Surabhi Singh

The Spark Foundation

Iris Task

Prediction of the optimum number of cluster and visual representation

Exploratory Data Analysis

Python libraries being used

A. Data Analysis

In [2]:

```
import pandas as pd
import numpy as np
```

B. Data Visualization

In [4]:

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import datasets
```

C. Model Prediction

In [8]:

```
from sklearn import datasets
from sklearn.cluster import KMeans
```

Data Extraction

Importing Data from Iris Data

In [9]:

```
import types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your c
redentials.
# You might want to remove those credentials before you share the notebook.
```

```
client_df59ba2aae8f4254afb2ed76c5028249 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='C0K_2AXLl0Mc0wYapj_sm2un8kk7QWfw3bhgF6k6e3mK',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3-api.us-geo.objectstorage.service.networklayer.com')

body = client_df59ba2aae8f4254afb2ed76c5028249.get_object(Bucket='tsf-donotdelete-pr-spz8
z9iaocdhoc',Key='Iris.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df = pd.read_csv(body)
df.head()
```

Out[9]:

	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 [10]:

```
df.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]:

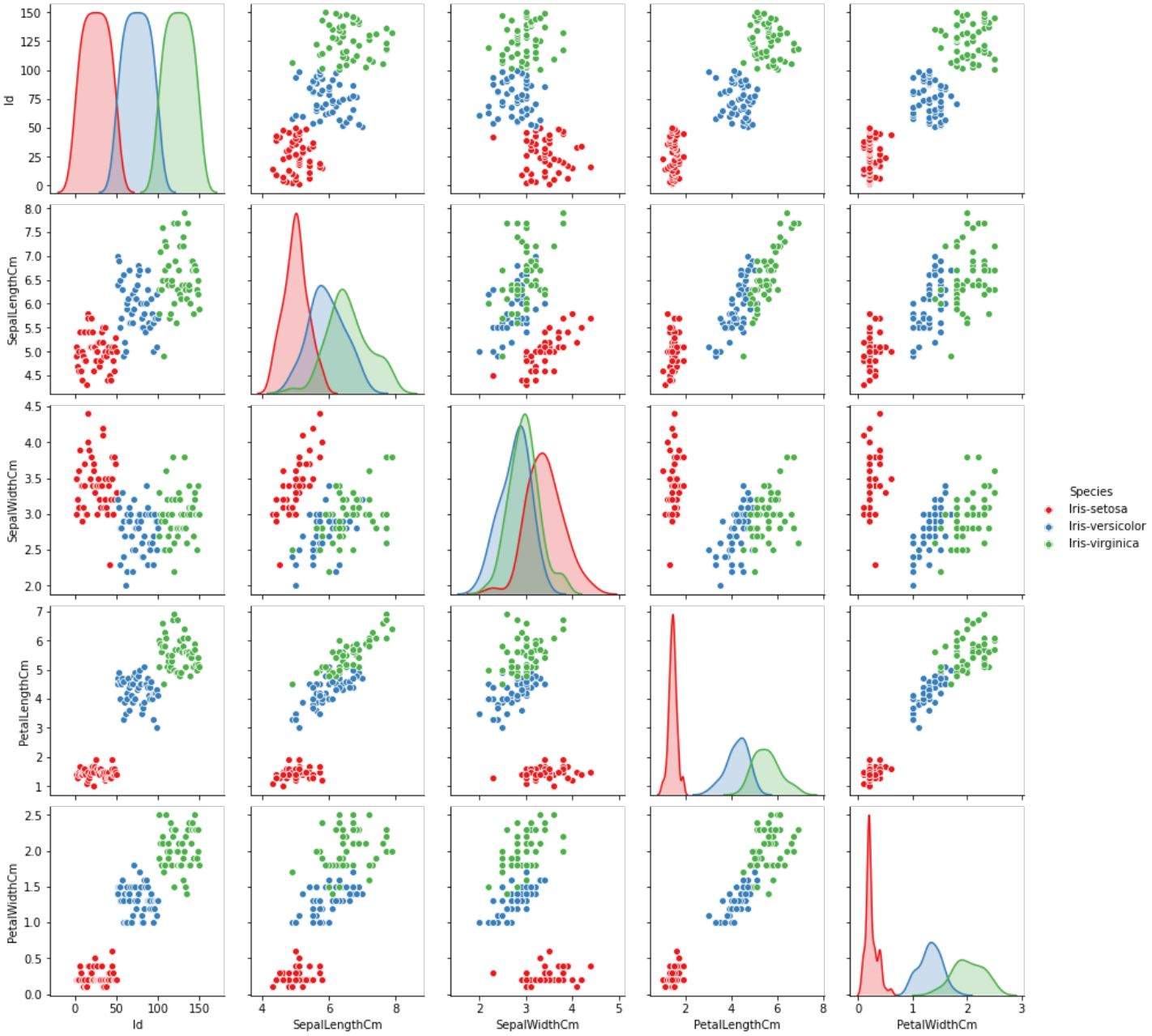
```
df.info()
```

```
<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
```

Data Visualization

In [13]:

```
sns.pairplot(data=df,hue="Species",palette="Set1")
plt.show()
```

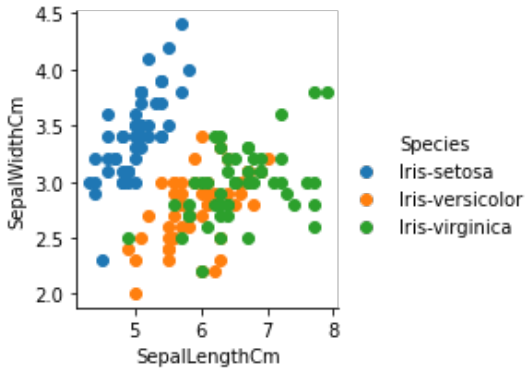


In [14]:

```
sns.FacetGrid(df, hue = "Species", height = 3)\
.map(plt.scatter, "SepalLengthCm","SepalWidthCm")\
.add_legend();
plt.show
```

Out[14]:

```
<function matplotlib.pyplot.show(*args, **kw)>
```



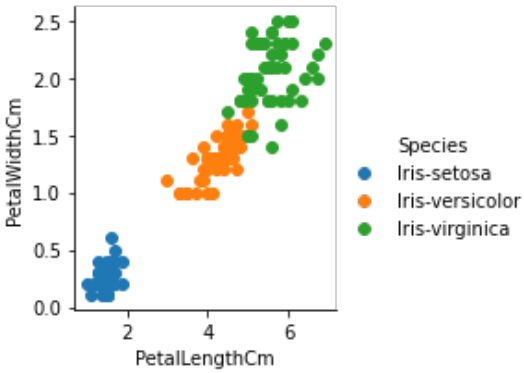
In [15]:

```
sns.FacetGrid(df, hue = "Species", height = 3)\
```

```
sns.FacetGrid(df, hue = "Species", height = 5)\
.map(plt.scatter, "PetalLengthCm","PetalWidthCm")\
.add_legend();
plt.show
```

Out[15]:

```
<function matplotlib.pyplot.show(*args, **kw)>
```

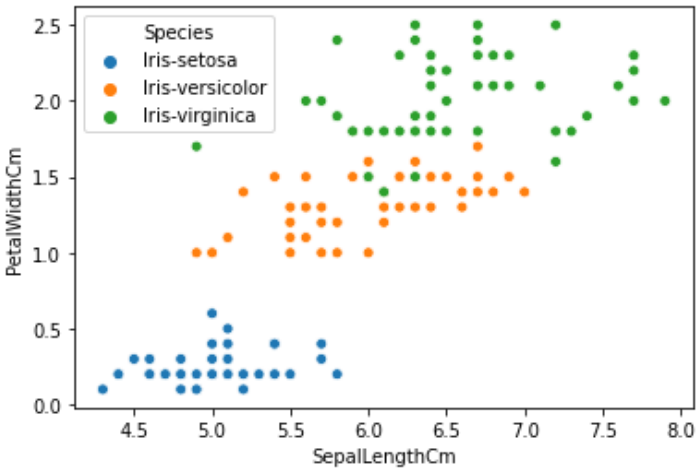


In [16]:

```
sns.scatterplot(x = 'SepalLengthCm', y = 'PetalWidthCm', hue = 'Species', data = df)
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fedf1c1c590>
```

