Practical no. 4

AIM: Perform the data classification algorithm using any Classification algorithm

CLASSIFICATION ALGORITHM -- DECISION TREE CLASSIFIER

Importing Librabries

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_scor
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets
```

Loading Dataset

In [2]:
```
# import the iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

Converting Dataset to Dataframe

In [3]:
```
df= pd.DataFrame(data=iris.data,columns=iris.feature_names)
```

In [4]:
```
df.head()
```

Out[4]:

	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 [5]:
```
df.tail()
```

Out[5]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

In [6]:
```
df.shape
```

Out[6]:
```
(150, 4)
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
dtype: int64
```

In [8]:
```
df.describe()
```

Out[8]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Train Test Split

In [9]:
```
# splitting X and y into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
```

Model Building

In [10]:
```
# DECISION TREE CLASSIFIER
dt = DecisionTreeClassifier(random_state=0)

# train the model
dt.fit(X_train, y_train)

# make predictions
dt_pred = dt.predict(X_test)
```

Model Evaluation

In [11]:
```
# print the accuracy
print("Accuracy of Decision Tree Classifier: ",accuracy_score(y_test, dt

# print other performance metrics
print("Precision of Decision Tree Classifier: ",precision_score(y_test,
print("Recall of Decision Tree Classifier: ",recall_score(y_test, dt_pre
print("F1-Score of Decision Tree Classifier: ",f1_score(y_test, dt_pred,
```

```
Accuracy of Decision Tree Classifier:  0.9555555555555556
Precision of Decision Tree Classifier:  0.9555555555555556
Recall of Decision Tree Classifier:  0.9555555555555556
F1-Score of Decision Tree Classifier:  0.9555555555555556
```

In [ ]: