CLUSTERING

Submitted by:

Seerat Chhabra

Finding similar time series in sales transaction data

Data Summary

- Dataset taken from UCI machine learning repository
- Weekly purchase quantities of 811 products for 52 weeks
- Also, contains normalized weekly purchase for 52 weeks

Normalized data

```
] #using only normalised values from training data
  required_columns = [col for col in trainData.columns if "Normalized" in col]
  trainData_new = trainData[required_columns]
  product_code = trainData["Product_Code"]
  trainData_new.head()
  print("Shape of training data with only normalized columns:" ,trainData_new.shape)
```

```
Shape of training data with only normalized columns: (811, 52)
```

Default K Means Clustering

Default model with :
- # of clusters : 8
- initial cluster centers : k-means++ method
- Number of time the k-means algorithm will be run with different centroid seeds: 10

```
[9] # using default k means with 8 clusters
    kmeans_model = KMeans()
    kmeans_model.fit(trainData_new)
    print("Silhouette score for clustering:",   silhouette_score(trainData_new, kmeans_model.labels_))
    print("Sum of squared distances for clustering:", kmeans_model.inertia_)
```

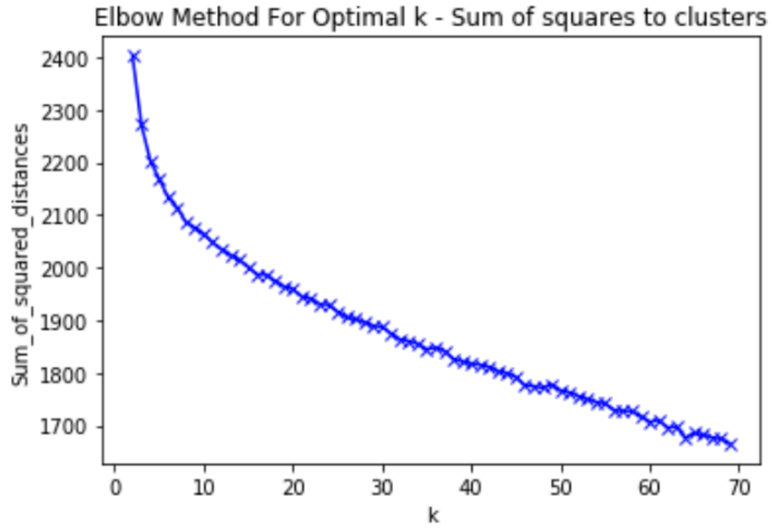
```
Silhouette score for clustering: 0.04161917200714276
Sum of squared distances for clustering: 2089.6171117859985
```

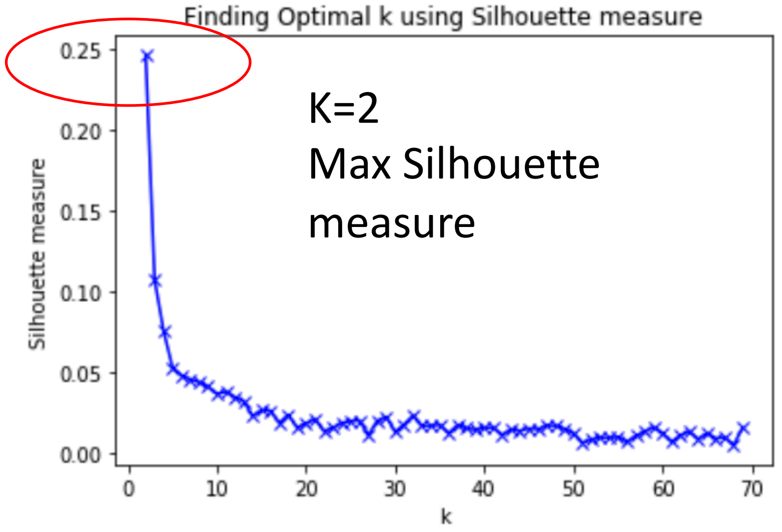
Finding optimal value of K

```
#Finding optimal value for number of clusters -k
Sum_of_squared_distances = []
sil_mat = []
K = range(2,70)
for k in K:
    kmeans_model = KMeans(n_clusters=k)
    kmeans_model = kmeans_model.fit(trainData_new)
    Sum_of_squared_distances.append(kmeans_model.inertia_)
    sil_mat.append(silhouette_score(trainData_new, kmeans_model.labels_))
```

```
#Plot elbow curve using sum of sqaures
plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k - Sum of squares to clusters')
plt.show()

#Plot elbow curve using silhouette measure
plt.plot(K, sil_mat, 'bx-')
plt.xlabel('k')
plt.ylabel('Silhouette measure')
plt.title('Finding Optimal k using Silhouette measure')
plt.show()
```





K means with optimal k clusters

```
# using k means with clusters =2
kmeans_model = KMeans(n_clusters= 2)
kmeans_model.fit(trainData_new)
print("Silhouette score for clustering:",   silhouette_score(trainData_new, kmeans_model.labels_))
print("Sum of squared distances for clustering:", kmeans_model.inertia_)
```

# of clusters : 2

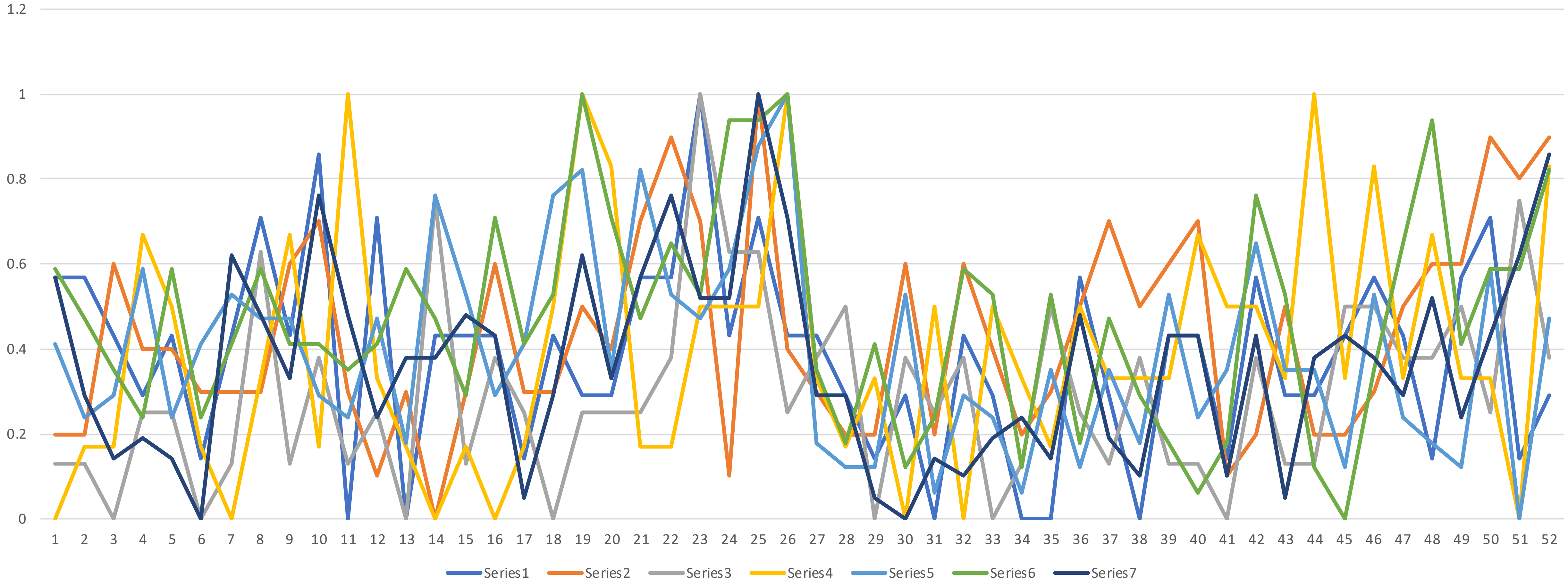
```
Silhouette score for clustering: 0.24637060375945008
Sum of squared distances for clustering: 2402.669678646214
```

```
kmeans_model.labels_
```

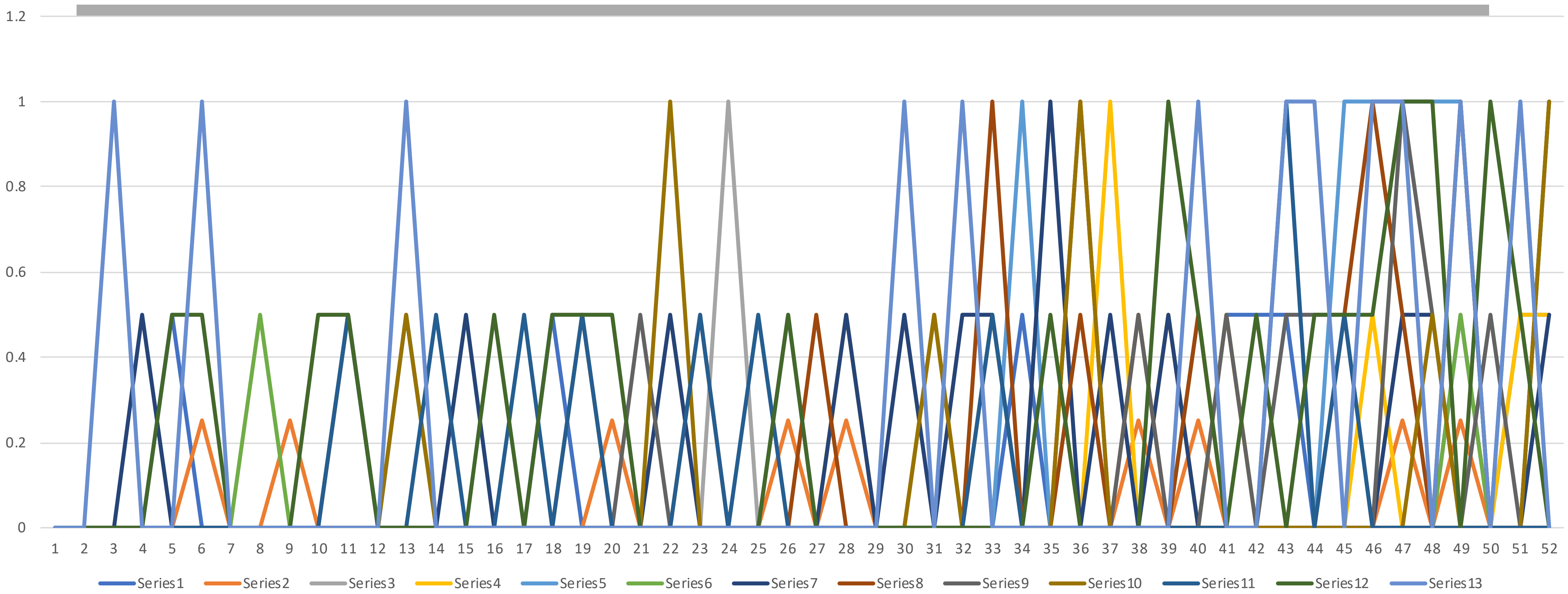
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
```

Divided into 0 and 1 cluster

Few samples from cluster -1



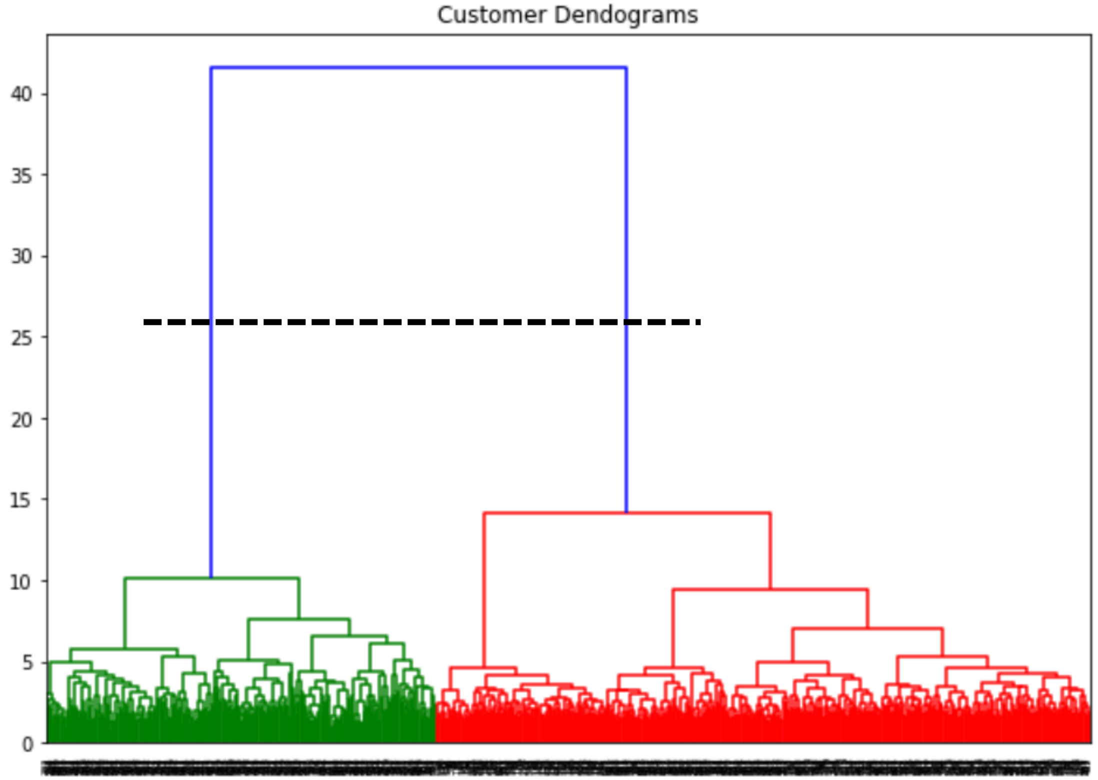
Few samples from cluster -0



AGGLOMERATIVE CLUSTERING

```
import scipy.cluster.hierarchy as shc

plt.figure(figsize=(10, 7))
plt.title("Customer Dendograms")
dend = shc.dendrogram(shc.linkage(trainData_new, method='ward'))
```



# of clusters =2

```
from sklearn.cluster import AgglomerativeClustering

cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage='ward')
cluster.fit_predict(trainData_new)
print("Silhouette score for clustering:", silhouette_score(trainData_new, cluster.fit_predict(trainData_new)))
```

```
Silhouette score for clustering: 0.24625996406892794
```

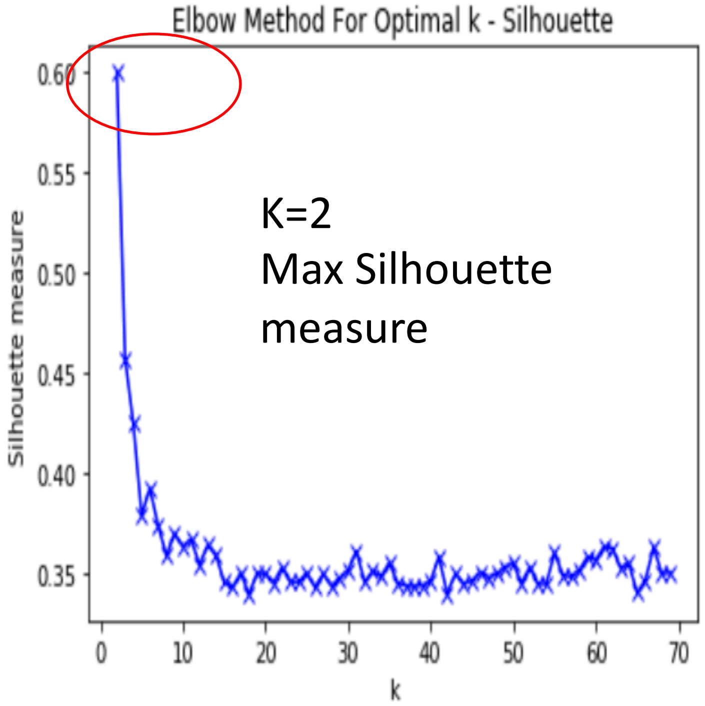
PCA & K means

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(trainData_new)
principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'principal component 2'])
```

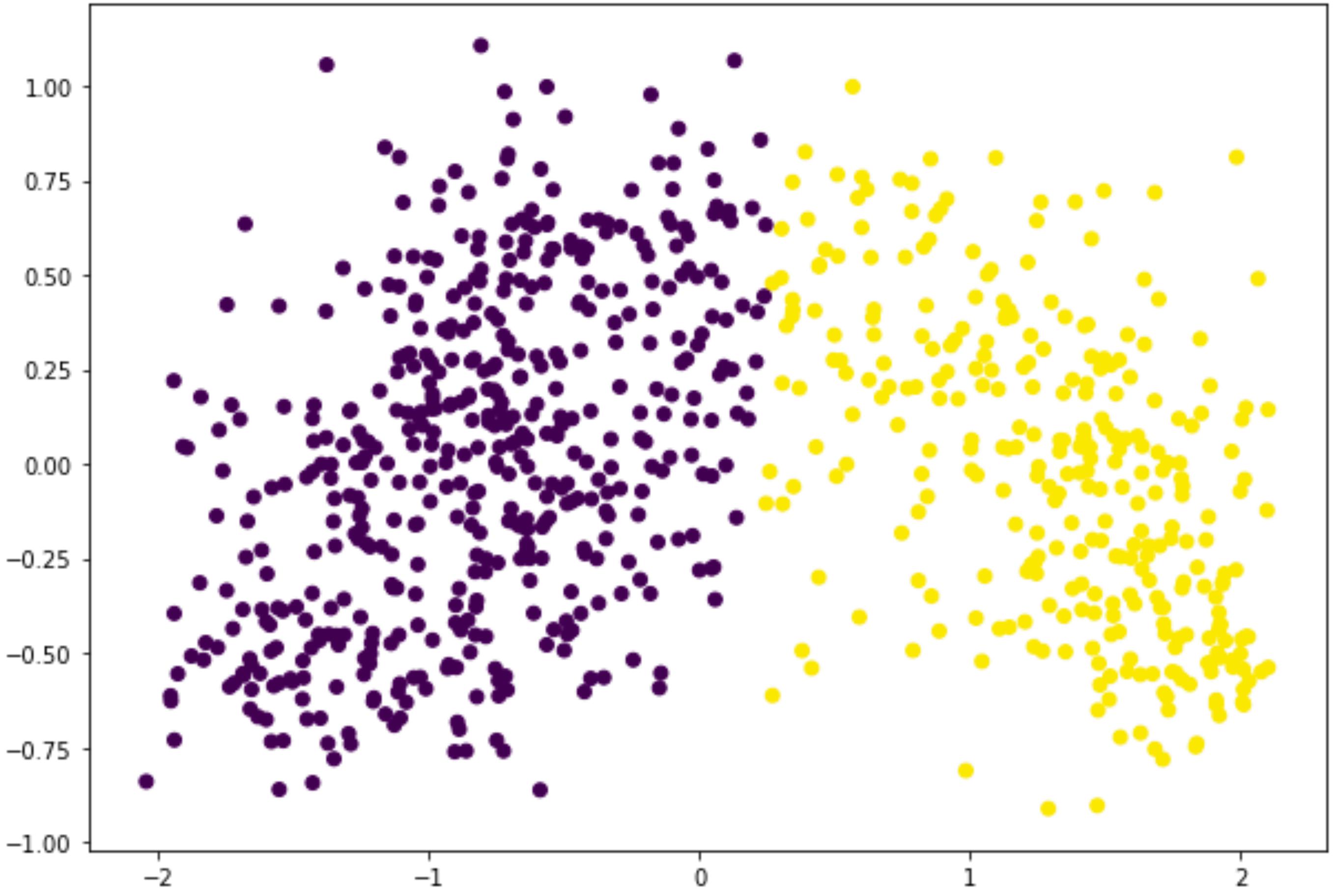
2 PCA Components

```
# using default k means
kmeans_model = KMeans(n_clusters= 2, init='k-means++', n_jobs = -1)
kmeans_model.fit(principalDf)
print("Silhouette score for clustering:",   silhouette_score(principalDf, kmeans_model.labels_))
print("Sum of squared distances for clustering:", kmeans_model.inertia_)
```

```
Silhouette score for clustering: 0.599992827302872
Sum of squared distances for clustering: 359.2476058189002
```



