# CS 403/603 Machine Learning:

Clustering Using KMeans Algorithm

Download datasets of your choices from the UCI Machine Learning Repository or use any synthetic generated dataset, and perform the following tasks:

a) Visualize this dataset.

b) Use KMeans clustering algorithm to fit data.

c) Use any preprocessing on the dataset, and again perform Kmeans clustering.

d) Plot a graph (called elbow plot) between Sum of squared error and Values of K.

import libraries

In [1]:

```
from sklearn.cluster import KMeans
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
%matplotlib inline
```

a) Dataset Visualization

In [2]:

```
from sklearn.datasets import make_blobs
```

In [3]:

```
features, true_labels = make_blobs(n_samples=200,centers=3,cluster_std=2.75,rand
om_state=42)
```

In [4]:

```
features[:5]
```

Out[4]:

```
array([[  9.77075874,   3.27621022],
       [ -9.71349666,  11.27451802],
       [ -6.91330582,  -9.34755911],
       [-10.86185913, -10.75063497],
       [ -8.50038027,  -4.54370383]])
```

In [5]:

```
true_labels[:5]
```

Out[5]:

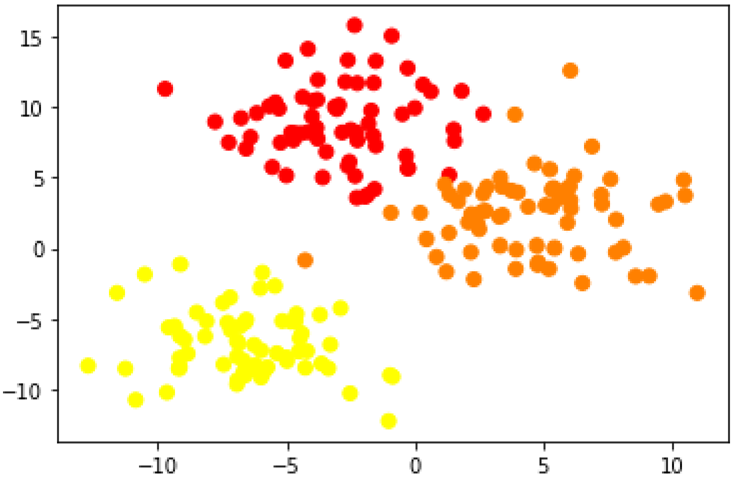
```
array([1, 0, 2, 2, 2])
```

In [6]:

```
plt.scatter(features[:,0],features[:,1],c=true_labels, s=50, cmap='autumn')
```

Out[6]:

```
<matplotlib.collections.PathCollection at 0x7f6dcf686f10>
```



b) Use K-Mean clustering algorithm to fit data.

In [7]:

```
km = KMeans(n_clusters=3)
```

In [8]:

```
y_predicted = km.fit_predict(features)
y_predicted
```

Out[8]:

```
array([2, 1, 0, 0, 0, 0, 1, 0, 1, 0, 2, 2, 2, 2, 2, 0, 1, 0, 2, 0,
2, 2,
       0, 1, 0, 1, 1, 0, 1, 2, 2, 2, 0, 0, 1, 1, 0, 1, 0, 1, 2, 1,
0, 2,
       0, 2, 2, 0, 2, 1, 0, 1, 0, 1, 1, 1, 0, 2, 1, 2, 0, 1, 0, 0,
0, 0,
       1, 1, 0, 1, 1, 0, 1, 2, 2, 2, 2, 1, 1, 1, 1, 2, 0, 0, 0, 0,
0, 1,
       2, 0, 2, 1, 1, 1, 2, 0, 1, 2, 2, 1, 0, 0, 1, 0, 1, 2, 0, 2,
2, 0,
       2, 2, 1, 0, 1, 0, 0, 1, 1, 1, 0, 2, 1, 0, 0, 2, 1, 1, 2, 1,
2, 0,
       1, 0, 0, 2, 2, 2, 1, 2, 1, 1, 0, 2, 2, 1, 2, 0, 0, 2, 1, 0,
2, 0,
       2, 0, 0, 1, 2, 2, 1, 2, 2, 0, 1, 2, 2, 1, 0, 2, 0, 1, 2, 0,
1, 2,
       1, 1, 1, 2, 1, 2, 0, 0, 0, 1, 2, 2, 2, 1, 1, 2, 0, 0, 1, 0,
1, 1,
       2, 2], dtype=int32)
```

In [9]:

```
km.cluster_centers_
```

Out[9]:

```
array([[-6.47709913, -6.71236134],
       [-2.75094934,  8.97602228],
       [ 4.69396233,  2.15039643]])
```

In [12]:

```
features1 = features[y_predicted==0]
features1
```

Out[12]:

```
array([[ -6.91330582,  -9.34755911],
       [-10.86185913, -10.75063497],
       [ -8.50038027,  -4.54370383],
       [ -4.82456978,  -5.20159136],
       [ -8.17498253,  -6.24197227],
       [ -4.64425636,  -5.14863028],
       [ -9.15936347,  -8.42060745],
       [ -9.14073328,  -6.20996976],
       [ -2.51808095, -10.28410222],
       [ -5.99215115,  -9.15499469],
       [ -5.00425652,  -7.73334317],
       [ -7.2970362 ,  -5.26223728],
       [-11.59074715,  -3.15696056],
       [ -9.14974448,  -7.76392066],
       [ -5.74406569,  -8.43035211],
       [ -6.20596912,  -8.27420333],
       [ -9.57877598,  -5.60932504],
       [ -9.12973407,  -1.12604459],
       [ -6.68767146,  -7.93972198],
       [ -0.85766912,  -9.10292988],
       [ -3.63616686,  -8.17034264],
       [ -1.01378541, -12.24835104],
       [ -4.26596164,  -8.46659465],
       [ -4.40572753,  -6.03503591],
       [ -6.93192176,  -9.63706535],
       [ -4.48074544,  -6.37591908],
       [ -6.33221303,  -8.53070601],
       [ -4.19804792,  -7.28451739],
       [ -6.26337287,  -6.84548049],
       [ -4.27360635,  -0.84389633],
       [ -5.98816972,  -7.23800299],
       [ -6.61288829,  -5.24342777],
       [ -5.37510939,  -7.43613939],
       [ -6.86520702,  -6.75091296],
       [ -7.4782505 ,  -3.85847325],
       [ -8.8553488 ,  -7.46708926],
       [ -3.28956047,  -6.82234903],
       [ -8.11730724,  -5.16727228],
       [ -7.45196338,  -8.23586216],
       [ -5.45142428,  -2.66507758],
       [-11.26430325,  -8.52839091],
       [ -4.98173121,  -7.98346589],
       [-12.72034043,  -8.3259359 ],
       [ -2.88898227,  -4.24211482],
       [ -0.95837543,  -8.99031539],
       [ -6.67116466,  -8.7423043 ],
       [ -5.93979826,  -1.72063979],
       [ -6.93710657,  -6.55745929],
       [ -6.81222421,  -5.51061429],
       [ -9.18886226,  -8.52843937],
       [-10.51026851,  -1.84359799],
       [ -4.52106323,  -7.31994055],
       [ -6.56745449,  -5.05925024],
       [ -6.93053832,  -7.67392085],
       [ -7.19461177,  -3.47611474],
       [ -3.69482229,  -4.70303718],
       [ -9.64617499, -10.21912828],
       [ -7.1787176 ,  -5.77540236],
       [ -6.10689955,  -8.59253327],
       [ -3.36604873,  -8.50693091],
       [ -5.8978346 ,  -8.78561098],
       [ -6.61101792,  -9.0058865 ],
       [ -5.16329769,  -5.15215944],
       [ -4.60973223,  -4.64295809],
       [ -9.35026754,  -5.52733187],
       [ -8.96724201,  -6.46652668],
       [ -6.02463139,  -2.82288   ]])
```

In [13]:

```
features2 = features[y_predicted==1]
features2
```

Out[13]:

```
array([[-9.71349666e+00,  1.12745180e+01],
       [-3.03819028e+00,  9.84354132e+00],
       [ 3.91207254e+00,  9.45363489e+00],
       [-2.70722546e+00,  1.17740016e+01],
       [-2.32349506e+00,  5.09622862e+00],
       [-4.37073312e+00,  1.06963959e+01],
       [-5.67442996e+00,  1.00474557e+01],
       [-1.93482274e+00,  3.62519329e+00],
       [-1.51535971e+00,  1.32438867e+01],
       [-2.23515637e+00,  7.62972808e+00],
       [-8.95340615e-01,  1.50380391e+01],
       [-2.25685549e+00,  3.54847161e+00],
       [-4.73255503e+00,  7.63445426e+00],
       [ 8.15820063e-03,  9.91835168e+00],
       [-4.00625012e+00,  9.31932325e+00],
       [-3.33859769e-01,  6.51347063e+00],
       [-6.57513310e+00,  7.03471456e+00],
       [-4.16389081e+00,  1.41080511e+01],
       [-4.43984363e+00,  8.11321523e+00],
       [-5.03761427e+00,  1.32766057e+01],
       [-5.43053284e+00,  1.03166653e+01],
       [-2.47199115e-01,  5.65696609e+00],
       [-4.78414528e-01,  9.48554890e+00],
       [-1.61796671e+00,  7.95530986e+00],
       [-5.00626383e+00,  5.13045095e+00],
       [-3.80025218e+00,  1.05063262e+01],
       [ 6.05625997e+00,  1.25681813e+01],
       [-3.82692678e+00,  8.50372394e+00],
       [-2.54631499e+00,  6.10558107e+00],
       [-2.26981819e+00,  8.19201591e+00],
       [-6.40146716e+00,  7.85751149e+00],
       [-6.77352206e+00,  9.20283431e+00],
       [-2.60465499e+00,  5.80042152e+00],
       [-4.81704581e+00,  8.16395209e+00],
       [-3.58749504e+00,  4.98962002e+00],
       [-3.45166254e+00,  6.80802364e+00],
       [-1.69486686e+00,  9.73218813e+00],
       [-1.56424733e+00,  4.16592570e+00],
       [-3.96603818e+00,  1.04257716e+01],
       [-7.25272166e+00,  7.46799542e+00],
       [-3.77595424e+00,  1.19213723e+01],
       [-2.82723040e+00,  8.18625097e+00],
       [-5.23317252e+00,  7.45696737e+00],
       [-1.59822319e+00,  1.16970352e+01],
       [-2.72025275e-01,  5.62940926e+00],
       [ 3.26051064e-01,  1.15753065e+01],
       [ 1.52133649e+00,  8.39340130e+00],
       [-2.24223436e+00,  1.16780599e+01],
       [-7.78581847e+00,  8.94137297e+00],
       [-5.29448320e+00,  9.87846629e+00],
       [-6.16170926e+00,  9.55565453e+00],
       [-5.55161880e+00,  5.72471791e+00],
       [-1.80093405e+00,  8.80955986e+00],
       [-2.74751611e-01,  1.27439462e+01],
       [-3.11329531e+00,  9.99634570e+00],
       [-4.66314418e+00,  8.12861696e+00],
       [-2.49513562e+00,  8.36917151e+00],
       [ 6.33565117e-01,  1.10821020e+01],
       [-2.34356455e+00,  1.57882019e+01],
       [-4.16095402e+00,  8.21212832e+00],
       [-2.95273333e+00,  1.01254260e+01],
       [-3.78359628e+00,  7.73352931e+00],
       [ 1.55501100e+00,  7.58904303e+00],
       [-2.60771923e+00,  1.33170562e+01],
       [-1.51469855e+00,  7.24020680e+00],
       [-1.84380138e+00,  3.75276546e+00],
       [ 2.67781361e+00,  9.49437511e+00],
       [ 1.83363762e+00,  1.11247316e+01]])
```

In [14]:

```
features3 = features[y_predicted==2]
features3
```

Out[14]:

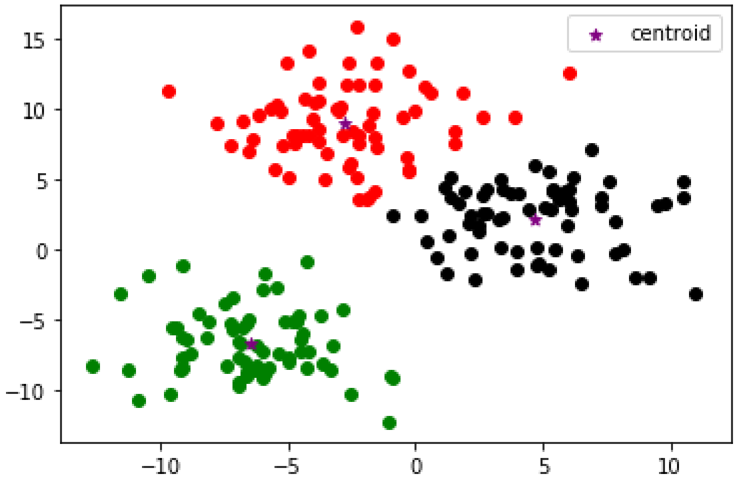
```
array([[ 9.77075874e+00,  3.27621022e+00],
       [ 2.09082004e+00,  1.80947495e+00],
       [ 5.26539366e+00,  5.56781226e+00],
       [ 7.61826975e+00,  4.87112533e+00],
       [ 3.30512908e+00,  2.19832357e+00],
       [-9.29263277e-01,  2.48591905e+00],
       [ 6.52709436e+00, -2.46179896e+00],
       [ 1.04758084e+01,  4.81244915e+00],
       [ 3.33377923e+00,  1.76514294e-01],
       [ 1.23826438e+00, -1.65808600e+00],
       [ 4.61611430e-01,  6.41525984e-01],
       [ 1.10051899e+01, -3.16180960e+00],
       [ 1.33906372e+00,  1.05329129e+00],
       [ 9.12900992e+00, -1.95971911e+00],
       [ 6.36046404e+00, -3.84013596e-01],
       [ 2.19371415e+00, -2.70308600e-01],
       [ 8.51288074e-01, -6.05849176e-01],
       [ 3.94531642e+00, -1.45823407e+00],
       [ 1.36379422e+00,  3.77869211e+00],
       [ 2.81996606e+00,  4.31736135e+00],
       [ 1.34848673e+00,  5.15919571e+00],
       [ 6.05622582e+00,  3.38608105e+00],
       [ 1.69492447e+00,  3.29996883e+00],
       [ 5.22863663e+00, -1.45261196e+00],
       [ 4.79995281e+00, -1.16999864e+00],
       [ 4.76520140e+00,  1.81268728e-01],
       [ 4.67563404e+00,  5.97038840e+00],
       [ 5.32873417e+00,  2.92590226e+00],
       [ 5.86038226e+00,  4.10341333e+00],
       [ 4.02535618e+00,  3.93667104e+00],
       [ 3.77288841e+00,  4.06033504e+00],
       [ 7.83857915e+00,  2.00131060e+00],
       [ 5.35455652e+00,  4.12318258e+00],
       [ 6.02859385e+00,  4.35399647e+00],
       [ 9.49487800e+00,  3.08686939e+00],
       [ 3.41196272e+00,  4.32826637e+00],
       [ 2.20927089e+00,  2.39591373e+00],
       [ 7.26338369e+00,  3.76449563e+00],
       [ 7.28916319e+00,  3.10831723e+00],
       [ 4.81664889e+00, -9.90628455e-01],
       [ 1.05357251e+01,  3.71644700e+00],
       [ 8.12388450e+00,  2.70786535e-02],
       [ 2.19299941e-01,  2.48091279e+00],
       [ 4.42784914e+00,  2.91133761e+00],
       [ 2.76981085e+00,  2.61186735e+00],
       [ 3.42975650e+00,  2.33270627e+00],
       [ 6.90054428e+00,  7.18935039e+00],
       [ 2.51460950e+00,  1.32191852e+00],
       [ 5.94128230e+00,  1.77289017e+00],
       [ 7.82601668e+00, -2.83706692e-01],
       [ 3.96506152e+00, -9.96047680e-02],
       [ 5.62379408e+00,  3.51532713e+00],
       [ 6.20982774e+00,  5.09597519e+00],
       [ 5.44582814e+00,  8.70328437e-03],
       [ 5.40077853e+00,  4.24792362e+00],
       [ 2.67279364e+00,  3.84206349e+00],
       [ 8.60338038e+00, -1.97545123e+00],
       [ 1.17244796e+00,  4.49729004e+00],
       [ 2.46044681e+00,  1.65764447e+00],
       [ 6.07521814e+00,  2.78987754e+00],
       [ 5.11612638e+00,  3.03279248e+00],
       [ 2.31119611e+00, -2.19266018e+00],
       [ 2.63137060e+00,  2.56843081e+00],
       [ 3.34206621e+00,  4.96778383e+00],
       [ 1.95950424e+00,  4.13765234e+00]])
```

In [16]:

```
plt.scatter(features1[:,0],features1[:,1],color='green')
plt.scatter(features2[:,0],features2[:,1],color='red')
plt.scatter(features3[:,0],features3[:,1],color='black')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',mar
ker='*',label='centroid')
plt.legend()
```

Out[16]:

```
<matplotlib.legend.Legend at 0x7f6dce6bcc90>
```



In [17]:

```
km.n_iter_
```

Out[17]:

```
4
```

c) Use any preprocessing on the dataset, and again perform Kmeans clustering.

Preprocessing: In this example, you’ll use the StandardScaler class. This class implements a type of feature scaling called standardization. Standardization scales, or shifts, the values for each numerical feature in your dataset so that the features have a mean of 0 and standard deviation of 1:

In [18]:

```
from sklearn.preprocessing import StandardScaler
```

In [19]:

```
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
```

In [20]:

```
scaled_features[:5]
```

Out[20]:

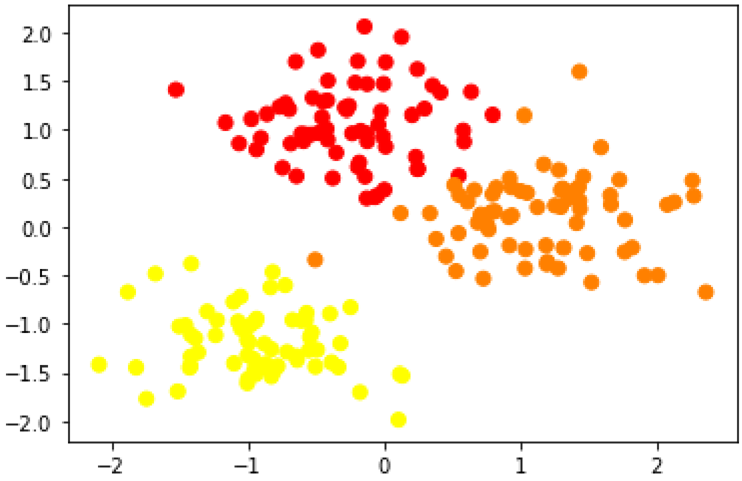
```
array([[ 2.13082109,  0.25604351],
       [-1.52698523,  1.41036744],
       [-1.00130152, -1.56583175],
       [-1.74256891, -1.76832509],
       [-1.29924521, -0.87253446]])
```

In [21]:

```
plt.scatter(scaled_features[:,0],scaled_features[:,1],c=true_labels, s=50, cmap=
'autumn')
```

Out[21]:

```
<matplotlib.collections.PathCollection at 0x7f6dce63d8d0>
```



In [22]:

```
km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(scaled_features)
y_predicted
```

Out[22]:

```
array([0, 2, 1, 1, 1, 1, 2, 1, 0, 1, 0, 0, 0, 0, 2, 1, 2, 1, 0, 1,
0, 0,
       1, 2, 1, 2, 2, 1, 2, 0, 0, 0, 1, 1, 2, 2, 1, 2, 1, 2, 0, 2,
1, 0,
       1, 0, 0, 1, 0, 2, 1, 2, 1, 2, 2, 2, 1, 0, 2, 0, 1, 2, 1, 1,
1, 1,
       2, 2, 1, 2, 2, 1, 2, 0, 0, 0, 0, 2, 0, 2, 2, 0, 1, 1, 1, 1,
1, 2,
       0, 1, 0, 2, 2, 2, 0, 1, 2, 0, 0, 2, 1, 1, 2, 1, 2, 0, 1, 0,
0, 1,
       0, 0, 2, 1, 2, 1, 1, 2, 2, 2, 1, 0, 2, 1, 1, 0, 2, 2, 0, 2,
0, 1,
       2, 1, 1, 0, 0, 0, 2, 0, 2, 2, 1, 0, 0, 2, 0, 1, 1, 0, 2, 1,
0, 1,
       0, 1, 1, 2, 0, 0, 2, 0, 0, 1, 2, 0, 0, 2, 1, 0, 1, 2, 0, 1,
2, 0,
       2, 2, 2, 0, 2, 0, 1, 1, 1, 2, 0, 0, 0, 2, 2, 0, 1, 1, 2, 1,
2, 2,
       0, 0], dtype=int32)
```

In [23]:

```
km.cluster_centers_
```

Out[23]:

```
array([[ 1.19539276,  0.13158148],
       [-0.91941183, -1.18551732],
       [-0.25813925,  1.05589975]])
```

In [24]:

```
scaled_features1 = scaled_features[y_predicted==0]
scaled_features1
```

Out[24]:

```
array([[ 2.13082109,  0.25604351],
       [ 1.0309618 ,  1.14757573],
       [ 0.68905553,  0.04436278],
       [ 1.28502265,  0.58676985],
       [ 1.72673134,  0.48622328],
       [ 0.91701946,  0.10048181],
       [ 1.52188347, -0.57207183],
       [ 2.26318104,  0.47775508],
       [ 0.92239799, -0.19130777],
       [ 0.52900408, -0.45607915],
       [ 0.38320194, -0.12419681],
       [ 2.36256256, -0.67309807],
       [ 0.54792728, -0.06477042],
       [ 2.01034469, -0.49961115],
       [ 1.49060173, -0.27220372],
       [ 0.70837199, -0.2557937 ],
       [ 0.45635648, -0.30421926],
       [ 1.03720271, -0.42723632],
       [ 0.55256997,  0.32856221],
       [ 0.82593913,  0.4063035 ],
       [ 0.54969627,  0.52779789],
       [ 1.43348667,  0.27190018],
       [ 0.61473351,  0.25947237],
       [ 1.43349308,  1.59707024],
       [ 1.2781222 , -0.42642493],
       [ 1.19764479, -0.38563789],
       [ 1.19112086, -0.1906216 ],
       [ 1.17430625,  0.64487005],
       [ 1.29691365,  0.20548671],
       [ 1.39672068,  0.3754263 ],
       [ 1.05222869,  0.35136189],
       [ 1.00483257,  0.3692092 ],
       [ 1.76809033,  0.0720487 ],
       [ 1.30176132,  0.37827942],
       [ 1.42829928,  0.41159072],
       [ 2.07902961,  0.22871765],
       [ 0.93707548,  0.40787732],
       [ 0.71129248,  0.12899822],
       [ 1.66010809,  0.32651335],
       [ 1.66494771,  0.23181302],
       [ 1.20077917, -0.359751  ],
       [ 2.27442928,  0.31957893],
       [ 1.8216511 , -0.21287447],
       [ 0.33771247,  0.14126537],
       [ 1.12778925,  0.20338472],
       [ 0.81652342,  0.16016487],
       [ 0.94041593,  0.11987606],
       [ 1.59199174,  0.82079189],
       [ 0.76861412, -0.02600186],
       [ 1.41190816,  0.03908283],
       [ 1.76573196, -0.25772733],
       [ 1.04090949, -0.23115755],
       [ 1.35230566,  0.2905531 ],
       [ 1.46232257,  0.51867384],
       [ 1.31889586, -0.21552642],
       [ 1.31043864,  0.39628218],
       [ 0.79831025,  0.33770803],
       [ 1.91166753, -0.50188163],
       [ 0.51664827,  0.43227099],
       [ 0.75844608,  0.02245045],
       [ 1.43705212,  0.18585548],
       [ 1.25700049,  0.22091321],
       [ 0.73042704, -0.53322944],
       [ 0.79053384,  0.15389606],
       [ 0.92395371,  0.50017313],
       [ 0.66440344,  0.38036771]])
```

In [25]:

```
scaled_features2 = scaled_features[y_predicted==1]
scaled_features2
```

Out[25]:

```
array([[-1.00130152, -1.56583175],
       [-1.74256891, -1.76832509],
       [-1.29924521, -0.87253446],
       [-0.6091802 , -0.96748146],
       [-1.23815784, -1.11763029],
       [-0.57532971, -0.95983806],
       [-1.42295704, -1.43205314],
       [-1.41945957, -1.11301166],
       [-0.17617985, -1.70099461],
       [-0.82837187, -1.53804066],
       [-0.64291304, -1.33286646],
       [-1.07333975, -0.97623394],
       [-1.87940407, -0.67239826],
       [-1.42115125, -1.33727943],
       [-0.78179844, -1.4334595 ],
       [-0.86851222, -1.41092395],
       [-1.50169393, -1.026326  ],
       [-1.41739467, -0.37929439],
       [-0.95894286, -1.36265126],
       [ 0.13553157, -1.53052661],
       [-0.38607967, -1.39593467],
       [ 0.10622364, -1.984477  ],
       [-0.50431192, -1.43869006],
       [-0.53055036, -1.08776503],
       [-1.00479631, -1.60761359],
       [-0.54463358, -1.13696163],
       [-0.89221216, -1.44794268],
       [-0.49156238, -1.26809147],
       [-0.8792887 , -1.20472921],
       [-0.50574707, -0.33857447],
       [-0.82762443, -1.26137846],
       [-0.94490372, -0.97351933],
       [-0.71253377, -1.28997371],
       [-0.99227186, -1.19108113],
       [-1.10735937, -0.77364128],
       [-1.36588394, -1.29444042],
       [-0.32101077, -1.20139086],
       [-1.22733037, -0.96252849],
       [-1.10242445, -1.40539051],
       [-0.72686046, -0.6014092 ],
       [-1.81812031, -1.44760856],
       [-0.63868433, -1.36896443],
       [-2.09146419, -1.41839005],
       [-0.24580967, -0.82900883],
       [ 0.11662583, -1.51427397],
       [-0.95584402, -1.47848076],
       [-0.81854359, -0.46510698],
       [-1.00576966, -1.1631617 ],
       [-0.98232533, -1.01207996],
       [-1.42849489, -1.44761555],
       [-1.67656432, -0.48285243],
       [-0.55220249, -1.27320377],
       [-0.93637437, -0.94693864],
       [-1.0045366 , -1.32429057],
       [-1.05411147, -0.71845891],
       [-0.39709113, -0.89552961],
       [-1.51434684, -1.6916175 ],
       [-1.05112764, -1.05029445],
       [-0.84991375, -1.45686565],
       [-0.33537002, -1.44451143],
       [-0.81066569, -1.48473082],
       [-0.94455259, -1.51652121],
       [-0.67277006, -0.96034739],
       [-0.56884845, -0.88685894],
       [-1.45879572, -1.01449266],
       [-1.38688981, -1.15003821],
       [-0.83446943, -0.62418341]])
```

In [26]:

```
scaled_features3 = scaled_features[y_predicted==2]
scaled_features3
```

Out[26]:

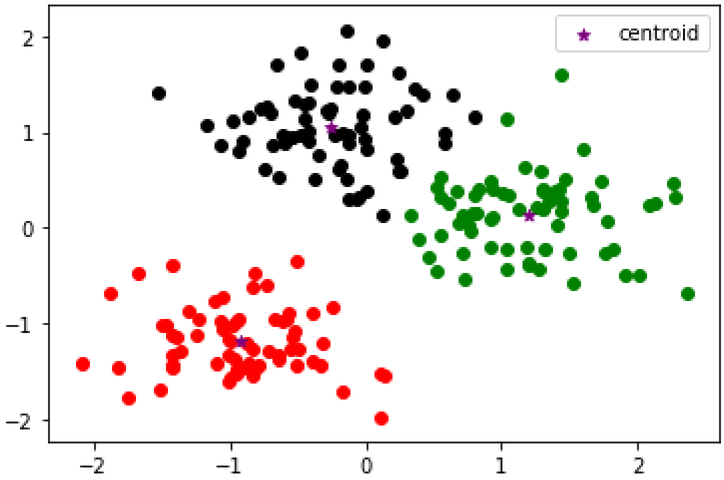
```
array([[-1.52698523,  1.41036744],
       [-0.27382069,  1.20384743],
       [ 0.1220911 ,  0.14198788],
       [-0.21168821,  1.48245342],
       [-0.13964997,  0.51871042],
       [-0.52398081,  1.32693227],
       [-0.76872563,  1.23327655],
       [-0.06668397,  0.3064091 ],
       [ 0.0120624 ,  1.69458874],
       [-0.12306602,  0.88434764],
       [ 0.12845945,  1.95352264],
       [-0.12713963,  0.29533655],
       [-0.59190614,  0.88502973],
       [ 0.29807454,  1.21464413],
       [-0.45555591,  1.12819174],
       [ 0.23386703,  0.72324823],
       [-0.93781588,  0.79847468],
       [-0.48515002,  1.81930583],
       [-0.53695502,  0.954125  ],
       [-0.64917534,  1.69931078],
       [-0.72293848,  1.27212916],
       [ 0.25013595,  0.59963662],
       [ 0.20672956,  1.1521816 ],
       [-0.00720015,  0.93133594],
       [-0.64328988,  0.52364942],
       [-0.41688363,  1.29950122],
       [-0.42189129,  1.01048362],
       [-0.18148026,  0.66438119],
       [-0.12957313,  0.9654976 ],
       [-0.90521333,  0.91722158],
       [-0.97505972,  1.11137994],
       [-0.19243251,  0.62034026],
       [-0.60776771,  0.9614474 ],
       [-0.37694244,  0.50332456],
       [-0.35144242,  0.76575842],
       [-0.02163672,  1.18777682],
       [ 0.00288466,  0.38444815],
       [-0.44800687,  1.28787549],
       [-1.06502052,  0.86100622],
       [-0.41232215,  1.5037221 ],
       [-0.2342169 ,  0.96466559],
       [-0.68588776,  0.85941464],
       [-0.00349367,  1.47134556],
       [ 0.2454753 ,  0.59565959],
       [ 0.357753  ,  1.45377754],
       [ 0.5821456 ,  0.99456174],
       [-0.12439479,  1.46860701],
       [-1.16509953,  1.07364557],
       [-0.6973977 ,  1.20888783],
       [-0.86020325,  1.16229931],
       [-0.74567012,  0.60941463],
       [-0.04154886,  1.05462217],
       [ 0.24496348,  1.62243682],
       [-0.28792027,  1.22590031],
       [-0.57887555,  0.95634779],
       [-0.17187229,  0.99106487],
       [ 0.41548304,  1.38259776],
       [-0.14341764,  2.06178691],
       [-0.48459869,  0.96840024],
       [-0.25777774,  1.24452931],
       [-0.4137568 ,  0.89932834],
       [ 0.58846736,  0.87847593],
       [-0.19300777,  1.70514863],
       [ 0.01218652,  0.82813153],
       [-0.04959641,  0.32482044],
       [ 0.79925265,  1.1534554 ],
       [ 0.64077432,  1.38875012]])
```

In [27]:

```
plt.scatter(scaled_features1[:,0],scaled_features1[:,1],color='green')
plt.scatter(scaled_features2[:,0],scaled_features2[:,1],color='red')
plt.scatter(scaled_features3[:,0],scaled_features3[:,1],color='black')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',mar
ker='*',label='centroid')
plt.legend()
```

Out[27]:

```
<matplotlib.legend.Legend at 0x7f6dce5b4d10>
```



In [29]:

```
# The number of iterations required to converge
km.n_iter_
```

Out[29]:

```
3
```

We can observe that with prepossing, Kmeans converge in 3 iteration and without preprocessing, it converges in 4 iterations

c) Use any preprocessing on the dataset, and again perform Kmeans clustering.

Preprocessing using min max scaler: scales all the data features in the range [0, 1] or else in the range [-1, 1] if there are negative values in the dataset

In [30]:

```
scaler = MinMaxScaler()
```

In [31]:

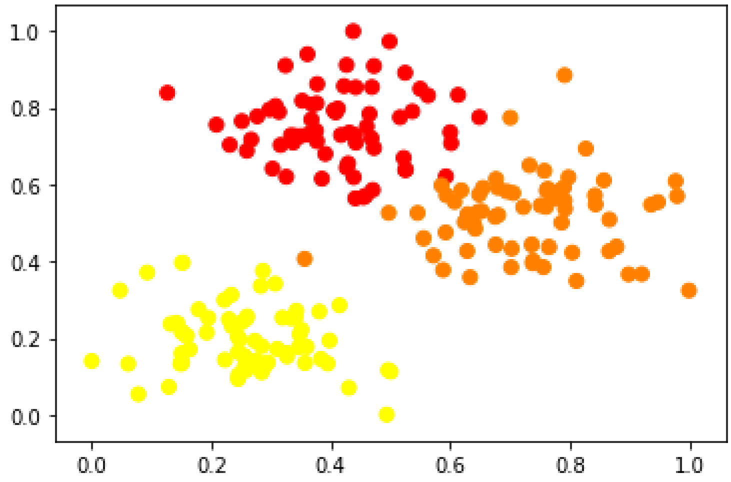
```
scaled_features = scaler.fit_transform(features)
```

In [32]:

```
plt.scatter(scaled_features[:,0],scaled_features[:,1],c=true_labels, s=50, cmap=
'autumn')
```

Out[32]:

```
<matplotlib.collections.PathCollection at 0x7f6dce51fe10>
```



In [33]:

```
km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(scaled_features)
y_predicted
```

Out[33]:

```
array([1, 2, 0, 0, 0, 0, 2, 0, 1, 0, 1, 1, 1, 1, 1, 0, 2, 0, 1, 0,
1, 1,
       0, 2, 0, 2, 2, 0, 2, 1, 1, 1, 0, 0, 2, 2, 0, 2, 0, 2, 1, 2,
0, 1,
       0, 1, 1, 0, 1, 2, 0, 2, 0, 2, 2, 2, 0, 1, 2, 1, 0, 2, 0, 0,
0, 0,
       2, 2, 0, 2, 2, 0, 2, 1, 1, 1, 1, 2, 1, 2, 2, 1, 0, 0, 0, 0,
0, 2,
       1, 0, 1, 2, 2, 2, 1, 0, 2, 1, 1, 2, 0, 0, 2, 0, 2, 1, 0, 1,
1, 0,
       1, 1, 2, 0, 2, 0, 0, 2, 2, 2, 0, 1, 2, 0, 0, 1, 2, 2, 1, 2,
1, 0,
       2, 0, 0, 1, 1, 1, 2, 1, 2, 2, 0, 1, 1, 2, 1, 0, 0, 1, 2, 0,
1, 0,
       1, 0, 0, 2, 1, 1, 2, 1, 1, 0, 2, 1, 1, 2, 0, 1, 0, 2, 1, 0,
2, 1,
       2, 2, 2, 1, 2, 1, 0, 0, 0, 2, 1, 1, 1, 2, 2, 1, 0, 0, 2, 0,
2, 2,
       1, 1], dtype=int32)
```

In [34]:

```
km.cluster_centers_
```

Out[34]:

```
array([[0.26314444, 0.19745615],
       [0.73435516, 0.52300439],
       [0.41031722, 0.75482568]])
```

In [35]:

```
scaled_features1 = scaled_features[y_predicted==0]
scaled_features1
```

Out[35]:

```
array([[0.2447589 , 0.10346464],
       [0.07833255, 0.05342012],
       [0.17786579, 0.27480722],
       [0.33279638, 0.25134187],
       [0.19158088, 0.21423385],
       [0.34039636, 0.25323087],
       [0.15009051, 0.1365269 ],
       [0.15087575, 0.21537531],
       [0.43001186, 0.07006028],
       [0.28358436, 0.11033298],
       [0.32522282, 0.16104005],
       [0.22858516, 0.24917877],
       [0.04761088, 0.3242692 ],
       [0.15049594, 0.15994942],
       [0.29404083, 0.13617933],
       [0.27457221, 0.1417488 ],
       [0.13241282, 0.23679894],
       [0.15133935, 0.39670734],
       [0.25426909, 0.15367899],
       [0.49999604, 0.11219001],
       [0.38288601, 0.14545327],
       [0.49341595, 0.        ],
       [0.35634098, 0.13488664],
       [0.35045003, 0.2216148 ],
       [0.24397426, 0.09313861],
       [0.34728813, 0.20945628],
       [0.2692512 , 0.13259993],
       [0.35920346, 0.17704864],
       [0.27215272, 0.19270809],
       [0.35601877, 0.40677093],
       [0.28375217, 0.17870771],
       [0.2574211 , 0.24984966],
       [0.30959186, 0.17164063],
       [0.2467862 , 0.1960811 ],
       [0.22094722, 0.29924784],
       [0.16290433, 0.17053672],
       [0.39749501, 0.19353314],
       [0.19401181, 0.25256595],
       [0.22205519, 0.14311634],
       [0.30637529, 0.34181354],
       [0.06137006, 0.13268251],
       [0.32617223, 0.15211874],
       [0.        , 0.13990362],
       [0.41437886, 0.28556421],
       [0.49575141, 0.11620671],
       [0.25496483, 0.1250527 ],
       [0.28579096, 0.37549949],
       [0.24375573, 0.20298115],
       [0.24901935, 0.24031973],
       [0.14884718, 0.13268078],
       [0.09315163, 0.37111385],
       [0.34558879, 0.17578518],
       [0.25933608, 0.25641885],
       [0.24403257, 0.16315951],
       [0.23290222, 0.31288569],
       [0.38041376, 0.26912416],
       [0.12957204, 0.07237776],
       [0.23357214, 0.23087534],
       [0.27874786, 0.13039469],
       [0.39427113, 0.13344794],
       [0.28755968, 0.12350805],
       [0.25749993, 0.11565133],
       [0.31851944, 0.25310499],
       [0.3418515 , 0.27126705],
       [0.14204415, 0.23972345],
       [0.15818818, 0.20622451],
       [0.28221536, 0.33618509]])
```

In [36]:

```
scaled_features2 = scaled_features[y_predicted==1]
scaled_features2
```

Out[36]:

```
array([[0.94797035, 0.55372575],
       [0.7010344 , 0.77406042],
       [0.624271  , 0.50141064],
       [0.75807511, 0.63546197],
       [0.85724576, 0.61061274],
       [0.67545253, 0.51527999],
       [0.49697844, 0.52553786],
       [0.81125414, 0.34906403],
       [0.97768726, 0.60851989],
       [0.6766601 , 0.44316665],
       [0.58833689, 0.37773064],
       [0.55560199, 0.45975256],
       [1.        , 0.32409624],
       [0.59258545, 0.47443929],
       [0.92092147, 0.36697207],
       [0.80423089, 0.4231739 ],
       [0.62860785, 0.4272295 ],
       [0.57202635, 0.41526153],
       [0.70243559, 0.3848589 ],
       [0.59362781, 0.57164813],
       [0.65500355, 0.59086124],
       [0.59298262, 0.62088755],
       [0.79140765, 0.55764459],
       [0.60758452, 0.55457316],
       [0.79140909, 0.8851492 ],
       [0.75652585, 0.38505943],
       [0.73845739, 0.3951396 ],
       [0.73699267, 0.44333623],
       [0.73321752, 0.64982095],
       [0.76074483, 0.54123106],
       [0.7831531 , 0.5832302 ],
       [0.70580916, 0.57728288],
       [0.69516797, 0.58169369],
       [0.86653151, 0.50825298],
       [0.76183321, 0.58393532],
       [0.790243  , 0.59216793],
       [0.93634233, 0.54697239],
       [0.67995543, 0.59125019],
       [0.62926354, 0.52232758],
       [0.84228777, 0.57114178],
       [0.84337435, 0.54773739],
       [0.73916111, 0.40153733],
       [0.98021267, 0.56942799],
       [0.87855675, 0.43783662],
       [0.54538888, 0.5253593 ],
       [0.72277371, 0.54071157],
       [0.65288957, 0.53003015],
       [0.68070541, 0.52007311],
       [0.82699457, 0.69329855],
       [0.64213317, 0.48402061],
       [0.78656293, 0.50010575],
       [0.86600202, 0.42675162],
       [0.70326782, 0.43331811],
       [0.77318122, 0.5622545 ],
       [0.79788177, 0.61863262],
       [0.76568019, 0.43718122],
       [0.76378141, 0.58838455],
       [0.64880042, 0.57390845],
       [0.89876688, 0.36641094],
       [0.58556282, 0.59727888],
       [0.63985028, 0.49599519],
       [0.79220815, 0.53637937],
       [0.75178369, 0.54504359],
       [0.63355956, 0.35866359],
       [0.64705449, 0.52848087],
       [0.67700938, 0.61406033],
       [0.61873621, 0.58445143]])
```

In [37]:

```
scaled_features3 = scaled_features[y_predicted==2]
scaled_features3
```

Out[37]:

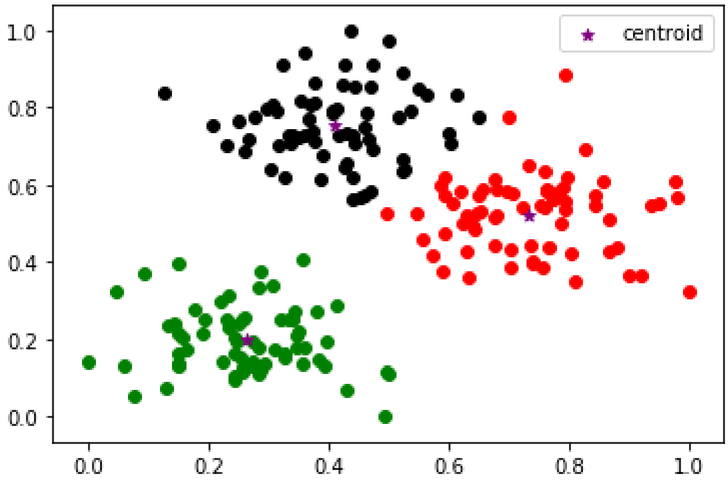
```
array([[0.12673452, 0.83900717],
       [0.40808994, 0.78796749],
       [0.42203967, 0.85682261],
       [0.4382134 , 0.61864166],
       [0.351925  , 0.81838687],
       [0.29697589, 0.79524065],
       [0.45459543, 0.56617318],
       [0.47227525, 0.90925007],
       [0.44193676, 0.70900582],
       [0.49840824, 0.9732434 ],
       [0.44102217, 0.56343669],
       [0.33667468, 0.70917439],
       [0.53648953, 0.79063581],
       [0.36728748, 0.76926983],
       [0.52207392, 0.66919145],
       [0.25901243, 0.68778304],
       [0.36064313, 0.94007285],
       [0.34901208, 0.7262507 ],
       [0.32381684, 0.91041708],
       [0.30725583, 0.80484275],
       [0.52572655, 0.63864189],
       [0.51598113, 0.77519872],
       [0.4679505 , 0.72061858],
       [0.32513822, 0.61986229],
       [0.37597003, 0.81160752],
       [0.37484573, 0.74017926],
       [0.42882183, 0.65464296],
       [0.44047581, 0.72906134],
       [0.26633222, 0.71713033],
       [0.2506506 , 0.76511493],
       [0.42636288, 0.64375862],
       [0.33311351, 0.72806037],
       [0.38493746, 0.61483917],
       [0.39066262, 0.67969749],
       [0.46470926, 0.78399578],
       [0.4702147 , 0.58545987],
       [0.36898236, 0.80873432],
       [0.23045296, 0.70323718],
       [0.37699415, 0.86207899],
       [0.41698162, 0.72885572],
       [0.31557431, 0.70284383],
       [0.46878266, 0.8540774 ],
       [0.52468016, 0.637659  ],
       [0.5498883 , 0.84973561],
       [0.60026801, 0.7362443 ],
       [0.44163843, 0.85340059],
       [0.20798363, 0.7557892 ],
       [0.31299015, 0.78921319],
       [0.27643771, 0.77769923],
       [0.30215222, 0.64105844],
       [0.46023866, 0.75108773],
       [0.52456525, 0.89141833],
       [0.40492436, 0.79341768],
       [0.33960026, 0.72680005],
       [0.43097898, 0.73538008],
       [0.56284961, 0.83214413],
       [0.4373675 , 1.        ],
       [0.36076692, 0.72977871],
       [0.41169183, 0.79802168],
       [0.37667205, 0.71270817],
       [0.60168735, 0.70755467],
       [0.42623373, 0.91185986],
       [0.47230312, 0.69511248],
       [0.45843186, 0.57072339],
       [0.64901201, 0.77551353],
       [0.6134311 , 0.83366463]])
```

In [38]:

```
plt.scatter(scaled_features1[:,0],scaled_features1[:,1],color='green')
plt.scatter(scaled_features2[:,0],scaled_features2[:,1],color='red')
plt.scatter(scaled_features3[:,0],scaled_features3[:,1],color='black')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',mar
ker='*',label='centroid')
plt.legend()
```

Out[38]:

```
<matplotlib.legend.Legend at 0x7f6dce494810>
```



In [40]:

```
km.n_iter_
```

Out[40]:

```
4
```

d) Plot a graph (called elbow plot) between Sum of squared error and Values of K. Elbow Plot

In [41]:

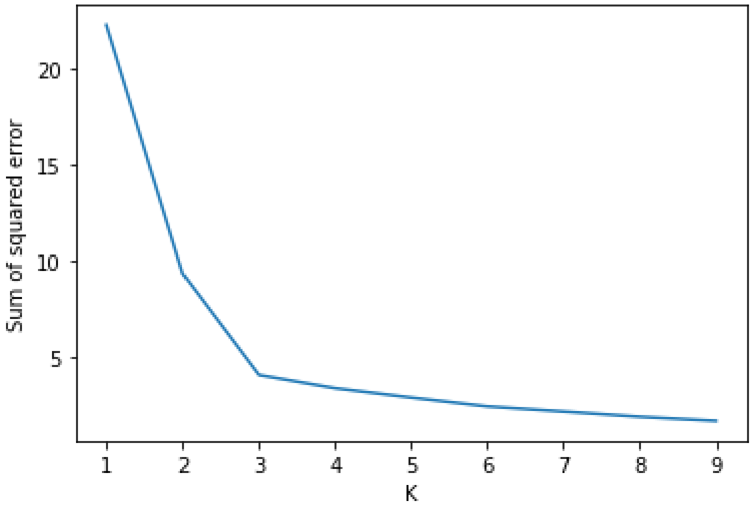
```
sse = []
k_rng = range(1,10)
for k in k_rng:
    km = KMeans(n_clusters=k)
    km.fit(scaled_features)
    sse.append(km.inertia_)
```

In [42]:

```
plt.xlabel('K')
plt.ylabel('Sum of squared error')
plt.plot(k_rng,sse)
```

Out[42]:

```
[<matplotlib.lines.Line2D at 0x7f6dce482ad0>]
```



In [ ]:

```

```

# CS 403/603 Machine Learning: Lab5 Q2

Clustering Using KMeans Algorithm

Use iris flower dataset from sklearn library and perform the following tasks:

a) Visualize this dataset.

b) Use KMeans clustering algorithm and ty to form clusters of flowers using petal width and length features. Drop other two features for simplicity.

c) Figure out if any preprocessing such as scaling would help here.

d) Plot a graph (called elbow plot) between Sum of squared error and Values of K.

import libraries

In [1]:

```
from sklearn.cluster import KMeans
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
from sklearn.datasets import load_iris
%matplotlib inline
```

Load Dataset

In [2]:

```
iris = load_iris()
```

a) Dataset Visualization

In [3]:

```
df = pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()
```

Out[3]:

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) |
|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 |

In [4]:

```
df['flower'] = iris.target
df.head()
```

Out[4]:

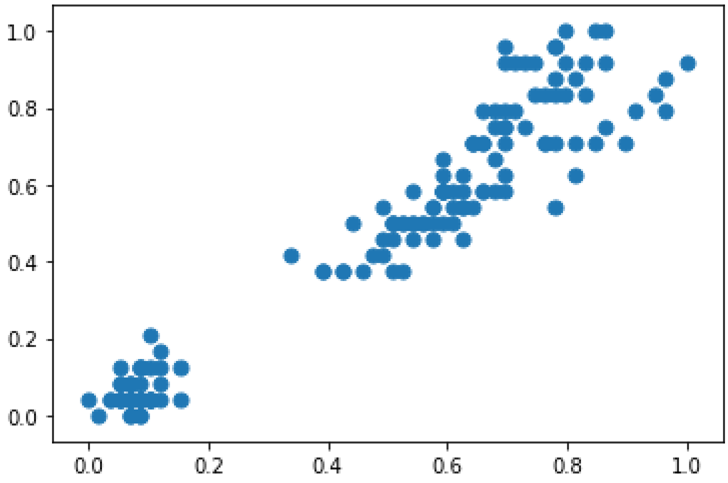
| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | flower |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | 0 |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | 0 |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | 0 |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | 0 |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | 0 |

In [27]:

```
plt.scatter(df['petal length (cm)'],df['petal width (cm)'], s=50, cmap='autumn')
```

Out[27]:

```
<matplotlib.collections.PathCollection at 0x7fdb9aa7a890>
```



b) Use K-Mean clustering algorithm and ty to form clusters of flowers using petal width and length features. Drop other two features for simplicity.

In [5]:

```
df.drop(['sepal length (cm)', 'sepal width (cm)', 'flower'],axis='columns',inpla
ce=True)
```

In [6]:

```
df.head(3)
```

Out[6]:

|   | petal length (cm) | petal width (cm) |
|---|---|---|
| **0** | 1.4 | 0.2 |
| **1** | 1.4 | 0.2 |
| **2** | 1.3 | 0.2 |

In [7]:

```
km = KMeans(n_clusters=3)
yp = km.fit_predict(df)
yp
```

Out[7]:

```
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, 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, 1, 2, 2, 2, 2, 2, 1, 2, 2,
2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1,
1, 1,
       1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=
int32)
```

In [8]:

```
df['cluster'] = yp
df.head(2)
```

Out[8]:

|   | petal length (cm) | petal width (cm) | cluster |
|---|---|---|---|
| **0** | 1.4 | 0.2 | 0 |
| **1** | 1.4 | 0.2 | 0 |

In [9]:

```
df.cluster.unique()
```

Out[9]:

```
array([0, 2, 1], dtype=int32)
```

In [10]:

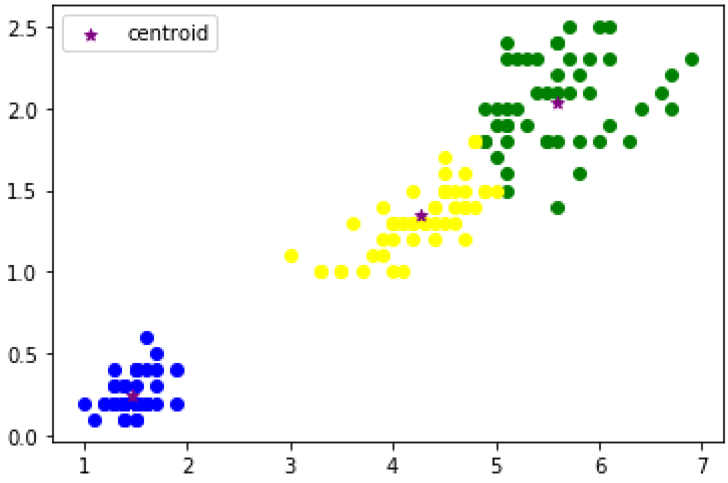
```
df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]
```

In [11]:

```
plt.scatter(df1['petal length (cm)'],df1['petal width (cm)'],color='blue')
plt.scatter(df2['petal length (cm)'],df2['petal width (cm)'],color='green')
plt.scatter(df3['petal length (cm)'],df3['petal width (cm)'],color='yellow')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',mar
ker='*',label='centroid')
plt.legend()
```

Out[11]:

```
<matplotlib.legend.Legend at 0x7fdb9adf7e90>
```



In [12]:

```
km.n_iter_
```

Out[12]:

```
7
```

c) Use any preprocessing on the dataset, and again perform Kmeans clustering.

Preprocessing using min max scaler: scales all the data features in the range [0, 1] or else in the range [-1, 1] if there are negative values in the datas

In [15]:

```
scaler = MinMaxScaler()
```

In [16]:

```
scaler.fit(df[['petal length (cm)']])
df['petal length (cm)'] = scaler.transform(df[['petal length (cm)']])
scaler.fit(df[['petal width (cm)']])
df['petal width (cm)'] = scaler.transform(df[['petal width (cm)']])
```

In [17]:

```
df.head()
```

Out[17]:

| | petal length (cm) | petal width (cm) | cluster |
|---|---|---|---|
| **0** | 0.067797 | 0.041667 | 0 |
| **1** | 0.067797 | 0.041667 | 0 |
| **2** | 0.050847 | 0.041667 | 0 |
| **3** | 0.084746 | 0.041667 | 0 |
| **4** | 0.067797 | 0.041667 | 0 |

In [18]:

```
km = KMeans(n_clusters=3)
yp = km.fit_predict(df)
yp
```

Out[18]:

```
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, 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, 0, 2, 2, 2, 2, 2, 0, 2, 2,
2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 2, 0,
0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0,
0, 0,
       0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=
int32)
```

In [19]:

```
df['cluster'] = yp
df.head(2)
```

Out[19]:

| | petal length (cm) | petal width (cm) | cluster |
|---|---|---|---|
| **0** | 0.067797 | 0.041667 | 1 |
| **1** | 0.067797 | 0.041667 | 1 |

In [20]:

```
df.cluster.unique()
```

Out[20]:

```
array([1, 2, 0], dtype=int32)
```

In [21]:

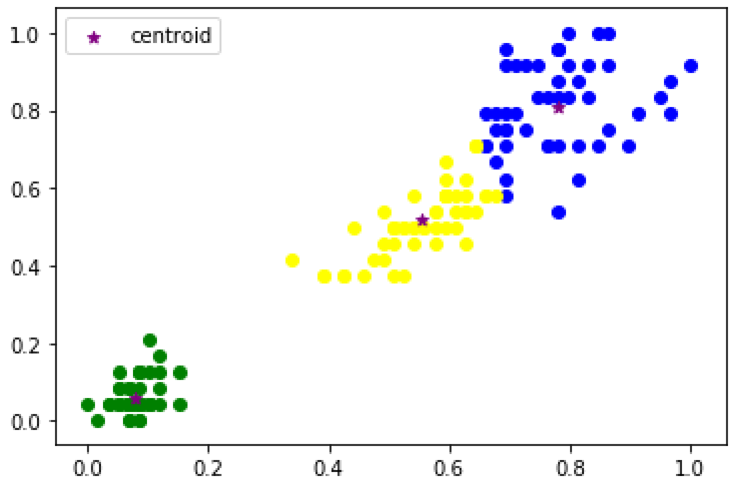
```
df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]
```

In [22]:

```
plt.scatter(df1['petal length (cm)'],df1['petal width (cm)'],color='blue')
plt.scatter(df2['petal length (cm)'],df2['petal width (cm)'],color='green')
plt.scatter(df3['petal length (cm)'],df3['petal width (cm)'],color='yellow')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',mar
ker='*',label='centroid')
plt.legend()
```

Out[22]:

```
<matplotlib.legend.Legend at 0x7fdb9acc2bd0>
```



In [23]:

```
km.n_iter_
```

Out[23]:

```
2
```

We can observe that with preprossing, Kmeans converge in 2 iteration and without preprocessing, it converges in 7 iterations

d) Plot a graph (called elbow plot) between Sum of squared error and Values of

In [23]:

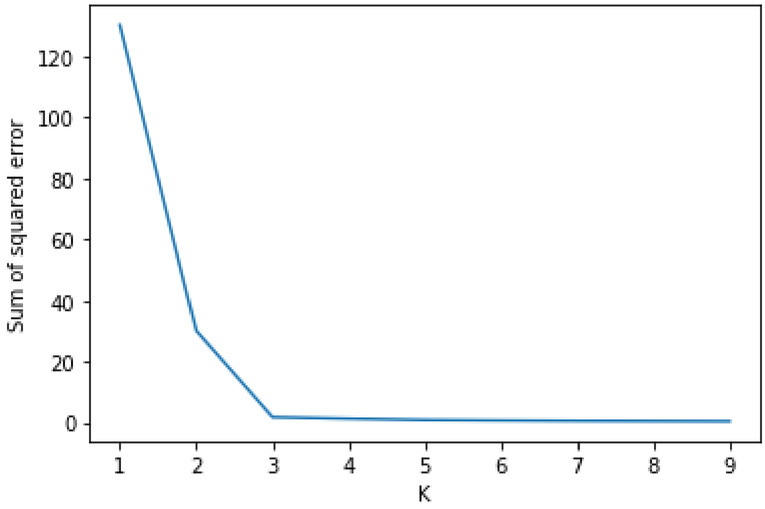
```
sse = []
k_rng = range(1,10)
for k in k_rng:
    km = KMeans(n_clusters=k)
    km.fit(df)
    sse.append(km.inertia_)
```

In [24]:

```
plt.xlabel('K')
plt.ylabel('Sum of squared error')
plt.plot(k_rng,sse)
```

Out[24]:

```
[<matplotlib.lines.Line2D at 0x7f7909e1f610>]
```



In [ ]: