INTERACTIVE MACHINE LEARNING DASHBOARDS USING PRINCIPAL COMPONENT ANALYSIS IN PYTHON

Aim:

- In this case study, We will be analysing a customer transaction dataset in order to investigate and interpret customer behaviour of a certain supplier.
- We will be doing exploratory data analysis on our data, before employing a variety of dimensionality reduction techniques, ranging from introductory to more advanced.
- We will visualize our new representation, before clustering our customers based on their behaviour.
- Finally, we will visualize our clusters in an interactive way to analyse them and their differences more thoroughly.
- Four different dataset s have been used to understand this case study.

Tools used:

These are the concepts or tools we will use in this project:

- Principal Component Analysis (PCA)
- Kernel Principal Component Analysis (KPCA)
- K-Means Clustering
- Elbow Method

Outline:

- Introduction
- Exploratory Data Analysis
- Principal Component Analysis
- Kernel Principal Component Analysis

- K-Means Clustering with Elbow Method
- Interactive Cluster Analysis

Procedure:

We will use unsupervised methods to reduce the dimensionality of this data, and plot the resulting 2-D data, and investigate what our models are learning:

In [3]:
```
import pandas as pd
df = pd.read_csv(r'C:\Users\jai\Desktop\Interactive Machine Learning Dashboards using Principa
df.head()
```

Out[3]:

	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

In [6]:
```
features = [#'Channel',
 #'Region',
 'Fresh',
 'Milk',
 'Grocery',
 'Frozen',
 'Detergents_Paper',
 'Delicatessen']
df[features].describe()
```

Out[6]:

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
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

Performing Exploratory Data Analysis:

In [3]:
```
df = pd.read_csv(r'C:\Users\jai\Desktop\Interactive Machine Learning Dashboards usi
df.head()
```

Out[3]:

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

In [4]:
```
df.Region.value_counts()
```

Out[4]:
```
Other     316
Lisbon     77
Porto      47
Name: Region, dtype: int64
```

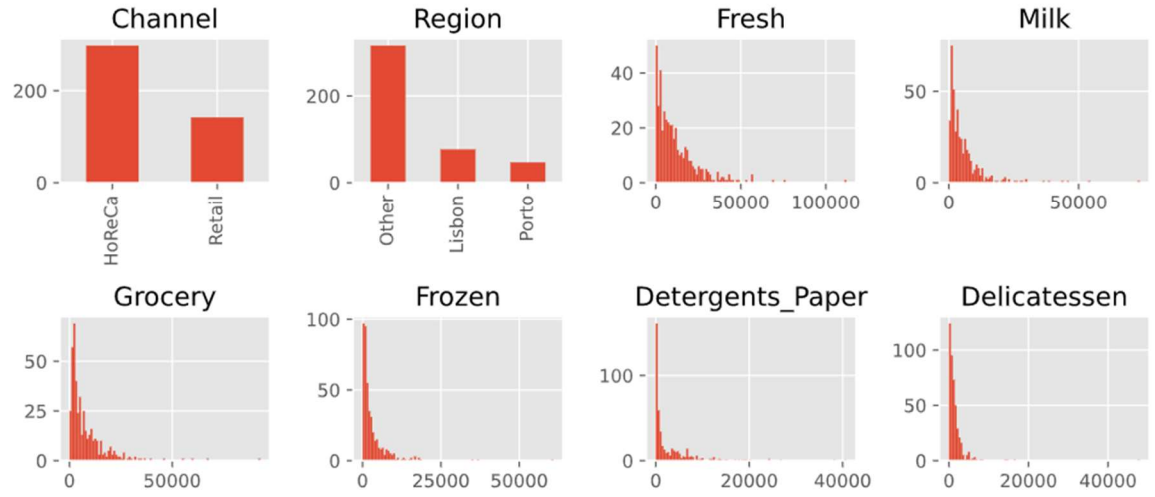
In [5]:
```
df.Channel.value_counts()
```

Out[5]:
```
HoReCa    298
Retail    142
Name: Channel, dtype: int64
```

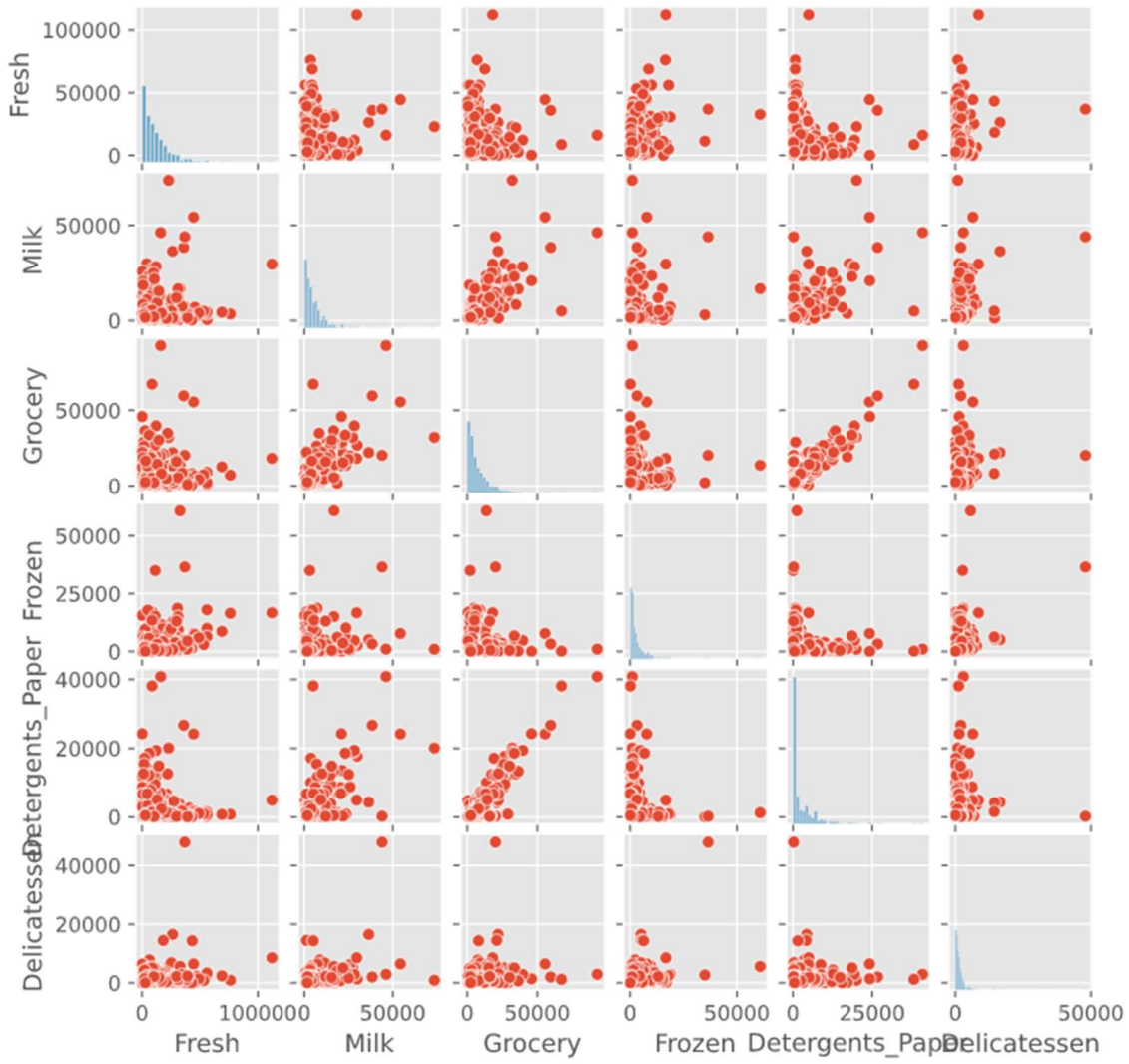
In [6]:
```
df.columns.tolist()
```

Out[6]:
```
['Channel',
 'Region',
 'Fresh',
 'Milk',
 'Grocery',
 'Frozen',
 'Detergents_Paper',
 'Delicatessen']
```

In [9]:
```
features = ['Channel',
 'Region',
 'Fresh',
 'Milk',
 'Grocery',
 'Frozen',
 'Detergents_Paper',
 'Delicatessen']
fig, axes = plt.subplots(2,4,figsize=(9,4))
for feature, ax in zip(features, axes.ravel()):
    if(feature=='Channel') | (feature=='Region'):
        df[feature].value_counts().plot.bar(ax=ax)
    else:
        ax.hist(df[feature],bins=100)
    ax.set_title(feature)
plt.tight_layout()
```



```
In [12]: pplot=sns.pairplot(df, vars = features[2:])
         pplot.fig.set_size_inches(7,7)
```



Performing Principal Component Analysis:

In [3]:
```
df = pd.read_csv(r'C:\Users\jai\Desktop\Interactive Machine Learning Dashboards using Principal C
```

In [4]:
```
features
```

Out[4]:
```
['Fresh', 'Milk', 'Grocery', 'Frozen', 'Detergents_Paper', 'Delicatessen']
```

In [5]:
```
data = df[features].to_numpy()
data.shape
```

Out[5]:
```
(440, 6)
```

In [6]:
```
scaler = StandardScaler()
data = scaler.fit_transform(data)
data[:, 0].std()
```

Out[6]:
```
1.0
```

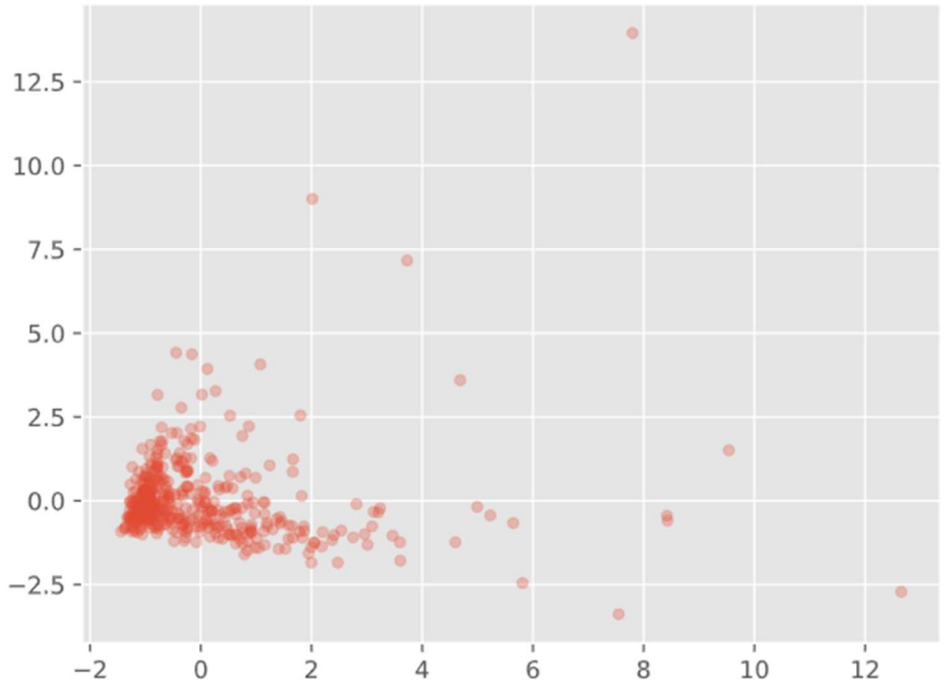
In [7]:
```
data
```

Out[7]:
```
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]])
```

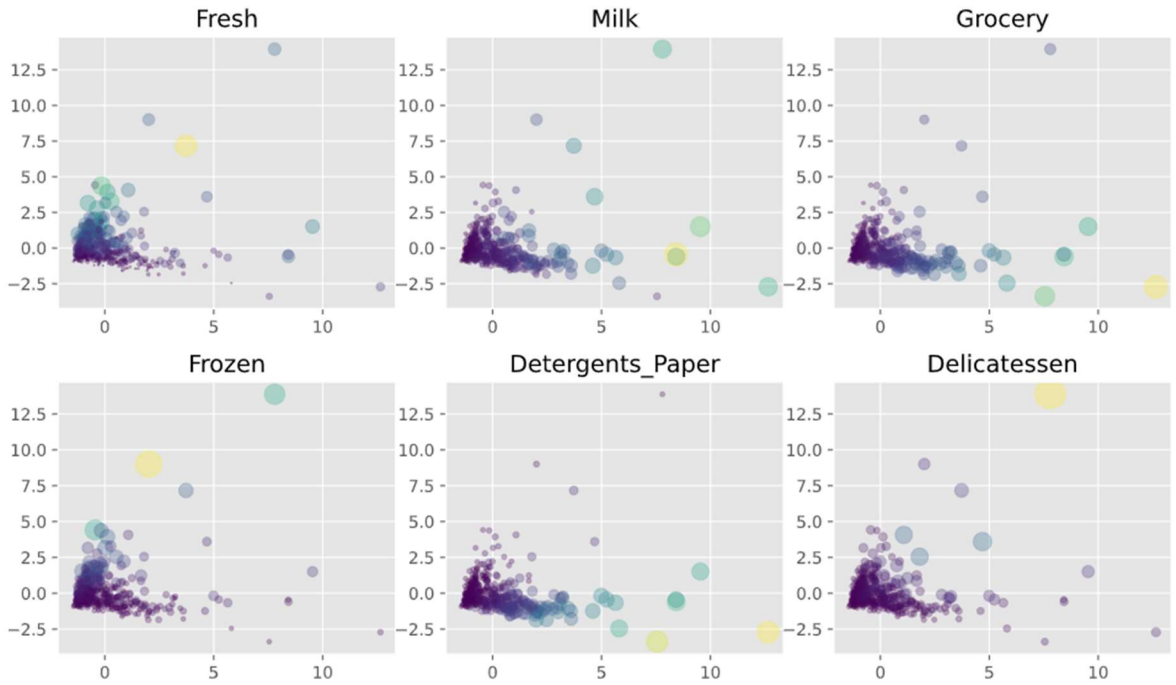
In [8]:
```
np.save(r'C:\Users\jai\Desktop\Interactive Machine Learning Dashboards using Principal Component
```

In [33]:
```
plt.scatter(res_pca[:,0],res_pca[:,1],s=20,alpha=0.3)
```

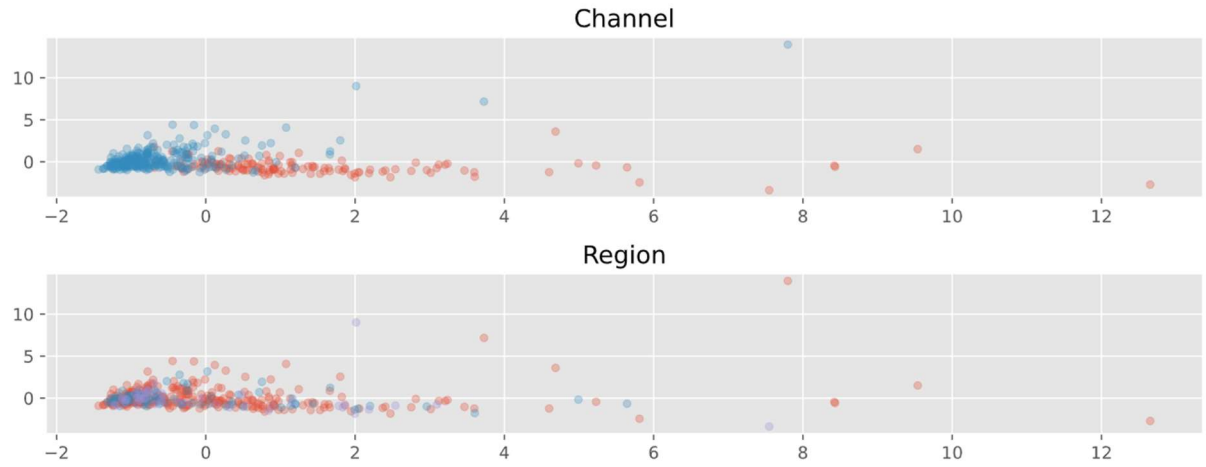
Out[33]:
```
<matplotlib.collections.PathCollection at 0x275a5fe1760>
```



In [21]:
```
fig, axes = plt.subplots(2, 3, figsize=(10, 6))

for feature, ax in zip(features, axes.ravel()):
    cols = 'viridis'
    sizes = 20+20*data[:, features.index(feature)]
    ax.scatter(res_pca[:, 0], res_pca[:, 1], s=sizes, alpha=0.3, c=df[feature], cmap=cols)
    ax.set_title(feature)
plt.tight_layout()
```



In [19]:
```
fig, axes = plt.subplots(2, 1, figsize=(10, 4))
for feature, ax in zip(['Channel', 'Region'], axes):
    cols = 'Pastel1'
    sizes = 20
    for unique_val in df[feature].unique():
        ax.scatter(res_pca[df[feature]==unique_val, 0], res_pca[df[feature]==unique_val, 1], s=20, alpha=0.3)
        ax.set_title(feature)
plt.tight_layout()
```

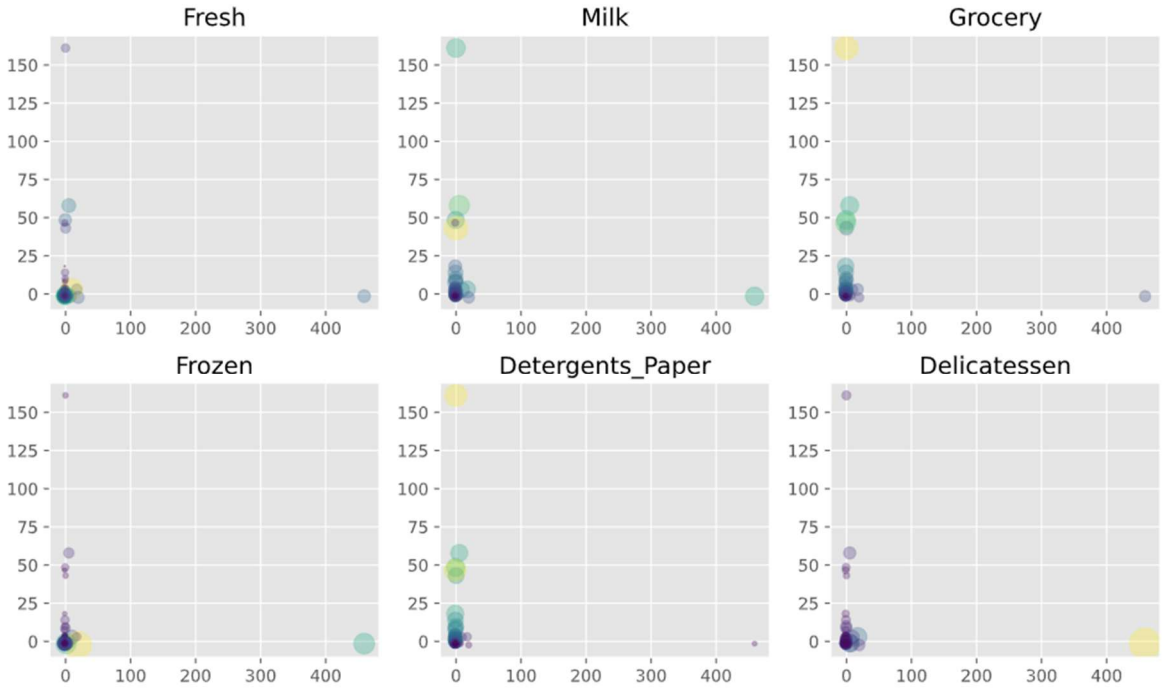


Performing Kernel Principal Component Analysis:

Polynomial Function:

In [3]:
```
kpca = KernelPCA(n_components=2, kernel='poly', degree=3)
res_kpca_poly = kpca.fit_transform(data)
```

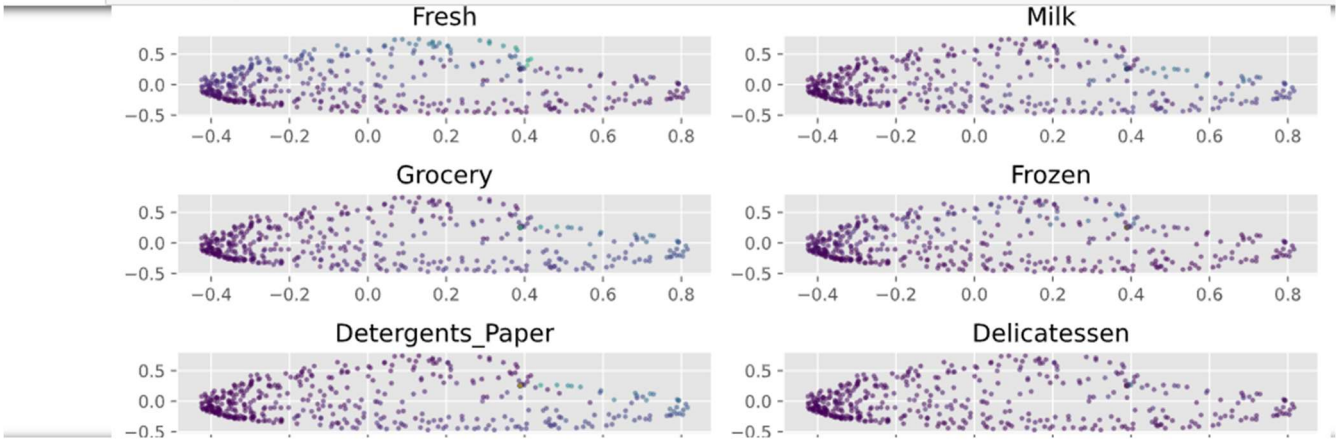
In [39]:
```
fig, axes = plt.subplots(2, 3, figsize=(10, 6))

for feature, ax in zip(features, axes.ravel()):
    cols = 'viridis'
    sizes = 20+20*data[:, features.index(feature)]
    ax.scatter(res_kpca_poly[:, 0], res_kpca_poly[:, 1], s=sizes, alpha=0.3, c=df[feature], cmap=cols)
    ax.set_title(feature)
plt.tight_layout()
```



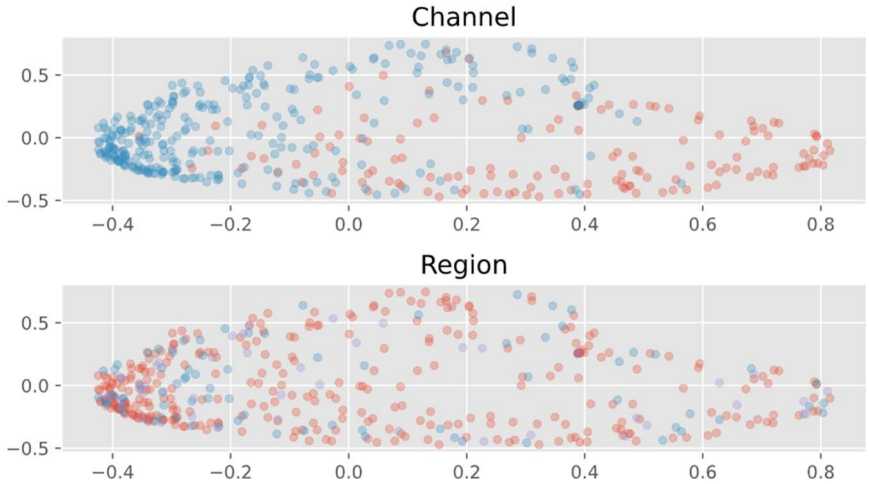
Radial Basis Function:

In [6]:
```
kpca = KernelPCA(n_components=2, kernel='rbf')
res_kpca_rbf = kpca.fit_transform(data)
```

In [37]:
```
fig, axes = plt.subplots(3, 2, figsize=(10, 4))

for feature, ax in zip(features, axes.ravel()):
    cols = 'viridis'
    sizes = 15+15*data[:, features.index(feature)]
    ax.scatter(res_kpca_rbf[:, 0], res_kpca_rbf[:, 1], s=5, alpha=0.5, c=df[feature], cmap=cols)
    ax.set_title(feature)
plt.tight_layout()
```

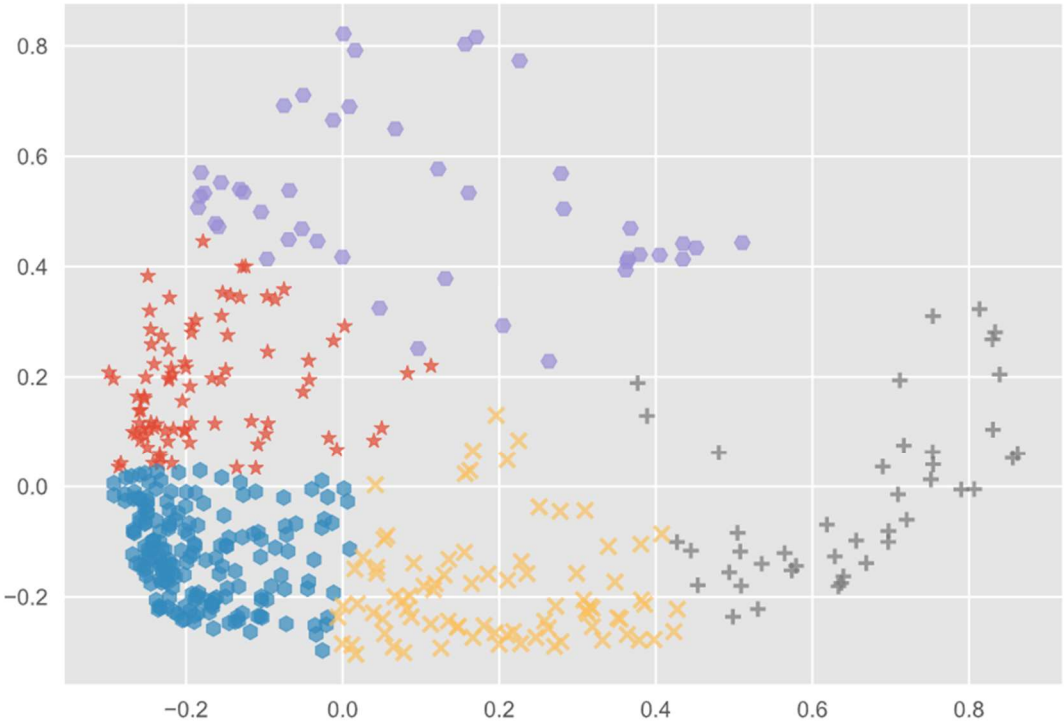


In [8]:
```
fig, axes = plt.subplots(2, 1, figsize=(7, 4))
for feature, ax in zip(['Channel', 'Region'], axes):
    cols = 'Pastel1'
    sizes = 20
    for unique_val in df[feature].unique():
        ax.scatter(res_kpca_rbf[df[feature]==unique_val, 0], res_kpca_rbf[df[feature]==unique_val, 1], s=20, alpha=0.3)
        ax.set_title(feature)
plt.tight_layout()
```

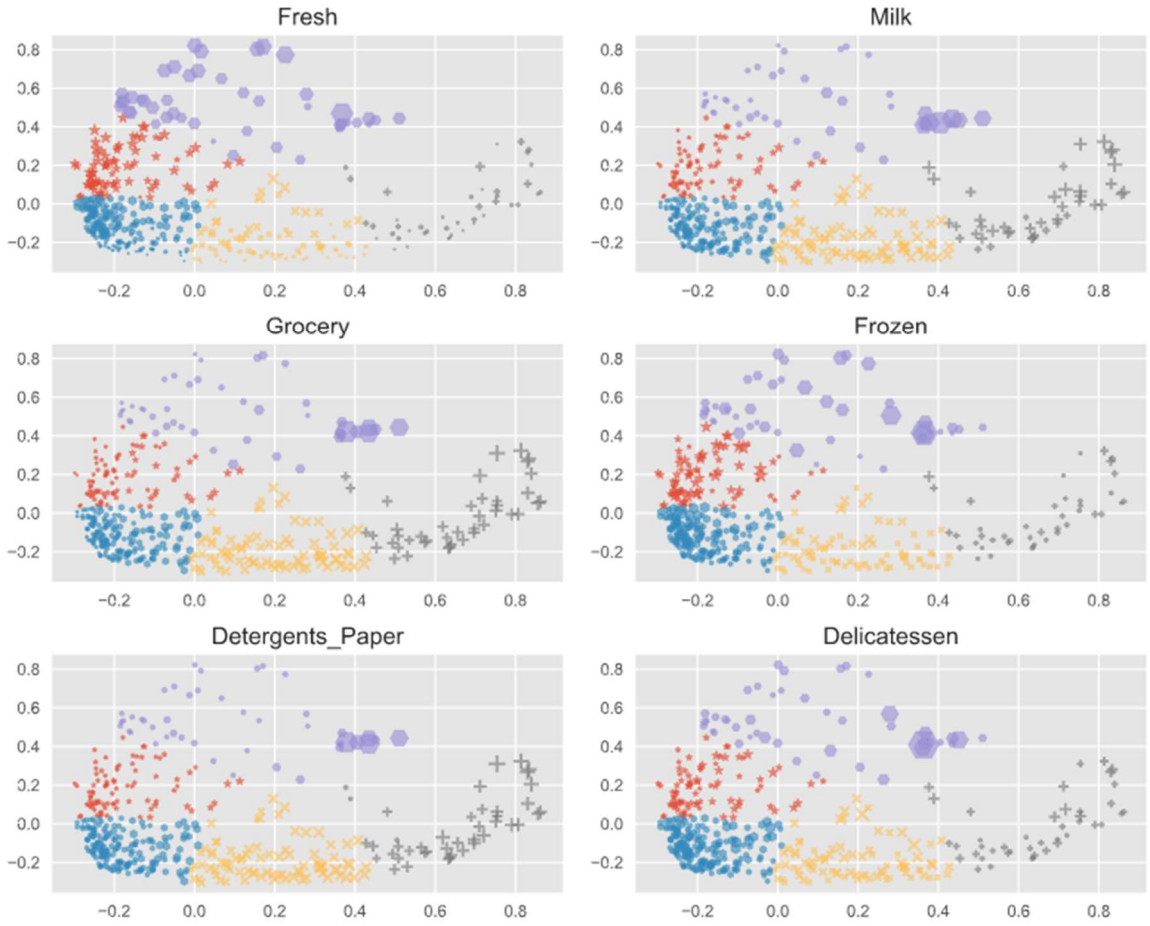


Performing K-means Clustering:

In [6]:
```
for cluster in np.unique(clusters):
    cluster_data = res_kpca[clusters==cluster]
    plt.scatter(cluster_data[:, 0], cluster_data[:, 1], alpha=0.7, marker=markers[cluster])
```

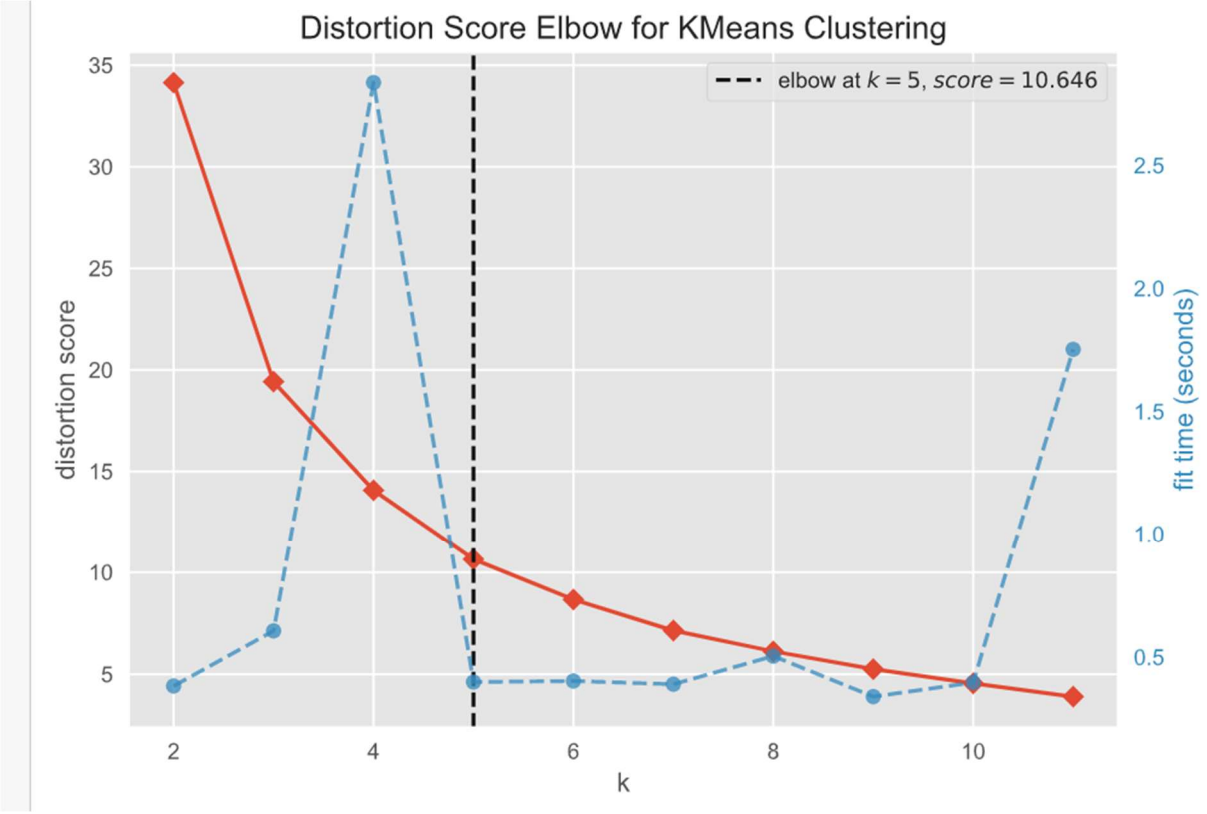


In [10]:
```
fig, axes = plt.subplots(3, 2, figsize=(10, 8))

for feature, ax in zip(features, axes.ravel()):
    cols = 'viridis'
    for cluster in np.unique(clusters):
        sizes = 20+20*data[:, features.index(feature)][clusters==cluster]
        cluster_data = res_kpca[clusters==cluster]
        ax.scatter(cluster_data[:, 0], cluster_data[:, 1], s=sizes, alpha=0.6, cmap=cols, marker=markers[cluster], la
    ax.set_title(feature)
plt.tight_layout()
```



Elbow Method:

In [9]:
```
clusterer = KMeans()
visualizer = KElbowVisualizer(clusterer, k=(2, 12), metric='distortion')

visualizer.fit(res_kpca)
visualizer.show()
```



Performing Interactive Cluster Analysis:

Cluster Distribution:

In [7]:
```
df.groupby(['cluster_kpca_rbf','Channel','Region'])[features].mean()
```

Out[7]:

cluster_kpca_rbf	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	HoReCa	Lisbon	7186.735294	2224.764706	2648.911765	2511.382353	805.352941	644.676471
		Other	6617.739496	2387.983193	2657.747899	1759.924370	518.823529	932.697479
		Porto	7411.941176	1716.823529	4019.000000	2383.705882	446.941176	852.647059
	Retail	Lisbon	2790.000000	2527.000000	5265.000000	5612.000000	788.000000	1360.000000
		Other	13953.500000	4443.333333	6634.750000	1018.166667	2582.333333	1085.916667
		Porto	10708.666667	3779.666667	6193.333333	898.666667	2860.666667	930.000000
1	HoReCa	Lisbon	4422.571429	9945.428571	8472.428571	1518.714286	2900.714286	1515.285714
		Other	3067.066667	6702.533333	9402.733333	1576.866667	3366.200000	1086.466667
	Retail	Lisbon	3341.571429	7110.714286	13179.571429	1090.142857	5953.000000	1928.285714
		Other	5984.403846	7408.000000	11534.000000	1271.326923	5021.230769	1399.461538
		Porto	2993.500000	8762.333333	11137.333333	692.166667	5765.833333	1018.500000
2	HoReCa	Lisbon	5909.000000	23527.000000	13699.000000	10155.000000	830.000000	3636.000000
		Other	10683.000000	21858.000000	15400.000000	3635.000000	282.000000	5120.000000
	Retail	Lisbon	4445.625000	16237.500000	27518.250000	2898.875000	12759.125000	1893.375000
		Other	6041.962963	14973.074074	24011.000000	1456.185185	10830.888889	1929.259259
		Porto	5135.571429	13998.142857	20566.428571	1338.142857	10581.857143	1114.285714
3	HoReCa	Lisbon	42521.600000	6157.000000	5525.200000	7690.600000	714.400000	3224.400000
		Other	41578.625000	7118.583333	7202.416667	10997.833333	1064.083333	5250.750000
		Porto	32717.000000	16784.000000	13626.000000	60869.000000	1272.000000	5609.000000
	Retail	Other	31176.000000	30966.777778	33162.444444	2589.000000	14175.444444	4563.777778
		Porto	8565.000000	4980.000000	67298.000000	131.000000	38102.000000	1215.000000
4	HoReCa	Lisbon	22284.083333	2397.500000	3903.916667	3323.833333	332.666667	1529.000000
		Other	20888.115385	3045.019231	3356.346154	5210.365385	537.250000	1191.000000
		Porto	16749.500000	1854.900000	4112.500000	5946.900000	464.600000	1086.100000
	Retail	Lisbon	15927.000000	5955.000000	7413.500000	5040.000000	1761.500000	1845.000000
		Other	21992.000000	6299.200000	9801.200000	3588.200000	2463.000000	2557.000000
		Porto	21952.500000	3872.500000	6766.500000	6462.000000	2221.500000	2812.500000

In [16]:
```
fig = go.Figure()

for cluster in np.unique(clusters):

    radii = df_normalized.loc[df_normalized.cluster_kpca_rbf==cluster, features].mean().tolist()
    thetas = features

    actual_values = df.loc[df.cluster_kpca_rbf==cluster, features].mean().tolist()
    cluster_size = len(df[df.cluster_kpca_rbf==cluster])
    print(cluster_size)
    fig.add_trace(
        go.Scatterpolar(
            r=radii + radii[:1],
            theta=thetas + thetas[:1],
            mode='lines',
            name=f'Cluster {cluster}',
            text = [f'Mean value: {x}' for x in actual_values + actual_values[:1]],
            line=dict(width=3),
            opacity=np.max([cluster_size/biggest_cluster, 0.6])
        )
    )

fig.update_layout(
    title='Cluster Analysis',
    showlegend=True,
    template='plotly_dark',
    width=800,
    autosize=False
)

fig.show()
```

```
186
87
44
40
83
```

Conclusion:

So, this is how Interactive Machine Learning Dashboards using Principal Component Analysis and other clustering analysis tools is performed in python.