The Sparks Foundation - GRIP - Data Science and Business Analytics

TASK 2 : Prediction using unsupervised ML

Problem Statement :

Predict the optimum number of clusters and represent it visualy

NAME: Divyashree K

Importing Libraries

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

In [4]:
```
import io
%cd "E:\Internship\Spark Foundation"
```

```
E:\Internship\Spark Foundation
```

Read data

In [5]:
```
data = pd.read_csv("iris.csv")
data
```

Out[5]:

	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
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [6]:
```
data.shape
```

Out[6]:
```
(150, 6)
```

In [7]:
```
data.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]:
```
data.columns
```

Out[8]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [9]:
```
# droping id columns

data= data.drop(['Id'],axis=1)
data
```

Out[9]:

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

150 rows × 5 columns

In [10]:
```
data.columns
```

Out[10]:
```
Index(['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [11]:
```
# check for null values

data.isnull().sum().sort_values(ascending = False)
```

Out[11]:
```
Species          0
PetalWidthCm     0
PetalLengthCm    0
SepalWidthCm     0
SepalLengthCm    0
dtype: int64
```

In [12]:
```
# check for duplicate values

print("number of duplicate rows: ",data.duplicated().sum())
```

```
number of duplicate rows:  3
```

In [13]:
```
# drop duplicate rows
data.drop_duplicates(inplace=True)
data.shape[0] #shape[0] shows number of rows
```

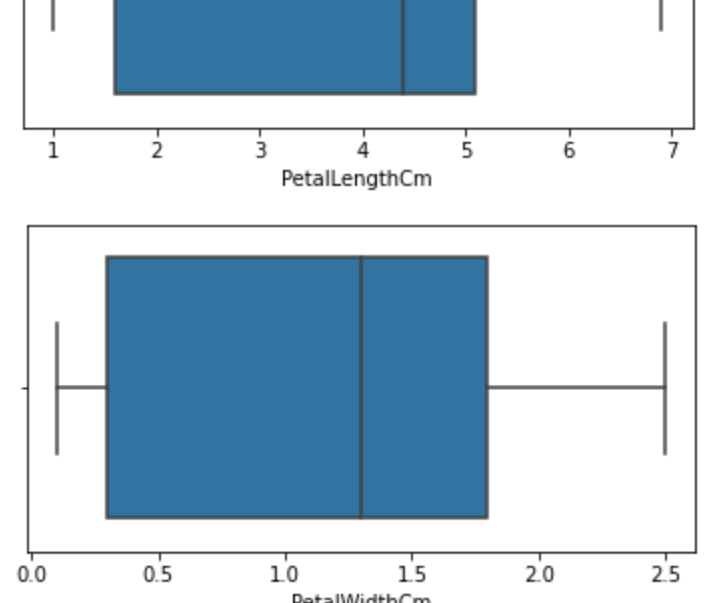
Out[13]:
```
147
```

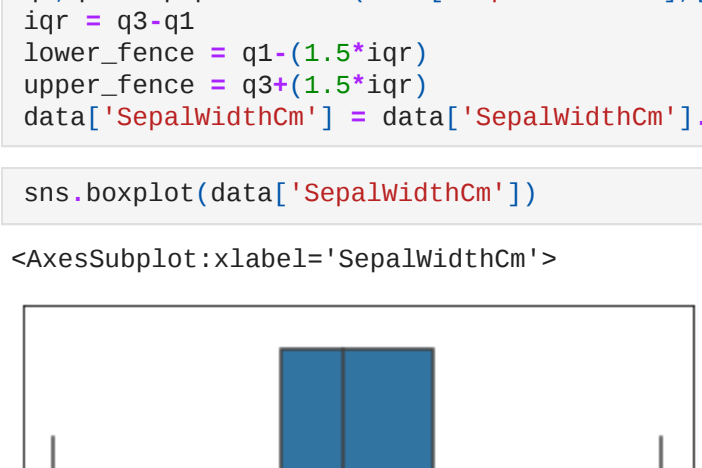
In [14]:
```
data.shape[1] # shape[1] shows number columns
```

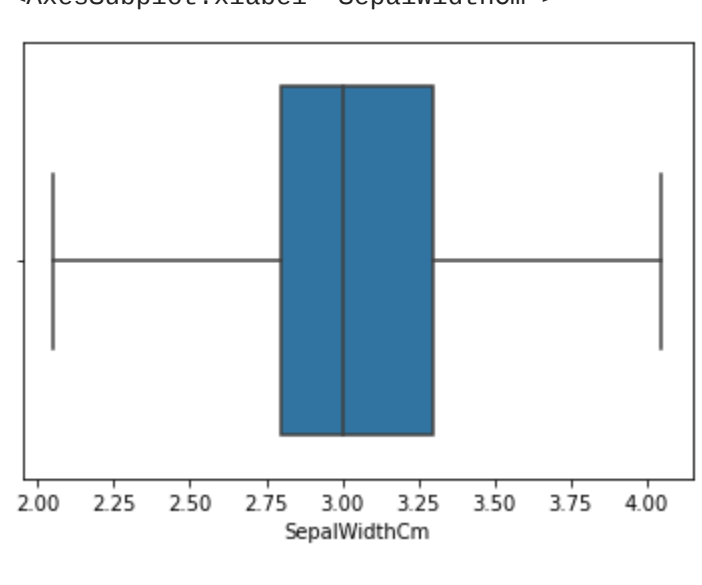
Out[14]:
```
5
```

In [15]:
```
# check for outliers in numerical data

for i in data.columns:
    if data[i].dtypes == "float64":
        plt.figure(figsize=(6,3))
        sns.boxplot(data[i])
        plt.show()
```





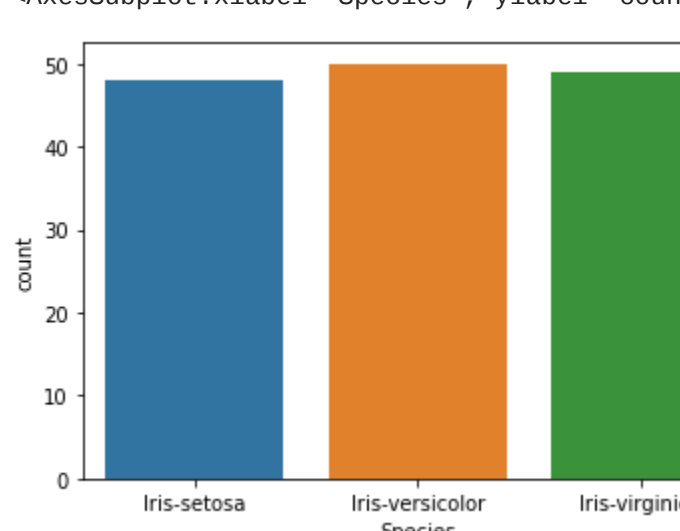


In [16]:
```
# treating outliers in sepalwidth

q1,q3 = np.percentile(data['SepalWidthCm'],[25,75])
iqr = q3-q1
lower_fence = q1-(1.5*iqr)
upper_fence = q3+(1.5*iqr)
data['SepalWidthCm'] = data['SepalWidthCm'].apply(lambda x: upper_fence if x>upper_fence else lower_fence if x<lower_fence else x)
```

In [17]:
```
sns.boxplot(data['SepalWidthCm'])
```

Out[17]:
```
<AxesSubplot:xlabel='SepalWidthCm'>
```

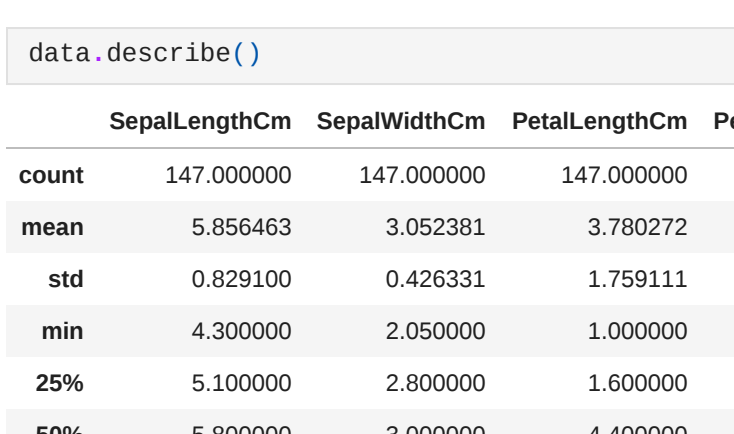


Understanding the data

In [18]:
```
# Target class

data.Species.value_counts()
sns.countplot(data.Species)
```

