Practical implementation of k-means clustering

CUSTOMER SEGMENTATION IN PYTHON

Karolis Urbonas
Head of Data Science, Amazon

Key steps

- Data pre-processing
- Choosing a number of clusters
- Running k-means clustering on pre-processed data
- Analyzing average RFM values of each cluster

Data pre-processing

We've completed the pre-processing steps and have these two objects:

- `datamart_rfm`
- `datamart_normalized`

```
import numpy as np
datamart_log = np.log(datamart_rfm)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(datamart_log)

datamart_normalized = scaler.transform(datamart_log)
```

Methods to define the number of clusters

- Visual methods - elbow criterion
- Mathematical methods - silhouette coefficient
- Experimentation and interpretation

Running k-means

```
# Import package
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2, random_state=1)
```

```
# Compute k-means clustering on pre-processed data
kmeans.fit(datamart_normalized)
```

```
# Extract cluster labels from labels_ attribute
cluster_labels = kmeans.labels_
```

Analyzing average RFM values of each cluster

```
# Create a cluster label column in the original DataFrame
datamart_rfm_k2 = datamart_rfm.assign(Cluster = cluster_labels)
```

```
# Calculate average RFM values and size for each cluster
datamart_rfm_k2.groupby(['Cluster']).agg({
    'Recency': 'mean',
    'Frequency': 'mean',
    'MonetaryValue': ['mean', 'count'],
}).round(0)
```

Analyzing average RFM values of each cluster

The result of a simple 2-cluster solution:

	Recency	Frequency	MonetaryValue	
	mean	mean	mean	count
cluster				
0	137.0	5.0	92.0	2023
1	32.0	35.0	719.0	1620

Let's practice running k-means clustering!

CUSTOMER SEGMENTATION IN PYTHON

Choosing number of clusters

CUSTOMER SEGMENTATION IN PYTHON

Karolis Urbonas
Head of Data Science, Amazon

Methods

- Visual methods - elbow criterion
- Mathematical methods - silhouette coefficient
- Experimentation and interpretation

Elbow criterion method

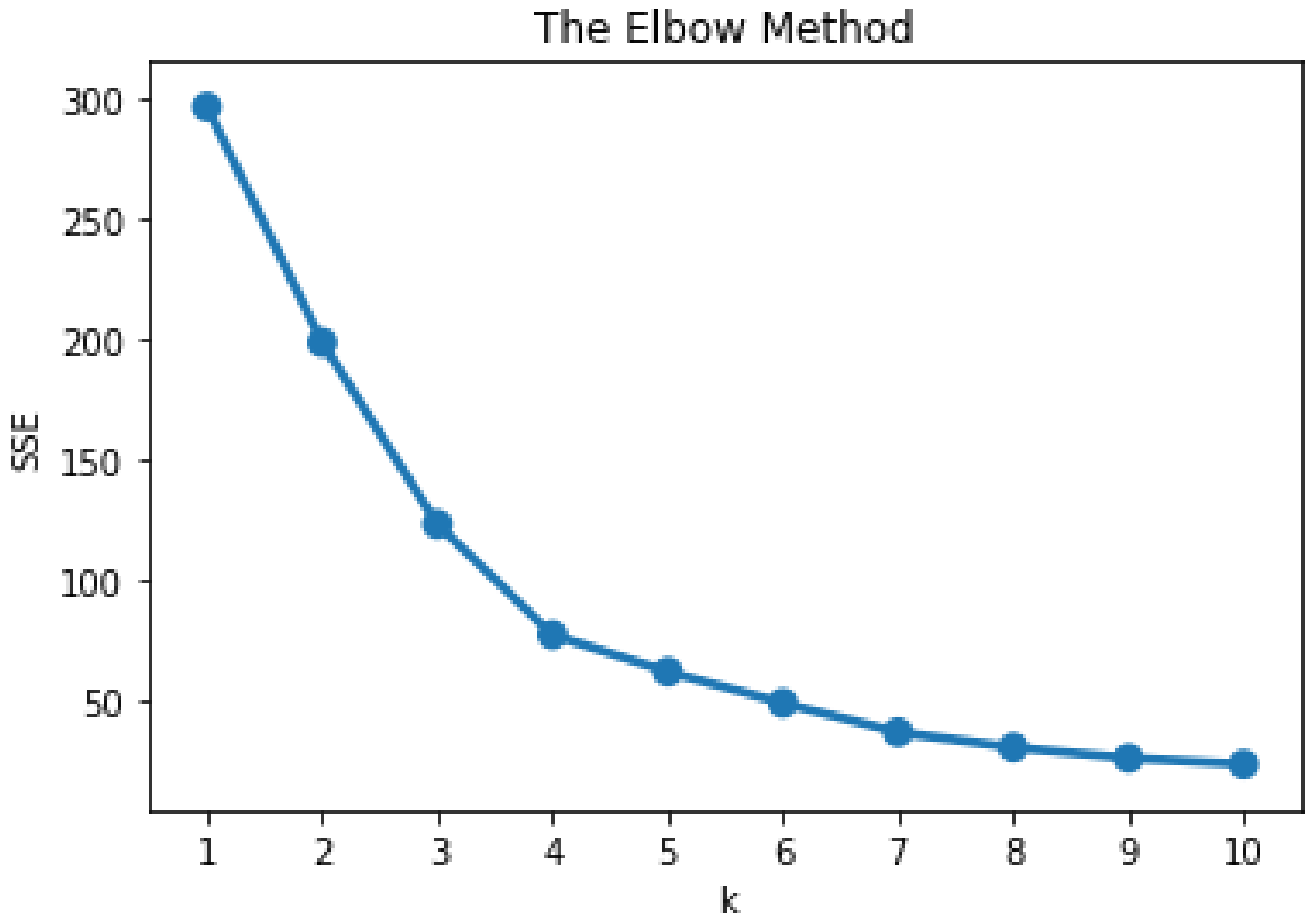
- Plot the number of clusters against within-cluster sum-of-squared-errors (SSE) - *sum of squared distances from every data point to their cluster center*
- Identify an "elbow" in the plot
- Elbow - a point representing an "optimal" number of clusters

