Field Goals

Sam Hughes

1/8/2022

With this code, I create and compare models to best predict field goal success by kick distance. 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.

My first attempt was a simple logarithmic model with no transformations, but it unfortunately struggled to fit both the train and test data. Therefore, I tried a few other models and compared them to see which seemed the best.

```
# Build polynomial model
fg_model_3 = glm(fg_result ~ kick_distance + square + cube, data = fg_train, family = "binomial")
summary(fg_model_3)
```

```
##
## Call:
## glm(formula = fg_result ~ kick_distance + square + cube, family = "binomial",
##     data = fg_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1904  -0.0014   0.3793   0.6053   1.9705
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    1.769e+01  1.786e+00   9.909  < 2e-16 ***
## kick_distance -9.973e-01  1.266e-01  -7.880 3.26e-15 ***
## square         2.175e-02  2.917e-03   7.457 8.87e-14 ***
## cube          -1.707e-04  2.196e-05  -7.775 7.55e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 11715  on 10268  degrees of freedom
## Residual deviance:  6840  on 10265  degrees of freedom
## AIC: 6848
##
## Number of Fisher Scoring iterations: 8
```

```
# Obtain equation constants from model
fg_intercept_3 = fg_model_3$coefficients["(Intercept)"]
fg_coef_3a = fg_model_3$coefficients["kick_distance"]
```

```r
fg_coef_3b = fg_model_3$coefficients["square"]
fg_coef_3c = fg_model_3$coefficients["cube"]

fg_null_3 = fg_model_3$null.deviance/(-2)
fg_proposed_3 = fg_model_3$deviance/(-2)

# McFadden R-squared
(fg_null_3 - fg_proposed_3)/fg_null_3
```

```
## [1] 0.4161212
```

```r
# Chi-Squared P-Value
1 - pchisq(2*(fg_proposed_3-fg_null_3), df = length(fg_model_3$coefficients) - 1)
```

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

```r
# Equation made by training model
fg_equation = function(x) {
  return(exp(fg_intercept_3 + fg_coef_3a*x + fg_coef_3b*x^2 + fg_coef_3c*x^3)/(1 + exp(fg_intercept_3 +
}
fg_equation(50)
```

```
## (Intercept)
##   0.7048411
```

```r
# Score model
fg_test$prediction = fg_equation(fg_test$kick_distance)
fg_test$binary = 0
fg_test[which(fg_test$prediction >= 0.5), "binary"] = 1
sum(fg_test$fg_result == fg_test$binary)/nrow(fg_test)
```

```
## [1] 0.8565302
```

```r
fg_test$fg_result = as.factor(fg_test$fg_result)
fg_test$binary = as.factor(fg_test$binary)
str(fg_test)
```

```
## 'data.frame':    2565 obs. of  8 variables:
##  $ fg_result    : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ kick_distance: num  18 18 19 19 19 19 19 19 19 19 ...
##  $ square       : num  324 324 361 361 361 361 361 361 361 361 ...
##  $ cube         : num  5832 5832 6859 6859 6859 ...
##  $ ln           : num  2.89 2.89 2.94 2.94 2.94 ...
##  $ exp          : num  6.57e+07 6.57e+07 1.78e+08 1.78e+08 1.78e+08 ...
##  $ prediction   : num  0.997 0.997 0.996 0.996 0.996 ...
##  $ binary       : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

```r
confusionMatrix(fg_test$binary, fg_test$fg_result)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  310   19
##          1  349 1887
##
##                Accuracy : 0.8565
##                  95% CI : (0.8424, 0.8699)
##     No Information Rate : 0.7431
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5506
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4704
##             Specificity : 0.9900
##          Pos Pred Value : 0.9422
##          Neg Pred Value : 0.8439
##              Prevalence : 0.2569
##          Detection Rate : 0.1209
##    Detection Prevalence : 0.1283
##       Balanced Accuracy : 0.7302
##
##        'Positive' Class : 0
##
```

```
# Build logorithmic model
fg_model_ln = glm(fg_result ~ ln, data = fg_train, family = "binomial")
summary(fg_model_ln)
```

```
##
## Call:
## glm(formula = fg_result ~ ln, family = "binomial", data = fg_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4490  -0.3578   0.2691   0.6030   1.6545
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   24.026      0.510   47.11   <2e-16 ***
## ln            -6.036      0.132  -45.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 11714.8  on 10268  degrees of freedom
## Residual deviance:  7264.5  on 10267  degrees of freedom
## AIC: 7268.5
##
## Number of Fisher Scoring iterations: 6
```

```r
# Obtain equation constants from model
fg_intercept_ln = fg_model_ln$coefficients["(Intercept)"]
fg_coef_ln = fg_model_ln$coefficients["ln"]


fg_null_ln = fg_model_ln$null.deviance/(-2)
fg_proposed_ln = fg_model_ln$deviance/(-2)

# McFadden R-squared
(fg_null_ln - fg_proposed_ln)/fg_null_ln
```

```
## [1] 0.3798889
```

```r
# Chi-Squared P-Value
1 - pchisq(2*(fg_proposed_ln-fg_null_ln), df = length(fg_model_ln$coefficients) - 1)
```

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

```r
# Equation made by training model
fg_equation_ln = function(x) {
  return(exp(fg_intercept_ln + fg_coef_ln*log(x))/(1 + exp(fg_intercept_ln + fg_coef_ln*log(x))))
}


# Score model
fg_test$prediction_ln = fg_equation_ln(fg_test$kick_distance)
fg_test$binary_ln = 0
fg_test[which(fg_test$prediction_ln >= 0.5), "binary_ln"] = 1
fg_test$binary_ln = as.factor(fg_test$binary_ln)
str(fg_test)
```

```
## 'data.frame':    2565 obs. of  10 variables:
##  $ fg_result    : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ kick_distance: num  18 18 19 19 19 19 19 19 19 19 ...
##  $ square       : num  324 324 361 361 361 361 361 361 361 361 ...
##  $ cube         : num  5832 5832 6859 6859 6859 ...
##  $ ln           : num  2.89 2.89 2.94 2.94 2.94 ...
##  $ exp          : num  6.57e+07 6.57e+07 1.78e+08 1.78e+08 1.78e+08 ...
##  $ prediction   : num  0.997 0.997 0.996 0.996 0.996 ...
##  $ binary       : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ prediction_ln: num  0.999 0.999 0.998 0.998 0.998 ...
##  $ binary_ln    : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

```r
sum(fg_test$fg_result == fg_test$binary_ln)/nrow(fg_test)
```

```
## [1] 0.8553606
```

```r
confusionMatrix(fg_test$binary_ln, fg_test$fg_result)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  348   60
##          1  311 1846
##
##                Accuracy : 0.8554
##                  95% CI : (0.8411, 0.8688)
##     No Information Rate : 0.7431
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5673
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5281
##             Specificity : 0.9685
##          Pos Pred Value : 0.8529
##          Neg Pred Value : 0.8558
##              Prevalence : 0.2569
##          Detection Rate : 0.1357
##    Detection Prevalence : 0.1591
##       Balanced Accuracy : 0.7483
##
##        'Positive' Class : 0
##
```

```
# Exponential model
fg_model_exp = glm(fg_result ~ exp, data = fg_train, family = "binomial")
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(fg_model_exp)
```

```
##
## Call:
## glm(formula = fg_result ~ exp, family = "binomial", data = fg_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9221   0.0000   0.5858   0.5858   0.5910
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.676e+00  2.884e-02  58.106  < 2e-16 ***
## exp         -3.107e-30  4.696e-31  -6.616 3.68e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 11714.8  on 10268  degrees of freedom
## Residual deviance:  7889.1  on 10267  degrees of freedom
## AIC: 7893.1
##
## Number of Fisher Scoring iterations: 25
```

```
# Obtain equation constants from model
fg_intercept_exp = fg_model_exp$coefficients["(Intercept)"]
fg_coef_exp = fg_model_exp$coefficients["exp"]


fg_null_exp = fg_model_exp$null.deviance/(-2)
fg_proposed_exp = fg_model_exp$deviance/(-2)

# McFadden R-squared
(fg_null_exp - fg_proposed_exp)/fg_null_exp
```

```
## [1] 0.3265722
```

```
# Chi-Squared P-Value
1 - pchisq(2*(fg_proposed_exp-fg_null_exp), df = length(fg_model_exp$coefficients) - 1)
```

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

```
# Equation made by training model
fg_equation_exp = function(x) {
  return(exp(fg_intercept_exp + fg_coef_exp*exp(x))/(1 + exp(fg_intercept_exp + fg_coef_exp*exp(x))))
}


# Score model
fg_test$prediction_exp = fg_equation_exp(fg_test$kick_distance)
fg_test$binary_exp = 0
fg_test[which(fg_test$prediction_exp >= 0.5), "binary_exp"] = 1
fg_test$binary_exp = as.factor(fg_test$binary_exp)
str(fg_test)
```

```
## 'data.frame':    2565 obs. of  12 variables:
##  $ fg_result     : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ kick_distance : num  18 18 19 19 19 19 19 19 19 19 ...
##  $ square        : num  324 324 361 361 361 361 361 361 361 361 ...
##  $ cube          : num  5832 5832 6859 6859 6859 ...
##  $ ln            : num  2.89 2.89 2.94 2.94 2.94 ...
##  $ exp           : num  6.57e+07 6.57e+07 1.78e+08 1.78e+08 1.78e+08 ...
##  $ prediction    : num  0.997 0.997 0.996 0.996 0.996 ...
##  $ binary        : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ prediction_ln : num  0.999 0.999 0.998 0.998 0.998 ...
##  $ binary_ln     : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ prediction_exp: num  0.842 0.842 0.842 0.842 0.842 ...
##  $ binary_exp    : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

```r
sum(fg_test$fg_result == fg_test$binary_exp)/nrow(fg_test)
```

```
## [1] 0.8545809
```

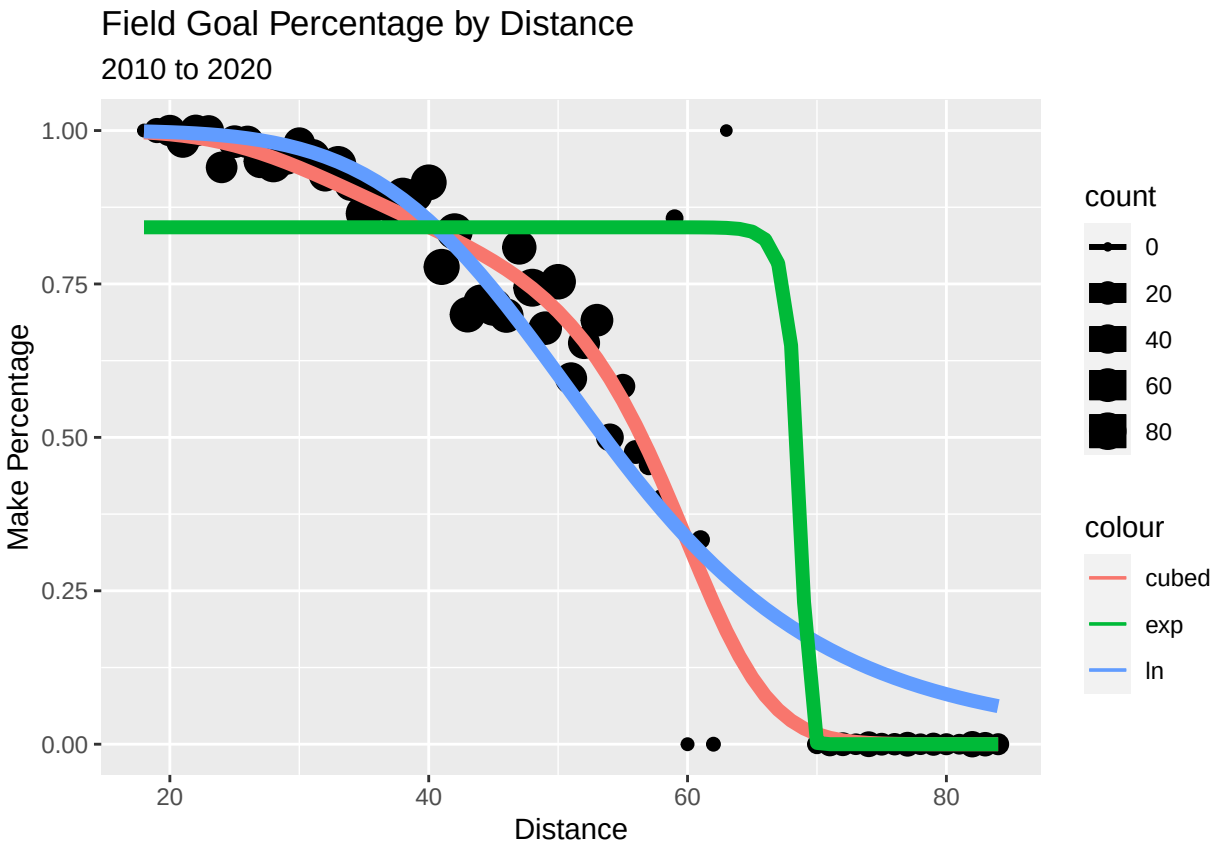
```r
confusionMatrix(fg_test$binary_exp, fg_test$fg_result)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  286    0
##          1  373 1906
##
##                Accuracy : 0.8546
##                  95% CI : (0.8403, 0.868)
##     No Information Rate : 0.7431
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5326
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4340
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.8363
##              Prevalence : 0.2569
##          Detection Rate : 0.1115
##    Detection Prevalence : 0.1115
##       Balanced Accuracy : 0.7170
##
##        'Positive' Class : 0
##
```

Now that the models are built, we can make a plot to visualize (using aggregated test data).

```r
# Make plot for visualization
ggplot(data = fg_test_agg,
       mapping = aes(x = kick_distance,
                     y = percentage))+
  ggtitle("Field Goal Percentage by Distance", subtitle = "2010 to 2020")+
  xlab("Distance")+
  ylab("Make Percentage")+
  geom_point(aes(size = count))+
  geom_line(aes(y = fg_equation(kick_distance),
                size = 7, colour = "cubed"))+
  geom_line(aes(y = fg_equation_ln(kick_distance),
                size = 7, colour = "ln"))+
  geom_line(aes(y = fg_equation_exp(kick_distance),
                size = 7, colour = "exp"))
```

```
## Warning: Removed 6 rows containing missing values (geom_point).
```



It seems that the best model is the polynomial. While the logarithmic model may arguably be slightly better than the polynomial model at shorter ranges, it really struggles at long range. Exponential is clearly the worst.