In [1]:

```
import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()
```

No description has been provided for this image

In [9]:

```
iris.feature_names
```

Out[9]:

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [11]:

```
iris.target_names
```

Out[11]:

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [13]:

```
df = pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()
```

Out[13]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
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 [15]:

```
df['target'] = iris.target
df.head()
```

Out[15]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
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

In [17]:

```
df[df.target==1].head()
```

Out[17]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
50	7.0	3.2	4.7	1.4	1
51	6.4	3.2	4.5	1.5	1
52	6.9	3.1	4.9	1.5	1
53	5.5	2.3	4.0	1.3	1
54	6.5	2.8	4.6	1.5	1

In [19]:

```
df[df.target==2].head()
```

Out[19]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
100	6.3	3.3	6.0	2.5	2
101	5.8	2.7	5.1	1.9	2
102	7.1	3.0	5.9	2.1	2
103	6.3	2.9	5.6	1.8	2
104	6.5	3.0	5.8	2.2	2

In [21]:

```
df['flower_name'] =df.target.apply(lambda x: iris.target_names[x])
df.head()
```

Out[21]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

In [23]:

```
df[45:55]
```

Out[23]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
45	4.8	3.0	1.4	0.3	0	setosa
46	5.1	3.8	1.6	0.2	0	setosa
47	4.6	3.2	1.4	0.2	0	setosa
48	5.3	3.7	1.5	0.2	0	setosa
49	5.0	3.3	1.4	0.2	0	setosa
50	7.0	3.2	4.7	1.4	1	versicolor
51	6.4	3.2	4.5	1.5	1	versicolor
52	6.9	3.1	4.9	1.5	1	versicolor
53	5.5	2.3	4.0	1.3	1	versicolor
54	6.5	2.8	4.6	1.5	1	versicolor

In [25]:

```
df0 = df[:50]
df1 = df[50:100]
df2 = df[100:]
```

In [27]:

```
import matplotlib.pyplot as plt
%matplotlib inline
```

Sepal length vs Sepal Width (Setosa vs Versicolor)

In [29]:

```
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'],color="green",marker=
plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'],color="blue",marker='
```

Out[29]:

```
<matplotlib.collections.PathCollection at 0x2394ce16c90>
```

Petal length vs Pepal Width (Setosa vs Versicolor)

In [31]:

```
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.scatter(df0['petal length (cm)'], df0['petal width (cm)'],color="green",marker=
plt.scatter(df1['petal length (cm)'], df1['petal width (cm)'],color="blue",marker='
```

Out[31]:

```
<matplotlib.collections.PathCollection at 0x2394ce170e0>
```

Train test split

In [33]:

```
from sklearn.model_selection import train_test_split
```

In [34]:
```
X = df.drop(['target','flower_name'], axis='columns')
y = df.target
```

In [36]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [37]:
```
len(X_train)
```

Out[37]:
```
120
```

In [39]:
```
len(X_test)
```

Out[39]:
```
30
```

Create KNN (K Neighrest Neighbour Classifier)

In [42]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
```

In [43]:
```
knn.fit(X_train, y_train)
```

Out[43]:
```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

In [46]:
```
knn.score(X_test, y_test)
```

Out[46]:
```
1.0
```

In [50]:
```
X_test[0:10]
```

Out[50]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
14	5.8	4.0	1.2	0.2
98	5.1	2.5	3.0	1.1
75	6.6	3.0	4.4	1.4
16	5.4	3.9	1.3	0.4
131	7.9	3.8	6.4	2.0
56	6.3	3.3	4.7	1.6
141	6.9	3.1	5.1	2.3
44	5.1	3.8	1.9	0.4
29	4.7	3.2	1.6	0.2
120	6.9	3.2	5.7	2.3

In [51]:
```
y_test[0:10]
```

```
Out[51]: 14     0
         98     1
         75     1
         16     0
         131    2
         56     1
         141    2
         44     0
         29     0
         120    2
         Name: target, dtype: int32
```

```
In [53]: knn.predict(X_test[0:10])
```

```
Out[53]: array([0, 1, 1, 0, 2, 1, 2, 0, 0, 2])
```

```
In [55]: knn.predict([[4.8,3.0,1.5,0.3]])
```

```
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not
have valid feature names, but KNeighborsClassifier was fitted with feature names
  warnings.warn(
```

```
Out[55]: array([0])
```

Plot Confusion Matrix

```
In [60]: from sklearn.metrics import confusion_matrix
         y_pred = knn.predict(X_test)
         cm = confusion_matrix(y_test, y_pred)
         cm
```

```
Out[60]: array([[11,  0,  0],
                [ 0, 13,  0],
                [ 0,  0,  6]], dtype=int64)
```

```
In [63]: %matplotlib inline
         import matplotlib.pyplot as plt
         import seaborn as sn
         plt.figure(figsize=(7,5))
         sn.heatmap(cm, annot=True)
         plt.xlabel('Predicted')
         plt.ylabel('Truth')
```

```
Out[63]: Text(58.222222222222214, 0.5, 'Truth')
```

Print classification report for precesion, recall and f1-score for each classes

```
In [70]: from sklearn.metrics import classification_report

         print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.92      0.96        13
           2       0.86      1.00      0.92         6

    accuracy                           0.97        30
   macro avg       0.95      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```