In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
Bad key "text.kerning_factor" on line 4 in
C:\Users\91920\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_tes
t_patch.mplstyle.
You probably need to get an updated matplotlibrc file from
https://github.com/matplotlib/matplotlib/blob/v3.1.3/matplotlibrc.template
or from the matplotlib source distribution
```

In [2]:

```
path = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
```

In [3]:

```
headernames = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class'
```

In [4]:

```
dataset = pd.read_csv(path, names = headernames)
print(type(dataset))
dataset.head()
```

```
<class 'pandas.core.frame.DataFrame'>
```

Out[4]:

	sepal-length	sepal-width	petal-length	petal-width	Class
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 [5]:

```
dataset.drop(['Class'],axis= 1 ,inplace=True)
dataset
```

Out[5]:

	sepal-length	sepal-width	petal-length	petal-width
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
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

define standard scaler

scaler = StandardScaler()

transform data

scaled = scaler.fit_transform(data) print(scaled)

In [6]:
```
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

In [ ]:
```

```

In [7]:
```
pca=PCA()
```

In [8]:
```
pca.fit(dataset)
pca.explained_variance_ratio_
```

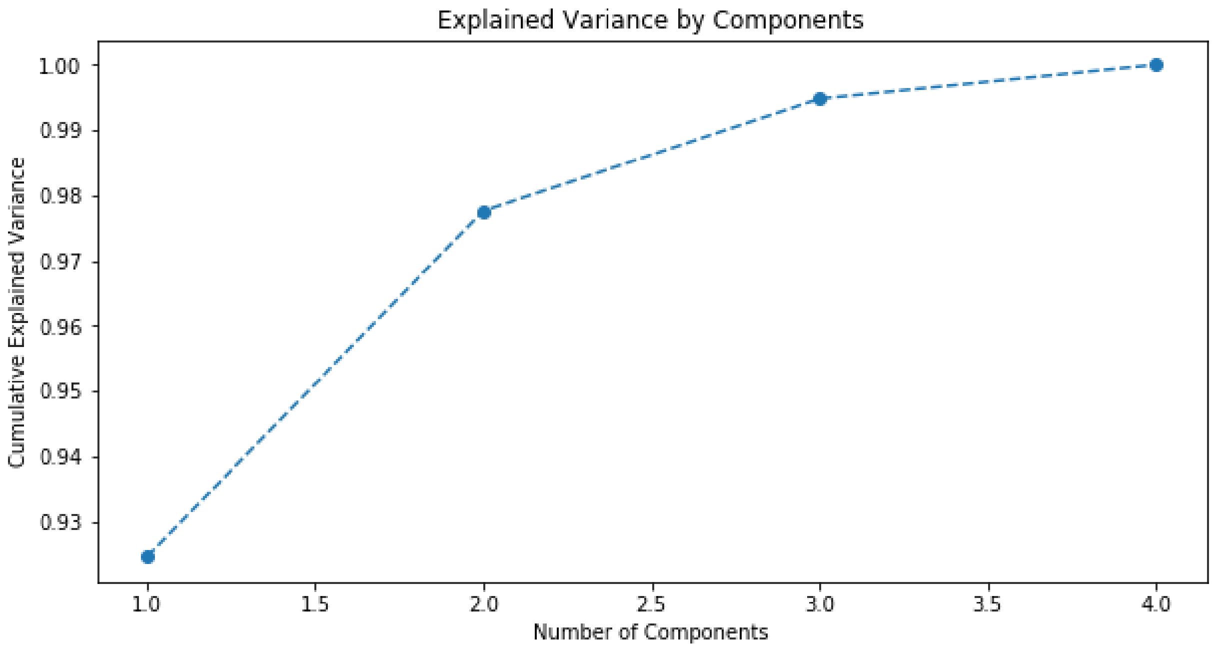
Out[8]:
```
array([0.92461621, 0.05301557, 0.01718514, 0.00518309])
```

In [9]:
```
pca.explained_variance_ratio_.cumsum()
```

Out[9]:
```
array([0.92461621, 0.97763178, 0.99481691, 1.        ])
```

In [15]:
```
# Plot the cumulative variance explained by total number of components.

# On this graph we choose the subset of components we want to keep.
# Generally, we want to keep around 80 % - 90% of the explained variance.
plt.figure(figsize=(10,5))

plt.plot (range (1,5), pca.explained_variance_ratio_.cumsum (), marker = 'o', linest

plt.title('Explained Variance by Components')

plt.xlabel('Number of Components')

plt.ylabel('Cumulative Explained Variance')
```

Out[15]:
```
Text(0, 0.5, 'Cumulative Explained Variance')
```



In [ ]:

In [12]:

```
pca=PCA(n_components= 2)
pca.fit(dataset)
pca.explained_variance_ratio_
pca.explained_variance_ratio_.cumsum()
```

Out[12]: array([0.92461621, 0.97763178])

In [11]:

```
df= pca.transform(dataset)
print(df)
#new_df


#-------------------------#
df1=np.transpose(df)
PCA1=df1[0]
PCA2=df1[1]
```

```
[[-2.68420713  0.32660731]
 [-2.71539062 -0.16955685]
 [-2.88981954 -0.13734561]
 [-2.7464372  -0.31112432]
 [-2.72859298  0.33392456]
 [-2.27989736  0.74778271]
 [-2.82089068 -0.08210451]
 [-2.62648199  0.17040535]
 [-2.88795857 -0.57079803]
 [-2.67384469 -0.1066917 ]
 [-2.50652679  0.65193501]
 [-2.61314272  0.02152063]
 [-2.78743398 -0.22774019]
 [-3.22520045 -0.50327991]
 [-2.64354322  1.1861949 ]
 [-2.38386932  1.34475434]
 [-2.6225262   0.81808967]
 [-2.64832273  0.31913667]
 [-2.19907796  0.87924409]
 [-2.58734619  0.52047364]
 [-2.3105317   0.39786782]
 [-2.54323491  0.44003175]
```

```
 [-3.21585769  0.14161557]
 [-2.30312854  0.10552268]
 [-2.35617109 -0.03120959]
 [-2.50791723 -0.13905634]
 [-2.469056    0.13788731]
 [-2.56239095  0.37468456]
 [-2.63982127  0.31929007]
 [-2.63284791 -0.19007583]
 [-2.58846205 -0.19739308]
 [-2.41007734  0.41808001]
 [-2.64763667  0.81998263]
 [-2.59715948  1.10002193]
 [-2.67384469 -0.1066917 ]
 [-2.86699985  0.0771931 ]
 [-2.62522846  0.60680001]
 [-2.67384469 -0.1066917 ]
 [-2.98184266 -0.48025005]
 [-2.59032303  0.23605934]
 [-2.77013891  0.27105942]
 [-2.85221108 -0.93286537]
 [-2.99829644 -0.33430757]
 [-2.4055141   0.19591726]
 [-2.20883295  0.44269603]
 [-2.71566519 -0.24268148]
 [-2.53757337  0.51036755]
 [-2.8403213  -0.22057634]
 [-2.54268576  0.58628103]
 [-2.70391231  0.11501085]
 [ 1.28479459  0.68543919]
 [ 0.93241075  0.31919809]
 [ 1.46406132  0.50418983]
 [ 0.18096721 -0.82560394]
 [ 1.08713449  0.07539039]
 [ 0.64043675 -0.41732348]
 [ 1.09522371  0.28389121]
 [-0.75146714 -1.00110751]
 [ 1.04329778  0.22895691]
 [-0.01019007 -0.72057487]
 [-0.5110862  -1.26249195]
 [ 0.51109806 -0.10228411]
 [ 0.26233576 -0.5478933 ]
 [ 0.98404455 -0.12436042]
 [-0.174864   -0.25181557]
 [ 0.92757294  0.46823621]
 [ 0.65959279 -0.35197629]
 [ 0.23454059 -0.33192183]
 [ 0.94236171 -0.54182226]
 [ 0.0432464  -0.58148945]
 [ 1.11624072 -0.08421401]
 [ 0.35678657 -0.06682383]
 [ 1.29646885 -0.32756152]
 [ 0.92050265 -0.18239036]
 [ 0.71400821  0.15037915]
 [ 0.89964086  0.32961098]
 [ 1.33104142  0.24466952]
 [ 1.55739627  0.26739258]
 [ 0.81245555 -0.16233157]
 [-0.30733476 -0.36508661]
 [-0.07034289 -0.70253793]
 [-0.19188449 -0.67749054]
 [ 0.13499495 -0.31170964]
 [ 1.37873698 -0.42120514]
 [ 0.58727485 -0.48328427]
 [ 0.8072055   0.19505396]
 [ 1.22042897  0.40803534]
 [ 0.81286779 -0.370679  ]
 [ 0.24519516 -0.26672804]
 [ 0.16451343 -0.67966147]
 [ 0.46303099 -0.66952655]
```

```
 [ 0.89016045 -0.03381244]
 [ 0.22887905 -0.40225762]
 [-0.70708128 -1.00842476]
 [ 0.35553304 -0.50321849]
 [ 0.33112695 -0.21118014]
 [ 0.37523823 -0.29162202]
 [ 0.64169028  0.01907118]
 [-0.90846333 -0.75156873]
 [ 0.29780791 -0.34701652]
 [ 2.53172698 -0.01184224]
 [ 1.41407223 -0.57492506]
 [ 2.61648461  0.34193529]
 [ 1.97081495 -0.18112569]
 [ 2.34975798 -0.04188255]
 [ 3.39687992  0.54716805]
 [ 0.51938325 -1.19135169]
 [ 2.9320051   0.35237701]
 [ 2.31967279 -0.24554817]
 [ 2.91813423  0.78038063]
 [ 1.66193495  0.2420384 ]
 [ 1.80234045 -0.21615461]
 [ 2.16537886  0.21528028]
 [ 1.34459422 -0.77641543]
 [ 1.5852673  -0.53930705]
 [ 1.90474358  0.11881899]
 [ 1.94924878  0.04073026]
 [ 3.48876538  1.17154454]
 [ 3.79468686  0.25326557]
 [ 1.29832982 -0.76101394]
 [ 2.42816726  0.37678197]
 [ 1.19809737 -0.60557896]
 [ 3.49926548  0.45677347]
 [ 1.38766825 -0.20403099]
 [ 2.27585365  0.33338653]
 [ 2.61419383  0.55836695]
 [ 1.25762518 -0.179137  ]
 [ 1.29066965 -0.11642525]
 [ 2.12285398 -0.21085488]
 [ 2.3875644   0.46251925]
 [ 2.84096093  0.37274259]
 [ 3.2323429   1.37052404]
 [ 2.15873837 -0.21832553]
 [ 1.4431026  -0.14380129]
 [ 1.77964011 -0.50146479]
 [ 3.07652162  0.68576444]
 [ 2.14498686  0.13890661]
 [ 1.90486293  0.04804751]
 [ 1.16885347 -0.1645025 ]
 [ 2.10765373  0.37148225]
 [ 2.31430339  0.18260885]
 [ 1.92245088  0.40927118]
 [ 1.41407223 -0.57492506]
 [ 2.56332271  0.2759745 ]
 [ 2.41939122  0.30350394]
 [ 1.94401705  0.18741522]
 [ 1.52566363 -0.37502085]
 [ 1.76404594  0.07851919]
 [ 1.90162908  0.11587675]
 [ 1.38966613 -0.28288671]]
```

In [13]:

```
from sklearn.cluster import KMeans
```

In [17]:

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

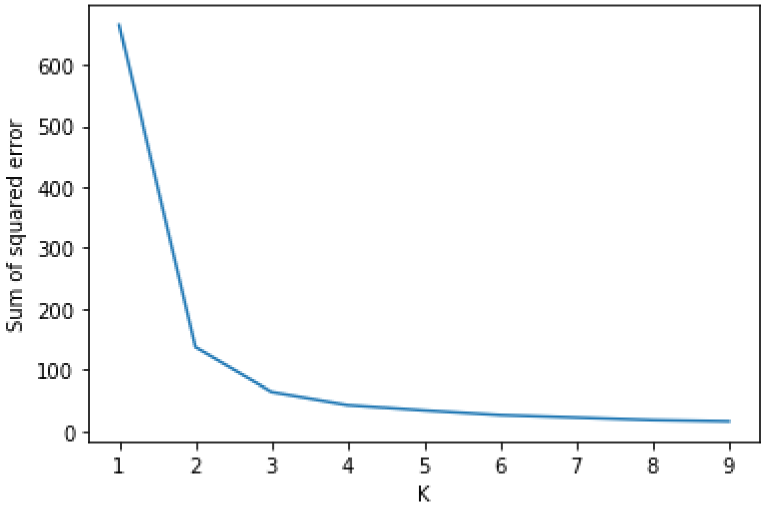
```
    sse.append(km.inertia_)

print(sse)
```

```
C:\Users\91920\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:882: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less c
hunks than available threads. You can avoid it by setting the environment variable O
MP_NUM_THREADS=1.
  f"KMeans is known to have a memory leak on Windows "
[665.5955666521963, 137.15100934920733, 63.87383806036229, 42.26258875648066, 33.604
13548665399, 26.072167353029073, 21.91886230330283, 18.0815543792374, 15.73798401908
9966]
```

In [16]:

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

Out[16]: [<matplotlib.lines.Line2D at 0x201185e1788>]



In [65]:

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

Out[65]:

```
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, 2, 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, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1,
       1, 1, 1, 2, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 1, 2, 2, 1, 1, 1, 1,
       1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 2])
```

In [66]:

```
km.cluster_centers_
```

Out[66]:

```
array([[-2.64084076,  0.19051995],
       [ 2.34645113,  0.27235455],
       [ 0.66443351, -0.33029221]])
```

In [75]:

```
df_plot= pd.DataFrame()
df_plot['pca1']=np.transpose(PCA1)
df_plot['pca2']=np.transpose(PCA2)
df_plot['cluster']=y_predicted
df_plot
```

Out[75]:

	pca1	pca2	cluster

	pca1	pca2	cluster
0	-2.684207	0.326607	0
1	-2.715391	-0.169557	0
2	-2.889820	-0.137346	0
3	-2.746437	-0.311124	0
4	-2.728593	0.333925	0
...	...	...	...
145	1.944017	0.187415	1
146	1.525664	-0.375021	2
147	1.764046	0.078519	1
148	1.901629	0.115877	1
149	1.389666	-0.282887	2

150 rows × 3 columns

In [76]:

```
df_plot1 = df_plot[df_plot.cluster==0]
df_plot2 = df_plot[df_plot.cluster==1]
df_plot3 = df_plot[df_plot.cluster==2]
```

In [79]:

```
df_plot1['pca1']
```

Out[79]:

```
0     -2.684207
1     -2.715391
2     -2.889820
3     -2.746437
4     -2.728593
5     -2.279897
6     -2.820891
7     -2.626482
8     -2.887959
9     -2.673845
10    -2.506527
11    -2.613143
12    -2.787434
13    -3.225200
14    -2.643543
15    -2.383869
16    -2.622526
17    -2.648323
18    -2.199078
19    -2.587346
20    -2.310532
21    -2.543235
22    -3.215858
23    -2.303129
24    -2.356171
25    -2.507917
26    -2.469056
27    -2.562391
28    -2.639821
29    -2.632848
30    -2.588462
31    -2.410077
32    -2.647637
```

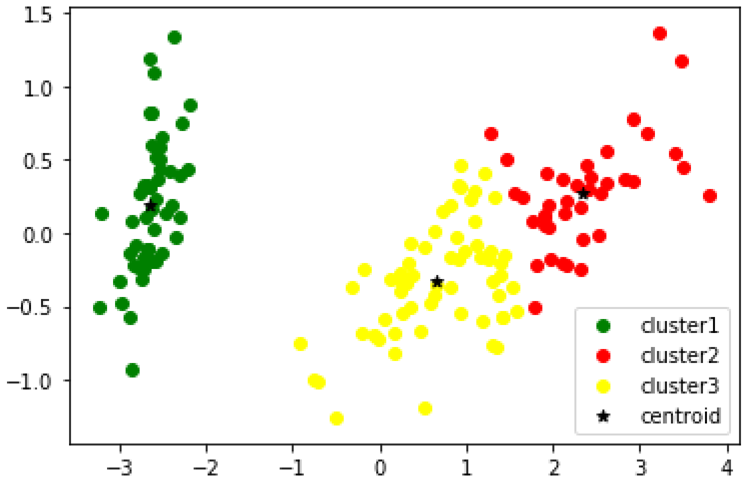
```
33   -2.597159
34   -2.673845
35   -2.867000
36   -2.625228
37   -2.673845
38   -2.981843
39   -2.590323
40   -2.770139
41   -2.852211
42   -2.998296
43   -2.405514
44   -2.208833
45   -2.715665
46   -2.537573
47   -2.840321
48   -2.542686
49   -2.703912
Name: pca1, dtype: float64
```

In [88]:

```
plt.scatter(df_plot1['pca1'],df_plot1['pca2'],color='green',label='cluster1')
plt.scatter(df_plot2['pca1'],df_plot2['pca2'],color='red',label='cluster2')
plt.scatter(df_plot3['pca1'],df_plot3['pca2'],color='yellow',label='cluster3')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='black',marker='
plt.legend()
```

Out[88]:

```
<matplotlib.legend.Legend at 0x233fe446fc8>
```



In [ ]: