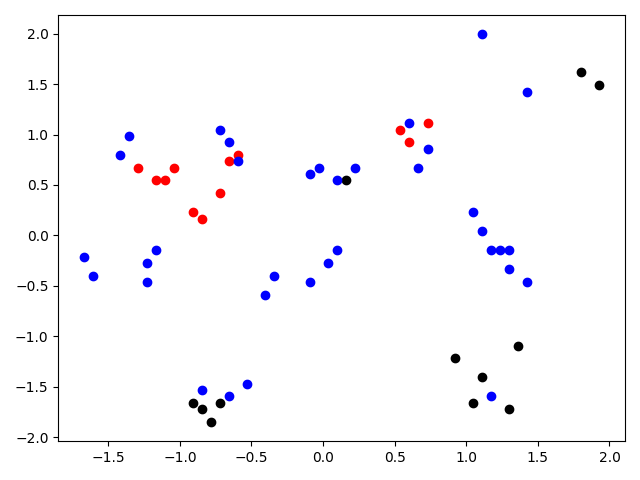
Nathan Sucher

Homework 4

1. 

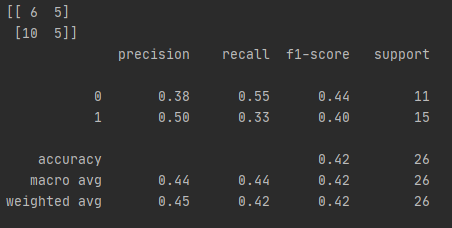
Class 0: **Red**

Class 1: **Black**

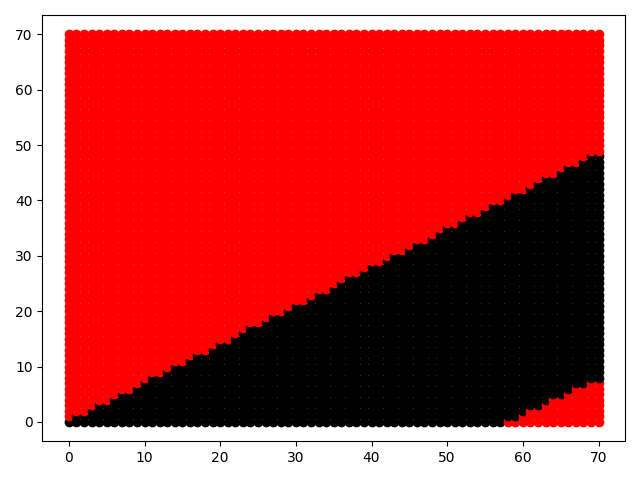
Misclassified: **Blue** (Count = 35)

import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from sklearn.neural\_network import MLPClassifier  
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/Homework4/HW4Data.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)  
  
scaler = StandardScaler()  
# Fit only to the training data  
scaler.fit(X\_train)  
  
# Apply the transformations to the data:  
X\_train = scaler.transform(X\_train)  
X\_test = scaler.transform(X\_test)  
  
# Make model with one layer with 2 neurons  
mlp = MLPClassifier(hidden\_layer\_sizes=(2,))  
  
# Fit training data to the model  
mlp.fit(X\_train, y\_train)  
  
# Predict the class for graph compare  
graph\_predict = mlp.predict(X\_train)  
  
i = 0  
misclassified = 0  
for y in y\_train:  
 if y == graph\_predict[i]:  
 if y == 0:  
 plt.plot(X\_train[i][0], X\_train[i][1], 'o', color='red')  
 elif y == 1:  
 plt.plot(X\_train[i][0], X\_train[i][1], 'o', color='black')  
 else:  
 plt.plot(X\_train[i][0], X\_train[i][1], 'o', color='blue')  
 misclassified += 1  
 i += 1  
  
print("Misclassified Count: ", misclassified)  
plt.show()





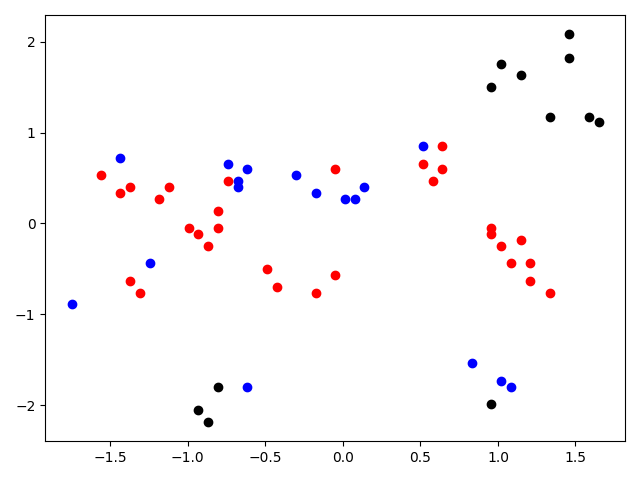
# Predict the class  
predictions = mlp.predict(X\_test)  
  
print(confusion\_matrix(y\_test, predictions))  
print(classification\_report(y\_test, predictions))

1. 

Class 0: **Red**

Class 1: **Black**

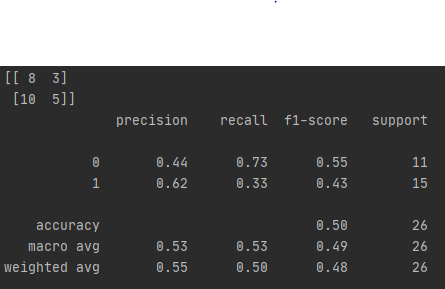
# Plot 70x70 grid  
for x in range(71):  
 for y in range(71):  
 x\_y = [[x, y]]  
 out = mlp.predict(x\_y)  
 if out == 0:  
 plt.plot(x, y, 'o', color='red')  
 elif out == 1:  
 plt.plot(x, y, 'o', color='black')  
  
plt.show()

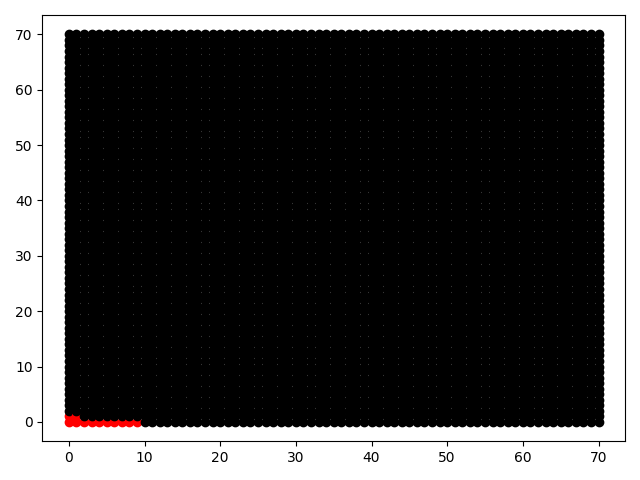
1. 

Class 0: **Red**

Class 1: **Black**

Misclassified: **Blue** (Count = 17)



****

Class 0: **Red**

Class 1: **Black**

The boundaries for both cases in Step #3 are distinct. The case with 2 neurons went from red to black to red with even sloping lines to separate the classes. The case with 6 neurons only has one transition from red to black. The boundaries for the NN are clearer and do not fluctuate like they did for the decision tree. The NN has distinct transitions where the decision tree was random.