Clusters After PCA + K Means



DBSCAN

```python
from sklearn.cluster import DBSCAN
clustering = DBSCAN(metric = "kulsinski", min_samples= 5, leaf_size = 20)
clustering.fit(trainData_new)
print(clustering.labels_)
print("Silhouette score for clustering:",   silhouette_score(trainData_new, clustering.labels_))
```

```
[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  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 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0  0  0
  0  0  0  0  0  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0  0 -1 -1
 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0 -1  0  0  0  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0  0  0  0  0
  0 -1  0  0  0  0  0  0  0  0  0  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0  0  0  0  0
  0  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 -1 -1 -1  0 -1 -1 -1 -1 -1  0 -1
 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1 -1 -1 -1 -1
 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1
 -1 -1  0 -1 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0 -1  0  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  0  0  0  0
 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1  0  0 -1 -1 -1 -1  0 -1 -1
  0  0  0 -1  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0 -1
 -1 -1 -1  0 -1 -1 -1  0  0 -1 -1 -1 -1 -1 -1 -1  0  0  0 -1 -1 -1 -1  0
 -1  0 -1 -1 -1 -1 -1 -1 -1  0 -1  0 -1 -1  0  0  0 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0 -1 -1  0  0  0  0  0
  0 -1  0  0  0  0  0  0 -1 -1  0  0  0  0 -1  0 -1 -1 -1]
Silhouette score for clustering: 0.2449400923947693
```

Created 2 clusters- divided into 0 and -1 cluster

Comparison

Clustering Technique	# of clusters	Silhouette measure
Default K means	8	0.041
Optimal K means	2	0.246
Agglomerative clustering	2	0.246
PCA +K means	2	0.599
DBSCAN	2	0.244

Clustering model selection using external variables

Data Summary

Dataset taken from UCI machine learning repository

Data consists of annual spending of 440 clients of wholesale distributor for 2 channels and 3 regions

Annual spending of 6 product categories – Fresh, milk, grocery, frozen, detergent_paper, delicatessen

Remove Channel & Region

```
# remove channel and region
trainData = trainData.iloc[:, 2:]
trainData.head()
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
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

K means

Default model with :
- # of clusters : 8
- initial cluster centers : k-means++ method
- Number of time the k-means algorithm will be run with different centroid seeds: 10

```
[4] # using default k means has 8 clusters
    kmeans_model = KMeans()
    kmeans_model.fit(trainData)
    print("Silhouette score for clustering:",   silhouette_scor
    print("Sum of squared distances for clustering:", kmeans_model.inertia_)
```

```
Silhouette score for clustering: 0.3642754191240703
Sum of squared distances for clustering: 36242393778.11966
```

```
[12] #Random Hyperparameter tuning for neural networks model
     print("RandomizedSearchCV-Kmeans")
     parameters={'n_clusters': range(2,30,1), 'init': ['k-means++', 'random'],'tol': [0.0001,0.00001]}
     nn_random = RandomizedSearchCV(kmeans_model,parameters, cv=5)
     nn_random.fit(trainData)
     nn_rand_parm=nn_random.best_params_
     print(nn_rand_parm)
```

Hyperparameter tuning – Random search

```
RandomizedSearchCV-Kmeans
{'tol': 0.0001, 'n_clusters': 28, 'init': 'k-means++'}
```

```
# using default k means
kmeans_model = KMeans(**nn_rand_parm)
kmeans_model.fit(trainData)
print("Silhouette score for clustering:",   silhouette_score(trainData, kmeans_model.labels_))
print("Sum of squared distances for clustering:", kmeans_model.inertia_)
```

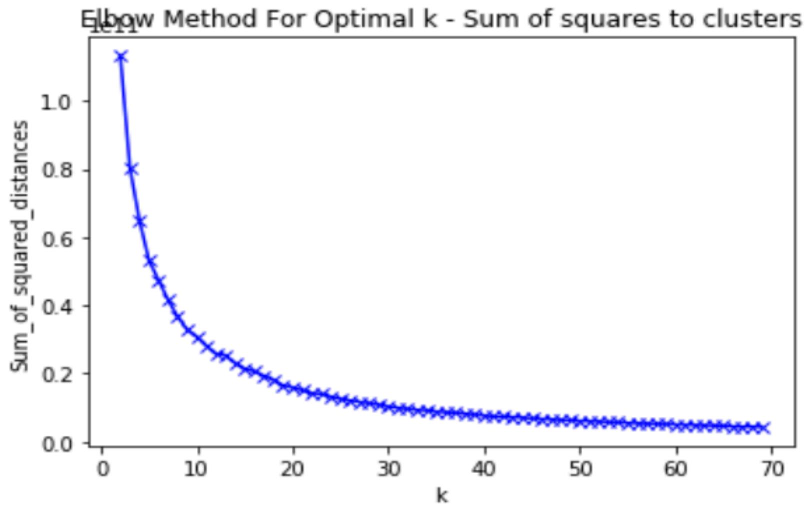
```
Silhouette score for clustering: 0.260645554453426
Sum of squared distances for clustering: 11053166574.037376
```

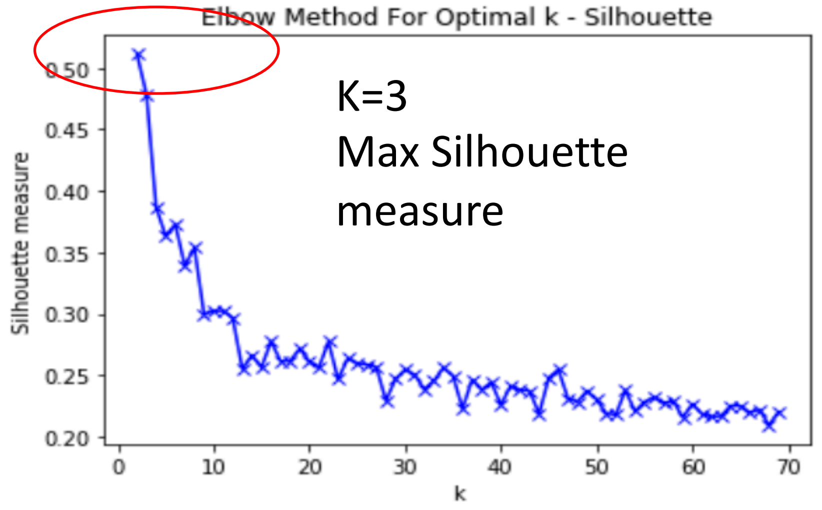
Finding optimal value of K

```python
#Finding optimal value for number of clusters -k
Sum_of_squared_distances = []
sil_mat = []
K = range(2,70)
for k in K:
    kmeans_model = KMeans(n_clusters=k)
    kmeans_model = kmeans_model.fit(trainData_new)
    Sum_of_squared_distances.append(kmeans_model.inertia_)
    sil_mat.append(silhouette_score(trainData_new, kmeans_model.labels_))
```

```python
#Plot elbow curve using sum of sqaures
plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k - Sum of squares to clusters')
plt.show()

#Plot elbow curve using silhouette measure
plt.plot(K, sil_mat, 'bx-')
plt.xlabel('k')
plt.ylabel('Silhouette measure')
plt.title('Finding Optimal k using Silhouette measure')
plt.show()
```





K means with optimal clusters

```
[ ] # using default k means
    kmeans_model = KMeans(n_clusters = 3)
    kmeans_model.fit(trainData)
    print("Silhouette score for clustering:",   silhouette_score(trainData, kmeans_model.labels_))
    print("Sum of squared distances for clustering:", kmeans_model.inertia_)
```

```
Silhouette score for clustering: 0.4783511093743595
Sum of squared distances for clustering: 80332419250.14606
```

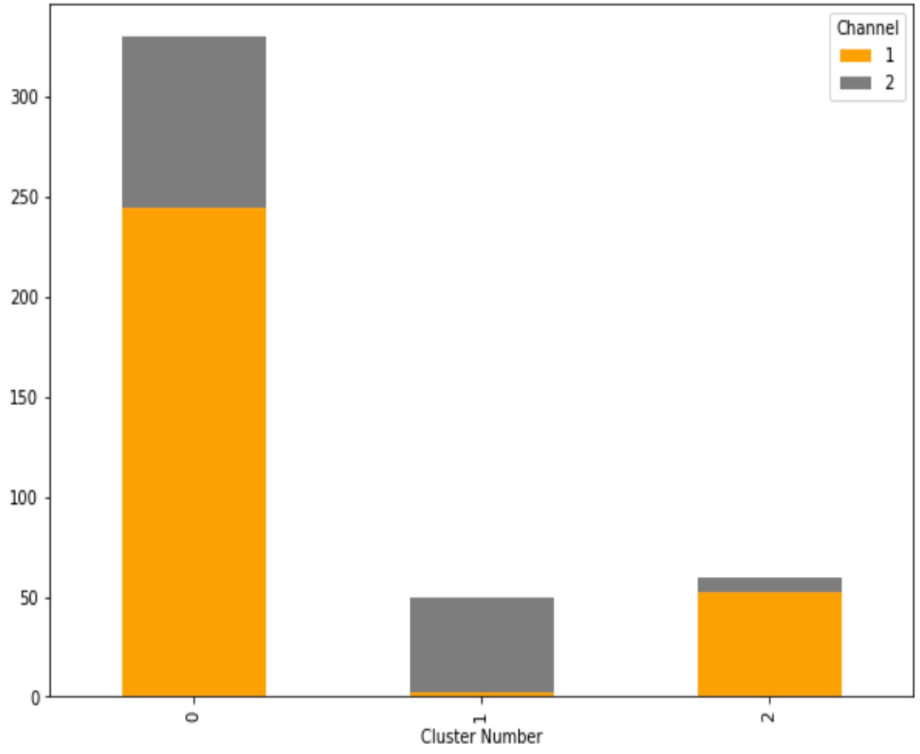
```
array([1, 1, 1, 1, 0, 1, 1, 1, 1, 2, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       0, 2, 0, 1, 1, 1, 2, 0, 1, 1, 1, 0, 1, 1, 0, 1, 2, 0, 0, 1, 1, 2,
       1, 2, 2, 2, 1, 2, 1, 1, 0, 1, 0, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 2,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 0,
       1, 0, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 2,
       1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 2, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 2, 2, 0, 1, 1, 2, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 2, 2, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1,
       1, 2, 1, 0, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 2, 1, 2, 1, 0, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 0, 1, 2, 1, 2,
       1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       2, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 2, 1, 1],
      dtype=int32)
```

Divided into 3 clusters:
0,1 and 2cluster

Channel Cluster break up

```
#channel cluster break up
pivot_df = df_kmeans.pivot_table(index='Cluster Number', columns='Channel', values='Milk', aggfunc = len)
print(pivot_df)
#channel cluster
colors = ["orange", "grey"]
pivot_df.plot.bar(stacked=True, color=colors, figsize=(10,7))
```

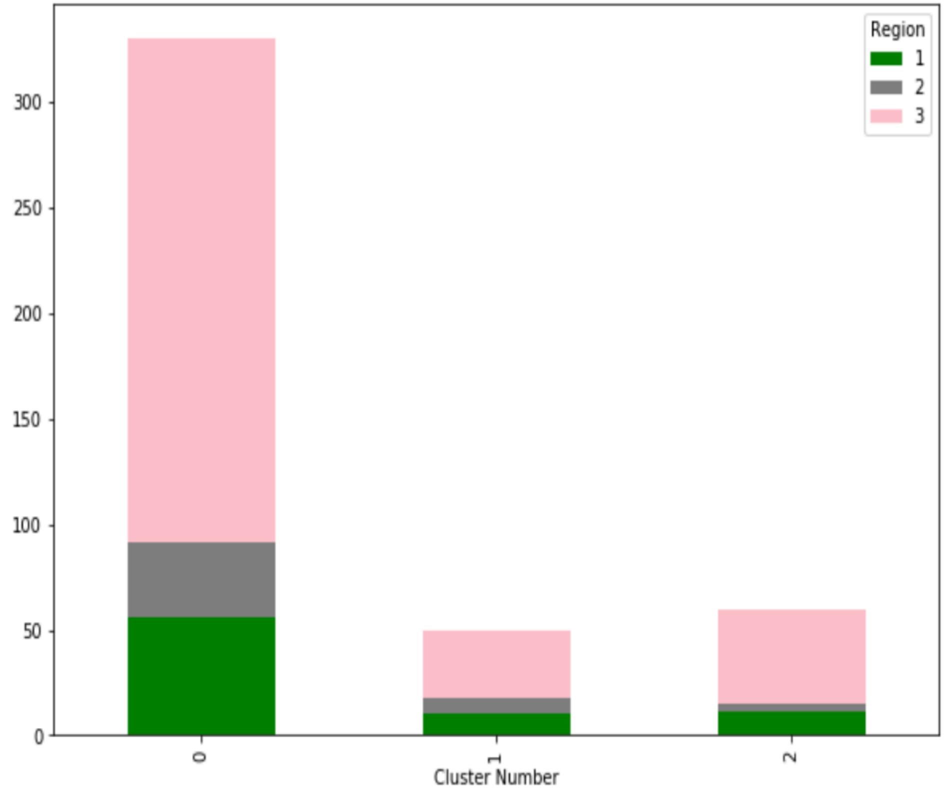
```
Channel           1   2
Cluster Number
0               244  86
1                 2  48
2                52   8
<matplotlib.axes._subplots.AxesSubplot at 0x7ff47a78e748>
```



Region Cluster break up

```
#region cluster break up
pivot_df = df_kmeans.pivot_table(index='Cluster Number', columns='Region', values='Milk', aggfunc = len)
print(pivot_df)
#region cluster
colors = ["green", "grey", "pink"]
pivot_df.plot.bar(stacked=True, color=colors, figsize=(10,7))
```

```
Region           1   2    3
Cluster Number
0               56  35  239
1               10   8   32
2               11   4   45
<matplotlib.axes._subplots.AxesSubplot at 0x7ff47a6749e8>
```



Product Cluster break up

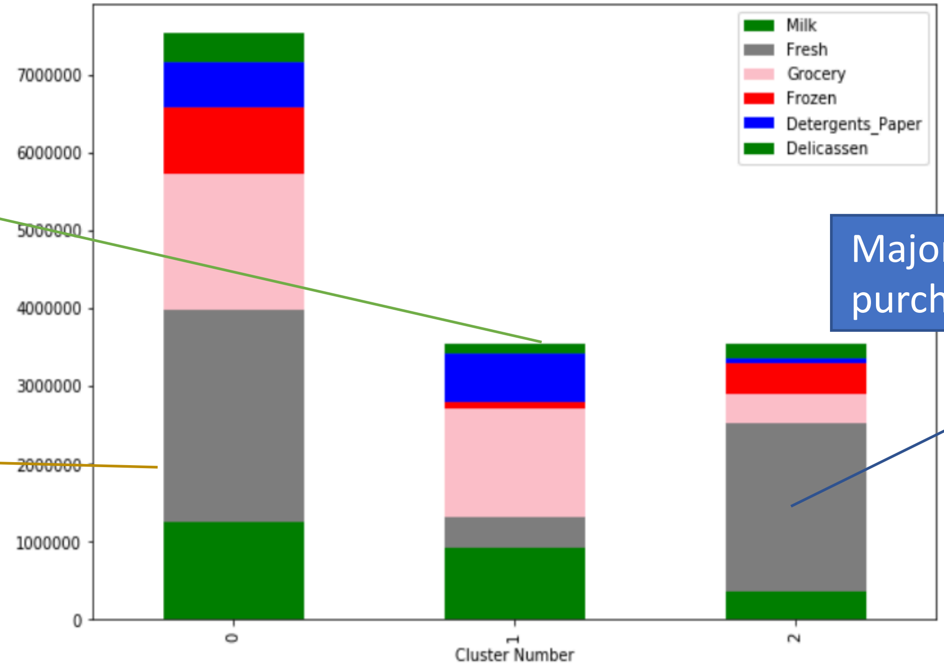
Grocery product category purchasing clusters

Majorly Fresh + Grocery product category purchasing clusters

```
#product cluster break up
pivot_df_1 = df_kmeans.pivot_table(index='Cluster Number', values='Milk', aggfunc = np.sum)
pivot_df_2 = df_kmeans.pivot_table(index='Cluster Number', values='Fresh', aggfunc = np.sum)
pivot_df_3 = df_kmeans.pivot_table(index='Cluster Number', values='Grocery', aggfunc = np.sum)
pivot_df_4 = df_kmeans.pivot_table(index='Cluster Number', values='Frozen', aggfunc = np.sum)
pivot_df_5 = df_kmeans.pivot_table(index='Cluster Number', values='Detergents_Paper', aggfunc = np.sum)
pivot_df_6 = df_kmeans.pivot_table(index='Cluster Number', values='Delicassen', aggfunc = np.sum)
pivot_df_1['Fresh'] = pivot_df_2['Fresh']
pivot_df_1['Grocery'] = pivot_df_3['Grocery']
pivot_df_1['Frozen'] = pivot_df_4['Frozen']
pivot_df_1['Detergents_Paper'] = pivot_df_5['Detergents_Paper']
pivot_df_1['Delicassen'] = pivot_df_6['Delicassen']
print(pivot_df_1)
#product cluster
colors = ["green", "grey", "pink","red","blue"]
pivot_df_1.plot.bar(stacked=True, color=colors, figsize=(10,7))
```

Cluster Number	Milk	Fresh	Grocery	Frozen	Detergents_Paper	Delicassen
0	1262119	2723645	1742550	848978	585109	375374
1	925571	400002	1378695	99834	620368	112601
2	362667	2156484	377317	402838	62380	182968

<matplotlib.axes._subplots.AxesSubplot at 0x7ff47a78e358>



Majorly Fresh product category purchasing clusters

PCA & K means

```python
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(trainData)
principalDf = pd.DataFrame(data = principalComponents, columns= ['principal component 1','principal component 2'] )
```

2 PCA Components

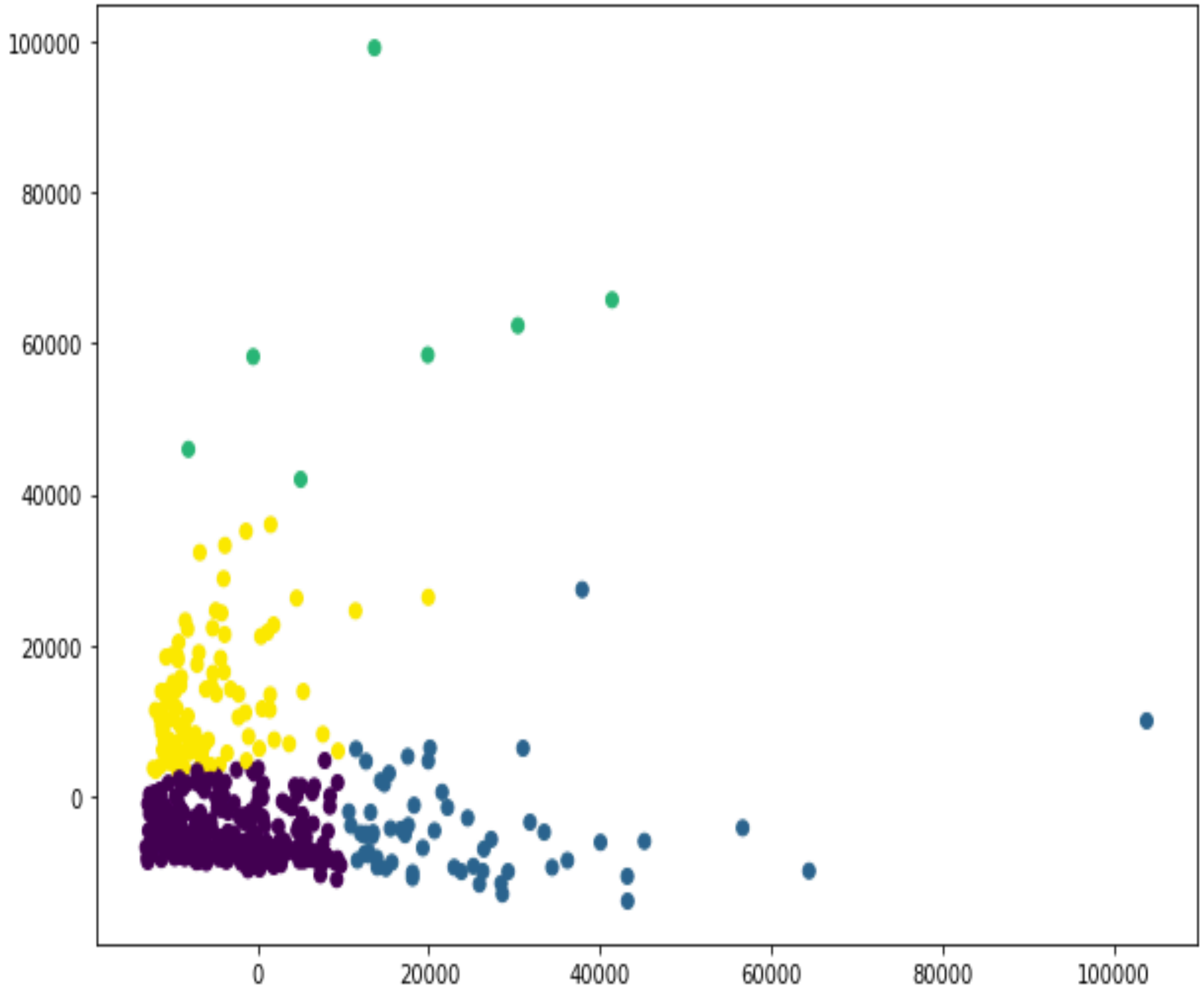
```python
# using default k means
kmeans_model_pca = KMeans(n_clusters= 4, init='k-means++', n_jobs = -1)
kmeans_model_pca.fit(principalDf)
print("Silhouette score for clustering:",   silhouette_score(principalDf, kmeans_model_pca.labels_))
print("Sum of squared distances for clustering:", kmeans_model_pca.inertia_)
```

```
Silhouette score for clustering: 0.4623571064398233б
Sum of squared distances for clustering: 43781540413.66852
```

```python
plt.figure(figsize=(10, 7))
plt.scatter(principalDf['principal component 1'], principalDf['principal component 2'], c=kmeans_model.labels_)
```

