Using advanced analytics to predict the winner of the 2021/22 Premier League Season

Jose Llamas

Professor. Zambom's Math 444

Abstract

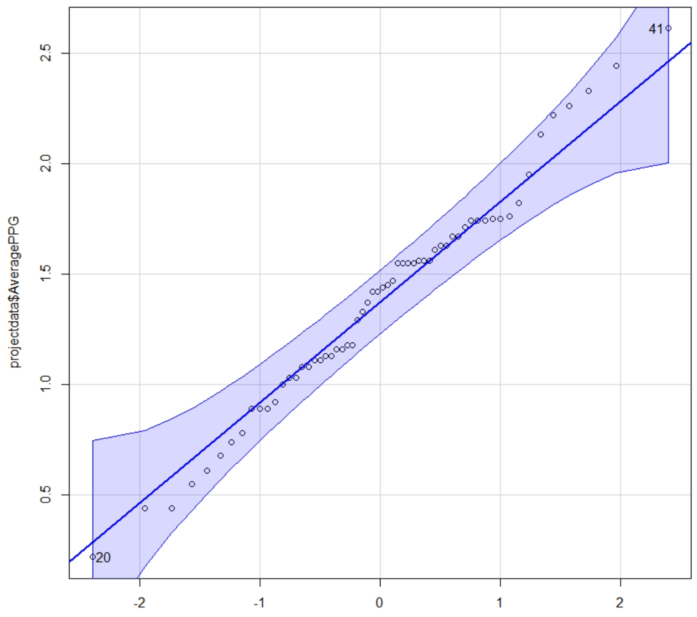
Introduction: By looking at a dataset composed of various sets of advanced analytics, I hope to see if there is a connection between performing well/winning games and various offensive,defensive, and external statistics.
Methodology: Using R, as well as various packages like SignifReg, and MASS, I created a model using forward selection based on an adjusted r-squared criteria. Using a box-cox transformation allowed me to view my ideal model, and allowed me to analyze the results.
Data Analysis: Using the model we created, we are able to see a strong influence of a team's ability to win games via the amount of shots they are able to get on target, their position on the ninth matchday, and where they rank in regards to the amount of goals they were expected to have allowed. We can also see a prediction of the winner of the current season, and compare the prediction of the past two seasons to the actual table.
Conclusion: Although our model does not span over that long a period, the model gives a good idea of what statistics influence a team's ability to win games, as well as provide me confidence in its ability to predict the top performing teams.

Introduction

Soccer has always grabbed my interest, from a young child to even today. It has always been something I had wanted to get myself involved in. With the rise of advanced analytics being used to recruit players, establish plays and strategies, and using this data to push teams to their best, I found it easy to base my project on the beautiful game. Naturally, I had to think about how to use a specific data set to come up with a goal. Eventually, I had decided to try and find a relationship between a specific set of advanced analytics ranging from the defensive and offensive end, and the ability of a club to win games or ultimately win the league. To do this, I plan on creating a model using a dataset I made using various websites that track various offensive and defensive stats over the course of the current season and the past two seasons, as well as some external pieces of data that may or may not have an impact on a club's performance. From there, I plan on using said model to create a prediction of the league table, along with the projected points that a team will earn, as well as compare the predictions of the past seasons with the actual results of the season. As mentioned, I plan on gathering data from the 20 clubs in the Premier League for the 2021-22 season, as well as the 2020-21 season and the 2019-20 season. The response variable, or the thing I want to try and find a relationship for and predict, would be **PPG** (points per game), a measure of how many points a team won per game on average. I figured this would be ideal as winning is indicative of a team's performance. As far as predictors go, I had quite a few. Included in my dataset was **Points**, how many points a team has earned so far in a season, **xG**, or expected goals, which is the amount of goals a team should have scored per game when taking into account the type of goal that was attempted. This is based off distance of goal, body part taken to shoot the attempt, whether or not there was an assist, etc. How many **Goals** a team has scored so far in a season, or their total tally for a season. I included **xPTS (expected points)**, which is how many points a team should have earned if the match was replayed several times. I also included a team's **shots per game**, which is how many attempts were taken, as well as **shots on target per game**, which is how many attempts were taken that were going towards the goal and required a save to be made. I had also decided to include **goals allowed,** which is how many goals a team allowed as well as **expected goals allowed**, which works like expected goals, but inversely in the sense that we are counting the expected goals for the enemy team. As I mentioned, I added other factors as predictors, such as the **position of the club on the league table on the ninth matchday** (It was the most recent data set available is the reason for the ninth matchday), whether or not they had **won the league** that season (denoted by 1s and 0s), and whether or not they were **participants in another European competition** hosted by UEFA, namely the Champions league, Europa league, and now the Conference League. Once I had collected all this data and put it into a .csv eligible for me to use, I was ready to attempt to create a model that would enable me to do what I set out to do.

Methodology:

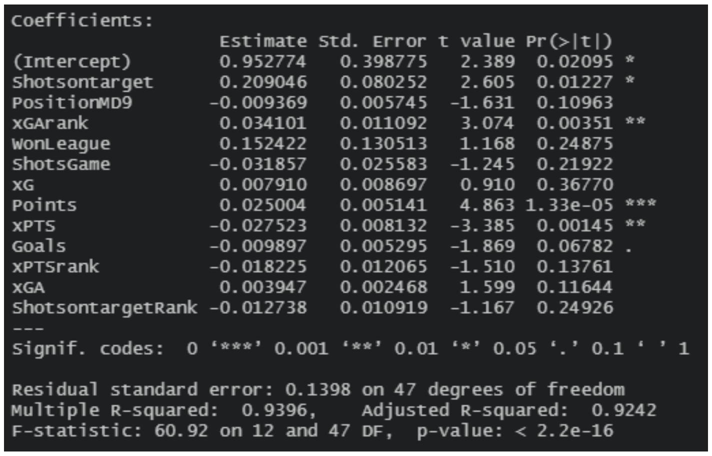
In order to create the model I wanted to create, I turned towards R Studio, a language that allows me to create models with ease compared to other languages. Additionally, I had to install some packages that would assist me in the creation of the model (namely SignifReg, car, caret, VIF, lme4, and MASS.) To start, I had to check to see if my data was normally distributed by using the qqp function in R, to create a plot with a dotted line denoting a prefect distribution and two lines denoting a standard deviation that looks as such:



After this, I created a model containing all of my predictors (Points, xPTS, xPTSrank, Goals, xG, xGrank, ShotsGame, ShotGamerank, Shotsontarget, ShotsontargetRank, Goalsallowed, xGA, xGArank, PositionMD9, WonLeague, and UEFA) as well as my response variable, denoted AveragePPG. After this, I created a "null model" containing no predictors and only my response variable AveragePPG. Using these two models, I was able to create a span in which I was going to use within the SignifReg function, where the span's range was all the predictors available in the dataset. Once I did this, I was able to use the SignifReg function to create two models. In one model, I based it off the null model, and using the "forward" direction that needs to be specified in the function, it built a model based on a specified criteria (in this case we want to maximize the adjusted r-squared of the model) essentially from scratch, as it starts off as the null model and works from the ground up to create a model with the highest adjusted r-squared. The second model built was based off of our full model with the intent of finding the highest adjusted r-squared again. This time however, we used the "backward" direction, in an attempt to "break down" our model from the original full model to create a model with the highest adjusted r-squared possible. Additionally, when choosing my model and running diagnostics, I had to perform a box-cox transformation on my model, in order to normalize my data again by raising my response variable to a λ power. Once doing this, I was able to analyze and drive a conclusion from my data.

<u>Data Analysis</u>

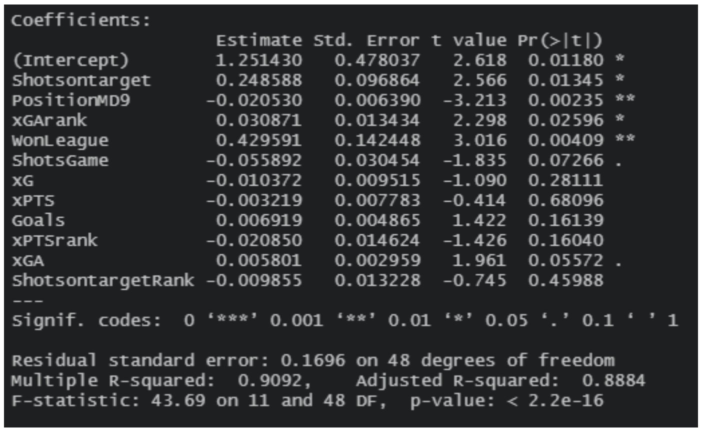
When creating the forward select model with SignifReg, we are met with a model that looks like the following one:



```
Coefficients:
                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)        0.952774   0.398775   2.389  0.02095 *  
Shotsontarget      0.209046   0.080252   2.605  0.01227 *  
PositionMD9       -0.009369   0.005745  -1.631  0.10963    
xGArank            0.034101   0.011092   3.074  0.00351 ** 
WonLeague          0.152422   0.130513   1.168  0.24875    
ShotsGame         -0.031857   0.025583  -1.245  0.21922    
xG                 0.007910   0.008697   0.910  0.36770    
Points             0.025004   0.005141   4.863 1.33e-05 ***
xPTS              -0.027523   0.008132  -3.385  0.00145 ** 
Goals             -0.009897   0.005295  -1.869  0.06782 .  
xPTSrank          -0.018225   0.012065  -1.510  0.13761    
xGA                0.003947   0.002468   1.599  0.11644    
ShotsontargetRank -0.012738   0.010919  -1.167  0.24926    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1398 on 47 degrees of freedom
Multiple R-squared:  0.9396,	Adjusted R-squared:  0.9242 
F-statistic: 60.92 on 12 and 47 DF,  p-value: < 2.2e-16
```

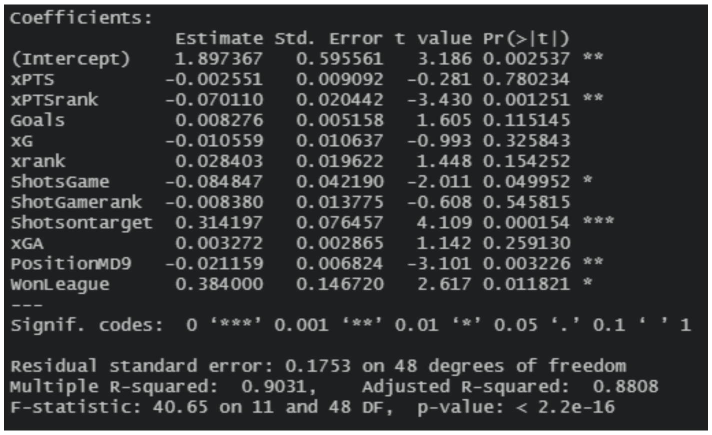
We find that this model has an adjusted r-squared of .9242, which means that 92.42% of the variability of our model can be explained by the predictor variables, which would prove to be very promising for the model! However, upon first glance I noticed that the Points predictor seemed to have much more influence over the other, which is when I acknowledged that the points a team earns and the average points a team earns per game go hand in hand. Once this was realized, I knew I had to remove this predictor to produce the best model possible, despite the probable decrease in the adjusted r-squared of our model. Once doing this, the model turned into the following:



```
Coefficients:
                   Estimate Std. Error t value Pr(>|t|)   
(Intercept)        1.251430   0.478037   2.618  0.01180 * 
Shotsontarget      0.248588   0.096864   2.566  0.01345 * 
PositionMD9       -0.020530   0.006390  -3.213  0.00235 **
xGArank            0.030871   0.013434   2.298  0.02596 * 
WonLeague          0.429591   0.142448   3.016  0.00409 **
ShotsGame         -0.055892   0.030454  -1.835  0.07266 . 
xG                -0.010372   0.009515  -1.090  0.28111   
xPTS              -0.003219   0.007783  -0.414  0.68096   
Goals              0.006919   0.004865   1.422  0.16139   
xPTSrank          -0.020850   0.014624  -1.426  0.16040   
xGA                0.005801   0.002959   1.961  0.05572 . 
ShotsontargetRank -0.009855   0.013228  -0.745  0.45988   
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1696 on 48 degrees of freedom
Multiple R-squared:  0.9092,	Adjusted R-squared:  0.8884 
F-statistic: 43.69 on 11 and 48 DF,  p-value: < 2.2e-16
```

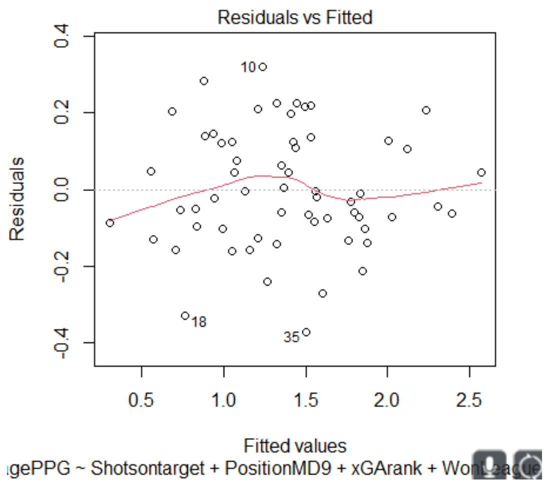
Immediately we see that the model loses an adjusted r-squared of .0358, or 3.58%. From here, I turned towards analyzing the backwards select model. In doing so, I encountered the same issue as the forward model with the Points predictor, so I had to remove that variable again and received the following model:

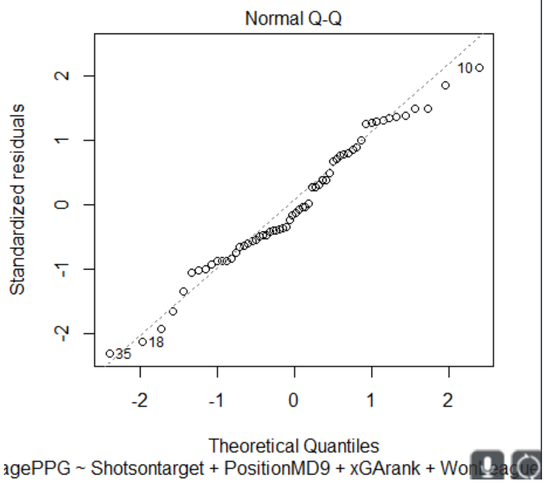


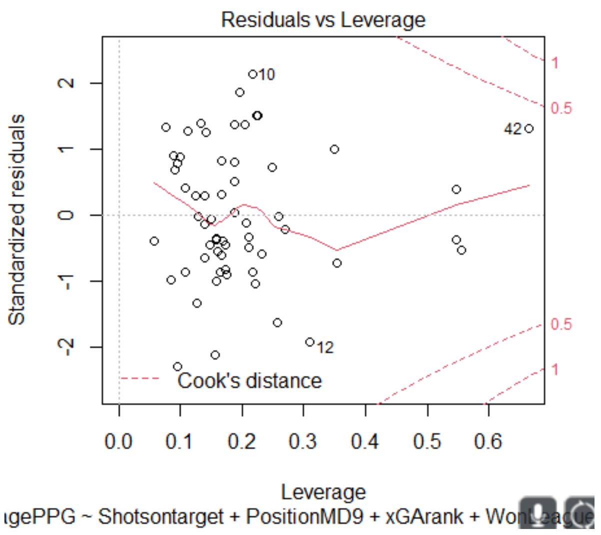
```
Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)     1.897367   0.595561   3.186 0.002537 ** 
xPTS           -0.002551   0.009092  -0.281 0.780234    
xPTSrank       -0.070110   0.020442  -3.430 0.001251 ** 
Goals           0.008276   0.005158   1.605 0.115145    
xG             -0.010559   0.010637  -0.993 0.325843    
xrank           0.028403   0.019622   1.448 0.154252    
ShotsGame      -0.084847   0.042190  -2.011 0.049952 *  
ShotGamerank   -0.008380   0.013775  -0.608 0.545815    
Shotsontarget   0.314197   0.076457   4.109 0.000154 ***
xGA             0.003272   0.002865   1.142 0.259130    
PositionMD9    -0.021159   0.006824  -3.101 0.003226 ** 
WonLeague       0.384000   0.146720   2.617 0.011821 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1753 on 48 degrees of freedom
Multiple R-squared:  0.9031,	Adjusted R-squared:  0.8808 
F-statistic: 40.65 on 11 and 48 DF,  p-value: < 2.2e-16
```

Although this model has a similar adjusted r-squared value, it is still smaller than the model created using forward selection, and as a result, the model I decided to work with was based off of the forward select model. Since I had decided on a model to base my diagnostics on, using the plot() function in R, I was able to look at residuals, the distribution of our residuals, as well as any potential leverage points in our model.

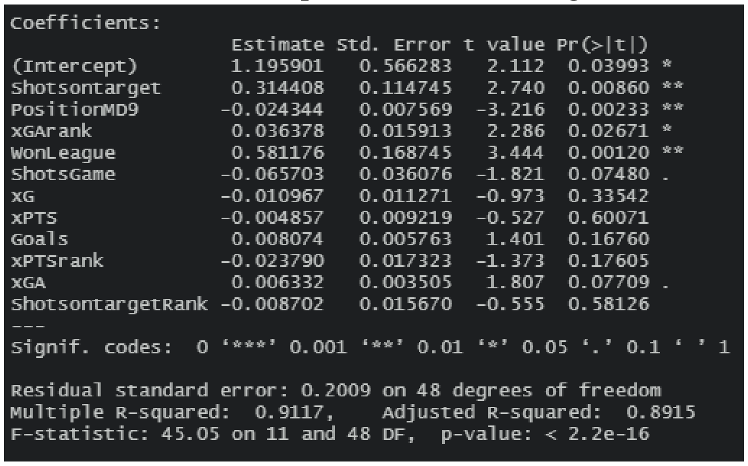






When looking at these diagnostics, we can immediately notice that our residuals showing some signs of heteroscedascity, which is not ideal in regards to our model. Additionally, when looking at the Q-Q plot, personally I found that it did not look good enough in the sense that some points seemed to be perhaps outside of that standard deviation I mentioned earlier when talking about the normality of my data. Fortunately, when looking at the residuals vs. Leverage plot, we find that no points have a leverage of 1, or even 0.5, which means none of our data points could be considered points of concern, and will not require any further action done to them.

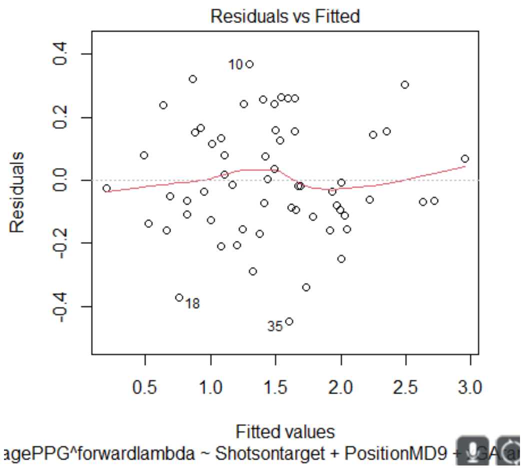
Unfortunately, these diagnostics did not look as optimal as I would have like, especially the distribution of my residuals in regards to the normal distribution I expected. As a result of this, I decided to enforce a box-cox transformation onto my model by using the boxcox function in R provided by the MASS package. By using the boxcox function, it meant I had to find the optimal lambda value for me to raise my response variable AveragePPG to that specific power. By using which.max in R, I was able to find out that the lambda value I would be using was 1.15151515… By raising AveragePPG to the 1.15151515 power, I was able to get the following model.

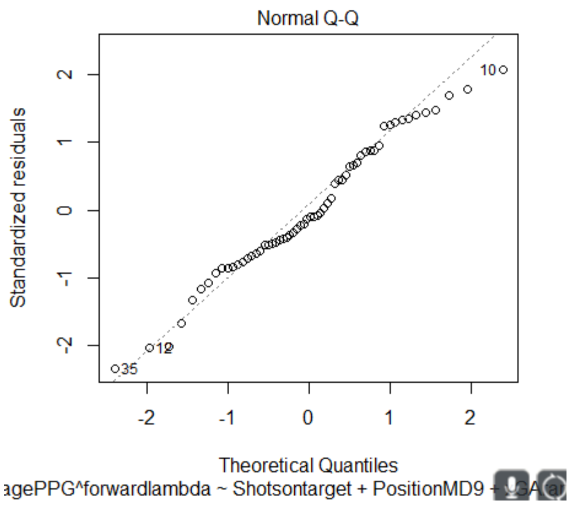


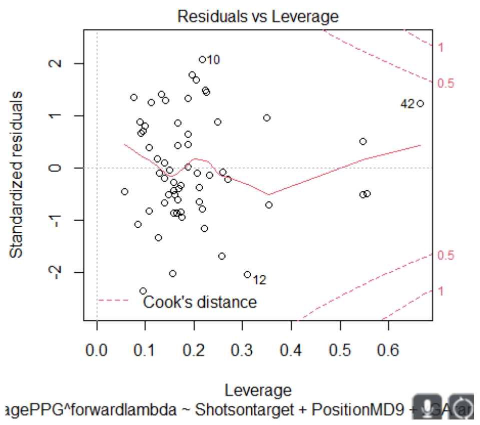
```
Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        1.195901   0.566283   2.112  0.03993 *
Shotsontarget      0.314408   0.114745   2.740  0.00860 **
PositionMD9       -0.024344   0.007569  -3.216  0.00233 **
xGArank            0.036378   0.015913   2.286  0.02671 *
WonLeague          0.581176   0.168745   3.444  0.00120 **
ShotsGame         -0.065703   0.036076  -1.821  0.07480 .
xG                -0.010967   0.011271  -0.973  0.33542
xPTS              -0.004857   0.009219  -0.527  0.60071
Goals              0.008074   0.005763   1.401  0.16760
xPTSrank          -0.023790   0.017323  -1.373  0.17605
xGA                0.006332   0.003505   1.807  0.07709 .
ShotsontargetRank -0.008702   0.015670  -0.555  0.58126
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2009 on 48 degrees of freedom
Multiple R-squared:  0.9117,    Adjusted R-squared:  0.8915
F-statistic: 45.05 on 11 and 48 DF,  p-value: < 2.2e-16
```

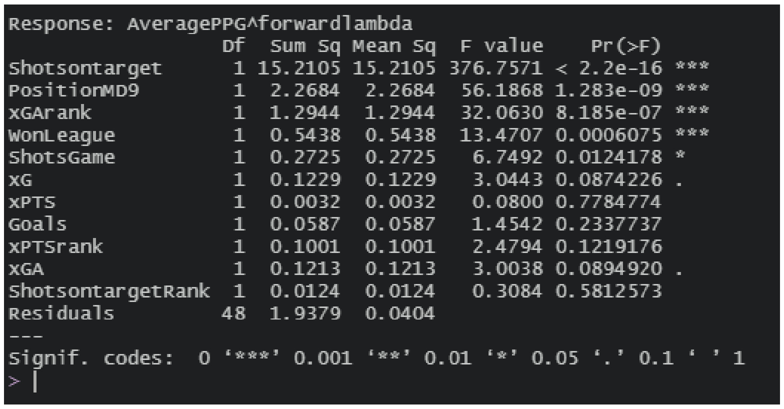
With this transformation, we found that our adjusted r-squared went up very slightly, which was still greatly appreciated. When running diagnostics on this newly transformed model, we received the following plots:







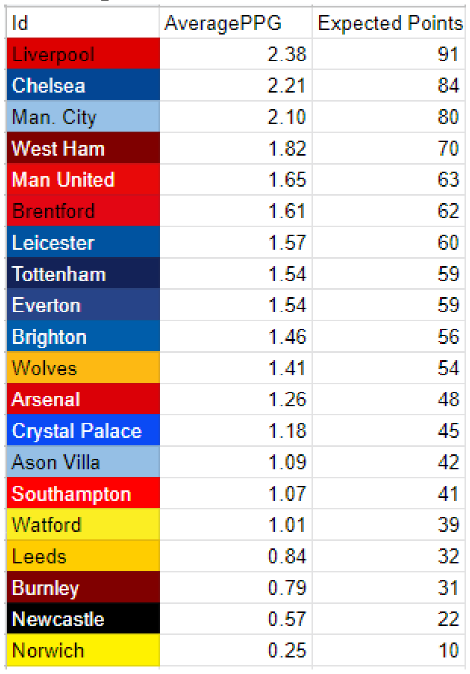
When looking at this set of diagnostic plots, I was much more satisfied. The residuals were showing signs of homoscedascity, which is ideal as this is a major assumption of the model heading into the diagnostics. Additionally, the Q-Q plot looks much more tightly bound to that center line denoting the perfect distribution that I mentioned at the beginning of the report. And thankfully this transformation did not cause any points to go beyond that leverage point of 0.5 still, meaning we do not have to consider any points a point of interest and potentially remove that point from our set of data for being a bad influential point. With the model being fully selected, and me being fully satisfied with it, I am now able to run the anova function in R to see what factors had the largest influence on a team's expected points per game. Doing so gives us an output that looks like the following:



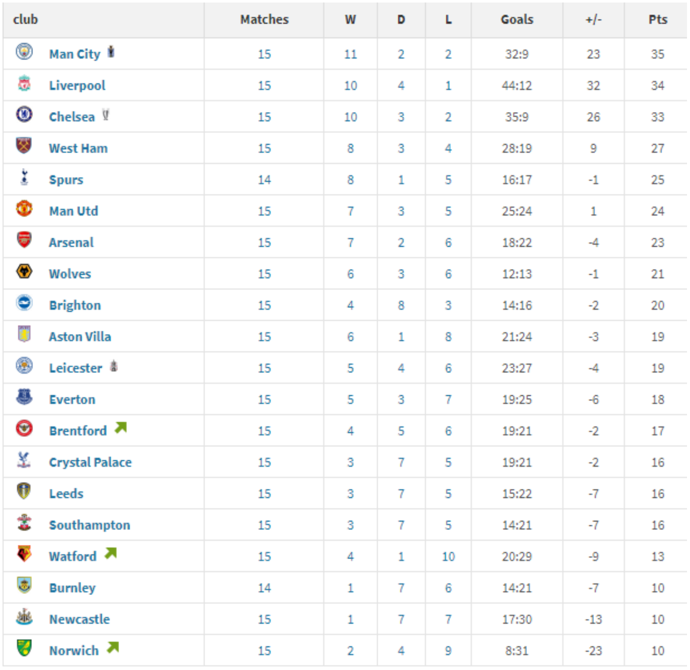
```
Response: AveragePPG^forwardlambda
                 Df  Sum Sq Mean Sq  F value    Pr(>F)    
Shotsontarget     1 15.2105 15.2105 376.7571 < 2.2e-16 ***
PositionMD9       1  2.2684  2.2684  56.1868 1.283e-09 ***
xGArank           1  1.2944  1.2944  32.0630 8.185e-07 ***
WonLeague         1  0.5438  0.5438  13.4707 0.0006075 ***
ShotsGame         1  0.2725  0.2725   6.7492 0.0124178 *  
xG                1  0.1229  0.1229   3.0443 0.0874226 .  
xPTS              1  0.0032  0.0032   0.0800 0.7784774    
Goals             1  0.0587  0.0587   1.4542 0.2337737    
xPTSrank          1  0.1001  0.1001   2.4794 0.1219176    
xGA               1  0.1213  0.1213   3.0038 0.0894920 .  
ShotsontargetRank  1  0.0124  0.0124   0.3084 0.5812573    
Residuals        48  1.9379  0.0404                      
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> |
```

Looking at this, we can see that the largest influence on our response variable AveragePPG is the ability for a team to get shots on target. This makes sense as it is sign of a better offense and at the end of the day, scoring goals wins games. However, it still stuck out as a surprise to me because I had come into the project thinking that xG and perhaps xGA would be the major contributors. Instead both do not hold much weight to Shots on target, and where they rank in expected goals allowed. Luckily, I did assume that a team's position on the ninth matchweek of the season had a fairly good influence, as starting off well with a large enough sample size (8 or 9 games in this case) means that a team is most likely well off for the rest of the season. Additionally, I was surprised to see Goals sit so low on the list. With this model, I am able to determine that a teams ability to perform and win games is very highly aligned with some of the predictors in our model. Namely, the ability to win games is greatly influenced by a team's ability to get shots on target, how well they start the season, and where they rank in the amount of expected goals they have let through.

Using this model, I was able to predict the outcome of this current season of the premier league, and the prediction resulted in a table that looked like the image on the left.

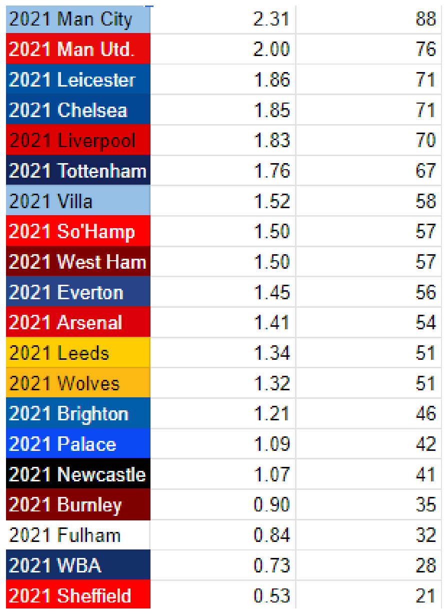


Id	AveragePPG	Expected Points
Liverpool	2.38	91
Chelsea	2.21	84
Man. City	2.10	80
West Ham	1.82	70
Man United	1.65	63
Brentford	1.61	62
Leicester	1.57	60
Tottenham	1.54	59
Everton	1.54	59
Brighton	1.46	56
Wolves	1.41	54
Arsenal	1.26	48
Crystal Palace	1.18	45
Ason Villa	1.09	42
Southampton	1.07	41
Watford	1.01	39
Leeds	0.84	32
Burnley	0.79	31
Newcastle	0.57	22
Norwich	0.25	10

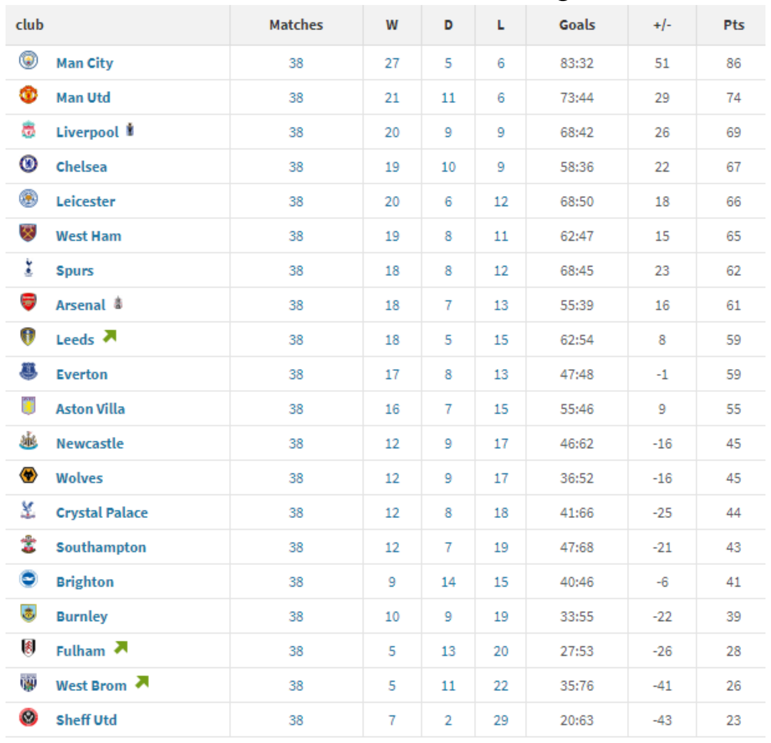


club	Matches	W	D	L	Goals	+/-	Pts
Man City	15	11	2	2	32:9	23	35
Liverpool	15	10	4	1	44:12	32	34
Chelsea	15	10	3	2	35:9	26	33
West Ham	15	8	3	4	28:19	9	27
Spurs	14	8	1	5	16:17	-1	25
Man Utd	15	7	3	5	25:24	1	24
Arsenal	15	7	2	6	18:22	-4	23
Wolves	15	6	3	6	12:13	-1	21
Brighton	15	4	8	3	14:16	-2	20
Aston Villa	15	6	1	8	21:24	-3	19
Leicester	15	5	4	6	23:27	-4	19
Everton	15	5	3	7	19:25	-6	18
Brentford	15	4	5	6	19:21	-2	17
Crystal Palace	15	3	7	5	19:21	-2	16
Leeds	15	3	7	5	15:22	-7	16
Southampton	15	3	7	5	14:21	-7	16
Watford	15	4	1	10	20:29	-9	13
Burnley	14	1	7	6	14:21	-7	10
Newcastle	15	1	7	7	17:30	-13	10
Norwich	15	2	4	9	8:31	-23	10

When comparing the predictions made on the left (Based on match week 9) and compare it to the current table (match week 15) we see that some clubs are currently underperforming, namely Brentford, and some are overperforming, namely Arsenal, when looking at the predicted placement on the tables. When looking at what clubs are doing well, we can see that it is very similar to the actual results, which is a good sign for the model. However, because we have data from the past 2 seasons and we want to find confidence in this model, we can compare the predictions made from the past 2 seasons with the actual results! The following can be seen:

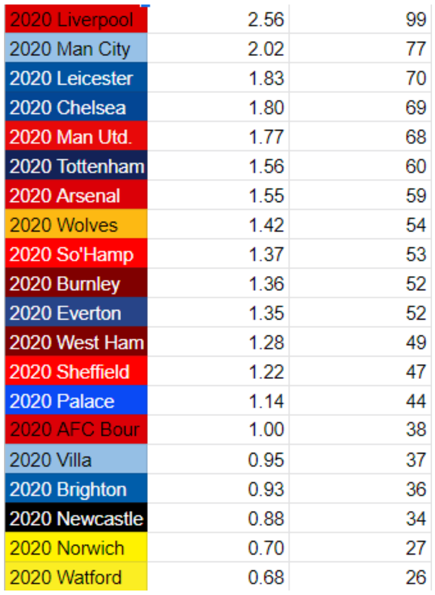


2021 Man City	2.31	88
2021 Man Utd.	2.00	76
2021 Leicester	1.86	71
2021 Chelsea	1.85	71
2021 Liverpool	1.83	70
2021 Tottenham	1.76	67
2021 Villa	1.52	58
2021 So'Hamp	1.50	57
2021 West Ham	1.50	57
2021 Everton	1.45	56
2021 Arsenal	1.41	54
2021 Leeds	1.34	51
2021 Wolves	1.32	51
2021 Brighton	1.21	46
2021 Palace	1.09	42
2021 Newcastle	1.07	41
2021 Burnley	0.90	35
2021 Fulham	0.84	32
2021 WBA	0.73	28
2021 Sheffield	0.53	21

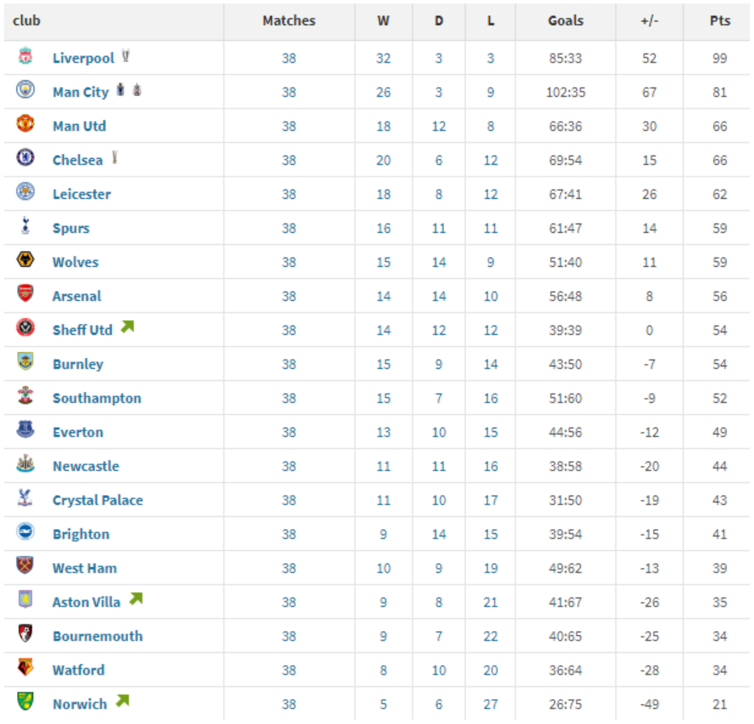


club	Matches	W	D	L	Goals	+/-	Pts
Man City	38	27	5	6	83:32	51	86
Man Utd	38	21	11	6	73:44	29	74
Liverpool	38	20	9	9	68:42	26	69
Chelsea	38	19	10	9	58:36	22	67
Leicester	38	20	6	12	68:50	18	66
West Ham	38	19	8	11	62:47	15	65
Spurs	38	18	8	12	68:45	23	62
Arsenal	38	18	7	13	55:39	16	61
Leeds	38	18	5	15	62:54	8	59
Everton	38	17	8	13	47:48	-1	59
Aston Villa	38	16	7	15	55:46	9	55
Newcastle	38	12	9	17	46:62	-16	45
Wolves	38	12	9	17	36:52	-16	45
Crystal Palace	38	12	8	18	41:66	-25	44
Southampton	38	12	7	19	47:68	-21	43
Brighton	38	9	14	15	40:46	-6	41
Burnley	38	10	9	19	33:55	-22	39
Fulham	38	5	13	20	27:53	-26	28
West Brom	38	5	11	22	35:76	-41	26
Sheff Utd	38	7	2	29	20:63	-43	23

And



2020 Liverpool	2.56	99
2020 Man City	2.02	77
2020 Leicester	1.83	70
2020 Chelsea	1.80	69
2020 Man Utd.	1.77	68
2020 Tottenham	1.56	60
2020 Arsenal	1.55	59
2020 Wolves	1.42	54
2020 So'Hamp	1.37	53
2020 Burnley	1.36	52
2020 Everton	1.35	52
2020 West Ham	1.28	49
2020 Sheffield	1.22	47
2020 Palace	1.14	44
2020 AFC Bour	1.00	38
2020 Villa	0.95	37
2020 Brighton	0.93	36
2020 Newcastle	0.88	34
2020 Norwich	0.70	27
2020 Watford	0.68	26



club	Matches	W	D	L	Goals	+/-	Pts
Liverpool	38	32	3	3	85:33	52	99
Man City	38	26	3	9	102:35	67	81
Man Utd	38	18	12	8	66:36	30	66
Chelsea	38	20	6	12	69:54	15	66
Leicester	38	18	8	12	67:41	26	62
Spurs	38	16	11	11	61:47	14	59
Wolves	38	15	14	9	51:40	11	59
Arsenal	38	14	14	10	56:48	8	56
Sheff Utd	38	14	12	12	39:39	0	54
Burnley	38	15	9	14	43:50	-7	54
Southampton	38	15	7	16	51:60	-9	52
Everton	38	13	10	15	44:56	-12	49
Newcastle	38	11	11	16	38:58	-20	44
Crystal Palace	38	11	10	17	31:50	-19	43
Brighton	38	9	14	15	39:54	-15	41
West Ham	38	10	9	19	49:62	-13	39
Aston Villa	38	9	8	21	41:67	-26	35
Bournemouth	38	9	7	22	40:65	-25	34
Watford	38	8	10	20	36:64	-28	34
Norwich	38	5	6	27	26:75	-49	21

We find that the predictions were accurate in terms of predicting whoever would win the league, as well as accurate in predicting the points allocated by the top 6 placing clubs, and the points allocated by the clubs facing relegation (the bottom 3 of each respective season.) Seeing this put confidence in my model, despite the sample size being too small in terms of seasons the predictions were accurate in. Regardless, one of the goals of the model was to potentially predict the winner of the season, and despite the end of the season being in May, it is nice to know that the model was somewhat accurate for the past 2 seasons.

Conclusion

Although the sample size was small, and it was nothing to be overly confident about, I was still incredibly satisfied with my model. It did what I set out to do and that was to find a relationship between winning (which we were able to find using the Anova table), and potentially predict the winner of the premier league (which we were also able to do, hopefully to the same accuracy as the last two seasons models.) However, that is not to say that I would not do things differently if I attempted to create this model again. For starters, I would definitely increase the span of seasons until the introduction of the advanced stats like xG, xGA, among others, as those were only tracked as recently as 2016. This is so that we could see if the model is "accurate" beyond 2 or potentially 3 seasons. Additionally, when creating the model, I would have liked to add some variance in my predictors using the lme4 or lmer function in R. I tried to do this initially, but I kept getting errors that resulted in me directly going in and altering the function, but when I did that it gave me separate errors denoting that the matrix was not able to be found, despite me changing the status of my variables to nested or crossed and adjusting the code accordingly. Regardless, variability in my predictors would have enabled me to create a mixed model, which would account for potential dependency within my model, as there are clubs that show up repeatedly. Lastly, I would have liked to find more predictors for my model. I had around 20 predictors, but almost half of them did not end up being used in the final model. By finding more predictors, hopefully more accuracy could have been found within my model. Despite this, I had a great time creating the model and putting my hypotheses to life, and am happy that I was able to create a model that provided me with confidence in its ability to determine which statistics benefit a team the most as well as predict the winner of the league.

Appendix

Code Used:

```
#Set Directory for csvs
setwd("C:/Users/josie/OneDrive/Documents/444")
#Install Necessary Packages
library(SignifReg)
library(car)
library(caret)
library(VIF)
library(lme4)
library(MASS)
#Read our data
projectdata = read.csv("PremierLeagueData.csv", header = TRUE)
#Check for normality within our data
projectdata$AveragePPG.t <- projectdata$AveragePPG + 1
qqp(projectdata$AveragePPG, "norm")
#Full model
projectmodelfull <- lm(AveragePPG ~ Points + xPTS + xPTSrank
                       + Goals + xG + xrank + ShotsGame
                       + ShotGamerank + Shotsontarget
                       + ShotsontargetRank + Goalsallowed
                       + xGA + xGArank + PositionMD9 + WonLeague
                       + UEFA, data =  projectdata)
#Null model
projectmodelnull <- lm(AveragePPG ~ 1, data = projectdata)
#Create a scope for SignifReg package to use
scope = list(lower=formula(projectmodelnull), upper = formula(projectmodelfull))
#Create a forward select r-adj model using SignifReg based on null model
forwardselectmodel = SignifReg(projectmodelnull, scope = scope, direction = "forward",
trace = FALSE, criterion = "r-adj", alpha = 1)

summary(forwardselectmodel)
#Removing the PTS predictor
forwardselectmodelreduced <- lm(AveragePPG ~ Shotsontarget
                                + PositionMD9 + xGArank
                                + WonLeague + ShotsGame + xG
                                 + xPTS + Goals + xPTSrank
                                + xGA + ShotsontargetRank, data = projectdata)
#Create a backward select r-adj model using SignifReg based on full model
backwardselectmodel = SignifReg(projectmodelfull, scope = scope, direction =
"backward", trace = FALSE, criterion = "r-adj", alpha = 1)
#Removing the PTS predictor
backwardselectmodelreduced <- lm(AveragePPG ~ xPTS + xPTSrank
                                 + Goals + xG + xrank + ShotsGame
                                 + ShotGamerank + Shotsontarget + xGA
                                 + PositionMD9 + WonLeague, data = projectdata)
#Doing boxcox to fix the residuals / normality of our model
bc <- boxcox(AveragePPG ~ Shotsontarget
             + PositionMD9 + xGArank
             + WonLeague + ShotsGame + xG
             + xPTS + Goals + xPTSrank
             + xGA + ShotsontargetRank, data = projectdata)
(forwardlambda <- bc$x[which.max(bc$y)])
```

```
bcbackward <- boxcox(AveragePPG ~ xPTS + xPTSrank
                     + Goals + xG + xrank + ShotsGame
                     + ShotGamerank + Shotsontarget + xGA
                     + PositionMD9 + WonLeague, data = projectdata)

(backwardlambda <- bcbackward$x[which.max(bc$y)])

boxcoxforwardselectreducedmodel <- lm(AveragePPG^forwardlambda ~ Shotsontarget
                         + PositionMD9 + xGArank
                         + WonLeague + ShotsGame + xG
                         + xPTS + Goals + xPTSrank
                         + xGA + ShotsontargetRank, data = projectdata)

boxcoxbackwardselectreducedmodel <- lm(AveragePPG^backwardlambda ~ xPTS + xPTSrank
                                       + Goals + xG + xrank + ShotsGame
                                       + ShotGamerank + Shotsontarget + xGA
                                       + PositionMD9 + WonLeague, data = projectdata)
#Running diagnostics on the model we chose
plot(boxcoxforwardselectreducedmodel)
#Checking for normality of residuals
qqp(boxcoxforwardselectreducedmodel,"norm")
#Creating the prediction of the results of the league
y_hat = as.numeric(predict(boxcoxforwardselectreducedmodel, projectdata))
y_hat = y_hat^(1/lambda)
output = data.frame(Id = projectdata$Team.Name, AveragePPG = y_hat)
write.csv(output, file = "project444.csv", row.names = FALSE)
```