Prachi Lal

Task2: From the given Iris dataset, predict the optimum number of clusters and represent it visually

Importing Necessary Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import metrics
```

Importing Dataset

In [2]:
```
df = pd.read_csv("Iris.csv")
df.head(100)
```

Out[2]:

	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
...	...	...	...	...	...	...
95	96	5.7	3.0	4.2	1.2	Iris-versicolor
96	97	5.7	2.9	4.2	1.3	Iris-versicolor
97	98	6.2	2.9	4.3	1.3	Iris-versicolor
98	99	5.1	2.5	3.0	1.1	Iris-versicolor
99	100	5.7	2.8	4.1	1.3	Iris-versicolor

100 rows × 6 columns

Defining features

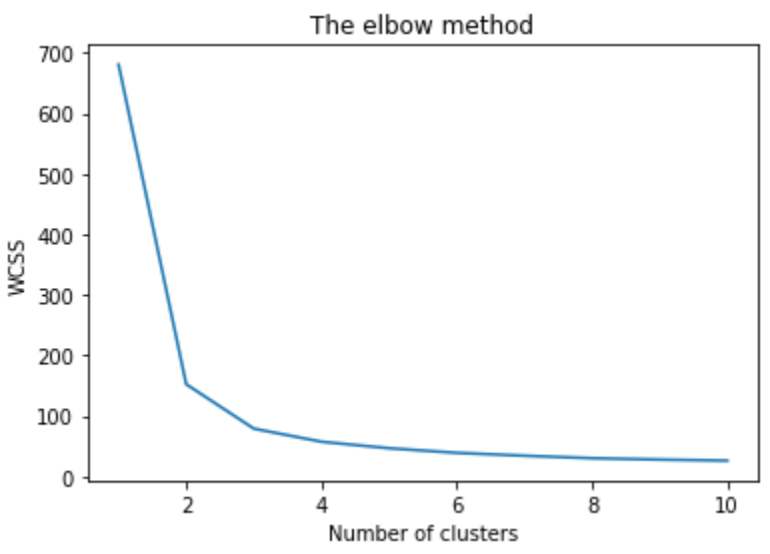
In [3]:
```
x = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm' ]]
x = x.values
```

Determining the number of clusters

Using the "Elbow Plot" method

In [4]:
```
wcss = [] #within cluster squared sum
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',
                    max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
```

In [5]:
```
plt.plot(range(1, 11), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') # Within cluster sum of squares
plt.show()
```



The Plot shows "3" to be an optimum value for the data

Other cluster fitness scoring methods

The Silhouette Coefficient and The Calinski Harbasz Score

Considering K as "3"

In [6]:
```
k_means = KMeans(n_clusters=3)

model = k_means.fit(x)
model

y_hat = k_means.predict(x)

labels = k_means.labels_

print("Sihouette Coefficient: ", metrics.silhouette_score(x, labels, metric = 'euclide
print("Calinski Score: " , metrics.calinski_harabasz_score(x, labels))
```

```
Sihouette Coefficient:  0.5525919445499757
Calinski Score:  560.3999242466402
```

Considering K as "4"

In [7]:
```
k_means = KMeans(n_clusters=4)

model = k_means.fit(x)

y_hat = k_means.predict(x)

labels = k_means.labels_

print("Sihouette Coefficient: ", metrics.silhouette_score(x, labels, metric = 'euclide
print("Calinski Score: " , metrics.calinski_harabasz_score(x, labels))
```

```
Sihouette Coefficient:  0.4972279726640147
Calinski Score:  529.1207190840455
```

Considering k as "2"

In [8]:
```
k_means = KMeans(n_clusters=2)

model = k_means.fit(x)

y_hat = k_means.predict(x)

labels = k_means.labels_

print("Sihouette Coefficient: ", metrics.silhouette_score(x, labels, metric = 'euclide
print("Calinski Score: " , metrics.calinski_harabasz_score(x, labels))
```

```
Sihouette Coefficient:  0.6808136202936816
Calinski Score:  513.3038433517568
```

K=3

Creating the classifier

In [9]:
```
kmeans = KMeans(n_clusters = 3, init = 'k-means++',
                max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(x)
```

In [10]:
```
y_kmeans
```

Out[10]:
```
array([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, 2, 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, 0, 0, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2,
       2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0])
```

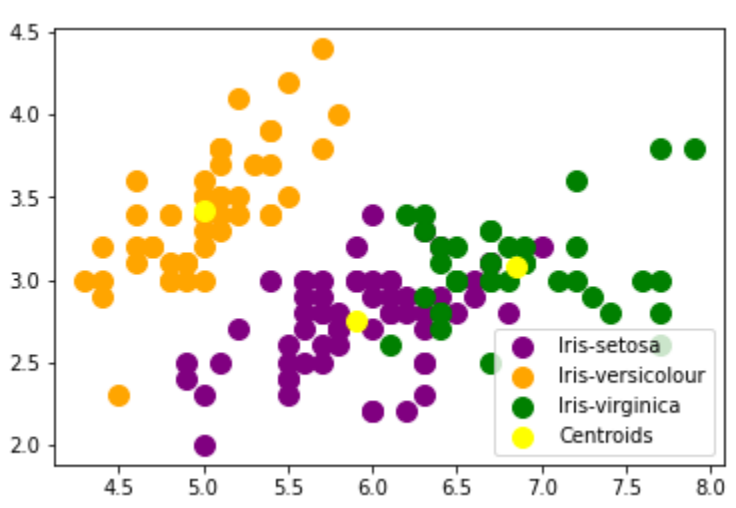
Visualizing Clusters along with the centroids

In [14]:
```
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c = 'purple', label =
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'orange', label =
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c = 'green', label = '

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1],
            s = 100, c = 'yellow', label = 'Centroids')

plt.legend()
```

Out[14]:
```
<matplotlib.legend.Legend at 0x1eccdbee820>
```



In [ ]: