GRIP:The Sparks Foundation ¶

Data Science and Business Analytics Intern

Author: Sudeshna Saha

In [1]:

```
#Importing all necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
```

In [2]:

```
#load the dataset
df=pd.read_csv("Iris.csv")
```

In [3]:

```
df.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 [4]:

```
#check the target class
df["Species"].value_counts()
```

Out[4]:

```
Iris-virginica     50
Iris-setosa        50
Iris-versicolor    50
Name: Species, dtype: int64
```

In [5]:

```
df.shape
```

Out[5]:

```
(150, 6)
```

In [6]:

```
df.describe()
```

Out[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [7]:

```
df.info()
```

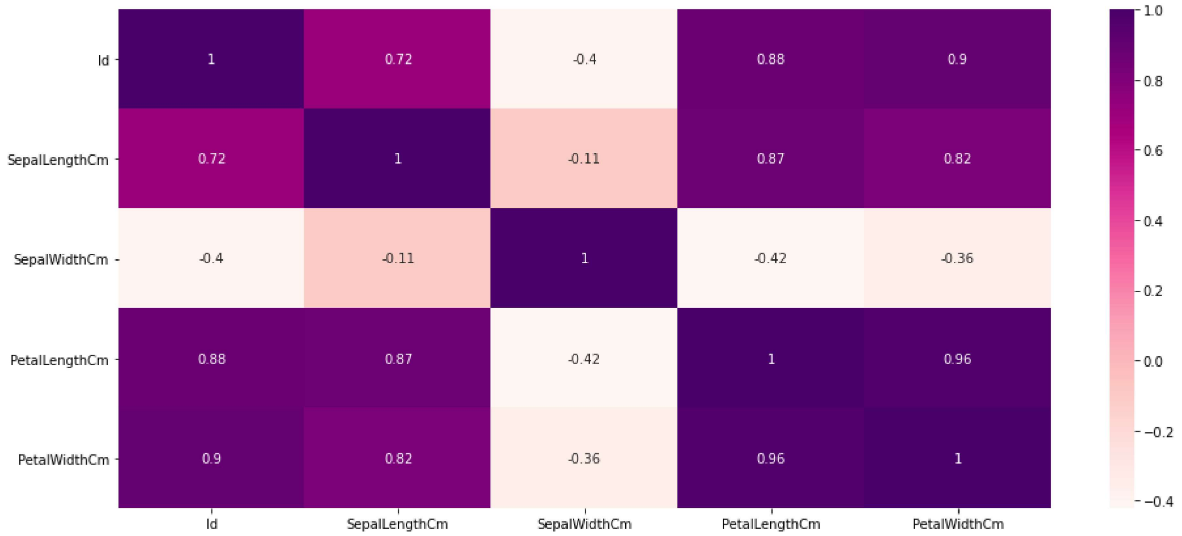
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [8]:

```
#Finding the correlation
plt.figure(figsize=(16,7))
sns.heatmap(df.corr(),cmap="RdPu",annot=True)
```

Out[8]:

```
<AxesSubplot:>
```



K-Means Clustering

In [9]:

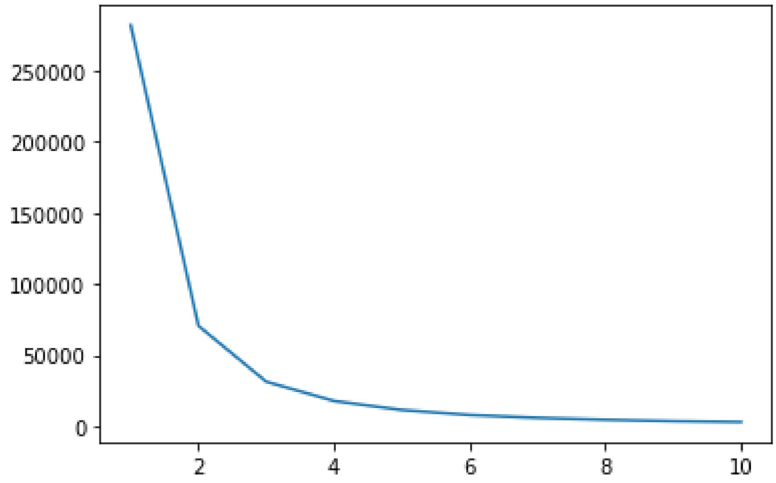
```
#Finding the optimum number of clusters
x=df.iloc[:, [0,1,2,3]].values
```

In [10]:

```
wcss=[]
for i in range(1,11):
    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
```

In [11]:

```
plt.plot(range(1,11),wcss)
plt.show()
```



This is a elbow method.From this we can choose the number of clusters as 3.

In [12]:

```
kmeans=KMeans(n_clusters=3,init='k-means++',random_state=0)
y_kmeans=kmeans.fit_predict(x)
```

In [13]:

```
y_kmeans
```

Out[13]:

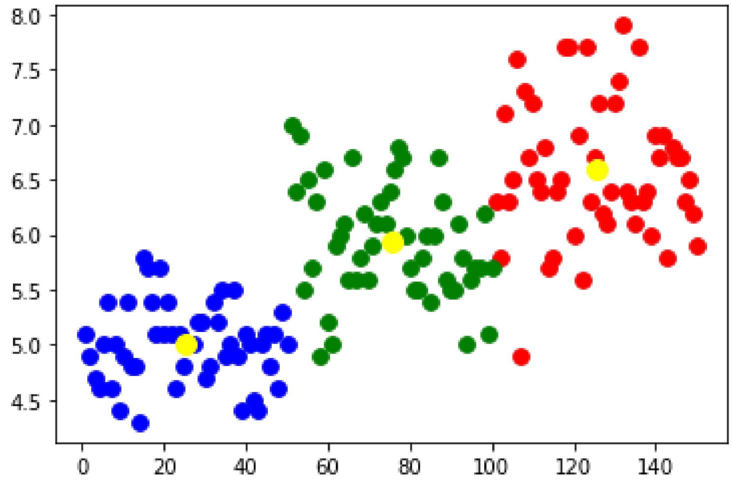
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
    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

In [14]:

```
#Visualizing the clusters-on the first two column
plt.scatter(x[y_kmeans==0,0],x[y_kmeans==0,1],s=60,c='red',label="Iris-setosa")
plt.scatter(x[y_kmeans==1,0],x[y_kmeans==1,1],s=60,c='blue',label="Iris-versicolor")
plt.scatter(x[y_kmeans==2,0],x[y_kmeans==2,1],s=60,c='green',label="Iris-verginica")
#Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=100,c='yellow',labe
```

Out[14]:

```
<matplotlib.collections.PathCollection at 0x1db58917310>
```



In [ ]: