Logistic Regression: James Harden - Part 2

Jason Spector, Jericho Lawson, and Dr. Ekstrom

Purpose: Each activity illustrates a little sports analytics, a little statistics and a little *R*.

Sports Analytics: Improve the model that we found in part 1 - that used the number of points scored by James Harden to predict the result of the Rockets.

Statistics: logistic regression, predictive accuracy, sensitivity, specificity

R: Continue our development and reinforcement of R skills, including: reading a csv, manipulating data, adjusting thresholds of logistic modeling, and plotting.

Model from Part 1

Reading and Manipulating the Data

The data comes from 545 games played by James Harden and the Houston Rockets from October 31, 2012 to April 9, 2019.

```
harden = read.csv("harden_rockets.csv", header = TRUE)
```

Notice that the column (`X.1`) that gives the results of the games is a bit messy - it gives not only W or L, but the point differential. I am going to create a column (i.e. variable) called `Result` that contains just W's and L's. I start by creating this column with just L's.

```
harden$Result = rep("L", 545)
```

Next I use a `for` loop that changes the value in the `Result` column if the entry in the `X.1` column contains a W.

```
for (i in 1:545){
  if(grepl("W", harden$X.1[i], fixed = TRUE))
    harden$Result[i] = "W"
}
```

My next task is to create a column called `Dummy` that has a 1 for a Rocket win, and 0 for a loss. Again, I start with a column of all 0's.

```
harden$Dummy = rep(0, 545)
```

Now I use `for` loop that changes the value of `Dummy` if the entry in `Result` is a W.

```
for (i in 1:545){
  if((harden$Result[i] == "W"))
    harden$Dummy[i] = 1
}
```

We will also want to add color based on the variable Result. So the next will make sure that R reads Result as a categorical variable (with two categories, W and L).

```
harden$Result = as.factor(harden$Result)
```

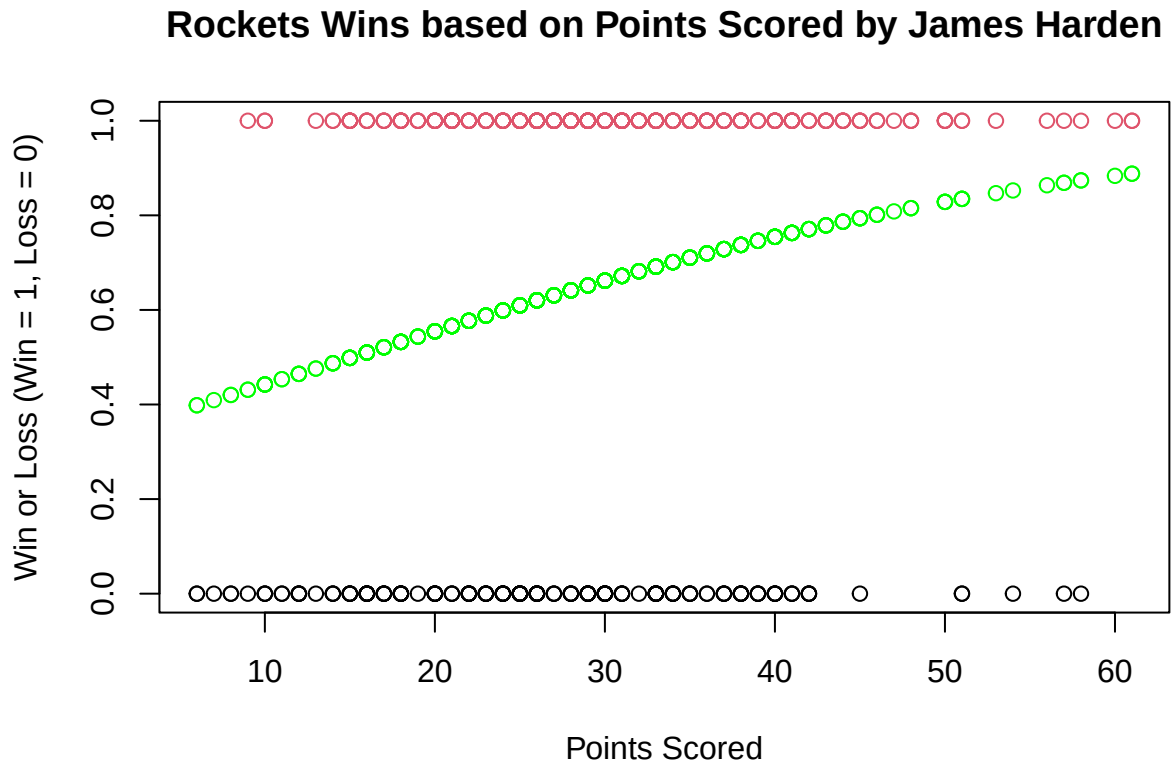
Creating the Model

We will now use logistic regression using the `glm` function. Then we use the `predict` function to determine the likelihood the Rockets get a win or loss based off each of Harden's performances.

```
harden_log = glm(Result~PTS, data = harden, family = "binomial")
harden_probs = predict(harden_log, type = "response")
```

We create a plot that combines the actual results of the Rockets vs the points scored by Harden, and the probabilities given by our model.

```
plot(harden$Dummy~harden$PTS, col = harden$Result,
     xlab = "Points Scored", ylab = "Win or Loss (Win = 1, Loss = 0)",
     main = "Rockets Wins based on Points Scored by James Harden")
points(harden$PTS,harden_probs,col="green")
```



Evaluating the Model

We can now compare our predictions from the model to the actual results.

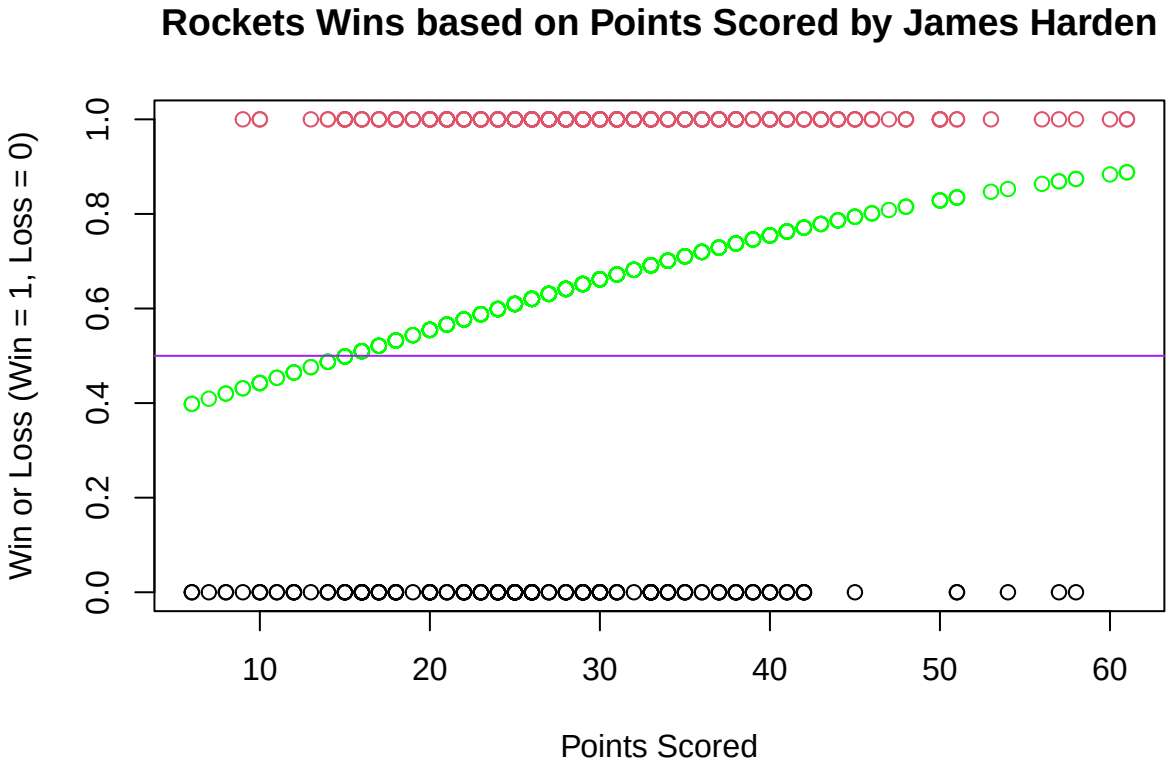
```
harden_log_pred = rep("L", 545)
harden_log_pred[harden_probs > .5] = "W"
harden_table = table(harden_log_pred, harden$Result)

harden_table
```

```
##
## harden_log_pred   L   W
##               L  24  13
##               W 169 339
```

How we read this table: the row L 24 13 says that when the model predicted a Loss, the Rockets lost 24 times and won 13 times, and the row W 169 339 says that when the model predicted a Win, the Rockets lost 169 times and won 339 times.

We can visualize this on our plot by adding a line at 0.5

```
plot(harden$Dummy~harden$PTS, col = harden$Result,
     xlab = "Points Scored", ylab = "Win or Loss (Win = 1, Loss = 0)",
     main = "Rockets Wins based on Points Scored by James Harden")
points(harden$PTS,harden_probs,col="green")
abline(a=0.5,b=0, col="purple")
```


Rockets Wins based on Points Scored by James Harden
Win or Loss (Win = 1, Loss = 0)
0.0
0.2
0.4
0.6
0.8
1.0
10
20
30
40
50
60
Points Scored

Prediciton Accuracy

To quantify how accurate our model is, there are certain calculations to compute using the table.

```
harden_table
```

```
##
## harden_log_pred   L   W
##               L  24  13
##               W 169 339
```

Of the 508 times the model predicted a W, it was correct 339 times. Of the 37 times the model predicted a L, it was correct 24 times. Overall, the model correctly predicted the result 363 out of 545 times.

```
# gives the total number of times the predicted result matched the actual result
sum(harden_log_pred == harden$Result)
```

```
## [1] 363
```

```
# gives the percentage of the time the predicted result matched the actual result.
mean(harden_log_pred == harden$Result)
```

```
## [1] 0.666055
```

The model's prediction accuracy is 0.666055.

Sensitivity

Here we focus on the model behavior when the team actually won. The Rockets won 352 games. The model correctly predicted 339 of those wins. So the model's sensitivity is 339/352

```
# calculates the entry in the 2nd row and 2nd column of the table
#   divided by the sum of the entries in the 2nd column
(sens_1 = harden_table[2,2] / sum(harden_table[,2]))
```

```
## [1] 0.9630682
```

Specificity

Here we focus on the model behavior when the team actually lost. The Rockets lost 193 games. The model correctly predicted 24 of those. So the model's specificity is 24/193.

```
(spec_1 = harden_table[1,1] / sum(harden_table[,1]))
```

```
## [1] 0.1243523
```

Putting It Together

So the model's sensitivity is 0.9630682 (which is excellent), it's specificity is 0.1243523 (which is terrible), and it's overall prediciton accuracy is 0.666055. Now the overall prediciton accuracy seems pretty good. But it looks much better than it is. It is worth noting that if your model ALWAYS predicted a win, it would have been correct 352 out of 545 times, giving a prediciton accuracy of 0.645872. So our model is not much better than just always predicting a win.

Adjusting Our Model

One thing we can do to is to balance out the predicted wins and predicted losses by adjusting the threshold between a predicted win and loss. In the original model, if the probability given by the logistic model is greater than 0.5, then we are predicting a win. What if we made that threshold 0.6, or 0.55?

```
harden_log_predM = rep("L", 545)
harden_log_predM[harden_probs > .55] = "W"
harden_tableM = table(harden_log_predM, harden$Result)
harden_tableM
```

```
##
## harden_log_predM   L   W
##                L  47  33
##                W 146 319
```

```
mean(harden_log_predM == harden$Result)
```

```
## [1] 0.6715596
```

```
(sens_M = harden_tableM[2,2] / sum(harden_tableM[,2]))
```

```
## [1] 0.90625
```

```
(spec_M = harden_tableM[1,1] / sum(harden_tableM[,1]))
```

```
## [1] 0.2435233
```

Of course, we can use R to automate this investigation.

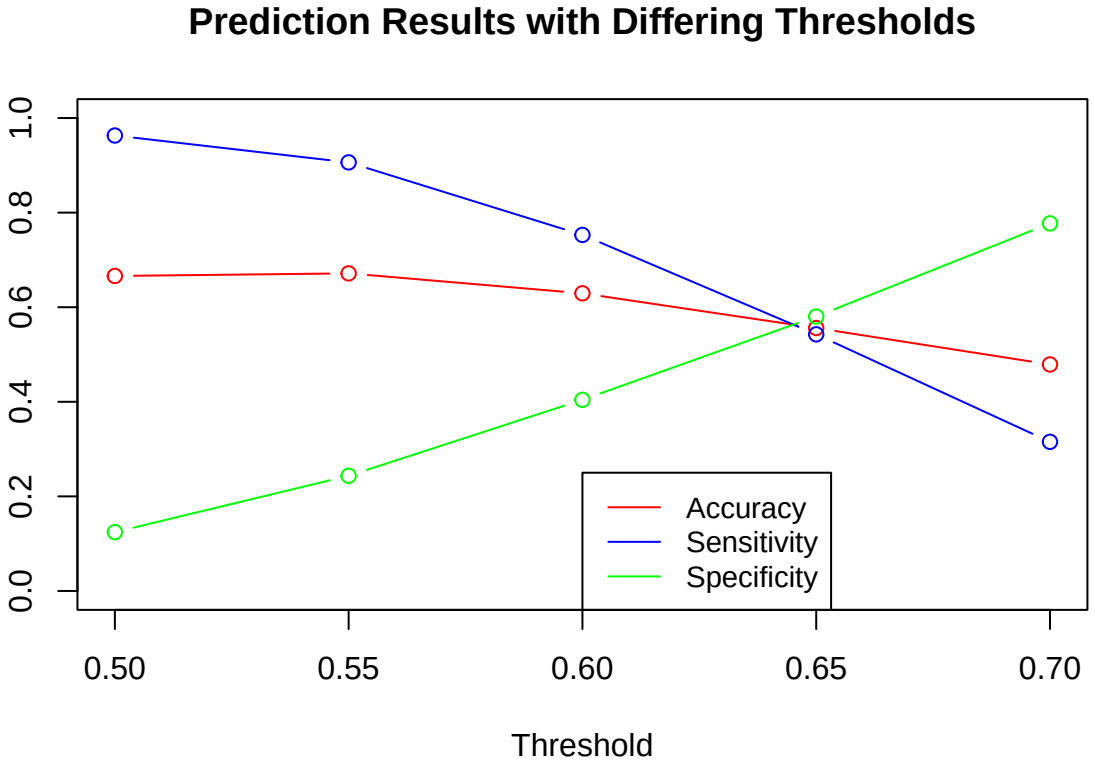
```
# this vector gives the thresholds to be investigated
potential_probs = c(.5, .55, .6, .65, .7)

accuracies = rep(-1, 5)
sensitivities = rep(-1, 5)
specificities = rep(-1, 5)


for (i in 1:5) {
  harden_log_predM = rep("L", 545)
  harden_log_predM[harden_probs > potential_probs[i]] = "W"
  accuracies[i] = mean(harden_log_predM == harden$Result)
  harden_tableM = table(harden_log_predM, harden$Result)

  if (dim(harden_tableM)[1] == 1 & rownames(harden_tableM)[1] == "L"){
    sensitivities[i] = 0 / sum(harden_tableM[,2])
    specificities[i] = harden_tableM[1,1] / sum(harden_tableM[,1])
  } else if (dim(harden_tableM)[1] == 1 & rownames(harden_tableM)[1] == "W"){
```

```
    sensitivities[i] = harden_tableM[1,2] / sum(harden_tableM[,2])
    specificities[i] = 0 / sum(harden_tableM[,1])
  } else {
    sensitivities[i] = harden_tableM[2,2] / sum(harden_tableM[,2])
    specificities[i] = harden_tableM[1,1] / sum(harden_tableM[,1])
  }

}

results_table = rbind(accuracies, sensitivities, specificities)
colnames(results_table) = potential_probs
results_table
```

```
##                     0.5      0.55       0.6      0.65       0.7
## accuracies    0.6660550 0.6715596 0.6293578 0.5559633 0.4788991
## sensitivities 0.9630682 0.9062500 0.7528409 0.5426136 0.3153409
## specificities 0.1243523 0.2435233 0.4041451 0.5803109 0.7772021
```

We can create a plot that visualizes the table above.

```
plot(potential_probs, accuracies, type = "b", col = "red", xlab = "Threshold", ylab = "",
     main = "Prediction Results with Differing Thresholds", ylim = c(0, 1))
lines(potential_probs, sensitivities, type = "b", col = "blue")
lines(potential_probs, specificities, type = "b", col = "green")
legend(0.6, 0.25, legend=c("Accuracy", "Sensitivity", "Specificity"),
       col=c("red", "blue", "green"), lty=1, cex=0.9)
```



Group Activity

Continuation of the activity from Monday.

1. Choose a sport you wish to investigate.
2. Determine variables you wish to perform a logistic regression and analysis on. Remember that the response variable must be binary.
3. Get the appropriate data by downloading it. (You may want to revisit your decisions for steps 1 and 2 if this proves difficult...)
4. Perform the logistic regression.
5. Use your resulting model to predict the results from your data as we did above. Create a table that compares predicted results to actual results Create a plot that includes points for each actual result, points for each predicted result, and a line at the cutoff (presumed to be 0.5).
6. Compute the prediction accuracy, sensitivity, specificity of the model you created.

Turn in your resulting RMD file and your CSV data file into the appropriate Gradescope Class Activity assignment. It is due at the end of class.

For the R-Assignment that is due this Friday: In addition to the steps above:

7. Determine a threshold that will maximize the prediction accuracy of your model. Be sure to include tables and graphs that support your answer.

Turn in your resulting RMD file and your CSV data file into the Gradescope: R Assignment 2 - Logistic Regression. It is due on Friday at 11:59.