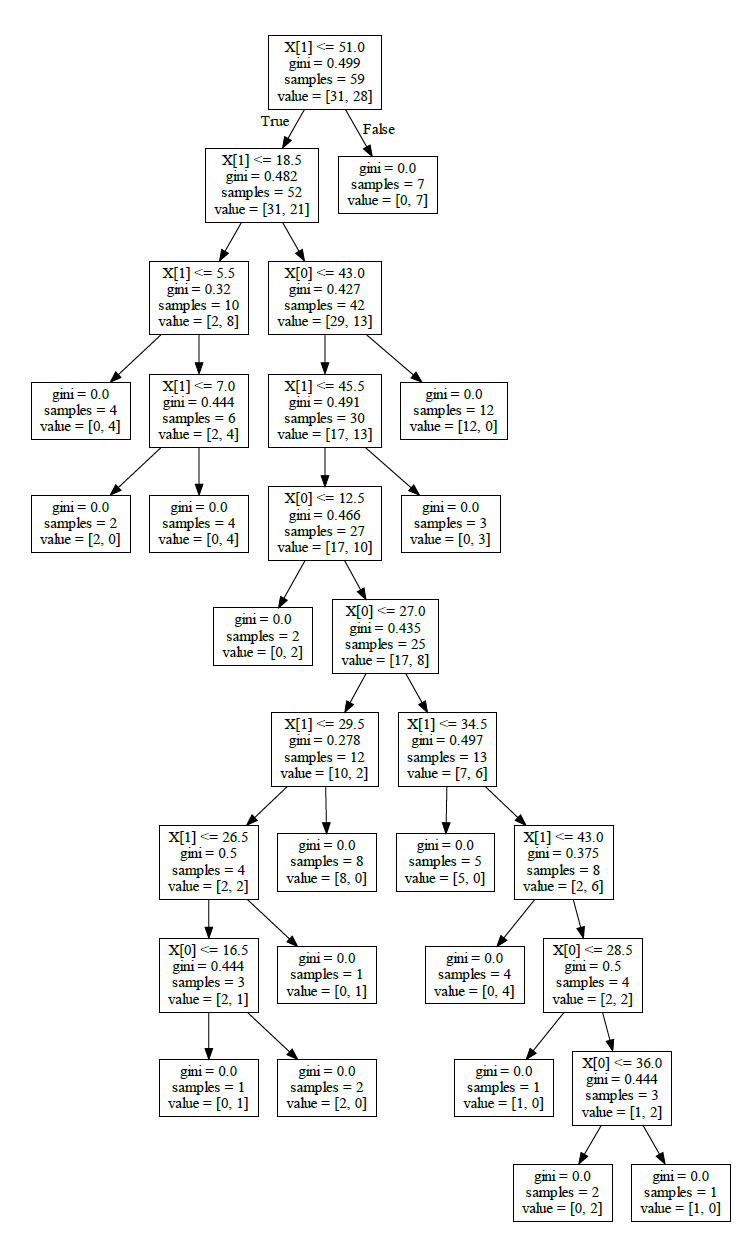
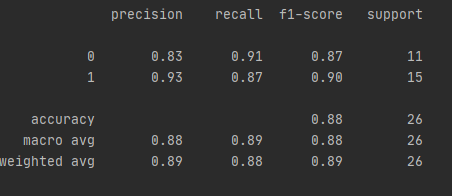
Nathan Sucher

Homework 3

1)

import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
from sklearn import tree  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.metrics import classification\_report, confusion\_matrix  
  
# Import the data from the excel file  
dataset = pd.read\_excel("C:/Users/Nathan/OneDrive - University of Cincinnati/4th Year CompE/AIPrinciplesAndApplications/CS4033\_AI/Homework3/HW3Data.xlsx", "Sheet1")  
  
# Create the X and y datasets  
X = dataset.drop('Class', axis=1)  
y = dataset['Class']  
  
# Split the data for training and testing  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30)  
  
# Create decision tree and fit training data to tree  
classifier = DecisionTreeClassifier(min\_impurity\_split=0.1)  
classifier = classifier.fit(X\_train, y\_train)  
  
# Export tree to dot file with graphviz  
tree.export\_graphviz(classifier, out\_file = 'tree.dot')

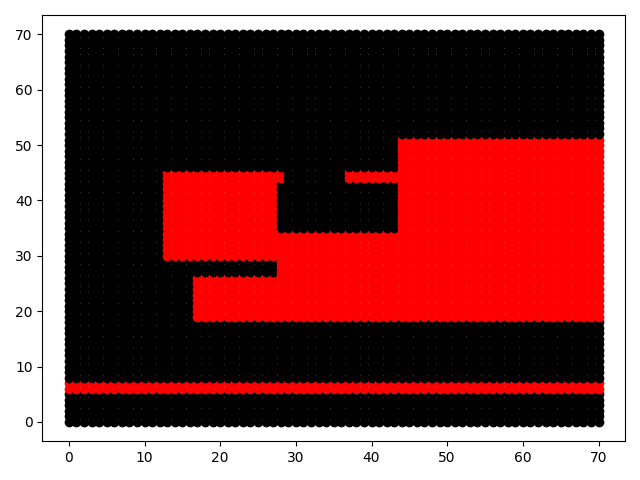
2)

**1**

**0**

**1 0**

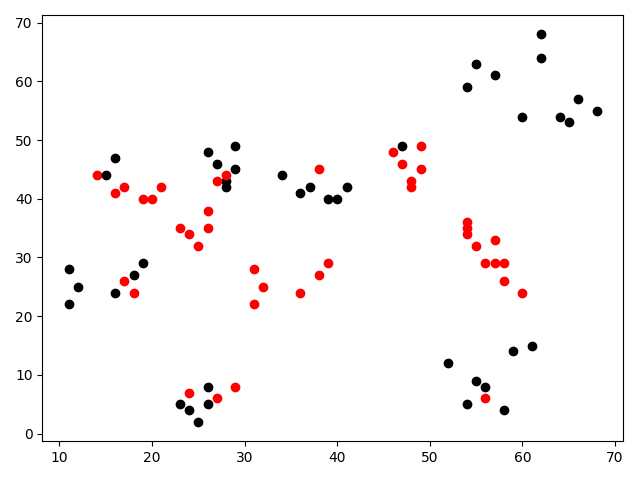
# Use test data to predict y and compare to actual data  
y\_pred = classifier.predict(X\_test)  
print(confusion\_matrix(y\_test, y\_pred))  
print(classification\_report(y\_test, y\_pred))

3)

Red: Class 0

Black: Class 1

# Plot 70x70 grid  
for x in range(71):  
 for y in range(71):  
 x\_y = [[x, y]]  
 out = classifier.predict(x\_y)  
 if out == 0:  
 plt.plot(x, y, 'o', color='red')  
 elif out == 1:  
 plt.plot(x, y, 'o', color='black')  
  
plt.show()

4)

Chart, scatter chart

Description automatically generated

Red: Class 0

Black: Class 1

The boundaries drawn are very similar to those that appear in the graph of Question #3. Some of the data points had more black points surrounding the red points, so those were excluded to keep from encapsulating too many black points in the red zones. My boundaries try to maximize the red collected without collection the black points.