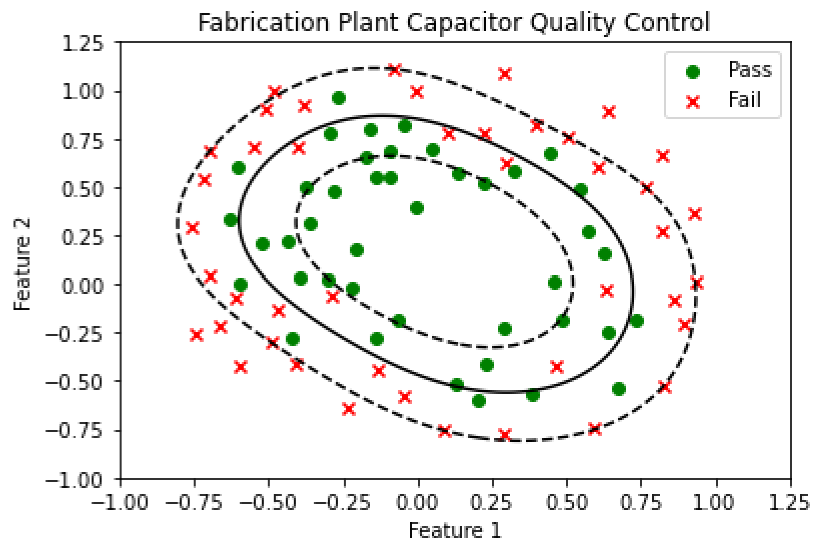
Sydney Trout
Dr Li
CPSC 4430
14 March 2022

SVM Report

Model Description: This model uses SVM as a binary classifier. Utilizing SVM functions in the Scikit-learn library, the hyperplane and margins were able to be produced. This, along with the proper kernel function, allowed for an accuracy above 70%.

Kernel Function Description: The kernel function I utilized was the Radial Basis Function, or RBF. It is most commonly used for non-linear classifiers and they are symmetric around the origin, which is what makes it so powerful in this instance and allows for an accuracy above 70%.

Plot of margin and hyperplane:



CONFUSION MATRIX ON TEST SET		Predicted Class	
		Negative	Positive
True Class	Negative	TN = 17	FP = 0
	Positive	FN = 5	TP = 11

The accuracy of the model on the testing data was 84.84%. The precision was 100%, the recall was 68.75%, and the F1 score was 81.48%.

CODE:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn import svm, datasets

#function that creates a meshgrid for the hyperplane
def make_meshgrid(x, y, h=.02):
    xmin, xmax = x.min() - 0.1, x.max() + 0.1
    ymin, ymax = y.min() - 0.1, y.max() + 0.1
    f1, f2 = np.meshgrid(np.arange(xmin, xmax, h), np.arange(ymin, ymax, h))
    return f1, f2

#reads in the name of the training file and opens it
fileName = input("Enter the name of your training file: ")
dataFile = open(fileName, "r")

#reads in the number of rows and features
readData = dataFile.readline()
fileInfo = readData.split("\t")
rows = int(fileInfo[0])
features = int(fileInfo[1])

xBinTrain = np.zeros((rows, features))
yBinTrain = []

#reads in the training data into bins
for i in range(rows):
    readData = dataFile.readline()
    data = readData.split("\t")
    xBinTrain[i,0] = float(data[0])
    xBinTrain[i,1] = float(data[1])
    yBinTrain.append(float(data[2]))

#creates the hyperplane and margins
model = svm.SVC(kernel='rbf')
fit = model.fit(xBinTrain, yBinTrain)
X0, X1 = xBinTrain[:, 0], xBinTrain[:, 1]
f1, f2 = make_meshgrid(X0, X1)
ax = plt.gca()
```

```
xy = np.vstack([f1.ravel(), f2.ravel()]).T
f = fit.decision_function(xy).reshape(f1.shape)
ax.contour(f1, f2, f, colors="k", levels=[-1, 0, 1], alpha=1, linestyles=["--", "-", "--"])

#plots the training data
for i in range(len(xBinTrain)):
    if(yBinTrain[i] == 1):
        s1 = plt.scatter(xBinTrain[i,0],xBinTrain[i,1], color = "green", marker = "o", label = "Pass")
    else:
        s2 = plt.scatter(xBinTrain[i,0],xBinTrain[i,1], color = "red", marker = "x", label = "Fail")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.title("Fabrication Plant Capacitor Quality Control")
plt.legend((s1,s2), ("Pass", "Fail"), loc="upper right")
plt.xticks((-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1, 1.25))
plt.yticks((-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1, 1.25))
plt.show()

#reads in the testing file name and opening it
fileNameT = input("Enter the name of your testing file: ")
dataFileT = open(fileNameT, "r")

#reads in the number of rows and features
readDataT = dataFileT.readline()
fileInfoT = readDataT.split("\t")
rowsT = int(fileInfoT[0])
featuresT = int(fileInfoT[1])



xBinTest = np.zeros((rowsT, featuresT))
xBinTest1 = np.zeros((rowsT,1))
xBinTest2 = np.zeros((rowsT,1))
yBinTest = np.zeros((rowsT,1))

#reads in the testing data into bins
for i in range(rowsT):
    readDataT = dataFileT.readline()
    dataT = readDataT.split("\t")
    xBinTest[i,0] = float(dataT[0])
    xBinTest[i,1] = float(dataT[1])
    xBinTest1[i,0] = float(dataT[0])
    xBinTest2[i,0] = float(dataT[1])
    yBinTest[i,0] = float(dataT[2])

TP = 0
TN = 0
FP = 0
FN = 0

#calculates the TP, FP, TN, FN
for i in range(rowsT):
    ans = fit.predict(xBinTest[i].reshape(1,-1))
    if(ans == 1 and yBinTest[i] == 1):
        TP = TP + 1
    elif(ans == 1 and yBinTest[i] == 0):
        FP = FP + 1
    elif(ans == 0 and yBinTest[i] == 0):
        TN = TN + 1
    else:
        FN = FN + 1

#calculates the accuracy, precision, recall, and F1 score
accuracy = (TP + TN) / (TP + TN + FP + FN)
precision = TP / (TP + FP)
recall = TP / (TP + FN)
f1 = 2 * (1 / ((1/precision) + (1/recall)))

#prints the results
print("True positives:", TP)
print("True negatives:", TN)
print("False positives:", FP)
print("False negatives:", FN)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1:", f1)
```