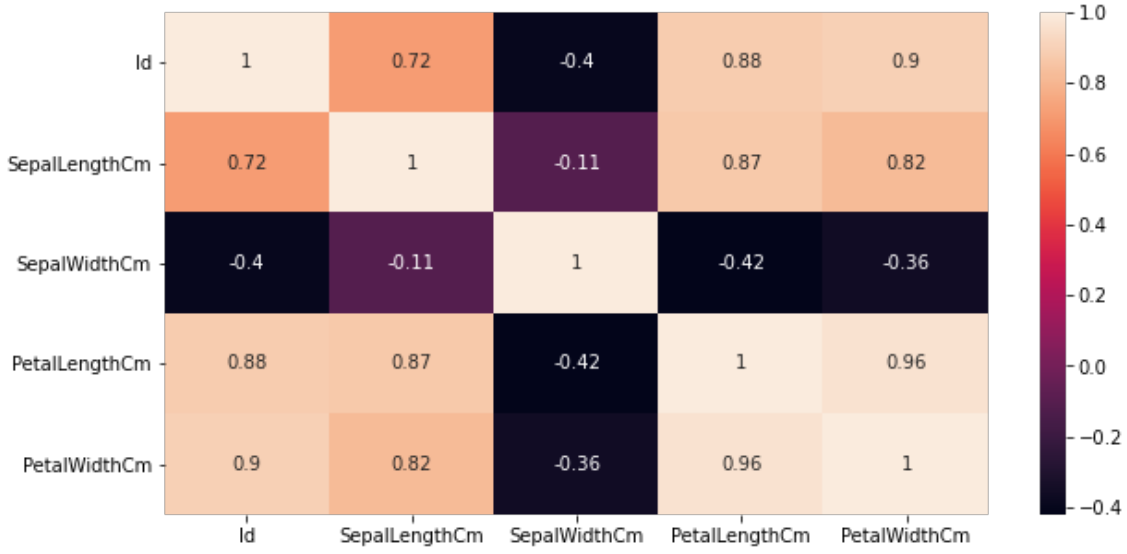


In [17]:

```
plt.figure(figsize=(10,5))
sns.heatmap(df.corr(), annot=True)
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fedfa1a0110>
```



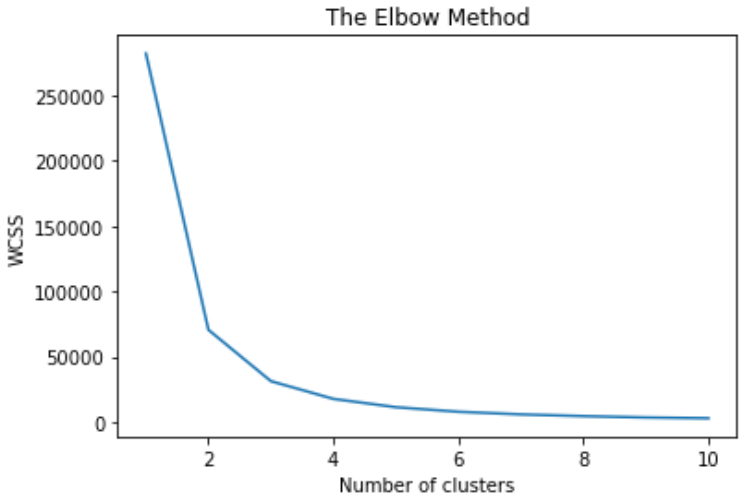
Elbow Mwthod to find out the optimal number of clusters

In [18]:

```
X = df.iloc[:, [0,1,2,3]].values
```

In [19]:

```
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



The elbow method' from the above graph, the optimum clusters is where the elbow occurs. This is when the within cluster sum of squares (WCSS) doesn't decrease significantly with every iteration.

From this we choose the number of clusters as '3'.

Training the K-Means model on the dataset

In [20]:

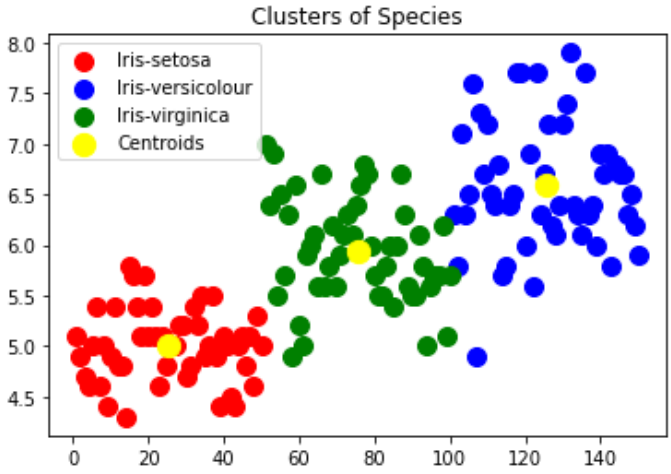
```
kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(X)
```

Clusters Visualization

In [21]:

```
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 = 'blue', label = 'Iris-versicolour')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Iris-virginica')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 150, c = 'yellow', label = 'Centroids')
```

```
plt.title('Clusters of Species')
plt.legend()
plt.show()
```



In [ ]: