clustering_secondary

October 22, 2024

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from minisom import MiniSom
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
```

1 Clustering Algorithms for Secondary Dataset

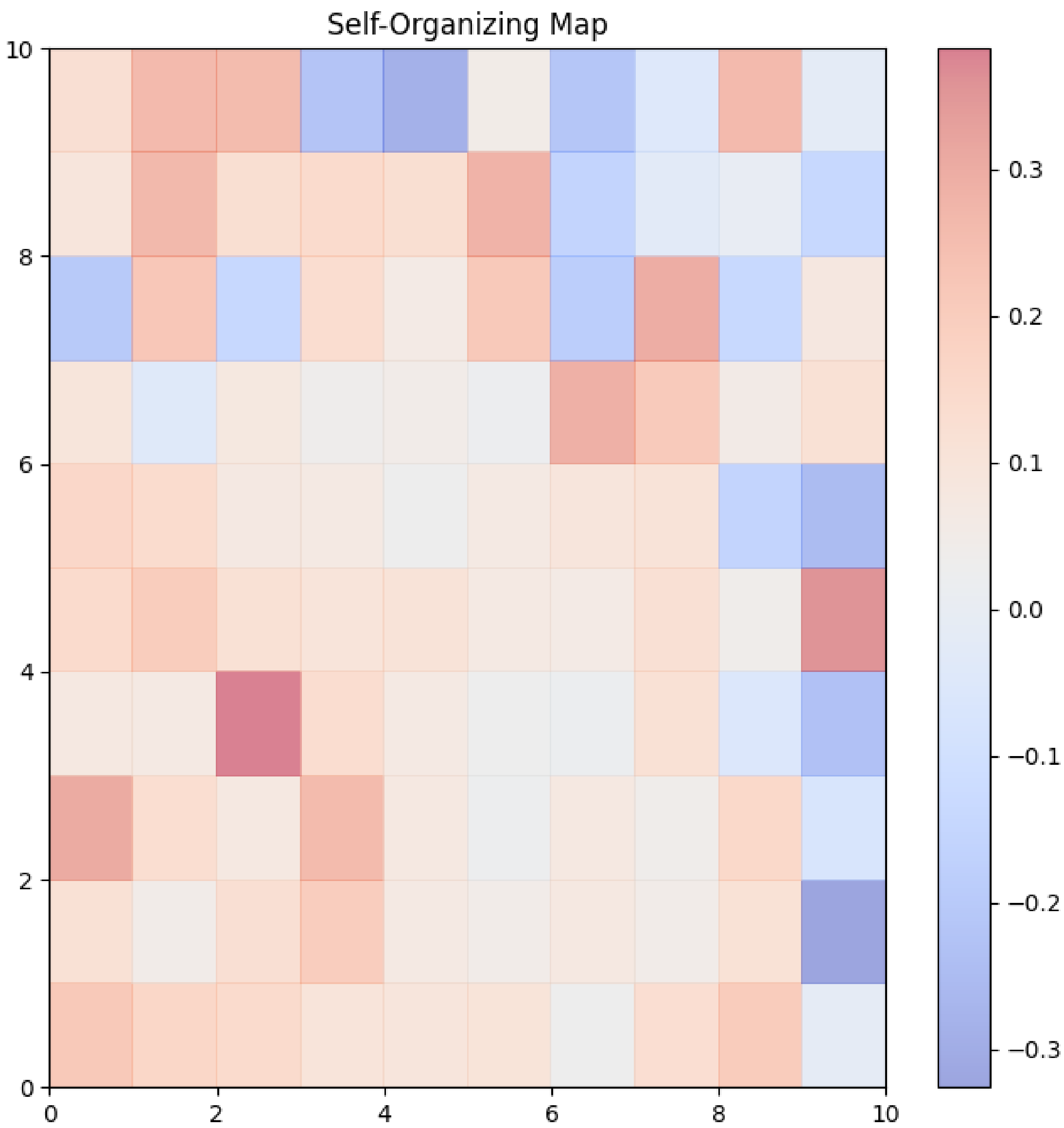
[2]:
```python
dataset = 'secondary'
data = pd.read_csv(f'../dataset/mapped_{dataset}.csv', sep=',')
data = data.drop(columns=['uuid'])
variable_names = data.columns.tolist()

# need to normalize data from SOMs
scaler = MinMaxScaler()
norm_data = pd.DataFrame(scaler.fit_transform(data), columns=data.columns)

print(norm_data)
```

```
           price    artist  cardFinish  colorIdentity    colors  edhrecRank  \
0       0.004237  0.581545         1.0       0.935484  0.935484    0.374964
1       0.033898  0.040057         1.0       0.806452  0.806452    0.239155
2       0.004237  0.410587         1.0       0.548387  0.548387    0.893443
3       0.004237  0.241059         1.0       0.000000  0.000000    0.834404
4       0.008475  0.791130         1.0       0.516129  0.516129    0.047660
…            …         …         …              …       …           …
228632  0.093220  0.348355         0.5       0.548387  0.548387    0.004531
228633  0.739407  0.844778         1.0       0.935484  0.935484    0.012343
228634  0.209746  0.426323         1.0       0.548387  0.548387    0.021440
228635  0.843220  0.426323         1.0       0.548387  0.548387    0.021440
228636  0.669492  0.426323         1.0       0.548387  0.548387    0.021440

        edhrecSaltiness  gameAvailability  isReprint  language  …  \
0              0.144295               0.0        1.0  0.222222  …
```

```
1               0.164430              0.0        1.0  0.222222  …
2               0.080537              0.0        1.0  0.222222  …
3               0.107383              0.0        1.0  0.222222  …
4               0.077181              0.0        1.0  0.222222  …
…                  …              …       …       …  …
228632          0.036913              1.0        1.0  0.444444  …
228633          0.526846              1.0        1.0  0.222222  …
228634          0.154362              1.0        1.0  0.555556  …
228635          0.154362              1.0        1.0  0.555556  …
228636          0.154362              1.0        1.0  0.555556  …

        originalType  power  priceProvider  providerListing  rarity   setCode  \
0           0.850566   1.00       0.000000              0.0     0.6  0.768317
1           0.850566   1.00       0.000000              0.0     0.6  0.768317
2           0.850566   0.65       0.000000              0.0     0.6  0.768317
3           0.850566   0.15       0.000000              0.0     0.6  0.768317
4           0.850566   1.00       0.000000              0.0     1.0  0.768317
…                …    …            …              …     …       …
228632      0.850566   0.15       0.333333              1.0     0.6  0.520792
228633      0.850566   0.85       0.333333              1.0     0.6  0.815842
228634      0.850566   1.00       0.333333              1.0     0.6  0.873267
228635      0.850566   1.00       0.333333              0.0     0.6  0.873267
228636      0.850566   1.00       0.666667              0.0     0.6  0.873267

        supertypes  toughness      type     types
0              0.5   1.000000  0.998076  1.000000
1              0.5   1.000000  0.666186  0.538462
2              0.5   0.666667  0.178932  0.230769
3              0.5   0.190476  0.605099  0.230769
4              0.5   1.000000  0.000000  0.000000
…             …        …       …       …
228632         0.5   0.190476  0.210678  0.230769
228633         0.5   0.904762  0.375661  0.230769
228634         0.5   1.000000  0.636845  0.307692
228635         0.5   1.000000  0.636845  0.307692
228636         0.5   1.000000  0.636845  0.307692

[228637 rows x 25 columns]
```

[3]:
```python
som_size = 10
som = MiniSom(som_size, som_size, norm_data.shape[1], sigma=1.0,␣
 ↪learning_rate=0.5)

som.train(norm_data.values, num_iteration=1000)

plt.figure(figsize=(8, 8))
plt.pcolor(som.get_weights()[:, :, 0], cmap='coolwarm', alpha=0.5)
```

```
plt.colorbar()
plt.title('Self-Organizing Map')
plt.show()
```



[4]:
```
win_map = som.win_map(norm_data.values)
cluster_characteristics = {}
```

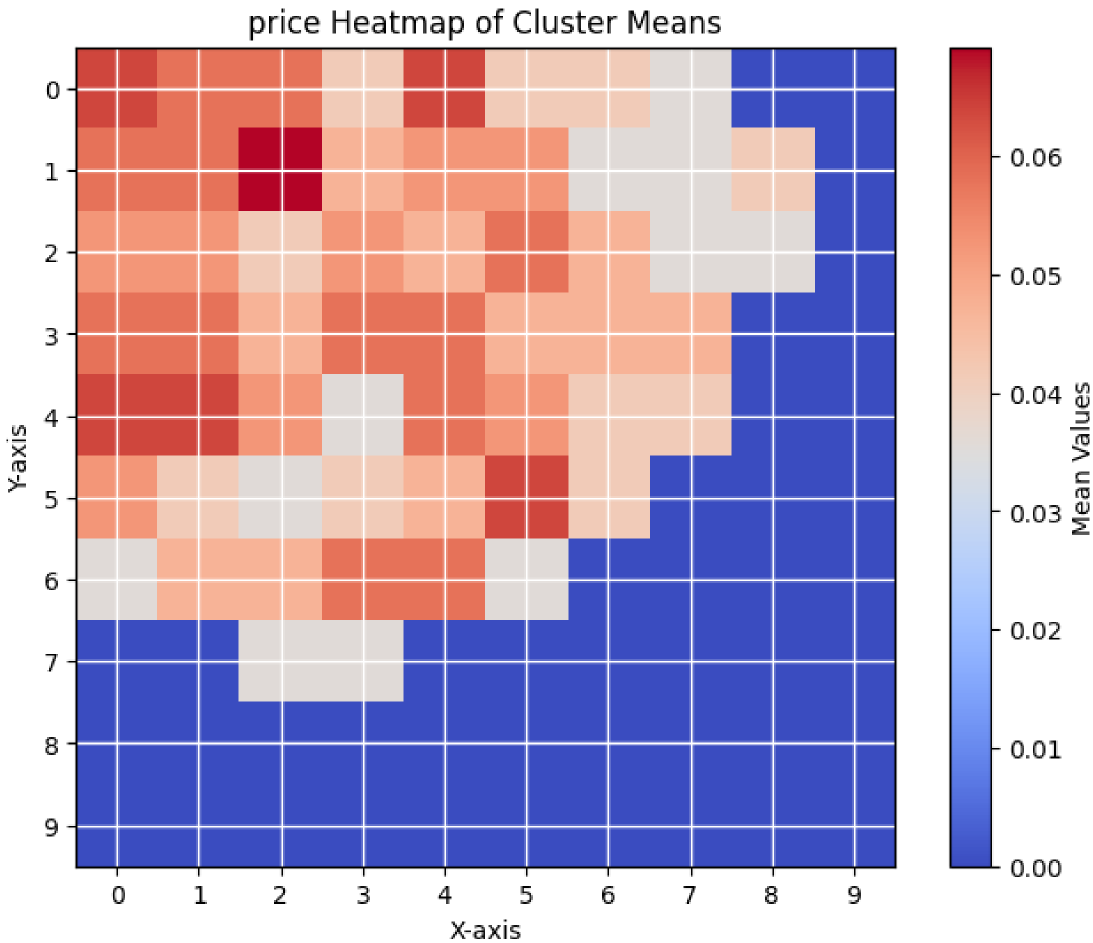
[5]:
```
def plt_som(win_map,col):

    cluster_characteristics = {}
```

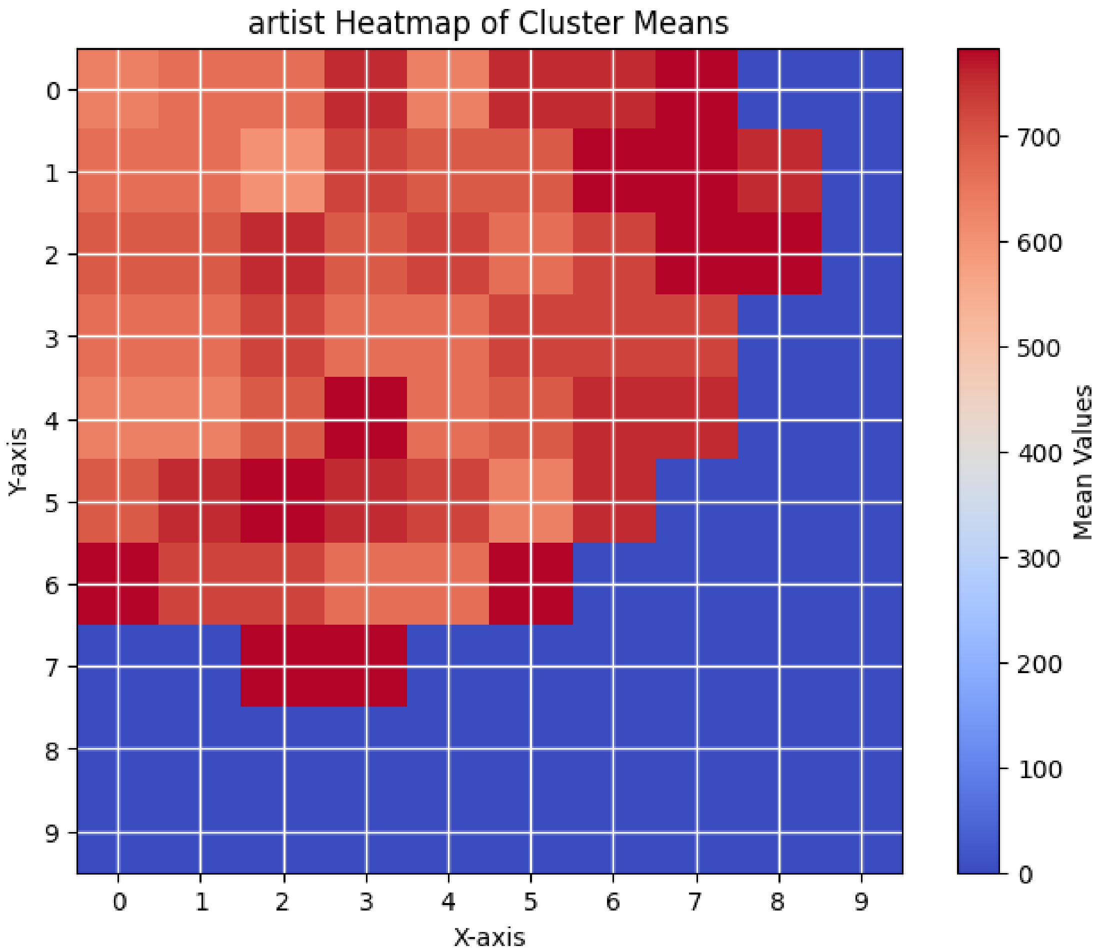
```
    # Loop through the win map to calculate mean prices for each cluster
    for (x, y), indices in win_map.items():
        for i in indices:
            values = data.iloc[i][col].values

        # Calculate mean price for the cluster
        mean_col = np.mean(values) if len(values) > 0 else 0
        median_col = np.median(values) if len(values) > 0 else 0
        std_col = np.std(values) if len(values) > 0 else 0

        # Store in the dictionary
        cluster_characteristics[(x, y)] = {f'mean_{col}': mean_col,␣
↪f'median_{col}': median_col, f'std_{col}': std_col}

    mean_values = np.zeros((10, 10))

    for (x, y), values in cluster_characteristics.items():
        mean_values[x, y] = values[f'mean_{col}']

    # Plotting the heatmap
    plt.figure(figsize=(8, 6))
    plt.imshow(mean_values, cmap='coolwarm', interpolation='nearest')
    plt.colorbar(label='Mean Values')
    plt.title(f'{col} Heatmap of Cluster Means')
    plt.xlabel('X-axis')
    plt.ylabel('Y-axis')

    # Optionally, add grid lines and labels
    plt.xticks(ticks=np.arange(10), labels=np.arange(10))
    plt.yticks(ticks=np.arange(10), labels=np.arange(10))
    plt.grid(color='white', linestyle='-', linewidth=1)

    plt.show()
```

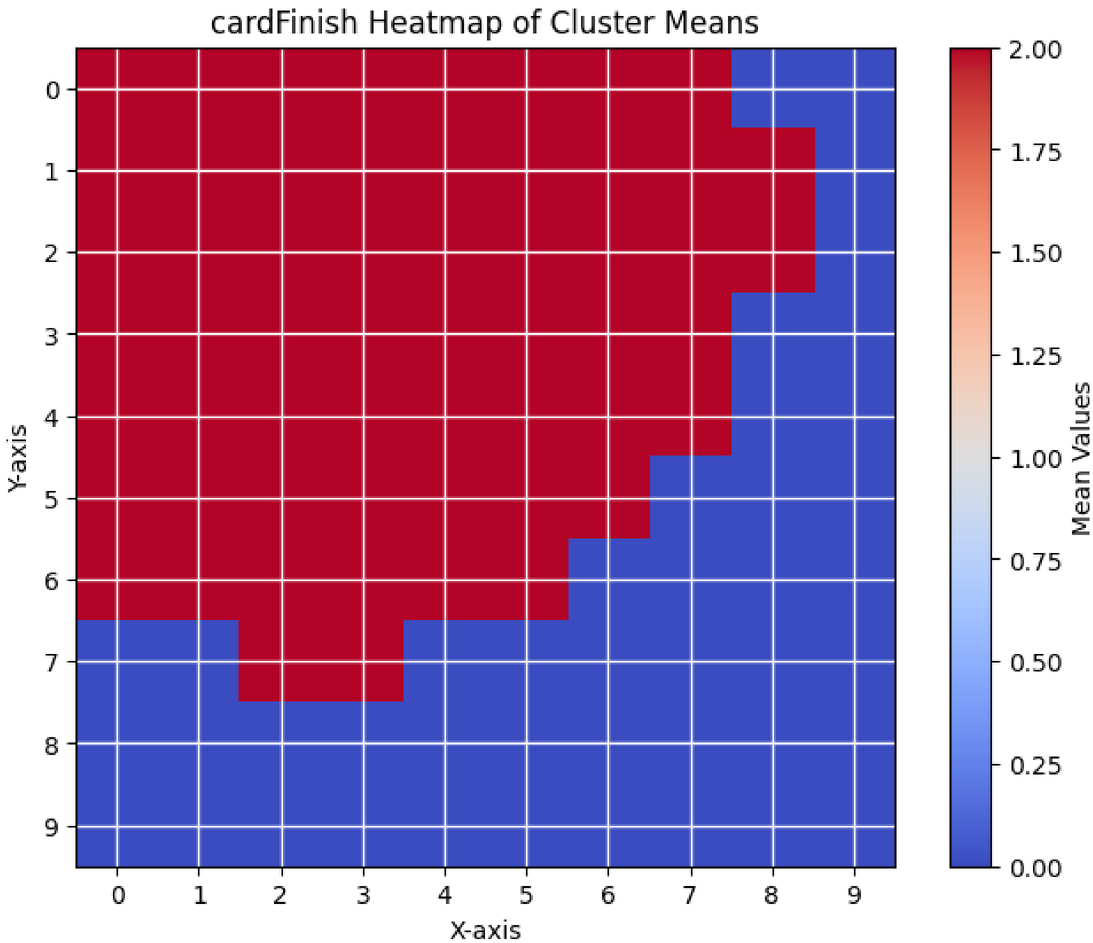
[6]:
```
print(len(data.columns))
```

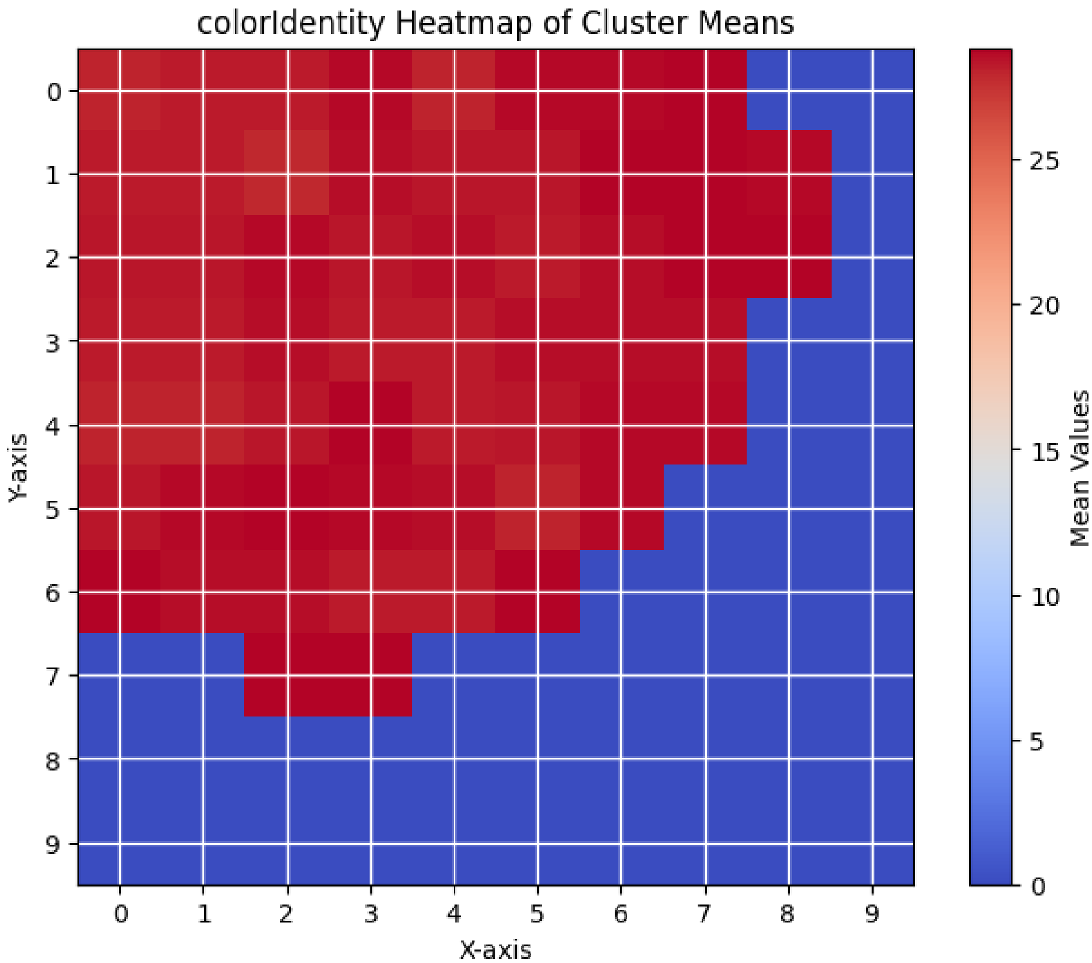
25

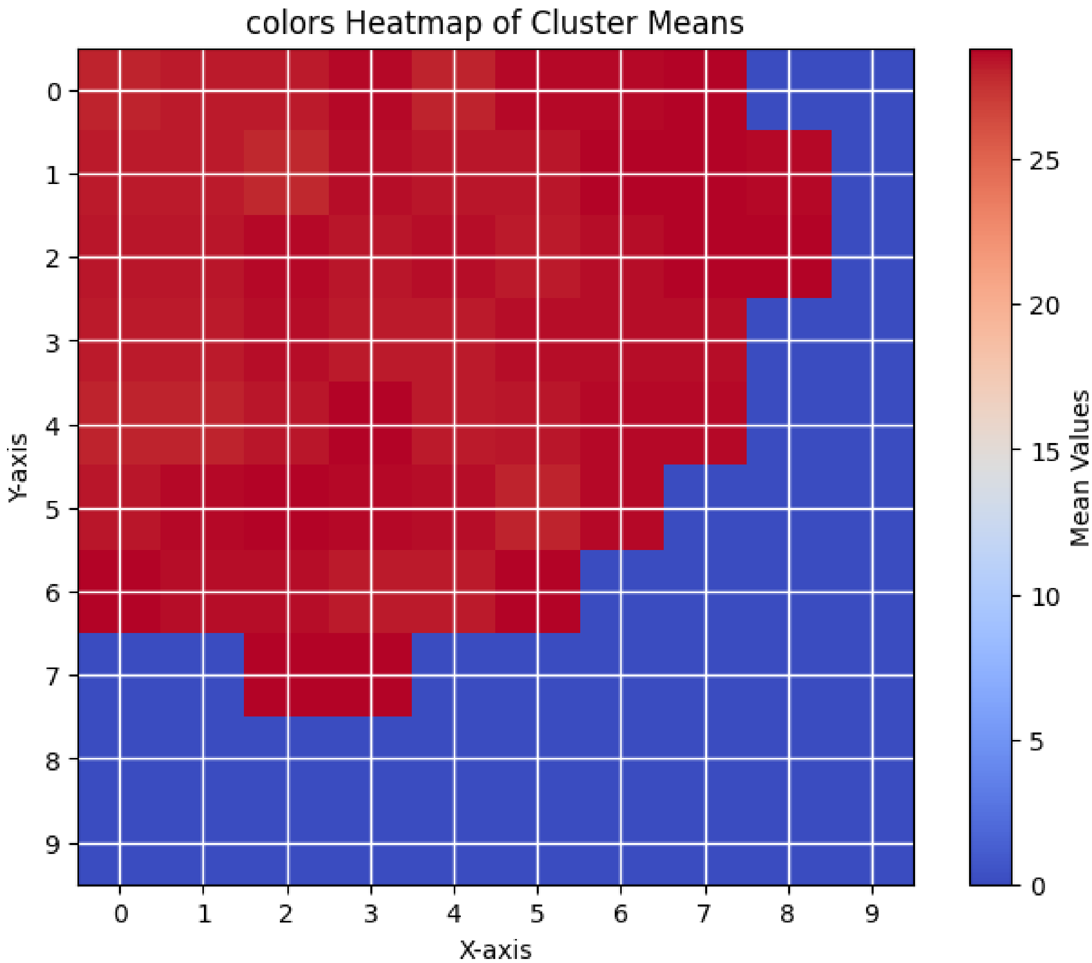
[7]:
```
for i in data.columns:
    plt_som(win_map, i)
```

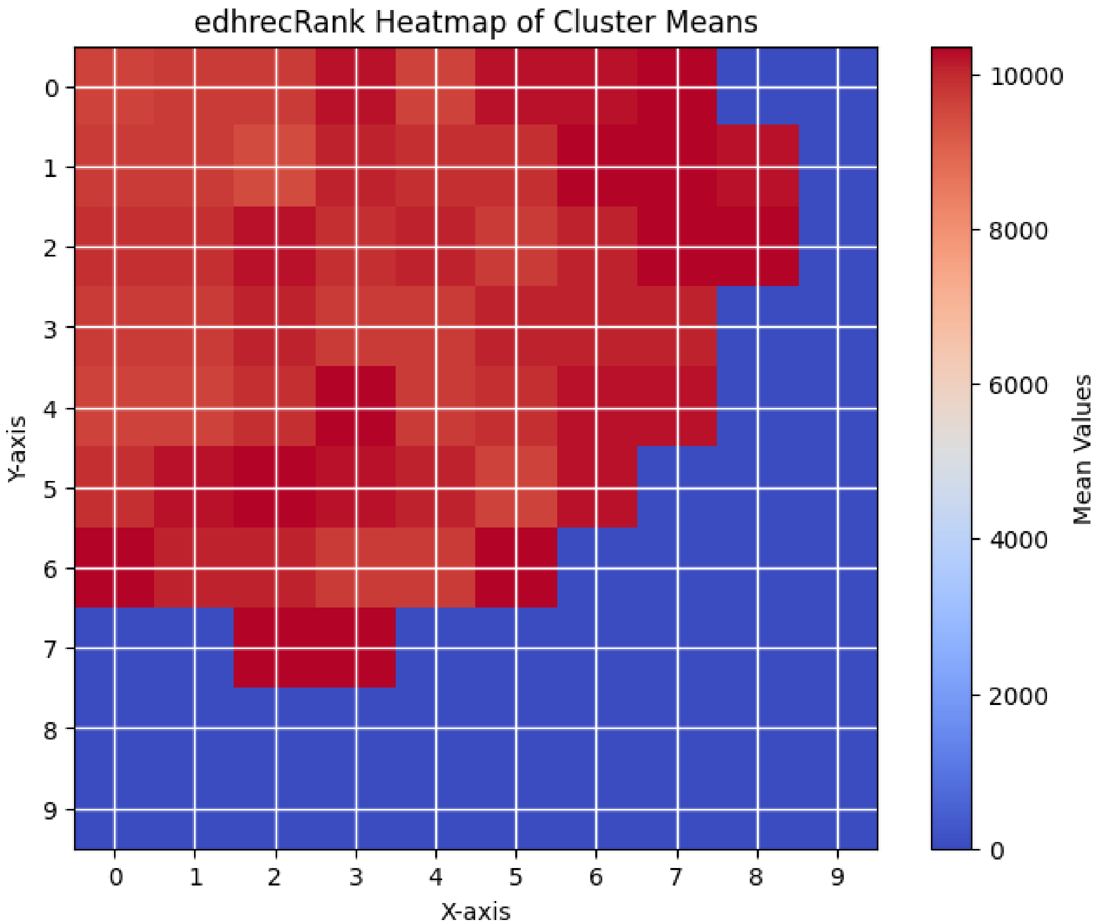

price Heatmap of Cluster Means
Y-axis
X-axis
Mean Values
0
1
2
3
4
5
6
7
8
9
0.00
0.01
0.02
0.03
0.04
0.05
0.06

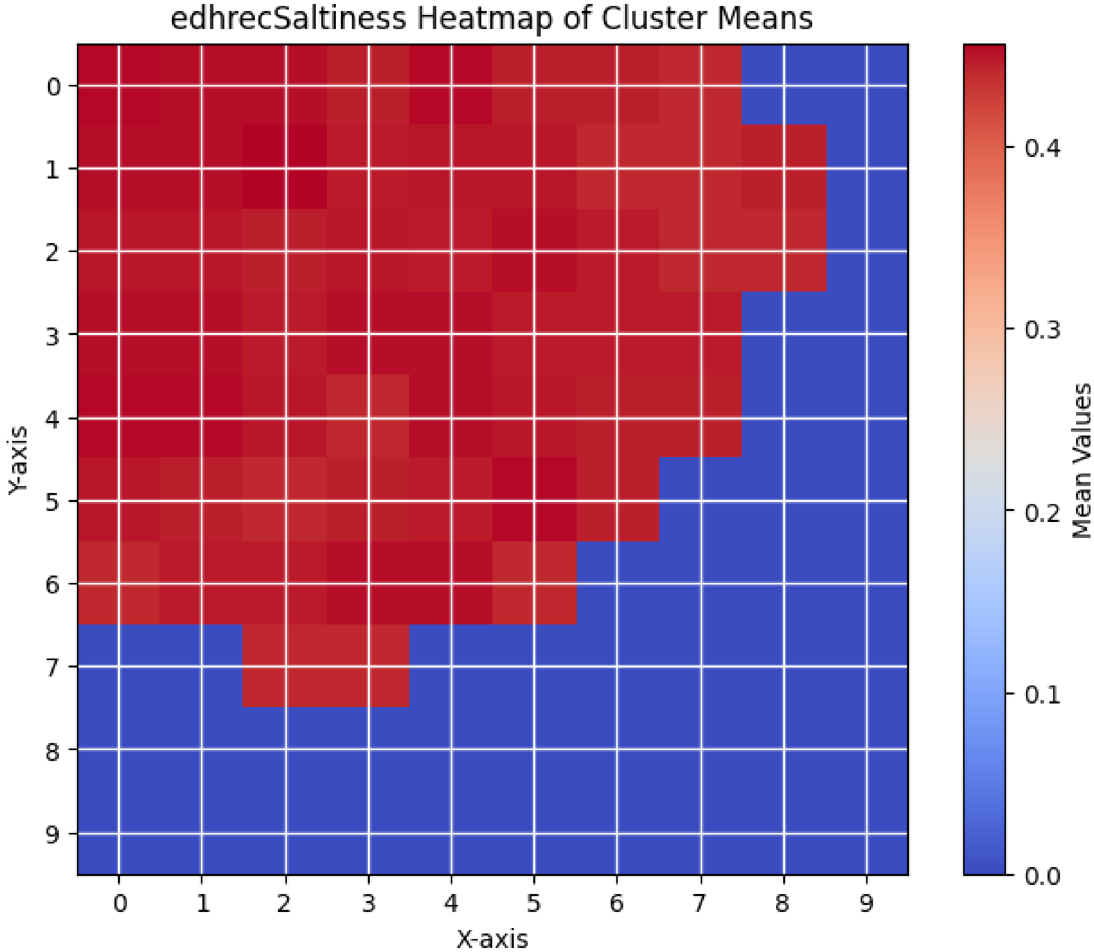

artist Heatmap of Cluster Means
Y-axis
0
1
2
3
4
5
6
7
8
9
X-axis
0
1
2
3
4
5
6
7
8
9
Mean Values
700
600
500
400
300
200
100
0


cardFinish Heatmap of Cluster Means
Y-axis
X-axis
Mean Values
0
1
2
3
4
5
6
7
8
9
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

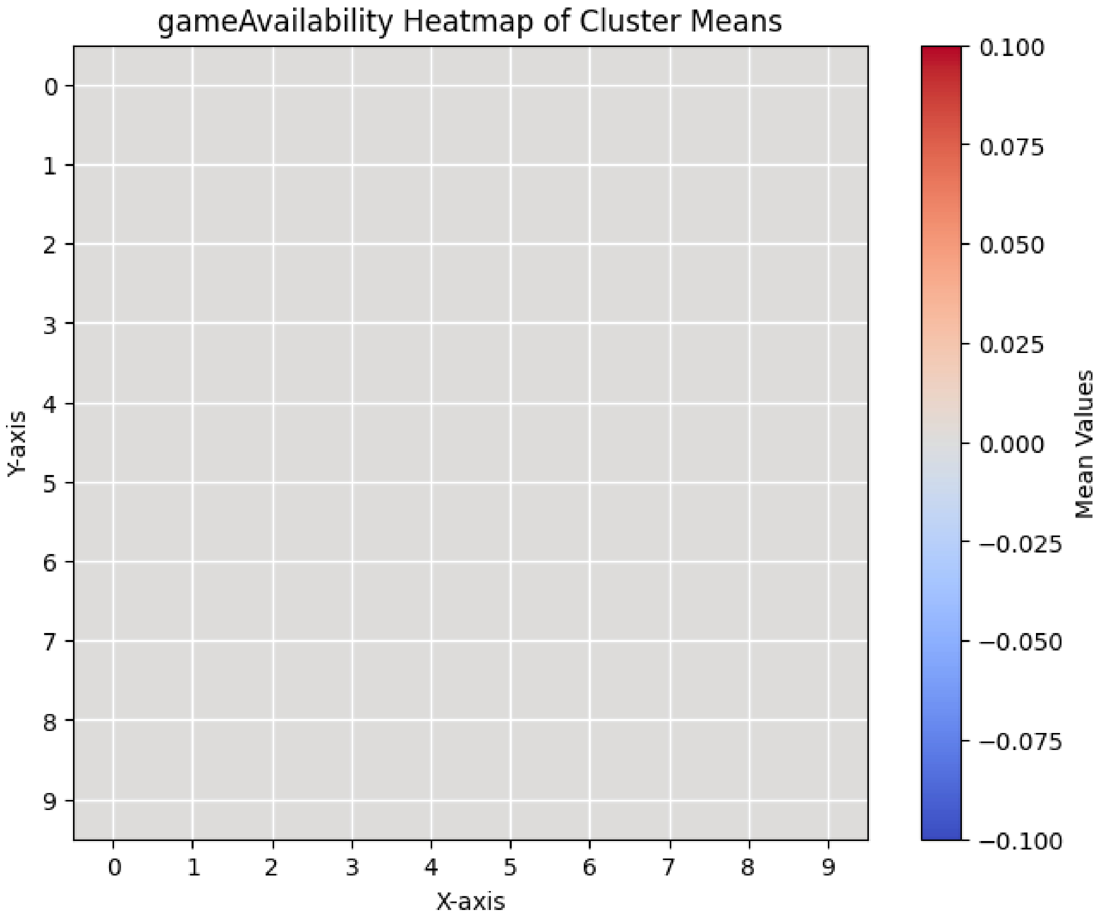

colorIdentity Heatmap of Cluster Means
Y-axis
X-axis
Mean Values
0
1
2
3
4
5
6
7
8
9
25
20
15
10
5
0

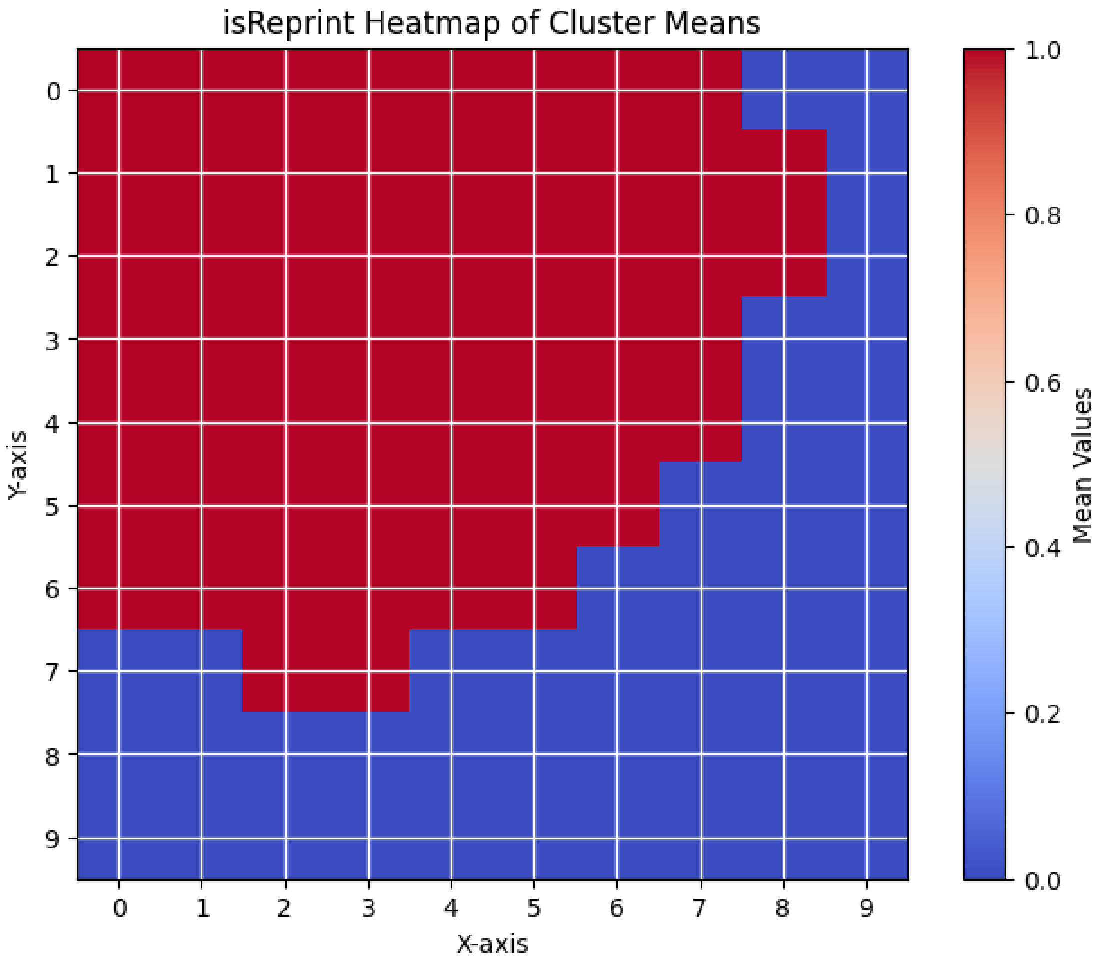

colors Heatmap of Cluster Means
Y-axis
X-axis
Mean Values
0
1
2
3
4
5
6
7
8
9
25
20
15
10
5
0


edhrecRank Heatmap of Cluster Means
Y-axis
0
1
2
3
4
5
6
7
8
9
X-axis
0
1
2
3
4
5
6
7
8
9
Mean Values
10000
8000
6000
4000
2000
0

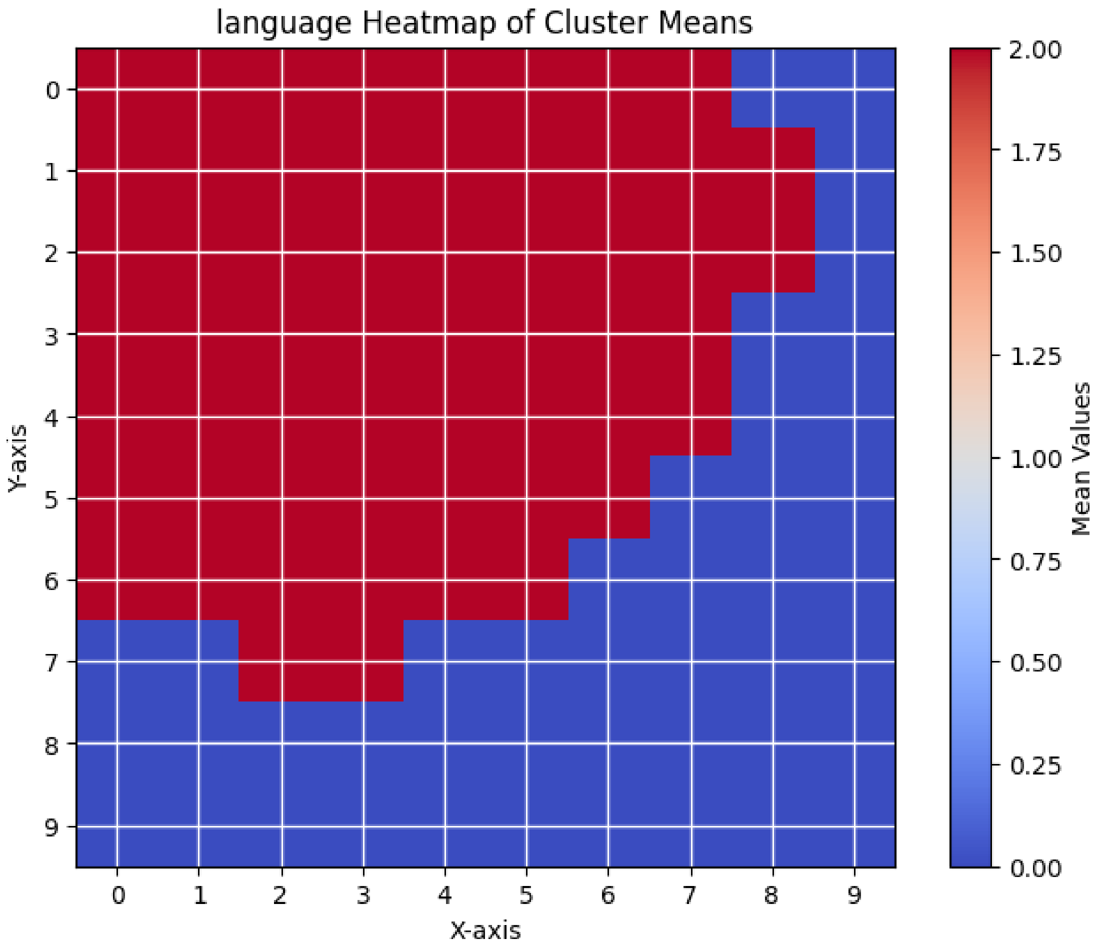


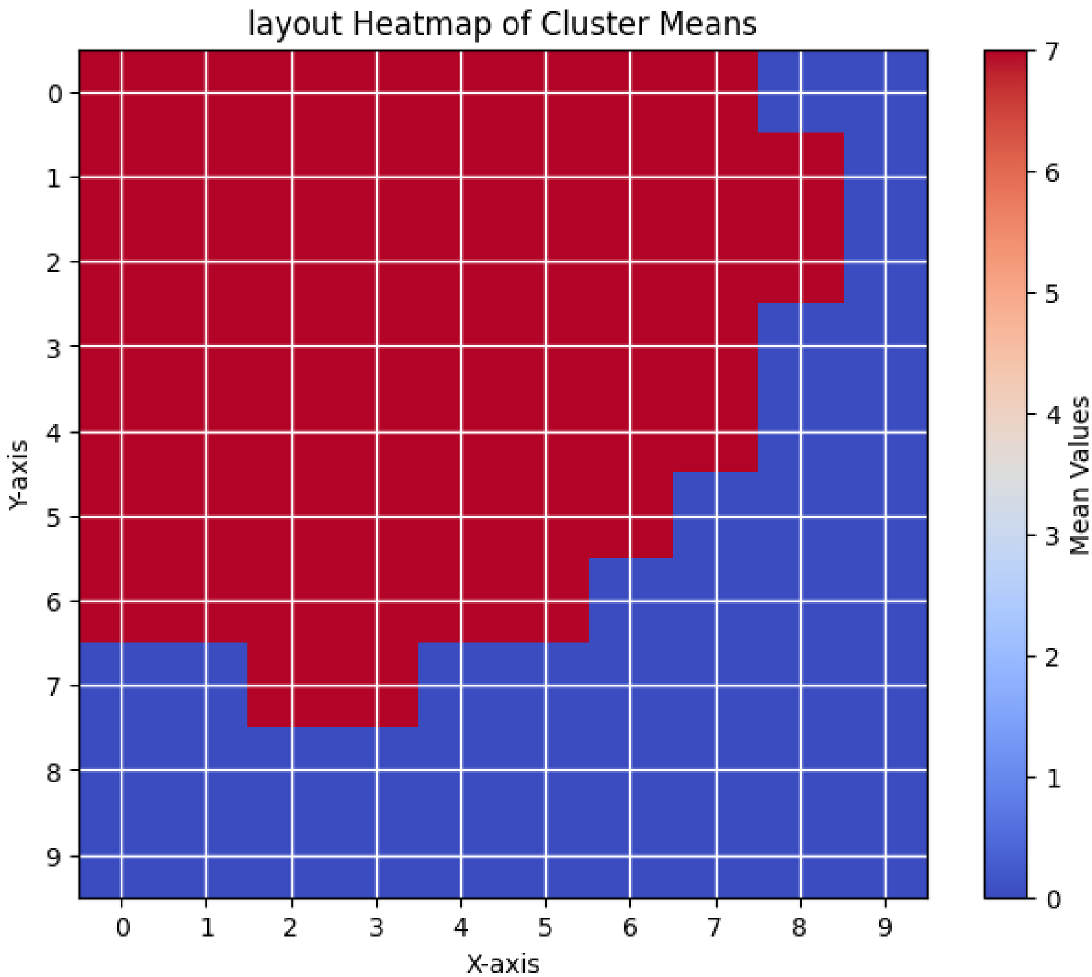
edhrecSaltiness Heatmap of Cluster Means
Y-axis
0
1
2
3
4
5
6
7
8
9
X-axis
0
1
2
3
4
5
6
7
8
9
Mean Values
0.0
0.1
0.2
0.3
0.4


gameAvailability Heatmap of Cluster Means
Y-axis
X-axis
Mean Values
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
0
1
2
3
4
5
6
7
8
9

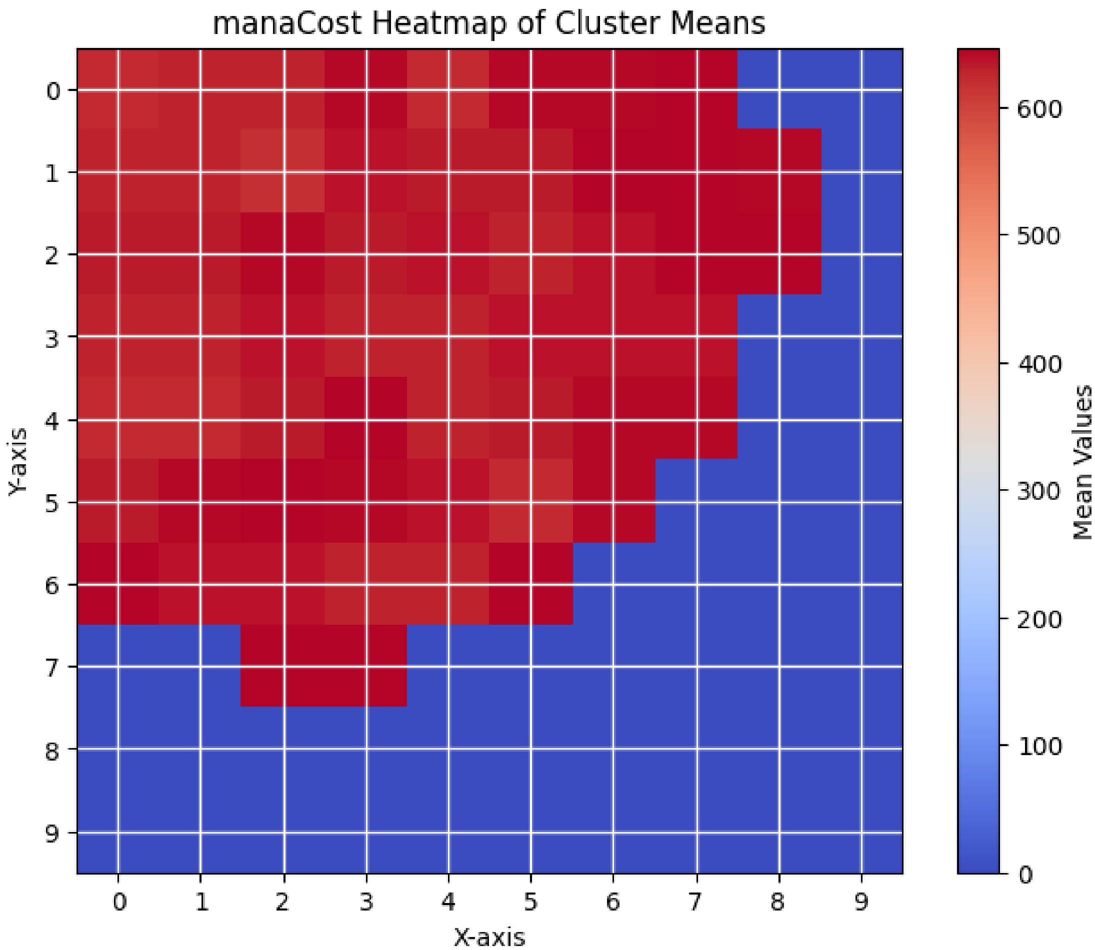


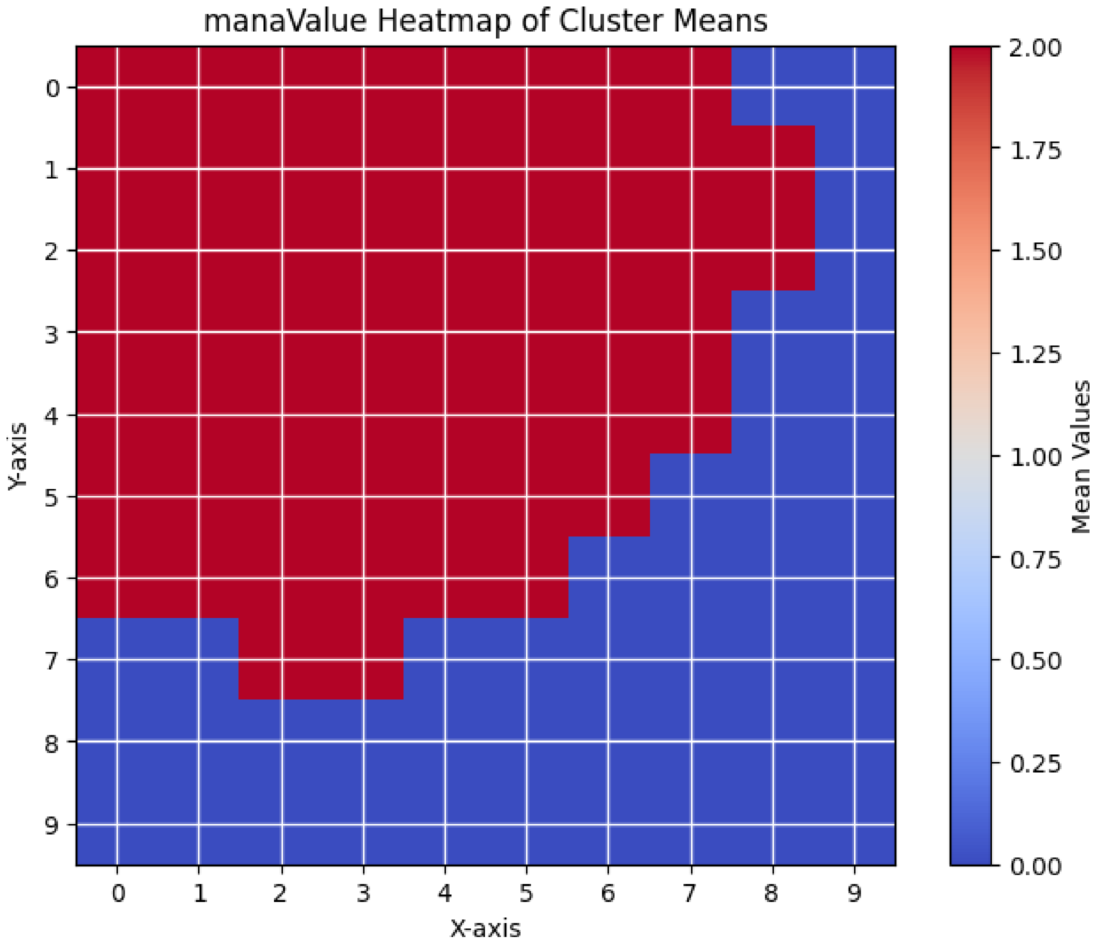
isReprint Heatmap of Cluster Means
Y-axis
X-axis
Mean Values
0
1
2
3
4
5
6
7
8
9
0.0
0.2
0.4
0.6
0.8
1.0

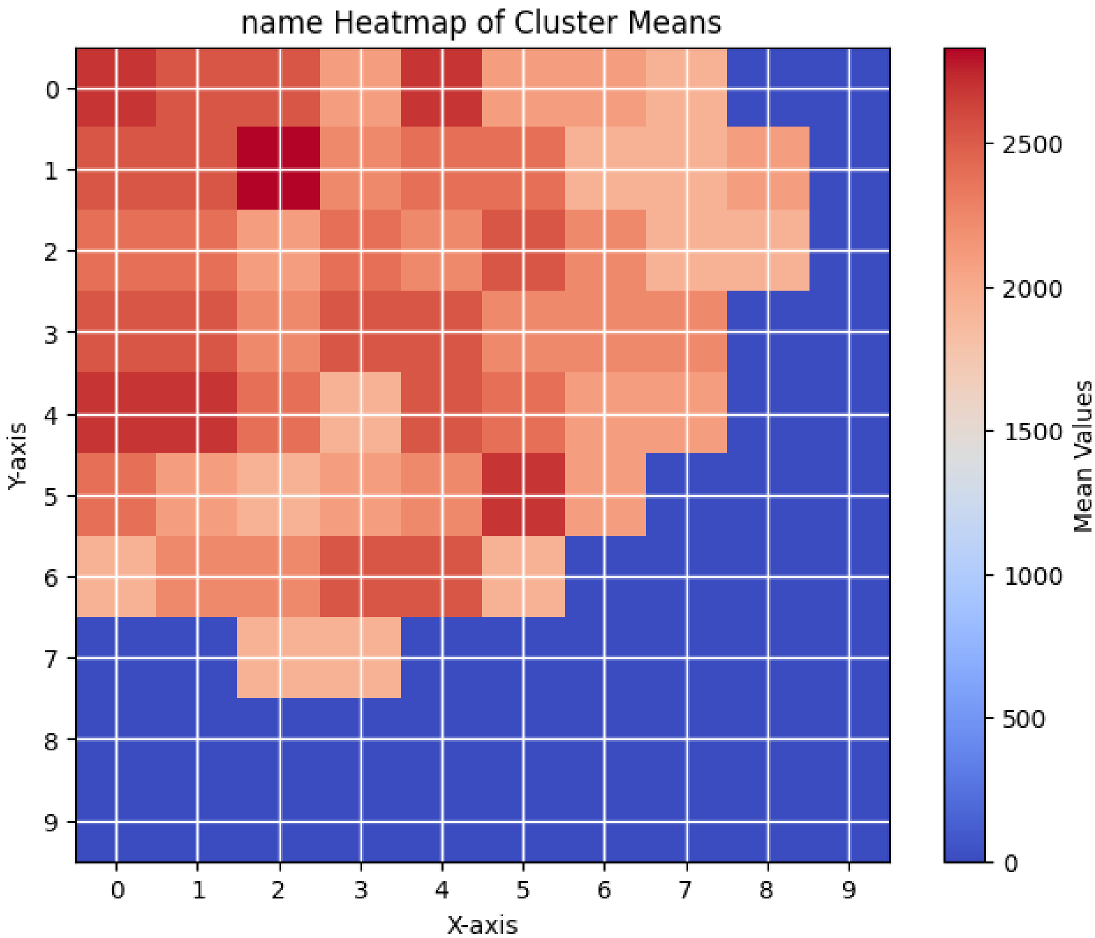

language Heatmap of Cluster Means
Y-axis
X-axis
0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Mean Values

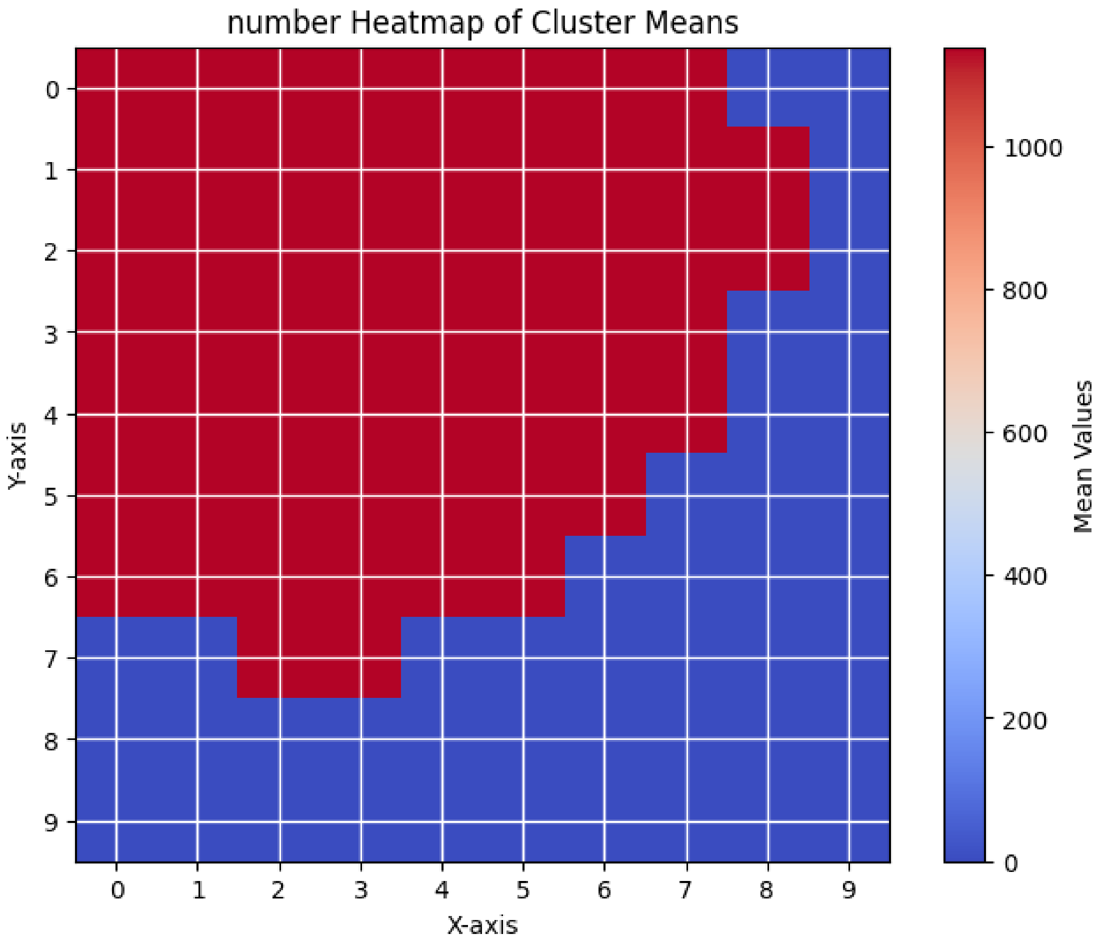


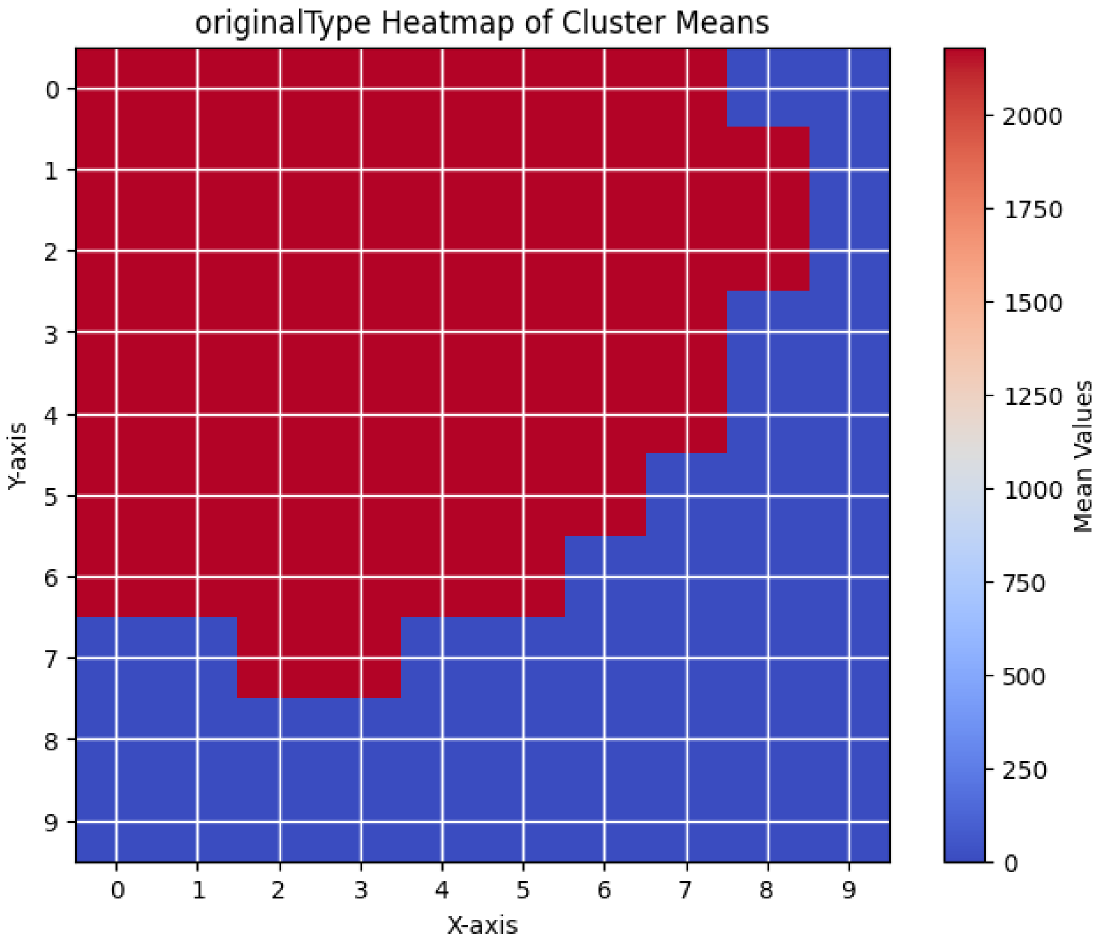
layout Heatmap of Cluster Means
Y-axis
X-axis
Mean Values
0
1
2
3
4
5
6
7
8
9

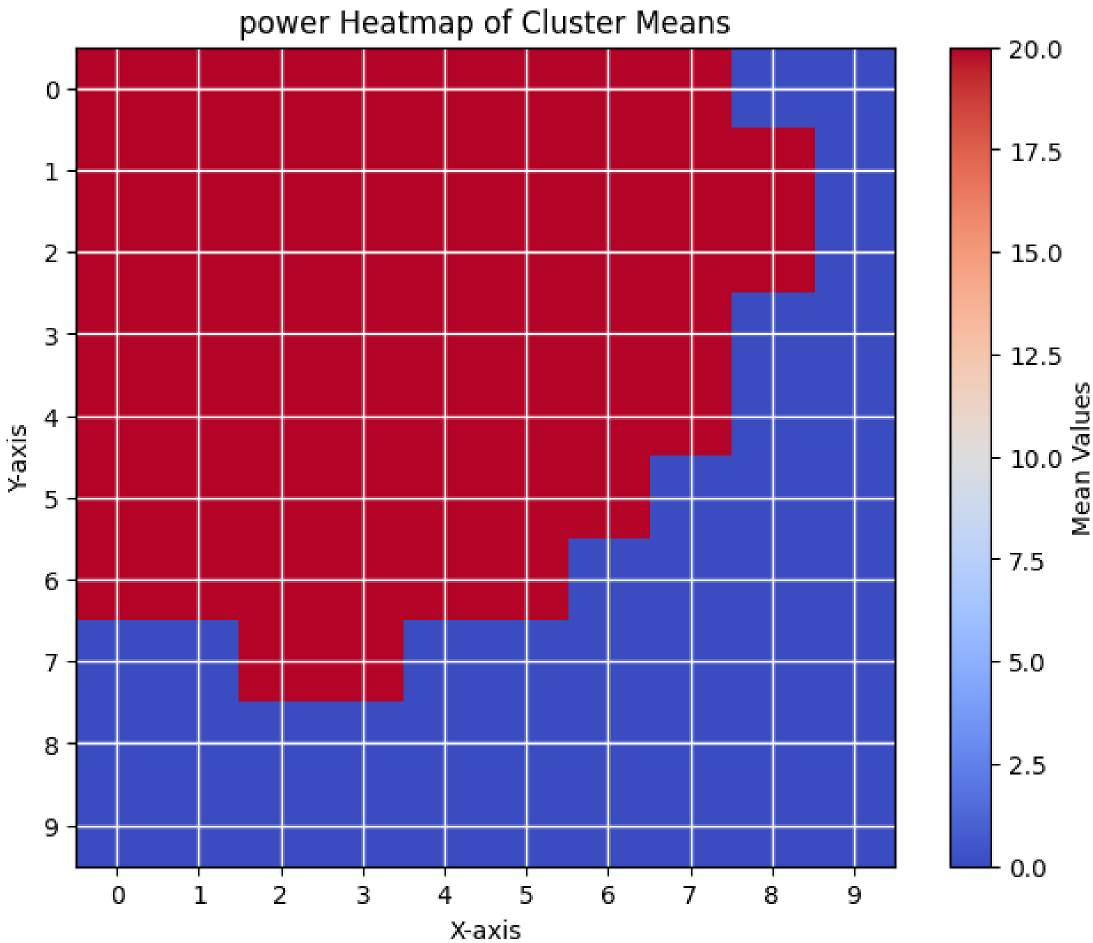

manaCost Heatmap of Cluster Means
Y-axis
0
1
2
3
4
5
6
7
8
9
X-axis
Mean Values
0
100
200
300
400
500
600

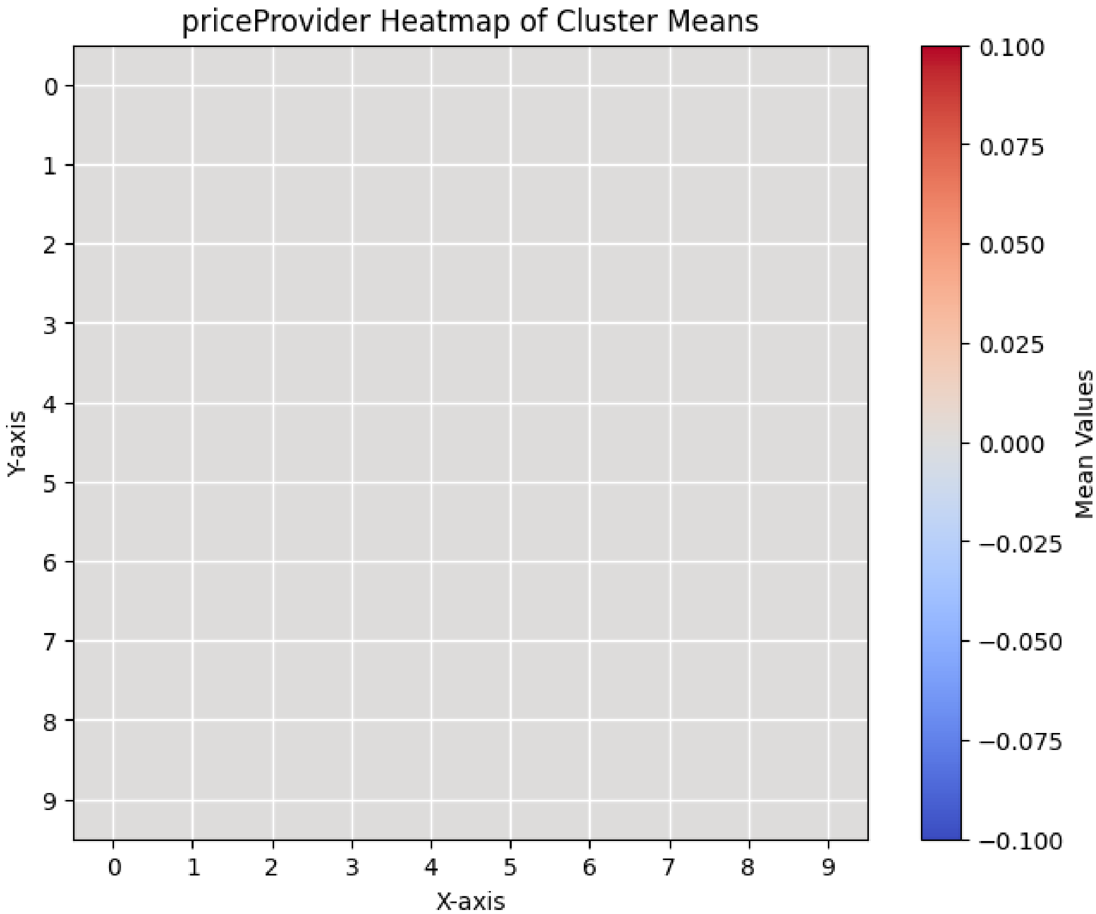

manaValue Heatmap of Cluster Means
Y-axis
X-axis
0
1
2
3
4
5
6
7
8
9
Mean Values
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

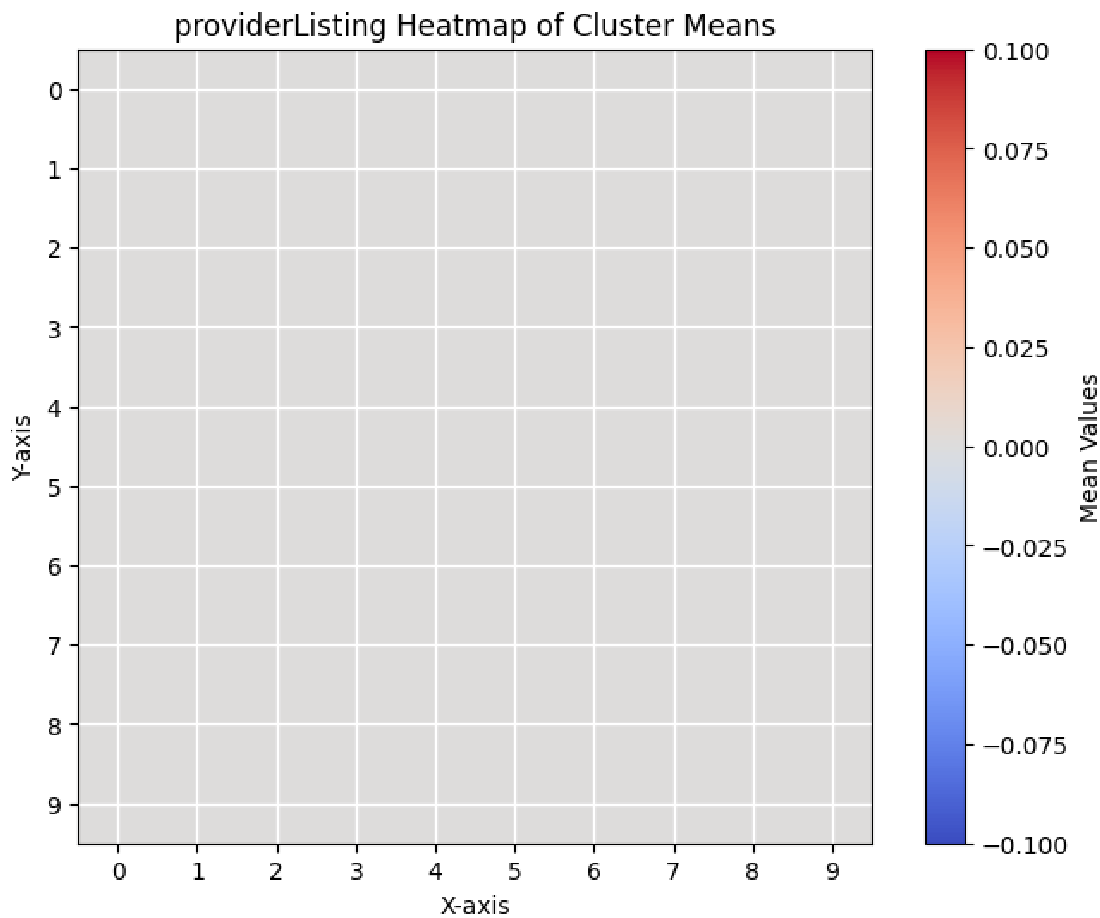

name Heatmap of Cluster Means
Y-axis
X-axis
Mean Values
0
1
2
3
4
5
6
7
8
9
0
500
1000
1500
2000
2500

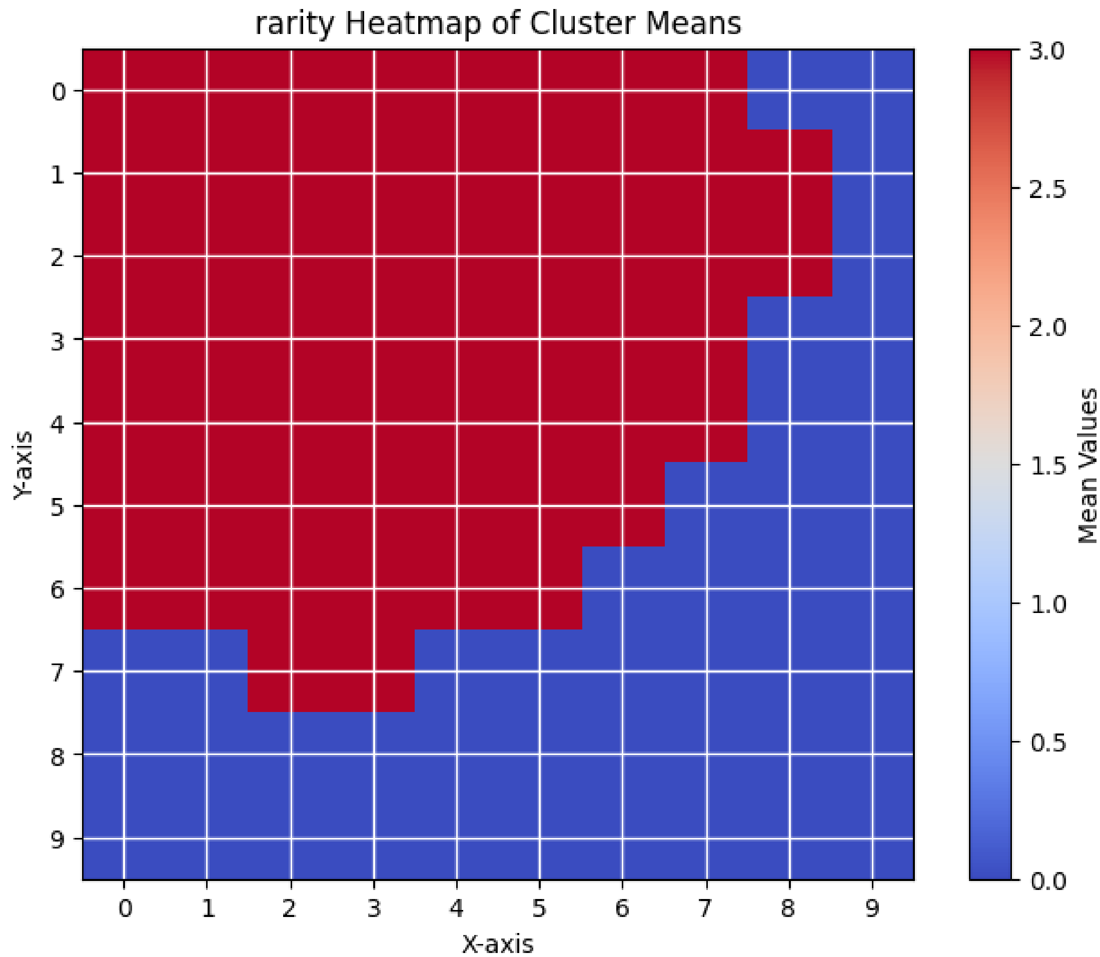

number Heatmap of Cluster Means
0
1
2
3
4
5
6
7
8
9
Y-axis
X-axis
1000
800
600
400
200
0
Mean Values

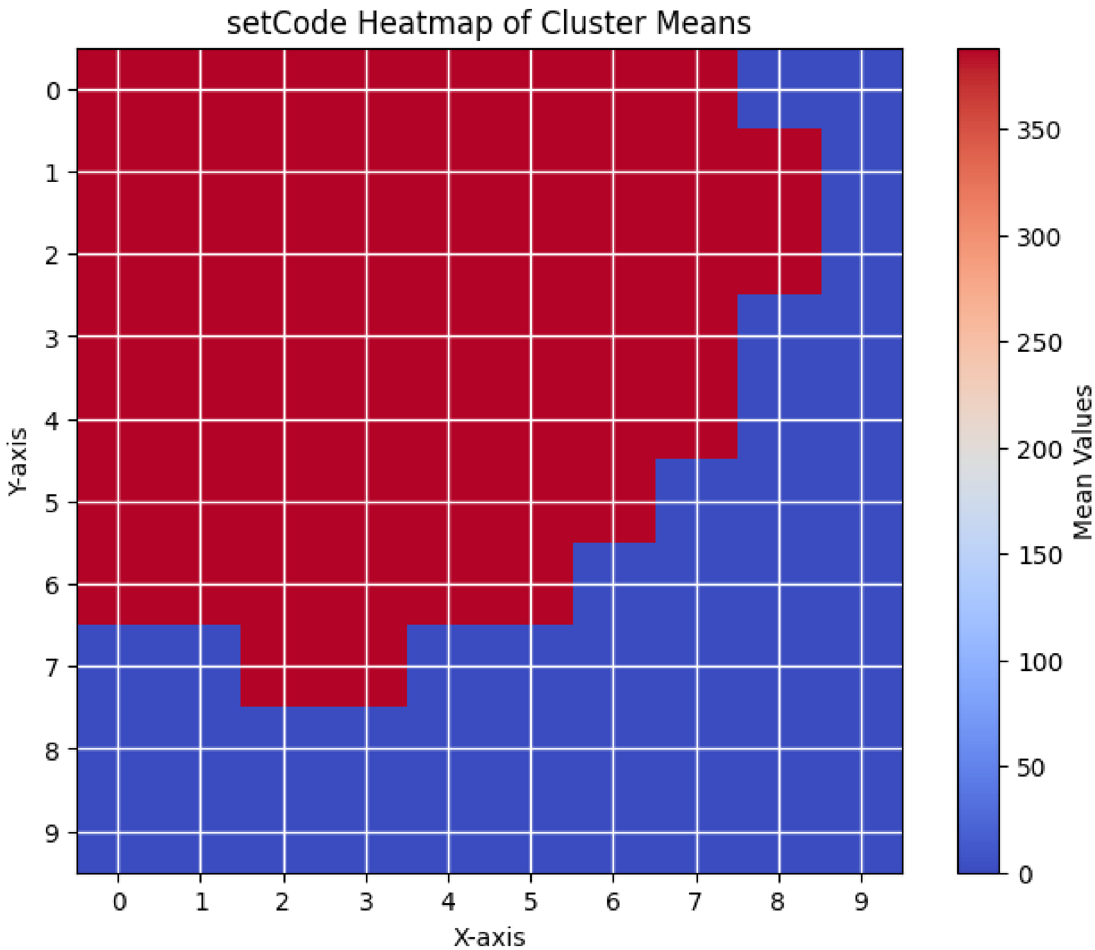

originalType Heatmap of Cluster Means
Y-axis
X-axis
0
1
2
3
4
5
6
7
8
9
Mean Values
2000
1750
1500
1250
1000
750
500
250
0

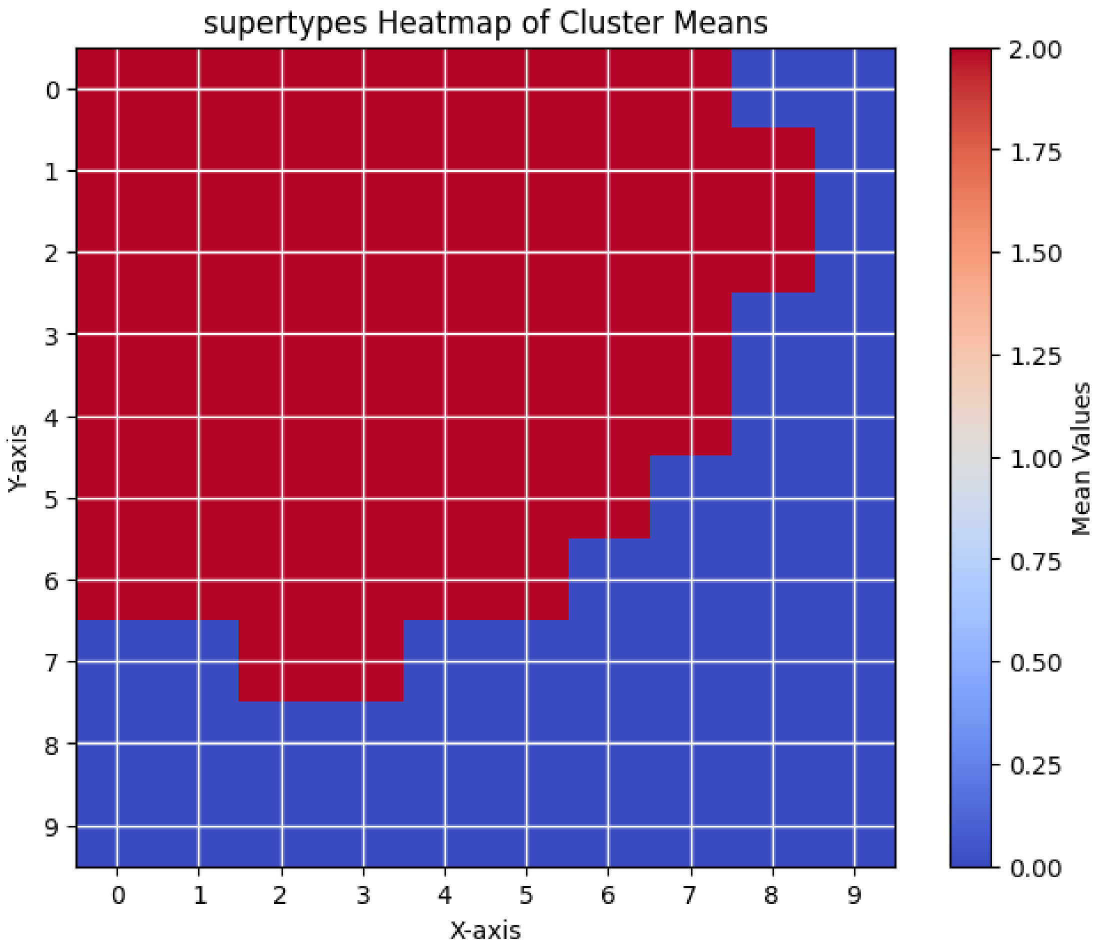

power Heatmap of Cluster Means
Y-axis
X-axis
Mean Values
0
1
2
3
4
5
6
7
8
9
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0


priceProvider Heatmap of Cluster Means
Y-axis
X-axis
0
1
2
3
4
5
6
7
8
9
Mean Values
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100

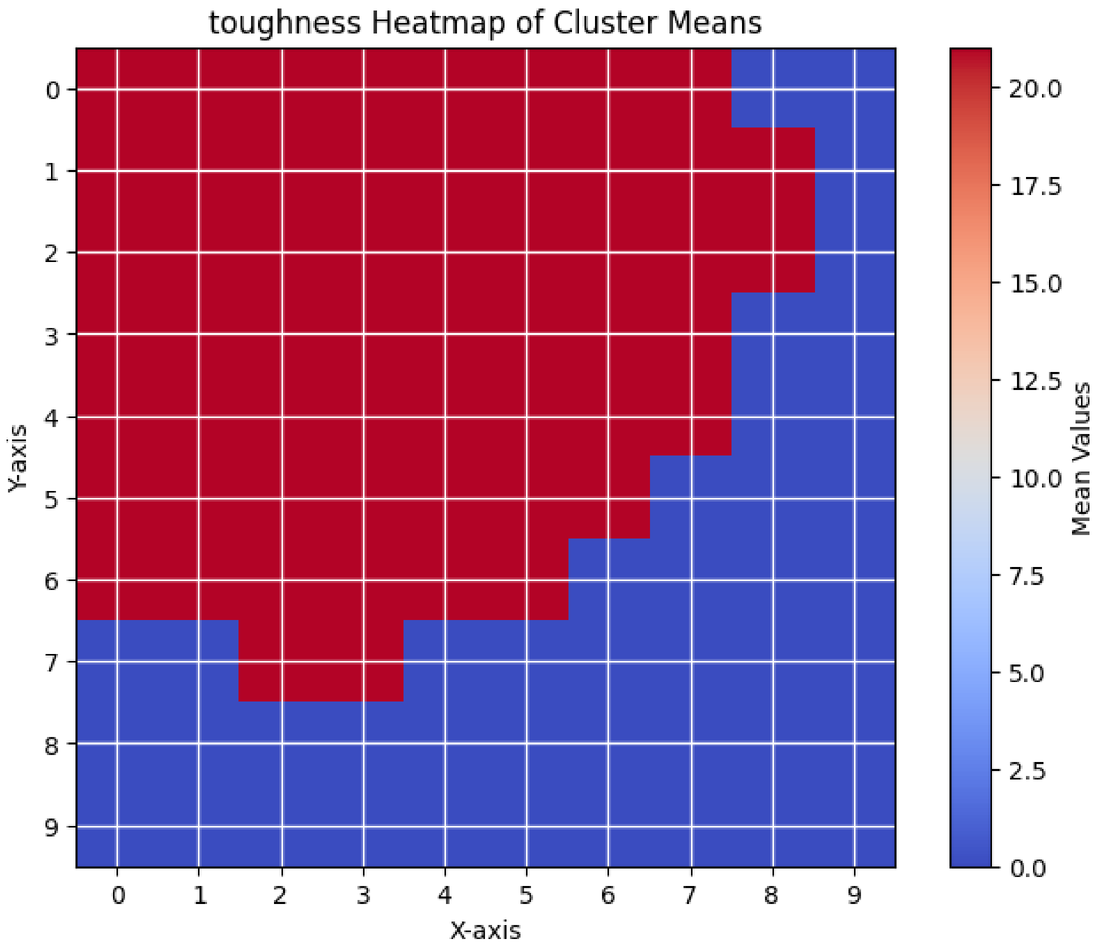

providerListing Heatmap of Cluster Means
Y-axis
0
1
2
3
4
5
6
7
8
9
X-axis
0
1
2
3
4
5
6
7
8
9
Mean Values
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100

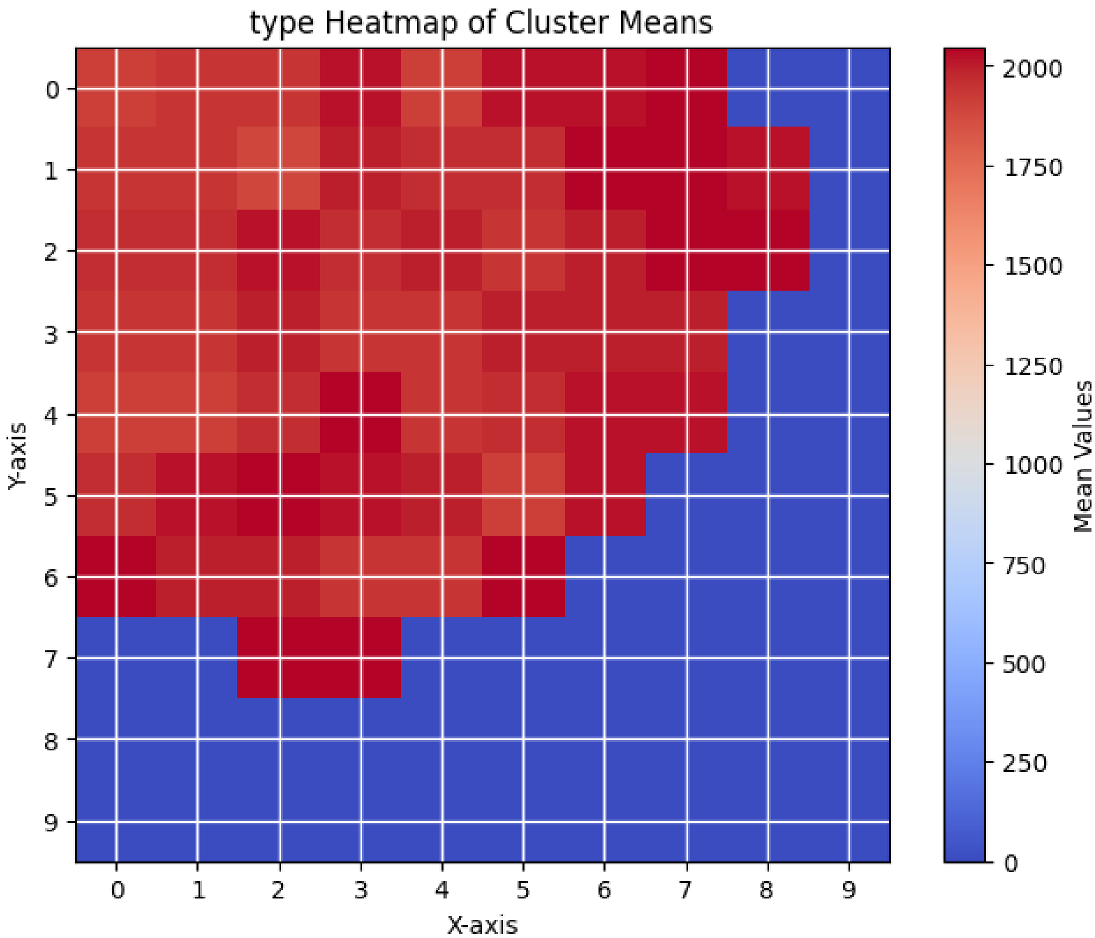

rarity Heatmap of Cluster Means
Y-axis
X-axis
Mean Values
0
1
2
3
4
5
6
7
8
9
0.0
0.5
1.0
1.5
2.0
2.5
3.0

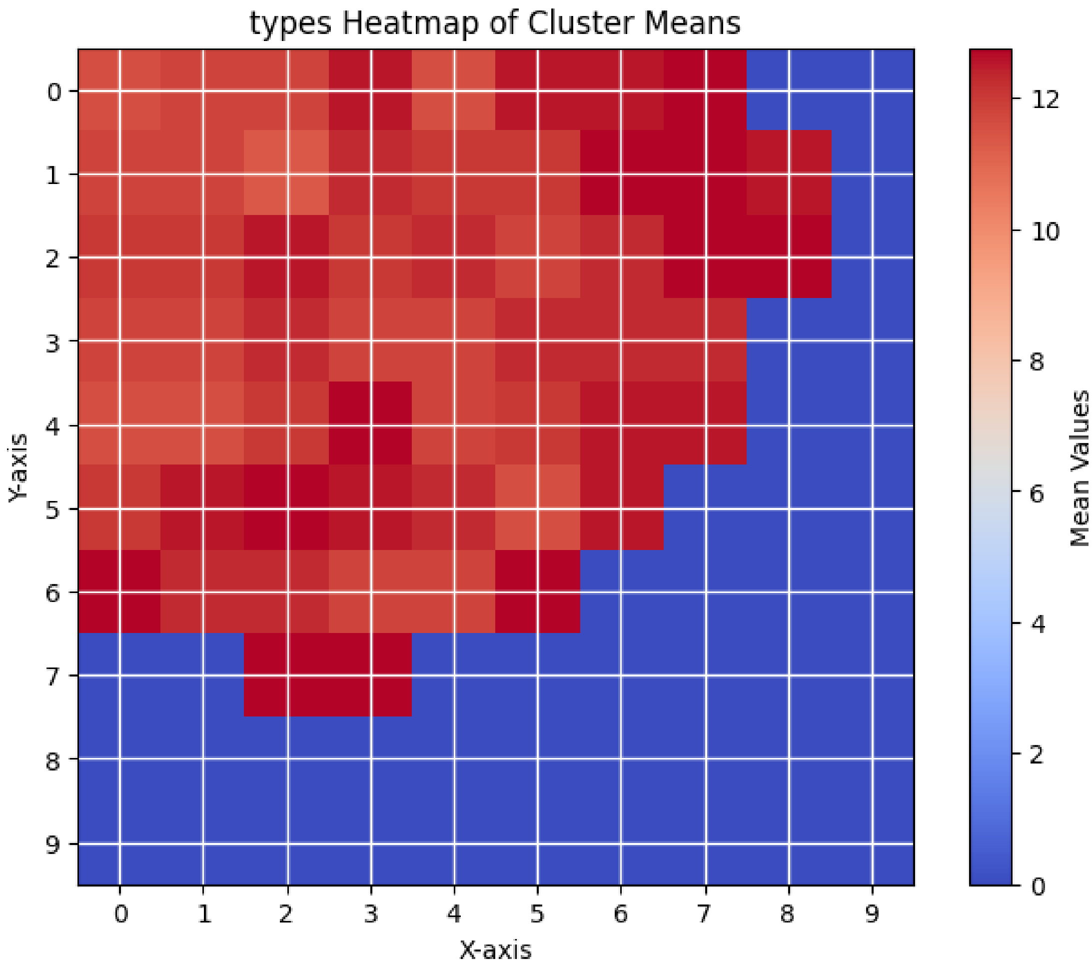

setCode Heatmap of Cluster Means
Y-axis
X-axis
0
1
2
3
4
5
6
7
8
9
Mean Values
350
300
250
200
150
100
50



supertypes Heatmap of Cluster Means
Y-axis
X-axis
0
1
2
3
4
5
6
7
8
9
Mean Values
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00


toughness Heatmap of Cluster Means
Y-axis
X-axis
Mean Values
0
1
2
3
4
5
6
7
8
9
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


type Heatmap of Cluster Means
Y-axis
X-axis
Mean Values
0
1
2
3
4
5
6
7
8
9
0
250
500
750
1000
1250
1500
1750
2000




```
[8]: scaler = StandardScaler()
     scaled_data = scaler.fit_transform(data)

     inertia = []
     silhouette_scores = []
     k_values = range(2,11)

     for k in k_values:
         kmeans = KMeans(n_clusters=k, random_state=42)
         kmeans.fit(scaled_data)
         inertia.append(kmeans.inertia_)
         silhouette_scores.append(silhouette_score(scaled_data, kmeans.labels_))

     # Step 4: Plot the Elbow Method results
     plt.figure(figsize=(12, 5))
```

```
# Inertia Plot
plt.subplot(1, 2, 1)
plt.plot(k_values, inertia, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')

# Silhouette Score Plot
plt.subplot(1, 2, 2)
plt.plot(k_values, silhouette_scores, marker='o')
plt.title('Silhouette Scores')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Silhouette Score')

plt.tight_layout()
plt.show()

# Step 5: Fit the K-Means model with the chosen k (e.g., k=3)
optimal_k = 3  # Choose based on the Elbow method or Silhouette score
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
kmeans.fit(scaled_data)

# Step 6: Add cluster labels to the original DataFrame
data['cluster'] = kmeans.labels_

# Step 7: Visualize the clusters
plt.figure(figsize=(8, 6))
plt.scatter(data['price'], data['cardFinish'], c=data['cluster'],␣
↪cmap='viridis', marker='o')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],␣
↪s=300, c='red', marker='X')  # Cluster centers
plt.title('K-Means Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.grid()
plt.show()
```