THE SPARKS FOUNDATION - GRIP November 2022

Data Science & Business Analytics

Name: Roshni Verma

Task 2: Prediction using Unsupervised ML

Dataset: https://bit.ly/3kXTdox

Importing important libraries

In [18]:

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

In [19]:

```
df = pd.read_csv("Iris.csv")
df.head(5)
```

Out[19]:

	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 [20]:

```
df_n = df.drop(['Id','Species'], axis='columns')
df_n.head()
```

Out[20]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
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 [21]:

```
df_n.shape
```

Out[21]:

```
(150, 4)
```

In [22]:

```
df_n.isnull().sum()
```

Out[22]:

```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
dtype: int64
```

In [23]:

```
df_n.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
dtypes: float64(4)
memory usage: 4.8 KB
```

Train the model

In [25]:

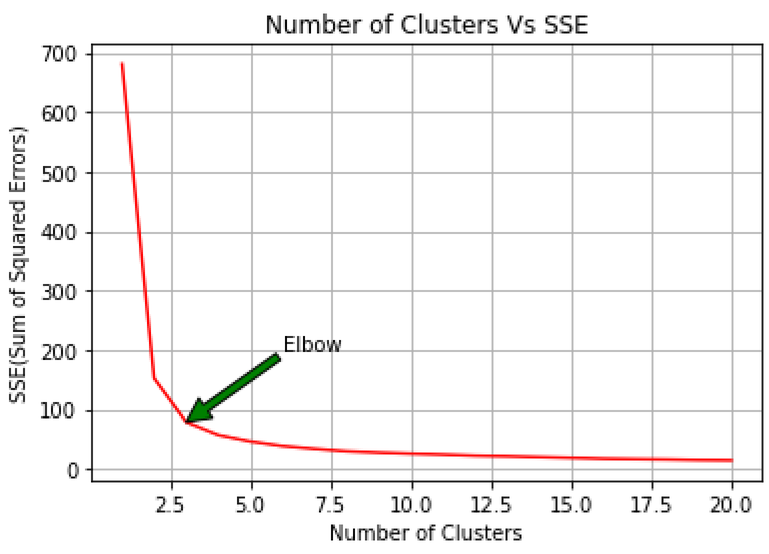
```
from sklearn.cluster import KMeans
x = df_n.iloc[:, :].values
sse = []

for i in range(1,21):
    model = KMeans(n_clusters = i, init = 'k-means++', max_iter = 250, n_init = 15, random_state = 0)
    model.fit(x)
    sse.append(model.inertia_)

print(sse)
```

```
C:\Users\sony\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there a
re less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
[680.8243999999996, 152.36870647733915, 78.94084142614601, 57.317873214285726, 46.535582051282034, 38.930963049671746, 34.190687924796634, 30.063874432733137,
27.84235606060608, 26.04820224804435, 24.53046205587498, 22.667550324675336, 21.67477651515153, 20.300946969696966, 19.0484107004107, 17.638879870129884, 16.8
7133802308803, 16.293817496229263, 15.329051587301597, 14.824639249639258]
```

Visualization

In [26]:

```
plt.plot(range(1,21),sse, color = 'r')
plt.title('Number of Clusters Vs SSE')
plt.xlabel('Number of Clusters')
plt.ylabel('SSE(Sum of Squared Errors)')
plt.annotate('Elbow', xytext=(6,200), xy=(3,79), arrowprops={'facecolor':'green'})
plt.grid()
plt.show()
```



In [28]:

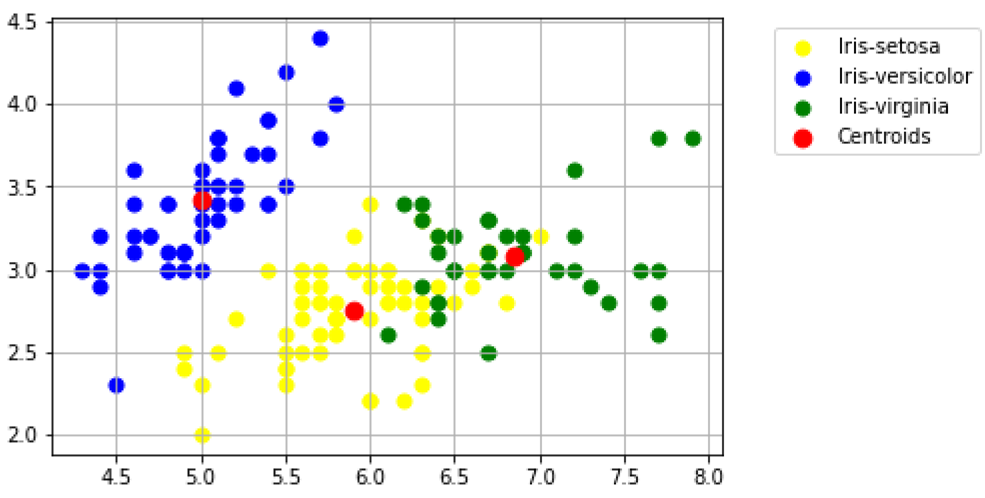
```
model =KMeans(n_clusters = 3, init = 'k-means++', max_iter = 250, n_init = 15, random_state = 0)
y = model.fit_predict(x)
y
```

Out[28]:

```
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])
```

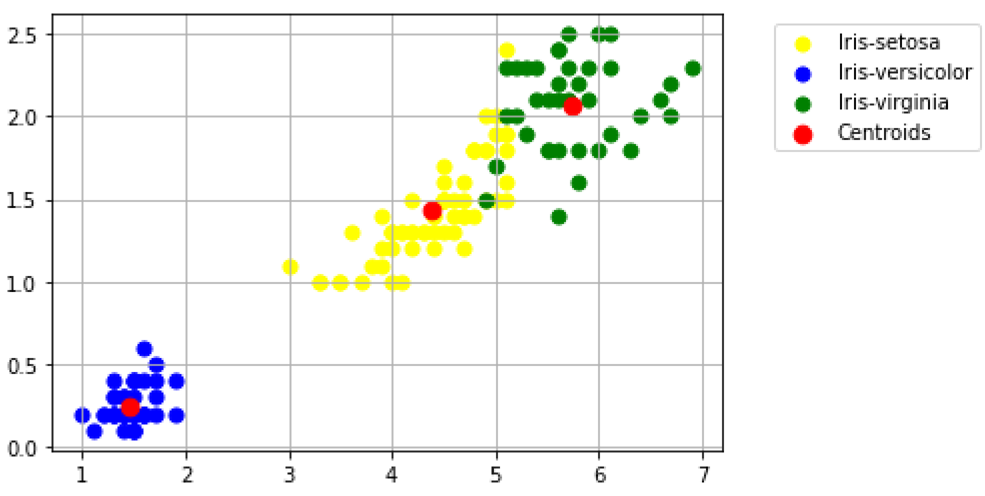
In [29]:

```
plt.scatter(x[y == 0, 0], x[y == 0, 1], s = 50, c = 'yellow', label = 'Iris-setosa')
plt.scatter(x[y == 1, 0], x[y == 1, 1], s =50, c = 'blue', label = 'Iris-versicolor')
plt.scatter(x[y == 2, 0], x[y ==2, 1], s = 50, c = 'green', label = 'Iris-virginia')

plt.scatter(model.cluster_centers_[:, 0], model.cluster_centers_[:, 1], s = 75, c = 'red', label = 'Centroids')

plt.legend(loc=1, bbox_to_anchor= (1.4, 1))
plt.grid()
```



In [30]:

```
plt.scatter(x[y == 0, 2], x[y == 0, 3], s = 50, c = 'yellow', label = 'Iris-setosa')
plt.scatter(x[y == 1, 2], x[y == 1, 3], s =50, c = 'blue', label = 'Iris-versicolor')
plt.scatter(x[y == 2, 2], x[y ==2, 3], s = 50, c = 'green', label = 'Iris-virginia')

plt.scatter(model.cluster_centers_[:, 2], model.cluster_centers_[:, 3], s = 75, c = 'red', label = 'Centroids')

plt.legend(loc=1, bbox_to_anchor= (1.4, 1))
plt.grid()
```



Observations: Optimum Number of Clusters - 3