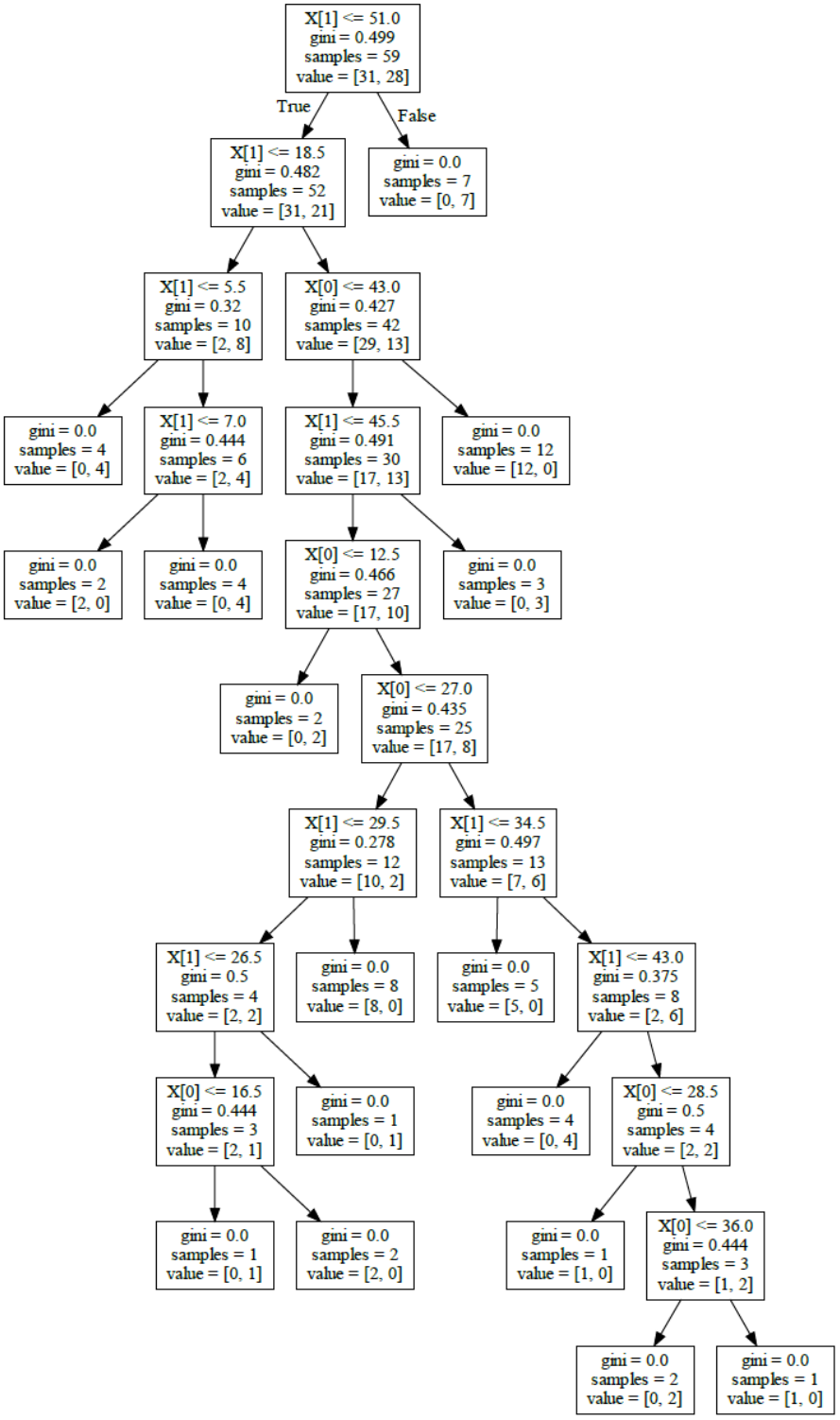
Nathan Sucher

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)

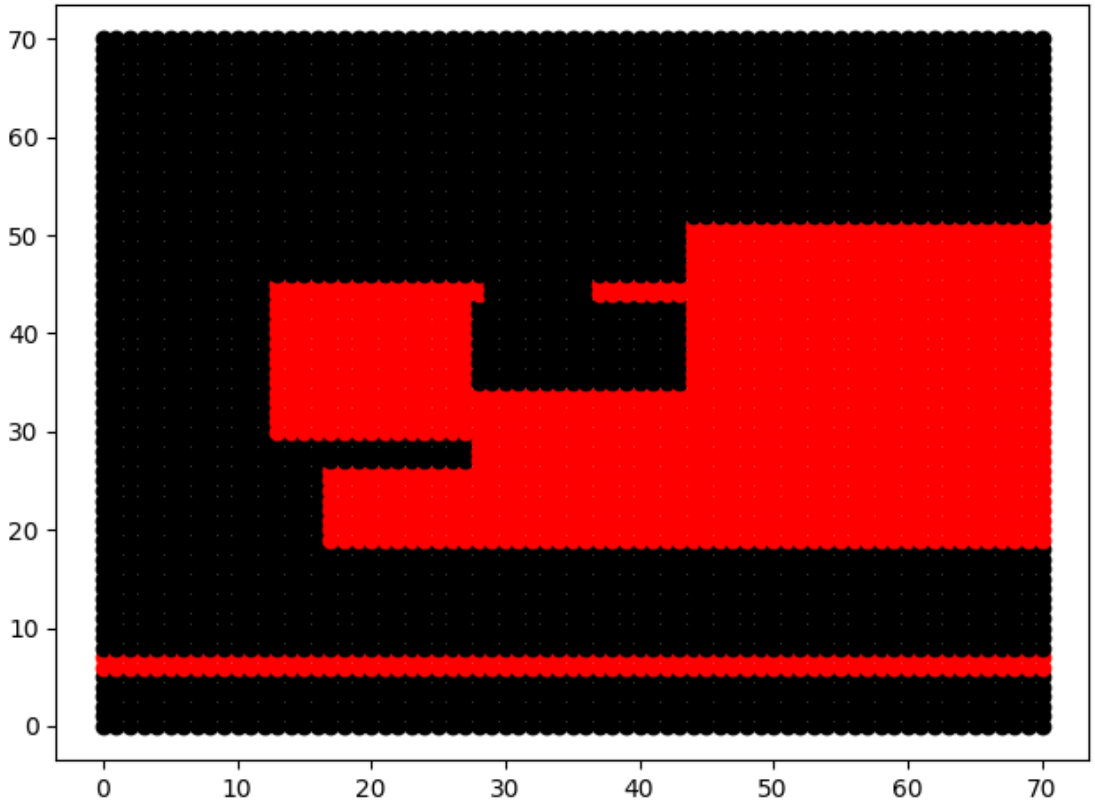
```
      1  0
1  [[10  1]
0   [ 2 13]]
              precision    recall  f1-score   support

           0       0.83      0.91      0.87        11
           1       0.93      0.87      0.90        15

    accuracy                           0.88        26
   macro avg       0.88      0.89      0.88        26
weighted avg       0.89      0.88      0.89        26
```

```
# 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)



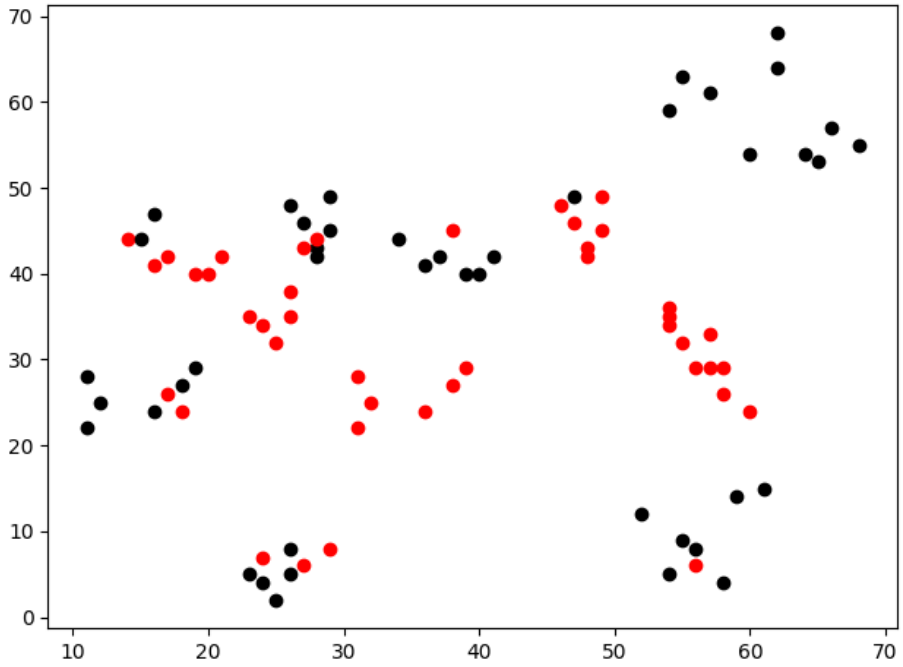
Red: Class 0

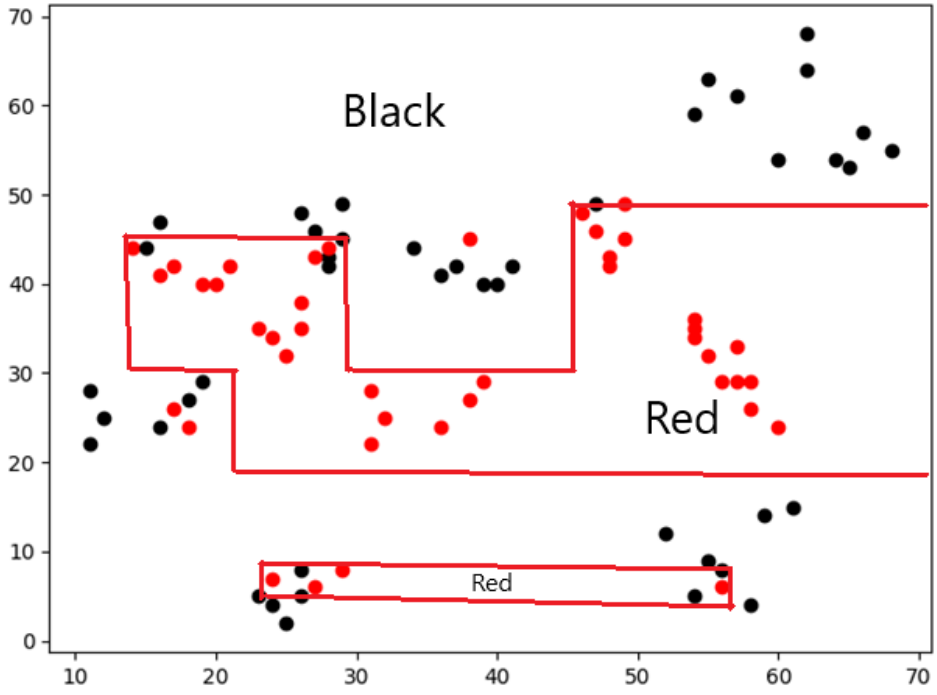
Black: Class 1

```
# 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)





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.