Fourth Down

Sam Hughes

1/8/2022

With this code, I created models for three occurrences vital in a fourth down calculator when a team chooses to go for it: conversion percentage, yards gained on a successful conversion attempt, and yards gained (or lost) on a failed conversion attempt. The purpose of this file is to show my findings along with cool data visualizations, so I will mute the cells unless they express important information about a model or a plot.

Objective 1: Use the data to make a logistic model to predict the conversion percentage when given a down and distance. Modeling conversion percentage will be difficult because the samples are relatively small when considering the amount of combinations between yards for first down and yards away from a touchdown. Therefore, I will make a few different models and choose the one that seems to perform the best via cross validation.

```r
# First simpler model
gfi_model = glm(conversion ~ ydstogo + yardline, data = gfi_train, family = "binomial")
summary(gfi_model)
```

```
##
## Call:
## glm(formula = conversion ~ ydstogo + yardline, family = "binomial",
##     data = gfi_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5528  -1.1903  -0.2801   1.0609   2.5560
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.363148   0.061542   5.901 3.62e-09 ***
## ydstogo     -0.145569   0.008220 -17.709  < 2e-16 ***
## yardline     0.007182   0.001463   4.908 9.20e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6401.8  on 4617  degrees of freedom
## Residual deviance: 6007.2  on 4615  degrees of freedom
## AIC: 6013.2
##
## Number of Fisher Scoring iterations: 3
```

```r
gfi_model_intercept = gfi_model$coefficients[1]
gfi_model_coef_a = gfi_model$coefficients[2]
gfi_model_coef_b = gfi_model$coefficients[3]

ydstogo_and_yardline = function(ydstogo, yardline) {
  exp(gfi_model_intercept + gfi_model_coef_a * ydstogo + gfi_model_coef_b * yardline)/(1 + exp(gfi_mode
}
ydstogo_and_yardline(1, 15)
```

```
## (Intercept)
##   0.5806184
```

```r
# Score model
gfi_test$prediction1 = ydstogo_and_yardline(gfi_test$ydstogo, gfi_test$yardline)
gfi_test$binary1 = 0
gfi_test[which(gfi_test$prediction1 >= 0.5), "binary1"] = 1
gfi_test$binary1 = as.factor(gfi_test$binary1)
gfi_test$conversion = as.factor(gfi_test$conversion)
str(gfi_test)
```

```
## tibble [1,154 x 6] (S3: tbl_df/tbl/data.frame)
##  $ yardline    : num [1:1154] 69 52 86 43 38 45 31 39 6 31 ...
##  $ ydstogo     : num [1:1154] 10 15 9 10 1 2 11 10 1 11 ...
##  $ yards_gained: num [1:1154] 9 0 0 30 2 3 31 8 0 -8 ...
##  $ conversion  : Factor w/ 2 levels "0","1": 1 1 1 2 2 2 2 1 1 1 ...
##  $ prediction1 : num [1:1154] 0.355 0.19 0.418 0.314 0.62 ...
##  $ binary1     : Factor w/ 2 levels "0","1": 1 1 1 1 2 2 1 1 2 1 ...
```

```r
sum(gfi_test$conversion == gfi_test$binary1)/nrow(gfi_test)
```

```
## [1] 0.6247834
```

```r
confusionMatrix(gfi_test$binary1, gfi_test$conversion)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 271 145
##          1 288 450
##
##                Accuracy : 0.6248
##                  95% CI : (0.5961, 0.6528)
##     No Information Rate : 0.5156
##     P-Value [Acc > NIR] : 5.094e-14
##
##                   Kappa : 0.243
##
##  Mcnemar's Test P-Value : 8.849e-12
##
##             Sensitivity : 0.4848
```

```
##              Specificity : 0.7563
##           Pos Pred Value : 0.6514
##           Neg Pred Value : 0.6098
##               Prevalence : 0.4844
##           Detection Rate : 0.2348
##     Detection Prevalence : 0.3605
##        Balanced Accuracy : 0.6205
##
##         'Positive' Class : 0
##
```

```
# second model with squared and cubed terms for ysdtogo
gfi_model_2 = glm(conversion ~ ydstogo + ydstogo_squared + ydstogo_cubed + yardline, data = gfi_train, f
summary(gfi_model_2)
```

```
##
## Call:
## glm(formula = conversion ~ ydstogo + ydstogo_squared + ydstogo_cubed +
##     yardline, family = "binomial", data = gfi_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6222  -1.1170  -0.2285   1.0365   2.5298
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      0.6566396  0.0854179   7.687 1.50e-14 ***
## ydstogo         -0.3445208  0.0440933  -7.813 5.56e-15 ***
## ydstogo_squared  0.0216148  0.0055968   3.862 0.000112 ***
## ydstogo_cubed   -0.0005438  0.0001847  -2.945 0.003234 **
## yardline         0.0076158  0.0014713   5.176 2.27e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6401.8  on 4617  degrees of freedom
## Residual deviance: 5978.3  on 4613  degrees of freedom
## AIC: 5988.3
##
## Number of Fisher Scoring iterations: 6
```

```
gfi_model_intercept2 = gfi_model_2$coefficients[1]
gfi_model_coef_a2 = gfi_model_2$coefficients[2]
gfi_model_coef_b2 = gfi_model_2$coefficients[3]
gfi_model_coef_c2 = gfi_model_2$coefficients[4]
gfi_model_coef_d2 = gfi_model_2$coefficients[5]

ydstogo_and_yardline_2 = function(ydstogo, yardline) {
  exp(gfi_model_intercept2 + gfi_model_coef_a2 * ydstogo + gfi_model_coef_b2 * ydstogo^2 + gfi_model_co
}

ydstogo_and_yardline_2(1, 15)
```

```
## (Intercept)
##   0.6100271
```

```r
# Score model
gfi_test$prediction2 = ydstogo_and_yardline_2(gfi_test$ydstogo, gfi_test$yardline)
gfi_test$binary2 = 0
gfi_test[which(gfi_test$prediction2 >= 0.5), "binary2"] = 1
gfi_test$binary2 = as.factor(gfi_test$binary2)
gfi_test$conversion = as.factor(gfi_test$conversion)
str(gfi_test)
```

```
## tibble [1,154 x 8] (S3: tbl_df/tbl/data.frame)
##  $ yardline    : num [1:1154] 69 52 86 43 38 45 31 39 6 31 ...
##  $ ydstogo     : num [1:1154] 10 15 9 10 1 2 11 10 1 11 ...
##  $ yards_gained: num [1:1154] 9 0 0 30 2 3 31 8 0 -8 ...
##  $ conversion  : Factor w/ 2 levels "0","1": 1 1 1 2 2 2 2 1 1 1 ...
##  $ prediction1 : num [1:1154] 0.355 0.19 0.418 0.314 0.62 ...
##  $ binary1     : Factor w/ 2 levels "0","1": 1 1 1 1 2 2 1 1 2 1 ...
##  $ prediction2 : num [1:1154] 0.344 0.252 0.393 0.301 0.651 ...
##  $ binary2     : Factor w/ 2 levels "0","1": 1 1 1 1 2 2 1 1 2 1 ...
```

```r
sum(gfi_test$conversion == gfi_test$binary2)/nrow(gfi_test)
```

```
## [1] 0.6334489
```

```r
confusionMatrix(gfi_test$binary2, gfi_test$conversion)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 304 168
##          1 255 427
##
##                Accuracy : 0.6334
##                  95% CI : (0.6049, 0.6613)
##     No Information Rate : 0.5156
##     P-Value [Acc > NIR] : 4.560e-16
##
##                   Kappa : 0.2627
##
##  Mcnemar's Test P-Value : 2.896e-05
##
##             Sensitivity : 0.5438
##             Specificity : 0.7176
##          Pos Pred Value : 0.6441
##          Neg Pred Value : 0.6261
##              Prevalence : 0.4844
##          Detection Rate : 0.2634
##    Detection Prevalence : 0.4090
##       Balanced Accuracy : 0.6307
##
```

```
##        'Positive' Class : 0
##
```

```
gfi_model_3 = glm(conversion ~ ydstogo_ln + yardline_ln, data = gfi_train, family = "binomial")
summary(gfi_model_3)
```

```
##
## Call:
## glm(formula = conversion ~ ydstogo_ln + yardline_ln, family = "binomial",
##     data = gfi_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5809  -1.0802  -0.6485   1.0629   1.8549
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.15348    0.09272   1.655   0.0979 .
## ydstogo_ln  -0.68594    0.03474 -19.747  < 2e-16 ***
## yardline_ln  0.16939    0.02873   5.896 3.72e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6401.8  on 4617  degrees of freedom
## Residual deviance: 5975.3  on 4615  degrees of freedom
## AIC: 5981.3
##
## Number of Fisher Scoring iterations: 4
```

```
ydstogo_and_yardline_3 = function(ydstogo, yardline) {
  exp(gfi_model_3$coefficients[1] + gfi_model_3$coefficients[2] * log(ydstogo) + gfi_model_3$coefficien
}
ydstogo_and_yardline_3(1, 15)
```

```
## (Intercept)
##   0.6484413
```

```
# Score model
gfi_test$prediction3 = ydstogo_and_yardline_3(gfi_test$ydstogo, gfi_test$yardline)
gfi_test$binary3 = 0
gfi_test[which(gfi_test$prediction3 >= 0.5), "binary3"] = 1
gfi_test$binary3 = as.factor(gfi_test$binary3)
gfi_test$conversion = as.factor(gfi_test$conversion)
str(gfi_test)
```

```
## tibble [1,154 x 10] (S3: tbl_df/tbl/data.frame)
##  $ yardline    : num [1:1154] 69 52 86 43 38 45 31 39 6 31 ...
##  $ ydstogo     : num [1:1154] 10 15 9 10 1 2 11 10 1 11 ...
##  $ yards_gained: num [1:1154] 9 0 0 30 2 3 31 8 0 -8 ...
##  $ conversion  : Factor w/ 2 levels "0","1": 1 1 1 2 2 2 2 1 1 1 ...
```

```
##  $ prediction1 : num [1:1154] 0.355 0.19 0.418 0.314 0.62 ...
##  $ binary1     : Factor w/ 2 levels "0","1": 1 1 1 1 2 2 1 1 2 1 ...
##  $ prediction2 : num [1:1154] 0.344 0.252 0.393 0.301 0.651 ...
##  $ binary2     : Factor w/ 2 levels "0","1": 1 1 1 1 2 2 1 1 2 1 ...
##  $ prediction3 : num [1:1154] 0.33 0.262 0.355 0.312 0.683 ...
##  $ binary3     : Factor w/ 2 levels "0","1": 1 1 1 1 2 2 1 1 2 1 ...
```

```
sum(gfi_test$conversion == gfi_test$binary3)/nrow(gfi_test)
```

```
## [1] 0.6360485
```

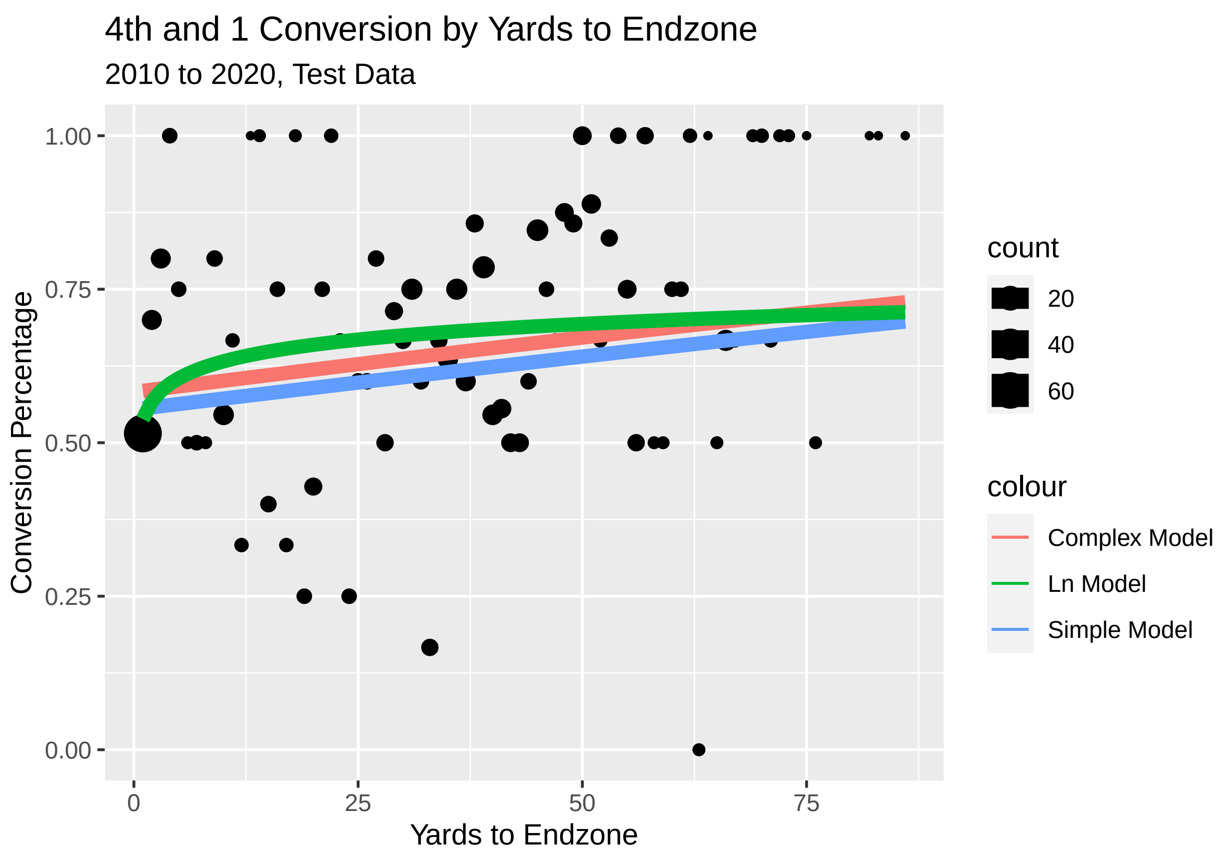
```
confusionMatrix(gfi_test$binary3, gfi_test$conversion)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 332 193
##          1 227 402
##
##                Accuracy : 0.636
##                  95% CI : (0.6075, 0.6639)
##     No Information Rate : 0.5156
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.27
##
##  Mcnemar's Test P-Value : 0.1073
##
##             Sensitivity : 0.5939
##             Specificity : 0.6756
##          Pos Pred Value : 0.6324
##          Neg Pred Value : 0.6391
##              Prevalence : 0.4844
##          Detection Rate : 0.2877
##    Detection Prevalence : 0.4549
##       Balanced Accuracy : 0.6348
##
##        'Positive' Class : 0
##
```

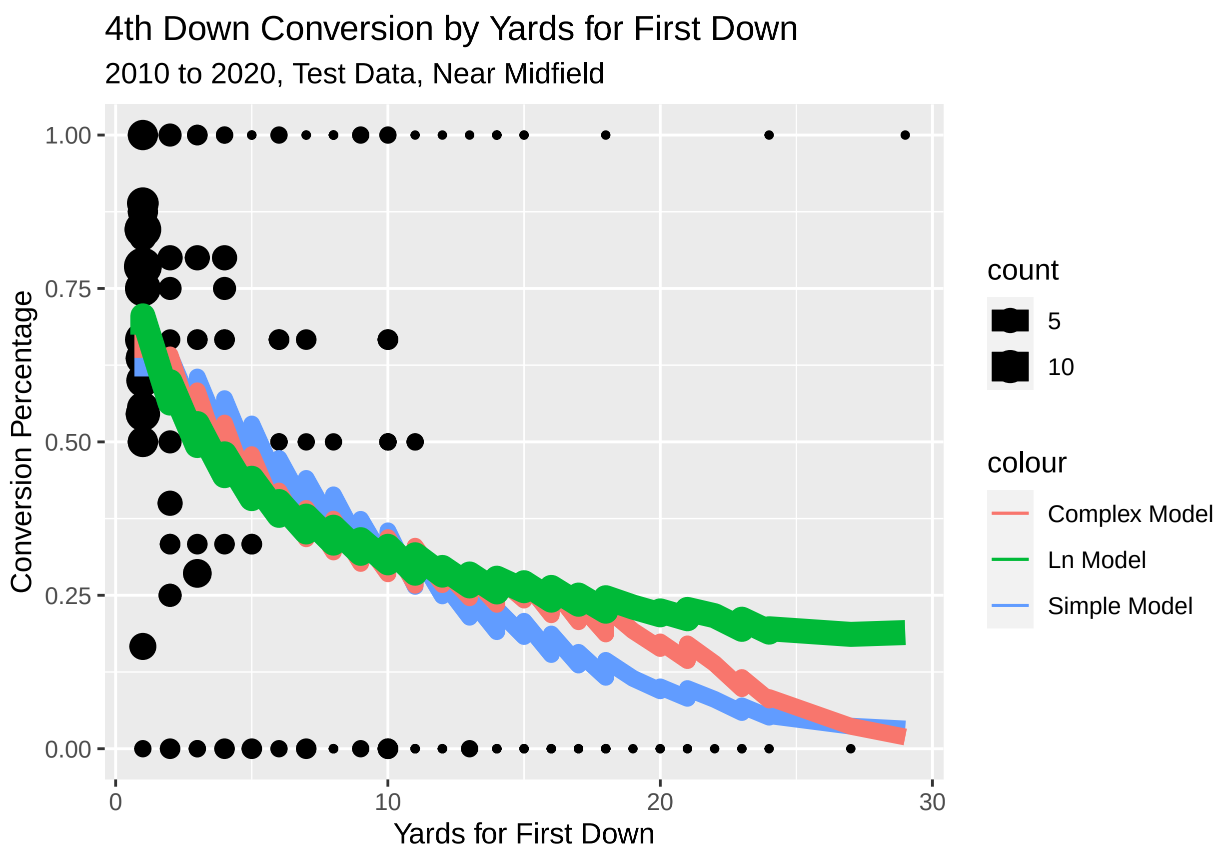
The confusion matrices give us information against the testing data, and we can also visualize this data via ggplot.

```
ggplot(data = subset(gfi_test_agg, ydstogo == 1),
       mapping = aes(x = yardline,
                     y = mean))+
  ggtitle("4th and 1 Conversion by Yards to Endzone", subtitle = "2010 to 2020, Test Data")+
  xlab("Yards to Endzone")+
  ylab("Conversion Percentage")+
  geom_point(aes(size = count))+
  geom_line(aes(y = ydstogo_and_yardline(ydstogo, yardline),
```

```r
                 size = 7, colour = "Simple Model"))+
  geom_line(aes(y = ydstogo_and_yardline_2(ydstogo, yardline),
                 size = 7, colour = "Complex Model"))+
  geom_line(aes(y = ydstogo_and_yardline_3(ydstogo, yardline),
                 size = 7, colour = "Ln Model"))
```




```r
ggplot(data = subset(gfi_test_agg, yardline >= 30 & yardline <= 70),
       mapping = aes(x = ydstogo,
                     y = mean))+
  ggtitle("4th Down Conversion by Yards for First Down",
          subtitle = "2010 to 2020, Test Data, Near Midfield")+
  xlab("Yards for First Down")+
  ylab("Conversion Percentage")+
  geom_point(aes(size = count))+
  geom_line(aes(y = ydstogo_and_yardline(ydstogo, yardline),
                 size = 3, colour = "Simple Model"))+
  geom_line(aes(y = ydstogo_and_yardline_2(ydstogo, yardline),
                 size = 3, colour = "Complex Model"))+
  geom_line(aes(y = ydstogo_and_yardline_3(ydstogo, yardline),
                 size = 7, colour = "Ln Model"))
```



I think the logarithmic model is the best choice because of its performance at short yardage (when going for it is a realistic possibility). Fourth and 1 mean percentage –> 0.65, 2 –> 0.56, 3 –> 0.49, all of which the logarithmic model is closest to predicting

Objective 2: When a conversion does happen, find out the excess yards gained on the play (the yards gained beyond the first down marker). Let's start with a simple model of just the distance for a first down and distance for a touchdown.

```
# while converting
gfi_success = subset(df_fourth_go_simple, conversion == 1)
gfi_success$excess = gfi_success$yards_gained - gfi_success$ydstogo

# Split data
set.seed(123)
training_samples_gfi = createDataPartition(gfi_success$excess, p = 0.8, list = FALSE)
gfi_train = gfi_success[training_samples_gfi, ]
gfi_test = gfi_success[-training_samples_gfi, ]

# for graph later
gfi_test_agg = gfi_test %>%
  group_by(ydstogo, yardline) %>%
  summarise("mean" = mean(excess), "count" = length(excess))
```

```
## 'summarise()' has grouped output by 'ydstogo'. You can override using the '.groups' argument.
```

```r
# for more complex models later
gfi_train$ydstogo_squared = gfi_train$ydstogo^2
gfi_train$ydstogo_cubed = gfi_train$ydstogo^3

gfi_train$yardline_squared = gfi_train$yardline^2
gfi_train$yardline_cubed = gfi_train$yardline^3
```

```r
# simple model
summary(lm(excess ~ ydstogo * yardline, data = gfi_train))
```

```
##
## Call:
## lm(formula = excess ~ ydstogo * yardline, data = gfi_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.251  -4.871  -2.329   1.654  56.503
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.654237   0.470890   3.513 0.000452 ***
## ydstogo           0.424376   0.118992   3.566 0.000369 ***
## yardline          0.098859   0.011158   8.860  < 2e-16 ***
## ydstogo:yardline -0.001608   0.002242  -0.717 0.473190
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.58 on 2313 degrees of freedom
## Multiple R-squared:  0.08653,    Adjusted R-squared:  0.08535
## F-statistic: 73.04 on 3 and 2313 DF,  p-value: < 2.2e-16
```

```r
anova(lm(excess ~ ydstogo * yardline, data = gfi_train))
```

```
## Analysis of Variance Table
##
## Response: excess
##                    Df Sum Sq Mean Sq  F value Pr(>F)
## ydstogo             1   7238  7238.3  98.3166 <2e-16 ***
## yardline            1   8855  8855.0 120.2758 <2e-16 ***
## ydstogo:yardline    1     38    37.9   0.5147 0.4732
## Residuals        2313 170289    73.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# interaction term is insignificant
summary(lm(excess ~ ydstogo + yardline, data = gfi_train))
```

```
##
## Call:
## lm(formula = excess ~ ydstogo + yardline, data = gfi_train)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.849  -4.873  -2.342   1.722  56.597
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.870536   0.361676   5.172 2.52e-07 ***
## ydstogo     0.347415   0.051483   6.748 1.89e-11 ***
## yardline    0.093711   0.008544  10.968  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.579 on 2314 degrees of freedom
## Multiple R-squared:  0.08633,    Adjusted R-squared:  0.08554
## F-statistic: 109.3 on 2 and 2314 DF,  p-value: < 2.2e-16
```

```r
anova(lm(excess ~ ydstogo + yardline, data = gfi_train))
```

```
## Analysis of Variance Table
##
## Response: excess
##             Df Sum Sq Mean Sq F value    Pr(>F)
## ydstogo      1   7238  7238.3  98.337 < 2.2e-16 ***
## yardline     1   8855  8855.0 120.301 < 2.2e-16 ***
## Residuals 2314 170327    73.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# looks solid

excess_model1 = lm(excess ~ ydstogo + yardline, data = gfi_train)
excess_model1_intercept = excess_model1$coefficients[1]
excess_model1_coef_a = excess_model1$coefficients[2]
excess_model1_coef_b = excess_model1$coefficients[3]

excess_model1_equation = function(ydstogo, yardline) {
  excess_model1_intercept + excess_model1_coef_a * ydstogo + excess_model1_coef_b * yardline
}

excess_model1_equation(5, 10)
```

```
## (Intercept)
##    4.544718
```

```r
excess_model1_equation(30, 90)
```

```
## (Intercept)
##    20.72696
```

```r
# check performance with test data
```

```r
gfi_test$prediction1 = excess_model1_equation(gfi_test$ydstogo, gfi_test$yardline)
gfi_test$error1 = abs(gfi_test$excess - gfi_test$prediction1)

# MAE
sum(gfi_test$error1)/nrow(gfi_test)
```

```
## [1] 5.74256
```

```r
# more advanced model
summary(lm(excess ~ ydstogo + ydstogo_squared + ydstogo_cubed + yardline + yardline_squared + yardline_c
```

```
##
## Call:
## lm(formula = excess ~ ydstogo + ydstogo_squared + ydstogo_cubed +
##     yardline + yardline_squared + yardline_cubed, data = gfi_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.247  -5.172  -2.361   1.426  58.413
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -2.621e+00  6.149e-01  -4.263 2.10e-05 ***
## ydstogo           1.721e+00  2.505e-01   6.873 8.08e-12 ***
## ydstogo_squared  -1.528e-01  3.124e-02  -4.891 1.07e-06 ***
## ydstogo_cubed     3.724e-03  9.660e-04   3.856 0.000119 ***
## yardline          3.836e-01  6.029e-02   6.362 2.39e-10 ***
## yardline_squared -6.627e-03  1.770e-03  -3.745 0.000185 ***
## yardline_cubed    3.891e-05  1.434e-05   2.713 0.006725 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.423 on 2310 degrees of freedom
## Multiple R-squared:  0.1209, Adjusted R-squared:  0.1187
## F-statistic: 52.97 on 6 and 2310 DF,  p-value: < 2.2e-16
```

```r
anova(lm(excess ~ ydstogo + ydstogo_squared + ydstogo_cubed + yardline + yardline_squared + yardline_cu
```

```
## Analysis of Variance Table
##
## Response: excess
##                    Df Sum Sq Mean Sq  F value    Pr(>F)
## ydstogo             1   7238  7238.3 102.0328 < 2.2e-16 ***
## ydstogo_squared     1   2404  2403.6  33.8814 6.672e-09 ***
## ydstogo_cubed       1   1260  1260.1  17.7619 2.600e-05 ***
## yardline            1   8678  8678.1 122.3280 < 2.2e-16 ***
## yardline_squared    1   2444  2444.1  34.4531 4.995e-09 ***
## yardline_cubed      1    522   522.0   7.3582  0.006725 **
## Residuals        2310 163874    70.9
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# every single term here is significant

excess_model2 = lm(excess ~ ydstogo + ydstogo_squared + ydstogo_cubed + yardline + yardline_squared + ya

excess_model2_equation = function(ydstogo, yardline) {
  excess_model2$coefficients[1] + excess_model2$coefficients[2] * ydstogo + excess_model2$coefficients[3
}

excess_model2_equation(1, 1)
```

```
## (Intercept)
##  -0.6718014
```

```r
excess_model2_equation(30, 90)
```

```
## (Intercept)
##    21.29527
```

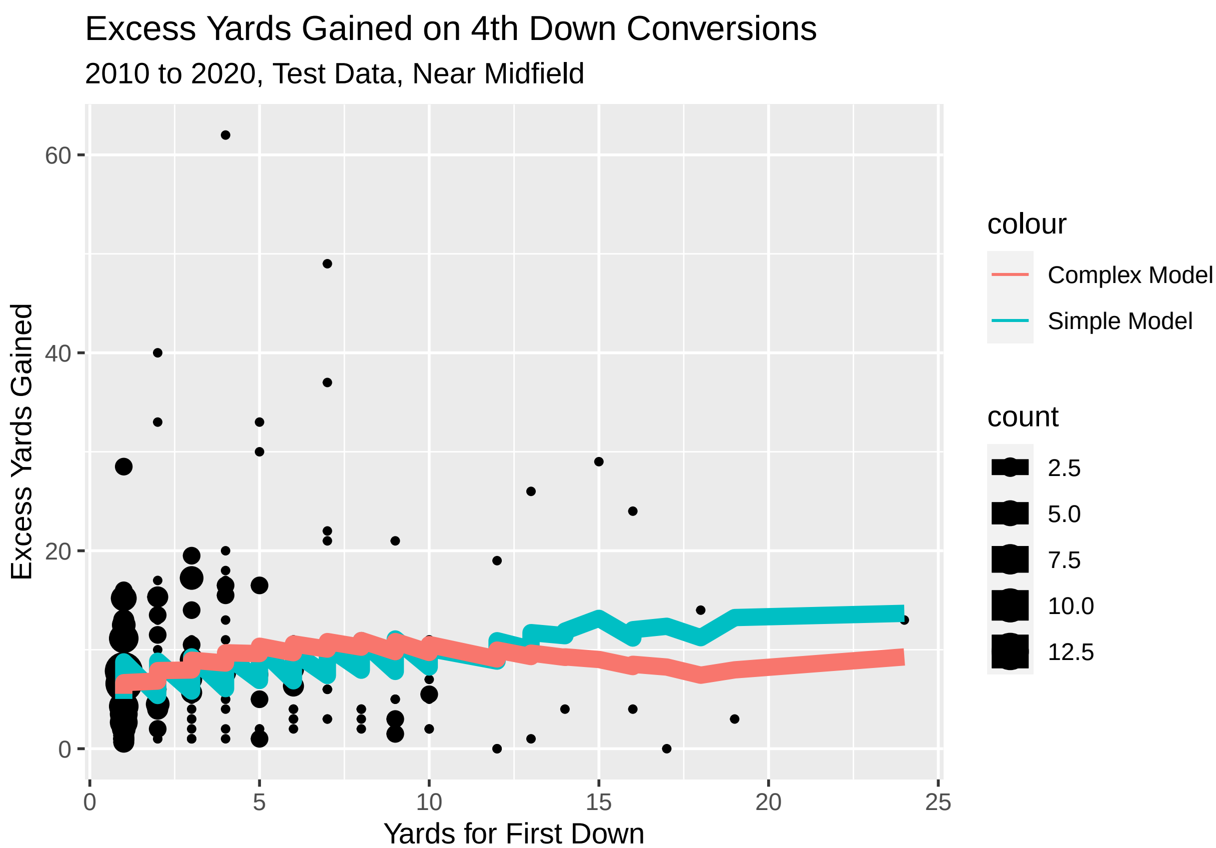
```r
gfi_test$prediction2 = excess_model2_equation(gfi_test$ydstogo, gfi_test$yardline)
gfi_test$error2 = abs(gfi_test$excess - gfi_test$prediction2)

# MAE
sum(gfi_test$error2)/nrow(gfi_test)
```

```
## [1] 5.641885
```

We can now visually compare the two models.

```r
ggplot(data = subset(gfi_test_agg, yardline >= 30 & yardline <= 70),
       mapping = aes(x = ydstogo,
                     y = mean))+
  ggtitle("Excess Yards Gained on 4th Down Conversions",
          subtitle = "2010 to 2020, Test Data, Near Midfield")+
  xlab("Yards for First Down")+
  ylab("Excess Yards Gained")+
  geom_point(aes(size = count))+
  geom_line(aes(y = excess_model1_equation(ydstogo, yardline),
                size = 3, colour = "Simple Model"))+
  geom_line(aes(y = excess_model2_equation(ydstogo, yardline),
                size = 3, colour = "Complex Model"))
```



There does not seem to be much difference at all between the two equations. The MAE values are similar and the models are similar visually. I would say the simpler one is slightly better because the complex model sometimes gives negatives in niche situations which should be impossible.

Objective 3: The opposite of Objective 2. When teams fail to convert on fourth down, predict the amount of yards short they are from the first down marker.

```r
# make simple model
summary(lm(shortage ~ ydstogo + yardline, data = gfi_train))
```

```
##
## Call:
## lm(formula = shortage ~ ydstogo + yardline, data = gfi_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.8721  -0.8952  -0.6129   0.0290  28.6493
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.950697   0.161827   5.875 4.85e-09 ***
## ydstogo     0.899811   0.015938  56.458  < 2e-16 ***
## yardline    0.002791   0.003850   0.725    0.469
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 3.948 on 2301 degrees of freedom
## Multiple R-squared:  0.6128, Adjusted R-squared:  0.6125
## F-statistic:  1821 on 2 and 2301 DF,  p-value: < 2.2e-16
```

```r
anova(lm(shortage ~ ydstogo + yardline, data = gfi_train))
```

```
## Analysis of Variance Table
##
## Response: shortage
##             Df Sum Sq Mean Sq   F value Pr(>F)
## ydstogo      1  56764   56764 3640.9645 <2e-16 ***
## yardline     1      8       8    0.5256 0.4685
## Residuals 2301  35873      16
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Yardline seems insignificant. Let's remove it

summary(lm(shortage ~ ydstogo, data = gfi_train))
```

```
##
## Call:
## lm(formula = shortage ~ ydstogo, data = gfi_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.9118  -0.9329  -0.6441   0.0671  28.6070
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.02911    0.12036    8.55   <2e-16 ***
## ydstogo      0.90376    0.01498   60.35   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.948 on 2302 degrees of freedom
## Multiple R-squared:  0.6127, Adjusted R-squared:  0.6125
## F-statistic:  3642 on 1 and 2302 DF,  p-value: < 2.2e-16
```

```r
anova(lm(shortage ~ ydstogo, data = gfi_train))
```

```
## Analysis of Variance Table
##
## Response: shortage
##             Df Sum Sq Mean Sq F value    Pr(>F)
## ydstogo      1  56764   56764  3641.7 < 2.2e-16 ***
## Residuals 2302  35881      16
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Looks better now

shortage_model1 = lm(shortage ~ ydstogo, data = gfi_train)
shortage_model1_intercept = excess_model1$coefficients[1]
shortage_model1_coef_a = excess_model1$coefficients[2]

shortage_model1_equation = function(ydstogo) {
  shortage_model1_intercept + shortage_model1_coef_a * ydstogo
}

shortage_model1_equation(1)
```

```
## (Intercept)
##    2.217951
```

```r
shortage_model1_equation(10)
```

```
## (Intercept)
##    5.344684
```

```r
# check performance with test data

gfi_test$prediction1 = shortage_model1_equation(gfi_test$ydstogo)
gfi_test$error1 = abs(gfi_test$shortage - gfi_test$prediction1)

# MAE
sum(gfi_test$error1)/nrow(gfi_test)
```

```
## [1] 3.270582
```

```r
# complex model
gfi_train$sq = gfi_train$ydstogo^2
gfi_train$cu = gfi_train$ydstogo^3

summary(lm(shortage ~ ydstogo + sq + cu, data = gfi_train))
```

```
##
## Call:
## lm(formula = shortage ~ ydstogo + sq + cu, data = gfi_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.1186  -0.7773  -0.6577  -0.0835  28.8444
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.5500514  0.1845917   2.980  0.00291 **
## ydstogo      1.1244930  0.0701341  16.033  < 2e-16 ***
## sq          -0.0171321  0.0061302  -2.795  0.00524 **
## cu           0.0002932  0.0001396   2.100  0.03586 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.939 on 2300 degrees of freedom
## Multiple R-squared:  0.6147, Adjusted R-squared:  0.6142
## F-statistic:  1223 on 3 and 2300 DF,  p-value: < 2.2e-16
```

```
anova(lm(shortage ~ ydstogo + sq + cu, data = gfi_train))
```

```
## Analysis of Variance Table
##
## Response: shortage
##             Df Sum Sq Mean Sq   F value    Pr(>F)
## ydstogo      1  56764   56764 3657.7228 < 2.2e-16 ***
## sq           1    120     120    7.7100  0.005536 **
## cu           1     68      68    4.4088  0.035862 *
## Residuals 2300  35693      16
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
shortage_model2 = lm(shortage ~ ydstogo + sq + cu, data = gfi_train)
shortage_model2_intercept = shortage_model2$coefficients[1]
shortage_model2_coef_a = shortage_model2$coefficients[2]
shortage_model2_coef_b = shortage_model2$coefficients[3]
shortage_model2_coef_c = shortage_model2$coefficients[4]

shortage_model2_equation = function(ydstogo) {
  shortage_model2_intercept + shortage_model2_coef_a * ydstogo + shortage_model2_coef_b * ydstogo^2 + sl
}

shortage_model2_equation(1)
```

```
## (Intercept)
##    1.657705
```

```
shortage_model2_equation(10)
```

```
## (Intercept)
##    10.37494
```

```
# check performance with test data

gfi_test$prediction2 = shortage_model2_equation(gfi_test$ydstogo)
gfi_test$error2 = abs(gfi_test$shortage - gfi_test$prediction2)

# MAE
sum(gfi_test$error2)/nrow(gfi_test)
```

```
## [1] 1.816694
```

```r
# model with interaction term
summary(lm(shortage ~ ydstogo * yardline, data = gfi_train))
```

```
##
## Call:
## lm(formula = shortage ~ ydstogo * yardline, data = gfi_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.9528  -0.8842  -0.5949  -0.0306  28.5089
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.6168550  0.2172696   2.839  0.00456 **
## ydstogo           0.9680506  0.0336715  28.750  < 2e-16 ***
## yardline          0.0114675  0.0053873   2.129  0.03339 *
## ydstogo:yardline -0.0014922  0.0006487  -2.300  0.02153 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.945 on 2300 degrees of freedom
## Multiple R-squared:  0.6137, Adjusted R-squared:  0.6132
## F-statistic:  1218 on 3 and 2300 DF,  p-value: < 2.2e-16
```

```r
anova(lm(shortage ~ ydstogo * yardline, data = gfi_train))
```

```
## Analysis of Variance Table
##
## Response: shortage
##                    Df Sum Sq Mean Sq   F value  Pr(>F)
## ydstogo             1  56764   56764 3647.7532 < 2e-16 ***
## yardline            1      8       8    0.5266 0.46811
## ydstogo:yardline    1     82      82    5.2903 0.02153 *
## Residuals        2300  35791      16
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
shortage_model3 = lm(shortage ~ ydstogo * yardline, data = gfi_train)
shortage_model3_intercept = shortage_model3$coefficients[1]
shortage_model3_coef_a = shortage_model3$coefficients[2]
shortage_model3_coef_b = shortage_model3$coefficients[3]
shortage_model3_coef_c = shortage_model3$coefficients[4]

shortage_model3_equation = function(ydstogo, yardline) {
  shortage_model3_intercept + shortage_model3_coef_a * ydstogo + shortage_model3_coef_b * yardline + sho
}

shortage_model3_equation(5, 10)
```

```
## (Intercept)
##    5.497175
```

```
gfi_test$prediction3 = shortage_model3_equation(gfi_test$ydstogo, gfi_test$yardline)
gfi_test$error3 = abs(gfi_test$shortage - gfi_test$prediction3)

# MAE
sum(gfi_test$error3)/nrow(gfi_test)
```

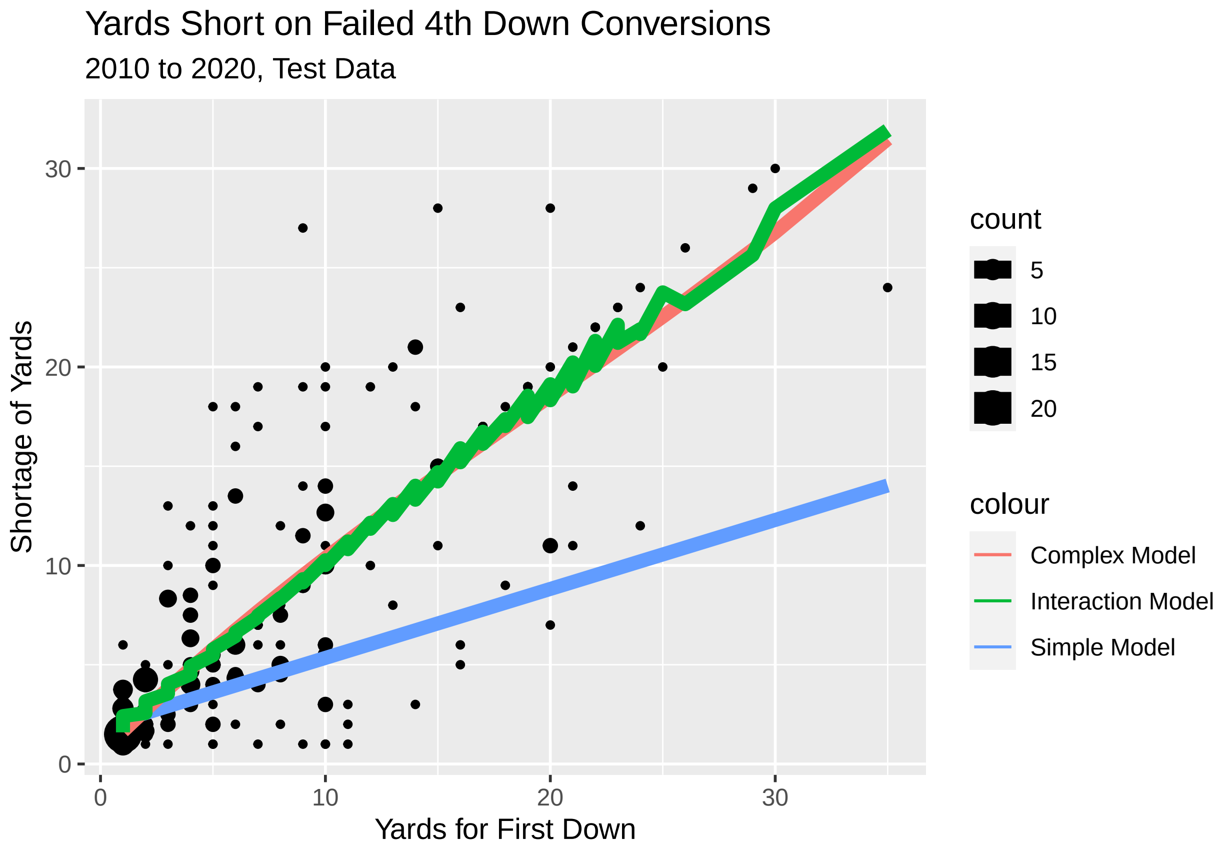
```
## [1] 1.797503
```

Visualition of Objective 3:

```
gfi_test_agg = gfi_test %>%
  group_by(ydstogo, yardline) %>%
  summarise("mean" = mean(shortage), "count" = length(shortage))
```

```
## `summarise()` has grouped output by 'ydstogo'. You can override using the `.groups` argument.
```

```
ggplot(data = subset(gfi_test_agg, yardline >= 0 & yardline <= 100),
       mapping = aes(x = ydstogo,
                     y = mean))+
  ggtitle("Yards Short on Failed 4th Down Conversions",
          subtitle = "2010 to 2020, Test Data")+
  xlab("Yards for First Down")+
  ylab("Shortage of Yards")+
  geom_point(aes(size = count))+
  geom_line(aes(y = shortage_model1_equation(ydstogo),
                size = 3, colour = "Simple Model"))+
  geom_line(aes(y = shortage_model2_equation(ydstogo),
                size = 3, colour = "Complex Model"))+
  geom_line(aes(y = shortage_model3_equation(ydstogo, yardline),
                size = 3, colour = "Interaction Model"))
```



It seems as if either the interaction term or complex model are the best. I think the best bet is the interaction term model because it has the lowest MAE.