HW 3

Andy Ackerman

11/27/2023

Let $E[X] = \mu$. Show that $Var[X] := E[(X - E[X])^2] = E[X^2] - (E[X])^2$. Note, all you have to do is show the second equality (the first is our definition from class).

$E[(X - E[X])^2]$

$E[(X - \mu)^2]$

$E[X^2 - 2X\mu + \mu^2]$

$E[X^2] - E[2\mu X] + E[\mu^2]$

Expectation (E) of constant μ is just μ:

$E[X^2] - 2\mu E[X] + \mu^2$

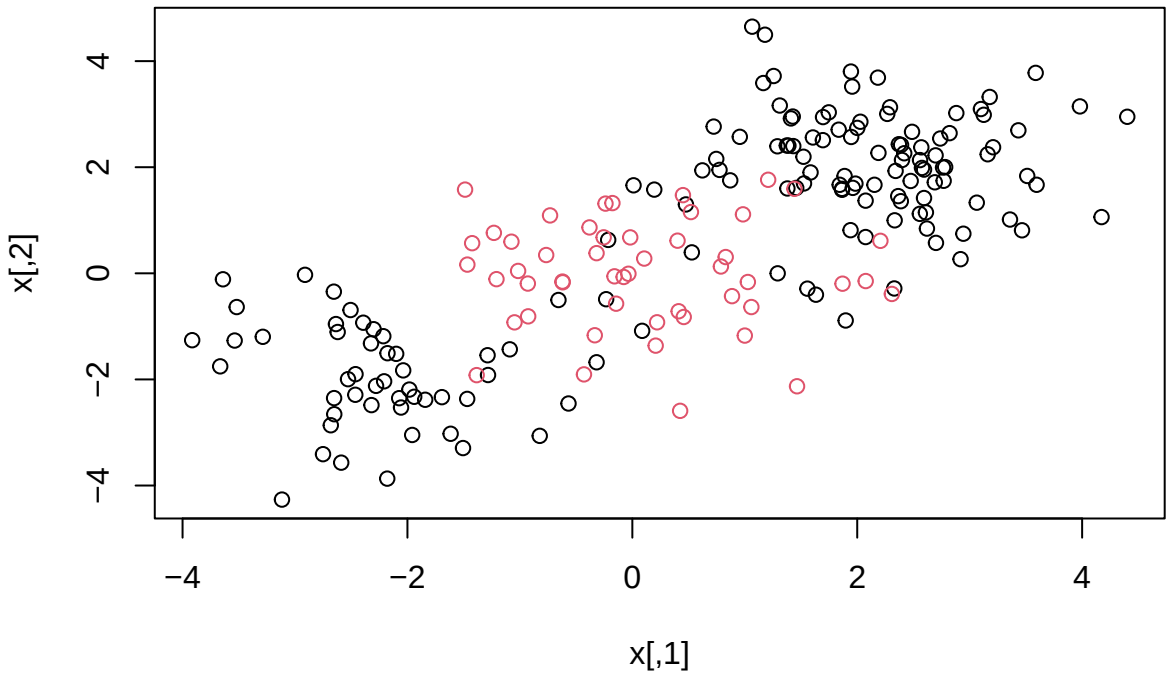
$E[X^2] - 2\mu^2 + \mu^2$

$E[X^2] - \mu^2$

$E[X^2] - (E[X])^2$

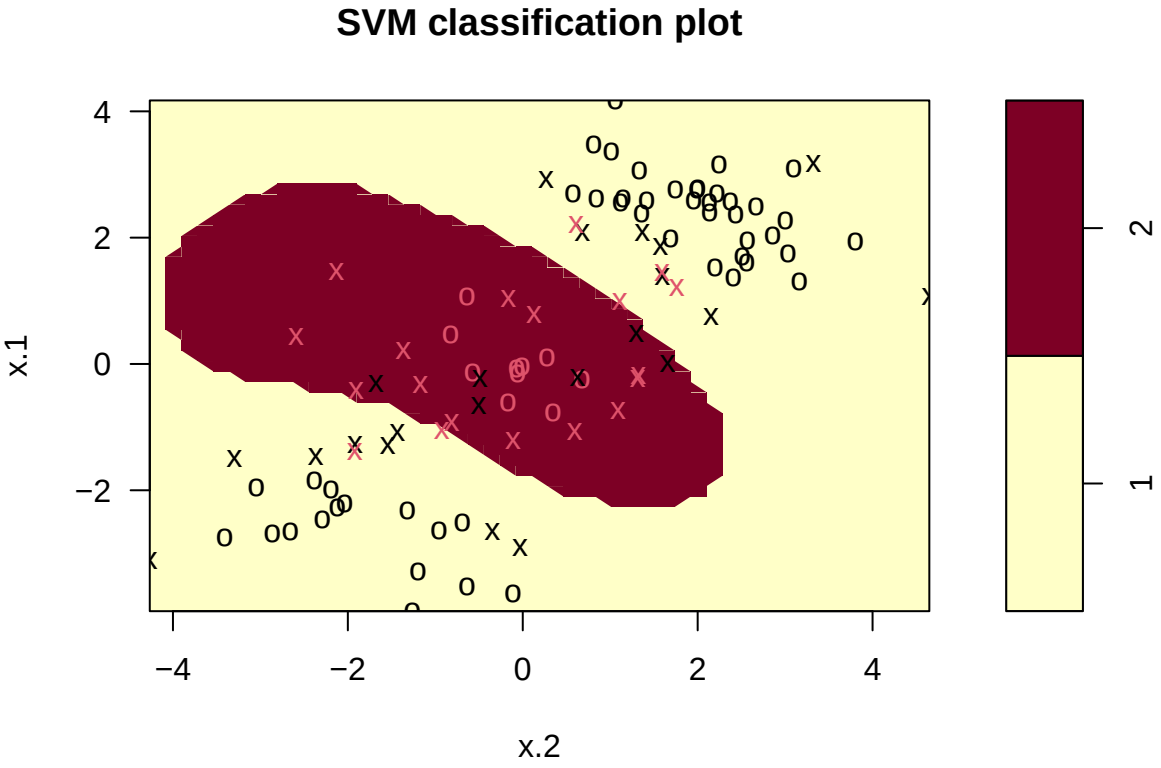
In the computational section of this homework, we will discuss support vector machines and tree-based methods. I will begin by simulating some data for you to use with SVM.

```
library(e1071)
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.

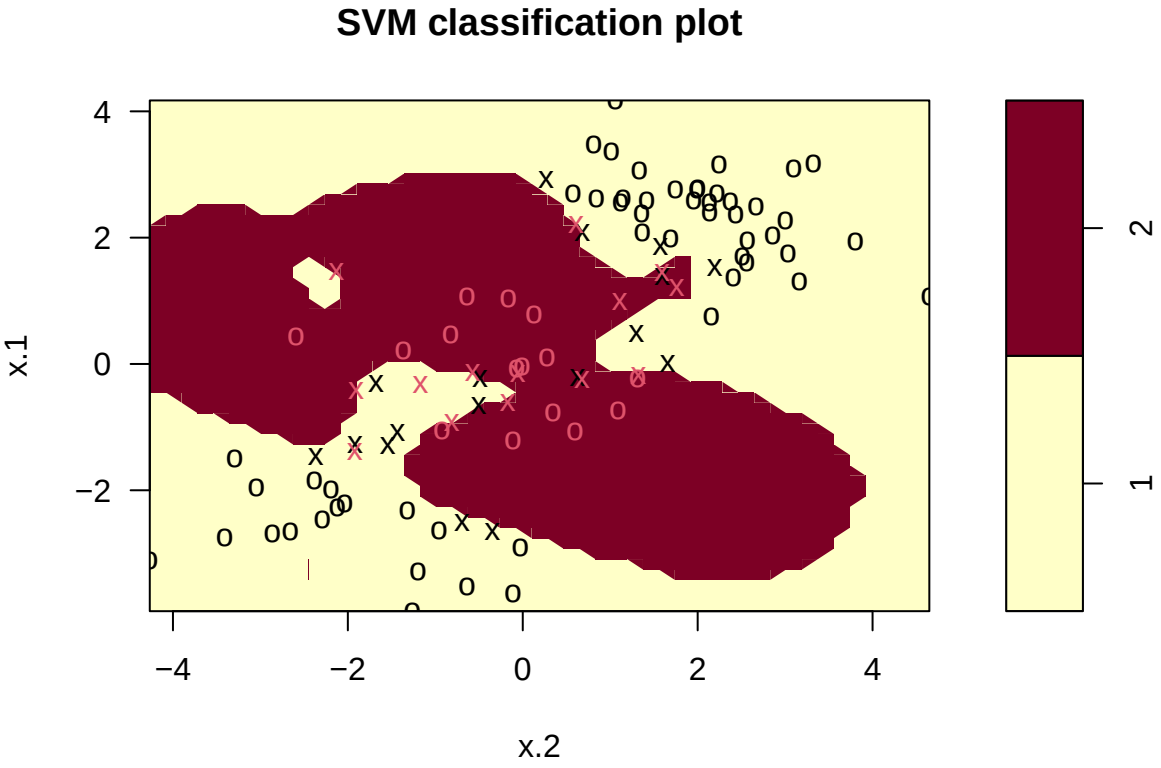
```
set.seed(1)
train=sample(200,100)
svmfit = svm(y ~ ., data = dat[train,], kernel = "radial", gamma = 1, cost = 1)
plot(svmfit, dat[train,])
```



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.

```
svmfit = svm(y ~ ., data = dat[train,], kernel = "radial", gamma = 1, cost = 10000)
plot(svmfit, dat[train,])
```

[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.

OVERFITTING

I'll add a little more to the pre-loaded response. This new training data fit has an amorphous and unnecessarily complex decision boundary, which is why it is deemed overfit.

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?

```
table(true=dat[-train,"y"], pred=predict(svmfit, newdata=dat[-train,]))
```

```
##      pred
## true  1  2
##    1 67 12
##    2  2 19
```

The svm misclassifies 12/31 points it predicts as a category 2. I would consider this significantly disparate as it is wrong nearly 39 percent of the time for the testing data. It does a much better job with category 1 testing points, predicting them to be 1 67/69 times.

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.

```r
sum(dat[train,3] == 2)/100
```

```
## [1] 0.29
```

29 percent is quite close to the underlying proportion. This disparity seems to be less an artifact of imbalance between training and testing and more just overfitting the decision boundary. If we have such an irregular decision boundary, even representative training data may not yield an unbiased model.

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`.

```r
set.seed(1)
tune.out <- tune(svm, y~., data = dat[train,], kernel = "radial", ranges = list(cost = c(0.1, 1, 10, 100
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1   0.5
##
## - best performance: 0.12
##
## - Detailed performance results:
##     cost gamma error dispersion
## 1  1e-01   0.5  0.28 0.15491933
## 2  1e+00   0.5  0.12 0.07888106
## 3  1e+01   0.5  0.15 0.10801234
## 4  1e+02   0.5  0.17 0.11595018
## 5  1e+03   0.5  0.23 0.14944341
## 6  1e-01   1.0  0.25 0.13540064
## 7  1e+00   1.0  0.14 0.09660918
## 8  1e+01   1.0  0.16 0.10749677
## 9  1e+02   1.0  0.21 0.15238839
## 10 1e+03   1.0  0.20 0.14142136
## 11 1e-01   2.0  0.28 0.14757296
## 12 1e+00   2.0  0.15 0.10801234
## 13 1e+01   2.0  0.19 0.15238839
## 14 1e+02   2.0  0.18 0.14757296
## 15 1e+03   2.0  0.23 0.12516656
## 16 1e-01   3.0  0.28 0.15491933
## 17 1e+00   3.0  0.15 0.10801234
## 18 1e+01   3.0  0.20 0.16329932
## 19 1e+02   3.0  0.20 0.13333333
## 20 1e+03   3.0  0.27 0.11595018
## 21 1e-01   4.0  0.29 0.14491377
## 22 1e+00   4.0  0.16 0.09660918
## 23 1e+01   4.0  0.18 0.13984118
## 24 1e+02   4.0  0.21 0.11972190
```

```
## 25 1e+03   4.0  0.31 0.15951315
```

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

```
##     pred
## true  1  2
##    1 72  7
##    2  1 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.

We now appear to not be overfitting as extremely as our decision boundary is likely not as irregular and our error rate is at a minimum (even on the testing set). However, because the original data had an imbalance in the classes, our classifier also misclassifies class 1 as class 2 much more often than the other way around. This is not necessarily a short-coming as much as an artifact of the underlying data, but it is worth being aware of.

Let's turn now to decision trees.

```
library(kmed)
data(heart)
library(tree)
```

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.

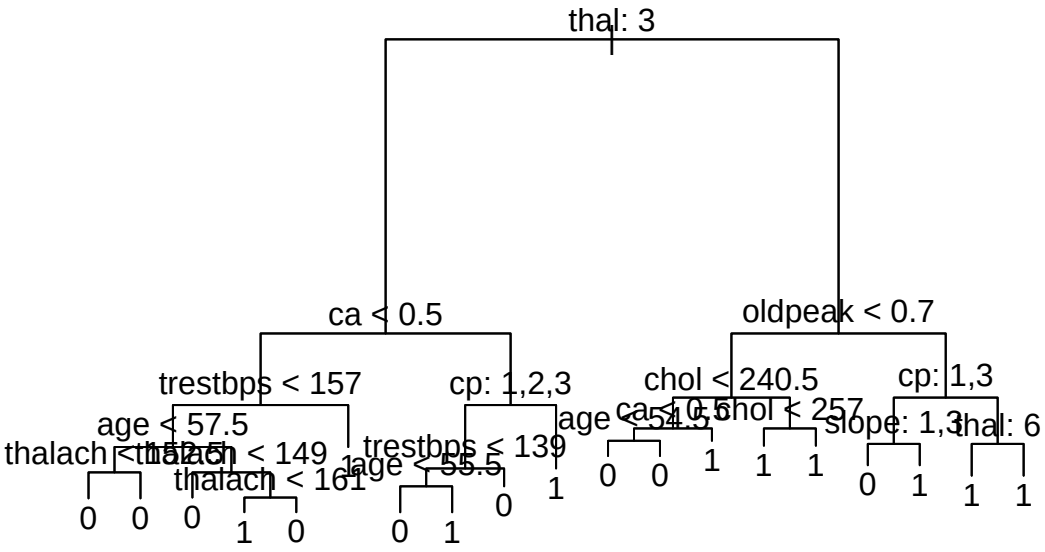
```
for (i in 1:length(heart$class)) {
  if (heart$class[i] > 0){
    heart$class[i] = 1
  }
}

heart$class = as.factor(heart$class)
```

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

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

tree.heart = tree(class~., heart, subset=train)
plot(tree.heart)
text(tree.heart, pretty=0)
```



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

```
tree.pred = predict(tree.heart, heart[-train,], type="class")
with(heart[-train,], table(tree.pred, class))
```

```
##          class
## tree.pred  0  1
##         0 28  3
##         1  8 18
```

```
1-(28+18)/57
```

```
## [1] 0.1929825
```

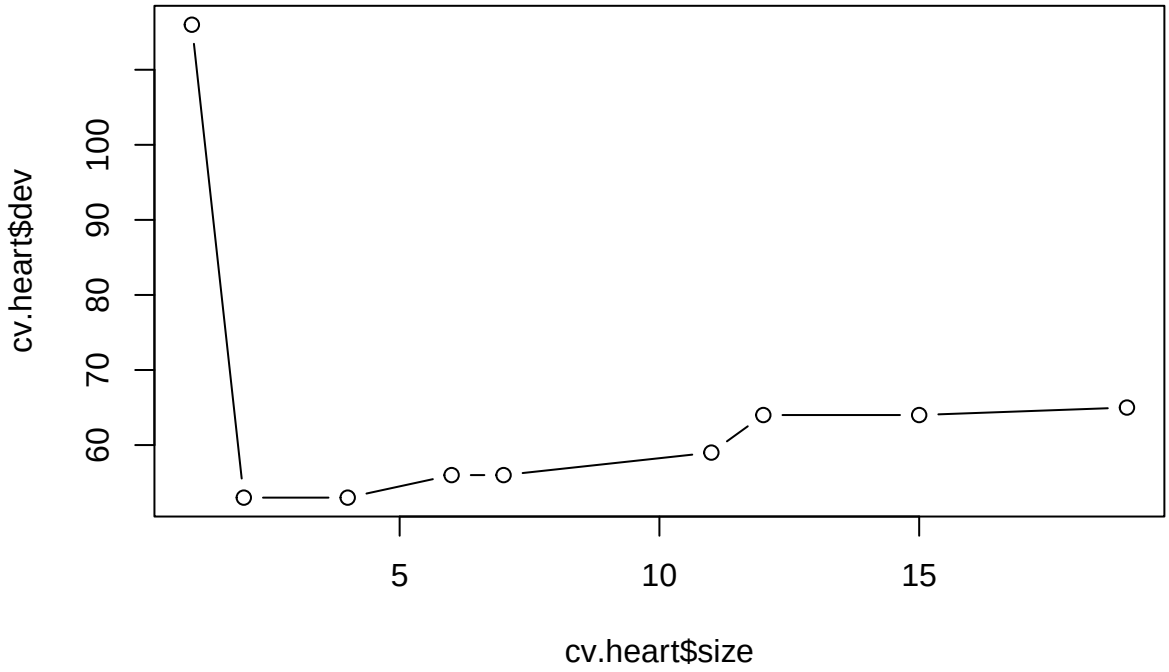
Above we have a fully grown (bushy) tree. Now, cross validate it using the `cv.tree` command. Specify cross validation to be done according to the misclassification rate. Choose an ideal number of splits, and plot this tree. Finally, use this pruned tree to test on the testing set. Report a confusion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

```
set.seed(101)
cv.heart = cv.tree(tree.heart, FUN = prune.misclass)
cv.heart
```

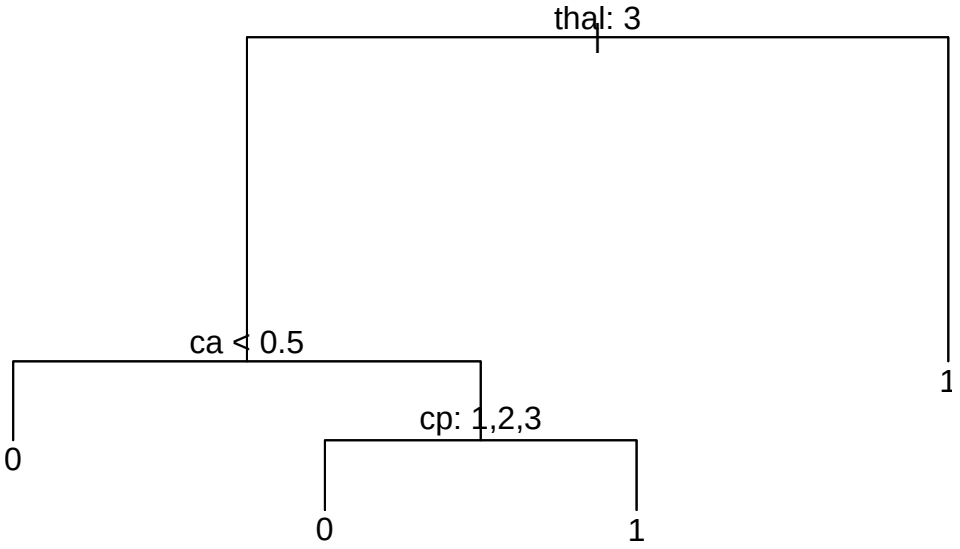
```
## $size
## [1] 19 15 12 11  7  6  4  2  1
##
## $dev
## [1]  65  64  64  59  56  56  53  53 116
##
## $k
## [1]       -Inf  0.0000000  0.6666667  1.0000000  1.5000000  2.0000000  3.5000000
## [8]  5.5000000 63.0000000
##
## $method
```

```
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
plot(cv.heart$size, cv.heart$dev, type = "b")
```




```
#three or four will probably do here
prune.heart = prune.misclass(tree.heart, best =3)
#figure out why this is above 100 percent and tell students in lecture
plot(prune.heart)
text(prune.heart, pretty=0)
```




```
tree.pred = predict(prune.heart, heart[-train,], type="class")
with(heart[-train,], table(tree.pred, class))
```

```
##          class
## tree.pred  0  1
##         0 26  4
##         1 10 17
```

```
1-((26+17)/57)
```

```
## [1] 0.245614
```

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

We have sacrificed a marginal amount of classification rate (about 4%) for a significantly more interpretable tree. It seems that `thal` *is the most influential variable followed by* `ca` *and* `cp`*.*

Discuss the ways a decision tree could manifest algorithmic bias.

A decision tree can manifest algorithmic bias when it is fully grown, or when branches haven't been pruned so that the tree is less complex (overfit) with impure nodes. As is the case with all classification algorithms we have covered thus far, a decision tree can also manifest algorithmic bias when it is grown using less than representative training/testing data.