NBA 5-min or less Data Model - Feature Selection

This is for the feature selection component of the overall project. This overall project is still a bit out from finishing, but I fulfilled the needs of the course with the feature selection. Please stay tune for the overall completion! This will be a hierarchal, random-intercept negative binomial which I will briefly cover why in the EDA section.

EXECUTIVE Summary

The overall goal is to project the win probabilites of NBA games based on play-by-play results. However, we want a large and robust consideration of various interaction terms, but this will end up creating the curse of dimensionality. So we used a bit of logical filtering with correlation strength & then from the deduplicated data, we performed a Ridge and LASSO with hierarchal, random-intercept regression to reduce our dataset for the running of the final algorithm. This aided by lambda values which reflect time windows of the game. In the end, we got 3 features which should serve as a simple but effective probability training data.

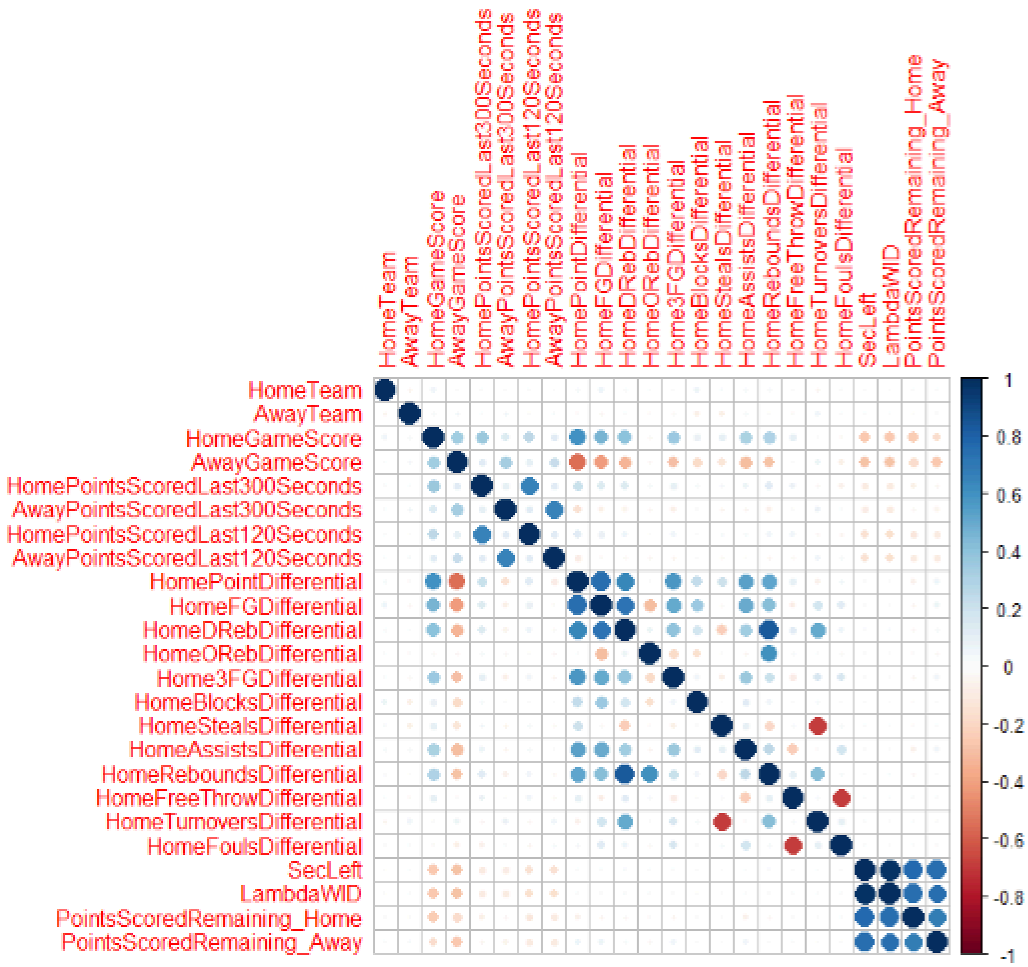
The final features which were determined for use in the model:

```
c("HomeGameScore","AwayGameScore",
  "SQRTHomeGameScore*AwayGameScore")
```

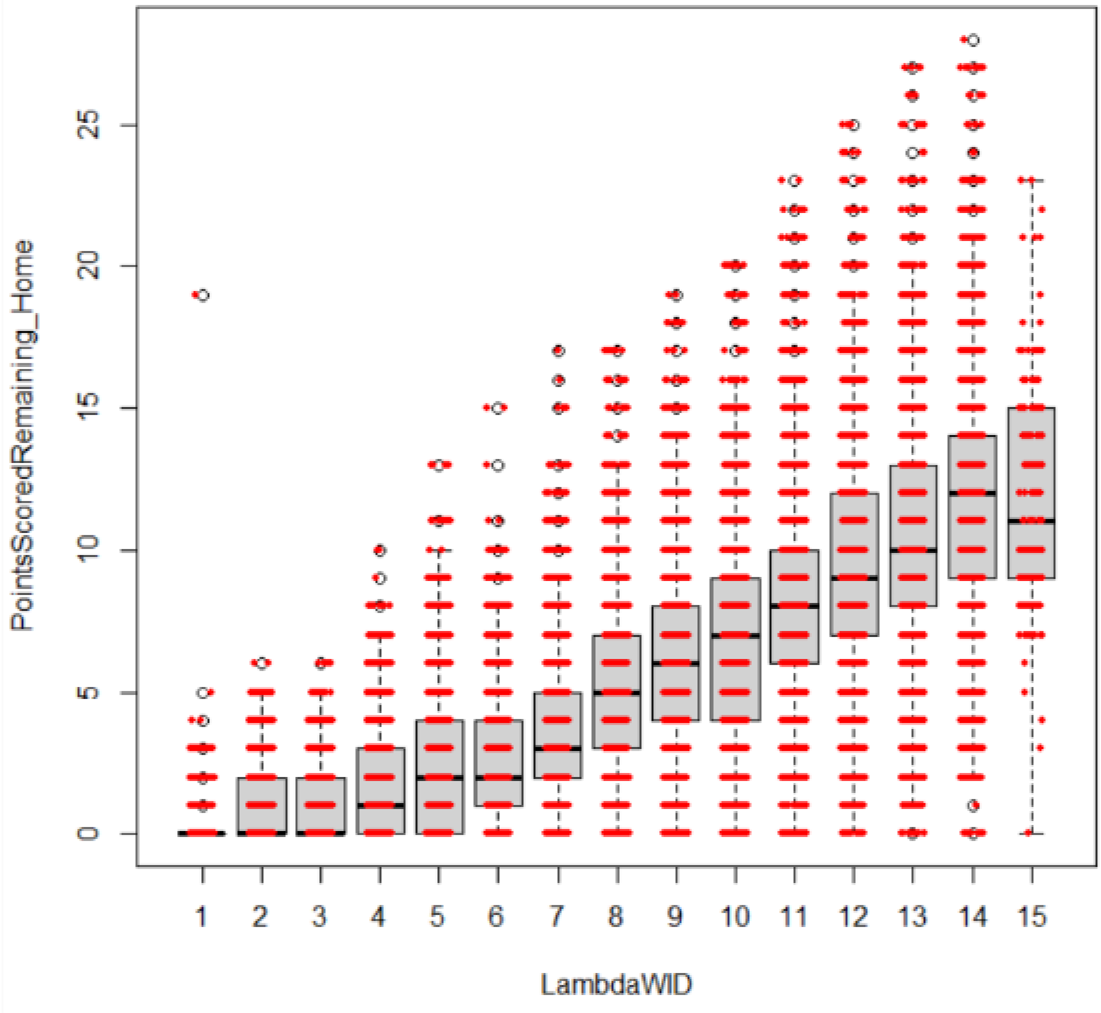
Data

NBA 2019-2020 Data within the last 5 mins of the 4 quarter-OT

EDA



For EDA, we did a systematic approach to creating variable interactions. This ultimately generated over 7000 features which we used a correlation strength of .1 to filter out the initial down to around 300. We then kept the strongest of the repetitive correlations before our Ridge and Lasso runs.



Next, we did some analysis of the dispersion of the data to determine decide the best model which this will ultimately run through. As this showcased overdispersion, it was determined a negative binomial would be best here. However, we also need to seperate each lambda into windows as seen above.

Target Variables

This was the amount of points scored by the team remaining in the game, `PointsScoredRemaining_` home or away.

Variables Transformations

During the EDA process, I wanted to explore various interaction terms of the different variables since it was not determined that any of the traditional statistics had overwhelming predicitive strength. To attempt to get more correlative features, I created interaction variables to explore potential non-linear variable related behavior. To review the entire function content, see the attached notebook below:

```
transformations <- function(data) {
  # Loop through selected columns
  for (col1 in colnames(data)[4:21]) {
```

```
      for (col2 in colnames(data)[4:21]) {
        # Skip when the columns are the same
        if (col1 == col2) {
          next
        }

        ## Transformations for col1 # log, inverse, sqrt, powers
        data[[paste0(col1, "^2")]] = data[[col1]]^2
        ...

        ## Interaction terms
        # 2nd-degree polynomial
        data[[paste0(col1, "^2*", col2, "^2")]] =
(data[[col1]]^2) * (data[[col2]]^2)
        # Powers
        data[[paste0(col1, "^2/", col2)]] = (data[[col1]]^2) /
(data[[col2]] + 1e-8)
        ...
        # Log
        data[[paste0("LogOf", col1, "/", col2)]] =
log(abs(data[[col1]]) + 1e-8) / (data[[col2]] + 1e-8)
        ...
        # Inverse
        data[[paste0("Inverse", col1, "/LogOf", col2)]] = (1 /
(data[[col1]] + 1e-8)) / log(abs(data[[col2]]) + 1e-8)
        ...
        # Sqrt
        data[[paste0("SQRT", col1, "/", col2, "^2")]] =
sqrt(abs(data[[col1]]) + 1e-8) / ((data[[col2]]^2) + 1e-8)
        ...
      }
    }
    return(data)
  }
```

In addition, I performed analysis of the medians by their lambda window. We will use a value similar to these for our priors on our lambda window data:

```
lambdaWindowMedianData = function(data) {
  for (col in colnames(data)[4:21]) {
    # Binary column for whether value is above/below lambda
window median
    data[[paste0("LamWin_MedianThreshold_", col)]] = with(data,
      ifelse(
        data[[col]] > ave(data[[col]], data[["LambdaWID"]], FUN
= function(x) median(x, na.rm = TRUE)),
        1,
        0
      )
    )

    # Difference from median column
```

```
    data[[paste0("LamWin_DifferenceFromMedian_", col)]] =
with(data,
      data[[col]] - ave(data[[col]], data[["LambdaWID"]], FUN =
function(x) median(x, na.rm = TRUE))
    )
  }
  return(data)  # Return the modified dataset
}
```

This resulted in about 8000 variables created, ready to be eliminated >:)

Reducing the variables logically-*but-systematically* - pre-LASSO and Ridge feature reduction

As stated in the summary, we used correlation strength to be the initial barrier to entry into the ultimate set of features. With a set threshold of 0.1, we were able to reduce our data to about 400 variables, but obviously this is a bit robust & computationally impractical even for a well running machine. If you were to review the output of the columns, you would find that many found there is a large amount of repetitive features in the GameScore interactions and the 'PointsScoredLast\d{3,}Seconds' interactions. To finish our manual discernment, we will only keep the 3 strongest of these to avoid too much redundancy.

We got to this point:

```
 [1] "HomePointsScoredLast120Seconds"
 [2] "HomeGameScore"
 [3] "AwayGameScore"
 [4] "SecLeft"
 [5] "SQRTHomeTurnoversDifferential*HomeFGDifferential"
 [6] "LogOfHomeGameScore*SQRTAwayGameScore"
 [7] "SQRTAwayGameScore*LogOfHomeGameScore"
 [8] "SQRTHomeGameScore*AwayGameScore"
 [9]
"HomePointsScoredLast120Seconds^2*AwayPointsScoredLast120Seconds"
[10]
"HomePointsScoredLast120Seconds^2*SQRTAwayPointsScoredLast120Seconds"
[11]
"SQRTAwayPointsScoredLast120Seconds*HomePointsScoredLast120Seconds^2"
```

However, the initial runs of the feature selection algorithms proved to be weighed down computationally by the dimensions of the data due to the random-hierarchal structure, even when run with a subsample. This provided another component I need to use some logical discernment, dropping the following:

- `SecLeft` since we are using time windows
- During the initial run which did not come close overall to convergence, only `SQRTHomeGameScore*AwayGameScore"` and `"HomePointsScoredLast120Seconds^2*AwayPointsScoredLast120Seconds"` converged, so these were decided to be the kept values: ##### Retained Variables

```
[1] "HomePointsScoredLast120Seconds"
[2] "HomeGameScore"
[3] "AwayGameScore"
[4] "SQRTHomeTurnoversDifferential*HomeFGDifferential"
[5] "SQRTHomeGameScore*AwayGameScore"
[6]
"HomePointsScoredLast120Seconds^2*AwayPointsScoredLast120Seconds"
```

Model and Convergence Diagnostics

LASSO

```
## LASSO
# laplace distribution for feature selection
ddexp = function(x, mu, tau) {
  0.5*tau*exp(-tau*abs(x-mu))
}

lasso_string = " model {
  for (i in 1:N) {
    home_y[i] ~ dnegbin(home_p[i], r)
    away_y[i] ~ dnegbin(away_p[i], r)
    home_p[i] = r / (r + home_mu[i])
    away_p[i] = r / (r + away_mu[i])
    log(home_mu[i]) = bhome_window[lambdaWindow[i]] +
inprod(X[i,], b_h[1:P])
    log(away_mu[i]) = baway_window[lambdaWindow[i]] +
inprod(X[i,], b_a[1:P])
  }

  for (w in 1:max(lambdaWindow)) {
    bhome_window[w] ~ dnorm(0, tau)
    baway_window[w] ~ dnorm(0, tau)
  }

  for (j in 1:P) {
    b_h[j] ~ ddexp(0.0, 1.0)
    b_a[j] ~ ddexp(0.0, 1.0)
  }

  tau ~ dgamma(1.0, 1.0)
  r ~ dgamma(5.0, 0.1)
}"

Iterations = 1541001:1561000
Thinning interval = 1
Number of chains = 2
Sample size per chain = 20000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

                   Mean      SD  Naive SE Time-series SE
```

```
b_a[1]       -0.005364 0.01469 7.343e-05      0.0001704
b_a[2]       -0.358656 0.07437 3.718e-04      0.0113674
b_a[3]       -0.741011 0.14421 7.211e-04      0.0219292
b_a[4]        0.048421 0.01294 6.471e-05      0.0001395
b_a[5]        0.921301 0.18431 9.216e-04      0.0284032
b_a[6]        0.101207 0.01824 9.121e-05      0.0002133
b_h[1]        0.001605 0.01494 7.469e-05      0.0001706
b_h[2]       -0.362894 0.07363 3.681e-04      0.0100136
b_h[3]       -0.547463 0.14123 7.061e-04      0.0214479
b_h[4]       -0.016858 0.01295 6.474e-05      0.0001431
b_h[5]        0.749712 0.18109 9.055e-04      0.0281350
b_h[6]        0.065575 0.01903 9.514e-05      0.0002200
b_window[1]  -1.801990 0.12024 6.012e-04      0.0007853
b_window[2]  -0.076065 0.08216 4.108e-04      0.0005372
b_window[3]   0.126578 0.07218 3.609e-04      0.0004813
b_window[4]   0.421336 0.04178 2.089e-04      0.0002653
b_window[5]   0.629242 0.04072 2.036e-04      0.0002636
b_window[6]   0.999699 0.03933 1.966e-04      0.0002533
b_window[7]   1.241981 0.02571 1.286e-04      0.0001642
b_window[8]   1.588974 0.02284 1.142e-04      0.0001437
b_window[9]   1.832063 0.02145 1.072e-04      0.0001359
b_window[10]  1.936753 0.02122 1.061e-04      0.0001375
b_window[11]  2.110818 0.01960 9.802e-05      0.0001274
b_window[12]  2.211774 0.02009 1.004e-04      0.0001296
b_window[13]  2.327225 0.01981 9.907e-05      0.0001287
b_window[14]  2.426003 0.01878 9.392e-05      0.0001257

### at 1,021,000 records
Potential scale reduction factors:

             Point est. Upper C.I.
b_a[1]             1.00       1.00
b_a[2]             1.04       1.09
b_a[3]             1.04       1.10
b_a[4]             1.00       1.00
b_a[5]             1.04       1.10
b_a[6]             1.00       1.00
b_h[1]             1.00       1.00
b_h[2]             1.02       1.09
b_h[3]             1.02       1.09
b_h[4]             1.00       1.00
b_h[5]             1.02       1.09
b_h[6]             1.00       1.00

### at 1,141,000 records
Potential scale reduction factors:

             Point est. Upper C.I.
b_a[1]             1.00       1.00
b_a[2]             1.02       1.03
b_a[3]             1.02       1.03
b_a[4]             1.00       1.00
b_a[5]             1.02       1.03
```

```
b_a[6]            1.00       1.00
b_h[1]            1.00       1.01
b_h[2]            1.12       1.44
b_h[3]            1.13       1.45
b_h[4]            1.00       1.00
b_h[5]            1.13       1.45
b_h[6]            1.00       1.01

### at 1,561,000 records
             Point est. Upper C.I.
b_a[1]            1.00       1.00
b_a[2]            1.01       1.01
b_a[3]            1.01       1.01
b_a[4]            1.00       1.00
b_a[5]            1.01       1.01
b_a[6]            1.00       1.00
b_h[1]            1.00       1.00
b_h[2]            1.06       1.17
b_h[3]            1.07       1.19
b_h[4]            1.00       1.00
b_h[5]            1.07       1.19
b_h[6]            1.00       1.00


             b_a[1]    b_a[2]    b_a[3]      b_a[4]    b_a[5]
b_a[6]      b_h[1]    b_h[2]    b_h[3]
Lag 0   1.000000000 1.0000000 1.0000000 1.000000000 1.0000000
1.000000000 1.000000000 1.0000000 1.0000000
Lag 1   0.640957094 0.9866447 0.9963453 0.578412415 0.9976969
0.645915090 0.643623189 0.9852825 0.9959537
Lag 5   0.161775908 0.9621782 0.9857225 0.133116621 0.9892334
0.163138246 0.158294509 0.9588997 0.9844306
Lag 10  0.045866661 0.9475848 0.9747778 0.019685918 0.9789175
0.043342987 0.036826100 0.9430982 0.9724297
Lag 50 -0.009287408 0.8713920 0.8954324 0.009084605 0.8997517
-0.009534518 0.009440508 0.8522970 0.8835642
             b_h[4]    b_h[5]      b_h[6]  b_window[1]
b_window[2]   b_window[3]   b_window[4]
Lag 0   1.000000000 1.0000000 1.000000000  1.000000000
1.000000000  1.0000000000  1.000000e+00
Lag 1   0.585845351 0.9974870 0.650071179  0.251493760
0.254683960  0.2532538514  2.326777e-01
Lag 5   0.143370218 0.9881766 0.160292727 -0.002185915
0.004209266  0.0070516281 -1.989509e-03
Lag 10  0.030543716 0.9767695 0.036807631 -0.004525012
-0.001404283 -0.0031793768 -1.079564e-03
Lag 50 -0.007192496 0.8864262 0.005396747  0.008158443
0.005687955 -0.0003499853  9.603072e-05
        b_window[5]  b_window[6]  b_window[7]   b_window[8]
b_window[9] b_window[10]  b_window[11]
Lag 0   1.000000000  1.000000000 1.000000e+00  1.0000000000
1.000000000  1.000000000  1.0000000000
Lag 1   0.252313364  0.248639737 2.397869e-01  0.2258907235
```

```
0.232136337  0.246982295  0.2397190040
Lag 5   0.012378932  0.003748118 3.608893e-04 -0.0008225186
0.001355475  0.005953334  0.0010314406
Lag 10  0.009340103 -0.009572944 2.137542e-03 -0.0070878577
-0.011569506 -0.006660922  0.0047033741
Lag 50 -0.004260871 -0.003816599 9.861099e-05 -0.0086121059
-0.005202749  0.006048247 -0.0009104794
       b_window[12] b_window[13] b_window[14]
Lag 0   1.000000000  1.000000000  1.000000000
Lag 1   0.249491677  0.244597845  0.272983366
Lag 5  -0.007482899  0.001738089  0.001773646
Lag 10  0.017294001 -0.001114184 -0.002418907
Lag 50  0.008571348 -0.003995680 -0.003422975
```

From this standpoint

Ridge

```
ridge_string = "
model {
  for (i in 1:N) {
    home_y[i] ~ dnegbin(home_p[i], r)
    away_y[i] ~ dnegbin(away_p[i], r)
    home_p[i] = r / (r + home_mu[i])
    away_p[i] = r / (r + away_mu[i])
    log(home_mu[i]) = b_window[meanWindow[i]] + inprod(X[i,],
b_h[1:P])
    log(away_mu[i]) = b_window[meanWindow[i]] + inprod(X[i,],
b_a[1:P])
  }

  for (w in 1:max(meanWindow)) {
    b_window[w] ~ dnorm(0,tau_w)
  }

  for (j in 1:P) {
    b_h[j] ~ dnorm(0.0, tau_b)
    b_a[j] ~ dnorm(0.0, tau_b)
  }

  tau_w ~ dgamma(1.0,1.0)
  tau_b = 1 / sigma_b^2
  sigma_b ~ dunif(0, 10)

  r ~ dgamma(5.0, 0.1)
}
"
```

```
Iterations = 8691001:8701000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 10000
```

1. Empirical mean and standard deviation for each variable, plus standard error of the mean:

	Mean	SD	Naive SE	Time-series SE
b_a[1]	-0.28240	0.08441	4.873e-04	0.0162755
b_a[2]	-0.58904	0.16535	9.546e-04	0.0319498
b_a[3]	0.04811	0.01277	7.375e-05	0.0001639
b_a[4]	0.72724	0.21092	1.218e-03	0.0433609
b_a[5]	0.09702	0.01308	7.552e-05	0.0001037
b_h[1]	-0.30916	0.07925	4.576e-04	0.0123675
b_h[2]	-0.44202	0.15365	8.871e-04	0.0261525
b_h[3]	-0.01664	0.01317	7.603e-05	0.0001693
b_h[4]	0.61433	0.19657	1.135e-03	0.0346055
b_h[5]	0.06707	0.01344	7.757e-05	0.0001079
b_window[1]	-1.79887	0.11996	6.926e-04	0.0009042
b_window[2]	-0.07376	0.08135	4.697e-04	0.0005939
b_window[3]	0.13089	0.07150	4.128e-04	0.0005433
b_window[4]	0.42315	0.04149	2.395e-04	0.0003114
b_window[5]	0.63115	0.04097	2.366e-04	0.0003075
b_window[6]	1.00056	0.03889	2.246e-04	0.0002864
b_window[7]	1.24172	0.02557	1.477e-04	0.0001857
b_window[8]	1.58857	0.02297	1.326e-04	0.0001727
b_window[9]	1.83147	0.02125	1.227e-04	0.0001543
b_window[10]	1.93527	0.02114	1.220e-04	0.0001565
b_window[11]	2.10961	0.01976	1.141e-04	0.0001480
b_window[12]	2.21194	0.02001	1.155e-04	0.0001470
b_window[13]	2.32729	0.01963	1.133e-04	0.0001457
b_window[14]	2.42595	0.01893	1.093e-04	0.0001441

2. Quantiles for each variable:

	2.5%	25%	50%	75%	97.5%
b_a[1]	-0.43928	-0.33509	-0.28629	-0.238310	-0.089382
b_a[2]	-0.89577	-0.69227	-0.59684	-0.503295	-0.211915
b_a[3]	0.02302	0.03950	0.04816	0.056692	0.073111
b_a[4]	0.24514	0.61944	0.73760	0.857871	1.119170
b_a[5]	0.07130	0.08821	0.09704	0.105824	0.122464
b_h[1]	-0.46073	-0.36167	-0.31272	-0.261998	-0.125199
b_h[2]	-0.73384	-0.54427	-0.45009	-0.350735	-0.077337
b_h[3]	-0.04266	-0.02541	-0.01669	-0.007698	0.008942
b_h[4]	0.14695	0.49706	0.62510	0.745509	0.990338
b_h[5]	0.04056	0.05801	0.06704	0.076162	0.093143
b_window[1]	-2.04111	-1.87849	-1.79704	-1.716604	-1.571568
b_window[2]	-0.23669	-0.12773	-0.07263	-0.018448	0.083067
b_window[3]	-0.01084	0.08288	0.13117	0.180044	0.269191
b_window[4]	0.34161	0.39503	0.42338	0.451722	0.503171
b_window[5]	0.54973	0.60369	0.63170	0.658834	0.710407
b_window[6]	0.92292	0.97464	1.00066	1.026820	1.075780
b_window[7]	1.19127	1.22454	1.24179	1.258943	1.291524
b_window[8]	1.54372	1.57312	1.58845	1.603812	1.633957
b_window[9]	1.78952	1.81716	1.83149	1.845881	1.872672
b_window[10]	1.89353	1.92120	1.93535	1.949583	1.976512

```
b_window[11]  2.07058  2.09635  2.10962  2.122875  2.147912
b_window[12]  2.17262  2.19851  2.21191  2.225329  2.251285
b_window[13]  2.28908  2.31415  2.32726  2.340597  2.365926
b_window[14]  2.38910  2.41326  2.42588  2.438716  2.463176
```

As we can see, the coefficients which are showing the most effect are: `HomeGameScore`, `AwayGameScore`, `SQRTHomeGameScore*AwayGameScore`

In conclusion

It was determined the best features for this model were `HomeGameScore`, `AwayGameScore`, `SQRTHomeGameScore*AwayGameScore` via the systematic and logical approach I used for feature selection.