College Football - Bayesian Statistics

Brian Cain

April 2022

1 Introduction

Prediction of college football games is an important task for sports bettors and sports analytic experts. A top model in industry is ESPN's FPI which attained 69.87% accuracy [1] for the 2021 season. Similar to other models, ESPN FPI's user-facing predictions take into account a variety of factors then output the probability of a team winning their scheduled game [2]. This win probability is useful but lacks robustness on the user-facing side. Other sports analytic developers don't often allow free access to their game predictions due to their advantages in the sports betting industry. In terms of public-facing models, the Catherwood Ratings model achieved the top results in 2021 with an accuracy of 77.027% and should be used as a benchmark for future models developed.

An important component of game prediction models is having the capability to measure uncertainty around a team's probability of winning. This project aims to utilize Bayesian statistics to model the parameters of a football game prediction model as unknown, resulting in a posterior distribution of win probabilities that accounts for greater uncertainty based off observed football data. This posterior distribution will also allow for further analysis to be done on potential net-winnings of placing bets on a specific set of games.

Specifically this project constructs a Bayesian Logistic Ridge regression to model the probability of a team winning a game. Seen later on, ridge regression is utilized out of necessity to combat col-linearity of predictive features. Due to lack of a conjugate prior relationship over the posterior distribution for the model parameters a Metropolis-Hastings algorithm (MH) is utilized the draw samples from the posterior distribution. A convergence study is also conducted to gain confidence that the MH-algorithm is sampling the correct posterior distribution. Finally the resulting Bayesian model is compared relative to other state of the art college game prediction models like the ESPN FPI and Catherwood Ratings.

2 Data Pipeline Construction

An intensive portion of this project included a *Data Wrangling* phase. Gathering sports data is often tedious and requires web-scraping. For this project a free college football API is utilized to pull in all data. Documentation for this data source can be found at https://collegefootballdata.com/.

To ensure reliable data was available for model construction, a structured regime was followed to conduct *Data Wrangling*, this is outlined in more detail below, and as well can be viewed in the Github repository. (Repository link listed under citations section).

Data Wrangling consisted of three phases:

1. Data Pull: *Process by which all necessary model data is pulled in from the API source and saved.*
2. Data Joins: *Saved API data spanning across multiple data-sets is joined together based on column name relationships to combine all data into a single data-set.*
3. Data Cleaning: *Inconsistencies such as missing values, outliers, and incorrect data-types are detected and corrected in this phase. Cleaned data is then ready for use in the feature engineering phase.*

Data Pull: Python code was constructed from scratch to create a custom pipeline from CollegeFootballData.com to desktop storage locations. *Figure 1* outlines the generic process designed to obtain the initial data to be used for model.

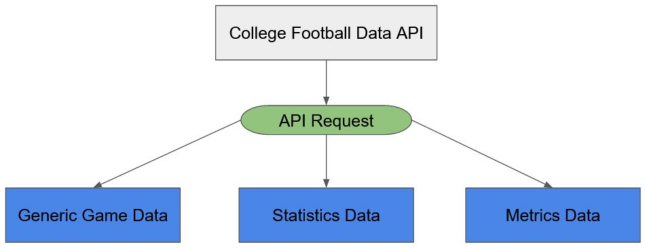


Figure 1: High-Level Data Pulling Process

Below *Figure 2* describes the data contained within each of the initial data-sets constructed from the API. The Game data-set contains high-level team match-up information. The statistics data-set has basic stats such as passing yards achieved by a team during a game. Finally more advanced metric data for every game was pulled so that the Bayesian model could obtain information more indicative of a teams chance of winning. Its also important to note that the data spans from the 2015-2021 college football seasons, resulting in approximately 3,800 games available for modeling.

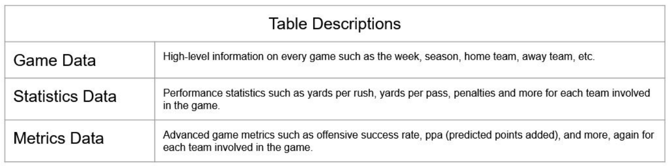


Table Descriptions	
Game Data	High-level information on every game such as the week, season, home team, away team, etc.
Statistics Data	Performance statistics such as yards per rush, yards per pass, penalties and more for each team involved in the game.
Metrics Data	Advanced game metrics such as offensive success rate, ppa (predicted points added), and more, again for each team involved in the game.

Figure 2: Description of Data Obtained from Data Pull

Data Joins: To effectively model the pulled data it is necessary to combine all the data into a single data-set. Every individual data-set obtained in the **Data Pull** process has a unique *Game ID* identifying every game. This *Game ID* is used to join the rows from the game, statistics,

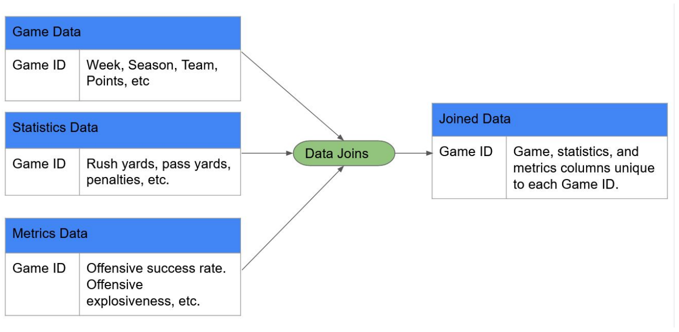

Game Data
Game ID
Week, Season, Team, Points, etc
Statistics Data
Game ID
Rush yards, pass yards, penalties, etc.
Metrics Data
Game ID
Offensive success rate. Offensive explosiveness, etc.
Data Joins
Joined Data
Game ID
Game, statistics, and metrics columns unique to each Game ID.

Figure 3: Data Joining Process

and metrics data into a single row for every game. *Figure 3* above shows the generic process for achieving these results.

Data Cleaning: A full data cleaning exploratory analysis is available in the data_cleaning_eda.ipynb in the Data_Wrangle folder in the Github. The cleaning process handled incorrect data-types, missing data, outliers, and more. To preserve briefness of the paper, I'll highlight the most important operation conducted during the data-cleaning process which is missing data.

10 columns that stored advanced statistical metrics for a teams performance during a game had 90 or more missing values within the dataset. These metrics are valuable for predicting the winner of a football game so it was necessary to perform missing value imputation. Missing value imputation is the process of replacing missing data values with data reflective of what the true value should be. In order to perform this imputation a random forest regression model was trained for each statistical metric with missing data using other statistics columns in the data-frame as features to predict what value should replace the missing value. *Figure 4* below displays this process. This was necessary over simpler approaches like mean value imputation because these advanced metrics are very important in prediction.

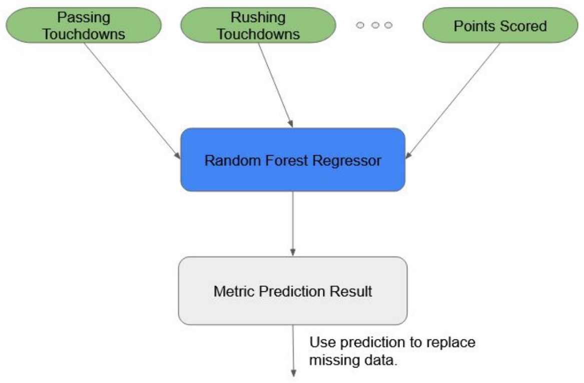


Figure 4: Random Forest Imputation

3 Bayesian Modeling

This section focuses on the selection of final features to be used in the Bayesian model. It also defines the prior, likelihood, and posterior distribution used for Bayesian logistic ridge regression. Finally posterior distribution output and model performance is analyzed in comparison to state-of-art models referenced in the introduction section.

3.1 Feature Selection and Col-linearity

This phase of the project was aimed at condensing a large amount of data features into a smaller subset that is useful for the predictive model. Coming into this portion of the project there were a total of 39 features available to use in the model. In order to start reducing this amount of features a Z-test was performed for every feature at an $\alpha = .05$ level. The Z-score was computed in the following way for every feature:

$$Z = \frac{(\bar{x_W} - \bar{x_L}) - (\mu_W - \mu_L)}{\sqrt{\frac{\sigma_W^2}{n_W} + \frac{\sigma_L^2}{n_L}}} = \frac{(\bar{x_W} - \bar{x_L})}{\sqrt{\frac{\sigma_W^2}{n_W} + \frac{\sigma_L^2}{n_L}}}$$

Where:

$$\bar{x_W} = mean\ of\ feature\ when\ team\ wins$$

$$\bar{x_L} = mean\ of\ feature\ when\ team\ loses$$

$$\sigma_W^2 = variation\ of\ feature\ when\ team\ wins$$

$$\sigma_L^2 = variation\ of\ feature\ when\ team\ loses$$

$$n_W = number\ of\ observations\ where\ team\ wins$$

$$n_L = number\ of\ observations\ where\ team\ loses$$

And the hypothesis:

- Null: $\mu_W = \mu_L$
- Alternative: $\mu_W \neq \mu_L$

The null hypothesis was rejected for 16 features and accepted for 23 features. The 16 features where the null was rejected were deemed significant and I kept them while the non-significant features were tossed out.

The second feature selection methodology examined the col-linearity of significant features remaining in the data-set. *Figure 5* below displays a pair-plot of the significant features.

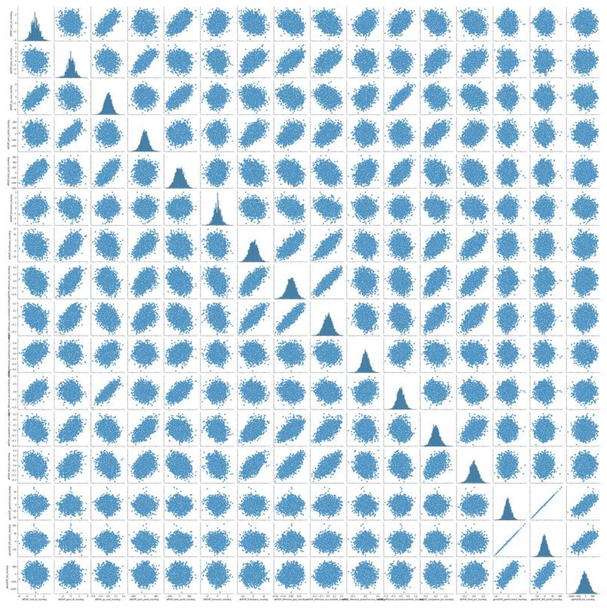
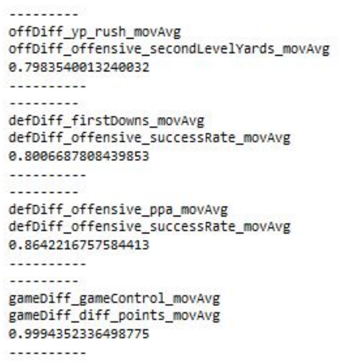


Figure 5: Pair-plot of Significant Features

Its visible that multiple pairs of features within the data-set have linear relationships. To resolve this issue I find feature pairs that have a correlation of greater than .75. This resulted in the following pairs being identified as highly correlated:



```
---------
offDiff_yp_rush_movAvg
offDiff_offensive_secondLevelYards_movAvg
0.7983540013240032
----------
---------
defDiff_firstDowns_movAvg
defDiff_offensive_successRate_movAvg
0.8006687808439853
----------
---------
defDiff_offensive_ppa_movAvg
defDiff_offensive_successRate_movAvg
0.8642216757584413
----------
---------
gameDiff_gameControl_movAvg
gameDiff_diff_points_movAvg
0.9994352336498775
----------
```

Figure 6: Highly Correlated Pairs

The features offDiff_offensive_secondLevelYards_movAvg, offensive_successRate_movAvg, and gameDiff_diff_points_movAvg were removed from the dataset. There were still noticeable correlated features in the dataset remaining after feature selection, however I felt these features were too valuable to model prediction to remove. As a result logistic ridge regression was implemented to try and help mitigate any remaining col-linearity.

A final table of the 13 remaining features used for modeling are described below with the corresponding model parameter they are associated with for reference later on:

Parameter	Feature Name
β_0 (Intercept)	
β_1	offDiff_rush_td_movAvg
β_2	defDiff_pass_td_movAvg touchdowns
β_3	offDiff_yp_rush_movAvg
β_4	defDiff_pass_yards_movAvg
β_5	offDiff_total_yards_movAvg
β_6	defDiff_turnovers_movAvg
β_7	defDiff_firstDowns_movAvg
β_8	defDiff_offensive_ppa_movAvg
β_9	offDiff_offensive_powerSuccess_movAvg
β_{10}	defDiff_completion_pct_movAvg
β_{11}	defDif_third_pct_movAvg
β_{12}	gameDiff_gameControl_movAvg
β_{13}	gameDiff_elo_movAvg

The features above have varying degrees of complexity in terms of interpretation so I feel best to leave these specific descriptions out. However, more information for specific metrics can be found at: https://collegefootballdata.com/glossary. It is important to note that every feature is essentially a moving average measure for a teams offensive or defensive advantage versus its opponent for a given match-up.

3.2 Bayesian Logistic Ridge Regression

In this section the posterior distribution necessary for Bayesian logistic ridge regression is described.

The posterior:

$$\pi(\vec{\beta}, \lambda | \vec{y}) \propto f(\vec{y} | \vec{\beta}, \lambda) \pi(\vec{\beta}, \lambda)$$

Where $\vec{\beta}$ are the vector of logistic regression coefficients $\beta_0, \beta_1, ..., \beta_k$ and λ is the ridge shrinkage parameter.

Note that the prior distribution can be broken down in a hierarchical manner:

$$\pi(\vec{\beta}, \lambda) = \pi(\vec{\beta} | \lambda) \pi(\lambda)$$

This is a result of the coefficients $\vec{\beta}$ being dependent upon the ridge shrinkage parameter λ and λ being treated as unknown in the Bayesian framework. This is contrary to frequentist ridge regression where λ is typically chosen in the process of cross-validation. The following priors are placed to attain ridge regression in the Bayesian framework [5]:

$$\beta_k|\lambda \sim N(0, \frac{1}{2\lambda}), \texttt{for every } k = 1, ...m$$

$$\lambda \sim exp(rate = 1.5)$$

I have chosen the exponential prior with rate 1.5 for λ as a result of ridge shrinkage parameters typically being close to 1 in practice with less common values after $\lambda > 2$.

Additionally for logistic regression the likelihood follows the form:

$$y|\vec{\beta}, \lambda \sim Binomial(n = 1, p)$$

Where $\vec{\beta}$ and λ are related to the binomial parameter p through the logistic link function.

To sample from this posterior distribution the Metropolis-Hastings algorithm will be used. It is necessary to derive the log-posterior in order to perform this algorithm so this is done directly below:

The Posterior:

$$\pi(\vec{\beta}, \lambda) \propto f(\vec{y}|\vec{\beta}, \lambda)\pi(\vec{\beta}|\lambda)\pi(\lambda)$$

$$= f(\vec{y}|\vec{\beta}, \lambda)[\prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\frac{1}{2\lambda}}} e^{-\frac{1}{2\frac{1}{2\lambda}}(\beta_i)^2}][1.5e^{-1.5\lambda} \cdot I_{(0,\infty)}(\lambda)] = f(\vec{y}|\vec{\beta}, \lambda)[\prod_{i=1}^{m} \frac{1}{\sqrt{\frac{\pi}{\lambda}}} e^{-\lambda\beta_i^2}][1.5e^{-1.5\lambda} \cdot I_{(0,\infty)}(\lambda)]$$

$$\propto f(\vec{y}|\vec{\beta}, \lambda)[\prod_{i=1}^{m} \frac{1}{\sqrt{\frac{1}{\lambda}}} e^{-\lambda\beta_i^2}][e^{-1.5\lambda} \cdot I_{(0,\infty)}(\lambda)]$$

Log-Posterior:

$$\textit{Recall its been proved: } \ln f(\vec{y}|\vec{\beta}, \lambda) \propto [\sum_{i=1}^{n} y_i\eta_i - \sum_{i=1}^{n} \ln(1 + e^{\eta_i})]$$

$$Thus, \ln \pi(\vec{\beta}, \lambda|y) \propto \ln(f(\vec{y}|\vec{\beta}, \lambda)) + \ln \pi(\vec{\beta}|\lambda) + \ln \pi(\lambda)$$

$$\propto [\sum_{i=1}^{n} y_i\eta_i - \sum_{i=1}^{n} \ln(1 + e^{\eta_i})] + \sum_{i=1}^{m} \ln(\lambda^{\frac{1}{2}} e^{-\lambda\beta_i^2}) + \ln(e^{-1.5\lambda} \cdot I_{(0,\infty)}(\lambda))$$

The above is the derived form of the log-posterior distribution that will be used in the MH-algorithm.

3.3 Metropolis-Hastings Sampler

The Metropolis-Hastings algorithm is used in order to sample from the posterior distribution derived in section 3.2. Note in section 3.2 that included in the posterior is an indicator variable $I_{(0,\infty)}(\lambda)$ which tells us that there is zero probability of $\lambda < 0$ occurring. To account for this the proposal distribution used in Metropolis-Hastings is:

$$\vec{\beta}, \lambda | \vec{\beta}^*, \lambda^* \sim N(\vec{\beta}^*, \Sigma) \cdot Unif(0,2)$$

Where

$$\vec{\beta} | \vec{\beta}^*, \lambda^* \sim N(\vec{\beta}^*, \Sigma)$$
$$\lambda | \vec{\beta}^*, \lambda^* \sim Unif(0,2)$$

This ensures the Metropolis-Hastings algorithm doesn't samples instances of $\lambda < 0$. Note that Σ is a randomly generated covariance matrix with entries close to 0. The MH-Algorithm using this joint proposal was run over 150,000 iterations for 3 separate initial values for the $\vec{\beta}$ parameters. These initial values are as follows:

- $\vec{\beta}^{(0)} = \hat{\vec{\beta}}_f$ = vector of frequentist ridge regression estimates
- $\vec{\beta}^{(1)} = \vec{-5}$ = vector of -5's as initial guesses
- $\vec{\beta}^{(2)} = \vec{5}$ = vector of 5's as initial guesses

Multiple initial values for $\vec{\beta}$ are used for different runs of the MH-Algorithm in order to gain confidence that the algorithm has converged to the correct posterior distribution. Jeffrey Arnold of the University of Washington writes, "In equilibrium, the distribution of samples from chains should be the same regardless of the initial starting values of the chains" [3]. Equilibrium in this case means that chain has converged to the target posterior distribution. *Figure 7* below shows the trace plots of all three initial value chains for each β_i parameter. It's visible that for almost all parameters, regardless of the initial value for the MH-Algorithm each chain converges to roughly the same sampling space. This supplies some evidence that the MH-Algorithm has achieved equilibrium and is indeed sampling from the target posterior.

Figure 8 below takes a closer look and explores the trace-plots at a burn-in of 50,000. A concerning observation in *Figure 8* are the trace plots for β_{12} and β_{13} which appear to have significant areas of non-overlapping sampling spaces and auto-correlation. Usually auto-correlation can be negated by a large enough metropolis-hastings sample; 150,000 iterations per chain should be large enough to still give convergence to the target distribution. To verify this *Figure 9* below explores the marginal posterior distributions resulting from the MH-algorithm after burn-in 50,000.

Figure 9 indicates that for all parameters aside from β_{12} and β_{13} the three chains corresponding to the differing initial values all attain approximately the same sampled posterior distribution. For

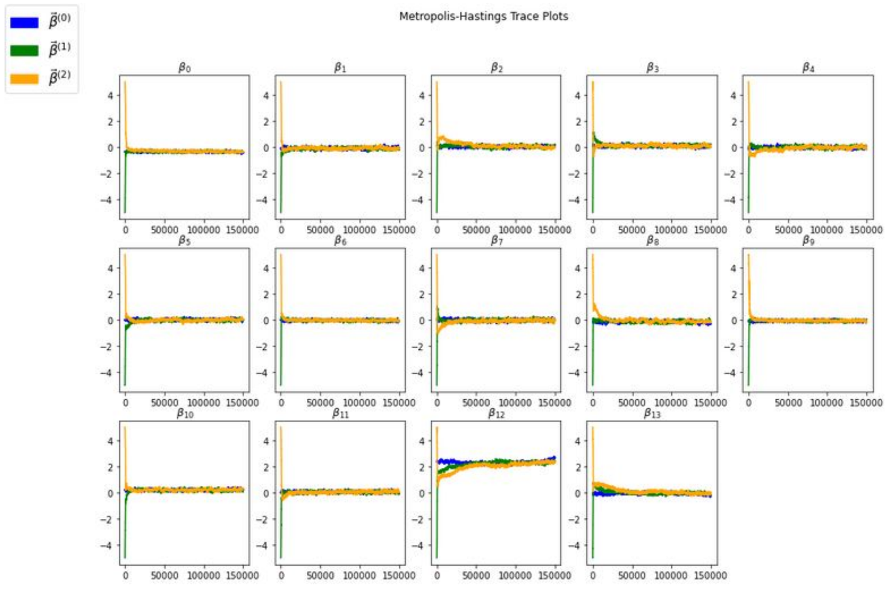


Figure 7: MH Trace Plots

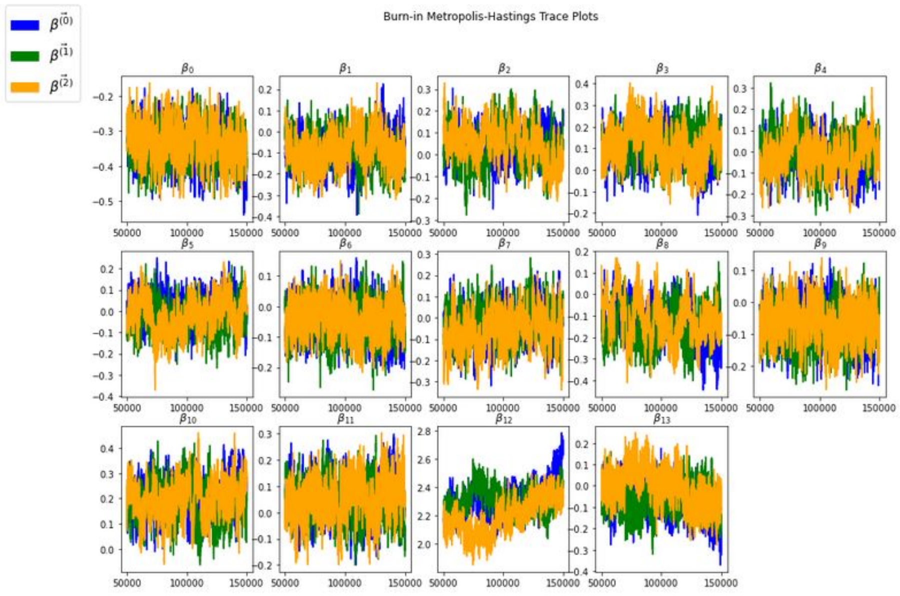


Figure 8: MH Trace Plot w/ Burn-in

β_{12} and β_{13} the sampled posterior distributions for the chains with initial values $\vec{\beta}^{(0)}$ and $\vec{\beta}^{(1)}$ are extremely similar. Since chains in equilibrium should be sampling from the same posterior