Out[18]:
```
<AxesSubplot:xlabel='Species', ylabel='count'>
```



In [19]:
```
data.describe()
```

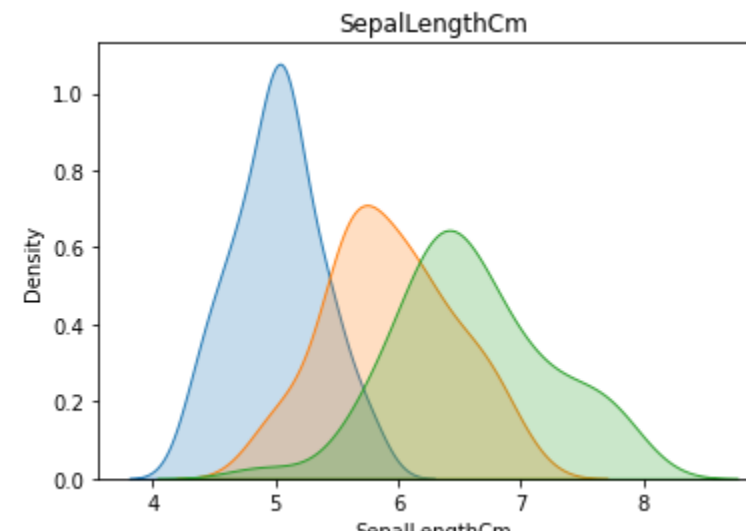
Out[19]:

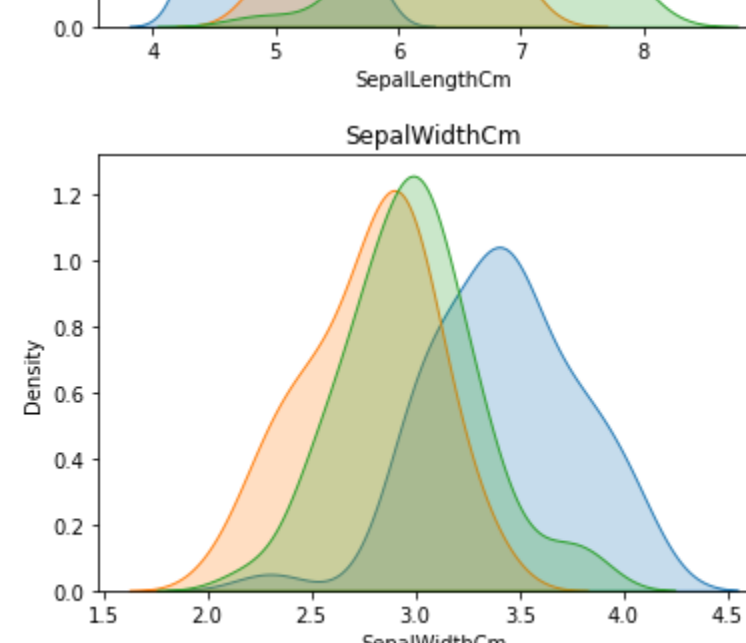
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	147.000000	147.000000	147.000000	147.000000
mean	5.856463	3.052381	3.780272	1.208844
std	0.829100	0.426331	1.759111	0.757874
min	4.300000	2.050000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.400000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.050000	6.900000	2.500000

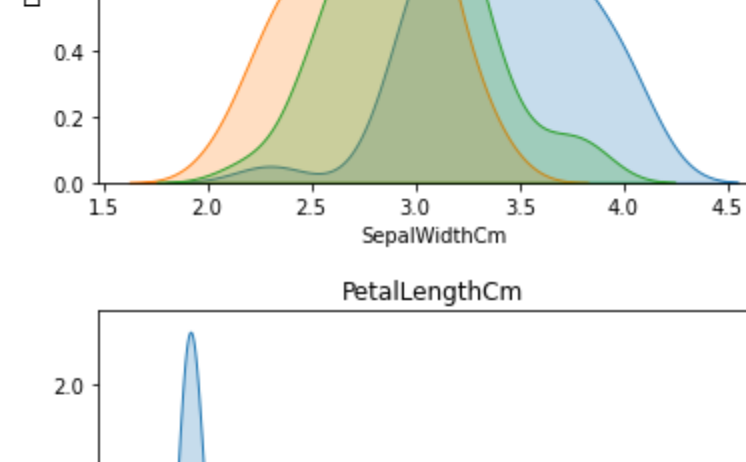
In [20]:
```
data.Species.unique()
```

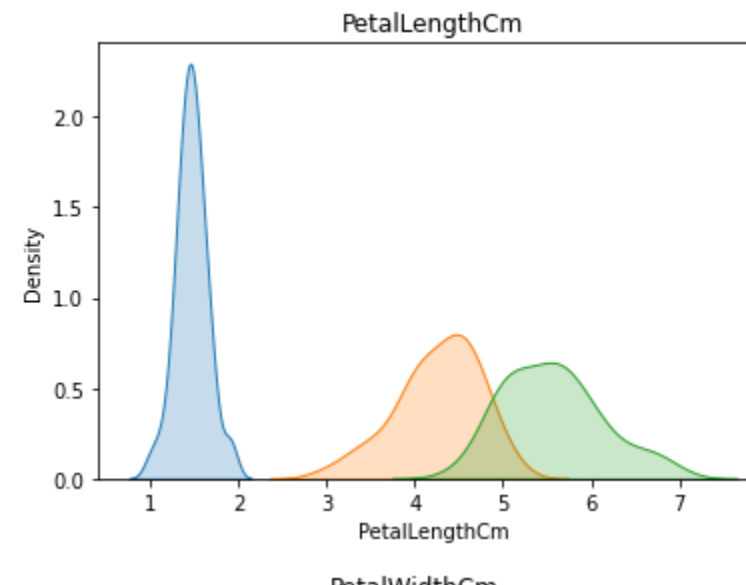
Out[20]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [21]:
```
#Distribution of features by Species
for i in data.columns[ : -1]:
    sns.kdeplot(data = data.loc[data.Species == 'Iris-setosa'][i],label = 'Iris-setosa',shade = True)
    sns.kdeplot(data = data.loc[data.Species == 'Iris-versicolor'][i], label = 'Iris-versicolor', shade =True)
    sns.kdeplot(data = data.loc[data.Species == 'Iris-virginica'][i], label = 'Iris-virginica', shade = True)
    plt.title(i)
    plt.show()
```









Correlation Matrix

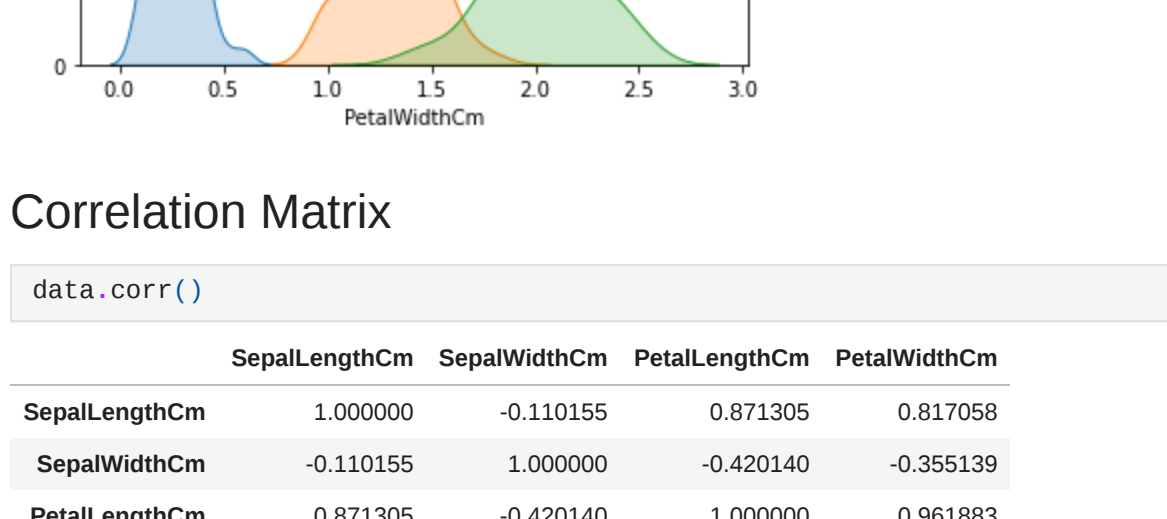
In [22]:
```
data.corr()
```

Out[22]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.110155	0.871305	0.817058
SepalWidthCm	-0.110155	1.000000	-0.420140	-0.355139
PetalLengthCm	0.871305	-0.420140	1.000000	0.961883
PetalWidthCm	0.817058	-0.355139	0.961883	1.000000

In [23]:
```
plt.figure(figsize=(10,5))
sns.heatmap(abs(data.corr()),cmap ='GnBu', annot=True)
```

Out[23]:
```
<AxesSubplot:>
```



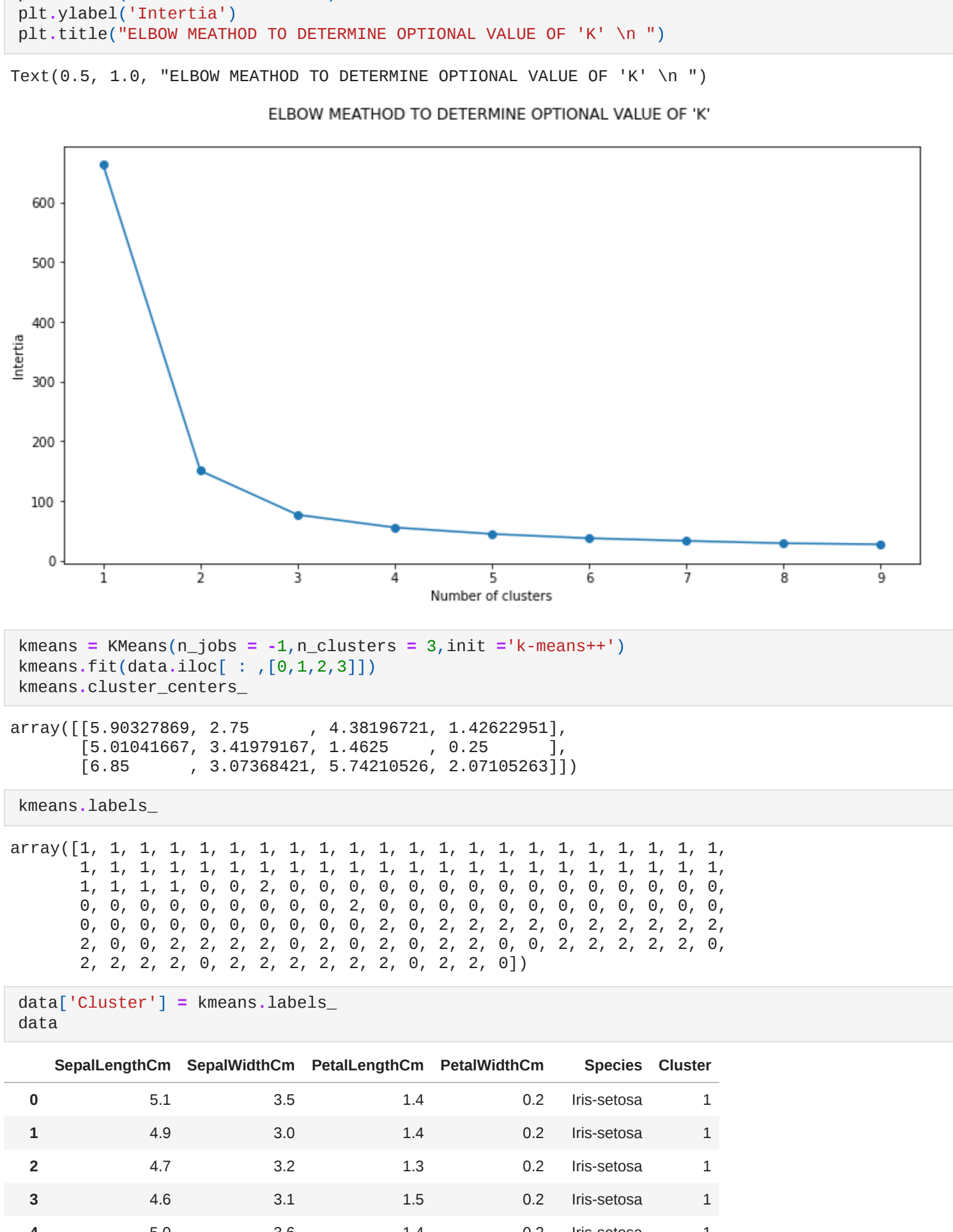
K-Mean clustering

In [24]:
```
from sklearn.cluster import KMeans
```

In [25]:
```
SSE =[]
for i in range(1,10):
    kmeans = KMeans(n_jobs = -1,n_clusters=i,init = 'k-means++')
    kmeans.fit(data.iloc[ : ,[0,1,2,3]])
    SSE.append(kmeans.inertia_)
```

In [26]:
```
df = pd.DataFrame({'Cluster':range(1,10), 'SSE':SSE})
plt.figure(figsize=(12,6))
plt.plot(df['Cluster'],df['SSE'],marker = 'o')
plt.xlabel('Number of clusters')
plt.ylabel('Intertia')
plt.title("ELBOW MEATHOD TO DETERMINE OPTIONAL VALUE OF 'K' \n ")
```

Out[26]:
```
Text(0.5, 1.0, "ELBOW MEATHOD TO DETERMINE OPTIONAL VALUE OF 'K' \n ")
```



In [27]:
```
kmeans = KMeans(n_jobs = -1,n_clusters = 3,init ='k-means++')
kmeans.fit(data.iloc[ : ,[0,1,2,3]])
kmeans.cluster_centers_
```

Out[27]:
```
array([[5.90327869, 2.75      , 4.38196721, 1.42622951],
       [5.01041667, 3.41979167, 1.4625    , 0.25      ],
       [6.85      , 3.07368421, 5.74210526, 2.07105263]])
```

In [28]:
```
kmeans.labels_
```

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, 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, 2, 2, 2, 0, 2, 2, 0])
```

In [29]:
```
data['Cluster'] = kmeans.labels_
data
```

Out[29]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	Cluster
0	5.1	3.5	1.4	0.2	Iris-setosa	1
1	4.9	3.0	1.4	0.2	Iris-setosa	1
2	4.7	3.2	1.3	0.2	Iris-setosa	1
3	4.6	3.1	1.5	0.2	Iris-setosa	1
4	5.0	3.6	1.4	0.2	Iris-setosa	1
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica	2
146	6.3	2.5	5.0	1.9	Iris-virginica	0
147	6.5	3.0	5.2	2.0	Iris-virginica	2
148	6.2	3.4	5.4	2.3	Iris-virginica	2
149	5.9	3.0	5.1	1.8	Iris-virginica	0

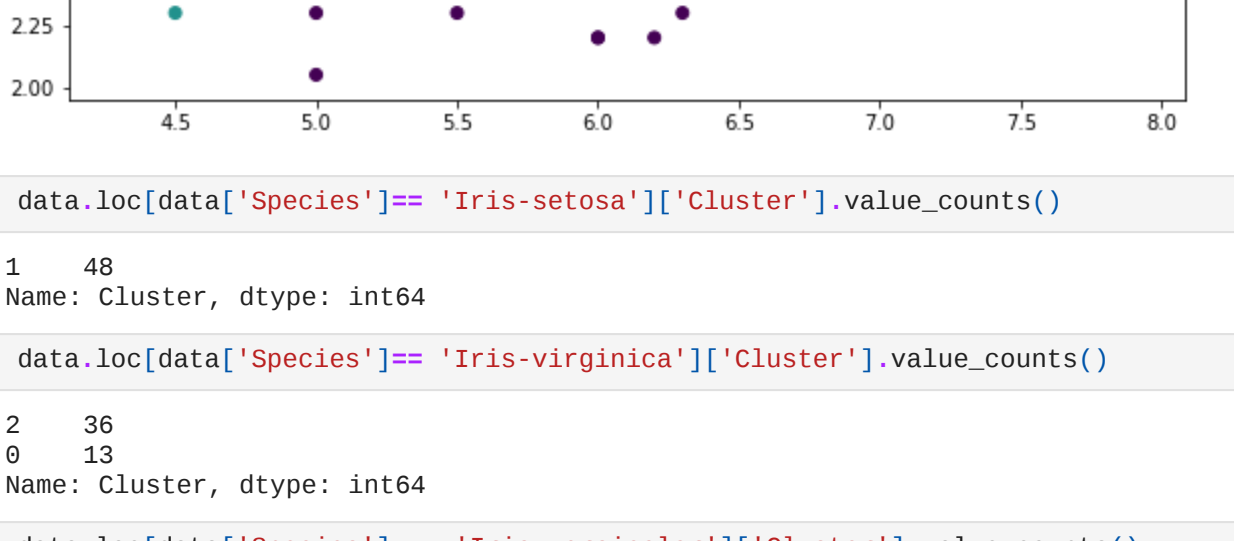
147 rows × 6 columns

In [30]:
```
display(data['Cluster'].value_counts(),data['Species'].value_counts())
```

```
0    61
1    48
2    38
Name: Cluster, dtype: int64
Iris-versicolor    50
Iris-virginica     49
Iris-setosa        48
Name: Species, dtype: int64
```

In [31]:
```
plt.figure(figsize = (10,5))
plt.scatter(data['SepalLengthCm'],data['SepalWidthCm'],c=data.Cluster)
plt.title('predicted cluster \n')
plt.scatter(kmeans.cluster_centers_[ :,0],kmeans.cluster_centers_[ : ,1],s = 200,c='red', label ='centroid')
plt.show
```

Out[31]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [32]:
```
data.loc[data['Species']== 'Iris-setosa']['Cluster'].value_counts()
```

Out[32]:
```
1    48
Name: Cluster, dtype: int64
```

In [33]:
```
data.loc[data['Species']== 'Iris-virginica']['Cluster'].value_counts()
```

Out[33]:
```
2    36
0    13
Name: Cluster, dtype: int64
```

In [34]:
```
data.loc[data['Species'] == 'Iris-versicolor']['Cluster'].value_counts()
```

Out[34]:
```
0    48
2     2
Name: Cluster, dtype: int64
```

In [35]:
```
data['Species_encoded']=data['Species'].apply(lambda x: 1 if x=='Iris-setosa' else 2 if x== 'Iris-virginica' else 0)
data
```

Out[35]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	Cluster	Species_encoded
0	5.1	3.5	1.4	0.2	Iris-setosa	1	1
1	4.9	3.0	1.4	0.2	Iris-setosa	1	1
2	4.7	3.2	1.3	0.2	Iris-setosa	1	1
3	4.6	3.1	1.5	0.2	Iris-setosa	1	1
4	5.0	3.6	1.4	0.2	Iris-setosa	1	1
...	...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica	2	2
146	6.3	2.5	5.0	1.9	Iris-virginica	0	2
147	6.5	3.0	5.2	2.0	Iris-virginica	2	2
148	6.2	3.4	5.4	2.3	Iris-virginica	2	2
149	5.9	3.0	5.1	1.8	Iris-virginica	0	2

147 rows × 7 columns

In [36]:
```
from sklearn.metrics import classification_report
print(classification_report(data['Species_encoded'],data['Cluster']))
```

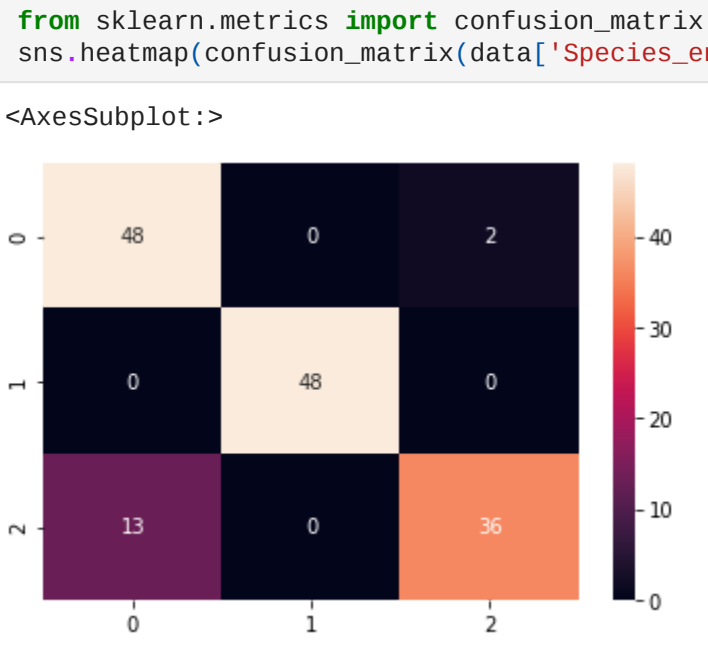
```
              precision    recall  f1-score   support

           0       0.79      0.96      0.86        50
           1       1.00      1.00      1.00        48
           2       0.95      0.73      0.83        49

    accuracy                           0.90       147
   macro avg       0.91      0.90      0.90       147
weighted avg       0.91      0.90      0.90       147
```

In [37]:
```
from sklearn.metrics import confusion_matrix
sns.heatmap(confusion_matrix(data['Species_encoded'],data['Cluster']),annot=True)
```

Out[37]:
```
<AxesSubplot:>
```



In [ ]: