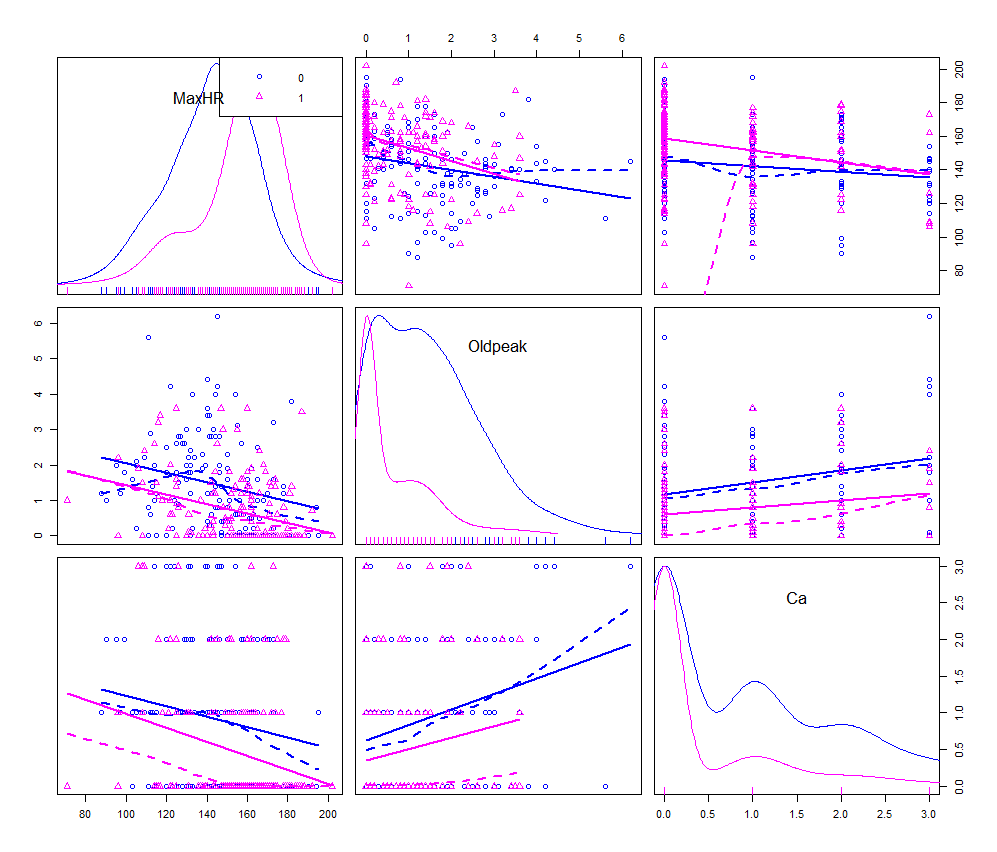
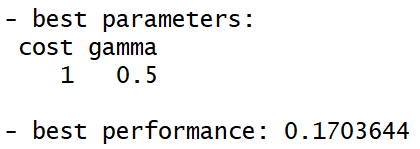
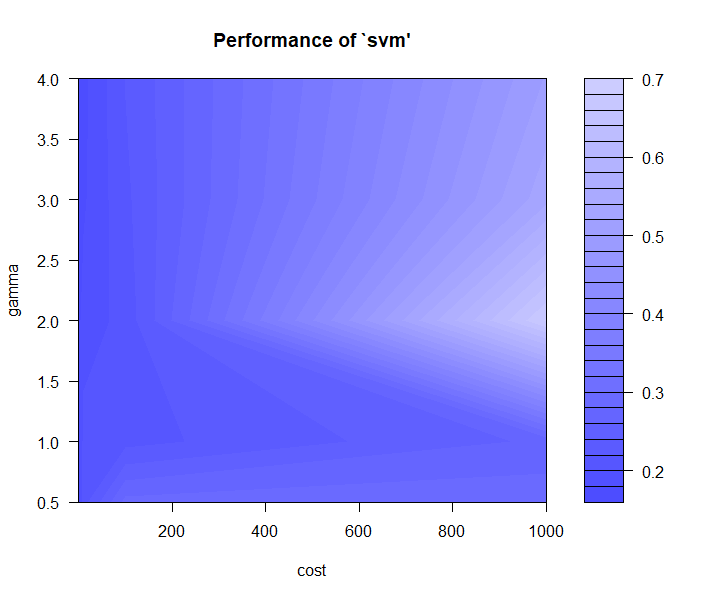
ECO 348K

**Instructor:** Sukjin Han

Assignment 5

Juan Acosta

JA45384

1. We can see in the scatterplot matrix the distribution of values for the patients with AHD\_Yes = 1 in pink and AHD\_Yes = 0 in blue. There is no clear separating hyerplane, therefore a SVM would be an appropriate clasifier as it will allow for a soft margin.
2. i) Optimal tuning parameters as calculated by R are:

Understanding the performance parameter as the MSE.

ii) looking at the performance of the SVM plot we can see how as gamma increases it lowers performance and then it begins to climb again, which is a possible indicator of bias. As the theory states, higher gamma values are linked to higher bias and lower variability.

On the other hand the cost parameter specifies the leverage that misclassification has on the hyperplane, therefore we can see how in this example a low C makes sense as the data is not very clearly linearly differentiated and we must allow for a “softer” margin. Nonetheless, this clearly increases bias as we can see from the performance of SVM plot’s left side, and from the theory, which states that lower values of C decrease variance but increase bias.

Code:

**R:**

library(car)

library(e1071)

hearts <- Hearts\_Dummy

traindata <- hearts[ which(hearts$Thal\_normal==1), ]

testdata <- hearts[ which(hearts$Thal\_normal==0), ]

traindata$Thal\_normal <- NULL

testdata$Thal\_normal <- NULL

traindata$Count <- NULL

testdata$Count <- NULL

scatterplotMatrix(~MaxHR+Oldpeak+Ca | Thal\_normal, data=hearts)

svmfit <- svm(AHD\_Yes~., data=traindata, kernel='radial', gamma=1, cost=1)

summary(svmfit)

set.seed(1)

tune.out = tune(svm, AHD\_Yes~., data=traindata, kernel='radial', ranges = list(cost=c(0.1,1,10,100,1000), gamma=c(0.5,1,2,3,4)))

summary(tune.out)

plot(tune.out)

Prediction <- predict(svmfit, testdata)

Tab <- table(pred=prediction, true=testdata$AHD\_Yes[1:135])

Tab

**Python:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import statsmodels.formula.api as smf

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

from sklearn import svm

from sklearn import metrics

df = pd.read\_csv('heart.csv')

print(df.head())

print()

colors = {0:'blue', 1:'red'}

fig, axs = plt.subplots(2, 2, sharey=False) #Sharey defines a shared y axis for the plots in each row. Similarly, sharex would share the x axis

df.plot(kind='scatter', x='Ca', y='Oldpeak', c= df['AHD\_Yes'].apply(lambda x: colors[x]), ax=axs[0][0], figsize=(16, 8))

df.plot(kind='scatter', x='Oldpeak', y='MaxHR', c= df['AHD\_Yes'].apply(lambda x: colors[x]), ax=axs[0][1])

df.plot(kind='scatter', x='Ca', y='MaxHR', c= df['AHD\_Yes'].apply(lambda x: colors[x]), ax=axs[1][0])

fig.savefig('graph.png')

train\_df = df.loc[df['Thal\_normal'] == 1]

test\_df = df.loc[df['Thal\_normal'] == 0]

print(train\_df.head())

print(test\_df.head())

clf = svm.SVC(gamma=1)

svmfit = clf.fit(train\_df[['MaxHR', 'Oldpeak', 'Ca']], train\_df[['AHD\_Yes']])

y\_pred = svmfit.predict(test\_df[['MaxHR', 'Oldpeak', 'Ca']])

print("Accuracy:",metrics.accuracy\_score(test\_df[['AHD\_Yes']], y\_pred))