```
<matplotlib.collections.PathCollection at 0x7ff47aa38a20>
```

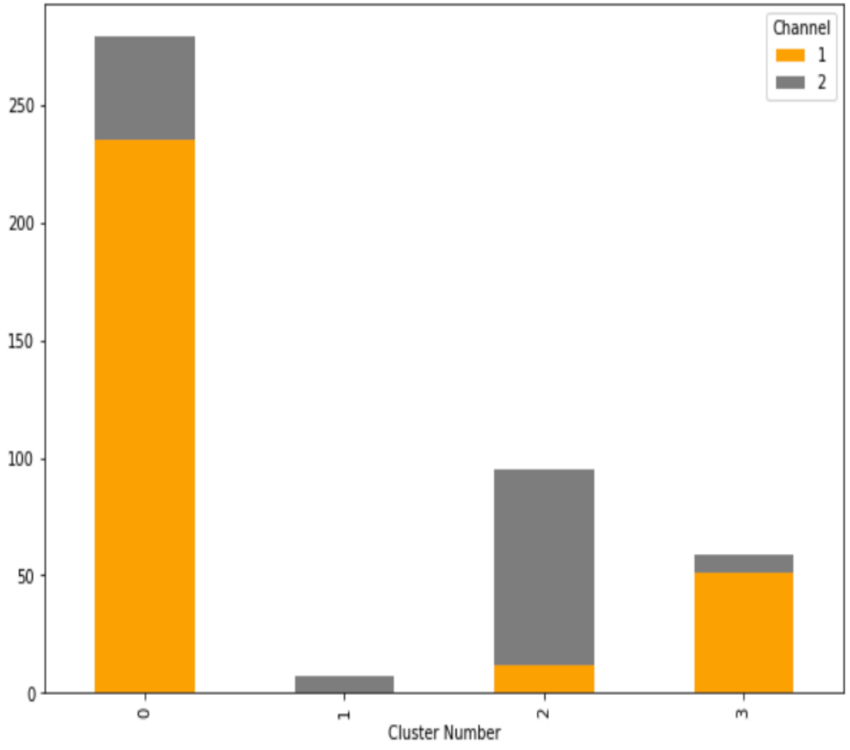
CLUSTERS AFTER PCA + K Means



Channel Cluster break up

```
#channel cluster break up
pivot_df = df_kmeans_pca.pivot_table(index='Cluster Number', columns='Channel', values='Milk', aggfunc = len)
print(pivot_df)
#channel cluster
colors = ["orange", "grey"]
pivot_df.plot.bar(stacked=True, color=colors, figsize=(10,7))
```

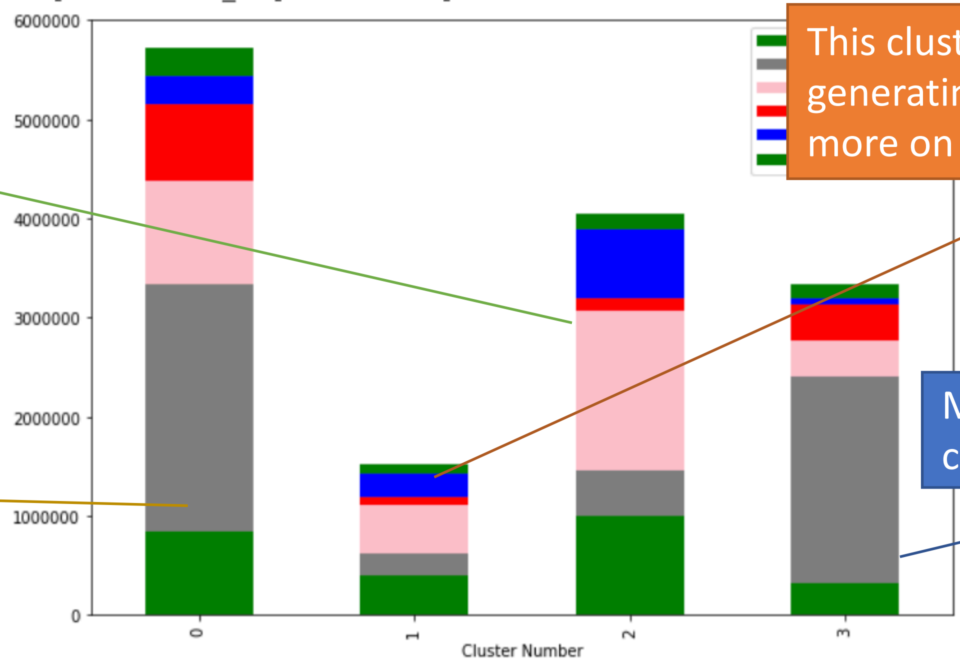
```
Channel             1     2
Cluster Number
0               235.0  44.0
1                 NaN   7.0
2                12.0  83.0
3                51.0   8.0
<matplotlib.axes._subplots.AxesSubplot at 0x7ff47a5b1f98>
```



Product Cluster break up

```
#region cluster break up
pivot_df_1 = df_kmeans_pca.pivot_table(index='Cluster Number', values='Milk', aggfunc = np.sum)
pivot_df_2 = df_kmeans_pca.pivot_table(index='Cluster Number', values='Fresh', aggfunc = np.sum)
pivot_df_3 = df_kmeans_pca.pivot_table(index='Cluster Number', values='Grocery', aggfunc = np.sum)
pivot_df_4 = df_kmeans_pca.pivot_table(index='Cluster Number', values='Frozen', aggfunc = np.sum)
pivot_df_5 = df_kmeans_pca.pivot_table(index='Cluster Number', values='Detergents_Paper', aggfunc = np.sum)
pivot_df_6 = df_kmeans_pca.pivot_table(index='Cluster Number', values='Delicassen', aggfunc = np.sum)
pivot_df_1['Fresh'] = pivot_df_2['Fresh']
pivot_df_1['Grocery'] = pivot_df_3['Grocery']
pivot_df_1['Frozen'] = pivot_df_4['Frozen']
pivot_df_1['Detergents_Paper'] = pivot_df_5['Detergents_Paper']
pivot_df_1['Delicassen'] = pivot_df_6['Delicassen']
print(pivot_df_1)
#region cluster
colors = ["green", "grey", "pink","red","blue"]
pivot_df_1.plot.bar(stacked=True, color=colors, figsize=(10,7))
```

```
                  Milk    Fresh  Grocery  Frozen  Detergents_Paper  Delicassen
Cluster Number
0               835570  2508140  1035970  777764            276829      287185
1               397566   218771   500692   69618            235588       92554
2               999876   456840  1606430  138946            693728      156834
3               317345  2096380   355470  365322             61712      134370
<matplotlib.axes._subplots.AxesSubplot at 0x7ff47a9e4cf8>
```



Grocery product category purchasing clusters

Majorly Fresh + Grocery product category purchasing clusters

This cluster is low revenue generating , so we can focus more on other clusters

Majorly Fresh product category purchasing clusters

Key learnings

- Different methods of doing clustering
- No correct solution to clustering
- Finding optimal value of k
- PCA can be used to do reduce high dimensional data