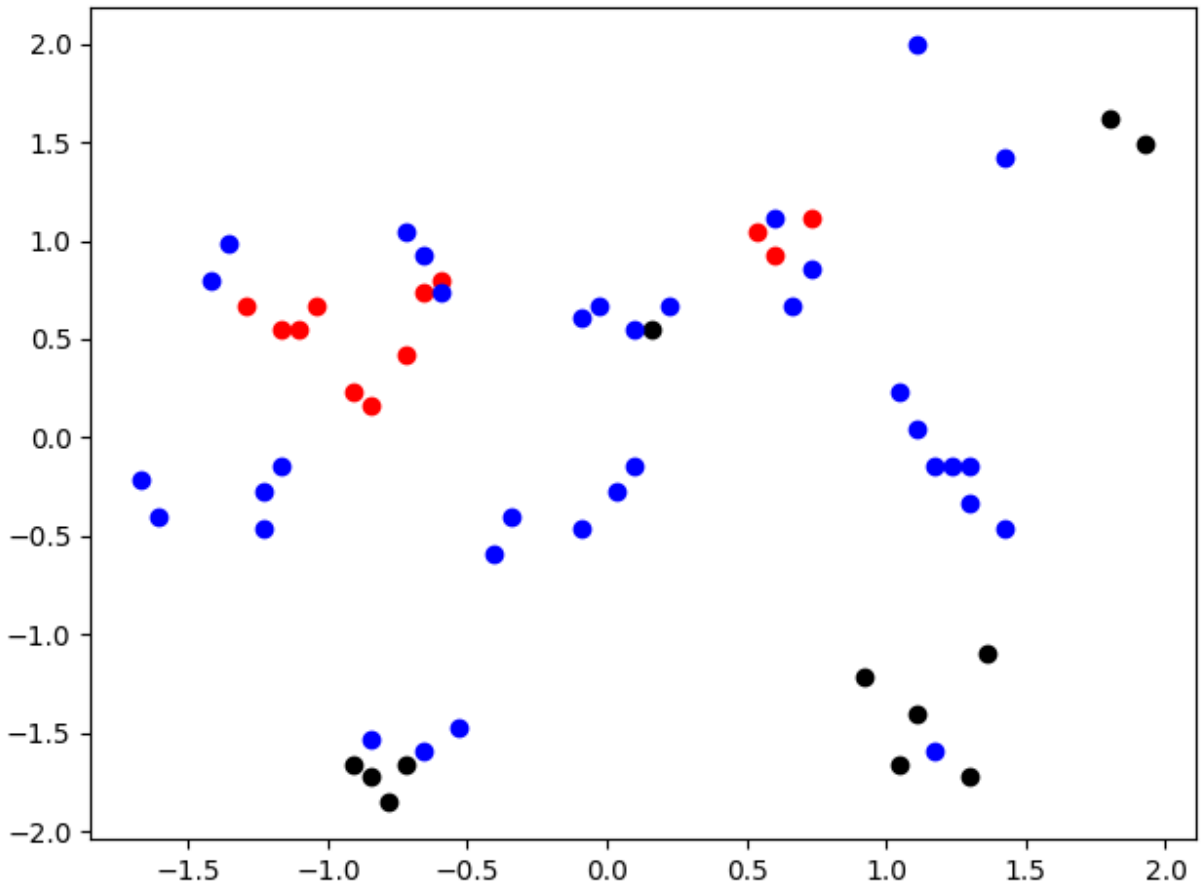
Nathan Sucher

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']
```

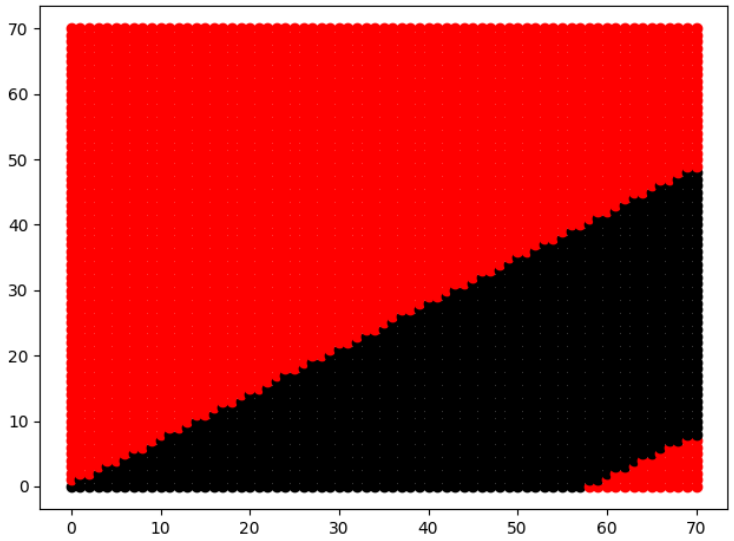
```
# Split the data for training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)

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()
```

2.

```
[[ 6  5]
 [10  5]]
              precision    recall  f1-score   support

           0       0.38      0.55      0.44        11
           1       0.50      0.33      0.40        15

    accuracy                           0.42        26
   macro avg       0.44      0.44      0.42        26
weighted avg       0.45      0.42      0.42        26
```

```
# Predict the class
predictions = mlp.predict(X_test)

print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

3.

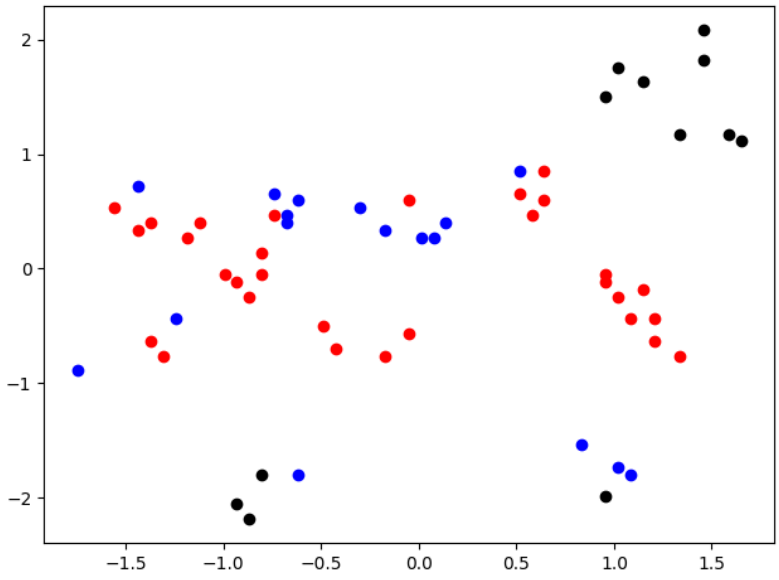


Class 0: **Red**

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()
```

4.

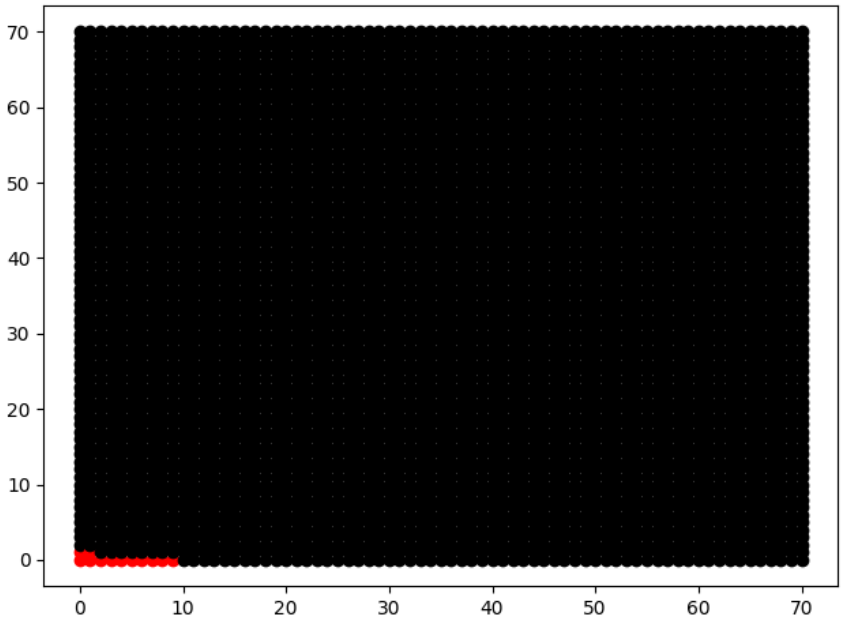


Class 0: **Red**

Class 1: **Black**

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

```
[[ 8  3]
 [10  5]]
              precision    recall  f1-score   support

           0       0.44      0.73      0.55        11
           1       0.62      0.33      0.43        15

    accuracy                           0.50        26
   macro avg       0.53      0.53      0.49        26
weighted avg       0.55      0.50      0.48        26
```



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.