IS621_hw3

Charley Ferrari

March 7, 2016

Question 2

	0	1
0	119	30
1	5	27

Class and scored.class have levels 0 and 1. I'm going to assume 0 is negative, and 1 is positive. Scored.class is the predicted class, and is represented horizontally. Class is the actual class, and is represented vertically. This means I can label the cells of the table as seen below:

	0	1
0	True Negative	False Positive
1	False Negative	True Positive

Lets write a few helper functions to help us evaluate the success of this classification model.

Question 3

This function will take in a data frame, with the actual and predicted classification columns identified, and will return the accuracy of the predictions.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

```r
accuracy.calc <- function(classification, actual, predicted){
  confusion <- table(select(classification, get(actual), get(predicted)))
  accuracy <- (confusion[1,1] + confusion[2,2])/sum(confusion)
  return(accuracy)
}

accuracy.calc(classification, actual, predicted)
```

```
## [1] 0.8066298
```

Question 4

This function will take in a data frame, with the actual and predicted classification columns identified, and will return the classification error rate of the predictions.

$$ClassificationErrorRate = \frac{FP + FN}{TP + FP + TN + FN}$$

```
cer.calc <- function(classification, actual, predicted){
  confusion <- table(select(classification, get(actual), get(predicted)))
  cer <- (confusion[1,2] + confusion[2,1])/sum(confusion)
  return(cer)
}

cer.calc(classification, actual, predicted)
```

```
## [1] 0.1933702
```

Lets verify that the accuracy and the error rate sum to one:

```
actualcol <- "class"
predictedcol <- "scored.class"

accuracy.calc(classification, actualcol, predictedcol) +
  cer.calc(classification, actualcol, predictedcol)
```

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

It checks out!

Question 5

This function will take in a data frame, with the actual and predicted classification columns identified, and will return the precision of the predictions.

$$Precision = \frac{TP}{TP + FP}$$

```
precision.calc <- function(classification, actual, predicted){
  confusion <- table(select(classification, get(actual), get(predicted)))
  precision <- confusion[2,2]/(confusion[1,2] + confusion[2,2])
  return(precision)
}

precision.calc(classification, actual, predicted)
```

```
## [1] 0.84375
```

Question 6

This function will take in a data frame, with the actual and predicted classification columns identified, and will return the sensitivity of the predictions.

For sensitivity and specificity, I'm adding some logic that checks if all observations get classified as 1 or 0. This was necessary due to the ROC curve calculation, which tests a wide range of possible filters, including those that may result in predicted values being all positive or all negative.

$$Sensitivity = \frac{TP}{TP + FN}$$

```r
sensitivity.calc <- function(classification, actual, predicted){
  confusion <- table(select(classification, get(actual), get(predicted)))
  if(dim(confusion)[2] == 1){
    if(colnames(confusion) == 0){
      sensitivity <- 0
    } else{
      sensitivity <- 1
    }
  } else{
      sensitivity <- confusion[2,2]/(confusion[2,1] + confusion[2,2])
  }

  return(sensitivity)
}

sensitivity.calc(classification, actual, predicted)
```

```
## [1] 0.4736842
```

Question 7

This function will take in a data frame, with the actual and predicted classification columns identified, and will return the specificity of the predictions.

$$Specificity = \frac{TN}{TN + FP}$$

```r
specificity.calc <- function(classification, actual, predicted){
  confusion <- table(select(classification, get(actual), get(predicted)))
  if(dim(confusion)[2] == 1){
    if(colnames(confusion) == 0){
      specificity <- 1
    } else{
      specificity <- 0
    }
  } else{
    specificity <- confusion[1,1]/(confusion[1,1] + confusion[1,2])
  }
  return(specificity)
}

specificity.calc(classification, actual, predicted)
```

```
## [1] 0.9596774
```

Question 8

This function will take in a data frame, with the acutal and predicted classification columns identified, and will return the F1 score of the predictions (drawing on my previous functions)

$$F1Score = \frac{2 \times Precision \times Sensitivity}{Precision + Sensitivity}$$

```r
f1.calc <- function(classification, actual, predicted){
  precision <- precision.calc(classification, actual, predicted)
  sensitivity <- sensitivity.calc(classification, actual, predicted)
  f1 <- (2 * precision * sensitivity)/(precision + sensitivity)
  return(f1)
}

f1.calc(classification, actual, predicted)
```

```
## [1] 0.6067416
```

Question 9

The F1 score is indeed bound by 0 and 1. Both the Precision and the Sensitivity are bound by 0 and 1 because they are of the form $\frac{a}{a+b}$, with a and b guarenteed to be positive (and one of them can be 0.)

Rewriting the F1 score, with precision as p and sensitivity as s, we have:

$$\frac{ps + ps}{p + s}$$

Since both p and s are bound by 0 and 1, ps < p and ps < s. This means that we can be sure that ps + ps < p + s, which means that the F1 score is also bound by 0 and 1.

Question 10

Now to write a funciton that generates an ROC curve from a given data set, with true classification columns and probability columns identified. I'll return a list that includes the plot of the ROC curve and a vector that contains the calculated area under the curve (AUC.)

Write a funciton that generates an ROC curve from a data set with a true classification column (class in our example) and a probability column (scored.probability in our example.) Your function should return a list that includes the plot of the ROC curve and a vector that contains the calculated area under the curve (AUC.)

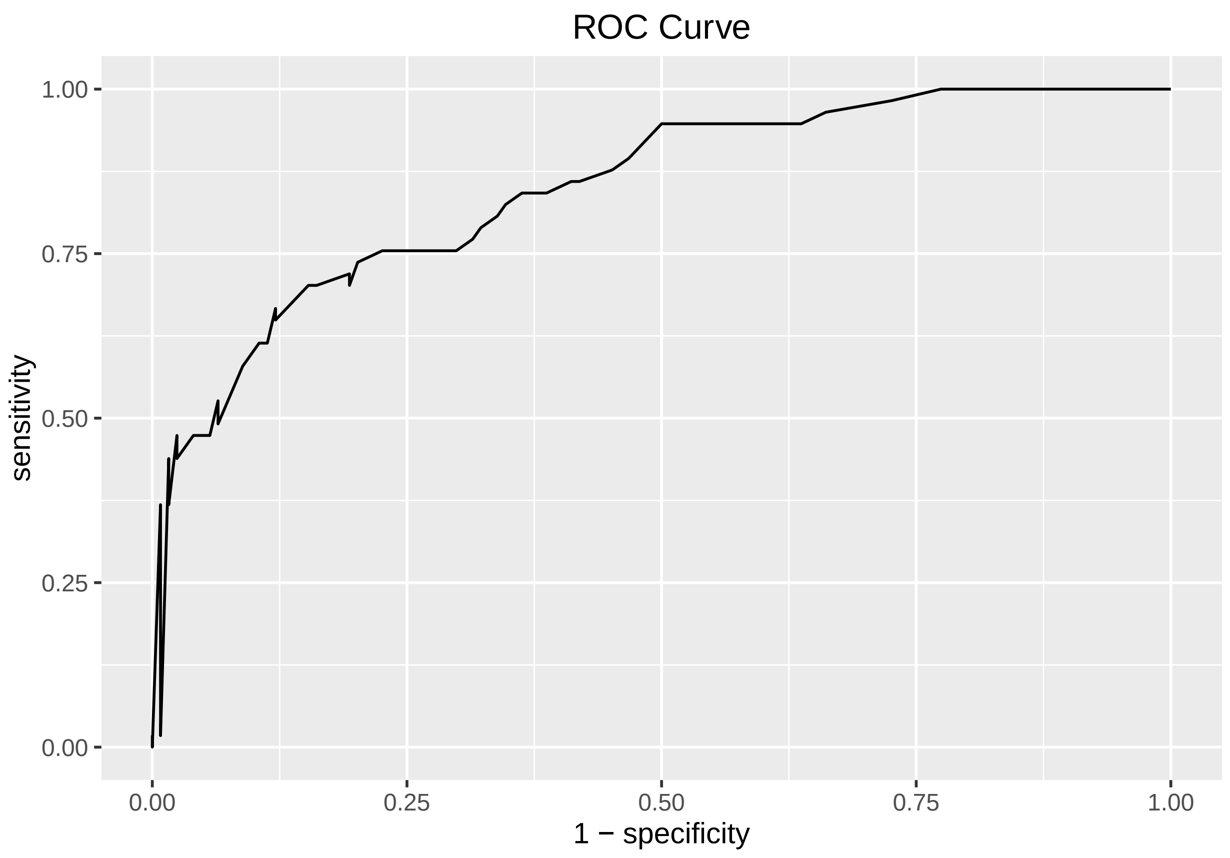
Before I start I'll note that the scored.class variable seems to be "tuned" based on a threshold of 0.5. If the scored.probability is greater than 0.5, the scored.class is recorded as 1. Otherwise, the scored.class is recorded as 0.

I'll build this function to create versions of scored.class based on 0.01 increments in this threshold.

To calculate the AUC, I will just subtract the 1-specificity value from each sensitivity value (to get the difference between the ROC curve and a straight y=x line at that point) and multiply each value by my step, 0.01.

```r
roc.gen <- function(classification, actual, probability){

  sensitivityvec <- c()
  specificityvec <- c()


  for(i in seq(0,1,by=0.01)){

    classification$new.scored.calc <-
```

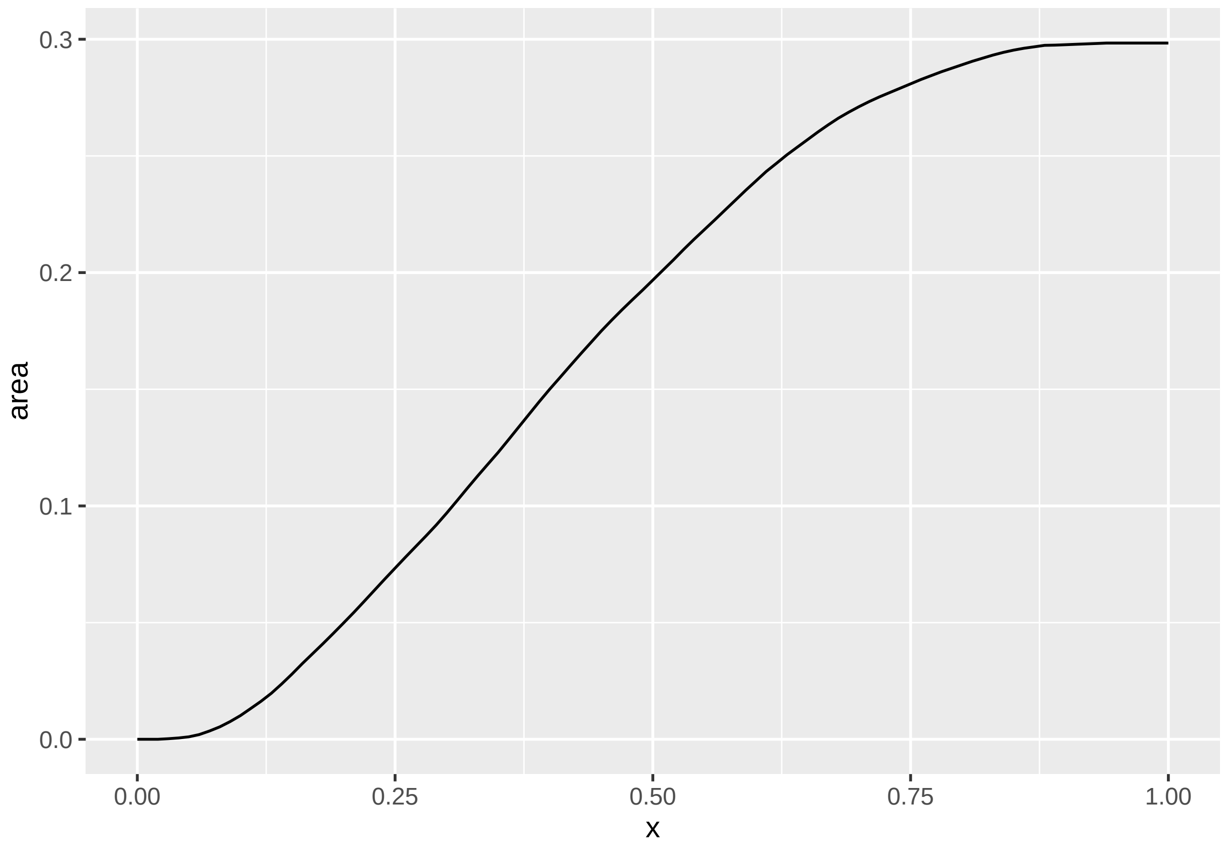
```r
      ifelse(classification$scored.probability < i, 0, 1)

    sensitivityvec <-
      c(sensitivityvec, sensitivity.calc(classification, "class", "new.scored.calc"))

    specificityvec <-
      c(specificityvec, specificity.calc(classification, "class", "new.scored.calc"))

    classification <- select(classification, -new.scored.calc)

  }

  rocdata <- data.frame(threshold = seq(0,1,by=0.01),
                        sensitivity = sensitivityvec,
                        specificity = specificityvec)

  auc <- cumsum(rocdata$sensitivity - (1-rocdata$specificity))*0.01

  g <- ggplot(rocdata, aes(x=1-specificity, y=sensitivity)) + geom_line() +
    ggtitle("ROC Curve")

  return(list(auc = auc, g = g))

}

roc.results <- roc.gen(classification, "class", "scored.probability")

roc.results$g
```



```
roc.results$auc
```

```
##   [1] 0.0000000000 0.0000000000 0.0000000000 0.0002419355 0.0005645161
##   [6] 0.0010483871 0.0020161290 0.0035483871 0.0053225806 0.0075806452
##  [11] 0.0101471420 0.0131833616 0.0162860781 0.0197113752 0.0237011885
##  [16] 0.0279329372 0.0324066214 0.0366765705 0.0409323713 0.0453353141
##  [21] 0.0498189021 0.0543689870 0.0590803622 0.0638723826 0.0686502547
##  [26] 0.0733333333 0.0780022637 0.0825764007 0.0871363894 0.0918576684
##  [31] 0.0969015280 0.1021873231 0.1075396152 0.1127971138 0.1178791737
##  [36] 0.1229612337 0.1283658744 0.1338511602 0.1393081494 0.1447651387
##  [41] 0.1500466893 0.1550580079 0.1601499717 0.1651471420 0.1700495190
##  [46] 0.1749518959 0.1795698925 0.1840124505 0.1882795699 0.1924518959
##  [51] 0.1967855122 0.2011191285 0.2054527448 0.2099476514 0.2142671194
##  [56] 0.2184111488 0.2225551783 0.2267798529 0.2310045274 0.2352292020
##  [61] 0.2392784380 0.2433276740 0.2468505942 0.2504541596 0.2537068478
##  [66] 0.2569595359 0.2602122241 0.2632894737 0.2661912847 0.2687422184
##  [71] 0.2711177136 0.2733177702 0.2753423882 0.2771915676 0.2790407470
##  [76] 0.2808899264 0.2827391058 0.2844128466 0.2860865874 0.2875848896
##  [81] 0.2890831919 0.2905814941 0.2919043577 0.2932272213 0.2943746463
##  [86] 0.2953466327 0.2961431805 0.2967642898 0.2973853990 0.2974801924
##  [91] 0.2976556310 0.2978310696 0.2980065082 0.2981819468 0.2983573854
##  [96] 0.2983573854 0.2983573854 0.2983573854 0.2983573854 0.2983573854
## [101] 0.2983573854
```

```r
ggplot(data.frame(x=seq(0,1,by=0.01), area=roc.results$auc), aes(x=x,y=area)) +
  geom_line()
```



Question 11

I displayed my results as I created each function.

Question 12

confusionMatrix, sensitivity, specificity, and lift in the caret package

```r
confusionMatrix(classification$scored.class, classification$class, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 119  30
##          1   5  27
##
##                Accuracy : 0.8066
##                  95% CI : (0.7415, 0.8615)
##     No Information Rate : 0.6851
##     P-Value [Acc > NIR] : 0.0001712
```

```
##
##                   Kappa : 0.4916
##  Mcnemar's Test P-Value : 4.976e-05
##
##             Sensitivity : 0.4737
##             Specificity : 0.9597
##          Pos Pred Value : 0.8438
##          Neg Pred Value : 0.7987
##              Prevalence : 0.3149
##          Detection Rate : 0.1492
##    Detection Prevalence : 0.1768
##       Balanced Accuracy : 0.7167
##
##        'Positive' Class : 1
##
```

```
sensitivity(factor(classification$scored.class),
            factor(classification$class), positive = "1")
```

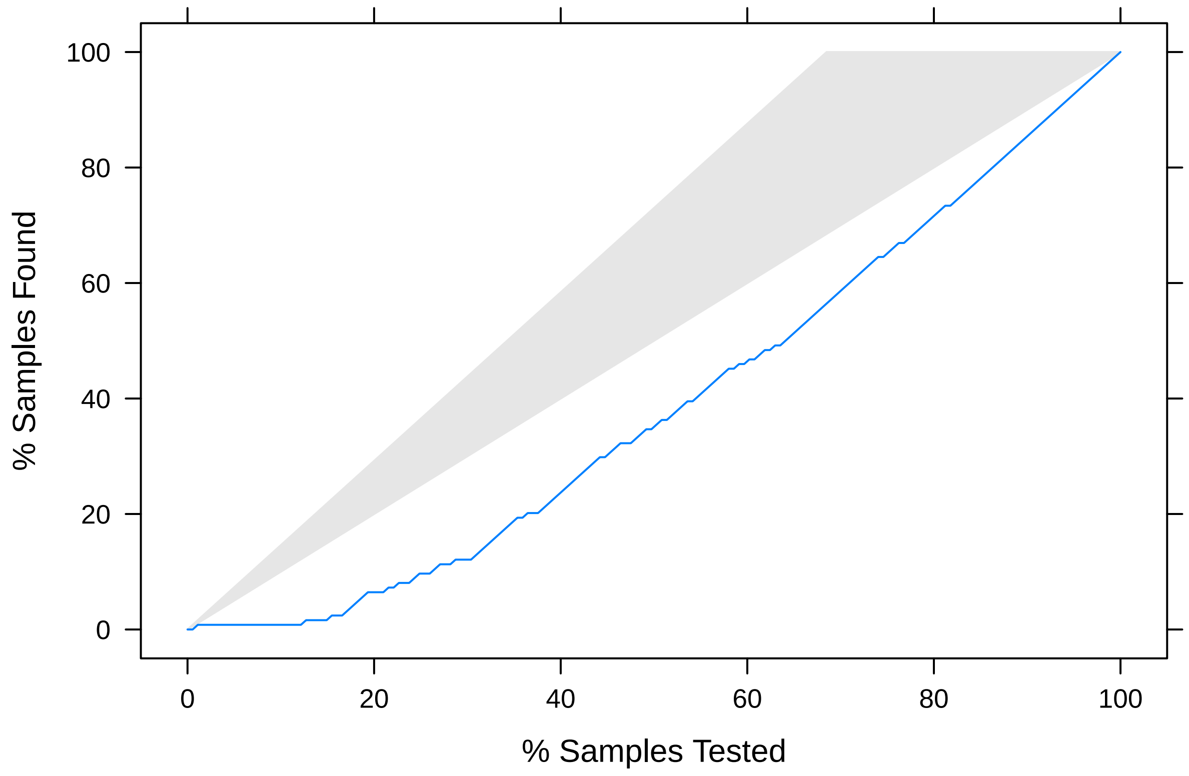
```
## [1] 0.4736842
```

```
specificity(factor(classification$scored.class),
            factor(classification$class), positive = "1")
```

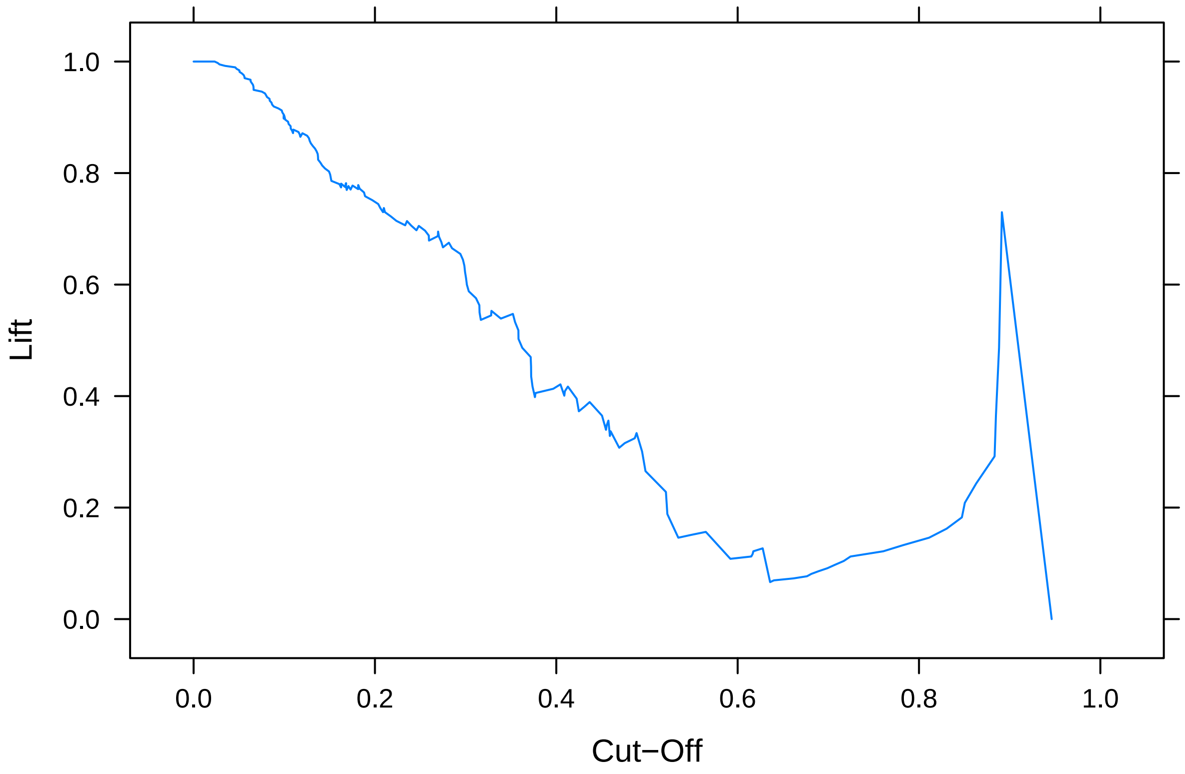
```
## [1] 0.4736842
```

These measures check out. I had to specify what the success was in this confusion matrix to make sure the measures matched. Sensitivity and Specificity are the same as what I calculated. lets check out the lift function:

```
lift0 <- lift(factor(class) ~ scored.probability, data=classification)

plot(lift0, plot = "gain")
```



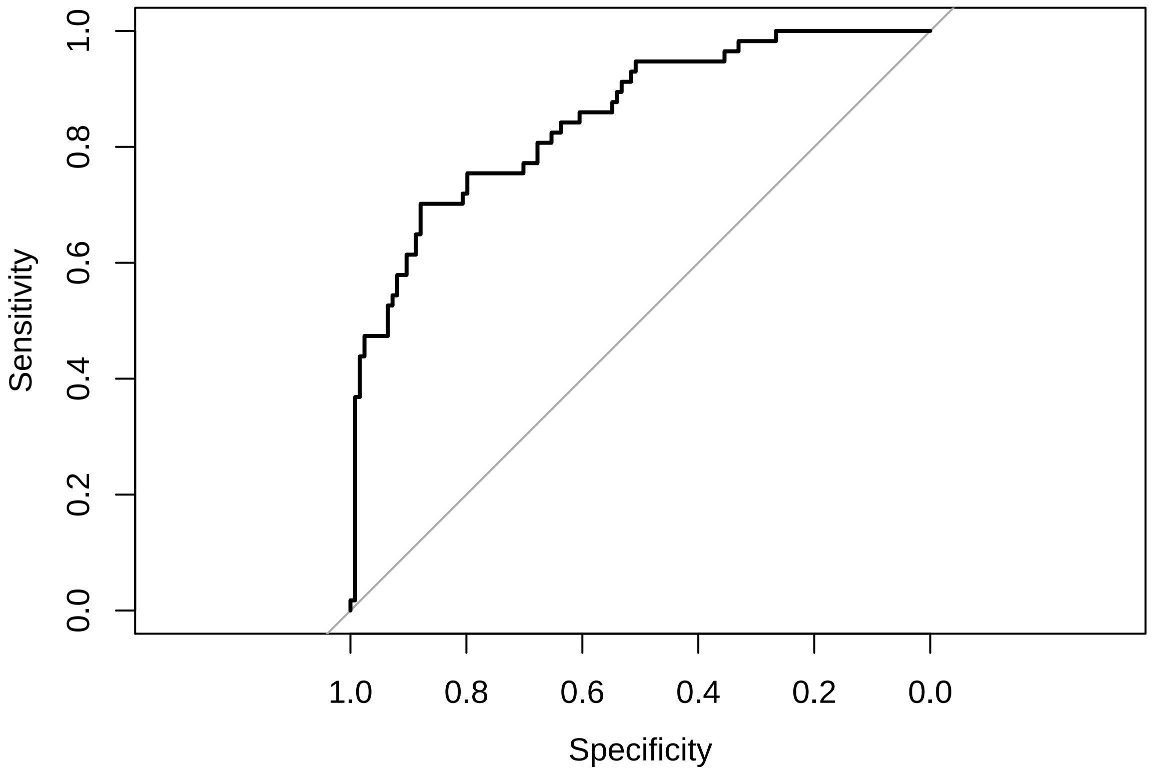
```r
plot(lift0, plot = "lift")
```



The gain seems to be related to the ROC curve, while the lift curve shows how much more successful we would be in picking certain percentages of the group and wind up with successes (intuitively it makes sense that as we pick higher proportions, we'll get closer to zero lift.) I'm not positive what could be causing the spike around 0.9. . .

Question 13

```
?pROC

?roc

roc0 <- roc(factor(class) ~ scored.probability, data=classification)

plot(roc0)
```



```
##
## Call:
## roc.formula(formula = factor(class) ~ scored.probability, data = classification)
##
## Data: scored.probability in 124 controls (factor(class) 0) < 57 cases (factor(class) 1).
## Area under the curve: 0.8503
```

It does look similar to my ROC plot. The shape seems to be influenced by the same factors, but the calculated ROC plot takes steps at each point where an observation passes through a filter, while my line chart just connects points with regular intervals. Having steps allows more precision in evaluating model behavior.