Elbow criterion method

```python
# Import key libraries
from sklearn.cluster import KMeans
import seaborn as sns
from matplotlib import pyplot as plt
# Fit KMeans and calculate SSE for each *k*
sse = {}
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=1)
    kmeans.fit(data_normalized)
    sse[k] = kmeans.inertia_ # sum of squared distances to closest cluster center
# Plot SSE for each *k*
plt.title('The Elbow Method')
plt.xlabel('k'); plt.ylabel('SSE')
sns.pointplot(x=list(sse.keys()), y=list(sse.values()))
plt.show()
```

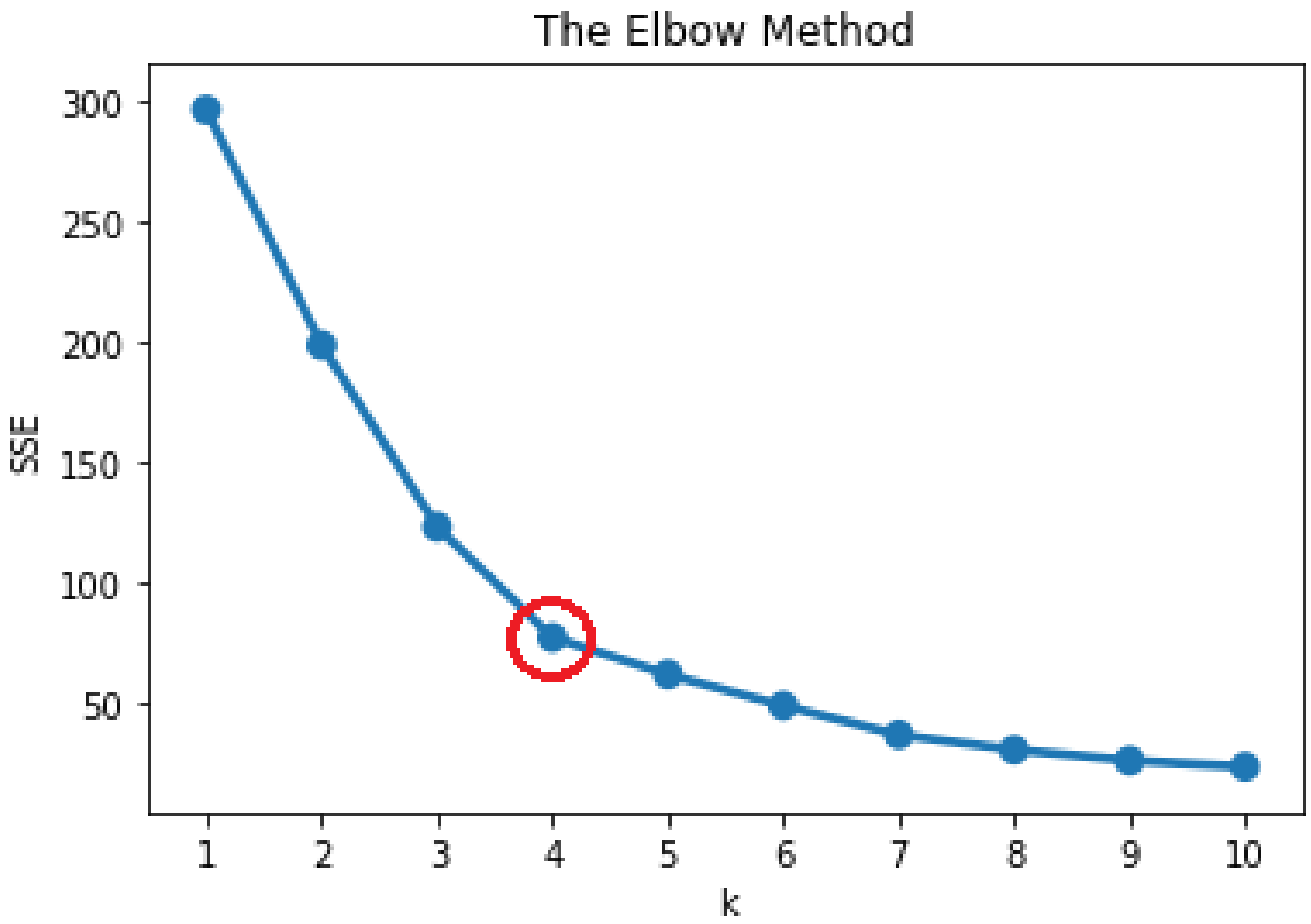
Elbow criterion method

The elbow criterion chart:



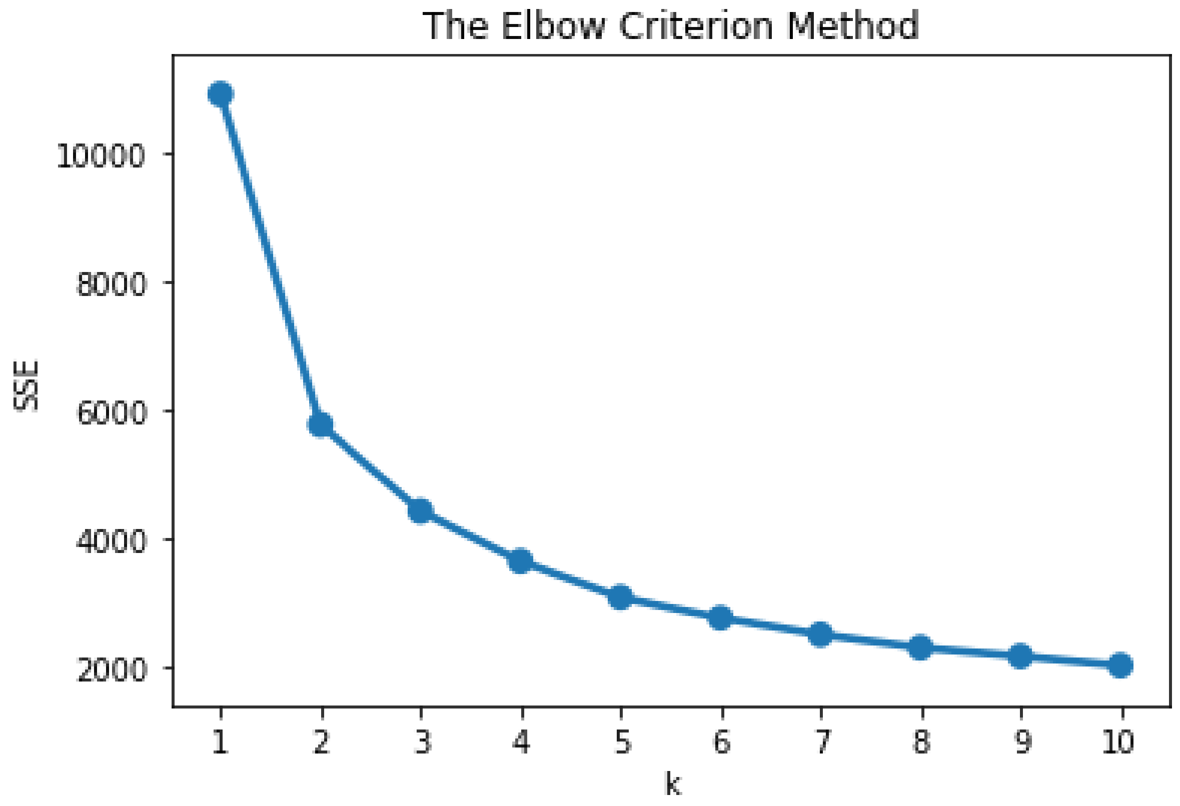
Elbow criterion method

The elbow criterion chart:



Using elbow criterion method

- Best to choose the point on elbow, or the next point
- Use as a guide but test multiple solutions
- Elbow plot built on `datamart_rfm`



Experimental approach - analyze segments

- Build clustering at and around elbow solution
- Analyze their properties - average RFM values
- Compare against each other and choose one which makes most business sense

Experimental approach - analyze segments

- Previous 2-cluster solution

	Recency	Frequency	MonetaryValue	
	mean	mean	mean	count
cluster				
0	137.0	5.0	92.0	2023
1	32.0	35.0	719.0	1620

- 3-cluster solution on the same normalized RFM dataset

	Recency	Frequency	MonetaryValue	
	mean	mean	mean	count
cluster				
0	16.0	50.0	1051.0	901
1	167.0	3.0	53.0	1156
2	77.0	12.0	216.0	1586

Let's practice finding the optimal number of clusters!

CUSTOMER SEGMENTATION IN PYTHON

Profile and interpret segments

CUSTOMER SEGMENTATION IN PYTHON

Karolis Urbonas
Head of Data Science, Amazon

Approaches to build customer personas

- Summary statistics for each cluster e.g. average RFM values
- Snake plots (from market research)
- Relative importance of cluster attributes compared to population

Summary statistics of each cluster

- Run k-means segmentation for several **k** values around the recommended value.
- Create a cluster label column in the **original** DataFrame:

```
datamart_rfm_k2 = datamart_rfm.assign(Cluster = cluster_labels)
```

Calculate average RFM values and sizes for each cluster:

```
datamart_rfm_k2.groupby(['Cluster']).agg({
    'Recency': 'mean',
    'Frequency': 'mean',
    'MonetaryValue': ['mean', 'count'],
}).round(0)
```

  - Repeat the same for **k=3**

Summary statistics of each cluster

- Compare average RFM values of each clustering solution

	Recency	Frequency	MonetaryValue	
	mean	mean	mean	count
cluster				
0	137.0	5.0	92.0	2023
1	32.0	35.0	719.0	1620

	Recency	Frequency	MonetaryValue	
	mean	mean	mean	count
cluster				
0	16.0	50.0	1051.0	901
1	167.0	3.0	53.0	1156
2	77.0	12.0	216.0	1586

Snake plots to understand and compare segments

- Market research technique to compare different segments
- Visual representation of each segment's attributes
- Need to first normalize data (center & scale)
- Plot each cluster's average normalized values of each attribute

Prepare data for a snake plot

Transform `datamart_normalized` as DataFrame and add a `Cluster` column

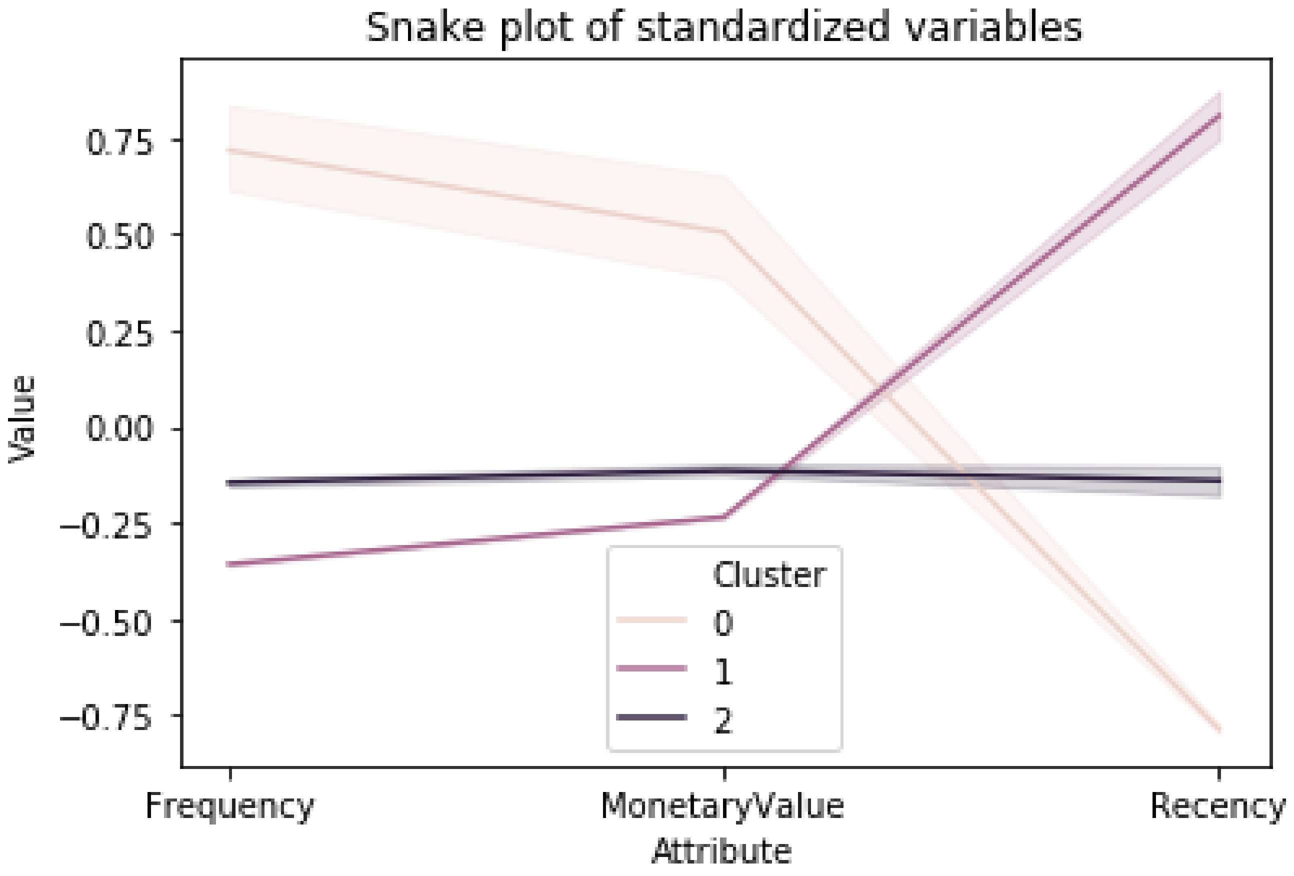
```
datamart_normalized = pd.DataFrame(datamart_normalized,
                                   index=datamart_rfm.index,
                                   columns=datamart_rfm.columns)
datamart_normalized['Cluster'] = datamart_rfm_k3['Cluster']
```

Melt the data into a long format so RFM values and metric names are stored in 1 column each

```
datamart_melt = pd.melt(datamart_normalized.reset_index(),
                    id_vars=['CustomerID', 'Cluster'],
                    value_vars=['Recency', 'Frequency', 'MonetaryValue'],
                    var_name='Attribute',
                    value_name='Value')
```

Visualize a snake plot

```
plt.title('Snake plot of standardized variables')
sns.lineplot(x="Attribute", y="Value", hue='Cluster', data=datamart_melt)
```



Relative importance of segment attributes

- Useful technique to identify relative importance of each segment's attribute
- Calculate average values of each cluster
- Calculate average values of population
- Calculate importance score by dividing them and subtracting 1 (*ensures 0 is returned when cluster average equals population average*)

```
cluster_avg = datamart_rfm_k3.groupby(['Cluster']).mean()
population_avg = datamart_rfm.mean()
relative_imp = cluster_avg / population_avg - 1
```

Analyze and plot relative importance

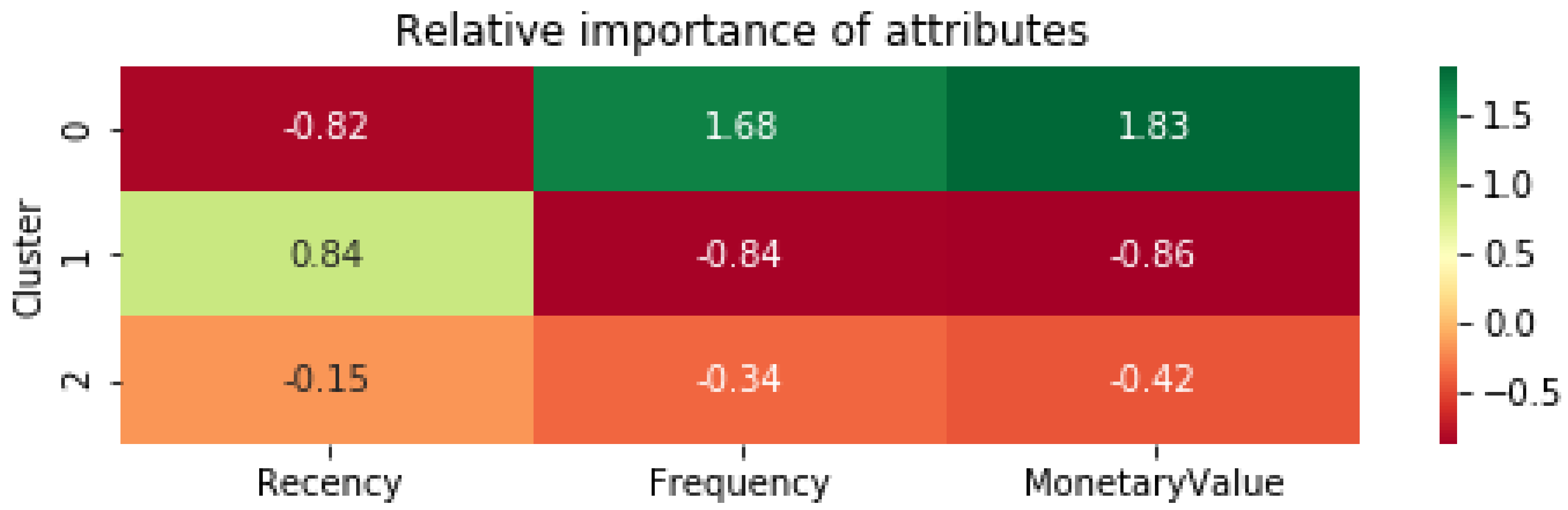
- As a ratio moves away from 0, attribute importance for a segment (relative to total pop.) increases.

```
relative_imp.round(2)
```

```
         Recency  Frequency  MonetaryValue
Cluster
0          -0.82       1.68           1.83
1           0.84      -0.84          -0.86
2          -0.15      -0.34          -0.42
```

```
# Plot heatmap
plt.figure(figsize=(8, 2))
plt.title('Relative importance of attributes')
sns.heatmap(data=relative_imp, annot=True, fmt='.2f', cmap='RdYlGn')
plt.show()
```

Relative importance heatmap



Cluster	Recency	Frequency	MonetaryValue
0	-0.82	1.68	1.83
1	0.84	-0.84	-0.86
2	-0.15	-0.34	-0.42

Your time to experiment with different customer profiling techniques!

CUSTOMER SEGMENTATION IN PYTHON

Implement end-to-end segmentation solution

CUSTOMER SEGMENTATION IN PYTHON

Karolis Urbonas
Head of Data Science, Amazon

Key steps of the segmentation project

- Gather data - updated data with an additional variable
- Pre-process the data
- Explore the data and decide on the number of clusters
- Run k-means clustering
- Analyze and visualize results

Updated RFM data

- Same RFM values plus additional `Tenure` variable
- Tenure - time since the first transaction
- Defines how long the customer has been with the company

CustomerID	Recency	Frequency	MonetaryValue	Tenure
12747	3	25	948.70	362
12748	1	888	7046.16	365
12749	4	37	813.45	214
12820	4	17	268.02	327
12822	71	9	146.15	88

Goals for this project

- Remember key pre-processing rules
- Apply data exploration techniques
- Practice running several k-means iterations
- Analyze results quantitatively and visually

Let's dig in!

CUSTOMER SEGMENTATION IN PYTHON

Final thoughts

CUSTOMER SEGMENTATION IN PYTHON

Karolis Urbonas
Head of Data Science, Amazon

What you have learned

- Cohort analysis and visualization
- RFM segmentation
- Data pre-processing for k-means
- Customer segmentation with k-means
  - Evaluating number of clusters
  - Reviewing and visualizing segmentation solutions

Congratulations!

CUSTOMER SEGMENTATION IN PYTHON