In [21]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans
```

In [23]:
```python
data = sns.load_dataset('iris')
data.head()

sliced_x = data.iloc[:,0:4]
sliced_x

sliced_y = data.iloc[:,-1]
sliced_y
sliced_y.value_counts()
```

Out[23]:
```
species
setosa        50
versicolor    50
virginica     50
Name: count, dtype: int64
```

In [24]:
```python
#Convert y values into int
sliced_y = LabelEncoder().fit_transform(sliced_y)
sliced_y.sort
sliced_y
```

Out[24]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 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, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [25]:
```python
# Convert x values betn 0 to 1
for col in sliced_x.columns:
    sliced_x[col] =MinMaxScaler().fit_transform(sliced_x[[col]])
sliced_x
```

Out[25]:

	sepal_length	sepal_width	petal_length	petal_width
0	0.222222	0.625000	0.067797	0.041667
1	0.166667	0.416667	0.067797	0.041667
2	0.111111	0.500000	0.050847	0.041667
3	0.083333	0.458333	0.084746	0.041667
4	0.194444	0.666667	0.067797	0.041667
...	...	...	...	...
145	0.666667	0.416667	0.711864	0.916667
146	0.555556	0.208333	0.677966	0.750000
147	0.611111	0.416667	0.711864	0.791667
148	0.527778	0.583333	0.745763	0.916667
149	0.444444	0.416667	0.694915	0.708333

150 rows × 4 columns

In [26]:
```python
# Find optimal cluster size
krange = range(1,10)
points=[]

for k in krange:
    km = KMeans(n_clusters=k,n_init=10)
    km.fit_predict(sliced_x,sliced_y)
    points.append(km.inertia_)
points
```

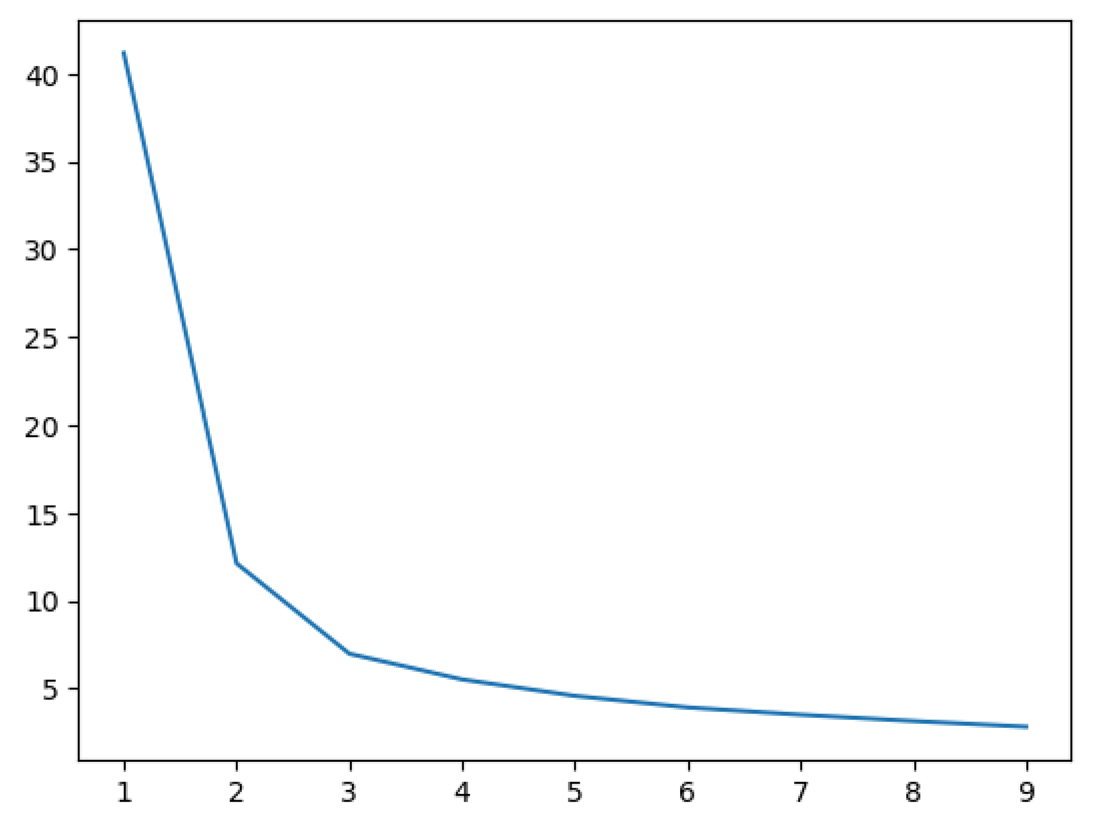
```
C:\Users\Shrey\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL,
when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Shrey\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:14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

Out[26]:
```
[41.16611042137329,
 12.127790750538193,
 6.982216473785235,
 5.516933472040371,
 4.580948640117294,
 3.92379168760291,
 3.5091173931693023,
 3.146928226430177,
 2.824930685729874]
```

In [19]:
```python
# krange=range(1,5)
# points = []

# for k in krange:
#     km = KMeans(n_clusters=k,n_init=5)
#     km.fit_predict(sliced_x,sliced_y)
#     points.append(km.inertia_)
# points
```

In [28]:
```python
plt.plot(krange,points)
```

Out[28]:
```
[<matplotlib.lines.Line2D at 0x2adb1b4b790>]
```



In [32]:
```python
# We choose cluster size = 3 & Predict

km = KMeans(n_clusters=3)
result = km.fit_predict(sliced_x,sliced_y)
result
```

```
C:\Users\Shrey\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Shrey\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: 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(
```

Out[32]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 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, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2,
       2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 2, 1, 1, 2, 2, 2, 2,
       2, 1, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 1])
```

In [34]:
```python
# Assign result to a column
sliced_x['cluster'] = result
sliced_x.head()
```

Out[34]:

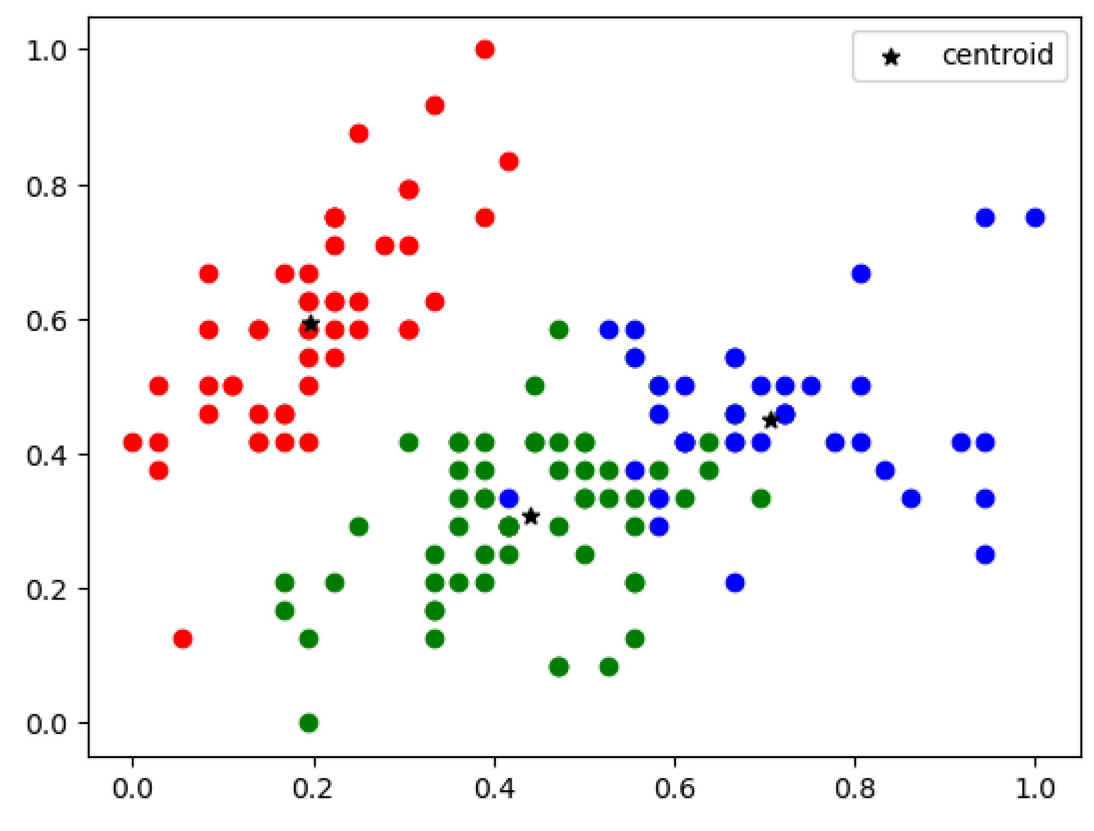
	sepal_length	sepal_width	petal_length	petal_width	cluster
0	0.222222	0.625000	0.067797	0.041667	0
1	0.166667	0.416667	0.067797	0.041667	0
2	0.111111	0.500000	0.050847	0.041667	0
3	0.083333	0.458333	0.084746	0.041667	0
4	0.194444	0.666667	0.067797	0.041667	0

In [37]:
```python
cluster0 = sliced_x[sliced_x.cluster==0]
cluster1=sliced_x[sliced_x.cluster==1]
cluster2=sliced_x[sliced_x.cluster==2]

# cluster1['cluster'].value_counts()

km.cluster_centers_
```

Out[37]:
```
array([[ 1.96111111e-01,  5.95000000e-01,  7.83050847e-02,
         6.08333333e-02,  1.00000000e+00],
       [ 4.41256831e-01,  3.07377049e-01,  5.75715477e-01,
         5.49180328e-01,  2.00000000e+00],
       [ 7.07264957e-01,  4.50854701e-01,  7.97044763e-01,
         8.24786325e-01, -8.88178420e-16]])
```

In [39]:
```python
plt.scatter(cluster0.sepal_length,cluster0.sepal_width,color="red")
plt.scatter(cluster1.sepal_length,cluster1.sepal_width,color="green")
plt.scatter(cluster2.sepal_length,cluster2.sepal_width,color="blue")
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],label="centroid",color="black",marker="*")
plt.legend()
```

Out[39]:
```
<matplotlib.legend.Legend at 0x2adafd826d0>
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: