**# How to Choose the Number of Clusters**

**Using the same code as in the previous exercise, find the WCSS for clustering solutions with 1 to 10 clusters (you can try with more if you wish).**

**Find the most suitable solutions, run them and compare the results.**

**## Import the relevant libraries**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

sns.set()

from sklearn.cluster import KMeans

**## Load the data**

**Load data from the csv file: 'Countries\_exercise.csv'**

**# Load the data**

raw\_data = pd.read\_csv('Countries\_exercise.csv')

# Check the data

raw\_data

**## Plot the data**

**Plot the *Longtitude* and *'Latitude'* columns.**

data = raw\_data.copy()

plt.scatter(data['Longitude'], data['Latitude'])

plt.xlim(-180,180)

plt.ylim(-90, 90)

plt.show()

**## Select the features**

Make sure to select the appropriate features since we are no longer using the categorical variable for our clustering but rather 'Longitude' and 'Laditude'.

x = data.iloc[:,1:3]

x

**## Clustering.** Use 4 clusters initially.

kmeans = KMeans(4)

kmeans.fit(x)

**### Clustering Resutls**

identified\_clusters = kmeans.fit\_predict(x)

identified\_clusters

data\_with\_clusters = data.copy()

data\_with\_clusters['Cluster'] = identified\_clusters

data\_with\_clusters

**Plot the data once again. This time use the <i> c </i> parameter to separate the data by the clusters we defined.**

plt.scatter(data['Longitude'], data['Latitude'], c=data\_with\_clusters['Cluster'], cmap = 'rainbow')

plt.xlim(-180,180)

plt.ylim(-90, 90)

plt.show()

**## Selecting the number of clusters**

**### WCSS**

**Use the ingerated sklearn method 'inertia\_' .**

kmeans.inertia\_

**Write a loop that calculates and saves the WCSS for any number of clusters from 1 up to 10 (or more if you wish).**

wcss = []

**# 'cl\_num' is a that keeps track the highest number of clusters we want to use the WCSS method for.**

**# Note that 'range' doesn't include the upper boundary**

cl\_num = 11

for i in range (1,cl\_num):

kmeans= KMeans(i)

kmeans.fit(x)

wcss\_iter = kmeans.inertia\_

wcss.append(wcss\_iter)

wcss

**### The Elbow Method**

number\_clusters = range(1,cl\_num)

plt.plot(number\_clusters, wcss)

plt.title('The Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel('Within-cluster Sum of Squares')

Based on the Elbow Curve, plot several graphs with the appropriate amount of clusters you believe would best fit the data.

In this specific case, that would be 2 or 3.

Compare the scatter plots to determine which one to use in any further analysis.

*Hint: we already created the scatter plot for 4 clusters, so we only have to slightly alter our code*

kmeans = KMeans(2)

kmeans.fit(x)

identified\_clusters = kmeans.fit\_predict(x)

data\_with\_clusters = data.copy()

data\_with\_clusters['Cluster'] = identified\_clusters

plt.scatter(data['Longitude'], data['Latitude'], c=data\_with\_clusters['Cluster'], cmap = 'rainbow')

plt.xlim(-180,180)

plt.ylim(-90, 90)

plt.show()

kmeans = KMeans(3)

kmeans.fit(x)

identified\_clusters = kmeans.fit\_predict(x)

data\_with\_clusters = data.copy()

data\_with\_clusters['Cluster'] = identified\_clusters

plt.scatter(data['Longitude'], data['Latitude'], c=data\_with\_clusters['Cluster'], cmap = 'rainbow')

plt.xlim(-180,180)

plt.ylim(-90, 90)

plt.show()