Brennan Gillgrist

Dr. Hitchcock

Bayesian Statistics: STAT 535

April 7, 2024

Bayesian Statistics Final Project Report

Table of Contents

Introduction:

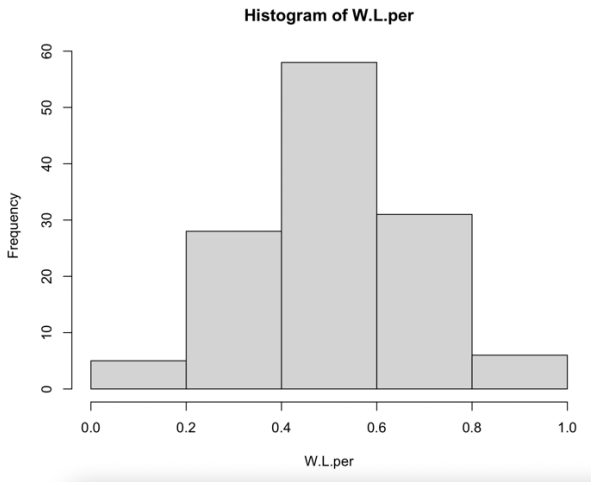
In this Bayesian Statistics final project, I will explore the correlation between the salary cap and the win percentage of teams in the NFL. Predicting how well a team will perform in an upcoming season is an important stat to track and while many teams, media agencies, or betting sites use many performance metrics to track this, there is a lot less research into how salary cap is managed and its relation to team success. The data set I will be analyzing predicts team win percentage for a season based on previous season win percentage and the percent of the salary cap the quarterbacks, running backs, wide receivers, offensive linemen, and defense takes up. An example of this data set is provided below:

Season	Team	W	L	W.L.per	Previous.W.L.per	Cap.percent.QB	Cap.percent.RB	Cap.percent.WR	Cap.percent.OL	Cap.percent.Def
2023	Buffalo Bills	11	6	0.647	0.813	0.0866	0.0134	0.1081	0.1524	0.3450
2023	Miami Dolphins	11	6	0.647	0.529	0.0590	0.0305	0.1403	0.1076	0.3402
2023	New York Jets	7	10	0.412	0.412	0.0416	0.0128	0.0563	0.0977	0.3331
2023	New England Patriots	4	13	0.235	0.471	0.0236	0.0159	0.0396	0.1170	0.2836
2023	Baltimore Ravens	13	4	0.765	0.588	0.1151	0.0199	0.0615	0.2166	0.3059
2023	Cleveland Browns	11	6	0.647	0.412	0.0109	0.0122	0.1173	0.0788	0.2473
2023	Pittsburgh Steelers	10	7	0.588	0.529	0.0443	0.0233	0.0993	0.1875	0.4320

Each salary cap percentage is calculated as the sum of all the salaries of players in that position group divided by the total salary cap percentage at the start of the season. For example, in the provided data above, the Buffalo Bills in 2023 allocated 8.66% of their salary cap to QBs including: Josh Allen ($18,636,281 or 8.18% of the salary cap) and Kyle Allen ($1,092,500 or 0.48% of the salary cap). This was calculated for every team back to 2020 (128 total teams). All the salary cap data was sourced from spotrac.com. The goal here will be to fit a model predicting the continuous variable W/L % from the 6 predictor variables.

Creating the Model:

1. What type of regression should I choose?
    a. My response variable has continuous support between 0 and 1 which led me to think that this would require some form of Beta regression. However, When looking at a histogram of my response data, it looks like we can treat this data as normal and continue with normal Bayesian regression.



I only gave this histogram 5 breaks because of the relatively small dataset but this does look like it would follow a normal distribution. I will continue using Normal Bayesian Regression.

2. Prior Choices:
    a. Intercept: Mean 0, Variance .15
        i. I chose a mostly uninformative prior here for the intercept however since I know that the response variable falls between 0 and 1 so I will make my variance smaller so that I ensure the model keeps smaller values.
    b. Previous Season W/L %: Mean .5, Variance 0.1
        i. I know that there is a positive relationship between previous season W/L % and upcoming season W/L% so I will choose a positive mean and a variance that keeps this beta small.
    c. Positional Salary Cap: Mean 0, Variance .2
        i. I know that these betas should be small and kept to lower values however I do not know if there is a positive or negative relationship with these variables so I set the mean to be 0.

3. Creating the Model / Diagnostics

Original Model Code

```
Estimates:
                   mean    sd      10%     50%     90%
(Intercept)      -0.1421  0.0631 -0.2217 -0.1429 -0.0612
Previous.W.L.per  0.3385  0.0558  0.2677  0.3382  0.4108
Cap.percent.QB    0.8055  0.2264  0.5169  0.8065  1.0947
Cap.percent.RB    0.7941  0.6388 -0.0301  0.8009  1.6049
Cap.percent.WR    0.5600  0.3099  0.1628  0.5592  0.9586
Cap.percent.OL    0.8325  0.2629  0.4969  0.8313  1.1673
Cap.percent.Def   0.8089  0.1654  0.5963  0.8104  1.0213
sigma             0.1333  0.0086  0.1227  0.1328  0.1446
```

W/L% = -0.142 + (0.338*Previous W/L%) + (0.805*QBs) + (0.794*RBs) + (0.56*WRs) + (0.832*OL) + (0.809*Def)

All the mcmc diagnostics looked good with this model and I had an MAE of 0.0811 which means that I expect my prediction to be 8% away from the true W/L %. This is good however I would like to see if I can reduce the number of variables in the model so I will apply a Gibbs sampler model selection technique.

4. Model Selection

Just looking at the 90% intervals for the estimates, the only predictor variable that includes 0 is the RB cap percentage. This would imply that every variable plays a role in affecting win loss percentage besides the RB cap percentage however I will apply a more statistical approach in variable selection:

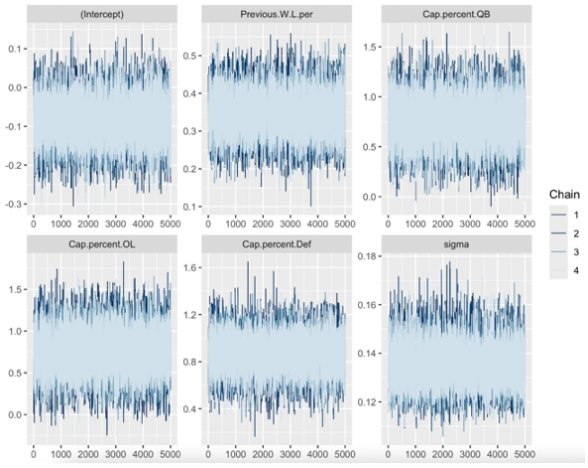
Variable Selection Code

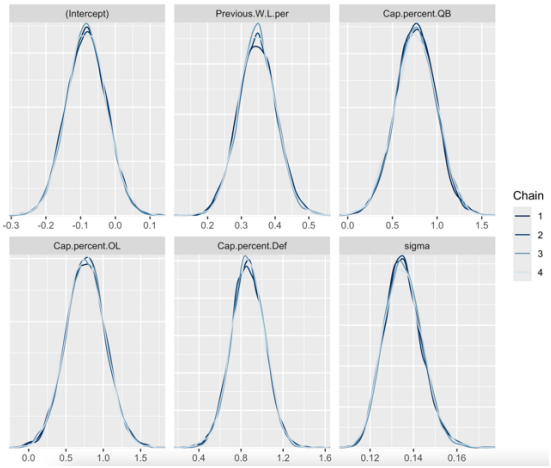
```
                     z.probs
[1,] 0 1 1 0 0 1 1  0.2804
[2,] 0 1 1 0 0 0 1  0.2660
[3,] 1 1 1 0 0 1 1  0.0761
[4,] 0 1 1 1 0 0 1  0.0584
[5,] 0 1 1 0 1 1 1  0.0467
```

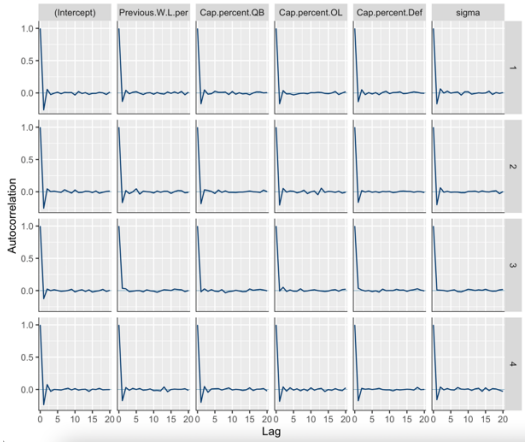
Here this shows us that the model with the highest probability would include no intercept, RB, or WR group percentages however I will continue to keep the intercept and choose a model that only does not include RB and WR.

5. The New Model / Diagnostics

Model and Diagnostics Code







Estimates:

	mean	sd	10%	50%	90%
(Intercept)	-0.0834	0.0576	-0.1574	-0.0833	-0.0097
Previous.W.L.per	0.3469	0.0561	0.2752	0.3467	0.4184
Cap.percent.QB	0.7615	0.2276	0.4711	0.7624	1.0510
Cap.percent.OL	0.7626	0.2625	0.4265	0.7632	1.0978
Cap.percent.Def	0.8666	0.1591	0.6646	0.8662	1.0695
sigma	0.1353	0.0088	0.1243	0.1349	0.1467

Out trace plots look very good indicating that the mcmc chain appropriately sampled the values. The distributions of the betas look good and we can see that none of them (besides the intercept however that is not a problem) have high probability at 0 implying that all these variables likely affect the response variable. There is not serious autocorrelation issues in the autocorrelation function plots. We are given an MAE of 0.088 meaning we expect our predictions to be 8.83% away from the true W/L %. This is now higher than the previous model but using less predictors

will outweigh the small cost here. In addition, we also see that the beta distributions show us which predictor variables affect the response variable the most since each predictor variable has the same support (0-1). We can see that spending big on defense helps predict season outcome more than any other variable and the previous season performance does the least.

Model Equation:

W/L% = -0.0834 + (0.3469*Previous W/L%) + (0.7615*QBs) + (0.7626*OL) + (0.8666*Def)

Conclusions:

While cap space alone is never going to fully predict win percentage for an upcoming season, this model suggest that the management of cap space does play a role in team performance in the upcoming season. A further advancement on this project that includes more preseason statistics, like QB experience, coach experience, etc., could be applied to make the model more accurate. I would be intrigued to see how these variables would play out and should I ever expand this project prior to the upcoming 2024 NFL season I will send you that model. I do want to test this model on the upcoming season and give a prediction for each team, however teams do not cut down to their 53-man rosters until late August. When this does happen, I will apply this model for the 2024 season and send you the results. I will apply the model to the current cap percentages (though we must be careful analyzing it right now) just to see how the numbers will play out.

TEAM	QB %	OL %	DEF %	Previous W/L %	Expected W/L %
Cleveland Browns	23.28%	17.03%	30.88%	0.647	0.716
Dallas Cowboys	24.43%	14.64%	30.23%	0.706	0.721
Denver Broncos	23.62%	17.87%	24.45%	0.471	0.608
Los Angeles Rams	21.13%	25.19%	24.74%	0.588	0.688
New York Giants	20.65%	20.29%	28.33%	0.353	0.597
Arizona Cardinals	18.97%	22.80%	33.02%	0.235	0.603
Kansas City Chiefs	16.47%	25.61%	28.96%	0.647	0.713
Las Vegas Raiders	14.62%	10.65%	32.15%	0.471	0.551
Detroit Lions	13.40%	21.31%	31.25%	0.706	0.697
Baltimore Ravens	13.58%	13.74%	42.12%	0.765	0.755
Buffalo Bills	13.01%	11.79%	39.79%	0.647	0.675
New York Jets	11.93%	10.52%	45.77%	0.412	0.627
Cincinnati Bengals	11.54%	17.70%	33.64%	0.529	0.614
Atlanta Falcons	11.36%	25.06%	43.06%	0.412	0.710
Miami Dolphins	11.45%	13.05%	37.65%	0.647	0.654
Seattle Seahawks	10.47%	7.43%	34.27%	0.529	0.533
Los Angeles Chargers	8.13%	10.11%	40.33%	0.294	0.507
New Orleans Saints	7.55%	14.21%	36.72%	0.529	0.584
Philadelphia Eagles	6.72%	20.50%	31.03%	0.647	0.617
Jacksonville Jaguars	6.29%	19.57%	27.40%	0.529	0.535
Houston Texans	6.14%	21.08%	34.41%	0.588	0.626
Green Bay Packers	5.48%	10.24%	52.54%	0.529	0.675
Indianapolis Colts	5.30%	26.53%	37.37%	0.529	0.667
Carolina Panthers	5.22%	27.51%	28.12%	0.118	0.451
Tampa Bay Buccaneers	3.88%	13.66%	45.13%	0.529	0.625
New England Patriots	3.68%	15.11%	38.55%	0.235	0.475
Minnesota Vikings	3.13%	16.25%	26.54%	0.412	0.437
Washington Commanders	3.01%	12.34%	40.34%	0.235	0.465
Tennessee Titans	2.42%	9.35%	39.29%	0.353	0.469
Pittsburgh Steelers	2.17%	15.91%	57.83%	0.588	0.760
San Francisco 49ers	1.61%	17.60%	46.56%	0.706	0.711
Chicago Bears	0.73%	13.97%	40.65%	0.412	0.524

The last column above displays the expected win percentage for each team in the upcoming season. One thing to note here is that only 5 teams are expected to have a win percentage below .500. This could be for many reasons however I believe it is due to the elevated roster sizes in the off season. Each cap space predictor is modeled as a linear line not a quadratic meaning almost

every time, more cap space allotted is better. This could also be due to a lack of variables however the focus of the project was not to perfectly predict win loss percentage but to explore the relationship between cap space and performance. I will continue to adjust this model after this class because I enjoyed the project and should I ever find new relationships, I will send them to you if I do.

Appendix

1. Original Model Code

```
> cap_model<-stan_glm(W.L.per ~ Previous.W.L.per + Cap.percent.QB + Cap.percent.RB +
Cap.percent.WR + Cap.percent.OL + Cap.percent.Def, data=cap, family=gaussian,
prior_intercept=normal(0,.15), prior=normal(c(.5,0,0,0,0,0), c(.1,.2,.2,.2,.2,.2),
autoscale=TRUE), prior_aux = exponential(1, autoscale=TRUE), chains=4, iter=5000*2)
> summary(cap_model, digits=4)
```

2. Variable Selection Code

```
> lpy.X <- function(y,X,g=length(y),nu0=1,s20=try(summary(lm(y~-
1+X))$sigma^2,silent=TRUE)) {
+  n<-dim(X)[1]; p<-dim(X)[2]
+  if(p==0) {Hg<-0; s20<-mean(y^2)}
+  if(p>0) {Hg<-(g/(g+1))*X%*%solve(t(X)%*%X)%*%t(X)}
+  SSRg<- t(y)%*%( diag(1,nrow=n) - Hg)%*%y
+
+  -.5*(n*log(pi)+p*log(1+g)+(nu0+n)*log(nu0*s20+SSRg)-
nu0*log(nu0*s20))+lgamma((nu0+n)/2)-lgamma(nu0/2)
+ }
```

The data:

```
> y<-W.L.per
> x1<-Previous.W.L.per
> x2<-Cap.percent.QB
> x3<-Cap.percent.RB
> x4<-Cap.percent.WR
> x5<-Cap.percent.OL
> x6<-Cap.percent.Def
> X<-as.matrix(cbind(rep(1, length(x1)), x1, x2, x3, x4, x5, x6))
```

Starting values for Gibbs Sampler:

```
> z<-rep(1,dim(X)[2])  # starting with z = all 1's (all terms in model)
> lpy.c<-lpy.X(y,X[,z==1,drop=FALSE])
> S <- 10000  # number of Monte Carlo iterations
> Z<-matrix(NA,S,dim(X)[2])
```

The Gibbs Sampler:

```
> for(s in 1:S)
+ {
+  for(j in sample(1:dim(X)[2]))
+  {
+  zp<-z; zp[j] <- 1-zp[j]
+  lpy.p<-lpy.X(y,X[,zp==1,drop=FALSE])
+  r<- (lpy.p - lpy.c)*(-1)^(zp[j]==0)
```

```
+  z[j]<-rbinom(1,1,1/(1+exp(-r)))
+  if(z[j]==zp[j]) {lpy.c<-lpy.p}
+  }
+ Z[s,]<-z
+ }
```

Considering all combinations (Only showing the top 5):

```
> poss.z.vectors <-  unique(Z,MARGIN=1)
> z.probs <- rep(0, times= nrow(poss.z.vectors))
>
> for(i in 1:nrow(poss.z.vectors)) {
+ z.probs[i] <- sum(apply(Z,1,identical, y=poss.z.vectors[i,]))
+ }
> z.probs <- z.probs/sum(z.probs)
>
> cbind(poss.z.vectors, z.probs)[rev(order(z.probs)),]
```

3. New Model Code

```
> cap_model_selected<-stan_glm(W.L.per ~ Previous.W.L.per + Cap.percent.QB +
Cap.percent.OL + Cap.percent.Def, data=cap, family=gaussian, prior_intercept = normal(0,.15),
prior=normal(c(.5,0,0,0), c(.1,.2,.2,.2), autoscale=TRUE), prior_aux = exponential(1,
autoscale=TRUE), chains=4, iter=5000*2)
> mcmc_trace(cap_model_selected)
> mcmc_dens_overlay(cap_model_selected)
> mcmc_acf(cap_model_selected)
> prediction_summary(cap_model_selected, data=cap)
> summary(cap_model_selected, digits=4)
```