TASK #4

Author: ISHWARYA D

THE SPARKS FOUNDATION INTERNSHIP

Prediction using Decision Tree Algorithm

● To Create the Decision Tree classifier and visualize it graphically. ● The purpose is if we feed any new data to this classifier, it would be able to predict the right class accordingly.

EXPLORING THE DATASET

In [32]:
```
from sklearn.datasets import load_iris
from sklearn import tree
import pandas as pd
from sklearn.model_selection import train_test_split
import seaborn as sns
from sklearn import metrics
from sklearn.metrics import plot_confusion_matrix
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
```

In [33]:
```
data = pd.read_csv('Iris.csv')
data.head()
```

Out[33]:

	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 [34]:
```
data.describe()
```

Out[34]:

	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 [35]:
```
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 [36]:
```
iris_data = load_iris()
X = pd.DataFrame(iris_data.data, columns=iris_data.feature_names)
y = iris_data.target
print(X.head())
```

```
   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
```

DATA ANALYSIS

In [37]:
```
data['Species'].unique()
```

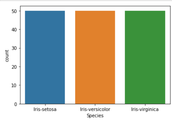
Out[37]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [52]:
```
X = data.iloc[:, 1:5].values
y = data.iloc[:, 5].values
```

In [53]:
```
X.shape, y.shape
```

Out[53]:
```
((150, 4), (150,))
```

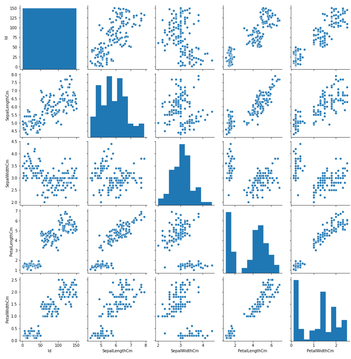
In [54]:
```
sns.countplot(data['Species'])
plt.show()
```



Data Visualization

In [38]:
```
sns.pairplot(data)
```

Out[38]:
```
<seaborn.axisgrid.PairGrid at 0x1d2fc0e8f40>
```



TRAINING THE MODEL

In [41]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=1, test_size=0.2)
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)
model = DecisionTreeClassifier()
model.fit(X_train ,y_train)
```

Out[41]:
```
DecisionTreeClassifier()
```

In [42]:
```
y_pred = model.predict(X_test)
y_pred
```

Out[42]:
```
array([0, 1, 1, 0, 2, 1, 2, 0, 0, 2, 1, 0, 2, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       2, 0, 2, 1, 0, 0, 1, 2])
```

In [51]:
```
print(f"Precision: {metrics.precision_score(y_test, y_pred, average = 'macro')}")
print(f"Recall: {metrics.recall_score(y_test, y_pred, average = 'macro')}")
print(f"F1 Score: {metrics.f1_score(y_test, y_pred, average = 'macro')}")
```

```
Precision: 0.9523809523809524
Recall: 0.9743589743589745
F1 Score: 0.9610256410256409
```

Actual and Predicted classification

In [44]:
```
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[44]:

	Actual	Predicted
0	0	0
1	1	1
2	1	1
3	0	0
4	2	2
5	1	1
6	2	2
7	0	0
8	0	0
9	2	2
10	1	1
11	0	0
12	2	2
13	1	1
14	1	1
15	0	0
16	1	1
17	1	1
18	0	0
19	0	0
20	1	1
21	1	1
22	1	2
23	0	0
24	2	2
25	1	1
26	0	0
27	0	0
28	1	1
29	2	2

Confusion Matrix for Iris Dataset Using decision tree

In [46]:
```
tree.plot_tree(clf)
pred=clf.predict(X_test)
fig=plot_confusion_matrix(clf, X_test, y_test,display_labels=["Setosa","Versicolor","Virginica"])
fig.figure_.suptitle("Confusion Matrix for Iris Dataset Using decision tree")
plt.show()
```