Objective : The objective of this TP is to explore the data using Python and build cluster using clustering techniques, in this practical we will focus on *k-means* and classification algorithms. In order to do so follow the steps below **:**

1) Running the Jupyter Notebook

- Files → Desktop → New → Python 3 → Folder
- Select directory → Rename Folder (change the name of the folder)
- Select the folder and create a python file.

2) Importing relevant libraries:

First we will import certain libraries required for performing K-means Clustering. We will also look into these packages in details.

1. **Numpy**: It is a third party package that helps us to deal with multidimensional arrays.
2. **Pandas**: Allows us to organize data in tabular form.
3. **Matplotlib**: Helps in visualizing the numpy computation.
4. **Seaborn**: This also adds to the visualization of Matplotlib
5. **Scikit-learn**: This is the most widely used machine learning library. It has various functionality as in this case we are importing KMeans from it.

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

3) Reading the data:

In this step we will load the data set into the variable data using pandas data frame. There are thirty observations with features satisfaction and Loyalty .The data here is in .csv format. Remember pandas helps us to organize data in tabular form.

Reading the data

```
In [2]: data=pd.read_csv('3.12. Example.csv')
        data
```

4) Plotting the data:

Now we will simply plot the scatter plot of the given data using. These are simple python code we will get accustomed to it once we start using it regularly

Plot data

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

5) Clustering:

For the first section in Selecting Feature just ignore the title for now we will see it later. We are just creating a copy of our data and storing it in variable x.
So now we will create a variable kmeans and by passing the argument 2 in KMeans we just said that we want to create 2 clusters. We don't have to do anything this simple line of code will perform the kmeans algorithm and will create two clusters. It will classify our data into two clusters.
You can see below the number of iterations that has been perfomed, here 300.
This all runs behind the scenes. Python will take care of everything.

Selecting Features

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

Clustering

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

```
Out[5]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
            n_clusters=2, n_init=10, n_jobs=1, precompute_distances='auto',
            random_state=None, tol=0.0001, verbose=0)
```

6) Clustering Result:

In this step we will again create a copy of x and store it in clusters. We will create a new column called cluster_predict which will have the value as predicted by our kmeans algorithm. Things will become more clear as we move ahead

Clustering result

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

7) Plot:

Now we will plot the clustered data, note here we have two parameters/features here 'Satisfaction' and

'Loyalty'. We can easily see the two clusters the one with all the red and the other with all the blues. But there is a problem.

```
In [7]: plt.scatter(clusters['Satisfaction'],clusters['Loyalty'],c=clusters['cluster_pred'],cmap='rainbow')
        plt.xlabel('Satisfaction')
        plt.ylabel('Loyalty')
        plt.show() #what this does is it takes the featurse as Satisfaction because it is large and classify accordingly
```

The Problem

The biggest problem here is that Satisfaction is choosen as a feature and loyalty has been neglected. We can see in the figure that all the element to the right of 6 forms one cluster and the other on the left forms another. This is a bias result because our algorithm has discarded the Loyalty feature. It has done the clustering only on the basis of satisfaction. This does not give an appropriate result through which we can analyze things.
Satisfaction was choosen as the feature because it had large values.
So here is the problem both the data are not scaled. First we have to standardize the data, so that both the data have equal weights in our clustering.
We can't neglect loyalty as it has an important role in the analyses of market segmentation

8) Standardizing the variables:

We will not go in depth of this as sklearn helps us to scale the data. The data is scaled around zero mean. Now we can see that both the data are equally scaled and now both will have equal chance of being selected as feature.

9) The Elbow Method:

Have we ever wondered why we initialized kmeans with 2 clusters only.
Yes, we could have initialized it with any value we wanted we could have got any number of clusters. But the analyses becomes difficult when there are a large number of clusters. So how we will know the exact number of cluster to start off. Note there are no such exact number as it changes with the problem in hand.
Here the elbow method comes handy when we are confused as to how may clusters do we need.
What elbow method does is it starts of with making one cluster to the number of clusters in our sample and with the kmeans inertia value we determine what would be the appropriate number of clusters.
Remember our goal- Our final goal was to minimize the within the cluster sum of square and maximize the distance between clusters.
With this simple line of code we get all the inertia value or the within the cluster sum of square

Take advantage of the elbow method

```
In [10]: wcss=[]

         for i in range(1,30):
             kmeans=KMeans(i)
             kmeans.fit(x_scaled)
             wcss.append(kmeans.inertia_)
         wcss
```

10) Visualizing the Elbow Method:

This graph looks like elbow and we have to determine that elbow point.
Here the elbow point comes at around 4 and this our optimal number of clusters for the above data which we should choose.
If we look at the figure carefully after 4 when we go on increasing the number of cluster there is no big change in the wcss and it remains constant.
Hurrah..!! we have got the optimal number of clusters for our problem.
We will now quickly perform the kmeans clustering with the new number of clusters which is 4 and then dive into some analysis.

```
In [11]: plt.plot(range(1,30),wcss)
         plt.xlabel('Number of clusters')
         plt.ylabel('WCSS')
         plt.show()
```

11) Stronger Clustering:

This is a simple code which perform clustering with 4 clusters. Here there are four clusters so our whole data is categorized into either 0,1,2 or 3.

```
In [12]: kmeans_new=KMeans(4)
         kmeans.fit(x_scaled)
         cluster_new=x.copy()
         cluster_new['cluster_pred']=kmeans_new.fit_predict(x_scaled)
         cluster_new
```

12) Plotting the newly cluster:

```
In [13]: plt.scatter(cluster_new['Satisfaction'],cluster_new['Loyalty'],c=cluster_new['cluster_pred'],cmap='rainbow')
         plt.xlabel('Satisfaction')
         plt.ylabel('Loyalty')
         plt.show()
```

13) Analysis (The final step):

Through the given figure following things can be interpreted:
1. The purple dots are the people who are less satisfied and less loyal and therefore can be termed as

alienated.
2. The red dots are people with high loyalty and less satisfaction.
3. The yellow dots are the people with high loyalty and high satisfaction and they are the fans.
4. The sky blue dots are the people who are in the midst of things.
The ultimate goal of any businessman would be to have as many people up there in the fans category. We are ready with a solution and we can target the audience as per our analysis. For example, the crowd who are supporters can easily be turned into fans by fulfilling their satisfaction level

In [13]:
```
plt.scatter(cluster_new['Satisfaction'],cluster_new['Loyalty'],c=cluster_new['cluster_pred'],cmap='rainbow')
plt.xlabel('Satisfaction')
plt.ylabel('Loyalty')
plt.show()
```