HW 3

Eric Rash

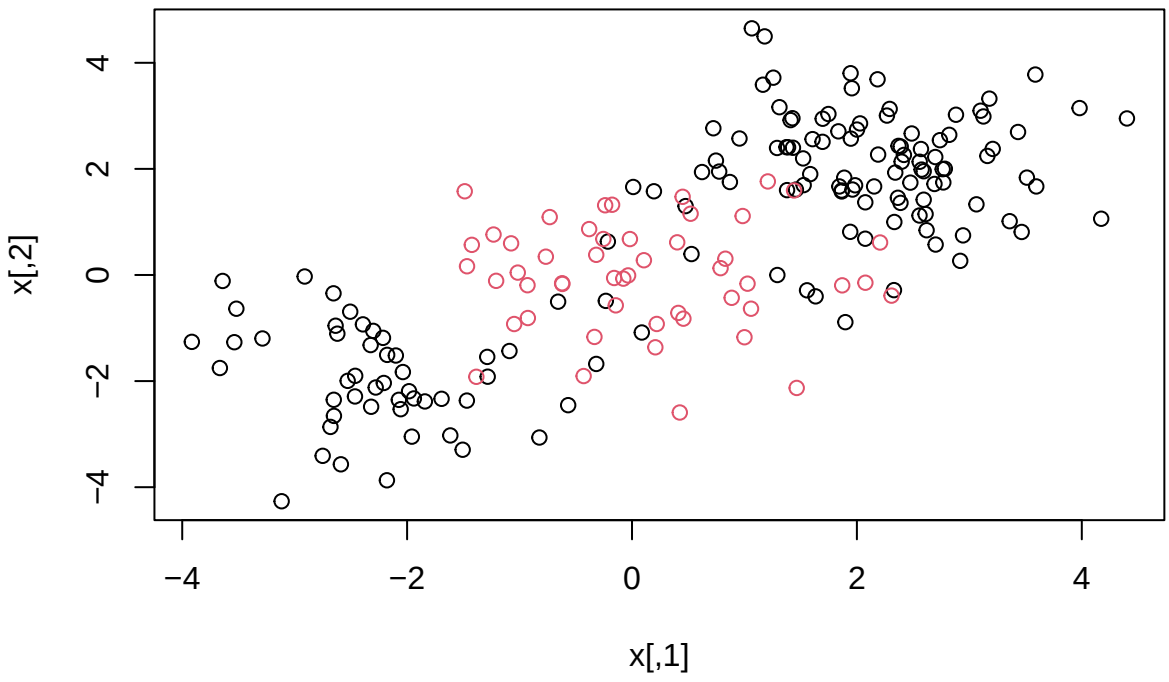
2/27/2024

In this homework, we will discuss support vector machines and tree-based methods. I will begin by simulating some data for you to use with SVM.

```
library(e1071)
```

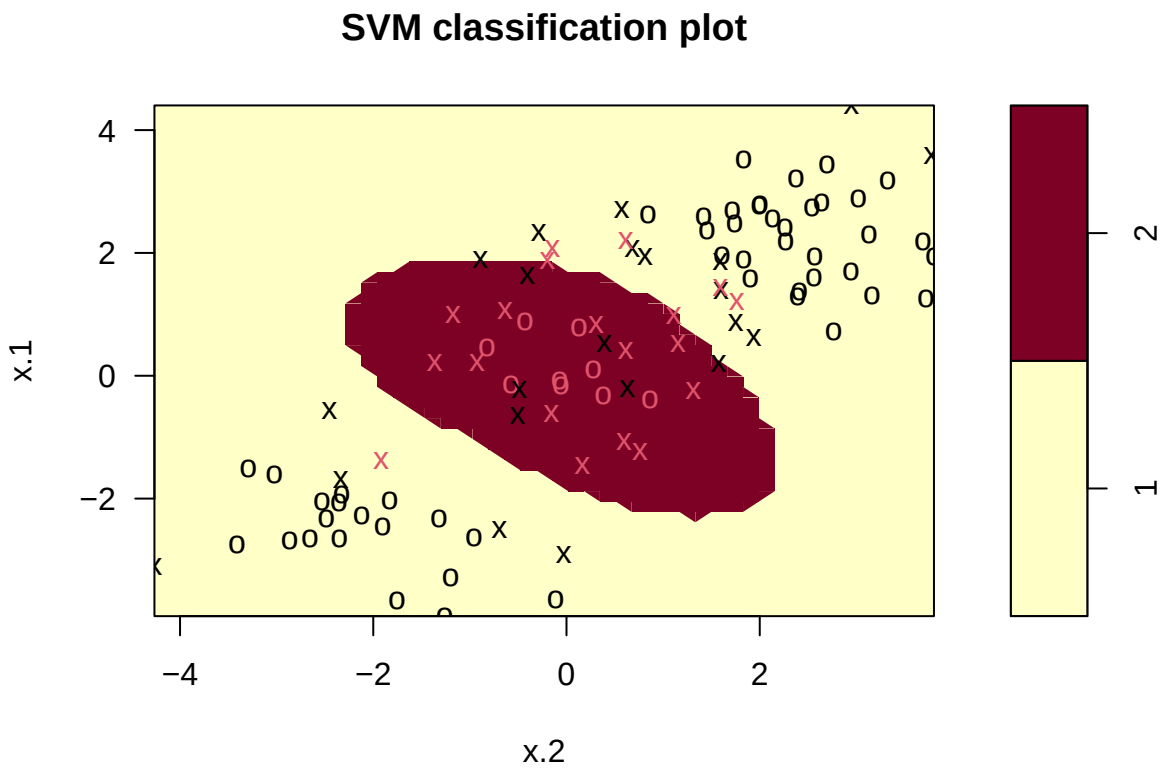
```
## Warning: package 'e1071' was built under R version 4.3.2
```

```
set.seed(1)
x=matrix(rnorm(200*2),ncol=2)
x[1:100,]=x[1:100,]+2
x[101:150,]=x[101:150,]-2
y=c(rep(1,150),rep(2,50))
dat=data.frame(x=x,y=as.factor(y))
plot(x, col=y)
```



Quite clearly, the above data is not linearly separable. Create a training-testing partition with 100 random observations in the training partition. Fit an svm on this training data using the radial kernel, and tuning parameters $\gamma = 1$, cost $= 1$. Plot the svm on the training data.

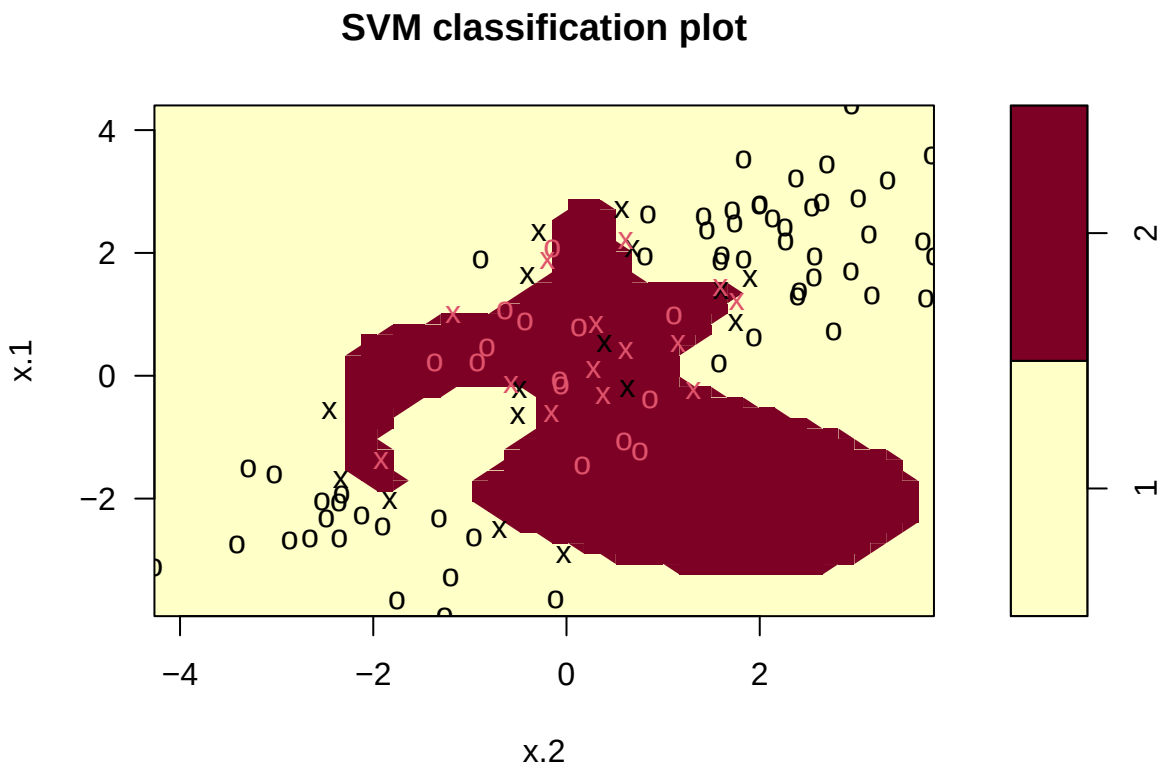
```
set.seed(55)
##creates test/train partition
train_index <- sample(1:200, 100)
train_data <- dat[train_index, ]
test_data <- dat[-train_index, ]
##fits SVM using radial kernel
model <- svm(y ~ ., data=train_data, kernel = "radial", gamma = 1, cost = 1)
##plots SVM on training data
plot(model, train_data)
```



Notice that the above decision boundary is decidedly non-linear. It seems to perform reasonably well, but there are indeed some misclassifications. Let's see if increasing the cost [1] helps our classification error rate. Refit the svm with the radial kernel, $\gamma = 1$, and a cost of 10000. Plot this svm on the training data.

```
model <- svm(y ~ ., data=train_data, kernel = "radial", gamma = 1, cost = 10000)
plot(model, train_data)
```

[1] Remember this is a parameter that decides how smooth your decision boundary should be



It would appear that we are better capturing the training data, but comment on the dangers (if any exist), of such a model.

The big concern here is overfitting the data. There is a chance that substatially increasing the cost in our model will make it much more likely to be affected by noise or outliers.

Create a confusion matrix by using this svm to predict on the current testing partition. Comment on the confusion matrix. Is there any disparity in our classification results?

```
#remove eval = FALSE in above
table(true=dat[-train_index,"y"], pred=predict(model, newdata=dat[-train_index,]))
```

```
##     pred
## true  1  2
##    1 68 10
##    2  7 15
```

Is this disparity because of imbalance in the training/testing partition? Find the proportion of class 2 in your training partition and see if it is broadly representative of the underlying 25% of class 2 in the data as a whole.

```
(sum(train_data$y==2))/(nrow(train_data))
```

```
## [1] 0.28
```

I do not think there is disparity in the classification results. Also, the split is 28% class 2, which is close to 25%.

Let's try and balance the above to solutions via cross-validation. Using the `tune` function, pass in the training data, and a list of the following cost and γ values: {0.1, 1, 10, 100, 1000} and {0.5, 1,2,3,4}. Save the output of this function in a variable called `tune.out`.

```
set.seed(1)
gammas = c(0.5,1,2,3,4)
costs=c(0.1,1,10,100,1000)


tune.out <- tune.svm(y ~ ., data= train_data, kernel="radial", rangers = list(cost=costs, gamma=gammas))
```

I will take `tune.out` and use the best model according to error rate to test on our data. I will report a confusion matrix corresponding to the 100 predictions.

```
table(true=dat[-train_index,"y"], pred=predict(tune.out$best.model, newdata=dat[-train_index,]))
```

```
##      pred
## true  1  2
##    1 72  6
##    2  2 20
```

Comment on the confusion matrix. How have we improved upon the model in question 2 and what qualifications are still necessary for this improved model.

The confusion matrix more accurately predicts on our data. It is still worth worrying about overfitting this data, especially as we increase our cost.

Let's turn now to decision trees.

```r
library(kmed)
```

```
## Warning: package 'kmed' was built under R version 4.3.2
```

```r
data(heart)
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.3.2
```

The response variable is currently a categorical variable with four levels. Convert heart disease into binary categorical variable. Then, ensure that it is properly stored as a factor.

```r
##just so you know, sometimes your questions can be confusing. For instance, which variable is our "resp

##I am leaving a second comment after finding the documentation related to these data. I wasted an hour

HeartDisease = ifelse(heart$class==0, "No", "Yes")
HeartDisease <-as.factor(HeartDisease)
class(HeartDisease)
```

```
## [1] "factor"
```

```r
heart$HeartDisease <- HeartDisease
heart = subset(heart, select = -class)
```

Train a classification tree on a 240 observation training subset (using the seed I have set for you). Plot the tree.

```r
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.3.2
```

```
## Loading required package: rpart
```

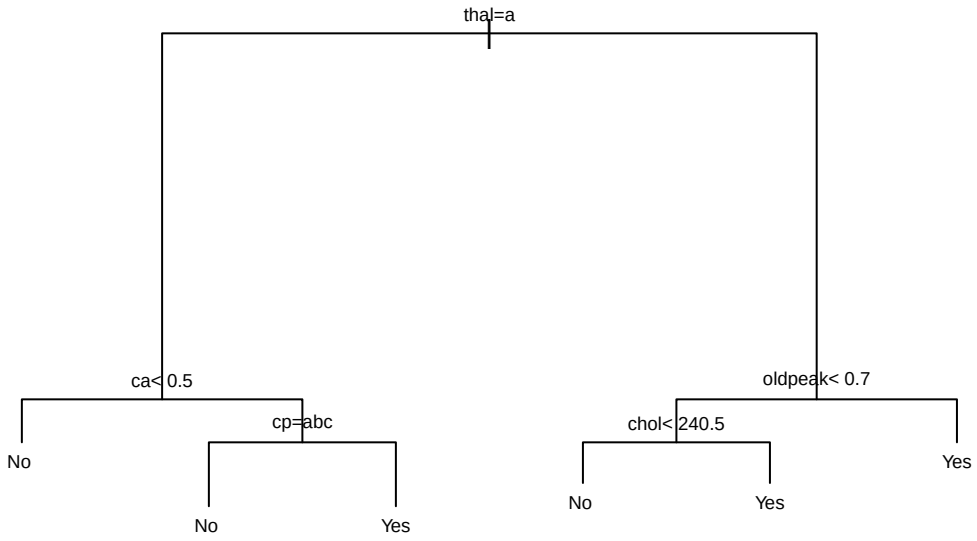
```r
set.seed(101)
train=sample(1:nrow(heart), 240)

tree_model <- rpart(HeartDisease ~ ., data = heart[train, ], method = "class")


summary(tree_model)
```

```
## Call:
## rpart(formula = HeartDisease ~ ., data = heart[train, ], method = "class")
##   n= 240
##
##           CP nsplit rel error    xerror       xstd
## 1 0.54310345      0 1.0000000 1.0000000 0.06673847
## 2 0.04741379      1 0.4568966 0.4568966 0.05539813
## 3 0.03017241      3 0.3620690 0.4224138 0.05383331
## 4 0.01000000      5 0.3017241 0.4137931 0.05342046
##
## Variable importance
##     thal       cp  thalach       ca    exang      sex  oldpeak    slope
##       26       15       13       12       11        9        5        3
##     chol trestbps  restecg      age
##        3        2        1        1
##
## Node number 1: 240 observations,    complexity param=0.5431034
##   predicted class=No   expected loss=0.4833333  P(node) =1
##     class counts:   124   116
##    probabilities: 0.517 0.483
##   left son=2 (127 obs) right son=3 (113 obs)
##   Primary splits:
##       thal    splits as  LRR,       improve=37.27507, (0 missing)
##       cp      splits as  LLLR,      improve=28.92816, (0 missing)
##       ca      < 0.5   to the left,  improve=27.24857, (0 missing)
##       exang   < 0.5   to the left,  improve=19.45949, (0 missing)
##       thalach < 147.5 to the right, improve=18.42845, (0 missing)
##   Surrogate splits:
##       cp      splits as  LLLR,      agree=0.700, adj=0.363, (0 split)
##       thalach < 150.5 to the right, agree=0.700, adj=0.363, (0 split)
##       sex     < 0.5   to the left,  agree=0.688, adj=0.336, (0 split)
##       exang   < 0.5   to the left,  agree=0.675, adj=0.310, (0 split)
##       ca      < 0.5   to the left,  agree=0.658, adj=0.274, (0 split)
##
## Node number 2: 127 observations,    complexity param=0.04741379
##   predicted class=No   expected loss=0.2204724  P(node) =0.5291667
##     class counts:    99    28
##    probabilities: 0.780 0.220
##   left son=4 (93 obs) right son=5 (34 obs)
##   Primary splits:
##       ca      < 0.5   to the left,  improve=7.255694, (0 missing)
##       cp      splits as  RLLR,      improve=6.274820, (0 missing)
##       age     < 54.5  to the left,  improve=5.953342, (0 missing)
##       chol    < 267.5 to the left,  improve=3.763888, (0 missing)
##       oldpeak < 1.7   to the left,  improve=3.746591, (0 missing)
##   Surrogate splits:
##       age     < 64.5  to the left,  agree=0.764, adj=0.118, (0 split)
##       thalach < 111.5 to the right, agree=0.756, adj=0.088, (0 split)
##       fbs     < 0.5   to the left,  agree=0.748, adj=0.059, (0 split)
##       cp      splits as  RLLL,      agree=0.740, adj=0.029, (0 split)
##
## Node number 3: 113 observations,    complexity param=0.03017241
##   predicted class=Yes  expected loss=0.2212389  P(node) =0.4708333
##     class counts:    25    88
```

```
##    probabilities: 0.221 0.779
##   left son=6 (37 obs) right son=7 (76 obs)
##   Primary splits:
##       oldpeak < 0.7   to the left,  improve=6.243885, (0 missing)
##       ca      < 0.5   to the left,  improve=5.087412, (0 missing)
##       cp      splits as  LRLR,      improve=4.805409, (0 missing)
##       thalach < 145.5 to the right, improve=3.760695, (0 missing)
##       chol    < 240.5 to the left,  improve=3.640247, (0 missing)
##   Surrogate splits:
##       slope    splits as  LRR,       agree=0.743, adj=0.216, (0 split)
##       thalach  < 146.5 to the right, agree=0.735, adj=0.189, (0 split)
##       trestbps < 109   to the left,  agree=0.708, adj=0.108, (0 split)
##       cp       splits as  RLRR,      agree=0.699, adj=0.081, (0 split)
##
## Node number 4: 93 observations
##   predicted class=No   expected loss=0.1182796  P(node) =0.3875
##     class counts:    82    11
##    probabilities: 0.882 0.118
##
## Node number 5: 34 observations,    complexity param=0.04741379
##   predicted class=No   expected loss=0.5  P(node) =0.1416667
##     class counts:    17    17
##    probabilities: 0.500 0.500
##   left son=10 (21 obs) right son=11 (13 obs)
##   Primary splits:
##       cp      splits as  LLLR,      improve=7.534799, (0 missing)
##       slope   splits as  LRR,       improve=3.051282, (0 missing)
##       thalach < 128   to the right, improve=2.248677, (0 missing)
##       age     < 55    to the left,  improve=2.185714, (0 missing)
##       sex     < 0.5   to the left,  improve=2.125000, (0 missing)
##   Surrogate splits:
##       thalach  < 121.5 to the right, agree=0.765, adj=0.385, (0 split)
##       exang    < 0.5   to the left,  agree=0.765, adj=0.385, (0 split)
##       trestbps < 115   to the right, agree=0.706, adj=0.231, (0 split)
##       oldpeak  < 0.9   to the left,  agree=0.706, adj=0.231, (0 split)
##       slope    splits as  LRR,       agree=0.706, adj=0.231, (0 split)
##
## Node number 6: 37 observations,    complexity param=0.03017241
##   predicted class=Yes  expected loss=0.4594595  P(node) =0.1541667
##     class counts:    17    20
##    probabilities: 0.459 0.541
##   left son=12 (19 obs) right son=13 (18 obs)
##   Primary splits:
##       chol     < 240.5 to the left,  improve=3.945630, (0 missing)
##       ca       < 0.5   to the left,  improve=3.160731, (0 missing)
##       trestbps < 122   to the left,  improve=1.585786, (0 missing)
##       age      < 50.5  to the right, improve=1.338696, (0 missing)
##       restecg  splits as  L-R,       improve=1.217664, (0 missing)
##   Surrogate splits:
##       thalach  < 147.5 to the right, agree=0.676, adj=0.333, (0 split)
##       trestbps < 115   to the left,  agree=0.622, adj=0.222, (0 split)
##       restecg  splits as  L-R,       agree=0.622, adj=0.222, (0 split)
##       exang    < 0.5   to the left,  agree=0.622, adj=0.222, (0 split)
##       slope    splits as  LR-,       agree=0.622, adj=0.222, (0 split)
```

```
##
## Node number 7: 76 observations
##   predicted class=Yes  expected loss=0.1052632  P(node) =0.3166667
##     class counts:     8    68
##    probabilities: 0.105 0.895
##
## Node number 10: 21 observations
##   predicted class=No   expected loss=0.2380952  P(node) =0.0875
##     class counts:    16     5
##    probabilities: 0.762 0.238
##
## Node number 11: 13 observations
##   predicted class=Yes  expected loss=0.07692308  P(node) =0.05416667
##     class counts:     1    12
##    probabilities: 0.077 0.923
##
## Node number 12: 19 observations
##   predicted class=No   expected loss=0.3157895  P(node) =0.07916667
##     class counts:    13     6
##    probabilities: 0.684 0.316
##
## Node number 13: 18 observations
##   predicted class=Yes  expected loss=0.2222222  P(node) =0.075
##     class counts:     4    14
##    probabilities: 0.222 0.778
```

```r
plot(tree_model)
par(xpd = NA)
text(tree_model, cex = 0.6)
```



Use the trained model to classify the remaining testing points. Create a confusion matrix to evaluate performance. Report the classification error rate.

```
test <- setdiff(1:nrow(heart), train)

predictions <- predict(tree_model, newdata = heart[test, ], type = "class")

actual_labels <- heart$HeartDisease[test]

conf_matrix<-table(Actual = actual_labels, Predicted = predictions)

error_rate <- 1 - sum(diag(conf_matrix)) / sum(conf_matrix)
error_rate
```

```
## [1] 0.2105263
```

```
conf_matrix
```

```
##       Predicted
## Actual No Yes
##    No  29   7
##    Yes  5  16
```

Above we have a fully grown (bushy) tree. Now, cross validate it using the `cv.tree` command. Specify cross validation to be done according to the misclassification rate. Choose an ideal number of splits, and plot this tree. Finally, use this pruned tree to test on the testing set. Report a confusion matrix and the misclassification rate.

```
##cv_result <- cv.tree(tree_model, FUN = prune.misclass)
tree_model <- rpart(HeartDisease ~ ., data = heart[train, ], method = "class")

##I clearly have made a mistake somewhere. The error I am getting just reads, "Not legitimate tree" I t

tree.heart <- tree(HeartDisease ~., heart[train, ])
cv_result <- cv.tree(tree.heart, FUN= prune.misclass)

pruned_tree <- prune(tree_model, cp = cv_result$dev[which.min(cv_result$dev)])

predictions_pruned <- predict(pruned_tree, newdata = heart[test, ], type = "class")
conf_matrix_pruned <- table(Actual = heart$HeartDisease[test], Predicted = predictions_pruned)
error_rate_pruned <- 1 - sum(diag(conf_matrix_pruned)) / sum(conf_matrix_pruned)

conf_matrix_pruned
```

```
##       Predicted
## Actual No Yes
##    No  36   0
##    Yes 21   0
```

```
error_rate_pruned
```

```
## [1] 0.3684211
```

Discuss the trade-off in accuracy and interpretability in pruning the above tree.

The pruned tree is substantially less accurate, with around a 15% increase in error rate. The trade-off here is essentially that our tree becomes more easily understood the more we prune it, however, it is generally less accurate when fitted on fewer data.

Discuss the ways a decision tree could manifest algorithmic bias.

A really easy way a decision tree could create algorithmic bias is if the tree is trained on biased data. The model has a chance to overstate those biases that may exist in these data.