Chase Moreno

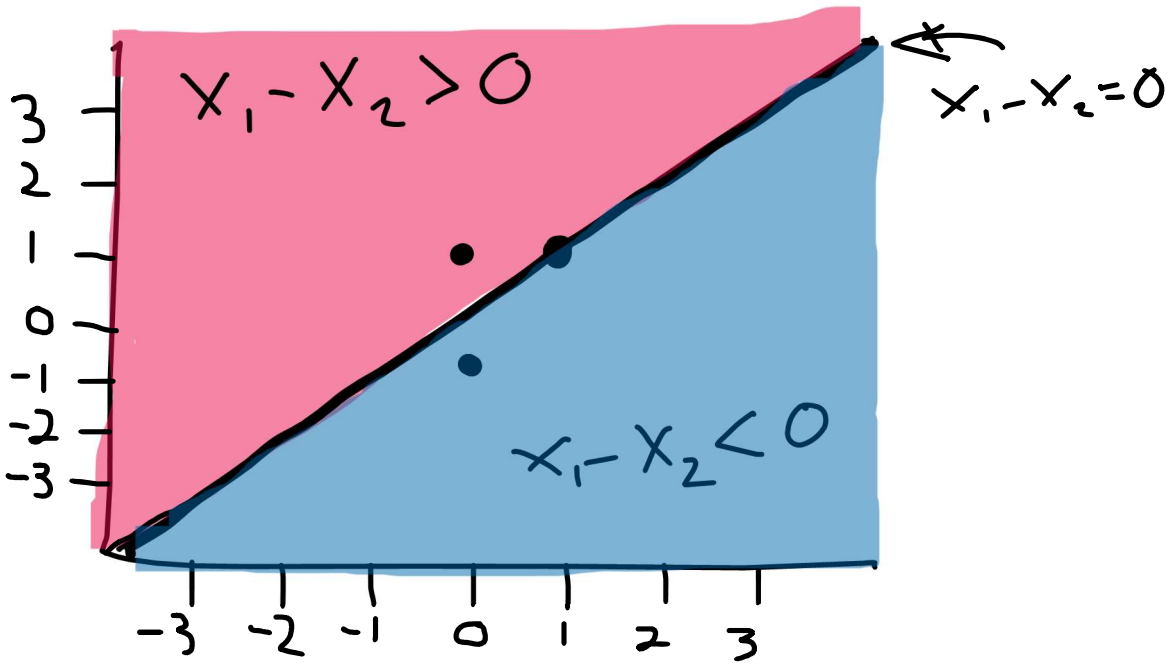
@2078503

MATH 4323

Homework 3

1. This problem involves hyperplanes in two dimensions.

a) Sketch the hyperplane $X_1 - X_2 = 0$. Indicate (by using either different colors or symbols) the set of points for which $X_1 - X_2 > 0$, as well as the set of points for which $X_1 - X_2 < 0$.



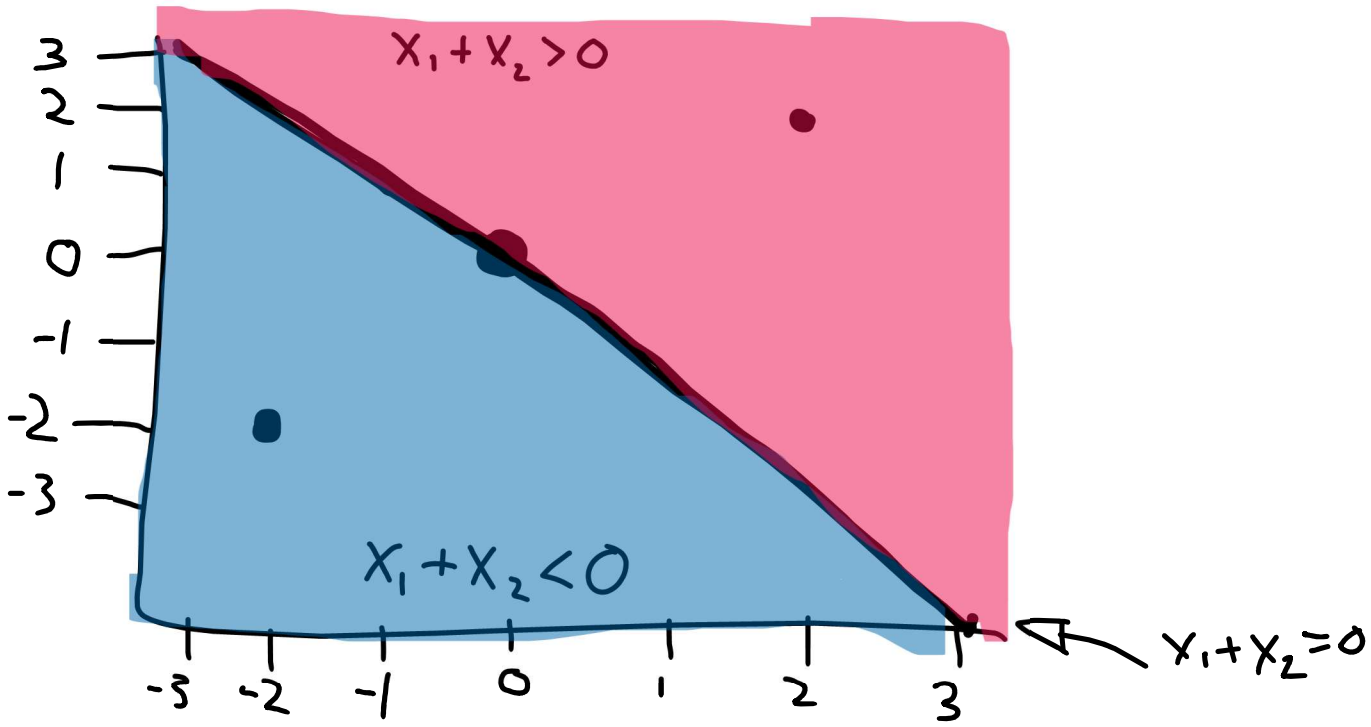

To which class do the following points belong:

i. $X_1 = 1, X_2 = 1$. **On the hyperplane, $X_1 - X_2 = 0$**

ii. $X_1 = 0, X_2 = -1$. **In red side, $X_1 - X_2 > 0$**

iii. $X_1 = 0, X_2 = 1$. **In blue side, $X_1 - X_2 < 0$**

b) On a different plot, sketch the hyperplane $X_1 + X_2 = 0$. Indicate (by using either different colors or symbols) the set of points for which $X_1 + X_2 > 0$, as well as the set of points for which $X1 + X_2 < 0$.




To which class do the following points belong:

i. $X_1 = 1, X_2 = 1$. **In red side, $X_1 + X_2 > 0$**

ii. $X_1 = -1, X_2 = -1$. **In blue side, $X_1 + X_2 < 0$**

iii. $X_1 = 1, X_2 = -1$. **On the Hyperplane, $X_1 + X_2 = 0$**

2. We have seen that in *p = 2* dimensions, a linear decision boundary takes the form $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$. We now investigate a non-linear decision boundary.

(a) Sketch the curve

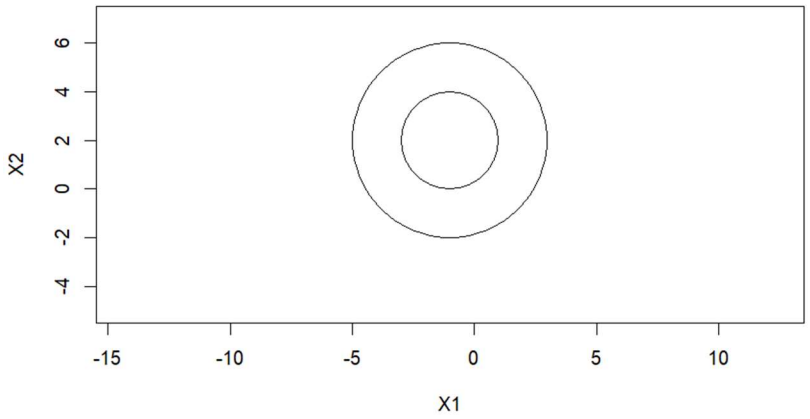
$$(1 + X_1)^2 + (2 - X_2)^2 = 4.$$

And

$$(1 + X_1)^2 + (2 - X_2)^2 = 16.$$

**My R code:**

```
> plot(NA, NA, type = "n", xlim = c(-7,5), ylim = c(-5,7),
+      asp = 1, xlab = "X1", ylab = "X2")
> symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)
> symbols(c(-1), c(2), circles = c(4), add = TRUE, inches = FALSE)
```




(b) On your sketch, indicate the set of points for which

$$(1 + X_1)^2 + (2 - X_2)^2 > 16,$$

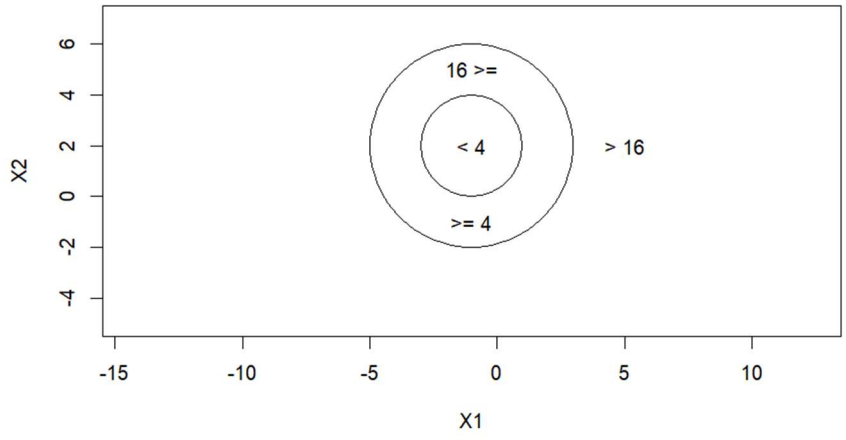
$$16 \geq (1 + X_1)^2 + (2 - X_2)^2 \geq 4,$$

as well as the set of points for which

$$(1 + X_1)^2 + (2 - X_2)^2 < 4.$$

**My R code:**

```
> text(c(5), c(2), "> 16")
> text(c(-1), c(5), "16 >=")
> text(c(-1), c(-1), ">= 4")
> text(c(-1), c(2), "< 4")
```



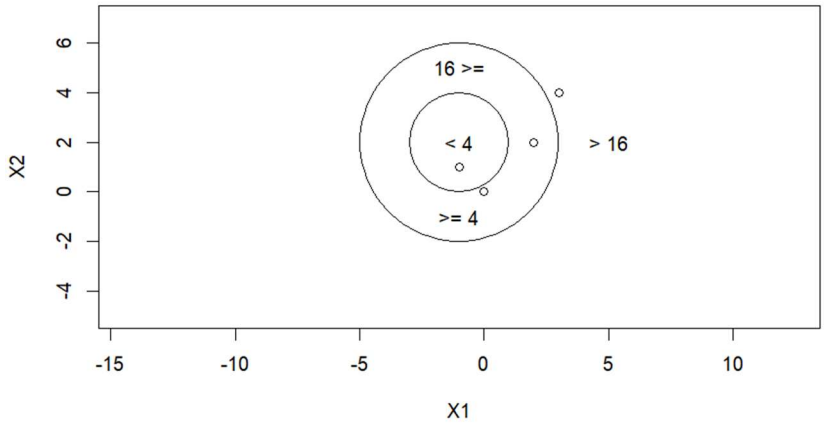

(c) Suppose that a classifier assigns an observation to the blue class if

$$16 \geq (1 + X_1)^2 + (2 - X_2)^2 \geq 4$$

and to the red class otherwise. To what class is the observation (0, 0) classified? (−1, 1)? (2, 2)? (3, 4)?

**My R code:**

```
> points(0,0)
> points(-1,1)
> points(2,2)
> points(3,4)
```




**Point (0, 0) is classified blue because it's in circle of $16 \geq (1 + X_1)^2 + (2 - X_2)^2 \geq 4$.**

**Point (-1,1) is classified red because it's less than 4.**

**Point (2,2) is classified blue because it's in the circle of $16 \geq (1 + X_1)^2 + (2 - X_2)^2 \geq 4$.**

**Point (3,4) is classified red because it's outside the circle of $16 \geq (1 + X_1)^2 + (2 - X_2)^2 \geq 4$.**

(d) Argue that while the decision boundary in (c) is not linear in terms of $X_1$ and $X_2$, it is linear in terms of $X_1$, $X_1^2$, $X_2$, and $X_2^2$.

**Here is the formula from the lecture that we can extend 2p-feature space:**

$$\beta_0 + \beta_{11}X_1 + \beta_{12}X_1^2 + \beta_{21}X_2 + \beta_{22}X_2^2 + \cdots + \beta_{p1}X_p + \beta_{p2}X_p^2 = 0$$

**Which we get,**

$$(1 + X_1)^2 + (2 - X_2)^2 = 4$$

$$1 + 2X_1 + X_1^2 + (2 - X_2)^2 = 4$$

$$1 + 2X_1 + X_1^2 + 4 - 4X_2 + X_2^2 = 4$$

$$1 + 2X_1 + X_1^2 - 4X_2 + X_2^2 = 0$$

**Now the decision boundary is linear due to terms of $X_1$, $X_1^2$, $X_2$, and $X_2^2$.**

3. Here we explore the maximal margin classifier on a toy data set.

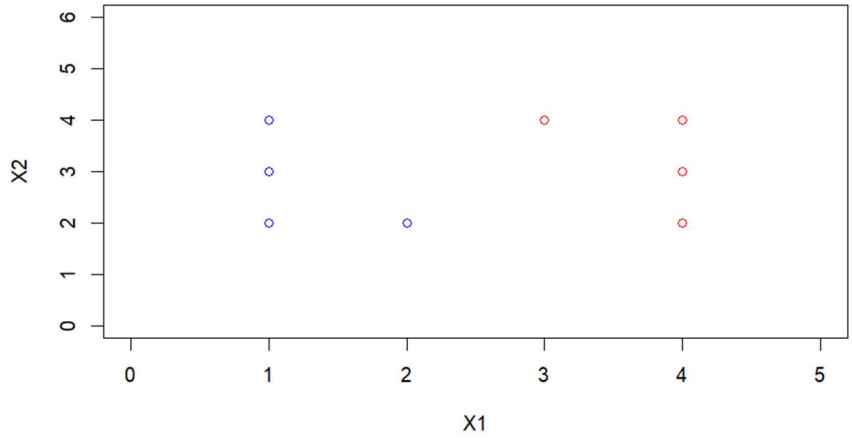
(a) We are given n = 8 observations in p = 2 dimensions. For each observation, there is an associated class label.

