Practical-11

Objective:

Implement k Means for Machine Learning Application.

Description:

- We are given a data set of items, with certain features, and values for these features (like a vector). The task is to categorise those items into groups. To achieve this, we will use the kMeans algorithm, an unsupervised learning algorithm
- The algorithm works as follows:
    1. First we initialise k points, called means, randomly.
    2. We categorise each item to its closest mean and we update the mean's coordinates, which are the averages of the items categorised in that mean so far
    3. We repeat the process for a given number of iterations and at the end, we have our clusters
    4. K means is one of the most popular Unsupervised Machine Learning Algorithms Used for Solving Classification Problems. K Means segregates the unlabelled data into various groups, called clusters, based on having similar features, common patterns.

Steps:

The working of the K-Means algorithm is explained in the below steps:

1. Select the value of K, to decide the number of clusters to be formed.
2. Select random K points which will act as centroids.
3. Assign each data point, based on their distance from the randomly selected points (Centroid), to the nearest/closest centroid which will form the predefined clusters.
4. Place a new centroid of each cluster. - Repeat step no.3, which reassign each
5. If any reassignment occurs, then go to step-4 else go to Step 7.
6. FINISH

Implementation & Output:

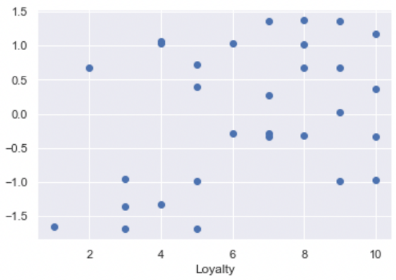
In [1]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from sklearn.cluster import KMeans
```

In [2]:
```
data = pd.read_csv('KMeansDataset.csv')
data.head()
```

Out[2]:

	Satisfaction	Loyalty
0	4	-1.33
1	6	-0.28
2	5	-0.99
3	7	-0.29
4	4	1.06

In [3]:
```
plt.scatter(data['Satisfaction'], data['Loyalty'])
plt.xlabel('Satisfaction')
plt.xlabel('Loyalty')
plt.show()
```



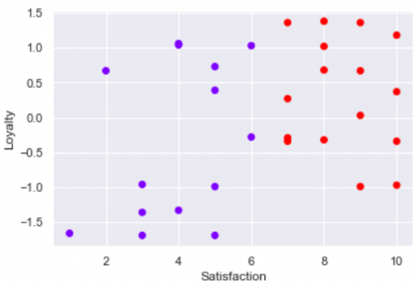
In [4]:
```
#Selecting Features
x=data.copy()
```

In [5]:
```
#Clustering
kmeans=KMeans(2)
kmeans.fit(x)
```

Out[5]: KMeans(n_clusters=2)

In [6]:
```
#Clustering Result
clusters=x.copy()
clusters['cluster_pred']=kmeans.fit_predict(x)
```

In [7]:
```
#Plot
plt.scatter(clusters['Satisfaction'], clusters['Loyalty'],c=clusters['cluster_pred'],cmap='rainbow')
plt.xlabel('Satisfaction')
plt.ylabel('Loyalty')
plt.show()
```



In [8]:
```
#Standardizing the variable
from sklearn import preprocessing
x_scaled=preprocessing.scale(x)
x_scaled
```

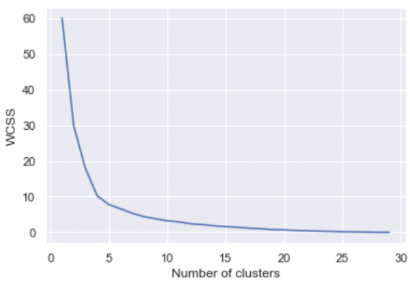
Out[8]: array([[-0.93138063, -1.3318111],
       [-0.15523011, -0.28117124],
       [-0.54330537, -0.99160391],
       [0.23284516, -0.29117733],
       [-0.93138063, 1.05964534],
       [-2.09560642, -1.6620122],
       [1.39707095, -0.97159172],
       [0.62092042, -0.32119561],
       [0.62092042, 1.01962097],
       [0.62092042, 0.67941378],
       [1.39707095, -0.3412078],
       [-0.54330537, 0.38923705],
       [-0.54330537, -1.69203048],
       [-1.70753116, 0.66940768],
       [0.23284516, 0.26916393],
       [1.00899568, 1.35982816],
       [0.62092042, 1.37984035],
       [0.23284516, 1.35982816],
       [0.23284516, -0.3412078],
       [1.00899568, 0.66940768],
       [1.39707095, 1.17971847],
       [-1.31945589, -1.69203048],
       [-0.93138063, 1.03963316],
       [-1.31945589, -0.96158562],
       [-0.15523011, 1.02962706],
       [1.00899568, -0.99160391],
       [1.39707095, 0.36922486],
       [1.00899568, 0.02901767],
       [-1.31945589, -1.36182938],
       [-0.54330537, 0.72944425]])

In [9]:
```
#Elbow Method
wcss=[]
for i in range(1,30):
   kmeans=KMeans(i)
   kmeans.fit(x_scaled)
   wcss.append(kmeans.inertia_)

wcss
```

Out[9]: [59.999999999999986,
 29.818973034723143,
 17.913349527387968,
 10.247181805928422,
 7.792695153937187,
 6.569489487091783,
 5.348079410290981,
 4.395247193896115,
 3.7799886162052667,
 3.2503144612222012,
 2.9080369240790245,
 2.3969501211705038,
 2.145058238505448,
 1.8156574192323445,
 1.6198133783661601,
 1.395897265180343,
 1.1645532493533495,
 1.0151995950549708,
 0.7689799439090226,
 0.6764410827034856,
 0.5146512302075544,
 0.42313027513905704,
 0.32271198172750115,
 0.2472105330779867,
 0.17170908442847233,
 0.11383861748989679,
 0.0559681505513213,
 0.0014517677692203244,
 0.00020024383023728806]

In [10]:
```
#Visualizing the Elbow Method
plt.plot(range(1,30),wcss)
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [14]:
```
#Strong clutering
kmeans_new=KMeans(4)
kmeans.fit(x_scaled)
cluster_new=x.copy()
cluster_new['cluster_pred']=kmeans_new.fit_predict(x_scaled)
cluster_new
```

Out[14]:

	Satisfaction	Loyalty	cluster_pred
0	4	-1.33	2
1	6	-0.28	0
2	5	-0.99	2
3	7	-0.29	0
4	4	1.06	3
5	1	-1.66	2
6	10	-0.97	0
7	8	-0.32	0
8	8	1.02	1
9	8	0.68	1
10	10	-0.34	0
11	5	0.39	3
12	5	-1.69	2
13	2	0.67	3
14	7	0.27	0

In [12]:

```
#Plotting the newly cluster
plt.scatter(cluster_new['Satisfaction'],cluster_new['Loyalty'],c=cluster_new['cluster_pred'],cmap='rainbow')
plt.xlabel('Satisfaction')
plt.ylabel('Loyalty')
plt.show()
```