e-01,  1.00973259e+00],
       [-5.94975717e-01,  1.16694939e+00],
       [ 1.04510060e+00,  7.71940477e-01],
       [-9.03771398e-01, -6.93336392e-01],
       [-7.95087350e-01, -6.65392768e-01],
       [-8.62292562e-01,  8.02325778e-01],
       [-4.45446099e-01, -6.80449769e-01],
       [-7.48779872e-01, -1.43688005e-01],
       [-8.28254587e-01, -2.77844534e-01],
       [-5.10593201e-01,  9.40416118e-01],
       [ 5.19807562e-01, -5.29201509e-01],
       [ 9.39624432e-03,  3.05612833e+00],
       [-3.72778979e-01, -6.48979279e-01],
       [-6.87353200e-01,  1.09296753e-01],
       [-7.62949169e-01,  1.02378671e-01],
       [-7.27802980e-01,  6.74002142e-01],
       [-7.49492295e-01,  1.46537645e+00],
       [ 4.78407884e-01, -5.86173948e-01],
       [-9.48495714e-01,  2.31651848e-01],
       [-6.21018726e-01, -7.36472668e-01],
       [ 2.18056950e-01, -6.64307578e-01],
       [-5.22783546e-01,  2.31516198e-01],
       [-5.50884665e-01, -6.35007467e-01],
       [-7.29069509e-01, -4.43471553e-01],
       [-3.74124666e-01, -6.96863258e-01],
       [ 5.38039287e-02, -3.46347110e-01],
       [ 6.95142715e-01,  1.69200128e-02],
       [-6.29963590e-01, -4.87286072e-01],
       [-8.01974103e-01, -7.07986448e-01],
       [-2.64728196e-01, -4.13764496e-01],
       [-7.34822683e-02,  2.61441220e-02],
       [-7.64374014e-01,  6.62200708e-01],
       [ 1.10518158e+00, -5.44801105e-01],
       [-8.73612168e-01, -1.10182785e-01],
       [ 2.85183004e-01, -4.19597388e-01],
       [-8.04428004e-01,  7.91827496e-02],
       [-2.66390516e-01, -6.75023823e-01],
       [ 5.29702322e-01, -2.65636155e-01],
       [-3.67079597e-01, -7.21822612e-01],
       [ 2.80953426e+00, -4.38045607e-01],
       [ 7.99156435e-01, -3.01311753e-01],
       [ 1.02341128e+00, -5.41952483e-01],
       [ 4.92814655e-01, -6.96863258e-01],
       [-6.25584529e-02, -3.01176105e-01],
       [-4.58982131e-01,  2.68412636e-01],
       [-7.07538511e-01,  4.85261515e-02],
       [-2.46600995e-01, -4.56900771e-01],
       [ 1.24175464e-01, -7.39321290e-01],
       [ 2.90565754e-01, -7.01664296e-02],
       [-3.13489575e-01, -2.71469047e-01],
       [-6.97327118e-01, -5.15907940e-01],
       [-4.64364881e-01,  2.35170482e+00],
       [-4.25656578e-01, -5.33949212e-01],
       [ 1.38730099e+00,  2.93643288e-01],
       [-1.28338820e-01,  1.65162206e+00],
       [ 1.55258307e+00, -7.19923531e-01],
       [-7.18991066e-02, -1.06927218e-01],
       [-2.79055809e-01, -5.24318157e-01],
       [ 3.48950219e+00, -1.67290873e-01],
       [ 3.26168523e+00, -1.13574002e-01],
       [-2.22220306e-01, -1.23612003e-01],
       [-3.27896346e-01, -6.35685710e-01],
       [ 8.91058969e-01, -6.00145760e-01],
       [-7.79493208e-01, -6.35007467e-01],
       [-8.64983937e-01,  5.26687692e-01],
       [-4.82175449e-01,  2.40515039e+00],
       [-9.04642137e-01,  5.38082180e-01],
       [ 7.03929262e-01, -6.20493060e-01],
       [-7.30611113e-03,  3.06122965e-01],
       [ 2.54707143e-01, -7.51258372e-01],
       [-5.76294410e-01, -6.46266306e-01],
       [-7.85034274e-01, -1.07062866e-01],
       [-9.09233305e-01,  3.42748104e-01],
       [ 1.96444257e+00, -3.73205545e-01],
       [-8.79153234e-01, -5.55246052e-01],
       [-8.96093064e-01, -5.67861378e-01],
       [ 1.25866911e+00, -2.77708886e-01],
       [-2.32669173e-01, -6.73802985e-01],
       [-2.43250989e-02, -4.71957773e-01],
       [ 4.24263756e-01,  5.47659901e-02],
       [-6.83474454e-01, -3.95044980e-01],
       [ 1.88160562e-02,  1.77934977e-01],
       [ 2.93381245e+00, -2.48408774e-01],
       [ 1.08974576e+00, -2.96157104e-01],
       [ 4.50810840e+00, -1.87909470e-01],
       [ 2.23650890e+00, -6.99440582e-01],
       [-3.84019427e-01, -4.81453179e-01],
       [ 2.65472642e-01, -5.01122235e-01],
       [ 3.37189865e-01, -7.05680421e-01],
       [ 2.43693717e+00, -7.47460209e-01],
       [-7.35560472e-01, -4.93254613e-01],
       [-4.73230586e-01, -3.31290108e-01],
       [-7.25269921e-01, -3.67643950e-01],
       [-6.33604861e-01,  7.19851390e-01],
       [ 7.34009333e-01, -5.13194967e-01],
       [-3.49269028e-01, -5.28930212e-01],
       [ 5.60969765e-01, -6.09369869e-01],
       [-3.09531671e-01, -3.52315651e-01],
       [-4.14970237e-01, -1.67697819e-01],
       [-9.14774371e-01, -6.67020552e-01],
       [ 3.52071584e-01,  6.05988826e-02],
       [-5.31728409e-01,  1.02004188e+00],
       [-7.21470333e-01, -6.70465103e-02],
       [-7.44188703e-01, -2.85983454e-01],
       [-9.37176109e-01,  2.25683307e-01],
       [-9.30685146e-01,  9.68902337e-01],
       [-4.37926081e-01,  9.59135634e-01],
       [ 4.21651539e-01, -4.64225799e-01],
       [-3.96843036e-01, -6.47894090e-01],
       [-8.77253440e-01,  2.01556743e+00],
       [-3.93122606e-01, -5.83867921e-01],
       [ 1.39592922e+00, -4.69516097e-01],
       [-7.80759737e-01, -2.79336670e-01],
       [-1.75437879e-01, -6.60780713e-01],
       [-1.09499196e-01, -5.42766375e-01],
       [-8.32845756e-01,  1.24603256e+00],
       [-3.85760905e-01, -5.99738814e-01],
       [-8.56355706e-01, -3.67372653e-01],
       [-1.58551841e-02, -4.18240902e-01],
       [-1.77416831e-01,  2.61459193e+00],
       [-8.09494121e-01, -2.84898265e-01],
       [-2.25228313e-01, -5.28930212e-01],
       [ 3.07188951e-01, -6.89538230e-01],
       [ 1.07631425e-01, -6.58203388e-01],
       [ 1.10636895e+00, -2.99683969e-01],
       [ 1.63989443e+00,  1.49047145e+00],
       [-6.00516783e-01, -5.67861378e-01],
       [-9.07016879e-01, -6.64307578e-01],
       [ 3.90463254e-01, -4.86607828e-01],
       [-5.45185283e-01, -5.84681812e-01],
       [-1.74962930e-01, -5.43987213e-01],
       [-6.15293978e-02,  1.23233204e+00],
       [ 8.16967004e-01, -3.50009624e-01],
       [-2.71931582e-01, -1.10725380e-01],
       [ 3.81755865e-01, -6.60373767e-01],
       [ 1.19383864e+00,  1.38461216e-01],
       [ 1.55917855e-01, -5.81290596e-01],
       [-2.09713329e-01, -6.03536977e-01],
       [-9.49683086e-01, -7.41084722e-01],
       [-7.42763858e-01, -6.25105114e-01],
       [-9.19761330e-01, -2.40269854e-01],
       [-7.66273808e-01, -4.51247613e-03],
       [-9.29735249e-01, -5.19071945e-03],
       [-8.16222558e-01,  1.58401569e-01],
       [-7.09042514e-01, -6.27275493e-01],
       [-8.55089177e-01, -4.33297903e-01],
       [-7.30573513e-01,  1.55010353e-01],
       [ 1.21742774e+00,  1.88244276e-01],
       [-6.78566652e-01, -4.63411907e-01],
       [-4.76476067e-01,  1.19977636e+00],
       [-6.71204951e-01, -6.37177845e-01],
       [-8.54139280e-01,  5.76199455e-01],
       [-9.01792446e-01, -6.36228305e-01],
       [-9.40659064e-01,  6.34475046e-02],
       [ 1.61379763e-01,  2.16594846e-01],
       [-9.34880524e-01, -6.87639149e-01],
       [ 8.45859704e-01, -7.68078806e-01],
       [-9.46991711e-01, -6.13303681e-01],
       [-8.89839575e-01,  1.74549094e+00],
       [-8.86910726e-01,  1.10871202e-02],
       [ 6.13134941e-01, -3.96672764e-01],
       [-5.75186196e-01, -7.03916988e-01],
       [-7.82026267e-01, -5.69082216e-01],
       [-5.83656111e-01, -6.51014009e-01],
       [-7.42922174e-01, -6.67834444e-01],
       [-5.78510836e-01,  4.49992863e-02],
       [-1.93090131e-01, -4.85522639e-01],
       [-6.74687906e-01, -6.49858822e-01],
       [ 8.18075217e-01, -6.24019925e-01],
       [-4.58269708e-01, -7.40542128e-01],
       [ 2.19099300e+00, -2.51257396e-01],
       [ 6.41552693e-01, -5.30286698e-01],
       [-6.42470567e-01, -2.70655155e-01],
       [ 2.43466695e-01,  6.24979639e-02],
       [-4.48216632e-01, -4.80232341e-01],
       [-4.87558199e-01, -5.89565164e-01],
       [-7.01680813e-01,  3.84392244e-01],
       [ 2.12085894e+00, -3.58419841e-01],
       [-6.94635743e-01, -3.39429028e-01],
       [-6.29488641e-01, -8.57660259e-02],
       [ 1.28684938e+00, -6.58203388e-01],
       [ 4.56718570e-01, -1.63492710e-01],
       [ 1.77739202e+00,  2.22292090e-01],
       [-8.05773691e-01, -6.01230950e-01],
       [-1.04274762e-01,  2.17875277e+00],
       [-2.89162677e-02, -6.61187659e-01],
       [-8.54455912e-01, -2.95207563e-01],
       [ 7.05908214e-01, -5.26352887e-01],
       [-1.75358721e-01, -6.71225660e-01],
       [-2.07021954e-01, -5.78713271e-01],
       [-6.84582667e-01, -6.26054655e-01],
       [-7.40389115e-01, -4.11729766e-01],
       [-9.08916673e-01, -2.19244311e-01],
       [ 9.20743251e-01,  1.26659782e-01],
       [ 2.98190297e-02, -6.68919633e-01],
       [-1.89844650e-01, -6.07742085e-01],
       [-5.92521817e-01,  8.40307404e-01],
       [-3.68320760e-02, -5.60265053e-01],
       [-4.41937777e-02, -6.46266306e-01],
       [-1.89053069e-01, -7.07986448e-01],
       [-5.97983724e-01, -6.63358038e-01],
       [ 1.20056707e+00, -4.06303819e-01],
       [ 1.15085580e+00, -1.41517627e-01],
       [ 1.07700131e+00,  7.06693470e-01],
       [ 4.80703469e-01, -1.60508440e-01],
       [-4.04996319e-01, -6.44367225e-01],
       [ 1.03425594e+00, -1.06655920e-01],
       [-3.67158755e-01,  9.56015714e-01],
       [-2.96312271e-01, -2.59938911e-01],
       [-2.60611976e-01, -2.93308482e-01],
       [-4.24864997e-01, -5.01936127e-01],
       [-7.81630476e-01, -3.40107271e-01],
       [-9.42242226e-01, -2.97242293e-01],
       [-5.55475834e-01, -1.27138868e-01],
       [-4.77425964e-01, -5.16314886e-01],
       [-3.29162875e-01,  3.38760958e-02],
       [-6.02495735e-01,  6.97740658e-01],
       [-5.48984871e-01, -4.03237235e-02],
       [-8.97676225e-01,  3.65944026e-01],
       [-2.48500789e-01, -2.73096831e-01],
       [-5.97192143e-01, -7.19298622e-02],
       [ 4.00753805e-01, -1.28766652e-01],
       [ 1.13985282e+00, -5.99467517e-01],
       [ 4.40491163e-01, -2.86254751e-01],
       [ 3.94183684e-01, -3.95044980e-01],
       [-5.99462361e-02, -4.59613744e-01],
       [ 8.97416983e-02,  4.81652336e-01],
       [ 1.50492990e+00,  1.47731353e+00],
       [-7.08725882e-01,  2.35667973e-02],
       [-2.68606942e-01, -5.48870565e-01],
       [-7.04767978e-01, -2.12461878e-01],
       [-2.74464640e-01, -3.93741828e-02],
       [ 7.21660672e-01, -6.28631980e-01],
       [-7.93029240e-01, -3.49738326e-01],
       [ 3.74473322e-01, -2.54241667e-01],
       [ 1.40131197e+00,  8.48446324e-01],
       [ 2.15529271e+00, -5.92142489e-01],
       [ 2.00325540e-01,  1.31467078e+00],
       [-1.35383889e-01, -5.17535724e-01],
       [-7.29306983e-01, -5.55924296e-01]])
```

In [13]:
```
kmeans = KMeans(n_clusters=3, random_state=0)
```

In [14]:
```
kmeans.fit(df2_scaled)
```

Out[14]:
```
KMeans(n_clusters=3, random_state=0)
```

In [15]:
```
kmeans.predict(df2_scaled)
```

Out[15]:
```
array([2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2,
       1, 0, 1, 2, 2, 2, 0, 1, 2, 2, 1, 1, 2, 2, 1, 2, 2, 1, 1, 2, 2, 2,
       2, 0, 2, 0, 2, 0, 2, 2, 1, 2, 1, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 0,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 0, 0, 1,
       2, 1, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 2, 1, 2, 1, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2,
       1, 2, 2, 2, 2, 1, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2,
       2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 1, 1,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 1, 0, 1, 2, 2, 1, 1, 2, 2, 1, 2,
       2, 0, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 1, 1, 1, 1,
       2, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 0, 2, 1, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2,
       2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2,
       2, 2, 2, 2, 1, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2,
       2, 2, 2, 1, 2, 2, 1, 2, 1, 2, 0, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2,
       2, 2, 2, 2, 2, 1, 1, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 1, 1, 2, 2, 2])
```

In [16]:
```
kmeans.cluster_centers_
```

Out[16]:
```
array([[ 0.05716626,  3.3479732 ],
       [ 1.68779057, -0.10761111],
       [-0.35366401, -0.18028656]])
```

In [17]:
```
clusterlabels = kmeans.predict(df2_scaled)
```

Attach cluster labels to the original dataset

In [18]:
```
# Append the cluster labels to a new column in the original dataset.

# Show a preview of rows in the dataset with cluster labels added.

df2
```

Out[18]:

	Fresh	Milk
0	12669	9656
1	7057	9810
2	6353	8808
3	13265	1196
4	22615	5410
...	...	...
435	29703	12051
436	39228	1431
437	14531	15488
438	10290	1981
439	2787	1698

440 rows × 2 columns

In [19]:
```
df2["c_label"] = clusterlabels.copy()
```

```
<ipython-input-19-602f1198b807>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2["c_label"] = clusterlabels.copy()
```

In [20]:
```
df2
```

Out[20]:

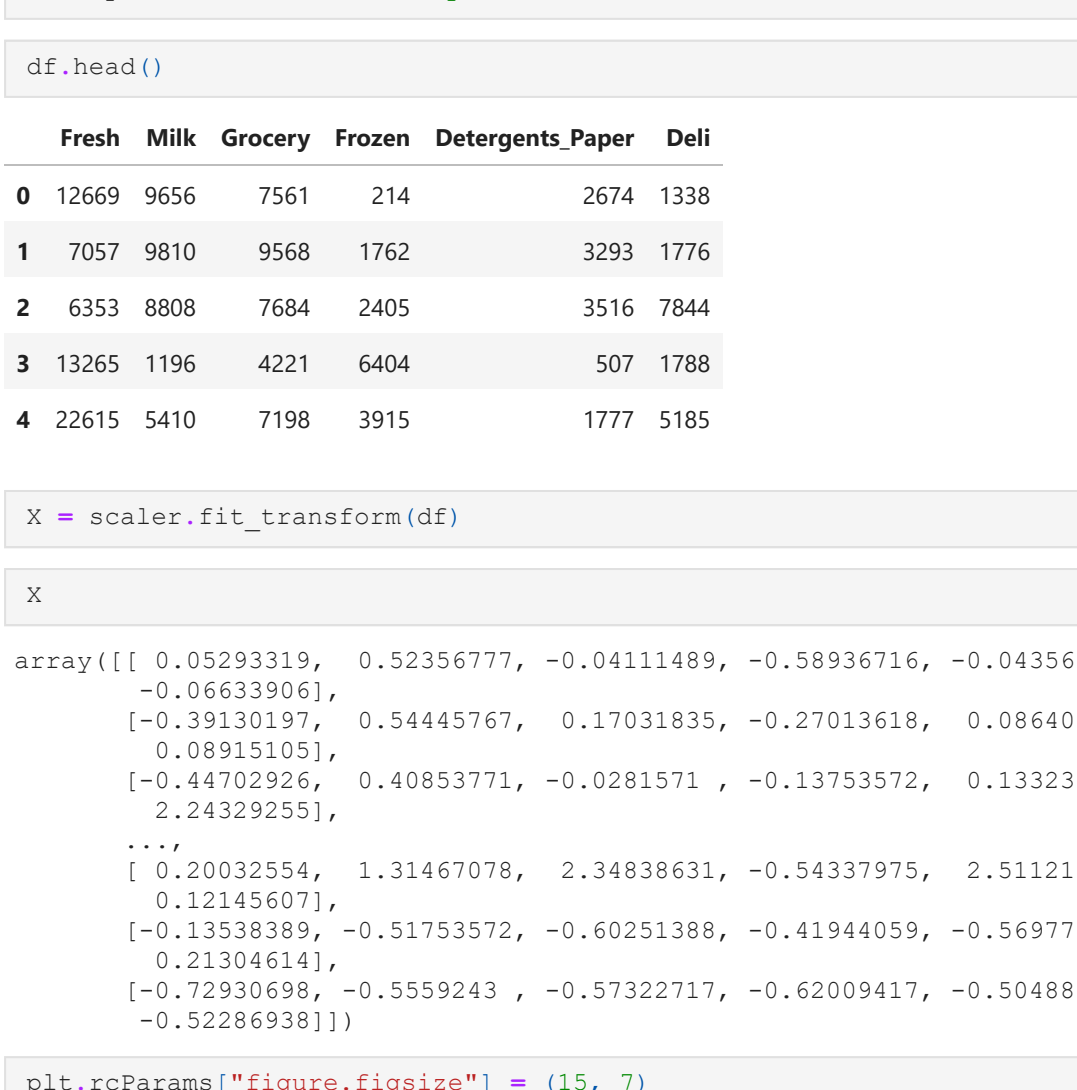
	Fresh	Milk	c_label
0	12669	9656	2
1	7057	9810	2
2	6353	8808	2
3	13265	1196	2
4	22615	5410	1
...	...	...	...
435	29703	12051	1
436	39228	1431	1
437	14531	15488	2
438	10290	1981	2
439	2787	1698	2

440 rows × 3 columns

Show clusters of customers based on fresh products and milk products sales

In [21]:
```
# Use Matplotlib to create a scatter plot of customer data.
# Use fresh products as the x-axis and milk products as the y-axis.
# Distinguish customers by color.
```

In [22]:
```
plt.figure(figsize=(8,8))

sns.scatterplot(x=df2.Fresh, y=df2.Milk, data=df2, hue=df2.c_label, palette=['r','y','k'])

plt.show()
```



Use the elbow method to determine the optimal number of clusters

In [23]:
```
# Set new DataFrame 'X' equal to the full dataset (all features).

# Use Yellowbrick to generate the elbow plot for 1 to 10 clusters on the full training data.

from yellowbrick.cluster import KElbowVisualizer
```

In [24]:
```
df.head()
```

Out[24]:

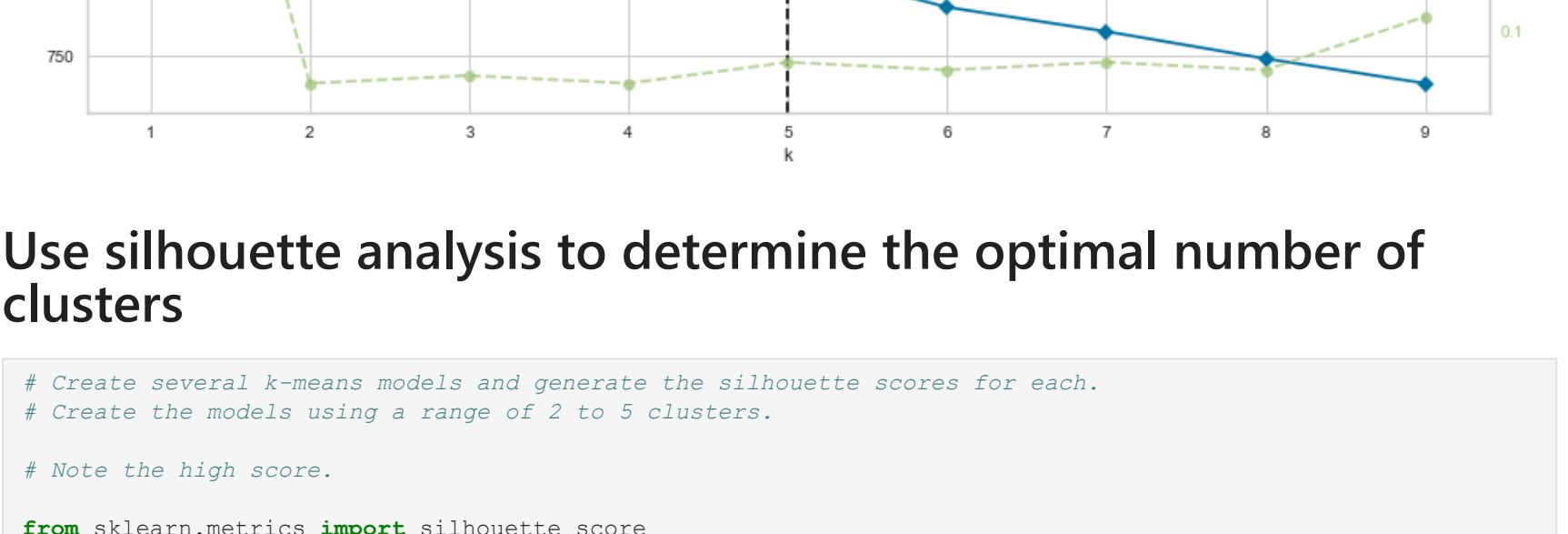
	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Deli
0	12669	9656	7561	214	2674	1338
1	7057	9810	9568	1762	3293	1776
2	6353	8808	7684	2405	3516	7844
3	13265	1196	4221	6404	507	1788
4	22615	5410	7198	3915	1777	5185

In [25]:
```
X = scaler.fit_transform(df)
```

In [26]:
```
X
```

Out[26]:
```
array([[ 0.05293319,  0.52356777, -0.04111489, -0.58936716, -0.04356873,
        -0.06633906],
       [-0.39130197,  0.54445767,  0.17031835, -0.27013618,  0.08640684,
         0.08915105],
       [-0.44702926,  0.40853771, -0.0281571 , -0.13753572,  0.13323164,
         2.24329255],
       ...,
       [ 0.20032554,  1.31467078,  2.34838631, -0.54337975,  2.51121768,
         0.12145607],
       [-0.13538389, -0.51753572, -0.60251388, -0.41944059, -0.56977032,
         0.21304614],
       [-0.72930698, -0.5559243 , -0.57322717, -0.62009417, -0.50488752,
        -0.52286938]])
```

In [27]:
```
plt.rcParams["figure.figsize"] = (15, 7)

visualizer = KElbowVisualizer(KMeans(init = 'k-means++', random_state = 0), k = (1, 10))
visualizer.fit(X)
visualizer.poof();
```



Use silhouette analysis to determine the optimal number of clusters

In [28]:
```
# Create several k-means models and generate the silhouette scores for each.
# Create the models using a range of 2 to 5 clusters.

# Note the high score.

from sklearn.metrics import silhouette_score
from sklearn.metrics import silhouette_samples
```

In [29]:
```
# The number of clusters to try out.
range_n_clusters = [2, 3, 4, 5]
```

In [30]:
```
for n in range_n_clusters:

    kmeans = KMeans(n_clusters = n, random_state = 10)
    cluster_labels = kmeans.fit_predict(X)
    silhouette_avg = silhouette_score(X, cluster_labels)
    print('\nWith {} clusters:'.format(n))
    print('    - Average silhouette score:', silhouette_avg)
```

```
With 2 clusters:
   - Average silhouette score: 0.5909041986285453

With 3 clusters:
   - Average silhouette score: 0.5391263624188073

With 4 clusters:
   - Average silhouette score: 0.3964105754321127

With 5 clusters:
   - Average silhouette score: 0.3690403826812808
```

In [ ]:

Generate and preview cluster labels using the full dataset

In [31]:

```python
# Construct a k-means clustering model class object and fit the full training data to it.
# Use your judgment to determine the desired number of clusters.

# Predict the cluster labels based on the full training data, then append the labels to the dataset.

# Show the first 20 rows in the dataset with cluster labels added.
```

In [32]:

```python
kmeans = KMeans(n_clusters = 2, random_state = 0)
cluster_labels = kmeans.fit_predict(X)
```

In [33]:

```python
cluster_labels
```

Out[33]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
       1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1,
       1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1])
```

In [34]:

```python
table = df.copy()
```

In [35]:

```python
df["c_label"] = cluster_labels.copy()
```

In [36]:

```python
df.head(20)
```

Out[36]:

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Deli	c_label
0	12669	9656	7561	214	2674	1338	1
1	7057	9810	9568	1762	3293	1776	1
2	6353	8808	7684	2405	3516	7844	1
3	13265	1196	4221	6404	507	1788	1
4	22615	5410	7198	3915	1777	5185	1
5	9413	8259	5126	666	1795	1451	1
6	12126	3199	6975	480	3140	545	1
7	7579	4956	9426	1669	3321	2566	1
8	5963	3648	6192	425	1716	750	1
9	6006	11093	18881	1159	7425	2098	0
10	3366	5403	12974	4400	5977	1744	1
11	13146	1124	4523	1420	549	497	1
12	31714	12319	11757	287	3881	2931	1
13	21217	6208	14982	3095	6707	602	1
14	24653	9465	12091	294	5058	2168	1
15	10253	1114	3821	397	964	412	1
16	1020	8816	12121	134	4508	1080	1
17	5876	6157	2933	839	370	4478	1
18	18601	6327	10099	2205	2767	3181	1
19	7780	2495	9464	669	2518	501	1

In [ ]: