Anisha Dhadge¶

Task 2: Prediction using Unsupervised ML¶

Predict the optimum number of clusters and represent it visually¶

Importing libraries¶

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
```

Loading data¶

In [2]:

```
df=pd.read_csv("C:\\Users\\dhadg\\Downloads\\Iris.csv")
```

In [3]:

```
df.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]:

```
df.shape
```

Out[4]:

```
(150, 6)
```

In [5]:

```
df.columns
```

Out[5]:

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

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

Checking for null values¶

In [7]:

```
df.isnull().sum()
```

Out[7]:

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
```

```
dtype: int64
```

As we can see there are no null values.

In [8]:

```
df.duplicated()
```

Out[8]:

```
0      False
1      False
2      False
3      False
4      False
       ...  
145    False
146    False
147    False
148    False
149    False
Length: 150, dtype: bool
```

There are no duplicates present in the data.

Removal of unwanted columns¶

In [9]:

```
df.drop(['Id'],axis=1,inplace=True)
```

In [10]:

```
df.head()
```

Out[10]:

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

In [11]:

```
df.describe()
```

Out[11]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

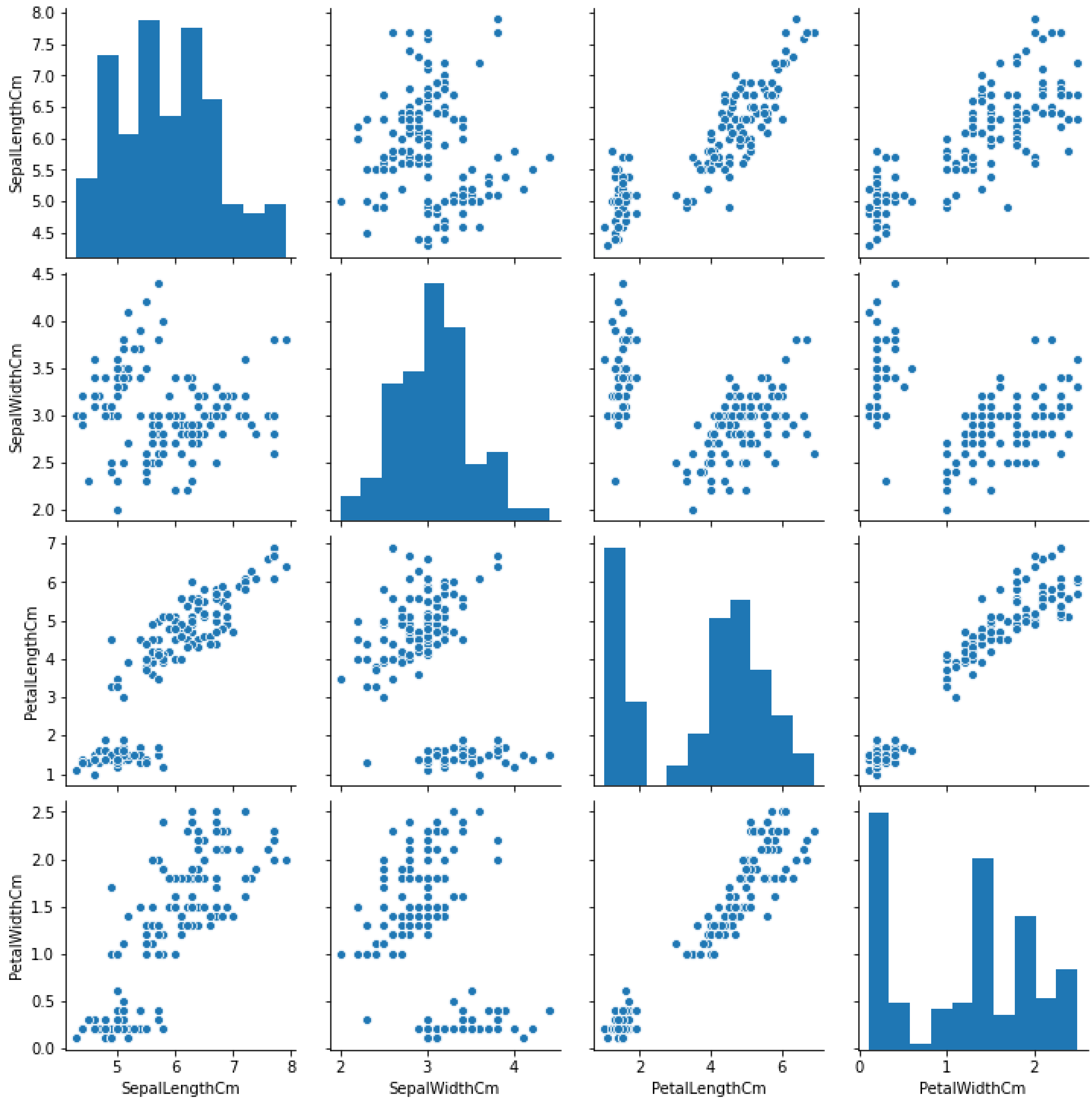
Data Visualization¶

In [12]:

```
sns.pairplot(df)
```

Out[12]:

```
<seaborn.axisgrid.PairGrid at 0x202be671fa0>
```



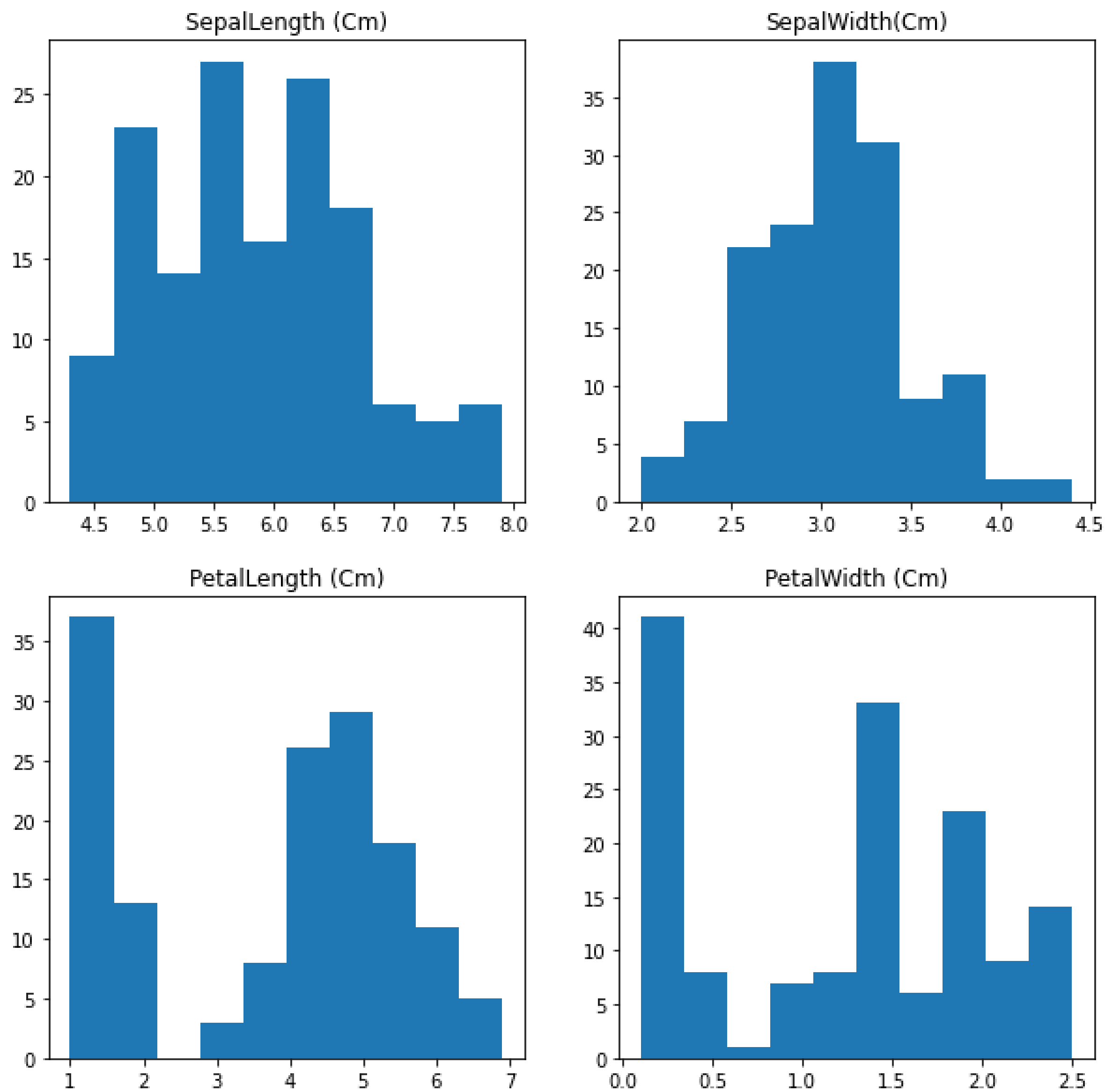
In [13]:

```
fig, ax = plt.subplots(figsize=(10,10))
plt.subplot(2,2,1)
plt.title("SepalLength (Cm)")
plt.hist(df["SepalLengthCm"])
plt.subplot(2,2,2)
plt.title("SepalWidth(Cm)")
plt.hist(df["SepalWidthCm"])
plt.subplot(2,2,3)
plt.title("PetalLength (Cm)")
```

```
plt.hist(df["PetalLengthCm"])
plt.subplot(2,2,4)
plt.title("PetalWidth (Cm)")
plt.hist(df["PetalWidthCm"])
```

Out[13]:

```
(array([41.,  8.,  1.,  7.,  8., 33.,  6., 23.,  9., 14.]),
 array([0.1 , 0.34, 0.58, 0.82, 1.06, 1.3 , 1.54, 1.78, 2.02, 2.26, 2.5 ]),
 <a list of 10 Patch objects>)
```



In [14]:

```
df.corr()
```

Out[14]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954

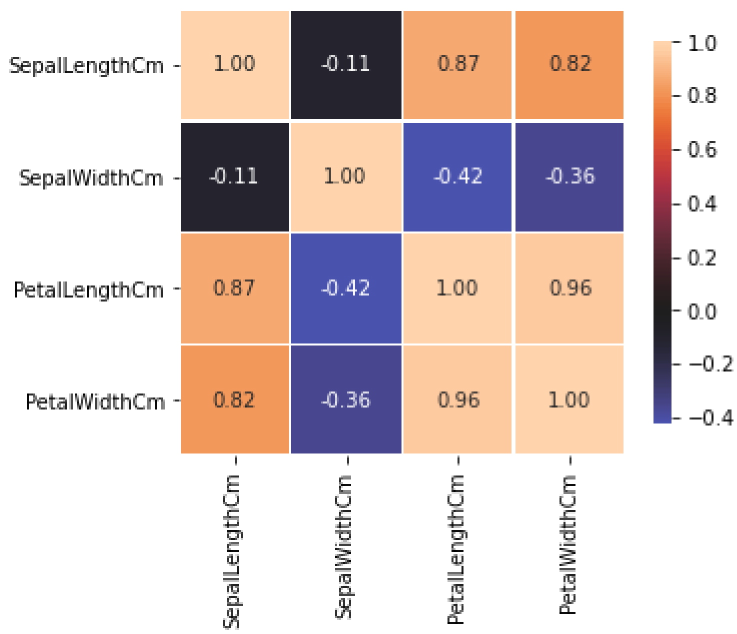
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

In [15]:

```
def correlation_heatmap(df):
    correlations = df.corr()

    fig, ax = plt.subplots(figsize=(5,5))
    sns.heatmap(correlations, vmax=1.0, center=0, fmt='.2f',
                square=True, linewidths=.5, annot=True, cbar_kws={"shrink": .70})
    plt.show();

correlation_heatmap(df)
```



Optimum number of clusters¶

In [16]:

```
x=df.drop(['Species'],axis=1)
```

In [17]:

```
kmeans10=KMeans(n_clusters=10)
ykmeans10=kmeans10.fit_predict(x)
print(ykmeans10)
kmeans10.cluster_centers_
```

```
[4 1 1 1 4 9 1 4 1 1 4 4 1 1 9 9 9 4 9 4 4 4 1 4 4 1 4 4 4 1 1 4 9 9 1 1 4
 1 1 4 4 1 1 4 4 1 4 1 4 4 7 7 7 6 7 2 7 6 7 6 6 2 6 7 6 7 2 2 7 6 3 2 3 7
 7 7 7 8 7 6 6 6 2 3 2 7 7 7 2 6 2 7 2 6 2 2 2 7 6 2 5 3 5 8 5 0 2 0 8 5 8
 8 8 3 3 8 8 0 0 3 5 3 0 3 5 0 3 3 8 0 0 0 8 3 3 0 5 8 3 8 5 8 3 5 5 8 3 8
 8 3]
```

Out[17]:

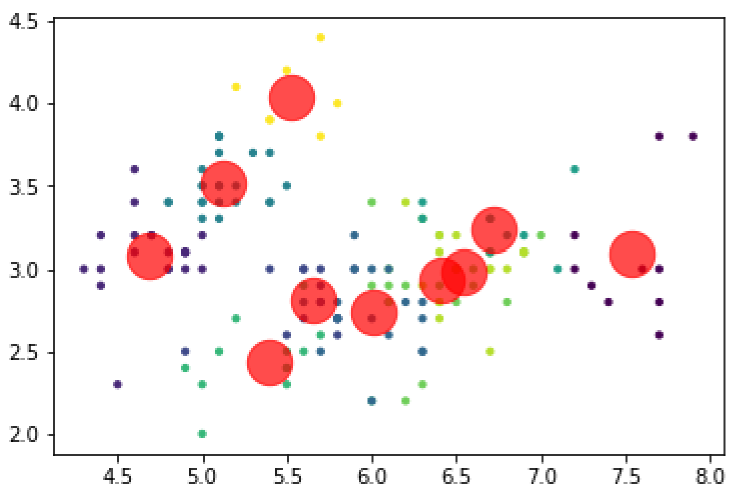
```
array([[7.54      , 3.09      , 6.36      , 2.        ],
       [4.69      , 3.085     , 1.385     , 0.19      ],
       [5.65333333, 2.80666667, 4.22666667, 1.32      ],
       [6.00588235, 2.73529412, 5.01176471, 1.78823529],
       [5.12173913, 3.5173913 , 1.53043478, 0.27826087],
       [6.72      , 3.24      , 5.8       , 2.33      ],
       [5.39230769, 2.43846154, 3.65384615, 1.12307692],
       [6.41578947, 2.93684211, 4.56842105, 1.42105263],
       [6.54375   , 2.9875    , 5.38125   , 2.03125   ],
       [5.52857143, 4.04285714, 1.47142857, 0.28571429]])
```

In [18]:

```
plt.scatter(x.iloc[:,0],x.iloc[:,1],c=ykmeans10,s=10)
centers=kmeans10.cluster_centers_
plt.scatter(centers[:,0],centers[:,1],c="red",s=500,alpha=0.7)
```

Out[18]:

```
<matplotlib.collections.PathCollection at 0x202bfe0fbe0>
```



Here we can see that there are about 3-4 number of clusters.

Elbow Method¶

The K-Elbow Visualizer implements the “elbow” method of selecting the optimal number of clusters for Kmeans clustering.The elbow method runs k-means clustering on the dataset for a range of values for k (say from 1-10) and then for each value of k computes an average score for all clusters.

In [19]:

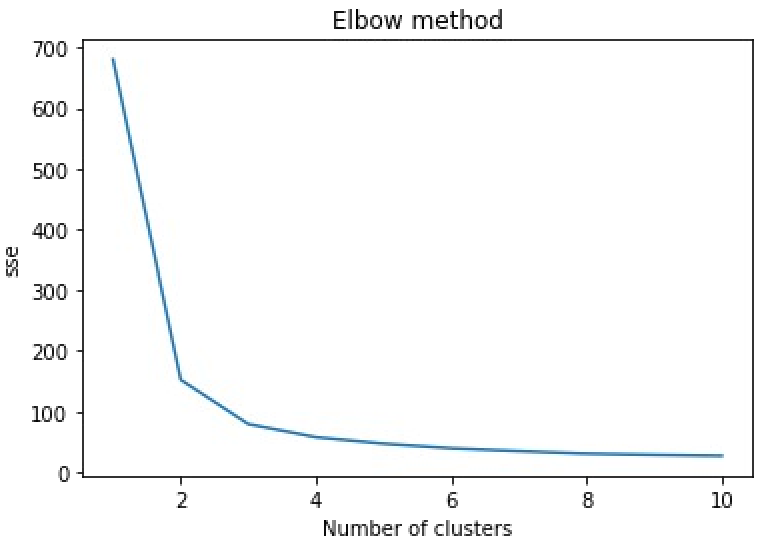
```
sse=[]
for i in range(1,11):
    kmeans=KMeans(n_clusters=i)
    kmeans.fit(x)
    sse.append(kmeans.inertia_)
```

In [20]:

```
plt.plot(range(1,11),sse)
plt.title("Elbow method")
plt.xlabel("Number of clusters")
plt.ylabel("sse")
```

Out[20]:

```
Text(0, 0.5, 'sse')
```



From the above elbow method we can clearly say that the optimum number of clusters is between 3 and 4. As we can say that 3 is a better option over 4. So we take optimum number of clusters as 3.

In [21]:

```
kmeans3=KMeans(n_clusters=3)
ykmeans3=kmeans3.fit_predict(x)
print(ykmeans3)
kmeans3.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 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 2 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 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]
```

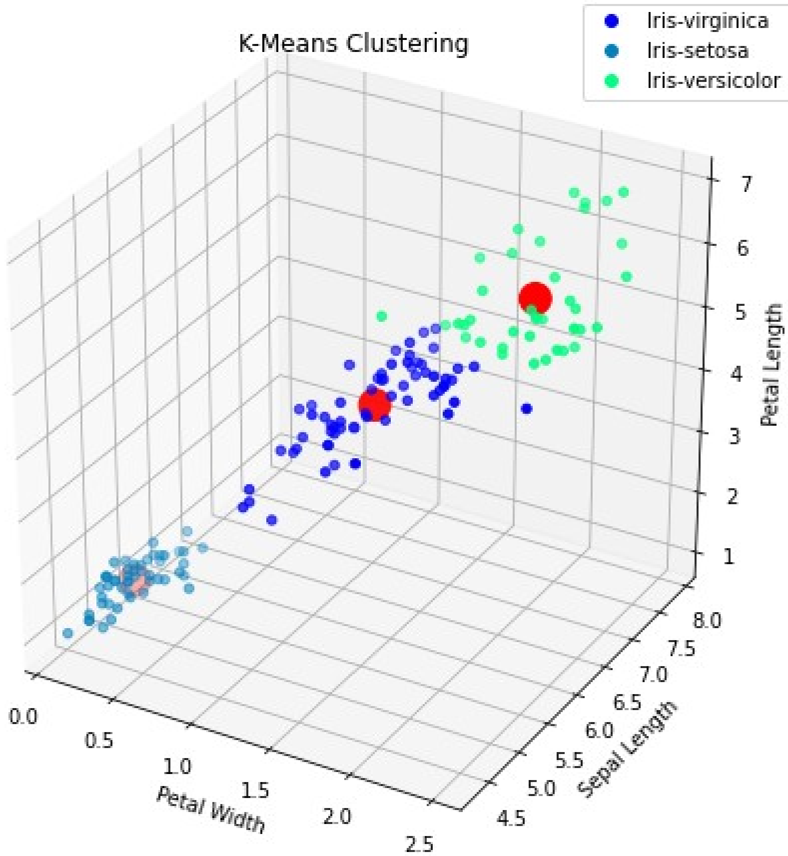
Out[21]:

```
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 [22]:

```
fig = plt.figure(figsize=(8,8))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(kmeans3.cluster_centers_[:, 3],kmeans3.cluster_centers_[:,
0],kmeans3.cluster_centers_[:, 2],s= 250,
                     marker='o',c='red',label='centroids')
scatter = ax.scatter(df['PetalWidthCm'],df['SepalLengthCm'],
df['PetalLengthCm'],c=ykmeans3,s=20, cmap='winter')

ax.set_title('K-Means Clustering')
ax.set_xlabel('Petal Width')
ax.set_ylabel('Sepal Length')
ax.set_zlabel('Petal Length')
t=ax.legend(*scatter.legend_elements())
t.get_texts()[0].set_text('Iris-virginica ')
t.get_texts()[1].set_text('Iris-setosa')
t.get_texts()[2].set_text('Iris-versicolor')
plt.show()
```



The optimum number of clusters is 3.¶

Thank you¶