TASK 3: IRIS FLOWER CLASSIFICATION

Name : Prathamesh Santosh Tondilkar

Batch : December

Domain : Data Science

The Iris flower dataset consists of three species: setosa, versicolor, and virginica. These species can be distinguished based on their measurements.

Objective is to train a machine learning model that can learn from these measurements and accurately classify the Iris flowers into their respective species.

IMPORTING IMPORTANT LIBRARIES

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sbn
```

IMPORTING DATASET

In [2]:

```
ds=pd.read_csv('IRIS.csv')
```

In [3]:

```
ds.head()
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	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

In [4]:

```
ds.tail()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	species
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

In [5]:

```
ds.shape
```

Out[5]:

```
(150, 5)
```

In [6]:

```
ds.dtypes
```

Out[6]:

```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

In [7]:

```
ds['species']
```

Out[7]:

```
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
Name: species, Length: 150, dtype: object
```

In [10]:

```
ds['species'],cat =pd.factorize(ds['species'])
ds.head(10)
```

Out[10]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
5	5.4	3.9	1.7	0.4	0
6	4.6	3.4	1.4	0.3	0
7	5.0	3.4	1.5	0.2	0
8	4.4	2.9	1.4	0.2	0
9	4.9	3.1	1.5	0.1	0

In [15]:
```
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    int64  
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
```

In [16]:
```
ds.info
```

Out[16]:
```
<bound method DataFrame.info of      sepal_length  sepal_width  petal_length  petal_width  species
0             5.1          3.5           1.4          0.2        0
1             4.9          3.0           1.4          0.2        0
2             4.7          3.2           1.3          0.2        0
3             4.6          3.1           1.5          0.2        0
4             5.0          3.6           1.4          0.2        0
..            ...          ...           ...          ...      ...
145           6.7          3.0           5.2          2.3        2
146           6.3          2.5           5.0          1.9        2
147           6.5          3.0           5.2          2.0        2
148           6.2          3.4           5.4          2.3        2
149           5.9          3.0           5.1          1.8        2

[150 rows x 5 columns]>
```

In [14]:

```
ds.describe()
```

Out[14]:

	sepal_length	sepal_width	petal_length	petal_width	species
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667	1.000000
std	0.828066	0.433594	1.764420	0.763161	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

In [13]:

```
ds.isnull().sum()
```

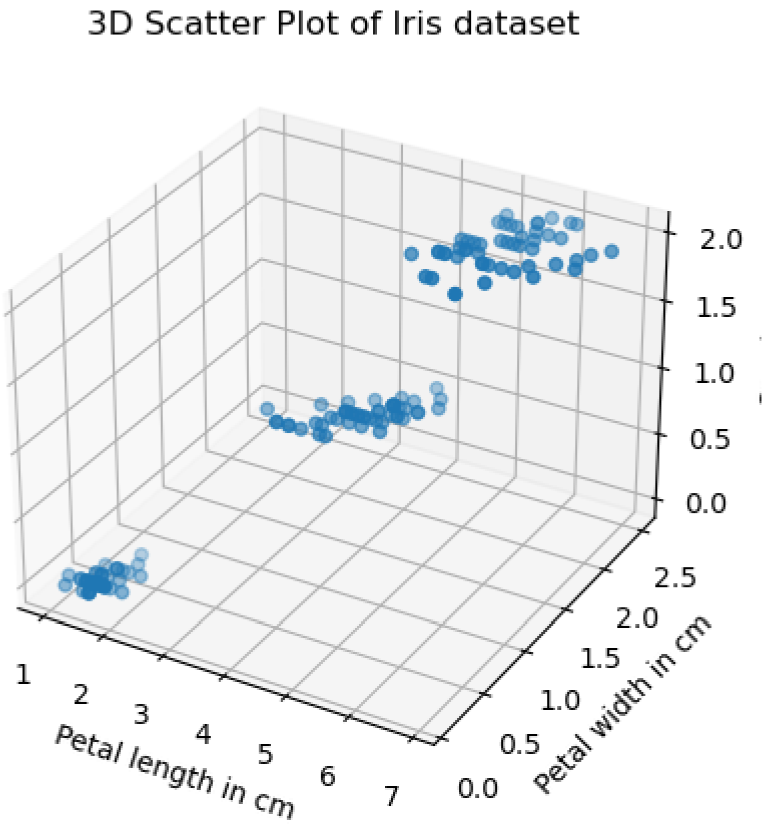
Out[13]:

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

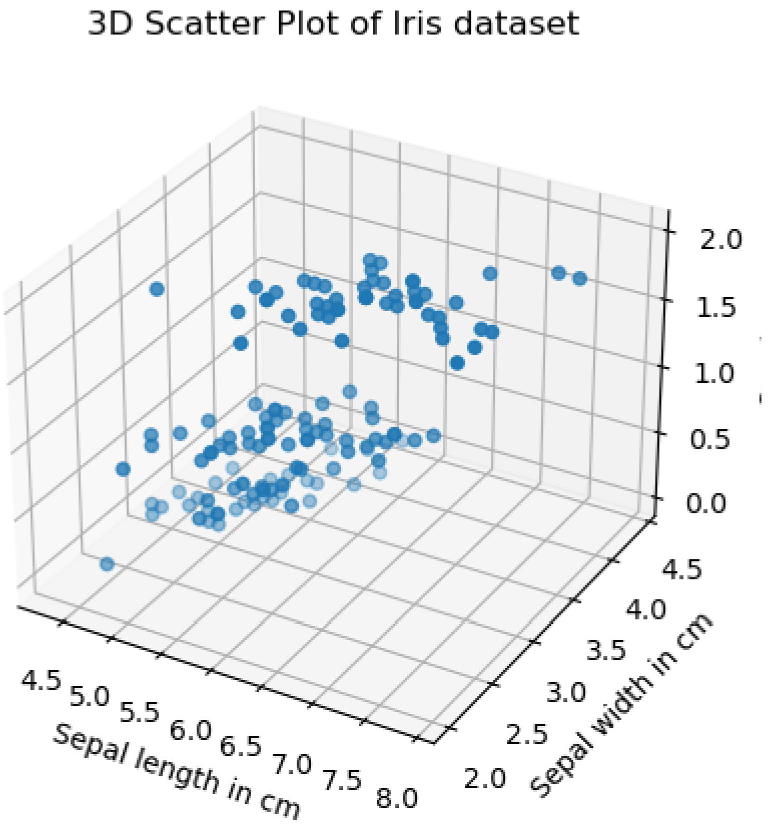
DATA VISUALIZATION

In [19]:

```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(ds.petal_length ,ds.petal_width,ds.species)
ax.set_xlabel('Petal length in cm')
ax.set_ylabel('Petal width in cm')
ax.set_zlabel('Species')
plt.title('3D Scatter Plot of Iris dataset')
plt.show()
```

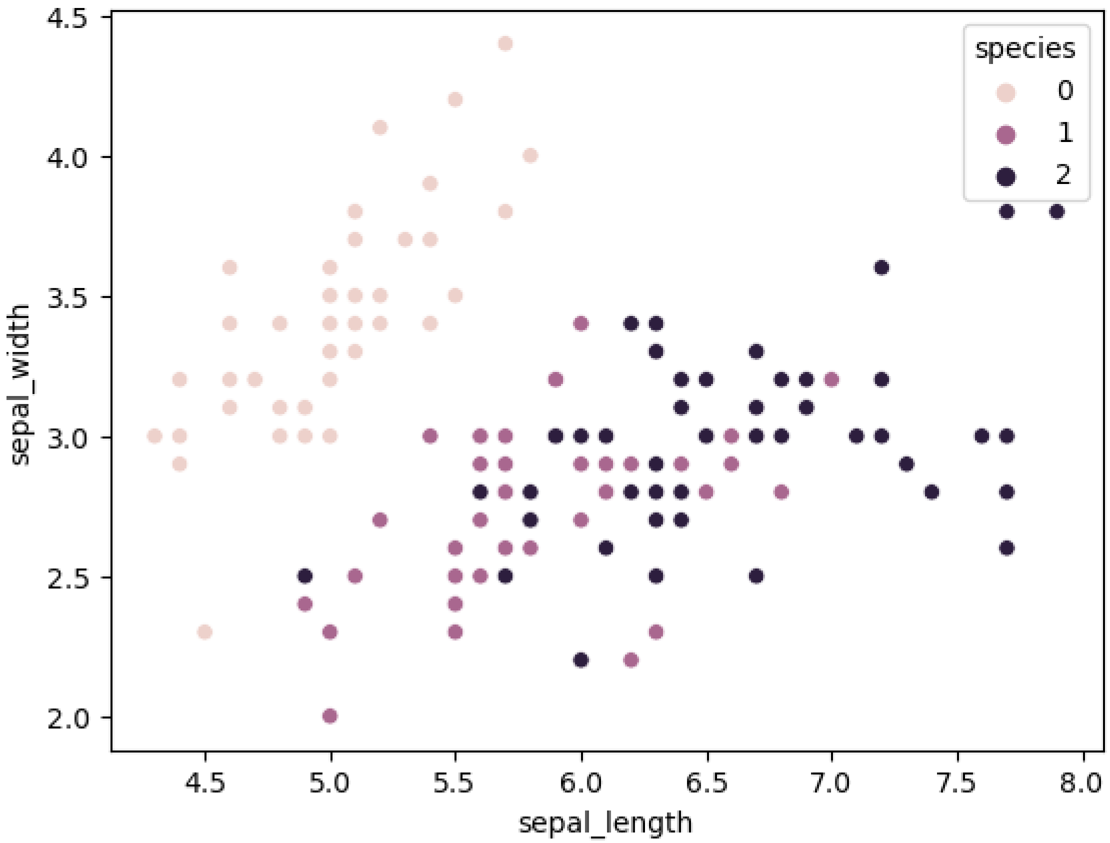


In [20]:
```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(ds.sepal_length ,ds.sepal_width,ds.species)
ax.set_xlabel('Sepal length in cm')
ax.set_ylabel('Sepal width in cm')
ax.set_zlabel('Species')
plt.title('3D Scatter Plot of Iris dataset')
plt.show()
```



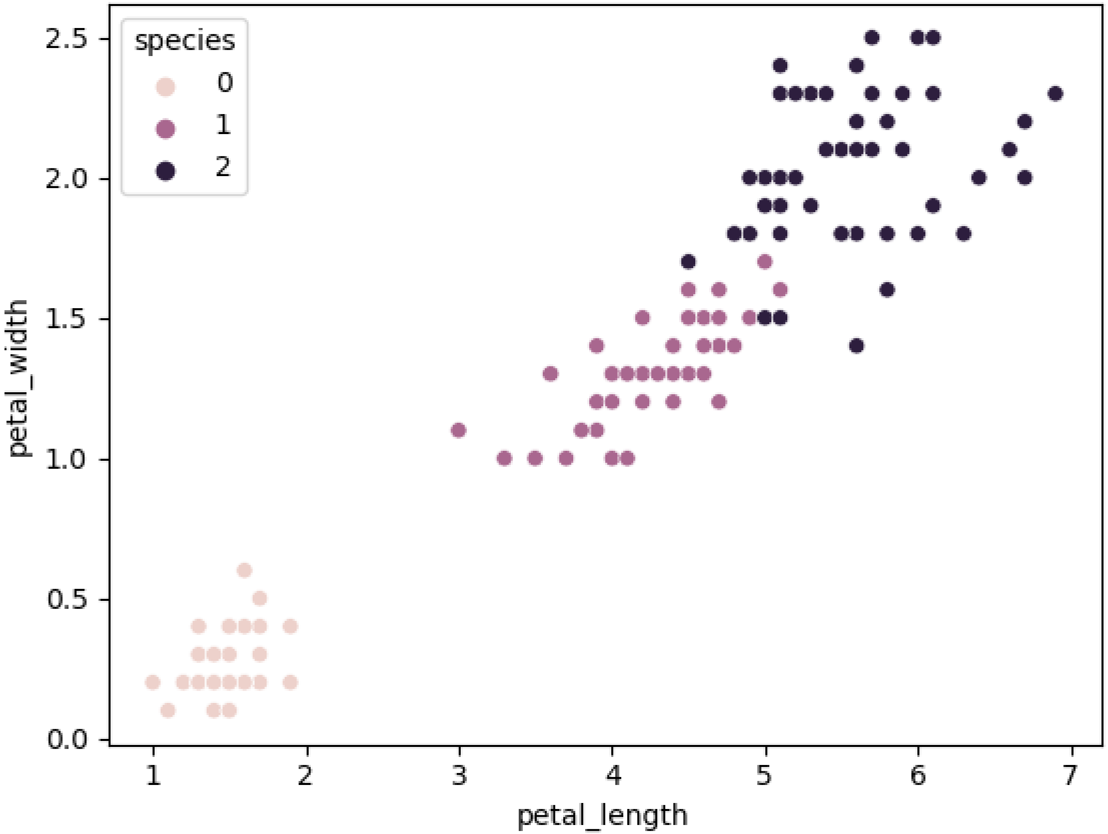
```
In [23]: sbn.scatterplot(data=ds , x='sepal_length', y='sepal_width' ,hue='species')
```

```
Out[23]: <Axes: xlabel='sepal_length', ylabel='sepal_width'>
```



```
In [24]: sbn.scatterplot(data=ds , x='petal_length', y='petal_width' ,hue='species')
```

```
Out[24]: <Axes: xlabel='petal_length', ylabel='petal_width'>
```



```
In [26]: from sklearn.cluster import KMeans
```

Elbow Technique

In [41]:

```
# Elbow Technique

sse_pk=[]
k_range=range(1,10)

for k in k_range:
    km=KMeans(n_clusters=k)
    km.fit(ds[['petal_length','petal_width']])
    sse_pk.append(km.inertia_)
```

```
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memor
y leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the enviro
nment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memor
y leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the enviro
nment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memor
y leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the enviro
nment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memor
y leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the enviro
nment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memor
y leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the enviro
nment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memor
y leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the enviro
nment variable OMP_NUM_THREADS=1.
```

```
  warnings.warn(
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memor
y leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the enviro
nment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memor
y leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the enviro
nment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memor
y leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the enviro
nment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [42]:

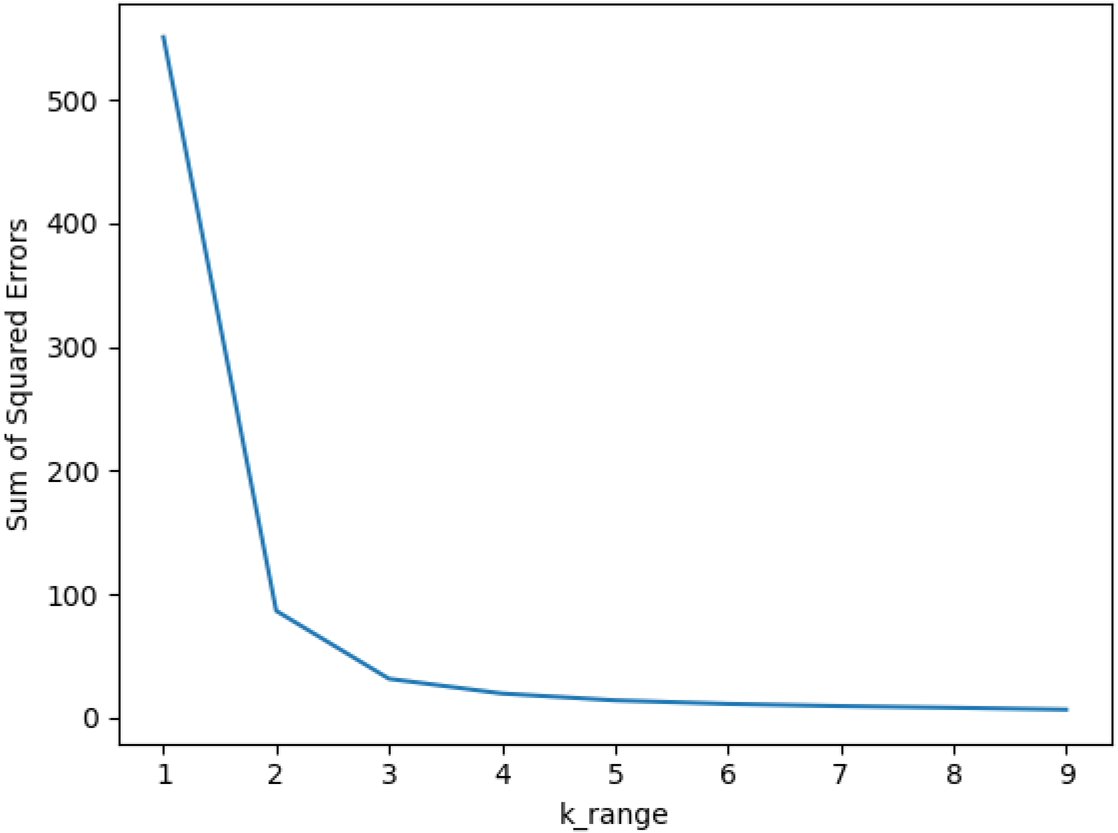
```
sse_pk
```

Out[42]:

```
[550.6434666666669,
 86.40394533571003,
 31.38775897435897,
 19.499400899685114,
 13.933308757908755,
 11.073657664362928,
 9.282035950878514,
 7.962352020202019,
 6.472894541406307]
```

```
In [43]: plt.xlabel('k_range')
         plt.ylabel('Sum of Squared Errors')
         plt.plot(k_range,sse_pk)

Out[43]: [<matplotlib.lines.Line2D at 0x1f1df8c0c90>]
```



KMeans Algorithm

In [69]:

```
km = KMeans(n_clusters=3,random_state=5)
y_predicted=km.fit_predict(ds[['petal_length','petal_width']])
y_predicted
```

```
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memor
y leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the enviro
nment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[69]:

```
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, 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, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 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, 2, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

In [70]:

```
ds['cluster']=y_predicted
ds.head(130)
```

Out[70]:

	sepal_length	sepal_width	petal_length	petal_width	species	cluster
0	5.1	3.5	1.4	0.2	0	0
1	4.9	3.0	1.4	0.2	0	0
2	4.7	3.2	1.3	0.2	0	0
3	4.6	3.1	1.5	0.2	0	0
4	5.0	3.6	1.4	0.2	0	0
...	...	...	...	...	...	...
125	7.2	3.2	6.0	1.8	2	1
126	6.2	2.8	4.8	1.8	2	2
127	6.1	3.0	4.9	1.8	2	1
128	6.4	2.8	5.6	2.1	2	1
129	7.2	3.0	5.8	1.6	2	1

130 rows × 6 columns

In [71]:

```
from sklearn.metrics import confusion_matrix
con_mat = confusion_matrix(ds.species , ds.cluster)
con_mat
```

Out[71]:

```
array([[50,  0,  0],
       [ 0,  2, 48],
       [ 0, 46,  4]], dtype=int64)
```

Accuracy Measure

Confusion Matrix

In [72]:

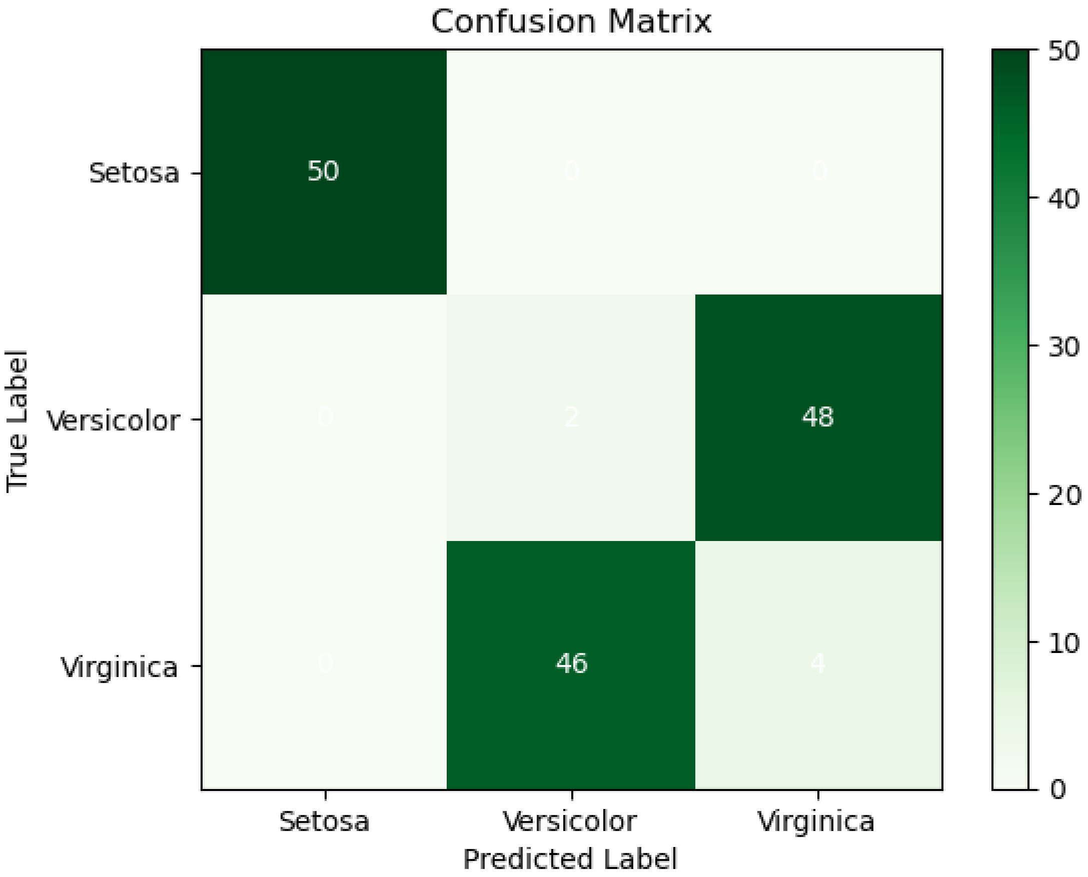
```
true_labels = ds.species
predicted_labels = ds.cluster

cm= confusion_matrix(true_labels ,predicted_labels)
class_labels = ['Setosa','Versicolor','Virginica']

plt.imshow(cm, interpolation = 'nearest' , cmap= plt.cm.Greens)
plt.title('Confusion Matrix')
plt.colorbar()
tick_marks = np.arange(len(class_labels))
plt.xticks(tick_marks , class_labels)
plt.yticks(tick_marks , class_labels)

for i  in range(len(class_labels)):
    for j in range(len(class_labels)):
        plt.text(j,i,str(cm[i][j]),ha='center',va='center',color='white')

plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```


Confusion Matrix
True Label
Setosa
Versicolor
Virginica
50
0
0
0
2
48
0
46
4
Setosa
Versicolor
Virginica
Predicted Label
50
40
30
20
10
0

In [ ]: