In [1]:
```
To Explore Unsupervised Machine Learning:K-Means Clustering


In this task we will use the iris dataset to predict the optimum number of cluste
```

In [ ]:
```

import numpy as np
import pandas as pd
from sklearn import datasets
import matplotlib.pyplot as plt
```

In [2]:
```

ds=pd.read_csv(r"C:\Users\Akankasha\OneDrive\Desktop\ds\Iris.csv")
```

In [3]:
```
ds.head()
```

Out[3]:

	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

In [4]:
```
ds.shape
```

Out[4]:
```
(150, 6)
```

In [5]:
```
ds.info
```

Out[5]:
```
<bound method DataFrame.info of       Id  SepalLengthCm  SepalWidthCm  PetalLeng
thCm  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 [6]:

```

# Finding the optimum number of clusters for k-means classification
x = ds.iloc[:, [0, 1, 2, 3]].values
from sklearn.cluster import KMeans
wcss = []
```

In [7]:

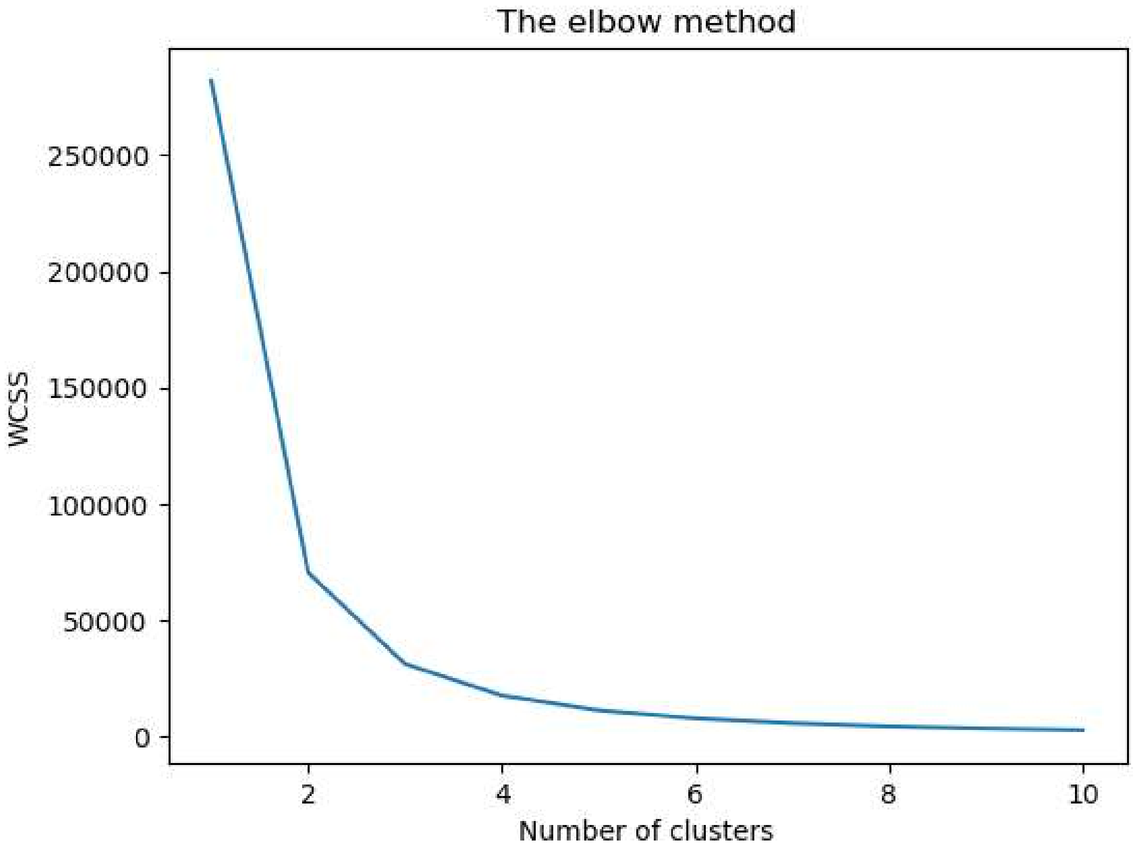
```
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',
                    max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
```

```
C:\Users\Akankasha\anaconda3\Lib\site-packages\joblib\externals\loky\backend\con
text.py:110: UserWarning: Could not find the number of physical cores for the fo
llowing reason:
[WinError 2] The system cannot find the file specified
Returning the number of logical cores instead. You can silence this warning by s
etting LOKY_MAX_CPU_COUNT to the number of cores you want to use.
  warnings.warn(
  File "C:\Users\Akankasha\anaconda3\Lib\site-packages\joblib\externals\loky\bac
kend\context.py", line 199, in _count_physical_cores
    cpu_info = subprocess.run(
               ^^^^^^^^^^^^^^^
  File "C:\Users\Akankasha\anaconda3\Lib\subprocess.py", line 548, in run
    with Popen(*popenargs, **kwargs) as process:
         ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Akankasha\anaconda3\Lib\subprocess.py", line 1026, in __init__
    self._execute_child(args, executable, preexec_fn, close_fds,
  File "C:\Users\Akankasha\anaconda3\Lib\subprocess.py", line 1538, in _execute_
child
    hp, ht, pid, tid = _winapi.CreateProcess(executable, args,
                       ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
C:\Users\Akankasha\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when the
re are less chunks than available threads. You can avoid it by setting the envir
onment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Akankasha\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when the
re are less chunks than available threads. You can avoid it by setting the envir
onment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Akankasha\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when the
re are less chunks than available threads. You can avoid it by setting the envir
onment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Akankasha\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when the
re are less chunks than available threads. You can avoid it by setting the envir
onment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Akankasha\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when the
re are less chunks than available threads. You can avoid it by setting the envir
onment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Akankasha\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when the
re are less chunks than available threads. You can avoid it by setting the envir
onment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Akankasha\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when the
re are less chunks than available threads. You can avoid it by setting the envir
onment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Akankasha\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when the
re are less chunks than available threads. You can avoid it by setting the envir
onment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Akankasha\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when the
re are less chunks than available threads. You can avoid it by setting the envir
```

```
onment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Akankasha\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when the
re are less chunks than available threads. You can avoid it by setting the envir
onment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [8]:

```python
# Plotting the results onto a line graph,
# `allowing us to observe 'The elbow'
plt.plot(range(1, 11), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



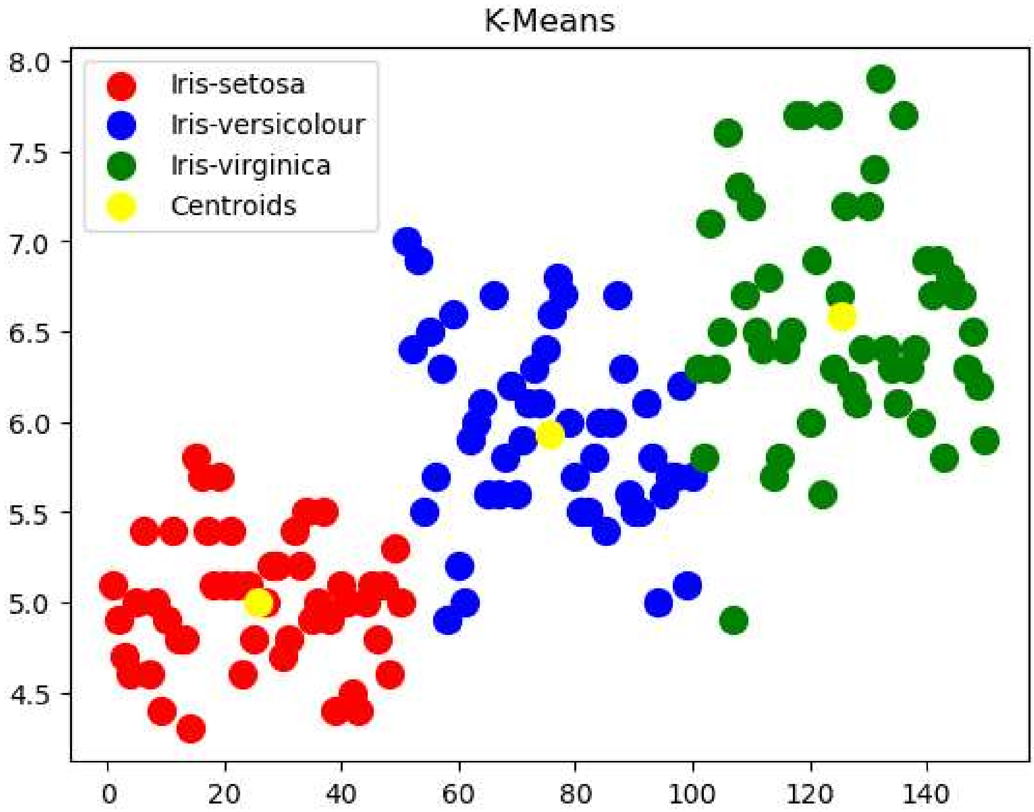
In [9]:

```python
# Applying kmeans to the dataset / Creating the kmeans classifier
kmeans = KMeans(n_clusters = 3, init = 'k-means++',max_iter = 300, n_init = 10, r
y_kmeans = kmeans.fit_predict(x)
```

```
C:\Users\Akankasha\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when the
re are less chunks than available threads. You can avoid it by setting the envir
onment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [10]:
```
# 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 =
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'blue', label
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1],s = 100, c = 'green', label

# Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1],
            s = 100, c = 'yellow', label = 'Centroids')
plt.title("K-Means")
plt.legend()
plt.show()
```



In [ ]:
```
#this concludes the K means Workshop
```