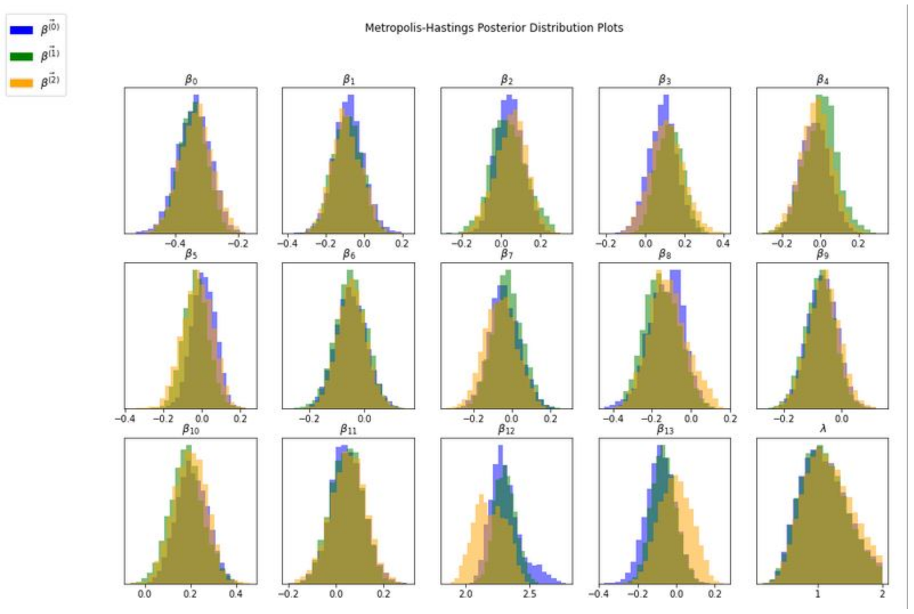


Figure 9: MH Posterior Distributions

regardless of initial values this observation indicates that the $\vec{\beta}^{(0)}$ and $\vec{\beta}^{(1)}$ chains have converged to the target posterior while the chain with $\vec{\beta}^{(2)}$ has not. Due to this result it was decided to use the $\vec{\beta}^{(0)}$ and $\vec{\beta}^{(1)}$ initial value chains as the posterior distribution resulting from the MH-Algorithm. Results from the next section, 3.4, indicate that this was a good decision.

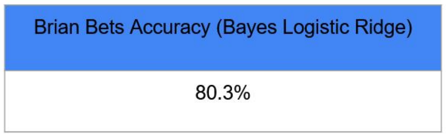
3.4 Model Analysis and Results

A note moving forward is that the Bayesian Logistic Ridge Regression model has been officially named the "Brian Bets" model and is referenced by this name in the text and several graphs. (As well in the oral presentation)

The table below displays the average values from the MH-algorithm sampled posterior distribution for each parameter and additionally the ridge regression frequentist estimate for each parameter using a shrinkage parameter of $\lambda = 1$. Almost all regression coefficients have approximately the same value for the average of the posterior and the frequentist estimate. Specifically, note that β_{12} and β_{13} have posterior average values similar to the frequentist version, further providing evidence that the $\vec{\beta}^{(0)}$ and $\vec{\beta}^{(1)}$ initial values chain did converge to the correct target posterior distribution in section 3.3. Note that it is expected to have some parameter values be different for the frequentist vs. Bayesian model as a result of the λ shrinkage parameter being treated as a random variable in the Bayesian model. The value of λ effects how likely certain parameters are and as a result of accepting varying λ values through the MH-algorithm the result does not have perfect alignment with the frequentist estimates.

Parameter	Posterior Average	Frequentist Estimate
β_0 (Intercept)	-0.34182	-0.347
β_1	-0.08398	-0.08263
β_2	0.04156	0.01881
β_3	0.10765	0.10586
β_4	-0.01124	-0.00289
β_5	-0.00992	-0.00298
β_6	-0.04726	-0.04499
β_7	-0.02984	-0.01772
β_8	-0.13939	-0.13904
β_9	-0.07148	-0.0764
β_{10}	0.19494	0.19342
β_{11}	0.0469	0.04688
β_{12}	2.30935	2.36074
β_{13}	-0.072	-0.10722

In order to evaluate the predictive power of the Bayesian logistic ridge regression a test set of data was held out. *Figure 10* below displays the accuracy results of the model:



Brian Bets Accuracy (Bayes Logistic Ridge)
80.3%

Figure 10: Test Accuracy of Bayesian Logistic Ridge Regression

Another performance metric is the ROC curve and AUC value in *Figure 11* below. The AUC value of .87625 indicates that the model does a excellent job at classifying true negatives and true positives. According to Statology [4] an AUC between .8-.9 is considered excellent.

