In [1]:
```
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = sns.load_dataset('iris')
df.head()
```

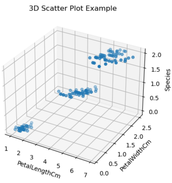
Out[2]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

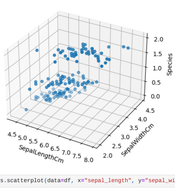
In [4]:
```
df['species'],categories =pd.factorize(df['species'])
df.species
```

Out[4]:
```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: species, Length: 150, dtype: int64
```

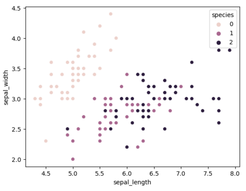
In [5]:
```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df.petal_length, df.petal_width, df.species)
ax.set_xlabel('PetalLengthCm')
ax.set_ylabel('PetalWidthCm')
ax.set_zlabel('Species')
plt.title('3D Scatter Plot Example')
plt.show()
```



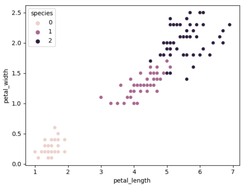
In [6]:
```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df.sepal_length, df.sepal_width, df.species)
ax.set_xlabel('SepalLengthCm')
ax.set_ylabel('SepalWidthCm')
ax.set_zlabel('Species')
plt.title('3D Scatter Plot Example')
plt.show()
```



In [7]:
```
sns.scatterplot(data=df, x="sepal_length", y="sepal_width",hue="species");
```



In [8]:
```
sns.scatterplot(data=df, x="petal_length", y="petal_width",hue="species");
```



In [10]:
```
k_rng = range(1,10)
sse=[]

for k in k_rng:
  km = KMeans(n_clusters=k)
  km.fit(df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']])
  sse.append(km.inertia_)
```

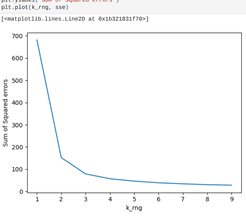
```
C:\Users\grati\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [11]:
```
sse
```

Out[11]:
```
[681.3706,
 152.34795176035792,
 78.85144142614601,
 57.228473214285714,
 46.446182052128205,
 39.03998724608725,
 34.4990997025974,
 30.440230745562326,
 28.263788900627215]
```

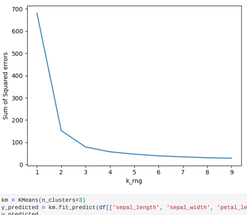
In [12]:
```
plt.xlabel('k_rng')
plt.ylabel("Sum of Squared errors")
plt.plot(k_rng, sse)
```

Out[12]:
```
[<matplotlib.lines.Line2D at 0x1b321831f70>]
```



In [13]:
```
plt.xlabel('k_rng')
plt.ylabel("Sum of Squared errors")
plt.plot(k_rng, sse)
```

Out[13]:
```
[<matplotlib.lines.Line2D at 0x1b3228807c0>]
```



In [25]:
```
km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(df[['sepal_length', 'sepal_width', 'petal_length','petal_width']])
y_predicted
```

Out[25]:
```
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, 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], dtype=int32)
```

In [26]:
```
df['cluster']= y_predicted
df
```

Out[26]:

	sepal_length	sepal_width	petal_length	petal_width	species	cluster
0	5.1	3.5	1.4	0.2	0	1
1	4.9	3.0	1.4	0.2	0	1
2	4.7	3.2	1.3	0.2	0	1
3	4.6	3.1	1.5	0.2	0	1
4	5.0	3.6	1.4	0.2	0	1
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2	0
146	6.3	2.5	5.0	1.9	2	2
147	6.5	3.0	5.2	2.0	2	0
148	6.2	3.4	5.4	2.3	2	0
149	5.9	3.0	5.1	1.8	2	2

150 rows × 6 columns

In [27]:
```
df.to_csv("task2.csv")
```

In [28]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(df.species, y_predicted)
```

In [29]:
```
cm
```

Out[29]:
```
array([[ 0, 50,  0],
       [ 2,  0, 48],
       [36,  0, 14]], dtype=int64)
```

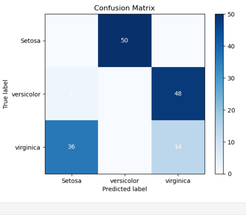
In [30]:
```
true_labels = df.species
predicted_labels= df.cluster

cm = confusion_matrix(true_labels, predicted_labels)
class_labels = ['Setosa', 'versicolor', 'virginica']

# Plot confusion matrix
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
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)

# Fill matrix with values
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()
```



In [ ]:
```
```