TASK II: PREDICTION USING SUPERVISED MACHINE LEARNING

AUTHOR: NEHA HRISHIKESH DEODHAR

DATASET USED FOR THE FOLLOWING PROJECT: :https://bit.ly/3kXTdox

OVERVIEW: IRIS DATASET, PREDICTS THE OPTIMUM NUMBER OF CLUSTERS AND REPRESENT IT VISUALLY

In [1]:

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn import datasets
```

In [2]:

```
iris=pd.read_csv("iris.csv")
iris.head()
```

Out[2]:

	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]:

```
iris.shape
```

Out[4]:

```
(150, 6)
```

In [17]:

```
iris.columns
```

Out[17]:

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

In [5]:

```
iris.dtypes
```

Out[5]:

```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
```

```
PetalWidthCm     float64
Species           object
dtype: object
```

In [6]:

```
iris.isnull().sum()
```

Out[6]:

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

In [7]:

```
iris.duplicated().sum()
```

Out[7]:

```
0
```

In [8]:

```
iris.describe
```

Out[8]:

```
<bound method NDFrame.describe of       Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  P
etalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]>
```

In [9]:

```
categorical=['Species']
continuous=['Id','SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']
```

In [10]:

```
iris[continuous].describe()
```

Out[10]:

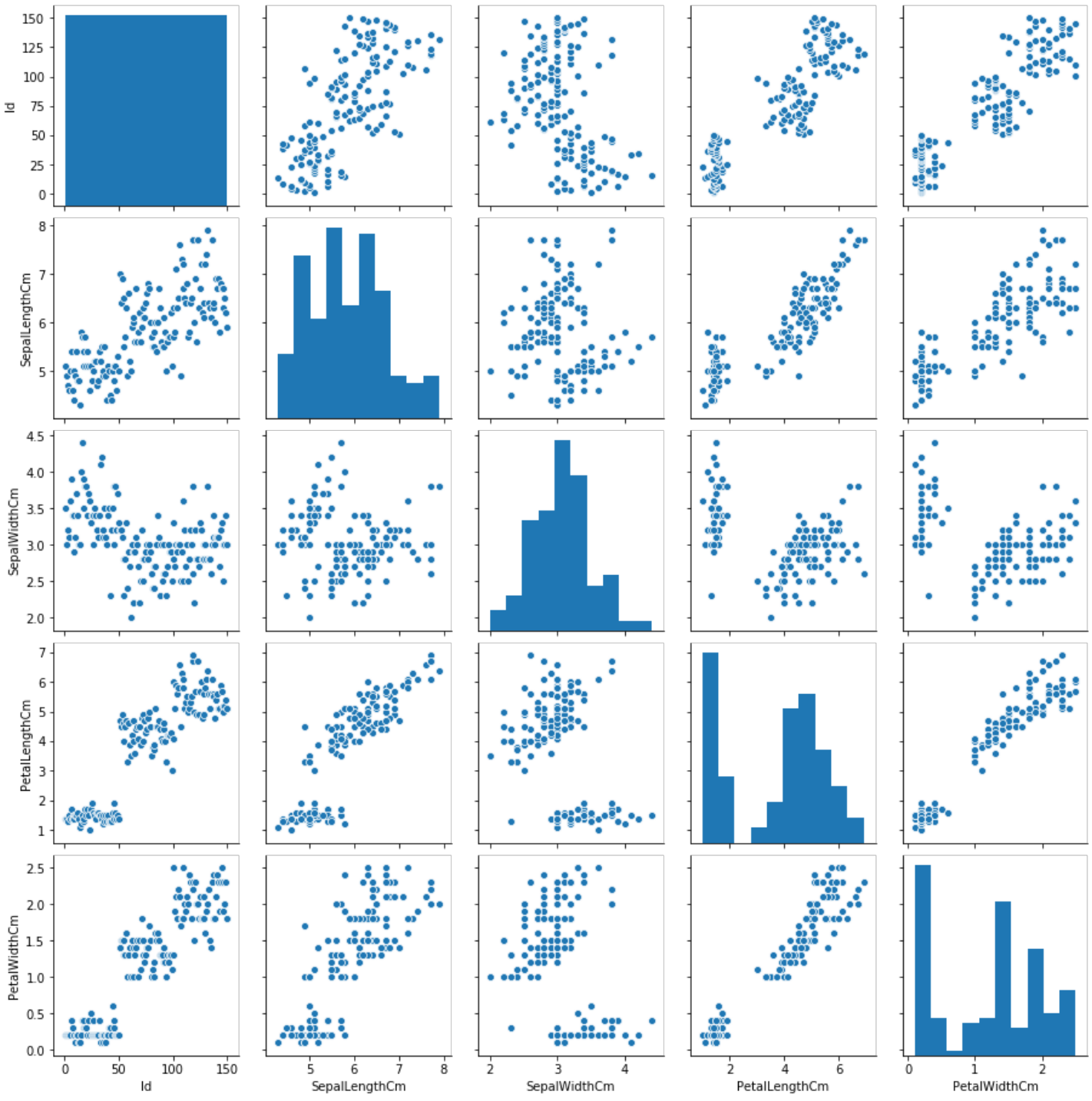
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	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 [11]:

```
sns.pairplot(iris)
```

Out[11]:

```
<seaborn.axisgrid.PairGrid at 0x284a488edc8>
```

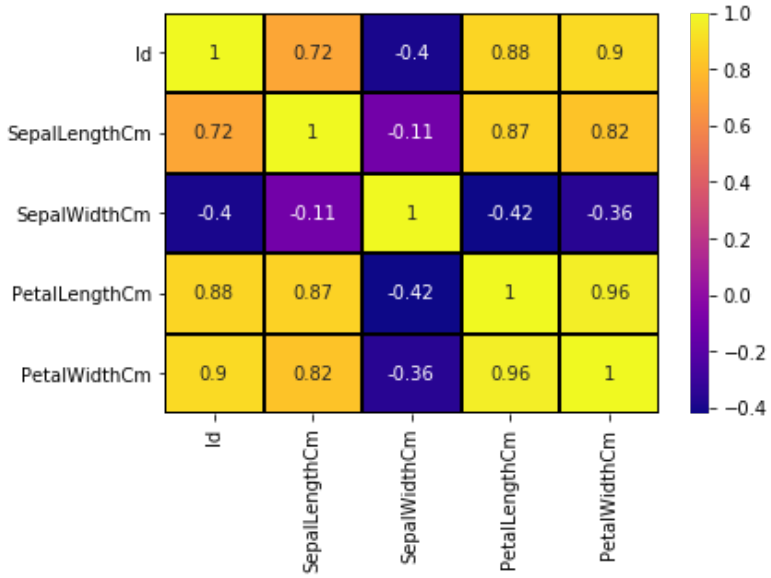


In [12]:

```
tc = iris.corr()
sns.heatmap(tc, annot = True, cmap ='plasma',
            linecolor ='black', linewidths = 1)
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x284a5b5fd08>
```

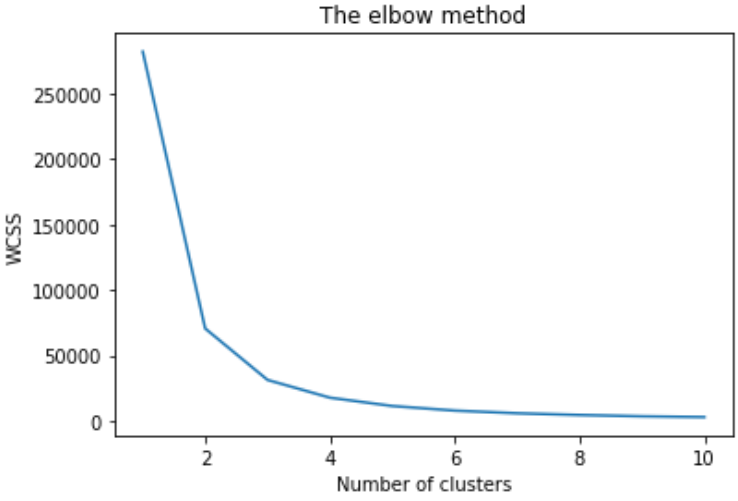


In [14]:

```
# Finding the optimum number of clusters for k-means classification

x = iris.iloc[:, [0, 1, 2, 3]].values

from sklearn.cluster import KMeans
wcss = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',
                    max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
```

In [15]:

```
# Plotting the results onto a line graph,
# `allowing us to observe 'The elbow'
plt.plot(range(1, 11), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') # Within cluster sum of squares
plt.show()
```



In [16]:

```
# Applying kmeans to the dataset / Creating the kmeans classifier
kmeans = KMeans(n_clusters = 3, init = 'k-means++',
                max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(x)
# Visualising the clusters - On the first two columns
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1],
            s = 100, c = 'red', label = 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1],
            s = 100, c = 'purple', label = 'Iris-versicolour')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1],
            s = 100, c = 'yellow', label = 'Iris-virginica')

# Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1],
            s = 100, c = 'red', label = 'Centroids')

plt.legend()
```

Out[16]:

```
<matplotlib.legend.Legend at 0x284a6f09d88>
```