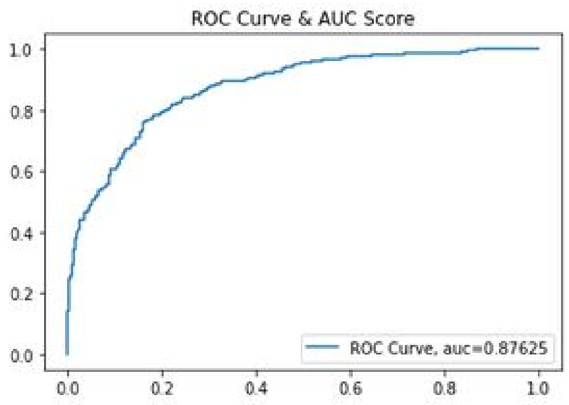
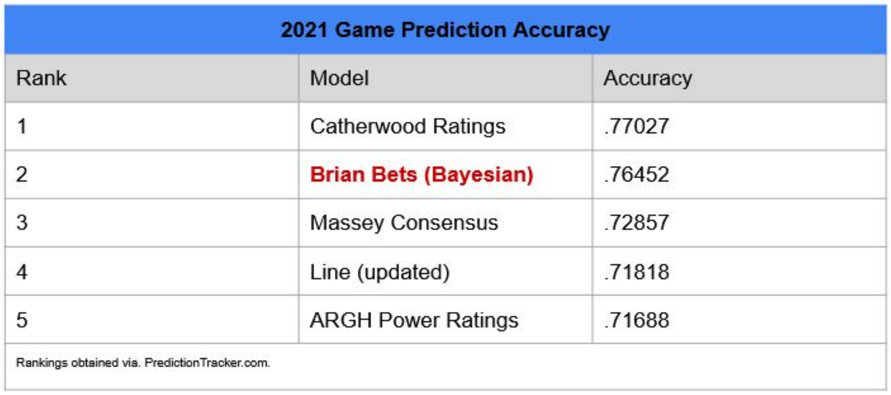


Figure 11: Test ROC and AUC

In order to get a closer comparison to the ”State-of-Art” models from the introduction the accuracy was also tested on the 2021 season game data. Below, *Figure 12* displays the *Brian Bets* model accuracy compared to the top available public facing models in industry. *Brian Bets* places second among all models. Its also important to note that Catherwood ratings and Brian Bets experience a similar accuracy around 77% and all other top models experience a significant drop-off with the next highest accuracy being approximately 73%.



2021 Game Prediction Accuracy

Rank	Model	Accuracy
1	Catherwood Ratings	.77027
2	**Brian Bets (Bayesian)**	.76452
3	Massey Consensus	.72857
4	Line (updated)	.71818
5	ARGH Power Ratings	.71688

Rankings obtained via. PredictionTracker.com.

Figure 12: Ranking of Bayesian Logistic Ridge Regression

In addition to typical model diagnostics I also chose to generate posterior distributions for the Colorado vs. Oregon State game from the fall of 2021. The following steps are performed to produce the posterior distribution over p and the predictive posterior distribution over $y*$:

1. Let x = data observation for the Colorado vs. Oregon State game, note the first entry of this array will be a 1 to account for the intercept parameter β_0
2. For $t = 1, ...T$ in the MH-algorithm sampled posterior distribution use the posterior sample $\hat{\vec{\beta}}^{(t)}$
3. Compute $\hat{p}^* = \frac{e^{x\hat{\vec{\beta}}^{(t)}}}{1+e^{x\hat{\vec{\beta}}^{(t)}}}$
4. Draw $y* \sim Binomial(1, \hat{p}^*)$

The resulting posterior distribution in *Figure 13* below has most of the probability density between p values of .15 and .25 and this is reflected in the posterior predictive distribution $\pi(y * |y)$ with Colorado losing most of the time. For any selected game the above posterior distributions can always be replicated to give a sports bettor or analytic expert an idea of how certain they should be about a team winning a particular game.

To further the Bayesian inference’s usefulness to sports bettors an additional probability distribution was generated over the net-winnings one can earn by placing bets on a certain set of games. The generation of this probability distribution is described below under *Figure 13.*

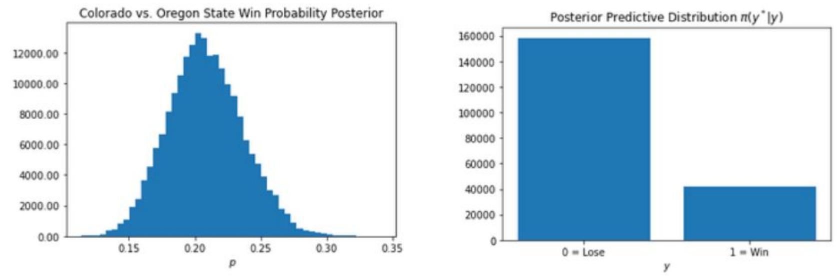


Figure 13: Posterior Predictive Distribution Example

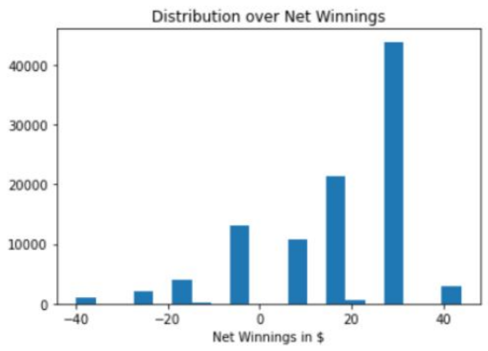
Necessary variables:

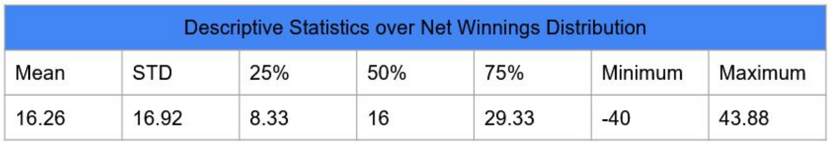
- Posterior Predictive List = list of posterior predictive distributions for a set of games
- Wager List = list of monetary wagers the bettor wishes to place on each game (eg. [$10,$15,....])
- Odds List = list of Vegas odds corresponding to each game (eg. [-200,+300,...])
- Bet List = boolean list of predictions for each game (eg. [1,0,....])
- Sample Number = number of times we want to draw from each games posterior predictive distribution

The Algorithm:

```
1. Winning_Distribution = []
2. For i in 1,....Sample Number
   (a) Winnings = 0
   (b) For $d$ in Posterior Predictive List [Iterate through each game we bet on]
       i. Draw Sample $s$ from $d$
       ii. If $s$ == Boolean Bet
           A. Winnings = Winnings + WinBet(odds,wager) [Note: WinBet is made-up function reflecting reward of winning a bet of certain odds and at a certain wager]
       iii. Else:
           A. Winnings = Winnings - wager
   (c) Winning_Distribution.append(Winnings) [Add sampled net-winnings to overall distribution]
```

Figure 14 below shows the results of running this algorithm on four random games with $10 placed on each game. It can be seen that there is a distribution over the potential net-winnings that the bettor will attain based on the individual posterior predictive distributions for each game. Additionally, a descriptive statistics report is generated to give a numerical summary of the distribution over net-winnings. It can tell a bettor on average how much they should expect to earn, the percentiles of earnings, and the standard deviation observed in the distribution. This level of increased detail should help bettors make more informed decisions.





Descriptive Statistics over Net Winnings Distribution						
Mean	STD	25%	50%	75%	Minimum	Maximum
16.26	16.92	8.33	16	29.33	-40	43.88

Figure 14: Sample Distribution over Net Winnings

4 Conclusions

The Bayesian logistic ridge regression (Brian Bets) out-performed all publicly available college football game prediction models except for the Catherwood Ratings model on 2021 season data. Specifically, the *Brian Bets* model outperforms one of the industry's most popular predictive tools, the ESPN FPI, which does not qualify for the top 5 most accurate models in 2021 at an accuracy of 69.87%. This supplies evidence that the model created in this paper is competitive with other state of the art solutions. The high accuracy result of the *Brian Bets* model combined with added interpret-ability of posterior and posterior predictive distributions resulting from the Bayesian framework offer a great alternative to use over other openly available college football prediction models.

Future work to improve the model would be utilizing more advanced MCMC methods for reducing auto-correlation observed in the trace-plots when sampling the posterior distributions. Additionally more rigorous feature selection could take place to try and further reduce col-linearity amongst the features. It may be possible that the col-linearity is a cause of the observed auto-correlation in the MH-Algorithm.

I plan to utilize this model for the coming college football season to track its effectiveness with small scale college football betting. As a result of this project all the data wrangling and modeling processes have been created so it should be scalable to everyday use when the football season starts this coming fall.

5 Citations

1. https://www.thepredictiontracker.com/ncaaresults.php?year=21
2. https://www.espn.com/blog/statsinfo/post/_/id/122612/an-inside-look-at-college-fpi
3. https://jrnold.github.io/bayesian_notes/mcmc-diagnostics.html
4. https://www.statology.org/what-is-a-good-auc-score/
5. https://stats.stackexchange.com/questions/474958/bayesian-interpretation-of-logistic-ridge-regression

6 Github Link

Please visit the following site to view the github repository for the code:

- https://github.com/btcain4/Bayesian_College_Football