K-Means Clustering

The goal of clustering is to maximize the similarity of observations within a cluster and to maximize the dissimilarity between clusters. Some of the examples are:

- Market segmentation
- Image segmentation
- Object recognition

Regression uses supervised learning (has labelled data, inputs, correct values for outputs. It uses input to make a model, predict output and uses correct values of output to get accuracy of the model). Cluster analysis is unsupervised learning – the output that we get must be understood and labelled by us.

So classification (regression analysis) is about predicting an output category, given input data whereas clustering is about grouping data points together based on the similarity among them and difference from others.

In K – mean clustering K stands for the number of clusters we need. Then specify the seeds (The initial centroid). Distance of each point is calculated from these seeds. The point closest to a seed is assigned to that cluster. This step is repeated till we no longer have any improvement.

Pros of using K-mean Clustering:

- *Simple to understand*
- *Fast to cluster and easy to implement*
- *Widely available and always gives a result*

Cons of using K-mean Clustering and some remedies

- *We need to pick K (Use Elbow method)*
- *Sensitive to initialization (Use k-means ++ as initial seeds)*
- *Sensitive to outliers (Remove outliers)*
- *Produces spherical clusters*
- *Standardization*

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from sklearn.cluster import KMeans
```

data= pd.read_csv('File name') (*Import the file Country_clusters.csv and store it in the dataframe named data*)

plt.scatter(data['Longitude'],data['Latitude']) (*From the dataframe data plot a scatter plot between the data under the column name Longitude and column name latitude*)

```
plt.xlim(-180,180) (* set the x axis limit from -180 to 180 as that is the range of the longitude*)
plt.xlim(-90,90)  (* Similarly set the y axis limit from -90 to 90 as that is the range of the latitude*)
plt.show
```

Question: Can you recognize some clusters? On what features can these clusters be made?

```
x=data.iloc[:,1:3]  (*Extracts the data from all rows but columns 1 and 2*)
kmeans=KMeans(2) (*Dividing the data using KMeans into 2 clusters*)
kmeans.fit(x) (*Fit the data stored in the variable x to the KMeans model with 2 clusters*)
identified_clusters=kmeans.fit_predict(x) (*stores an array of cluster identifiers, in this case 2, in the variable named identified_clusters*)
```

In order to view the results properly, following commands may be used

```
data_with_clusters=data.copy()
data_with_clusters[‘Clusters]=identified_clusters  (*Adds another column with Clustre identifiers*)
data_with_clusters
plt.scatter(data_with_clusters[‘Longitude’],data_with_clusters[‘Latitude’],c=data_with_clusters[‘Cluster’],cmap=’rainbow’)  (*Sets as many colours as the number of clusters from the rainbow colours*)
plt.xlim(-180,180)
plt.xlim(-90,90)
plt.show
```

Question: Can you explain the formation of clusters? Experiment with different number of clusters in each case defend the number of clusters taken.

Using Categorical data to make clusters

```
data_mapped=data.copy()
data_mapped[‘continent’]=data_mapped[‘continent’].map({‘Africa’:0,’Antartica’:1,’Asia’:2,’Europe’:3,’North America’:4,’Oceania’:5,’South America’:6,’Seven seas’:7})
x=data_mapped.iloc[:,3:4]
```

Now run the same code but with 8 clusters

```
kmeans=KMeans(8)
kmeans.fit(x)
identified_clusters=kmeans.fit_predict(x)
data_with_clusters=data.copy()
data_with_clusters[‘Clusters]=identified_clusters
plt.scatter(data_with_clusters[‘Longitude’],data_with_clusters[‘Latitude’],c=data_with_clusters[‘Cluster’],cmap=’rainbow’)
plt.xlim(-180,180)
plt.xlim(-90,90)
plt.show
```

In order to calculate the number of clusters we try to find WCSS (within cluster sum squares) such that it is low.

```
wcss=[] (*creates an empty list*)

for i in range(1,241):  (*A loop is run for the number of obsevations*)
kmeans=KMeans(i)
kmeans.fit(x) (*Recall x=data_mapped.iloc[:,3:4]*)
wcss_iner=kmeans.inertia_
wcss_append(wcss_iter)
wcss (*You should get a decreasing array of numbers*)
```

We plot these values

```
number_clusters=range(1,241)
plt.plot(number_clusters,wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
```

Question: Can you find the optimal number of Clusters?

Sometimes one needs to standardize the data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from sklearn.cluster import KMeans

data= pd.read_csv('File name')
```

(*In the file Clustering, you will notice that the range of the parameter satisfaction is from 0 to 10 where as the loyalty factor ranges from -2.5 to 2.5. *)

```
x=data.copy()
kmeans = KMeans(2) (*You may try with different number of clusters and try to analyse the results*)
kmeans.fit(x)
clusters=x.copy()
clusters['Cluster Predict']=kmeans.fit_predic(x)
```

```
plt.scatter(clusters[‘Satisfaction’],clusters[‘Loyalty’],c=clusters[‘Cluster
Predict’],cmap=’rainbow’)
plt.xlabel(‘Satisfaction’)
plt.ylabel(‘Loyalty’)
```

(*In case we do not standardize these parameters the weight of the satisfaction parameter will be higher and the clustering will tilt towards the satisfaction parameter*)

```
from sklearn import preprocessing
x_scaled=preprcessing_scale(x)  (*This standardize each variable so that its mean is 0 and
variance 1*)
```

Exercise: Use the Elbow method used earlier to find the optimal number of clusters and analyse your results