Lets import the data using sklearn.dataset

In [1]:
```
#loading the iris data
from sklearn.datasets import load_iris
```

Now, load the important libraries

In [2]:
```
#loading sckit random forest classifier and model selection, metrices
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, confusion_matrix, accuracy_score
```

In [3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
```

In [4]:
```
np.random.seed(0)
```

In [5]:
```
#creating an object
iris=load_iris()
```

In [6]:
```
#creating a df with the data from the object iris

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

In [7]:
```
#lets see how our data looks like
df.head()
```

Out[7]:

	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 [8]:
```
#lets add a new column for species name
df['Target'] = pd.Categorical.from_codes(iris.target,iris.target_names)
```

In [9]:
```
df.head()
```

Out[9]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Target
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 [10]:
```
df['Target'].unique()
```

Out[10]:
```
['setosa', 'versicolor', 'virginica']
Categories (3, object): ['setosa', 'versicolor', 'virginica']
```

Now lets map the unique Traget values to float from object dtype

In [11]:
```
label_map = {'setosa': 0, 'versicolor': 1, 'virginica': 2}
df['Target'] = df['Target'].map(label_map)
```

In [12]:
```
df['Target'] = df['Target'].astype(float)
```

In [13]:
```
df['Target'].unique()
```

Out[13]:
```
array([0., 1., 2.])
```

In [14]:
```
df.dtypes
```

Out[14]:
```
sepal length (cm)    float64
sepal width (cm)     float64
petal length (cm)    float64
petal width (cm)     float64
Target               float64
dtype: object
```

In [15]:
```
df
```

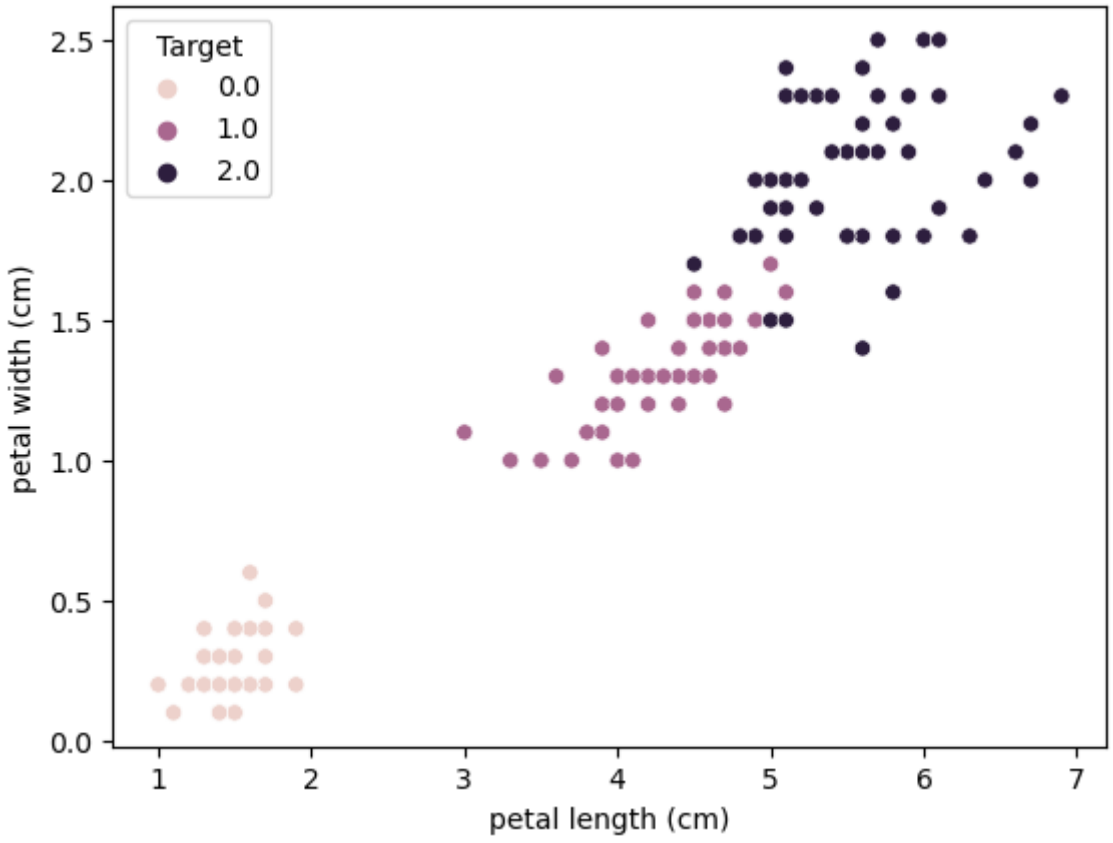
Out[15]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Target
0	5.1	3.5	1.4	0.2	0.0
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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2.0
146	6.3	2.5	5.0	1.9	2.0
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	2.0

Now, Lets plot a scatter plot: This will help us to decide which ML algorithm will be a better fit.

In [16]:
```
sns.scatterplot(x=df['petal length (cm)'], y=df['petal width (cm)'], data=d
```

Out[16]:
```
<Axes: xlabel='petal length (cm)', ylabel='petal width (cm)'>
```



The above plot shows the different species of iris flower (Target column) is shown with 3 colors as shown in above plot, the saggeration is bit easier we might not get 100 % accuracy for 1 and 2 Target but surely by proper training we can achieve a better accuracy

Here, we are soecifically using Random forest and by knowing more about Random forest you can know why its a better fit.(Look at below statement)This is just an overview about random forest Random Forest:

Random Forest is a popular ensemble learning method used in machine learning for both classification and regression tasks. It operates by constructing multiple decision trees during the training phase and outputs the mode of the classes (for classification tasks) or the mean prediction (for regression tasks) of the individual trees.

Here's a breakdown of how Random Forest works:

Ensemble of Decision Trees: Random Forest builds multiple decision trees during the training phase. Each tree is trained independently on different random subsets of the training data using a technique called bagging (bootstrap aggregating). Bagging involves randomly selecting subsets of the training data with replacement, so each tree sees a slightly different set of data.

Random Feature Selection: When building each individual decision tree, Random Forest also performs random feature selection. Instead of considering all features at each split in the tree, it randomly selects a subset of features, typically the square root of the total number of features, and chooses the best split among those features.

Random Forests are highly flexible and robust against overfitting, and they tend to perform well in a wide range of applications.

if you wish to know about the Random forest parameters please visit [https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)

In [17]:
```
#prepration for train and test data
X = df.drop('Target', axis = 1)
Y = df['Target']
```

In [18]:
```
# Splitting the dataset into the Training set and Test set
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2) #2
```

In [19]:
```
print("The shape of train dataset :")
print(X_train.shape)

print("\n The shape of test dataset :")
print(X_test.shape)
```

```
The shape of train dataset :
(120, 4)

 The shape of test dataset :
(30, 4)
```

In [20]:
```
random_classifier = RandomForestClassifier(n_estimators=4, max_depth=2, cri
random_classifier.fit(X_train, Y_train)
```

Out[20]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=2, n_estimators=4)
```

In [21]:
```
Y_test.value_counts()
```

Out[21]:
```
1.0    13
0.0    11
2.0     6
Name: Target, dtype: int64
```

In [22]:
```
y_pred = random_classifier.predict(X_test)
```

In [23]:
```
accuracy_score(Y_test,y_pred) *100
```

Out[23]:
```
96.66666666666667
```

Lets check the accuracy of the training data

In [24]:
```
y_pred_train = random_classifier.predict(X_train)
```

In [26]:
```
accuracy_score(Y_train,y_pred_train)*100
```

Out[26]:
```
97.5
```

The model is performing pretty good both in training as well as in test data; to have a better overview of the data lets plot precision, recall and F1 score. also confusion matrix here will give the best details about the test data and we can see if tthe model is overfiiting.

Classification Metrics:

Accuracy: Measures the proportion of correctly predicted instances out of the total instances.

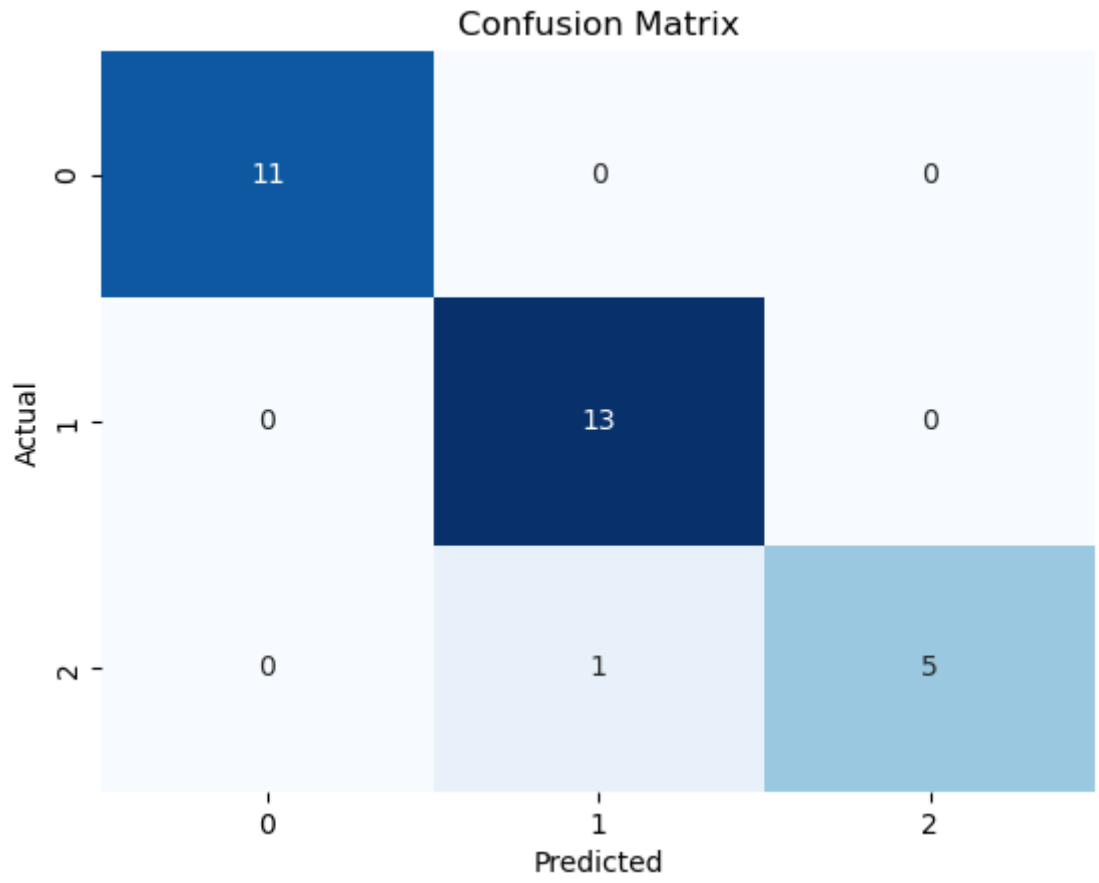
Precision: The ratio of correctly predicted positive observations to the total predicted positive observations.

Recall (Sensitivity): The ratio of correctly predicted positive observations to the all observations in the actual class.

F1 Score: The harmonic mean of precision and recall, providing a balance between the two.

In [27]:
```
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt


# Generate classification report
print(classification_report(Y_test, y_pred))

# Generate confusion matrix
conf_matrix = confusion_matrix(Y_test, y_pred)

# Plot confusion matrix using Seaborn and Matplotlib

sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        11
         1.0       0.93      1.00      0.96        13
         2.0       1.00      0.83      0.91         6

    accuracy                           0.97        30
   macro avg       0.98      0.94      0.96        30
weighted avg       0.97      0.97      0.97        30
```



if you will check the scatter which was plotted above; you can see that some of the 2 nd type of irirs flower falls in the bucket of one, and it performed as expected