K-means clustering is one of the simplest unsupervised machine learning algorithms. Here, we'll explore what it can do and work through a simple implementation in Python.

- Some facts about k-means clustering:
- K-means converges in a finite number of iterations. Since the algorithm iterates a function whose domain is a finite set, the iteration must eventually converge.
- The computational cost of the k-means algorithm is O(k*n*d), where n is the number of data points, k the number of clusters, and d the number of attributes.
- Compared to other clustering methods, the k-means clustering technique is fast and efficient in terms of its computational cost.
- It's difficult to predict the optimal number of clusters or the value of k. To find the number of clusters, we need to run the k-means clustering algorithm for a range of k values and compare the results.

In [19]:
```
## import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

In [20]:
```
## import data set
df=pd.read_csv("iris.csv")
```

In [21]:
```
df.head(10)
```

Out[21]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa
5	5.4	3.9	1.7	0.4	Setosa
6	4.6	3.4	1.4	0.3	Setosa
7	5.0	3.4	1.5	0.2	Setosa
8	4.4	2.9	1.4	0.2	Setosa
9	4.9	3.1	1.5	0.1	Setosa

In [22]:
```
x=df.iloc[:,[0,1,2,3]].values
```

k-means clustering model with k=5

In [30]:
```
kmeans=KMeans(n_clusters=5)
y_kmeans=kmeans.fit_predict(x)
```

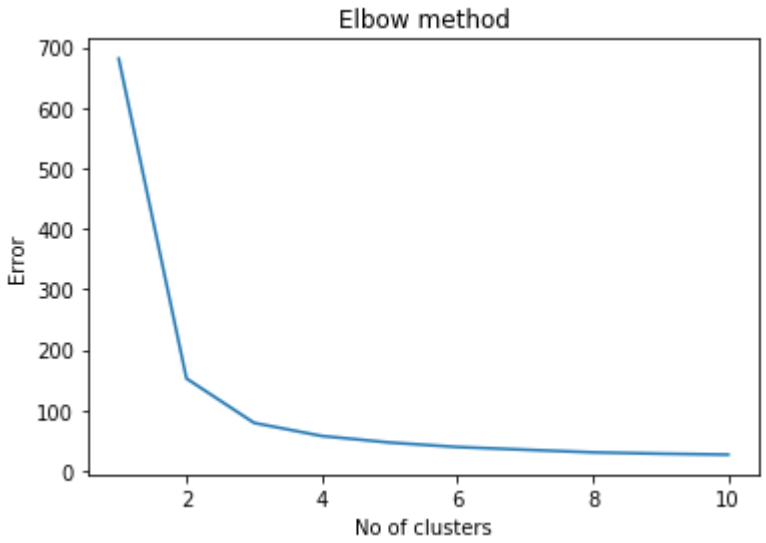
In [24]:
```
print(y_kmeans)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 4 3 4 4 4 3 4 3 3 4 3 4 3 4 4 3 4 3 4 3 4 4
 4 4 4 4 4 3 3 3 3 4 3 4 4 4 3 3 3 4 3 3 3 3 4 3 3 1 4 2 1 1 2 3 2 1 2 1
 1 1 4 1 1 1 2 2 4 1 4 2 4 1 2 4 4 1 2 2 2 1 4 4 2 1 1 4 1 1 1 4 1 1 1 4 1
 1 4]
```

In [25]:
```
kmeans.cluster_centers_
```

Out[25]:
```
array([[5.006     , 3.428     , 1.462     , 0.246     ],
       [6.52916667, 3.05833333, 5.50833333, 2.1625    ],
       [7.475     , 3.125     , 6.3       , 2.05      ],
       [5.508     , 2.6       , 3.908     , 1.204     ],
       [6.20769231, 2.85384615, 4.74615385, 1.56410256]])
```

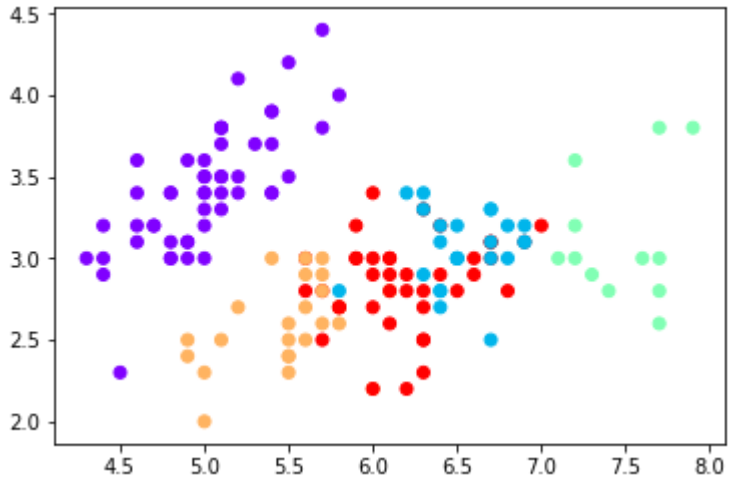
In [26]:
```
Error =[]
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i).fit(x)
    kmeans.fit(x)
    Error.append(kmeans.inertia_)
import matplotlib.pyplot as plt
plt.plot(range(1, 11), Error)
plt.title('Elbow method')
plt.xlabel('No of clusters')
plt.ylabel('Error')
plt.show()
```



Visulazing the kmeans clustering

In [27]:
```
plt.scatter(x[:,0],x[:,1],c=y_kmeans,cmap='rainbow')
```

Out[27]:
```
<matplotlib.collections.PathCollection at 0xfc889ad860>
```



k-means clustering model with k=3

In [28]:
```
kmeans=KMeans(n_clusters=3)
y_kmeans=kmeans.fit_predict(x)
print(y_kmeans)
kmeans.cluster_centers_
```

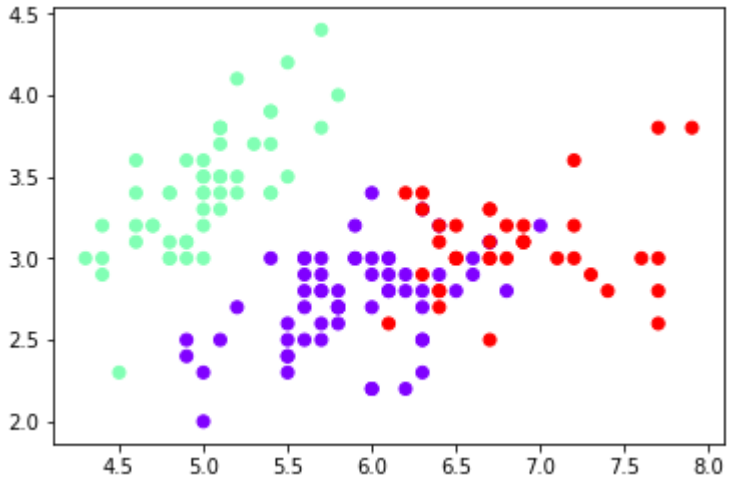
```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2
 2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 0 2
 2 0]
```

Out[28]:
```
array([[5.9016129 , 2.7483871 , 4.39354839, 1.43387097],
       [5.006     , 3.428     , 1.462     , 0.246     ],
       [6.85      , 3.07368421, 5.74210526, 2.07105263]])
```

Visulazing the kmeans clustering

In [29]:
```
plt.scatter(x[:,0],x[:,1],c=y_kmeans,cmap='rainbow')
```

Out[29]:
```
<matplotlib.collections.PathCollection at 0xfc88a01ef0>
```



In [ ]: