Vashu Agarwal

E21CSEU0054

In [1]:
```
# import Libraries
import pandas as pd                          # used for import the data
import numpy as np                           # used for multidimensiona
import matplotlib.pyplot as plt
# used for plot the Graph

from sklearn.preprocessing import LabelEncoder
#Task: import the dataset (iris.csv)

dataset = pd.read_csv("/Users/vashuagarwal/Downloads/BENNETT  thing

dataset['Species'] = LabelEncoder().fit_transform(dataset['Species'

dataset.head()
```

Out[1]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0
2	3	4.7	3.2	1.3	0.2	0
3	4	4.6	3.1	1.5	0.2	0
4	5	5.0	3.6	1.4	0.2	0

In [4]:
```
X = dataset.iloc[:,[2,3,4]].values      # predictor attribute
Y = dataset.iloc[:,-1].values         # target attribute
```

In [5]:
```
# Task : split the dataset into test set and train set

from sklearn.model_selection import train_test_split as tts

X_train,X_test,Y_train,Y_test = tts(X,Y,test_size = 0.25,random_sta

print(X_train.shape,X_test.shape,Y_train.shape,Y_test.shape)
```

```
(112, 3) (38, 3) (112,) (38,)
```

In [22]:
```
# Task : scale the feature

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [25]:
```
# Part 2: Building the Model
# import SVM regression model from sscikit learn
from sklearn.svm import SVC

# Task 3: init the model and Set the kernel to ‘radial basic functi
classifier = SVC(kernel='rbf',C=0.1, gamma=0.1)




# #task 4: fit the training model into our SVM classifer regression


classifier.fit(X_train, Y_train)
```

Out[25]:
```
SVC(C=0.1, gamma=0.1)
```

In [26]:
```
#Part 3: Making a prediction and visualize the result
# Task 5 : making a prediction

Y_pred = classifier.predict(X_test)
print(Y_pred)
```

```
[2 1 0 2 0 2 0 2 2 1 2 2 1 2 2 0 2 1 0 0 2 2 0 0 2 0 0 1 1 0 2 2 0
2 2 1 0
 2]
```

In [28]:
```
#Task 6: print accuracy_score, classification report, confusion mat

from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(Y_test, Y_pred))
```

```
Accuracy :  0.7631578947368421
```

In [29]:
```
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        13
           1       1.00      0.44      0.61        16
           2       0.50      1.00      0.67         9

    accuracy                           0.76        38
   macro avg       0.83      0.81      0.76        38
weighted avg       0.88      0.76      0.76        38
```

In [30]:
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(Y_test, Y_pred)
print(cm)
```

```
[[13  0  0]
 [ 0  7  9]
 [ 0  0  9]]
```

In [31]:
```
import math

def minimax (curDepth, nodeIndex,maxTurn, scores,targetDepth):
    if (curDepth == targetDepth):
        return scores[nodeIndex]
    if (maxTurn):
        return max(minimax(curDepth + 1, nodeIndex * 2,False, score
                   minimax(curDepth + 1, nodeIndex * 2 + 1,False, s
    else:
        return min(minimax(curDepth + 1, nodeIndex * 2,True, scores
                   minimax(curDepth + 1, nodeIndex * 2 + 1,True, sc

scores = [3, 5, 2, 9]
treeDepth = math.log(len(scores), 2)

print("The optimal value is : ", end = "")
print(minimax(0, 0, True, scores, treeDepth))
```

```
The optimal value is : 3
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: