HW4

Chris Jantzen

Q1)

from pandas import read\_excel

from random import sample

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier, export\_graphviz

from sklearn.metrics import classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

import graphviz

def read\_data():

table = read\_excel("P4Data.xlsx")

data\_sets = []

classes = []

for x, y, c in zip(table['X'], table['Y'], table['Class']):

data\_sets.append([x, y])

classes.append([c])

return (data\_sets, classes)

def get\_testing\_indicies(training\_indices, data\_size):

testing\_indicies = []

for index in range(data\_size):

if index not in training\_indices:

testing\_indicies.append(index)

return testing\_indicies

def build\_50\_by\_50():

lst = []

for i in range(1, 51):

for j in range(1, 51):

lst.append([i, j])

return lst

def sort\_pred(coords, pred):

pred0x = []

pred0y = []

pred1x = []

pred1y = []

for i in range(len(pred)):

if pred[i] == 0:

pred0x.append(coords[i][0])

pred0y.append(coords[i][1])

else:

pred1x.append(coords[i][0])

pred1y.append(coords[i][1])

return (pred0x, pred0y, pred1x, pred1y)

def create\_graph(coords, pred):

pred0x, pred0y, pred1x, pred1y = sort\_pred(coords, pred)

plt.plot(pred0x, pred0y, 'ks', pred1x, pred1y, 'rs')

plt.xlabel('x')

plt.ylabel('y')

plt.title('Decision Tree Predictions')

plt.axis([0, 50, 0, 50])

plt.show()

def main():

(data\_sets, classes) = read\_data()

x\_train, x\_test, y\_train, y\_test = train\_test\_split(

data\_sets, classes, test\_size=.30)

classifier = DecisionTreeClassifier()

classifier.fit(x\_train, y\_train)

y\_pred = classifier.predict(x\_test)

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

dot\_data = export\_graphviz(classifier, out\_file=None, feature\_names=[

'X', 'Y'], class\_names=['0', '1'])

graph = graphviz.Source(dot\_data)

graph.render("iris")

n\_test = build\_50\_by\_50()

n\_pred = classifier.predict(n\_test)

m\_test = data\_sets

m\_pred = classifier.predict(m\_test)

create\_graph(n\_test, n\_pred)

create\_graph(m\_test, m\_pred)

if \_\_name\_\_ == "\_\_main\_\_":

main()

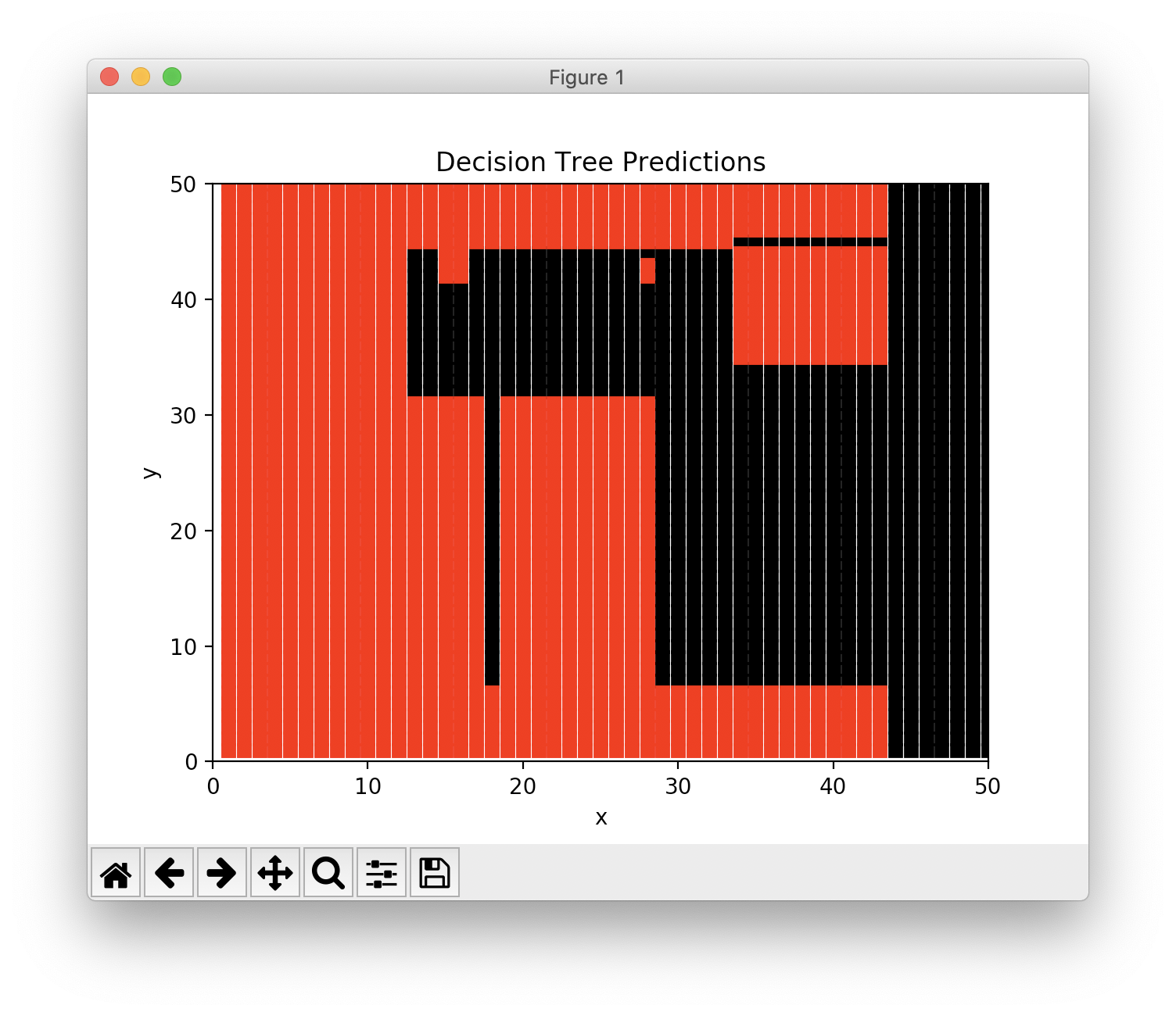




A screenshot of a computer screen

Description automatically generated







50x50 data points plotted with the classifier

A screenshot of a cell phone

Description automatically generated

Plotted excel document points for reference

In the graphs above, the black points represent coordinate points that the classifier predicted to be in class 0 and the red points are those predicted to be in class 1. Looking at the 50x50 graph, the class 1 points seem to trend towards the left (generally x < 30) and at the top and bottom (y > 45 and y < 7). The class 0 categorized points tended to be in the x < 44 area and in a couple of boxes in the middle area.



The black line at about x = 18 seems out of place, looking at the scatter plot of the excel values. There doesn’t appear to be any indication that that would be a class 0 area, so I would just have made that area red from y = 31 down to y=0 at x = 18. Otherwise, given the small amount of data the classifier has to work with, the areas appear to be fairly correct.

Q2) Perceptron

A screenshot of a computer

Description automatically generated

Final: 4, -1, 10, 3

3 are correctly classified after 2 epochs.