Market segmentation example

In this notebook we explore a bit more sophisticated example of clustering

Import the relevant libraries

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# Set the styles to Seaborn
sns.set()
# Import the KMeans module so we can perform k-means clustering with sklearn
from sklearn.cluster import KMeans
```

Load the data

In [ ]:

```
# Load the data
data = pd.read_csv ('3.12. Example.csv')
```

In [ ]:

```
# Check what's inside
data
```

Plot the data

Create a preliminary plot to see if you can spot something

In [ ]:

```
# We are creating a scatter plot of the two variables
plt.scatter(data['Satisfaction'],data['Loyalty'])
# Name your axes
plt.xlabel('Satisfaction')
plt.ylabel('Loyalty')
```

Select the features

In [ ]:

```
# Select both features by creating a copy of the data variable
x = data.copy()
```

Clustering

In [ ]:

```
# Create an object (which we would call kmeans)
# The number in the brackets is K, or the number of clusters we are aiming for
kmeans = KMeans(2)
# Fit the data
kmeans.fit(x)
```

Clustering results

In [ ]:

```
# Create a copy of the input data
clusters = x.copy()
# Take note of the predicted clusters
clusters['cluster_pred']=kmeans.fit_predict(x)
```

In [ ]:

```
# Plot the data using the longitude and the latitude
# c (color) is an argument which could be coded with a variable
# The variable in this case has values 0,1, indicating to plt.scatter, that there are t
# All points in cluster 0 will be the same colour, all points in cluster 1 - another or
# cmap is the color map. Rainbow is a nice one, but you can check others here: https://
plt.scatter(clusters['Satisfaction'],clusters['Loyalty'],c=clusters['cluster_pred'],cma
plt.xlabel('Satisfaction')
plt.ylabel('Loyalty')
```

Standardize the variables

Let's standardize and check the new result

In [ ]:

```
# Import a library which can do that easily
from sklearn import preprocessing
# Scale the inputs
# preprocessing.scale scales each variable (column in x) with respect to itself
# The new result is an array
x_scaled = preprocessing.scale(x)
x_scaled
```

Take advantage of the Elbow method

In [ ]:

```
# Createa an empty list
wcss =[]

# Create all possible cluster solutions with a loop
# We have chosen to get solutions from 1 to 9 clusters; you can ammend that if you wis
for i in range(1,10):
    # Clsuter solution with i clusters
    kmeans = KMeans(i)
    # Fit the STANDARDIZED data
    kmeans.fit(x_scaled)
    # Append the WCSS for the iteration
    wcss.append(kmeans.inertia_)

# Check the result
wcss
```

In [ ]:

```
# Plot the number of clusters vs WCSS
plt.plot(range(1,10),wcss)
# Name your axes
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
```