Name: Yassen Ayman // Track: Data Science and Bussiness Analytics

predection using unsupervised ML for Sparks intern

In [1]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
```

In [3]:
```
data=pd.read_csv('Iris.csv')
data.head()
```

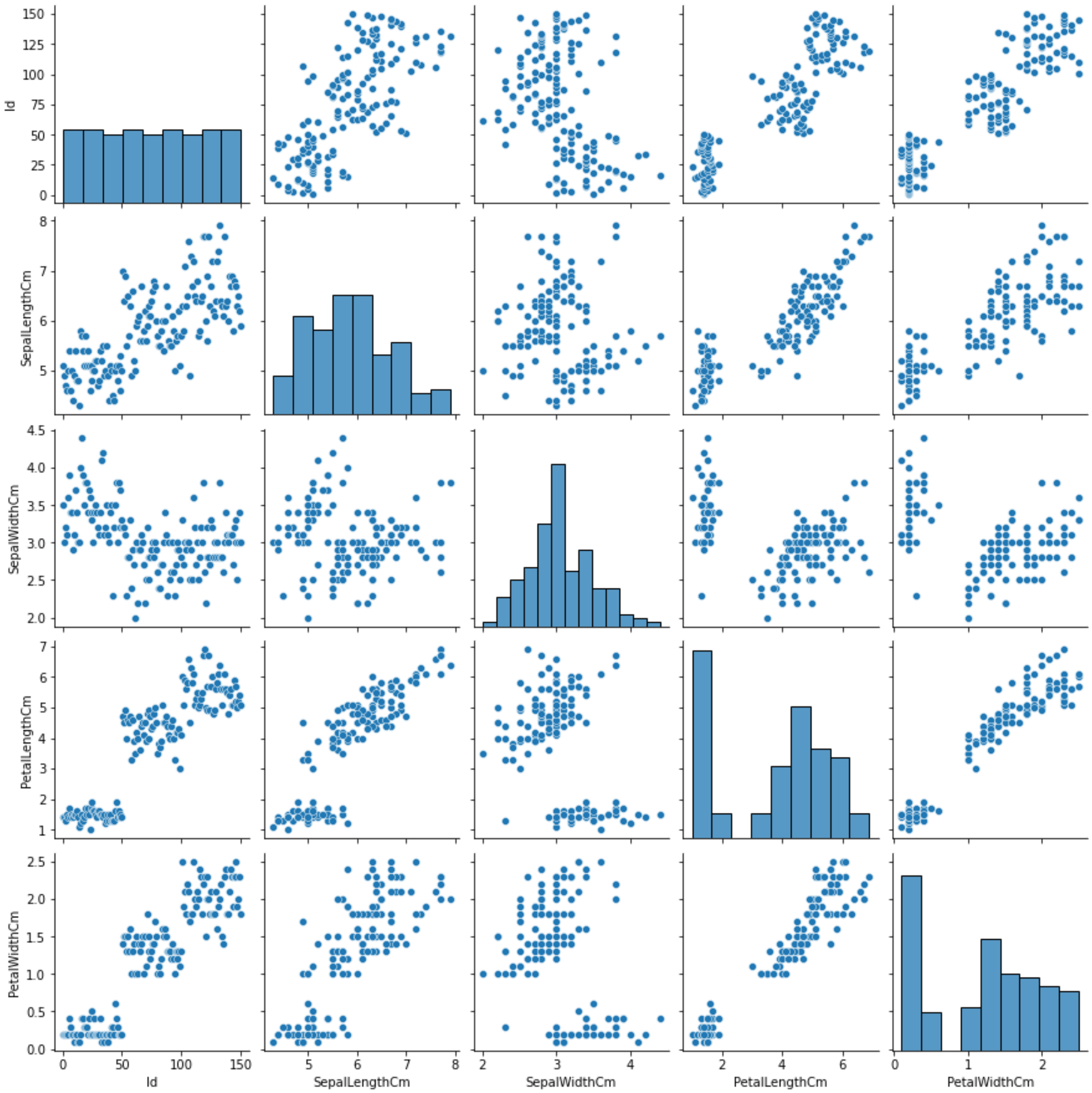
Out[3]:

	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 [ ]:

In [11]:
```
sns.pairplot(data)
```

Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x1d25d615340>
```



so i think that SepalLength and petalWidth is more effective

In [30]:
```
data['Species'].unique()
```

Out[30]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [ ]:

In [ ]:

In [19]:

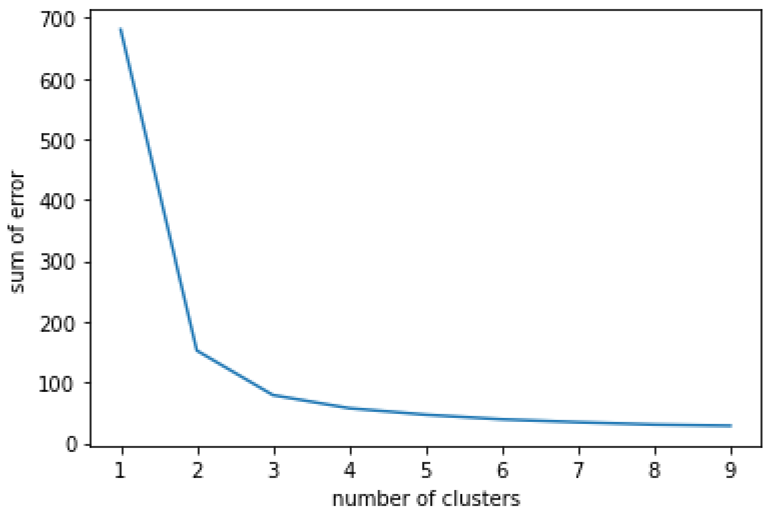
Elbow blot to know number of clusters

In [20]:
```
sse=[]
k_rng=range(1,10)
for k in k_rng:
    km=KMeans(n_clusters= k)
    km.fit(data[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']])
    sse.append(km.inertia_)
```

```
C:\Users\Yaseen\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: Use
rWarning: 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 environ
ment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [22]:
```
plt.xlabel('number of clusters')
plt.ylabel('sum of error')
plt.plot(k_rng,sse)
```

Out[22]: `[<matplotlib.lines.Line2D at 0x1d25f88a160>]`



We made sure that the number of clusters is 3

Apply KMean on Iris

In [42]:
```
km=KMeans(n_clusters=3,max_iter=300,random_state=11,init='k-means++')
y_predicted=km.fit_predict(data[['SepalLengthCm','SepalWidthCm','PetalLengthCm','
y_predicted
```

Out[42]:
```
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, 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, 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])
```

In [27]:
```
data['cluster']=y_predicted
data.tail()
```

Out[27]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	cluster
145	146	6.7	3.0	5.2	2.3	Iris-virginica	2
146	147	6.3	2.5	5.0	1.9	Iris-virginica	0
147	148	6.5	3.0	5.2	2.0	Iris-virginica	2
148	149	6.2	3.4	5.4	2.3	Iris-virginica	2
149	150	5.9	3.0	5.1	1.8	Iris-virginica	0

In [28]:
```
df1=data[data['cluster']==0]
df2=data[data['cluster']==1]
df3=data[data['cluster']==2]
```

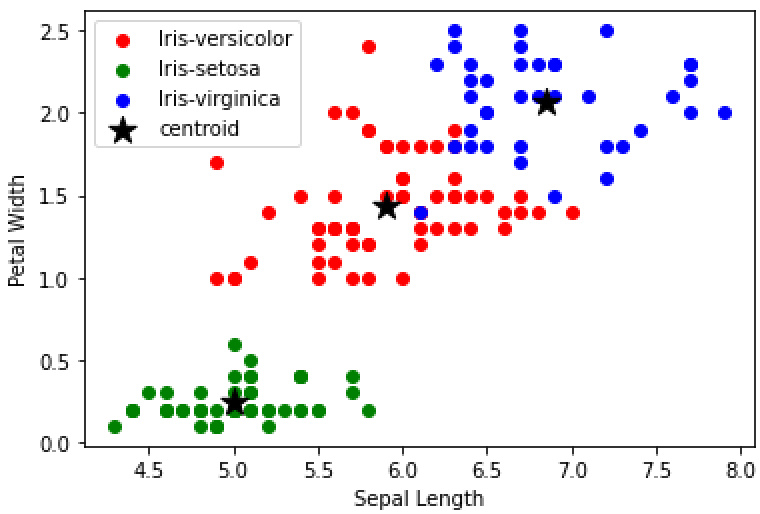
In [43]:
```
km.cluster_centers_
```

Out[43]:
```
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]])
```

Visualize Clusters

In [46]:
```
plt.scatter(df1['SepalLengthCm'],df1['PetalWidthCm'],color='r',label='Iris-versic
plt.scatter(df2['SepalLengthCm'],df2['PetalWidthCm'],color='g',label='Iris-setosa
plt.scatter(df3['SepalLengthCm'],df3['PetalWidthCm'],color='b',label='Iris-virgir
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,3],label='centroid',ma
plt.legend()
plt.xlabel('Sepal Length')
plt.ylabel('Petal Width')
```

Out[46]:
```
Text(0, 0.5, 'Petal Width')
```



other solution using PCA

In [82]:
```
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
newdata=pca.fit_transform(data[['SepalLengthCm','SepalWidthCm','PetalLengthCm','P
newdata=pd.DataFrame(newdata)
newdata.tail()
```

Out[82]:

	0	1
145	1.944017	0.187415
146	1.525664	-0.375021
147	1.764046	0.078519
148	1.901629	0.115877
149	1.389666	-0.282887

Apply KMean

In [81]:
```
km=KMeans(n_clusters=3,max_iter=300,random_state=11,init='k-means++')
y_predicted=km.fit_predict(newdata[[0,1]])
y_predicted
```

Out[81]:
```
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, 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, 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])
```

In [84]:
```
newdata['clusters']=y_predicted
newdata
```

Out[84]:

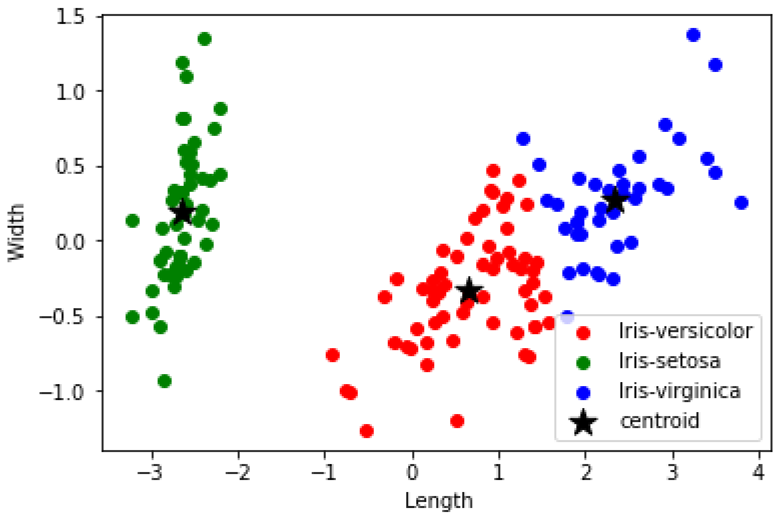
	0	1	clusters
0	-2.684207	0.326607	1
1	-2.715391	-0.169557	1
2	-2.889820	-0.137346	1
3	-2.746437	-0.311124	1
4	-2.728593	0.333925	1
...	...	...	...
145	1.944017	0.187415	2
146	1.525664	-0.375021	0
147	1.764046	0.078519	2
148	1.901629	0.115877	2
149	1.389666	-0.282887	0

150 rows × 3 columns

In [85]:
```
df01=newdata[newdata['clusters']==0]
df02=newdata[newdata['clusters']==1]
df03=newdata[newdata['clusters']==2]
```

In [87]:
```
plt.scatter(df01[0],df01[1],color='r',label='Iris-versicolor')
plt.scatter(df02[0],df02[1],color='g',label='Iris-setosa')
plt.scatter(df03[0],df03[1],color='b',label='Iris-virginica')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],label='centroid',ma
plt.legend()
plt.xlabel(' Length')
plt.ylabel(' Width')
```

Out[87]: Text(0, 0.5, ' Width')



In [4]:
```
print(np.array(4.8))
```

4.8

In [ ]: