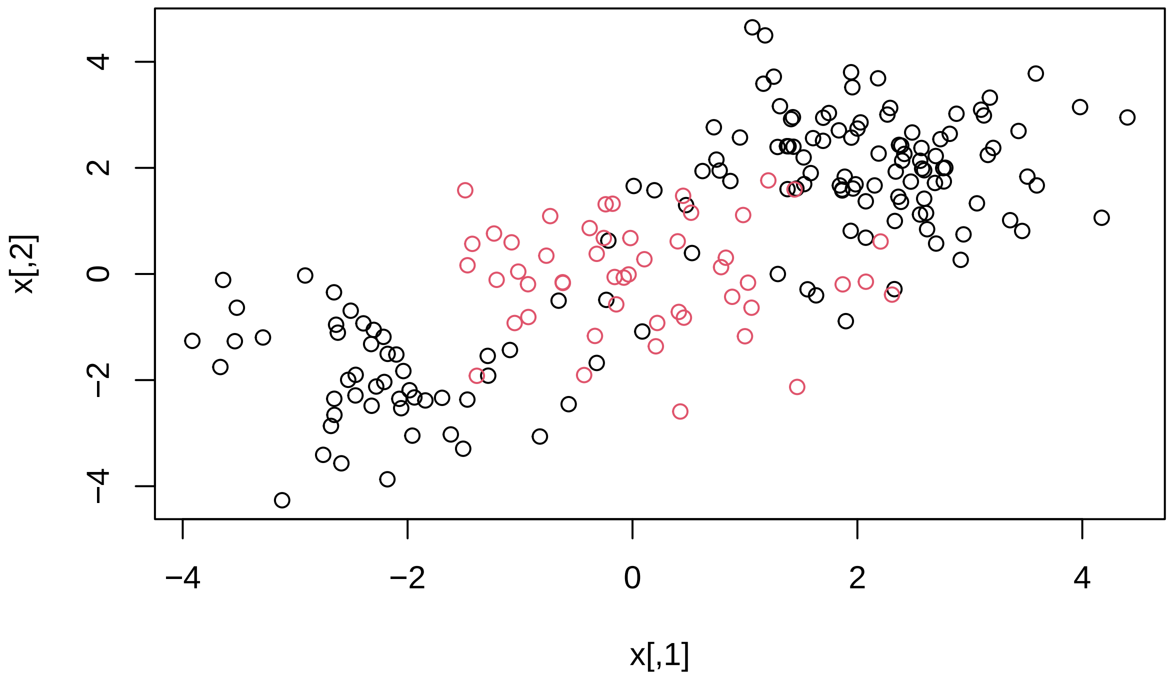
HW 3

Student Name

11/27/2023

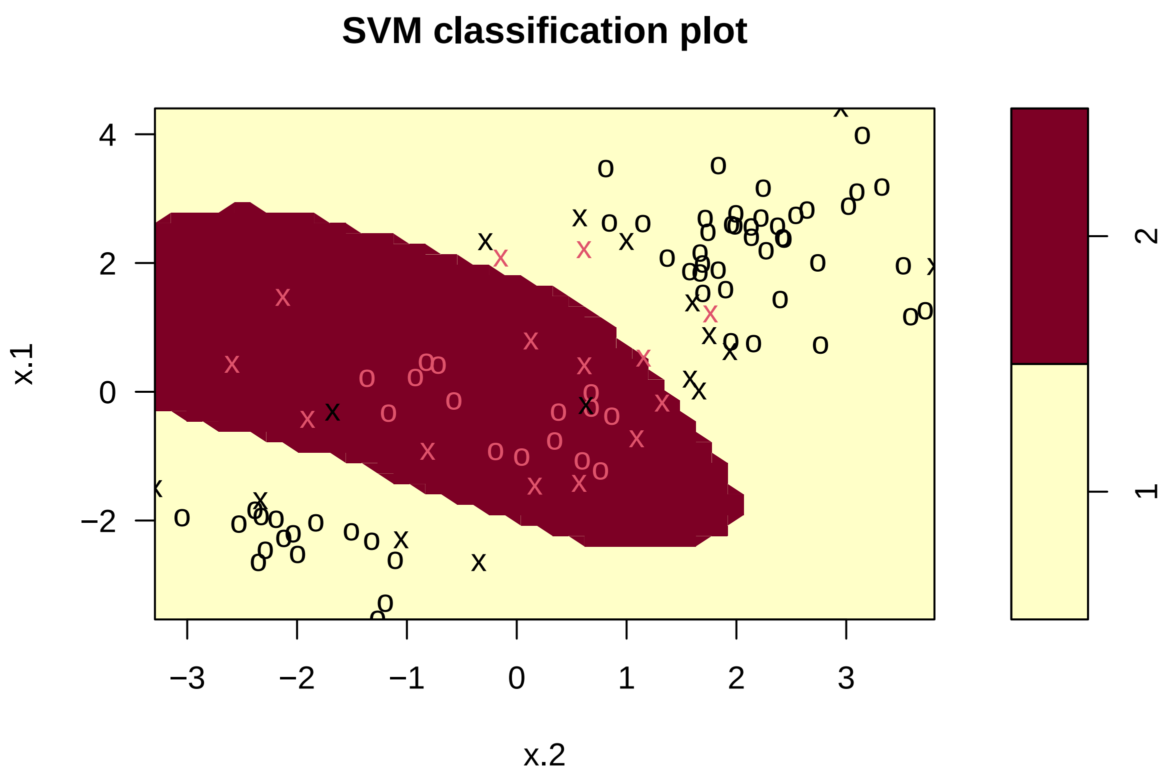
In this homework, we will discuss support vector machines and tree-based methods. I will begin by simulating some data for you to use with SVM.

```r
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.

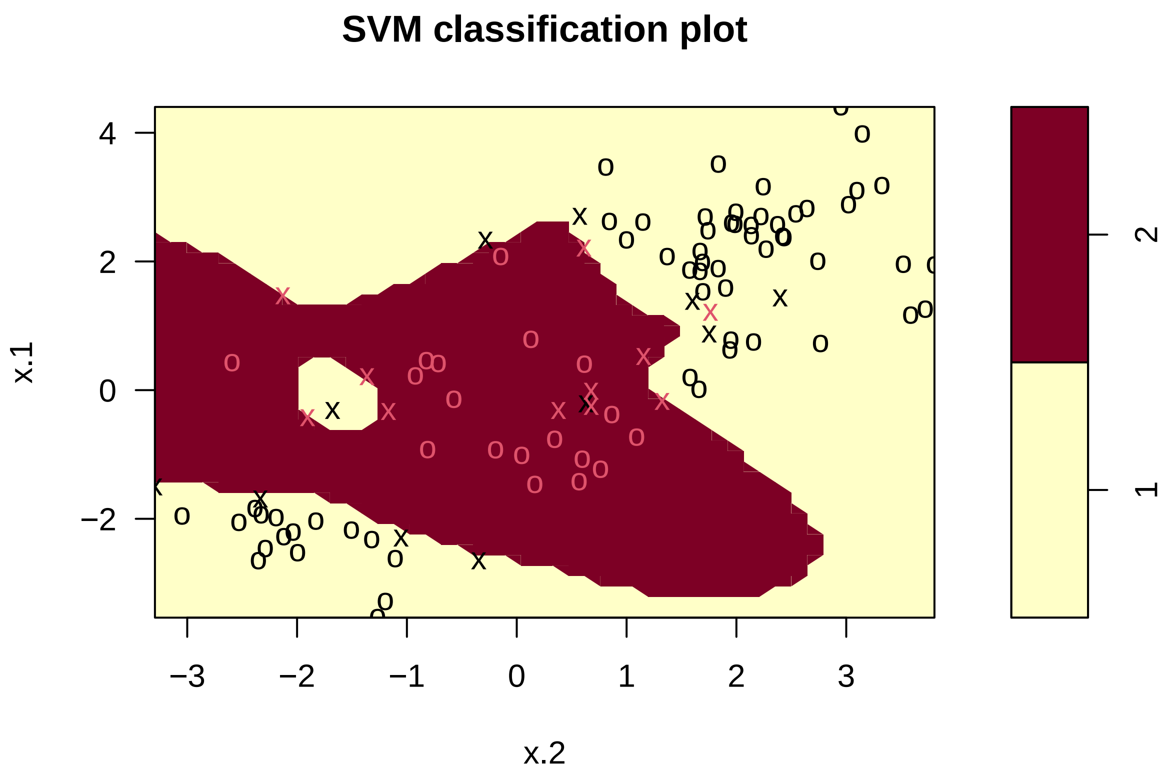
```r
set.seed(130)
datasamp <- sample(1:nrow(dat),100)
train <- dat[datasamp, ]
test <- dat[-datasamp, ]
svmfit <- svm(y ~ ., data = train, kernel = "radial", gamma = 1, cost = 1)
plot(svmfit, train)
```



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.

```r
set.seed(130)
svmfit <- svm(y ~ ., data = train, kernel = "radial", gamma = 1, cost = 10000)
plot(svmfit, 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.

It would appear that we are better capturing the training data, but we could be concerned about overfitting the data to the training set. Because we have a higher cost, our model is more sensitive to the individual data points of the training data and may poorly generalize the data outside of this training set.

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[-datasamp,"y"], pred=predict(svmfit, newdata=dat[-datasamp,]))
```

```
##      pred
## true  1  2
##    1 64 15
##    2  3 18
```

Based on the confusion matrix, our model correctly classified 64 class 1 and 18 class 2. However, it misclassified 3 class 2's and 15 class 1's.

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(train$y == "2") / nrow(train)
```

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

```r
sum(dat$y == "2") / nrow(dat)
```

```
## [1] 0.25
```

This disparity could be because of an imbalance in the training/testing partition. This is because the proportion of class 2 in my training partition is 4% higher than in the entire dataset. Therefore, it may not be broadly representative of the underlying class distribution of the data as a whole.

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)

cost_values <- c(0.1, 1, 10, 100, 1000)
gamma_values <- c(0.5, 1, 2, 3, 4)

tune.out <- tune(svm, y ~ ., data = train, kernel = "radial", ranges = list(C = cost_values, gamma = ga

print(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##    C gamma
##  0.1     1
##
## - best performance: 0.05
```

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.

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

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.

Based on the confusion matrix, our model has correctly classified 69 instances of class 1 and 16 instances of class 2, but has misclassified 5 instances of class 2 and 10 instances of class 1. Compared to our previous confusion matrix, our model has slightly improved with an increase of 5 correctly classified class 1's and only 10 misclassified 1's. For the improvement of this model, we could look into potential class imbalances, context of the data, or the potential of overfitting the data.

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.

```
#original dataset
str(heart)
```

```
## 'data.frame':    297 obs. of  14 variables:
##  $ age     : num  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : logi  TRUE TRUE TRUE TRUE FALSE TRUE ...
##  $ cp      : Factor w/ 4 levels "1","2","3","4": 1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: num  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : num  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : logi  TRUE FALSE FALSE FALSE FALSE FALSE ...
##  $ restecg : Factor w/ 3 levels "0","1","2": 3 3 3 1 3 1 3 1 3 3 ...
##  $ thalach : num  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : logi  FALSE TRUE TRUE FALSE FALSE FALSE ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope   : Factor w/ 3 levels "1","2","3": 3 2 2 3 1 1 3 1 2 3 ...
##  $ ca      : num  0 3 2 0 0 0 2 0 1 0 ...
##  $ thal    : Factor w/ 3 levels "3","6","7": 2 1 3 1 1 1 1 1 3 3 ...
##  $ class   : int  0 2 1 0 0 0 3 0 2 1 ...
##  - attr(*, "na.action")= 'omit' Named int [1:6] 88 167 193 267 288 303
##   ..- attr(*, "names")= chr [1:6] "88" "167" "193" "267" ...
```

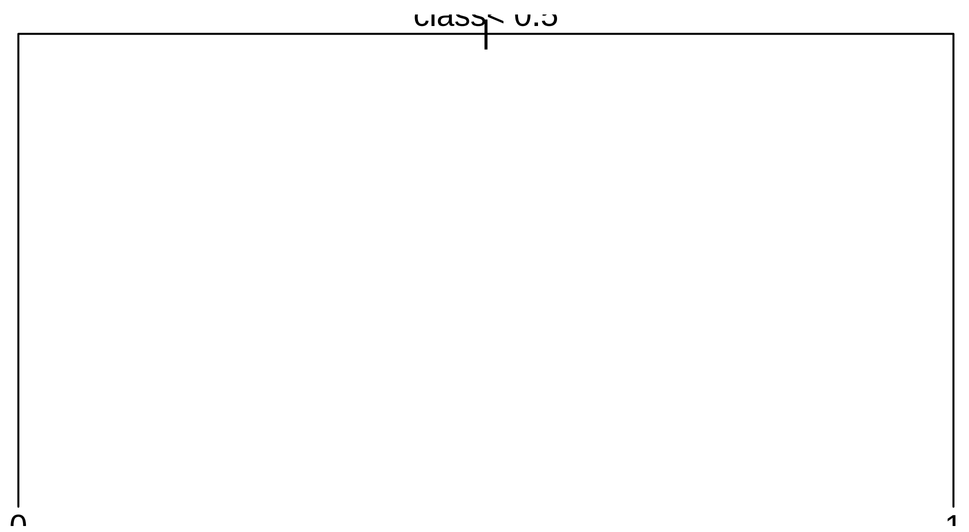
```
heart$class_binary <- ifelse(heart$class == 0, 0, 1)
heart$class_binary <- as.factor(heart$class_binary)

#modified dataset
str(heart)
```

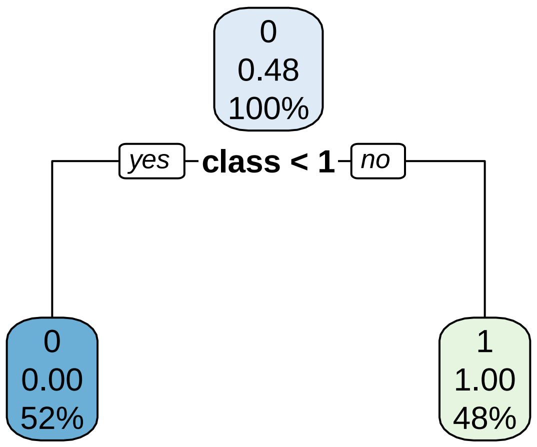
```
## 'data.frame':    297 obs. of  15 variables:
##  $ age         : num  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex         : logi  TRUE TRUE TRUE TRUE FALSE TRUE ...
##  $ cp          : Factor w/ 4 levels "1","2","3","4": 1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps    : num  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol        : num  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs         : logi  TRUE FALSE FALSE FALSE FALSE FALSE ...
##  $ restecg     : Factor w/ 3 levels "0","1","2": 3 3 3 1 3 1 3 1 3 3 ...
##  $ thalach     : num  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang       : logi  FALSE TRUE TRUE FALSE FALSE FALSE ...
##  $ oldpeak     : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope       : Factor w/ 3 levels "1","2","3": 3 2 2 3 1 1 3 1 2 3 ...
##  $ ca          : num  0 3 2 0 0 0 2 0 1 0 ...
##  $ thal        : Factor w/ 3 levels "3","6","7": 2 1 3 1 1 1 1 1 3 3 ...
##  $ class       : int  0 2 1 0 0 0 3 0 2 1 ...
##  $ class_binary: Factor w/ 2 levels "0","1": 1 2 2 1 1 1 2 1 2 2 ...
##  - attr(*, "na.action")= 'omit' Named int [1:6] 88 167 193 267 288 303
##   ..- attr(*, "names")= chr [1:6] "88" "167" "193" "267" ...
```

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

```r
library(rpart)
library(rpart.plot)
set.seed(101)
train1 <- sample(1:nrow(heart), 240)
train2 <- heart[train1, ]
test2 <- heart[-train1, ]
tree <- rpart(class_binary ~ ., data = train2, method = "class")
plot(tree)
text(tree, pretty = 0)
```



```r
rpart.plot(tree)
```



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

```
prediction <- predict(tree, newdata = test2, type = "class")
confusion_matrix <- table(true = test2$class_binary, pred = prediction)
error_rate <- 1 - sum(diag(confusion_matrix)) / sum(confusion_matrix)
error_rate
```

```
## [1] 0
```

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.

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

The trade-off in accuracy and interpretability in pruning a tree can be to create a simpler and more interpretable data that can be more generalized than a very complex and possibly more accurate tree.

Discuss the ways a decision tree could manifest algorithmic bias.

A decision tree can manifest algorithmic bias with a biased training dataset or class imbalance. If the training data is biased, the decision tree will additionally model this bias. Furthermore, if there is a bias in class distribution, the decision tree will also model this imbalance.