| Obs. | $X_1$ | $X_2$ | $Y$ |
|---|---|---|---|
| 1 | 1 | 4 | Blue |
| 2 | 3 | 4 | Red |
| 3 | 4 | 3 | Red |
| 4 | 2 | 2 | Blue |
| 5 | 4 | 2 | Red |
| 6 | 4 | 4 | Red |
| 7 | 1 | 2 | Blue |
| 8 | 1 | 3 | Blue |

Sketch the observations.

**My R code:**

```
> X1 <- c(1, 3, 4, 2, 4, 4, 1, 1)
> X2 <- c(4, 4, 3, 2, 2, 4, 2, 3)
> Y <- c("blue", "red", "red", "blue", "red", "red", "blue", "blue")
> plot(X1, X2, col = Y, xlim = c(0, 5), ylim = c(0, 6))
```




(b) Sketch the optimal separating hyperplane and provide the equation for this hyperplane (of the form (9.1) from the book pdf).

**From seeing the plot, we can see that we have a pattern from these points forming sort of a rectangular shape from points (1, 4) to (4, 2). So, there are two missing points, which are (2, 4) and (3, 2). I figure we can use these two points for our hyperplane because we see that the blue points are on the left side of the plot and the red points are on the right side of the plot.**

**I calculate the slope by using points (2, 4) and (3, 2):**

**Slope = (4 – 2) / (2 – 3) => 2 / -1 => -2.**

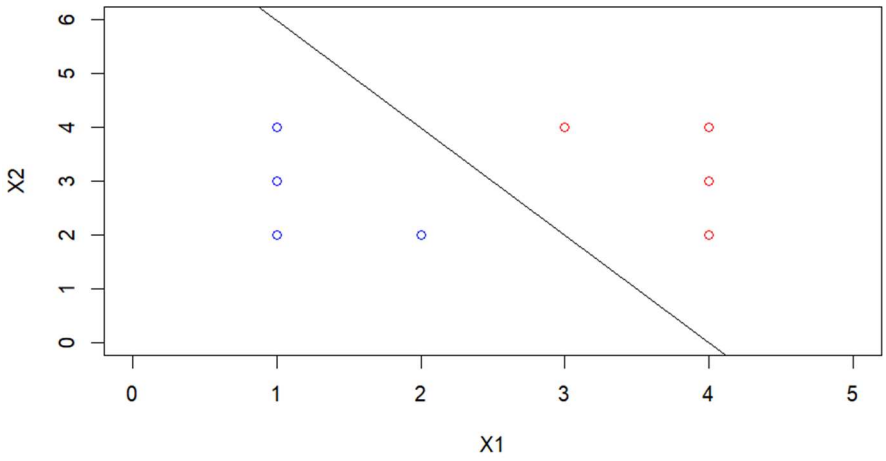
**I calculate the y-intercept using point (3, 2):**

**Y = mx + b => 2 = -2(3) + b => 2 = -6 + b => b = 8**

**Now I have my equation for this hyperplane which is y = -2x + 8 => 8 – $2X_1$ – $X_2$ = 0**

**My R code:**

```
> abline(8, -2)
```



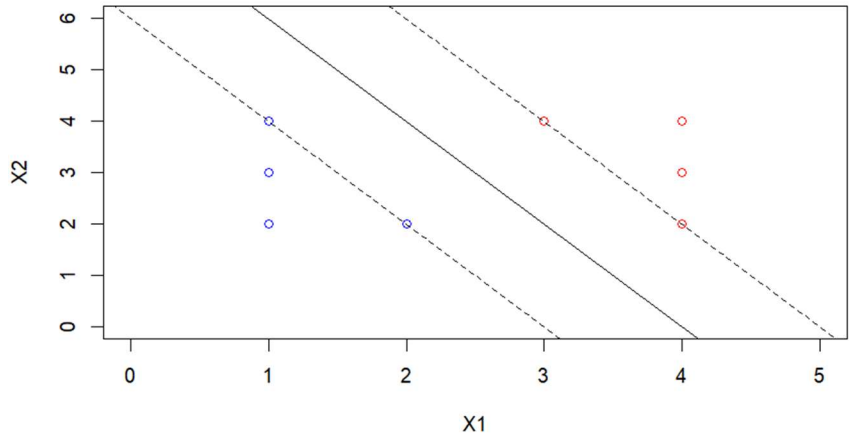

(c) Describe the classification rule for the maximal margin classifier. It should be something along the lines of "Classify to Red if $\beta_0 + \beta_1X_1 + \beta_2X_2 > 0$ and classify to Blue otherwise." Provide the values for $\beta_0$, $\beta_1$, and $\beta_2$.

**Classify to Red if -8 + $2X_1$ + $X_2$ > 0 and classify to Blue otherwise. My beta values are $\beta_0$ = -8, $\beta_1$ = 2 and $\beta_2$ = 1.**

(d) On your sketch, indicate the margin for the maximal margin hyperplane.

**My R code:**

```
> abline(6, -2, lty = "dashed")
> abline(10, -2, lty = "dashed")
```




(e) Indicate the support vectors for the maximal margin classifier.

**The support vectors for blue observations are (1, 4) and (2, 2). The support vectors for red observations are (3, 4) and (4, 2).**

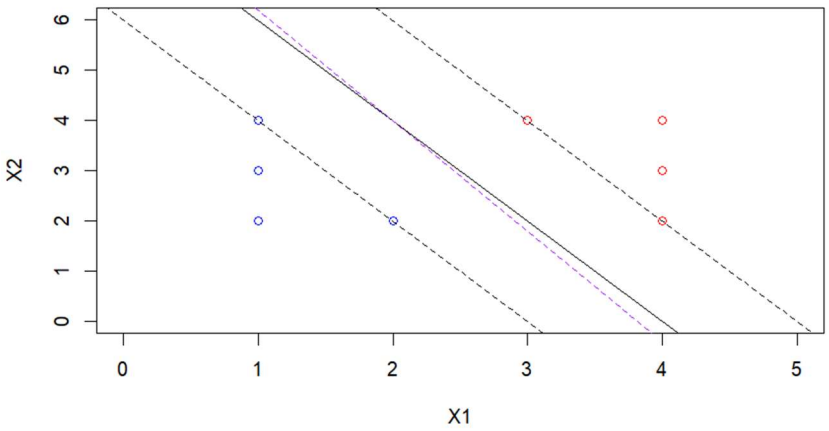
(f) Argue that a slight movement of the seventh observation would not affect the maximal margin hyperplane.

**The seventh observation is far away from the maximal margin hyperplane, so it will not affect the maximal margin hyperplane. Also, this seventh observation is not a support vector and is not on the wrong side of the margin.**

(g) Sketch a hyperplane that is not the optimal separating hyperplane and provide the equation for this hyperplane.

**My R code:**

```
> abline(8.4, -2.2, col = "purple", lty = "dashed")
```

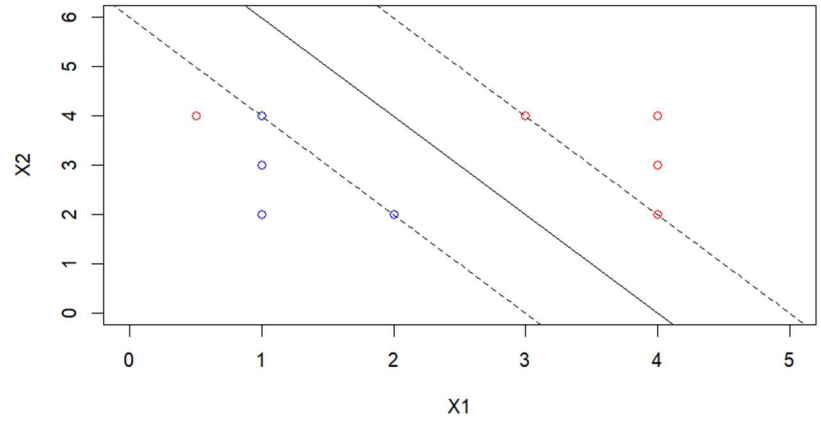



(h) Draw an additional observation on the plot so that the two classes are no longer separable by a hyperplane.

**Adding point (0.5, 4) and colored red.**

**My R code:**

```
> X1 <- c(1, 3, 4, 2, 4, 4, 1, 1, 0.5)
> X2 <- c(4, 4, 3, 2, 2, 4, 2, 3, 4)
> Y <- c("blue", "red", "red", "blue", "red", "red", "blue", "blue", "red")
> plot(X1, X2, col = Y, xlim = c(0, 5), ylim = c(0, 6))
> abline(8, -2)
> abline(6, -2, lty = "dashed")
> abline(10, -2, lty = "dashed")
```




4. Using the Boston data set of library MASS, fit classification models in order to predict whether a given suburb has a house value (medv) above or below the median (median(medv)). In particular, explore various KNN models.

(a) Create a medv01 variable, where:

$$medv01 = \begin{cases} 0, & medv \leq median(medv), \\ 1, & medv > median(medv) \end{cases}$$

Add it as a column in Boston data frame, while disposing of the original medv variable (via Boston$medv = NULL).

**My R code:**

```
> medv01 <- rep(0, length(Boston$medv))
> medv01[Boston$medv > median(Boston$medv)] <- 1
> Boston = data.frame(Boston, medv01)
> Boston$medv = NULL
```

(b) Decide on three values of K and three subsets of predictors (including full set of all 13 predictors). Totally your judgment call. When picking predictor subsets, you could use

- Your own logical considerations (which variables appear most important to predict the price),
- Correlation matrix (to determine if there are groups of correlated variables, and you could just retain one of them in the model, dropping the rest).

**I decided to use 1, 5, 7 for values of K.**

**The three subsets of predictors:**

**Subset #1: (age, tax, lstat)**

**Subset #2 (crim, black, lstat)**

**Subset #3: (All 13 predictors)**

(c) What stable method was introduced in the class in order to compare predictive quality of different models? Proceed to use this method & obtain test error estimates of all 3 × 3 = 9 models. Which model (the combination of K & predictor subset) won?

**The stable method we will be using is Leave-One-Out-Cross-Validation (LOOCV), which I will be using knn.cv function.**

**Subset #1:**

```
> library(class)
> set.seed(1)
> for(K in c(1, 5, 7))
+ {
+   set.seed(1)
+   knn.pred <- knn.cv(train = s1, cl = Boston$medv01, k = K)
+   print(table(knn.pred, Boston$medv01))
+   print(mean(knn.pred != Boston$medv01))
+ }
```

**K = 1**

```
knn.pred   0   1
       0 209  49
       1  47 201
[1] 0.1897233
```

**K = 5**

```
knn.pred   0   1
       0 199  46
       1  57 204
[1] 0.2035573
```

**K = 7**

```
knn.pred   0   1
       0 204  34
       1  52 216
[1] 0.1699605
```

**Subset #2:**

```
> set.seed(1)
> for(K in c(1, 5, 7))
+ {
+   set.seed(1)
+   knn.pred <- knn.cv(train = s2, cl = Boston$medv01, k = K)
+   print(table(knn.pred, Boston$medv01))
+   print(table(knn.pred, Boston$medv01))
+ }
```

**K = 1**

```
knn.pred   0   1
       0 204  58
       1  52 192
[1] 0.2173913
```

**K = 5**

```
knn.pred   0   1
       0 210  59
       1  46 191
[1] 0.2075099
```

**K = 7**

```
knn.pred   0   1
```

```
       0 224  59
       1  32 191
[1] 0.1798419
```

**Subset # 3:**

```
> set.seed(1)
> for(K in c(1, 5, 7))
+ {
+   set.seed(1)
+   knn.pred <- knn.cv(train = s3, cl = Boston$medv01, k = K)
+   print(table(knn.pred, Boston$medv01))
+   print(mean(knn.pred != Boston$medv01))
+ }
```

**K = 1**

```
knn.pred   0   1
       0 210  52
       1  46 198
[1] 0.1936759
```

**K = 5**

```
knn.pred   0   1
       0 209  42
       1  47 208
[1] 0.1758893
```

**K = 7**

```
knn.pred   0   1
       0 206  44
       1  50 206
[1] 0.1857708
```

**As a results, subset #1 won because it has the lowest test error, which is 0.1699605 when k = 7 . The predictors in subset #1 are age, tax, and lstat.**

(d) Are all the variables in Boston data set on the same scale? If not, how do we deal with it?

**No, they are not on the same scale. To deal with it is by using scale function.**

(e) Proceed to apply scaling to the predictors in Boston data and repeat part (c) for the standardized data set. Did the results (the test errors & the winning model) change?

**Subset #1:**

```
> s1 <- scale(s1)
> set.seed(1)
> for(K in c(1, 5, 7))
```

```
+ {
+   set.seed(1)
+   knn.pred <- knn.cv(train = s1, cl = Boston$medv01, k = K)
+   print(table(knn.pred, Boston$medv01))
+   print(mean(knn.pred != Boston$medv01))
+ }
```

**K = 1**

```
knn.pred   0   1
       0 203  52
       1  53 198
[1] 0.2075099
```

**K = 5**

```
knn.pred   0   1
       0 210  52
       1  46 198
[1] 0.1936759
```

**K = 7**

```
knn.pred   0   1
       0 210  45
       1  46 205
[1] 0.1798419
```

**Subset #2:**

```
> s2 <- scale(s2)
> set.seed(1)
> for(K in c(1, 5, 7))
+ {
+   set.seed(1)
+   knn.pred <- knn.cv(train = s2, cl = Boston$medv01, k = K)
+   print(table(knn.pred, Boston$medv01))
+   print(mean(knn.pred != Boston$medv01))
+ }
```

**K = 1**

```
knn.pred   0   1
       0 209  59
       1  47 191
[1] 0.2094862
```

**K = 5**

```
knn.pred   0   1
       0 220  57
       1  36 193
[1] 0.1837945
```

**K = 7**

```
knn.pred   0   1
       0 219  58
       1  37 192
```

```
[1] 0.187747
```

**Subset #3:**

```
> s3 <- scale(s3)
> set.seed(1)
> for(K in c(1, 5, 7))
+ {
+   set.seed(1)
+   knn.pred <- knn.cv(train = s2, cl = Boston$medv01, k = K)
+   print(table(knn.pred, Boston$medv01))
+   print(mean(knn.pred != Boston$medv01))
+ }
```

**K = 1**

```
knn.pred   0   1
       0 209  59
       1  47 191
[1] 0.2094862
```

**K = 5**

```
knn.pred   0   1
       0 220  57
       1  36 193
[1] 0.1837945
```

**K = 7**

```
knn.pred   0   1
       0 219  58
       1  37 192
[1] 0.187747
```

**Subset #1 is still the winner after scaling the data which is 0.1798419 when K = 7. The results are mixed, which some test errors improved and some got worsen.**

5. In this problem, you will use support vector approaches in order to predict whether a given car gets high or low gas mileage based on the *Auto* data set.

(a) Create a binary variable that takes on a 1 for cars with gas mileage above the median, and a 0 for cars with gas mileage below the median. Make sure to dispose of original *mpg* value in the end *(Auto$mpg = NULL).*

**My R code:**

```
> View(Auto)
> library(ISLR)
> data("Auto")
> Auto$mpg01 <- as.factor(ifelse(Auto$mpg > median(Auto$mpg), 1, 0))
> Auto$mpg <- NULL
```

(b) Run *set.seed*(1). Fit a support vector classifier to the data with various values of *cost* (similar to the lab), in order to predict whether a car gets high or low gas mileage. Report the cross-validation errors associated with different values of this parameter.

**My R code:**

```
> library(e1071)
> set.seed(1)
> svm.tune <- tune(svm, mpg01 ~ ., data = Auto, kernel = "linear",
+                  ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
> summary(svm.tune)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost
    1

- best performance: 0.09179487

- Detailed performance results:
   cost      error dispersion
1 1e-03 0.13288462 0.05551120
2 1e-02 0.09179487 0.05812971
3 1e-01 0.09692308 0.06369443
4 1e+00 0.09179487 0.04543280
5 5e+00 0.10198718 0.04338864
6 1e+01 0.11480769 0.05828005
7 1e+02 0.11730769 0.06521821
```

(c) Which cost value gave you the misclassification error? Print the confusion matrix for the optimal cost from part (c). Describe the two types of errors your model can make, and which of those two types does it commit more frequently?

**Cost 1 gave me the misclassification error.**

**My R code**:

```
> bestmod = svm.tune$best.model
> pred <- predict(bestmod)
> table(predict = pred, truth = Auto$mpg01)
       truth
predict   0   1
      0 191   6
      1   5 190
```

**The two types of errors from my model I have is 6 false positives and 5 false negatives. So, this model commit false positives more frequently than false negatives.**

(d) Plot the fitted boundary of the optimal support vector classifier for a couple of predictor pairings.

**Hint:** In the lab, we used the *plot*() function for svm objects only in cases with p = 2. When p > 2, you can use the *plot*() function to create plots displaying pairs of variables at a time. Essentially, instead of typing

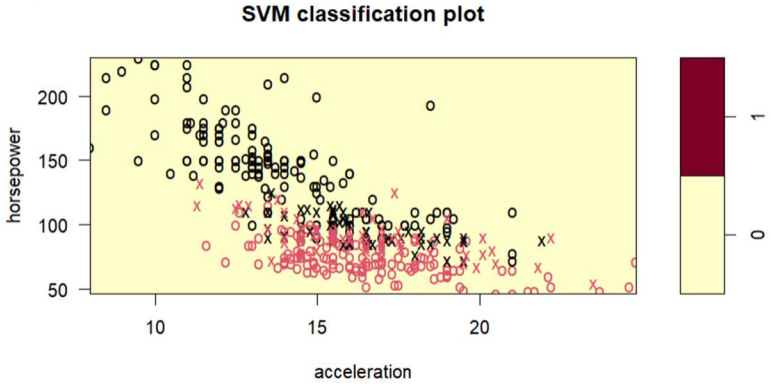
> plot(svmfit, dat)

where *svm fit* contains your fitted model and *dat* is a data frame containing your data, you can type.

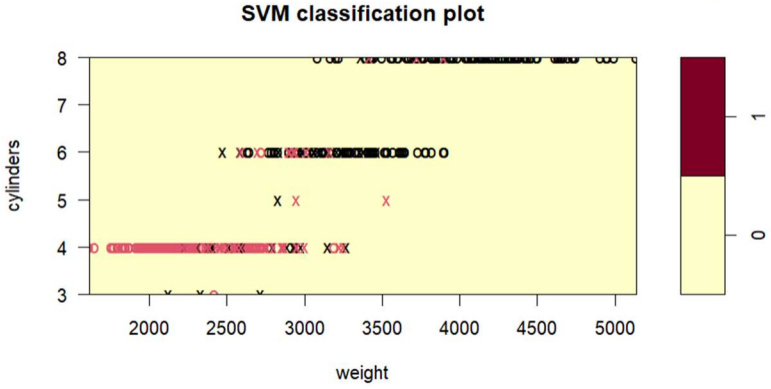
> plot(svmfit, dat, x1~x4 )

in order to plot just the first and fourth variables. However, you must replace *x*1 and *x*4 with the correct variable names. To find out more, type ?*plot.svm*.

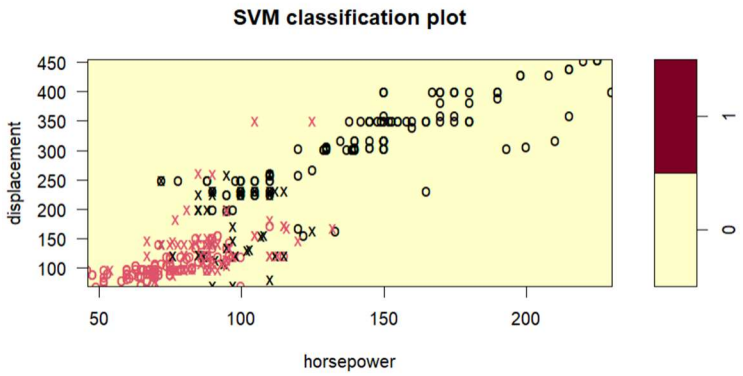
```
> plot(bestmod, data = Auto, horsepower ~ acceleration)
```




```
> plot(bestmod, data = Auto, cylinders ~ weight)
```




```
> plot(bestmod, data = Auto, displacement ~ horsepower)
```




6. This problem involves the *OJ* data set which is part of the *ISLR* package.

(a) Use *set.seed*(1). Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

**My R code:**

```
> set.seed(1)
> train <- sample(nrow(OJ), 800)
> trainOJ <- OJ[train, ]
> testOJ <- OJ[-train, ]
```

(b) Fit a support vector classifier to the training data using *cost* = 0.01, with *Purchase* as the response and the other variables as predictors. Use the *summary()* function to produce summary statistics, and describe the results obtained.

**My R code:**

```
> linearOJ <- svm(Purchase ~ ., data = trainOJ,
+                 kernel = "linear", cost = 0.01)
> summary(linearOJ)
Call:
svm(formula = Purchase ~ ., data = trainOJ, kernel = "linear", cost = 0.01)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.01

Number of Support Vectors:  432

 ( 215 217 )


Number of Classes:  2

Levels:
```

```
 CH  MM
```

**I trained 432 observations and split into two classes, which are CH and MM. There are 215 training observations in CH and 217 training observations in MM.**

(c) What are the training and test error rates?

```
> mean(predict(linearOJ, trainOJ) != trainOJ$Purchase)
[1] 0.16625
```

**The training error rate is 0.16625**

```
> mean(predict(linearOJ, testOJ) != testOJ$Purchase)
[1] 0.1814815
```

**The testing error rate is 0.1814815**

(d) Use the *tune()* function to select an optimal cost. Consider values in the range 0.01 to 10 (close to how we did in the lab).

**My R code:**

```
> svm.tune <- tune(svm, Purchase ~ ., data = trainOJ, kernel = "linear",
+                  ranges = list(cost = c(0.001, 0.01, 0.1, 1, 10)))
> summary(svm.tune)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost
  0.1

- best performance: 0.1625

- Detailed performance results:
   cost   error dispersion
1 1e-03 0.34250 0.04937104
2 1e-02 0.16625 0.05138701
3 1e-01 0.16250 0.04894725
4 1e+00 0.16875 0.04723243
5 1e+01 0.16500 0.04993051
```

**My best cost is 0.1**

(e) Compute the training and test error rates using this new value for *cost*.

**My R code:**

```
> bestmod = svm.tune$best.model
> table(predict(bestmod), truth = trainOJ$Purchase)
    truth
      CH  MM
  CH 438  71
  MM  56 235
```

```
> mean(predict(bestmod) != trainOJ$Purchase)
[1] 0.15875
> mean(predict(bestmod, testOJ) != testOJ$Purchase)
[1] 0.1888889
```

**The training error rate is 0.15875 for cost = 0.1**

**The testing error rate is 0.1888889 for cost = 0.1**