

Task-2: To Explore Unsupervised Machine Learning

From the given 'Iris' dataset, predict the optimum number of clusters and represent it visually.

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

```
In [4]: data = pd.read_csv(r'C:\Users\USER\Downloads\Iris.csv')
data.head()
```

```
Out[4]:
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Exploring the Data

```
In [5]: data.shape
```

```
Out[5]: (150, 6)
```

```
In [6]: data.columns
```

```
Out[6]: Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
              'Species'],
              dtype='object')
```

```
In [7]: data.info
```

```
Out[7]: <bound method DataFrame.info of      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0      1             5.1           3.5           1.4           0.2
1      2             4.9           3.0           1.4           0.2
2      3             4.7           3.2           1.3           0.2
3      4             4.6           3.1           1.5           0.2
4      5             5.0           3.6           1.4           0.2
..    ...             ...           ...           ...           ...
145   146            6.7           3.0           5.2           2.3
146   147            6.3           2.5           5.0           1.9
147   148            6.5           3.0           5.2           2.0
148   149            6.2           3.4           5.4           2.3
149   150            5.9           3.0           5.1           1.8

      Species
0      Iris-setosa
1      Iris-setosa
2      Iris-setosa
3      Iris-setosa
4      Iris-setosa
..    ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]>
```

```
In [8]: data.describe
```

```
Out[8]: <bound method NDFrame.describe of      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0      1             5.1           3.5           1.4           0.2
1      2             4.9           3.0           1.4           0.2
2      3             4.7           3.2           1.3           0.2
3      4             4.6           3.1           1.5           0.2
4      5             5.0           3.6           1.4           0.2
..    ...             ...           ...           ...           ...
145   146            6.7           3.0           5.2           2.3
146   147            6.3           2.5           5.0           1.9
147   148            6.5           3.0           5.2           2.0
148   149            6.2           3.4           5.4           2.3
149   150            5.9           3.0           5.1           1.8

      Species
0      Iris-setosa
1      Iris-setosa
2      Iris-setosa
3      Iris-setosa
4      Iris-setosa
..    ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]>
```

```
In [9]: data.Species.unique()
```

```
Out[9]: array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

Finding the optimum number of clusters

```
In [10]: X = data.iloc[:, [1,2,3,4]].values
```

```
In [11]: from sklearn.cluster import KMeans
```

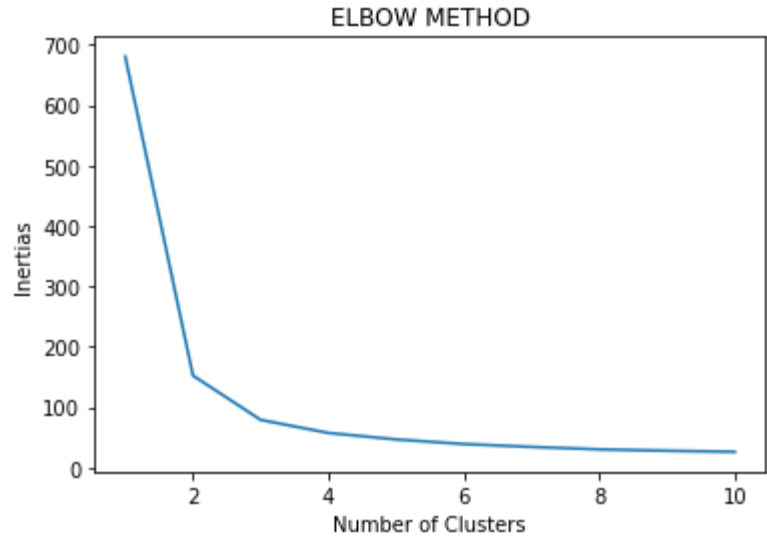
```
In [12]: def elbowMethod(num_clusters, inertias):
plt.plot(num_clusters, inertias)
plt.title("ELBOW METHOD")
plt.xlabel("Number of Clusters")
plt.ylabel("Inertias")
plt.show()
```

```
In [13]: inertias = []
clusters = range(1,11)

for i in clusters:
    kmeans = KMeans(n_clusters = i, init='k-means++', max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(X)
    inertias.append(kmeans.inertia_)
elbowMethod(clusters, inertias)
```

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\cluster_kmeans.py:881: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.

warnings.warn(



```
In [14]: kmeans = KMeans(n_clusters = 3, init='k-means++', max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(X)
```

```
In [15]: kmeans.cluster_centers_
```

```
Out[15]: array([[5.9016129 , 2.7483871 , 4.39354839, 1.43387097],
               [5.006      , 3.418      , 1.464      , 0.244      ],
               [6.85      , 3.07368421, 5.74210526, 2.07105263]])
```

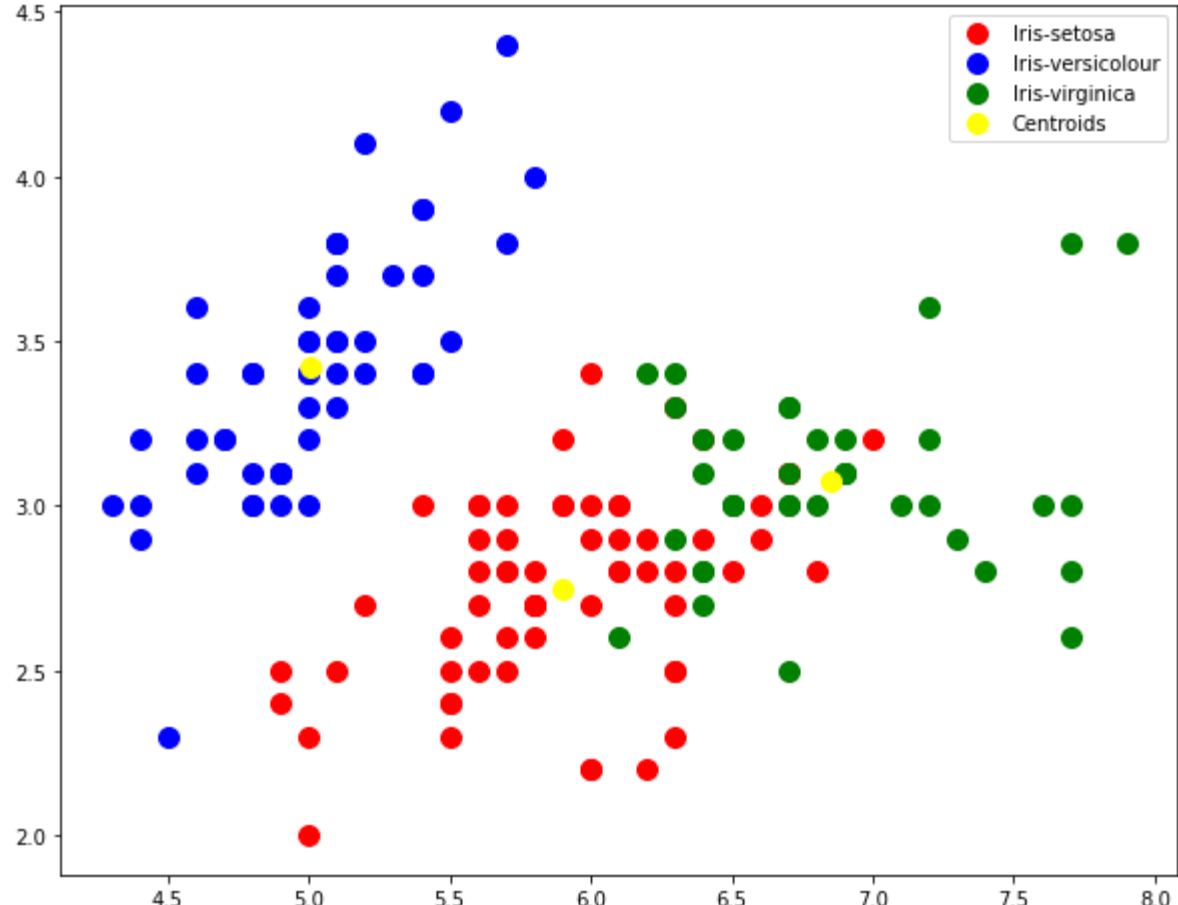
```
In [16]: plt.figure(figsize = (10,8))

# Visualising the clusters - On the first two columns
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1],
            s = 100, c = 'red', label = 'Iris-setosa')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1],
            s = 100, c = 'blue', label = 'Iris-versicolour')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1],
            s = 100, c = 'green', label = 'Iris-virginica')

# Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[0, 0], kmeans.cluster_centers_[0,1],
            s = 100, c = 'yellow', label = 'Centroids')

plt.legend()

plt.show()
```



```
In [ ]:
```