Author : Abhishek Dhawan

Task 2: Prediction using Unsupervised Learning GRIP @ THE SPARKS FOUNDATION Technologies:

- Programming Language: Python
- Libraries: Numpy, Pandas, Matplotlib, Scikitlearn
- Batch : April 2024 In this K-means clustering task I tried to predict the optimum number of clusters and represent it visually from the given ‘Iris’ dataset.

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

2. Reading the data

In [9]:

```
data=pd.read_csv(r"C:\Users\abhis\Downloads\Iris.csv")
print(data)
```

```
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
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 [10]:
```
data.head()
```

Out[10]:

	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 [12]:
```
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 [13]:
```
data.describe()
```

Out[13]:

	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 [15]:
```
data.shape
```

Out[15]:
```
(150, 6)
```

In [16]:
```
data['Species'].unique()
```

Out[16]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [17]:
```
data.isnull().sum()
```

Out[17]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

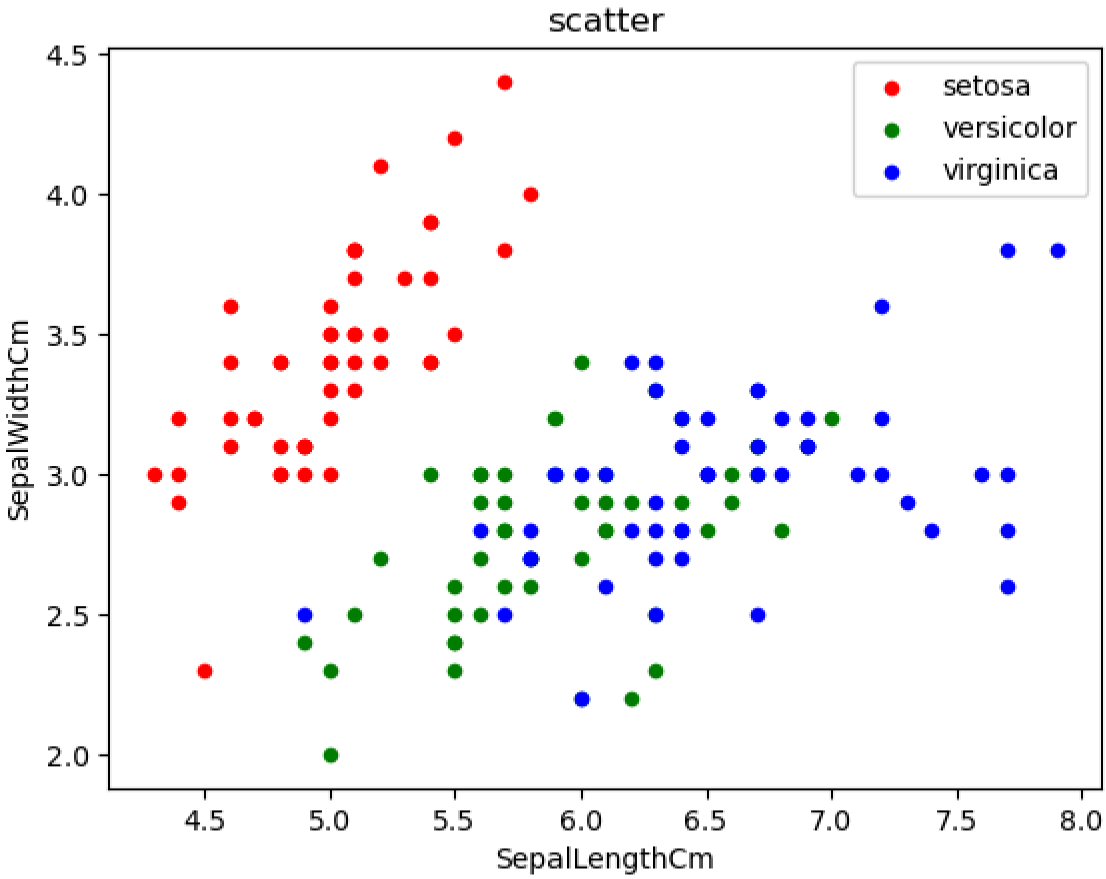
3. Visualizing the input data

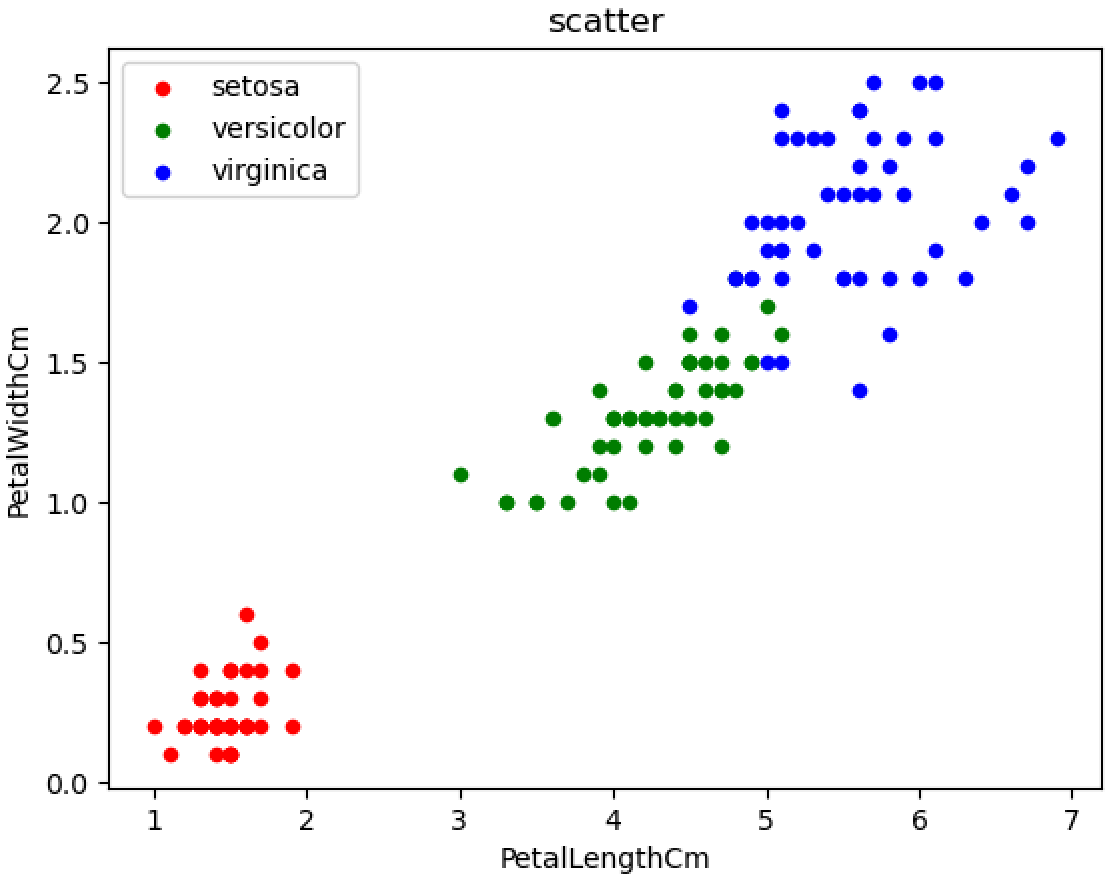
In [18]:
```
# scatter plot using pandas

ax = data[data.Species=='Iris-setosa'].plot.scatter(x='SepalLengthCm', y='S
                                                    color='red', label='set
data[data.Species=='Iris-versicolor'].plot.scatter(x='SepalLengthCm', y='Se
                                                color='green', label='versi
data[data.Species=='Iris-virginica'].plot.scatter(x='SepalLengthCm', y='Sep
                                                color='blue', label='virgin
ax.set_title("scatter")

ax = data[data.Species=='Iris-setosa'].plot.scatter(x='PetalLengthCm', y='P
                                                    color='red', label='set
data[data.Species=='Iris-versicolor'].plot.scatter(x='PetalLengthCm', y='Pe
                                                color='green', label='versi
data[data.Species=='Iris-virginica'].plot.scatter(x='PetalLengthCm', y='Pet
                                                color='blue', label='virgin
ax.set_title("scatter")
```

Out[18]:
```
Text(0.5, 1.0, 'scatter')
```





4. Data Preprocessing

In [19]:

```
#Features
X = data.drop(['Id','Species'],axis=1)
X.head()
```

Out[19]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [20]:

```
#Labels
Y = data['Species'].map({'Iris-setosa':0 , 'Iris-versicolor':1, 'Iris-virgi
Y.head()
```

Out[20]:

```
0    0
1    0
2    0
3    0
4    0
Name: Species, dtype: int64
```

In [21]:
```
X.shape,Y.shape
```

Out[21]:
```
((150, 4), (150,))
```

5. Model Training

In [22]:
```
#Define the model for the algorithm
kmodel = KNeighborsClassifier(n_neighbors=3)
```

In [23]:
```
xtrain,xtest,ytrain,ytest = train_test_split(X,Y,test_size=0.3,random_state
```

In [24]:
```
xtest.shape
```

Out[24]:
```
(45, 4)
```

In [25]:
```
kmodel.fit(xtrain,ytrain)
```

Out[25]:
```
KNeighborsClassifier(n_neighbors=3)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

6. Calculate the Training, testing and validation score

In [26]:
```
#Calculating training accuracy
Yptr = kmodel.predict(xtrain)
(Yptr == ytrain).mean()
```

Out[26]:
```
0.9714285714285714
```

In [27]:
```
#calculating testing accuracy on unknown values for model
Ypts = kmodel.predict(xtest)
(Ypts == ytest).mean()
```

Out[27]:
```
0.9555555555555556
```

In [28]:
```
# validation score
kmodel.score(xtest,ytest)
```

Out[28]:
```
0.9555555555555556
```

In [29]:
```
Ypred = kmodel.predict(X)
Ypred
```

Out[29]:
```
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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2], dtype=int64)
```

In [30]:
```
# confusion matrix
confusion_matrix(Ypred,Y)
```

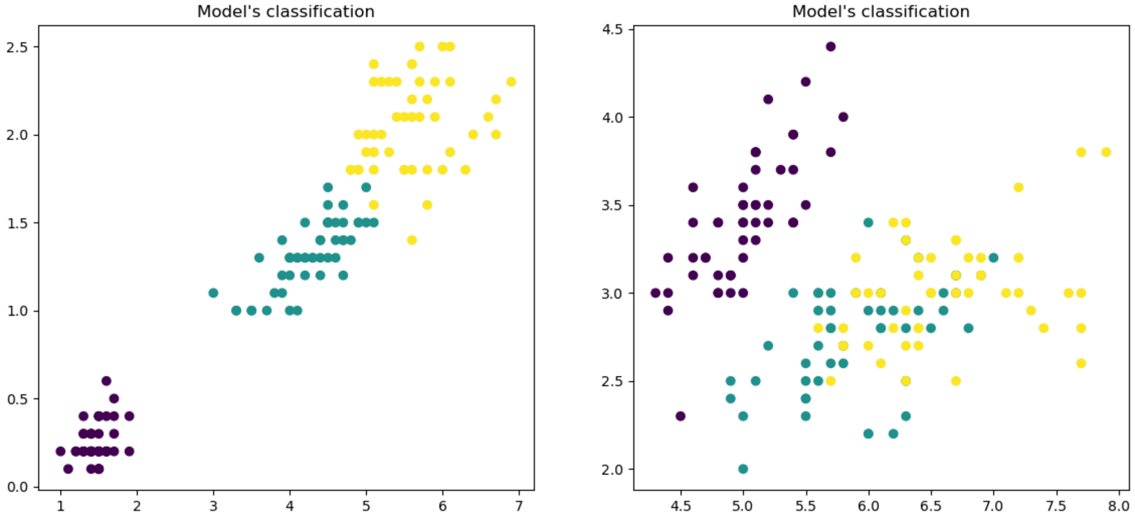
Out[30]:
```
array([[50,  0,  0],
       [ 0, 48,  3],
       [ 0,  2, 47]], dtype=int64)
```

7. Visualizing the Model cluster

In [31]:
```
plt.figure(figsize=(14,6))

# visualizing the scatters
plt.subplot(1, 2, 1)
plt.scatter(X['PetalLengthCm'] ,X['PetalWidthCm'],c = Ypred)
plt.title('Model\'s classification')
plt.subplot(1, 2, 2)
plt.scatter(X['SepalLengthCm'] ,X['SepalWidthCm'],c = Ypred)
plt.title('Model\'s classification')
```

Out[31]:
```
Text(0.5, 1.0, "Model's classification")
```



Conclusion I am able to successfully carry-out prediction using Unsupervised Machine Learning task and was able to evaluate the model's clustering accuracy score.

Thanks!

In [ ]: