# AMOD 5440H – Data Mining Assignment 2: Cluster Analysis

Jugal Chauhan – 0755856

## Introduction:

Cluster analysis is a data mining method that groups similar data together in order to form clusters. The goal is to divide dataset in such a way that data points in one group have similar properties and away from data points that have different properties. Clustering is useful in identifying patterns or relationships within data that may not be initially discovered. K-means clustering, hierarchical clustering and density-based clustering are some of the famous clustering algorithms. It has applications in many fields such as understanding customer behaviour patterns and anomaly detection. In this study we will apply clustering algorithms k-means clustering and dbscan on a wholesale dataset which includes the annual spending in monetary units (m.u.) on diverse product categories. We will evaluate the performance of algorithms on several performance matrix and obtain valuable insights from the wholesale data.

## Dataset Description

The wholesale data (Cardoso, 2014) contains 440 variables and 8 attributes of integer type. Detailed description of data is given in table 1. The data does not contain any null or duplicated values.

| Sr No. | Attribute | Description | Data Type | Mean | Sd |
|---|---|---|---|---|---|
| 1 | Channel | Customer channel (Ex: Hotel or Retail) | Nominal | - | - |
| 2 | Region | Customer region | Nominal | - | - |
| 3 | Fresh | Annual spending (m.u.) on fresh products | Continuous | 12000.30 | 12647.329 |
| 4 | Milk | Annual spending on milk products | Continuous | 5796.27 | 7380.377 |
| 5 | Grocery | Annual spending on grocery products | Continuous | 7951.28 | 9503.16 |
| 6 | Frozen | Annual spending on frozen products | Continuous | 3071.93 | 4854.67 |
| 7 | Detergents_Paper | Annual spending on detergents and paper products | Continuous | 2881.49 | 4767.85 |
| 8 | Delicatessen | Annual spending on delicatessen products | Continuous | 1524.87 | 2820.10 |

*Table 1: Wholesale Dataset Description*

Method

The distribution of data and its attributes is visualized using boxplot and histogram. Box plot reveals presence of few outliers within data; however, these are ignored as it does not affect the final analysis of data.

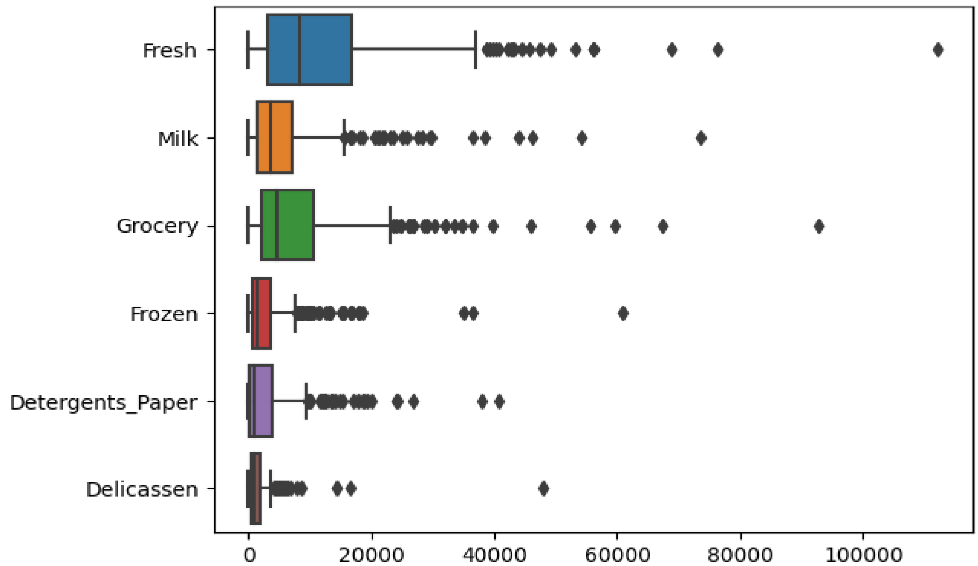



*Figure 1: Box plot of Wholesale Data*

Furthermore, histogram reveals non-normal distribution of data. K-means clustering depends on the distance between data points, so it is best to normalize the data before applying the algorithm. The distribution of data before normalization is shown in Figure 2 and after normalization is shown in Figure 3.

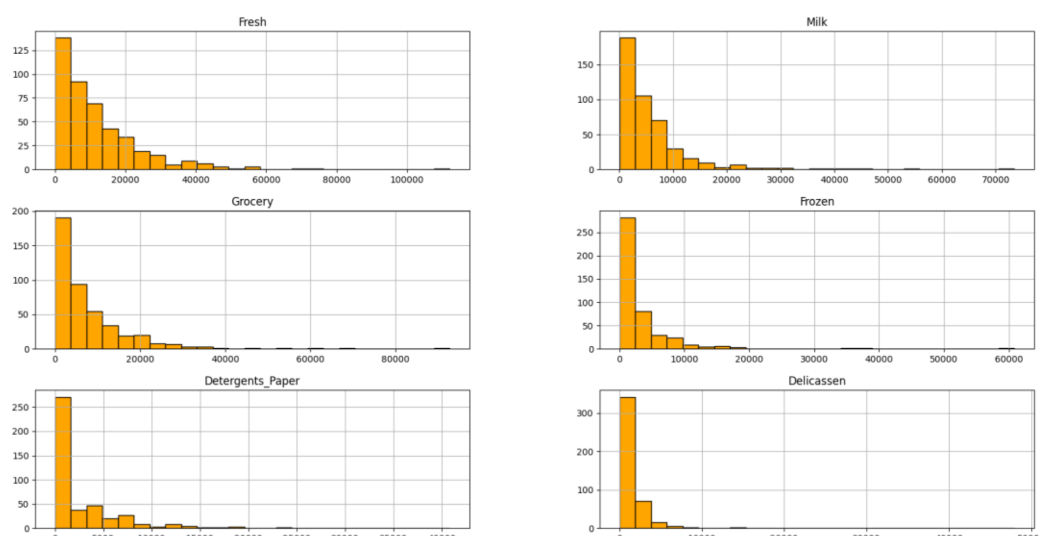



*Figure 2: Distribution before Normalization*

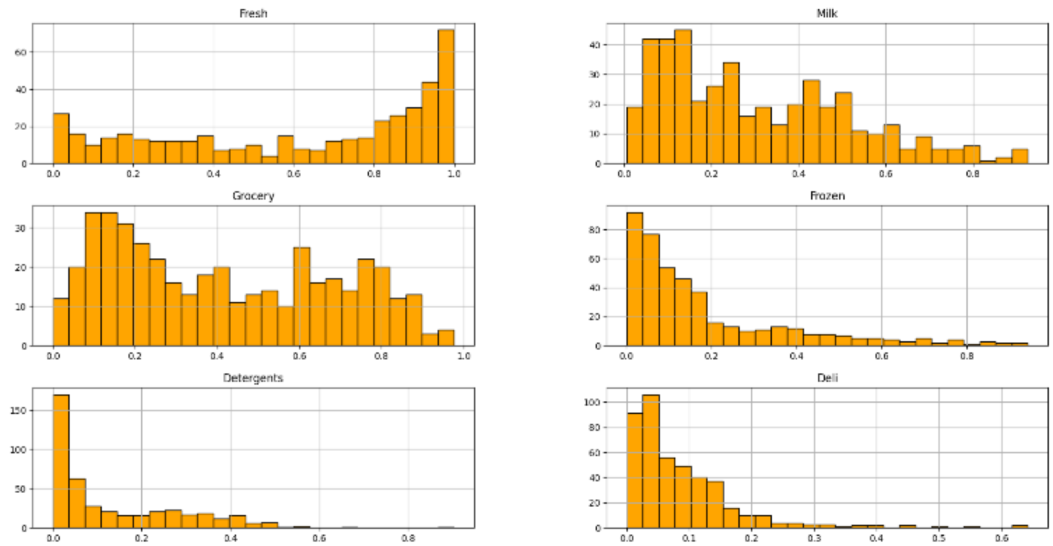



*Figure 3: Distribution after Normalization*

After normalization of data, it is ready for k-means clustering and DBSCAN. K-means clustering is prototype based, partitional clustering technique where number of clusters to be formed is decided and then each data point is assigned to closest centroid, the entire collection of points assigned to centroid ultimately forms a cluster. This process is repeated until no point changes cluster.

The important aspect is to determine the number of clusters to form. It is done by a trial-error method where we apply several values of number of cluster and determine the best option using an elbow graph. The k-means algorithm is implemented in python using sci-kit learn library. The k-means algorithm assigns labels to each data points that indicates the cluster that the data points belong. The clusters can be analyzed by visualizing the data points based on the cluster labels assigned to them.

Another type of clustering algorithm that is based on density of data, known as dbscan is used on the wholesale data. It groups data points together based on density criterion, that is dense regions are defined as clusters separated by regions of lower density, particularly noise or outliers. Two important parameters eps and minimum samples are decided based on trial-error methods and applied to wholesale data.

<u>Results</u>

On determining the number of clusters for k-means, the optimum value obtained through the elbow chart is n_clusters = 2. Thus, the wholesale data is divided into two clusters based on the centroid regions. The cluster labels are visualized for individual attributes using a pairplot shown in figure 4.

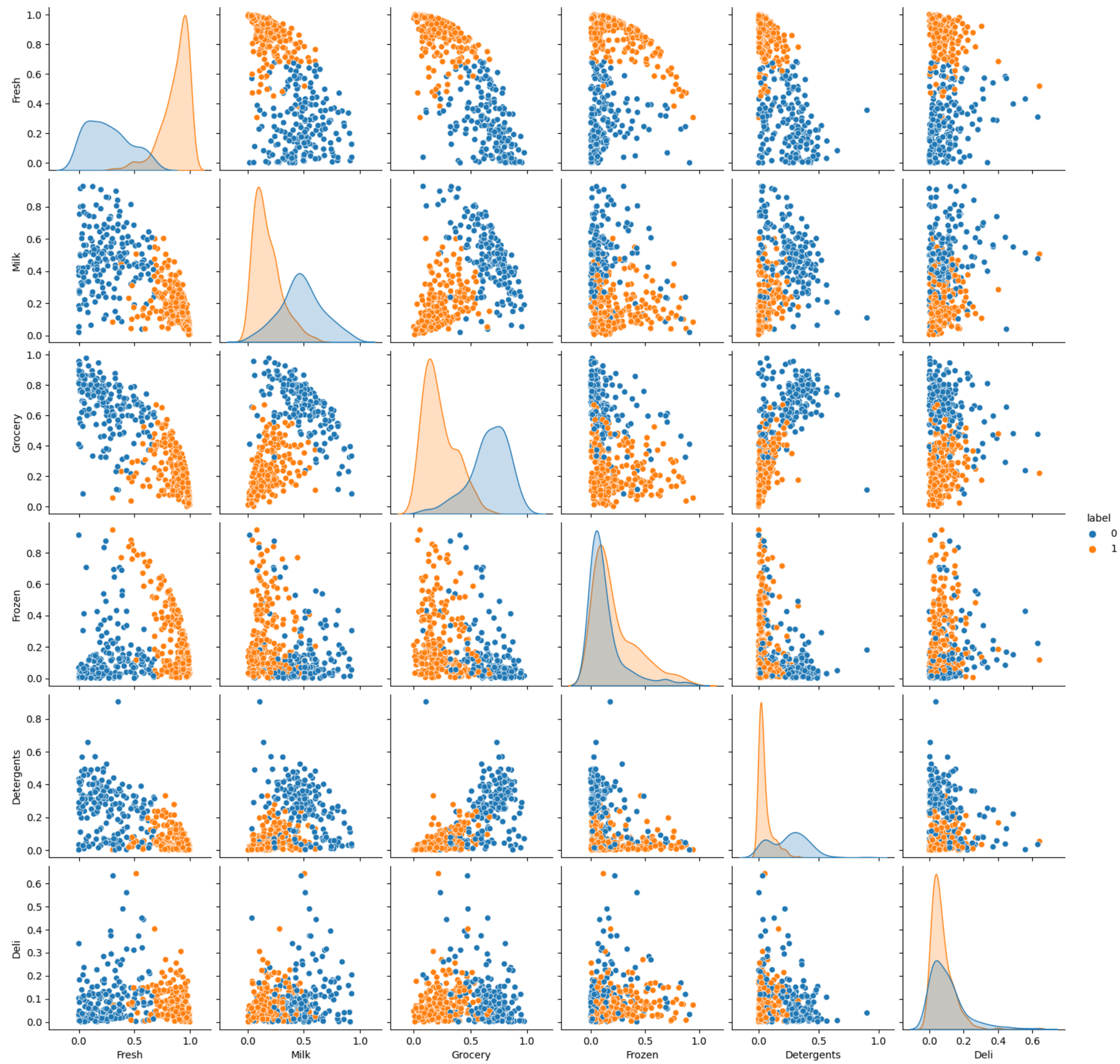



*Figure 4: K-means clustering pair plot*

However, these results can be better visualized using a parallel coordinates plot shown in figure 5. Additionally, similar plot for the centroid value shows even more insightful information regarding the grouping of attributes.

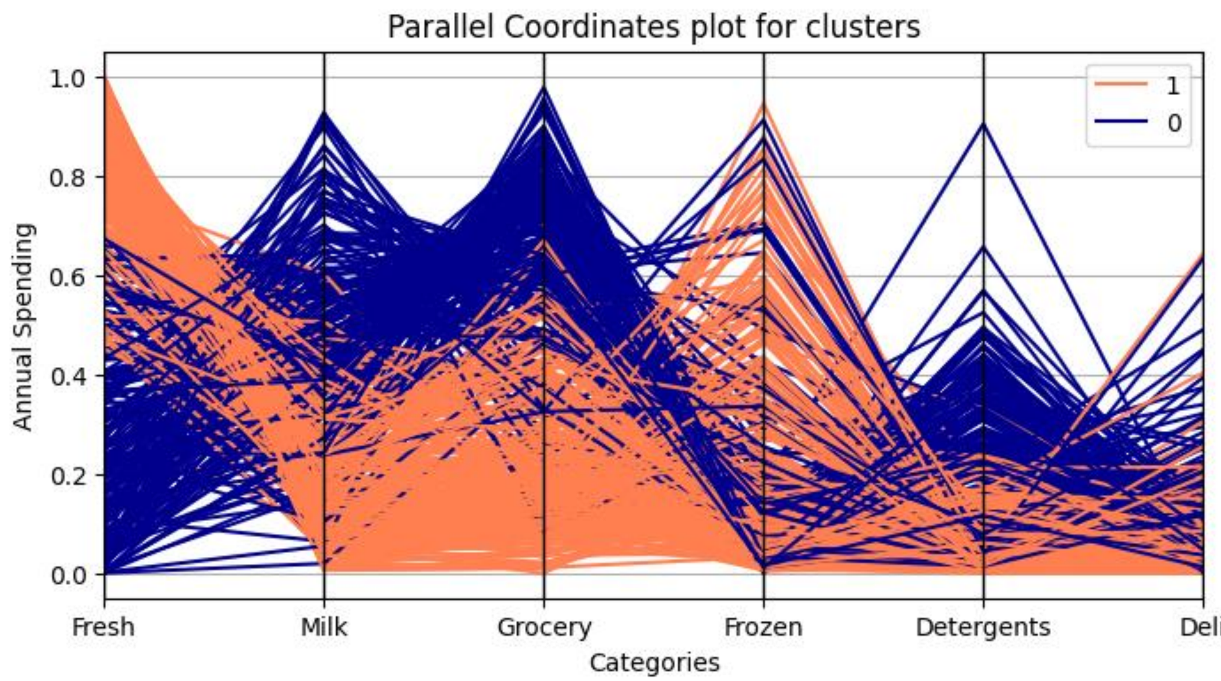



*Figure 5: Parallel Coordinates plot k-means clustering*

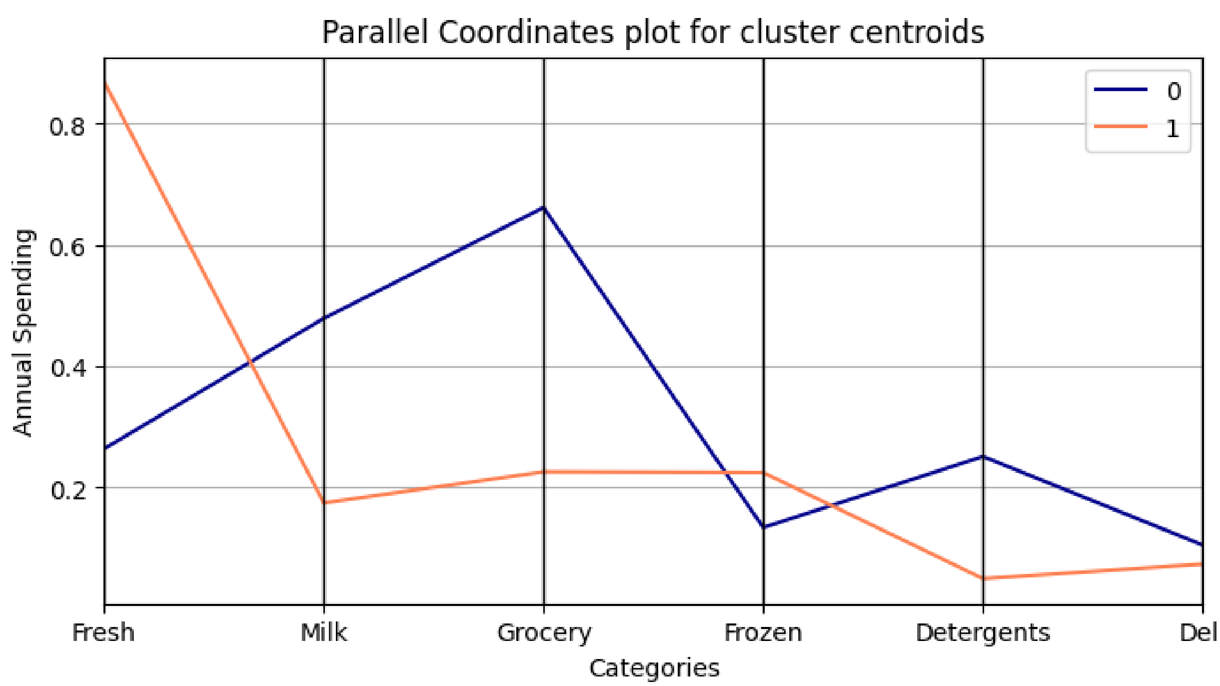



*Figure 6: Parallel Coordinates plot for centroids*

The performance of both dbscan and k-means clustering is evaluated using silhouette score that gives an index of performance score for both the algorithms. We obtain the best silhouette score of 0.5 for k-means when n_clusters=2 and in case of dbscan we obtain slightly lower score of 0.422.

## Discussion

From the results, we obtain a better silhouette score for k-means clustering indicating it is a better choice for wholesale data. The visualization indicates that customers belonging to the cluster label 0 spend more on grocery tend to spend more on milk and less on frozen and deli goods, indicating a behaviour pattern of retail stores. Additionally, the second cluster label indicates customer spending highly on fresh goods spend less on milk, detergents, and deli indicating these could be the restaurant/hotel customers. Thus, in this way we can perform customer segmentation using cluster analysis.

## Conclusion

Based on the results, k-means clustering was determined to be a better choice for the wholesale data, indicating its effectiveness in customer segmentation. The visualization and analysis of the clusters revealed distinct spending patterns for different customer segments, such as retail stores and restaurant/hotel customers, based on their preferences for specific product categories.

In conclusion, cluster analysis proved valuable in understanding customer behavior patterns and segmenting customers in the wholesale dataset, providing actionable insights for targeted marketing strategies and business decision-making.

Appendix 1: References

1. Cardoso,Margarida. (2014). Wholesale customers. UCI Machine Learning Repository. https://doi.org/10.24432/C5030X.

Appendix 2: Python Code

## Importing Required Libraries

In [78]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.cluster import KMeans
from pandas.plotting import parallel_coordinates
from sklearn.metrics import silhouette_samples, silhouette_score
from scipy.cluster.hierarchy import linkage, dendrogram
from sklearn.neighbors import NearestNeighbors
from sklearn import cluster
from sklearn import metrics
```

Loading the dataset into a variable

In [2]:
```
customer = pd.read_csv('Wholesale customers data.csv')
customer.head()
```

Out[2]:

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

The data set refers to clients of a wholesale distributor. It includes the annual spending in monetary units (m.u.) on diverse product categories. Let's explore the wholesale customers data

### Data Preprocessing

In [3]:
```
customer.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440 entries, 0 to 439
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Channel           440 non-null    int64
 1   Region            440 non-null    int64
 2   Fresh             440 non-null    int64
 3   Milk              440 non-null    int64
 4   Grocery           440 non-null    int64
 5   Frozen            440 non-null    int64
 6   Detergents_Paper  440 non-null    int64
 7   Delicassen        440 non-null    int64
dtypes: int64(8)
memory usage: 27.6 KB
```

The data contains 8 attributes with 440 rows, all of integer data type.

In [4]:
```
customer.isnull().sum()
```

Out[4]:
```
Channel             0
Region              0
Fresh               0
Milk                0
Grocery             0
Frozen              0
Detergents_Paper    0
Delicassen          0
dtype: int64
```

There are no null values is our data!

In [5]:
```
customer.duplicated().sum()
```

Out[5]:
```
0
```

There are no duplicates in the data!

Let's see the summary statistics for wholesale customer data

In [6]:
```
customer.describe()
```

Out[6]:

| | Channel | Region | Fresh | Milk | Grocery | Frozen | Deterg |
|---|---|---|---|---|---|---|---|
| **count** | 440.000000 | 440.000000 | 440.000000 | 440.000000 | 440.000000 | 440.000000 | |
| **mean** | 1.322727 | 2.543182 | 12000.297727 | 5796.265909 | 7951.277273 | 3071.931818 | 2 |
| **std** | 0.468052 | 0.774272 | 12647.328865 | 7380.377175 | 9503.162829 | 4854.673333 | 4 |
| **min** | 1.000000 | 1.000000 | 3.000000 | 55.000000 | 3.000000 | 25.000000 | |
| **25%** | 1.000000 | 2.000000 | 3127.750000 | 1533.000000 | 2153.000000 | 742.250000 | |
| **50%** | 1.000000 | 3.000000 | 8504.000000 | 3627.000000 | 4755.500000 | 1526.000000 | |
| **75%** | 2.000000 | 3.000000 | 16933.750000 | 7190.250000 | 10655.750000 | 3554.250000 | 3 |
| **max** | 2.000000 | 3.000000 | 112151.000000 | 73498.000000 | 92780.000000 | 60869.000000 | 4( |

We can observe that attributes have high variability

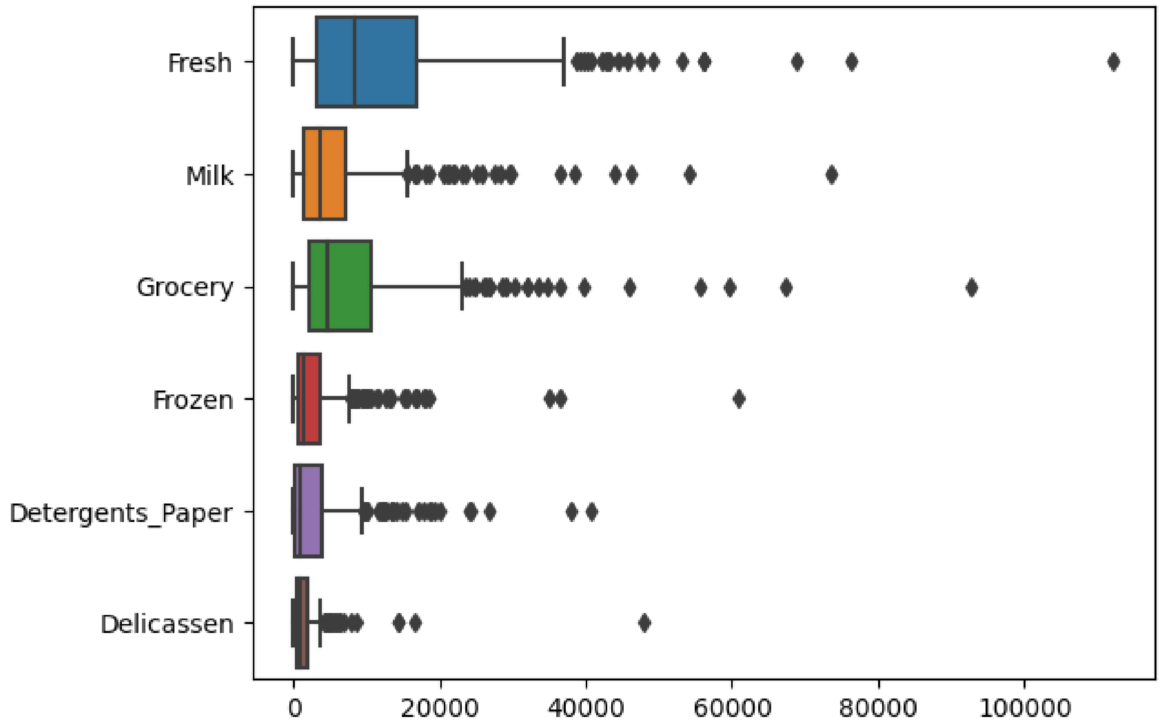
Let's remove the channel and region columns from our data for better analysis of numerical values only.

In [7]:
```
customer = customer.drop('Channel', axis=1)
customer = customer.drop('Region', axis=1)
```

Let's visualize our data using box plot and histogram to learn the distribution of data

In [8]:
```
sns.boxplot(data = customer, orient = 'h')
```

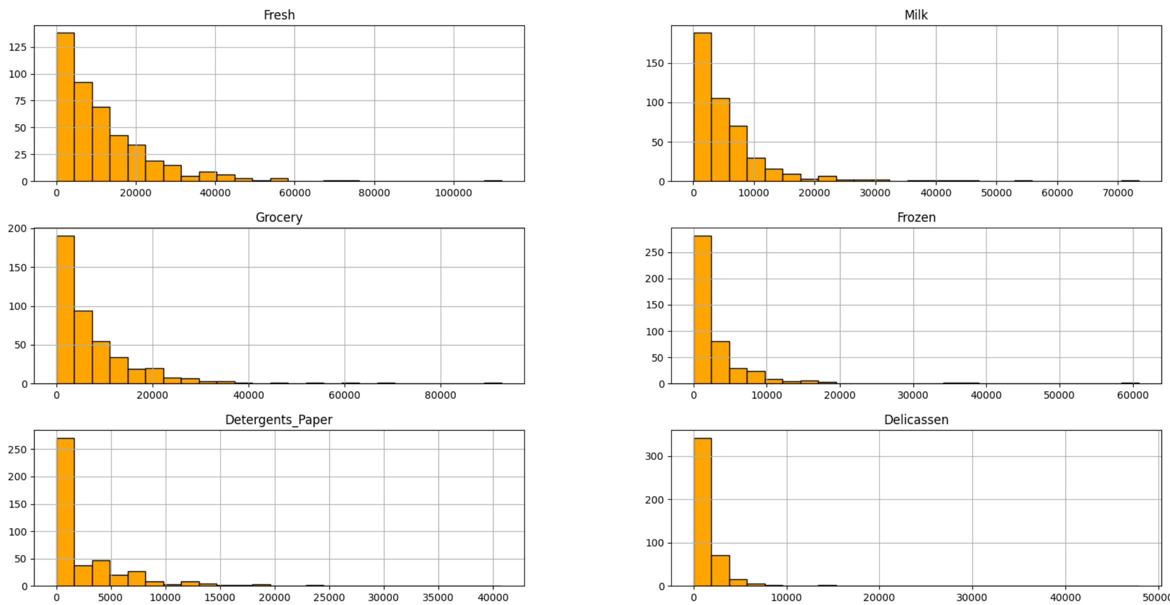
Out[8]: <Axes: >




Let's check the distribution of attributes

In [9]:
```python
customer.hist(figsize = (20,10), bins = 25, color='orange', edgecolor='black')
```

Out[9]:
```
array([[<Axes: title={'center': 'Fresh'}>,
        <Axes: title={'center': 'Milk'}>],
       [<Axes: title={'center': 'Grocery'}>,
        <Axes: title={'center': 'Frozen'}>],
       [<Axes: title={'center': 'Detergents_Paper'}>,
        <Axes: title={'center': 'Delicassen'}>]], dtype=object)
```



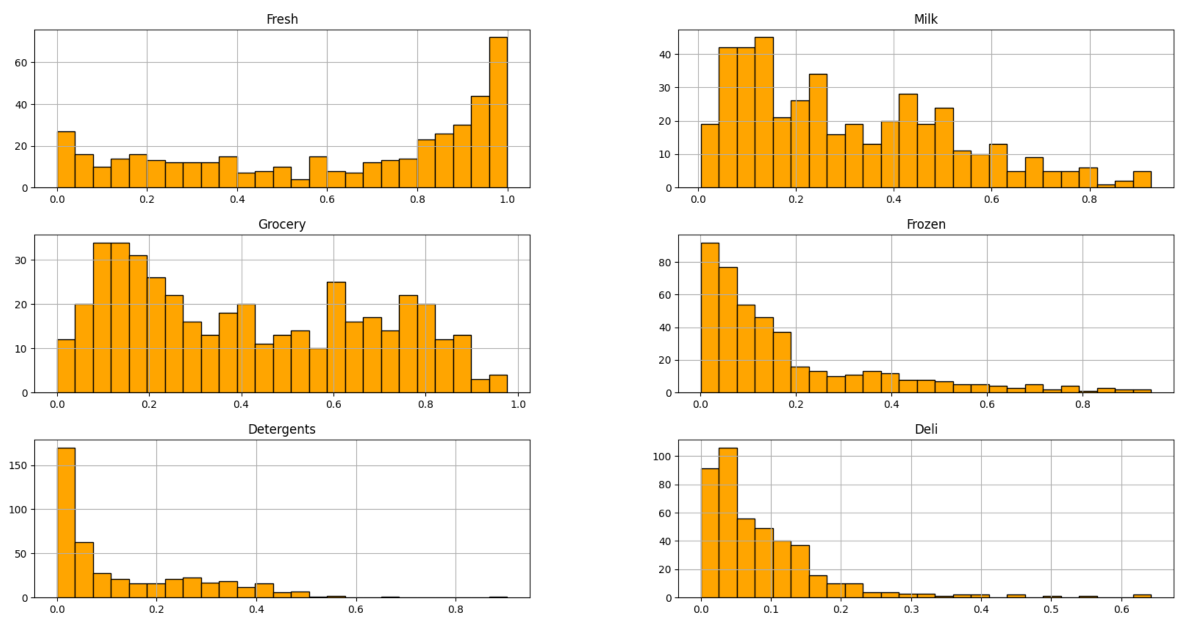

From the histogram we can observe that the data is not normally distributed. Thus, we will normalize the wholesale customers data.

We will use sci-kit learn library to normalize the data.

In [10]:
```
cust_norm = preprocessing.normalize(customer)
cust_norm = pd.DataFrame(cust_norm)
cust_norm.rename(columns = {0:'Fresh',1:'Milk',2:'Grocery',3:'Frozen',4:'Deter
cust_norm.info()
cust_norm.hist(figsize = (20,10), bins = 25, color='orange', edgecolor='black'
```

```
<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    float64
 1   Milk        440 non-null    float64
 2   Grocery     440 non-null    float64
 3   Frozen      440 non-null    float64
 4   Detergents  440 non-null    float64
 5   Deli        440 non-null    float64
dtypes: float64(6)
memory usage: 20.8 KB
```

Out[10]:
```
array([[<Axes: title={'center': 'Fresh'}>,
        <Axes: title={'center': 'Milk'}>],
       [<Axes: title={'center': 'Grocery'}>,
        <Axes: title={'center': 'Frozen'}>],
       [<Axes: title={'center': 'Detergents'}>,
        <Axes: title={'center': 'Deli'}>]], dtype=object)
```




This doesn't look like a normal distribution but its an improvement from what we had before. We have completed the preprocessing steps. Now let's apply the K-means clustering algorithm

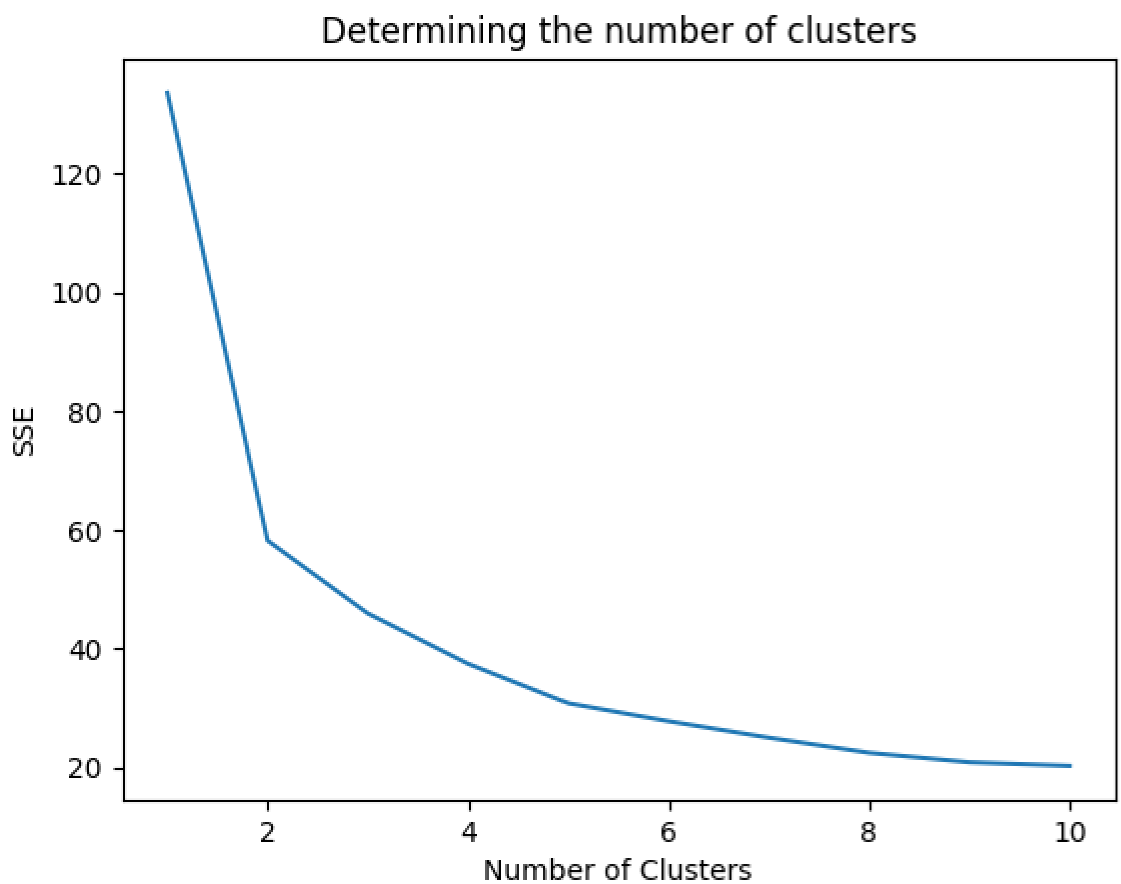
### K-means Clustering

We will apply kmeans clustering from sci-kit learn library. We will apply the number of clusters ranging from 1 to 10 and then select the most appropriate number of clusters using the elbow method

In [11]:
```
SSE = []
clusters = [1,2,3,4,5,6,7,8,9,10]
for k in clusters:
    kmeans = KMeans(n_clusters = k, n_init='auto')
    kmeans.fit(cust_norm)
    SSE.append(kmeans.inertia_)

plt.plot(clusters, SSE)
plt.xlabel('Number of Clusters')
plt.ylabel('SSE')
plt.title('Determining the number of clusters')
```

Out[11]:
```
Text(0.5, 1.0, 'Determining the number of clusters')
```




Looking at the elbow plot, we can see that 2 number of clusters seems to be the ideal choice for this dataset. So let's fit the algorithm again for 2 clusters.

In [12]:
```
kmeans1 = KMeans(n_clusters=2, n_init='auto')
customer_kmeans = kmeans1.fit_predict(cust_norm)
```

Now, let's obtain the labels from our model and check if every data point has been assigned a label or not

In [13]:
```
len(cust_norm) == len(kmeans1.labels_)
```

Out[13]:
```
True
```

We have labels for every record. Now let's concat labels field to our data and rename the columns

In [14]:
```
labels = pd.DataFrame(kmeans1.labels_)
labels.rename(columns = {0:'label'}, inplace = True)
data = pd.concat([cust_norm, labels], axis=1)
```

In [15]:
```
data.head()
```

Out[15]:

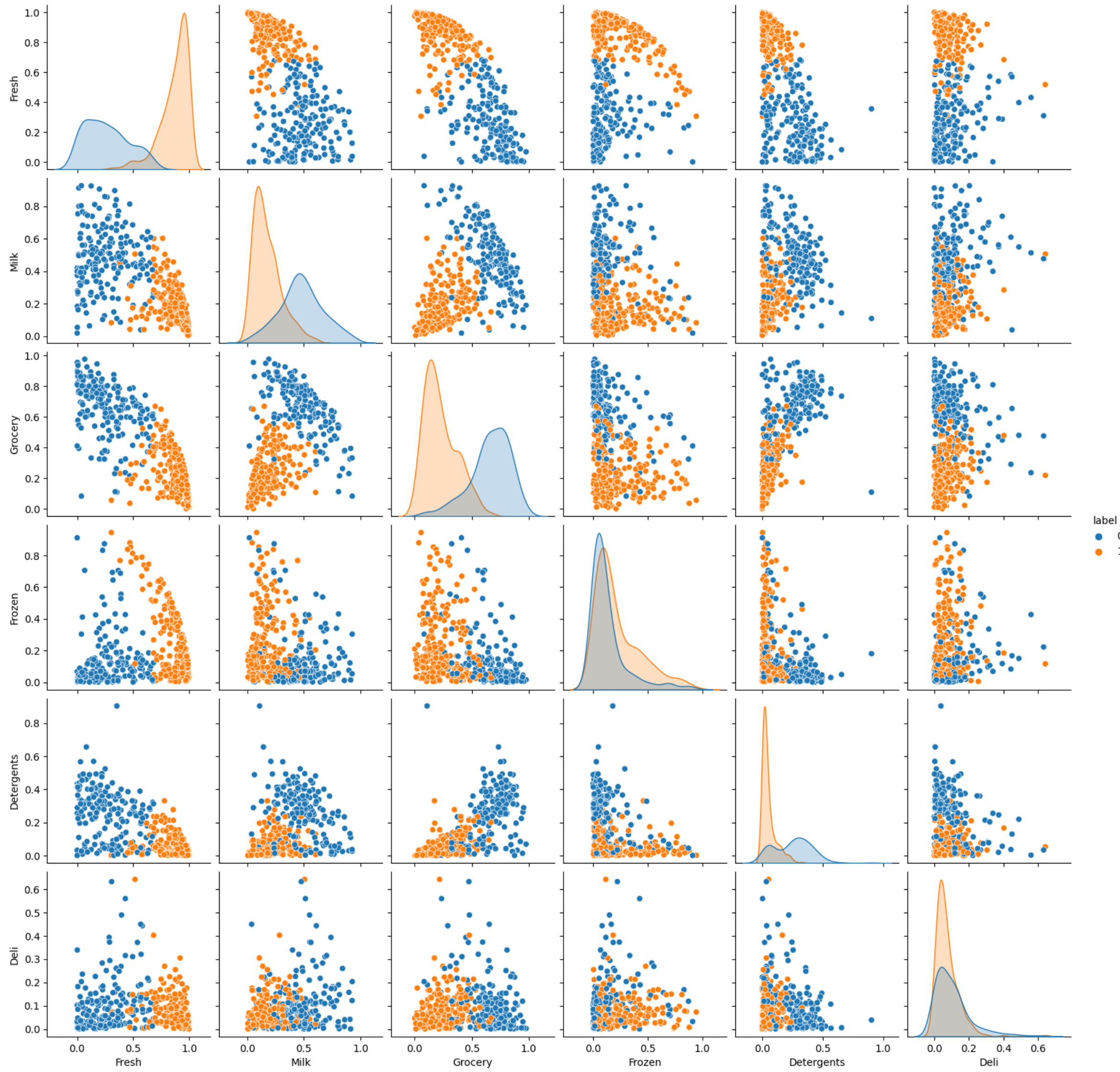
| | Fresh | Milk | Grocery | Frozen | Detergents | Deli | label |
|---|---|---|---|---|---|---|---|
| **0** | 0.708333 | 0.539874 | 0.422741 | 0.011965 | 0.149505 | 0.074809 | 1 |
| **1** | 0.442198 | 0.614704 | 0.599540 | 0.110409 | 0.206342 | 0.111286 | 0 |
| **2** | 0.396552 | 0.549792 | 0.479632 | 0.150119 | 0.219467 | 0.489619 | 0 |
| **3** | 0.856837 | 0.077254 | 0.272650 | 0.413659 | 0.032749 | 0.115494 | 1 |
| **4** | 0.895416 | 0.214203 | 0.284997 | 0.155010 | 0.070358 | 0.205294 | 1 |

## Visualizing Clusters

Finally, let's visualize our clustering model using scatterplots

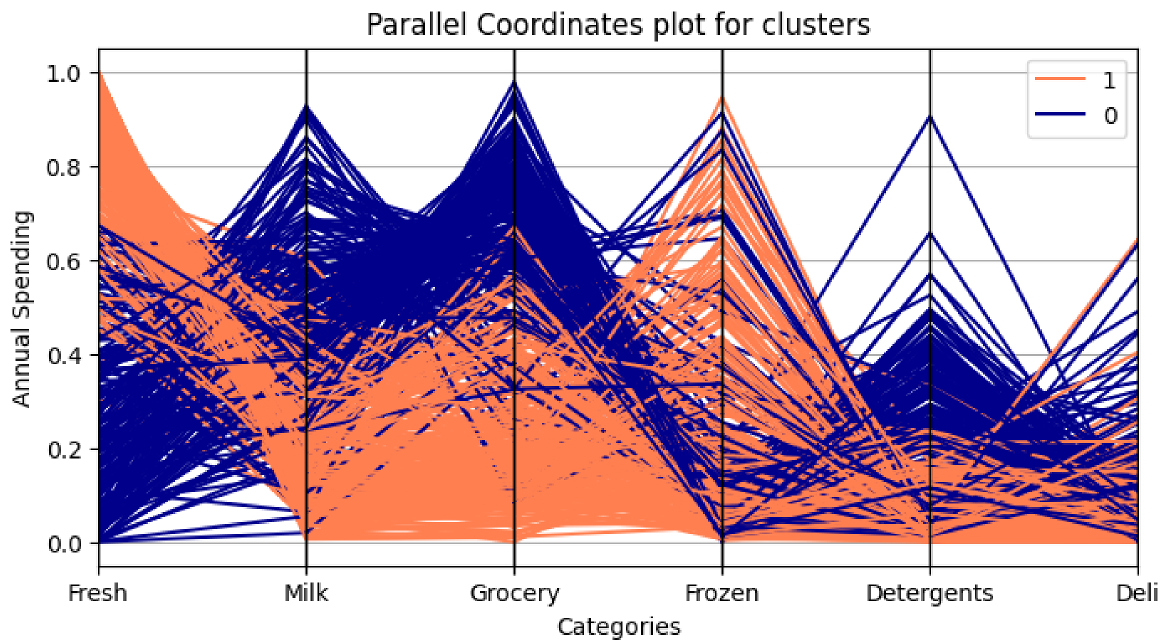
```python
In [16]: sns.pairplot(data, hue = 'label')
```

Out[16]: <seaborn.axisgrid.PairGrid at 0x1e326451950>




This is bit difficult to interpret, let's plot a parallel coordinates plot for the clusters to obtain better understanding of results.

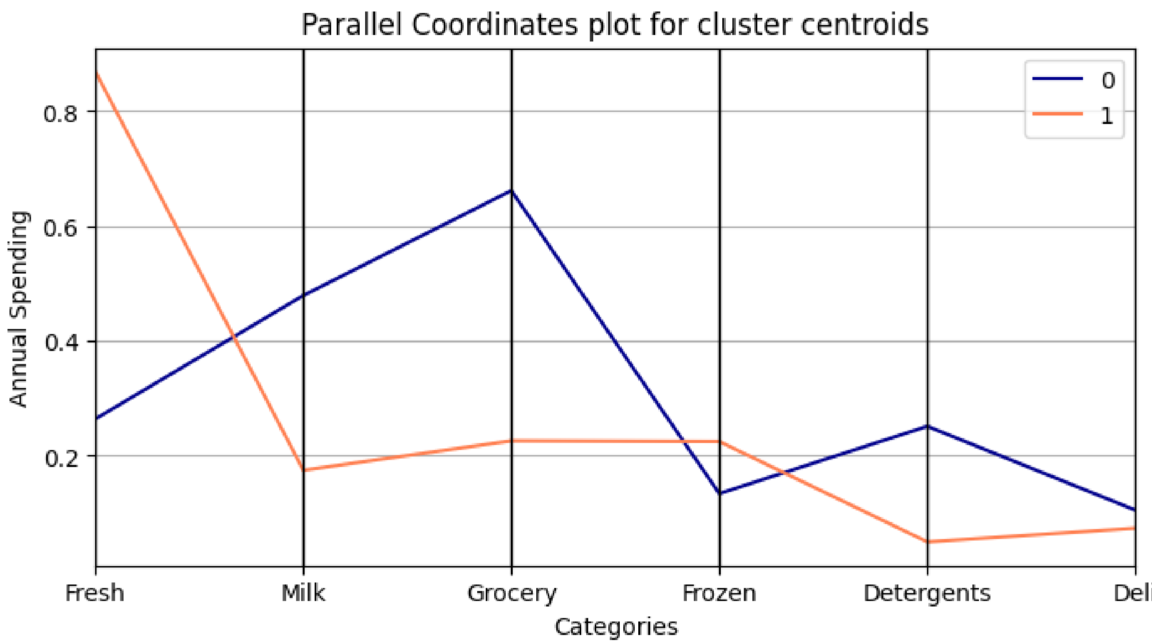
In [17]:
```
plt.figure(figsize=(8,4))
parallel_coordinates(data, 'label', color=('coral','darkblue'))
plt.title('Parallel Coordinates plot for clusters')
plt.ylabel('Annual Spending')
plt.xlabel('Categories')
plt.show()
```




We will also plot the parallel coordinates plot for the centroids of the clustering result.

In [18]:
```
# Create a dataframe containing centroids value from clustering output
centroids = pd.DataFrame(kmeans1.cluster_centers_, columns = data.columns[0:6]
centroids['cluster'] = centroids.index
```

In [109]:
```
plt.figure(figsize=(8,4))
parallel_coordinates(centroids, 'cluster', color=('darkblue','coral'))
plt.title('Parallel Coordinates plot for cluster centroids')
plt.ylabel('Annual Spending')
plt.xlabel('Categories')
plt.show()
```




## Silhouette Analysis

In [20]:
```
n_clusters = [2,3,4,5,6,7,8,9,10]
for k in n_clusters:
    sil_kmeans = KMeans(n_clusters = k, n_init='auto')
    cluster_labels = sil_kmeans.fit_predict(cust_norm)

    #The silhouette score gives the average value for all the samples
    sil_avg = silhouette_score(cust_norm,cluster_labels)
    print("For n_cluster =",k,"The average silhouette score is: ",sil_avg)
```

```
For n_cluster = 2 The average silhouette score is:  0.5002248259665941
For n_cluster = 3 The average silhouette score is:  0.4365632328906848
For n_cluster = 4 The average silhouette score is:  0.3798290278483874
For n_cluster = 5 The average silhouette score is:  0.3747453809922325
For n_cluster = 6 The average silhouette score is:  0.3643696320545916
For n_cluster = 7 The average silhouette score is:  0.3195688089575444
For n_cluster = 8 The average silhouette score is:  0.3329818912816675
For n_cluster = 9 The average silhouette score is:  0.2680799795960205
For n_cluster = 10 The average silhouette score is:  0.3141539002326934
```

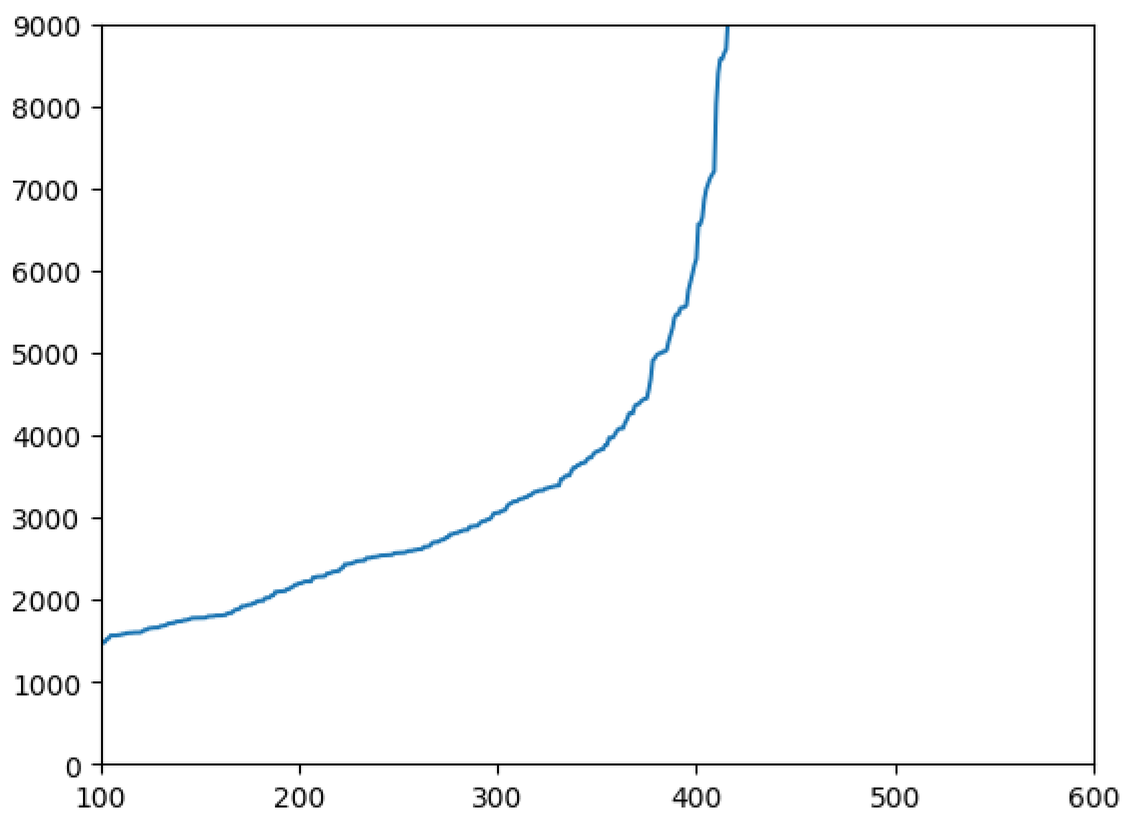
Thus, the highest silhouette score is obtained for n_clusters = 2.

## DBSCAN

Let's pick min_samples = 20 and We will determine the eps value using knn

In [110]:
```
neighbors = NearestNeighbors(n_neighbors=20)
neighbors_fit = neighbors.fit(customer)
distances, indices = neighbors_fit.kneighbors(customer) #will find k neighbors
```

In [111]:
```
distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.plot(distances)
plt.ylim(0,9000)
plt.xlim(100,600)
plt.show()
```

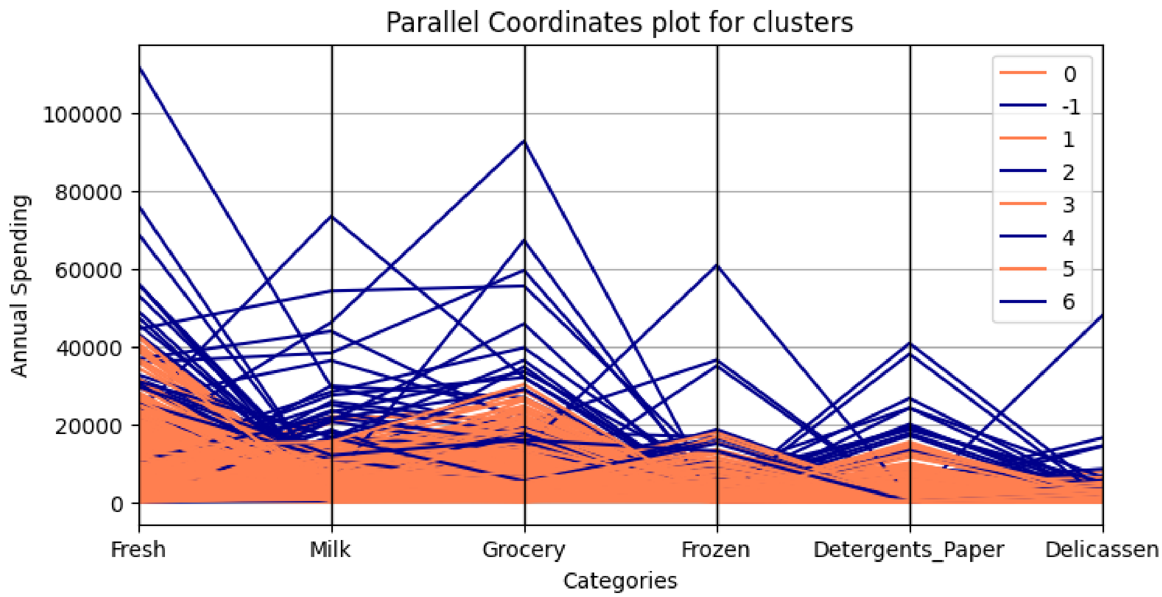



We determine for minimum 20 samples in a cluster the eps value to be 5000

In [112]:
```
dbscan = cluster.DBSCAN(eps=6000, min_samples=2)
clustering_labels = dbscan.fit_predict(customer.to_numpy())
```

In [113]:
```
customer['db_label'] = clustering_labels
```

In [114]:
```
plt.figure(figsize=(8,4))
parallel_coordinates(customer, 'db_label', color=('coral','darkblue'))
plt.title('Parallel Coordinates plot for clusters')
plt.ylabel('Annual Spending')
plt.xlabel('Categories')
plt.show()
```




In [115]:
```
metrics.silhouette_score(customer, customer['db_label'])
```

Out[115]: 0.14647081024822362

In [ ]: