GRIP - THE SPARK FOUNDATION

Data Science and Business Analytics Internship

Prediction using Unsupervised ML

TASK-2 : From the given 'Iris' dataset, predict the optimum number of clusters and represent it visually.

Author - Vijaya Lakshmi Basireddy

importing the libraries

In [1]:
```
# Import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
print("libraries are imported successfully")
```

```
libraries are imported successfully
```

load the dataset

In [2]:
```
#loading the given dataset
df = pd.read_csv('Iris.csv')
print("data is imported successfully")
```

```
data is imported successfully
```

In [3]:
```
#to get the type of data
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 [4]:
```
#describing the given dataset
df.describe()
```

Out[4]:

	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 [5]:
```
#printing the tail part of the given data
df.tail()
```

Out[5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
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

In [6]:
```
# Removal of Id column as it is not required for data analysis i.e clustering
daf = df.iloc[ : , 1:]
```

In [7]:
```
#printing the head part of the given data
df.head()
```

Out[7]:

	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

Separation of independent and dependent matrix

In [8]:
```
# Independent matrix
X = df.iloc[:,1:-1].values
# Dependent matrix
y = df.iloc[:,-1].values
```

In [9]:
```
X[0]
```

Out[9]:
```
array([1. , 5.1, 3.5, 1.4, 0.2])
```

In [10]:
```
y[0]
```

Out[10]:
```
'Iris-setosa'
```

Handling of categorical data

In [11]:
```
from sklearn.preprocessing import LabelEncoder

y_labelencoder = LabelEncoder()

y = y_labelencoder.fit_transform(y)

y
```

Out[11]:
```
array([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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       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, 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])
```

Applying suitable ML algorithm

In [13]:
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3, n_init=150)

kmeans.fit(X)
```

Out[13]:
```
KMeans(n_clusters=3, n_init=150)
```

In [14]:
```
y_kmeans = kmeans.predict(X)

y_kmeans
```

Out[14]:
```
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, 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, 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])
```

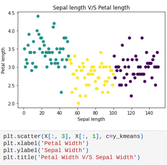
Visualization of Iris flower dataset with optimum 3 clusters

In [15]:
```
plt.scatter(X[:, 0], X[:, 2], c=y_kmeans)
plt.xlabel('Sepal length')
plt.ylabel('Petal length')
plt.title('Sepal length V/S Petal length')

pd.DataFrame({'Color':['purple', 'blue', 'yellow'], 'Iris species':['setosa', 'versicolor', 'virginica']})
```

Out[15]:

	Color	Iris species
0	purple	setosa
1	blue	versicolor
2	yellow	virginica

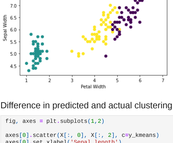


In [16]:
```
plt.scatter(X[:, 1], X[:, c=y_kmeans)
plt.xlabel('Petal Width')
plt.ylabel('Sepal width')
plt.title('Petal Width V/S Sepal Width')

pd.DataFrame({'Color':['purple', 'blue', 'yellow'], 'Iris species':['setosa', 'versicolor', 'virginica']})
```

Out[16]:

	Color	Iris species
0	purple	setosa
1	blue	versicolor
2	yellow	virginica



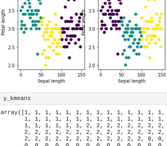
Difference in predicted and actual clustering values

In [17]:
```
fig, axes = plt.subplots(1,2)

axes[0].scatter(X[:, 0], X[:, 2], c=y_kmeans)
axes[0].set_xlabel('Sepal length')
axes[0].set_ylabel('Petal length')
axes[0].set_title('Predicted')

axes[1].scatter(X[:, 0], X[:, 2], c=y)
axes[1].set_xlabel('Sepal length')
axes[1].set_ylabel('Petal length')
axes[1].set_title('Actual')
```

Out[17]:
```
Text(0.5, 1.0, 'Actual')
```



In [18]:
```
y_kmeans
```

Out[18]:
```
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, 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, 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])
```

In [19]:
```
pd.DataFrame({'Predicted':y_kmeans,'Actual':y})
```

Out[19]:

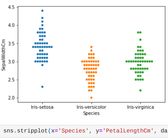
	Predicted	Actual
0	1	0
1	1	0
2	1	0
3	1	0
4	1	0
...	...	...
145	0	2
146	0	2
147	0	2
148	0	2
149	0	2

150 rows × 2 columns

Direct cluster analysis using Seaborn

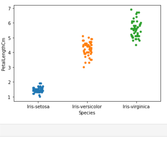
In [20]:
```
sns.swarmplot(x='Species', y='SepalWidthCm', data=df)
```

Out[20]:
```
<AxesSubplot:xlabel='Species', ylabel='SepalWidthCm'>
```



In [21]:
```
sns.stripplot(x='Species', y='PetalLengthCm', data=df)
```

Out[21]:
```
<AxesSubplot:xlabel='Species', ylabel='PetalLengthCm'>
```



In [ ]:
```
```