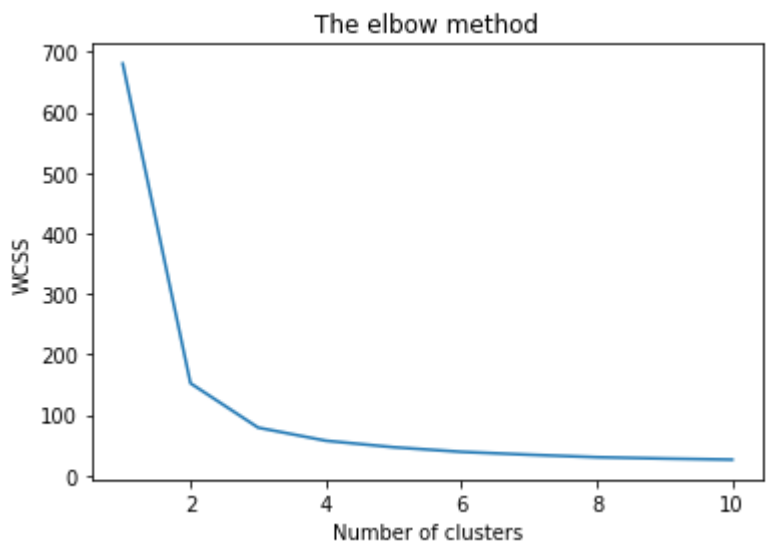
In [1]:
```
# Simple K-means clustering on the Iris dataset
#importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#importing the Iris dataset with pandas
dataset = pd.read_csv('Iris.csv')
x = dataset.iloc[:, [1, 2, 3, 4]].values
```

In [2]:
```
# Now we will implement 'The elbow method' on the Iris dataset.
# The elbow method allows us to pick the optimum amount of clusters for classification.
# although we already know the answer is 3 it is still interesting to run.

#Finding the optimum number of clusters for k-means classification
from sklearn.cluster import KMeans
wcss = []

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_)

#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') #within cluster sum of squares
plt.show()
print('''
We can clearly see why it is called 'The elbow method' from the above graph, the optimum clusters is wh
ere the elbow occurs.
This is when the within cluster sum of squares (WCSS) doesn't decrease significantly with every iterati
on.
Now that we have the optimum amount of clusters, we can move on to applying K-means clustering to the I
ris dataset.
''')
```

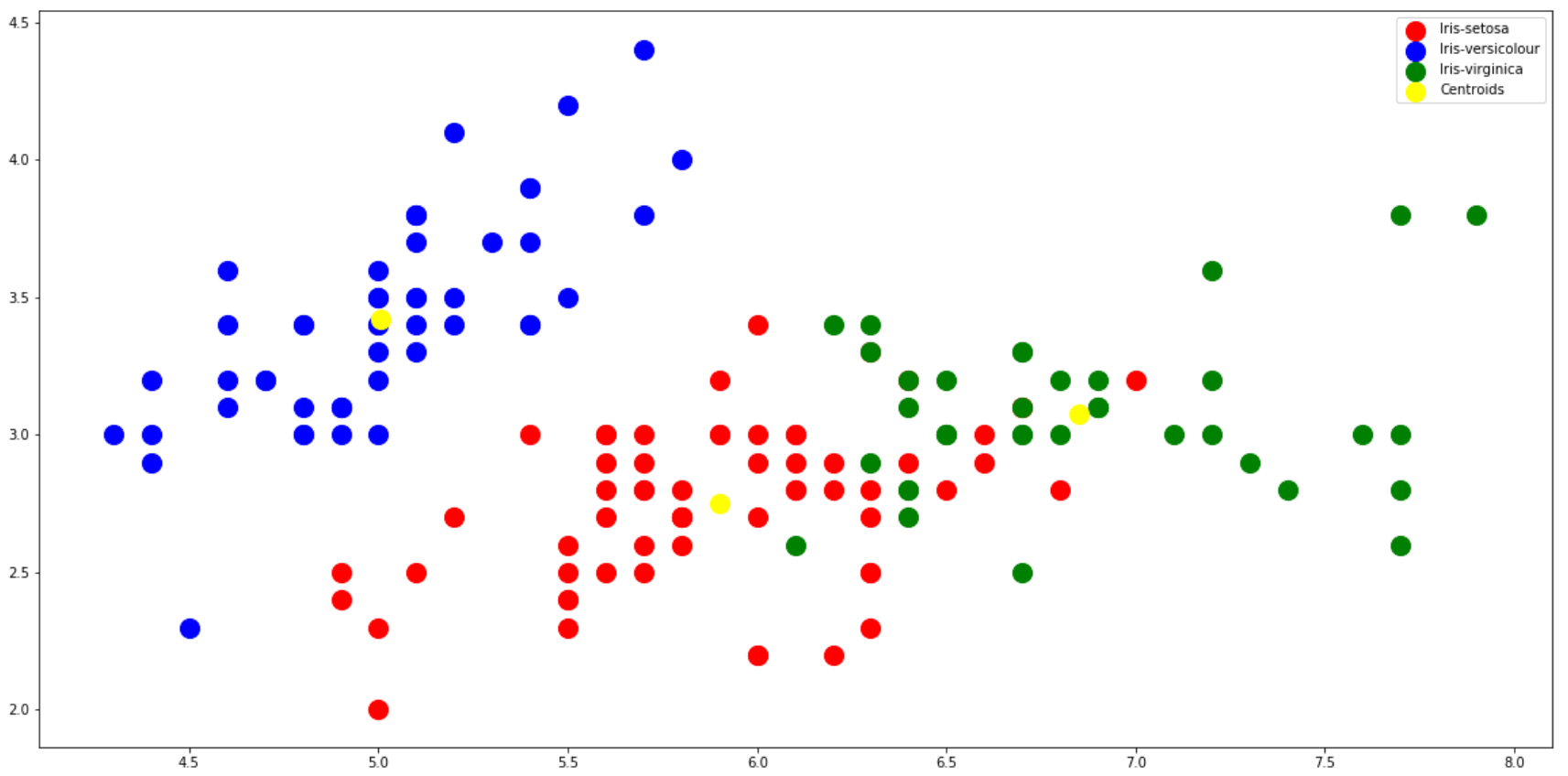



```
We can clearly see why it is called 'The elbow method' from the above graph, the optimum clusters is
where the elbow occurs.
This is when the within cluster sum of squares (WCSS) doesn't decrease significantly with every itera
tion.
Now that we have the optimum amount of clusters, we can move on to applying K-means clustering to the
Iris dataset.
```

In [3]:
```
#Applying kmeans to the dataset / Creating the kmeans classifier
kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(x)
```

In [4]:
```
#Visualising the clusters
plt.figure(figsize=(20, 10))
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 200, c = 'red', label = 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 200, c = 'blue', label = 'Iris-versicolour')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 200, c = 'green', label = 'Iris-virginica')

#Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], s = 200, c = 'Yellow', label =
 'Centroids')

plt.legend()
```

Out[4]:
```
<matplotlib.legend.Legend at 0x2bb003d3948>
```



In [ ]: