Predicting Win Percentage Using Shrinkage Methods

Austin McCumber, Nathan Heng, Nichole Scarveles

08/07/2022

1 Introduction

Linear regression is a common statistical technique that is used to predict the value of a variable based on a function of one or more predictor variables. When multiple predictor variables are present, models can be improved through selection methods that remove variables and create subsets of the original model. Regularization, or shrinkage, methods that utilize all variables but constrain or regularize the coefficients can also improve upon a linear model using all available predictors.

Linear regression has long been a standard baseline method for prediction of various sports features and outcomes. There have also been many papers comparing statistical methods (including linear models) for sports prediction (*See Koseler & Stephan 2017*). The goal of this paper is to compare the performance of shrinkage methods, specifically ridge and lasso regression, to linear regression models where no coefficient regularization occurs. NCAA Division I Female Softball and NCAA Division 1 Men's Soccer data will be used for comparison, and can be found at: `https://stats.ncaa.org/`. A more detailed description of the data sets used in the analysis can be found in section 3. The conclusion will address the ability of each model, and further opportunities for research.

2 Methods

All models tested are linear, and thus produce a model of the form: [3]

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \epsilon \tag{1}$$

The differences between models will be described in the following passages.

2.1 Linear Regression with Ordinary Least Squares

The first set of methods used involve no penalization term, and rely on the Ordinary Least Squares (OLS) method. Coefficient estimates are calculated by minimizing the residual sum of squares (RSS) equation shown below, where y_i is the value of the i^{th} observation of the target variable, x_{ij} is the value for the j^{th} predictor variable corresponding to the i^{th} observation, β_0 is the estimated intercept, β_j is the estimated coefficient of the j^{th} predictor variable, p is the number of predictor variables, and n is the number of observations.[3]

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \tag{2}$$

2.1.1 Full Linear Model

The first linear regression model tested, known in this paper as the full linear model, uses every available predictor variable in the estimation of a team's regular season winning percentage for the 2021/2022 season.

Therefore the number of parameters to estimate, including the intercept, will be $p+1$. The estimated full models are shown in section 3.

OLS can be inaccurate when correlation exists between predictor variables[2]; which is known as multicollinearity. When multicollinearity is present, variable coefficients can become too large in absolute value or have an incorrect sign. The more multicollinearity present, the worse these problems can become.[4] The following methods should work to address this.

2.1.2 Forward Selection

Forward selection is a form of model selection that uses a defined measurement criteria to select the most appropriate model from a set of models with a varying number of estimated parameters. The process begins with a model containing no predictors. The algorithm then selects a model with one predictor, and then two predictors, and so on until all available predictors have been included. In the process of model selection, each step of forward selection will compare the reduction in RSS from adding a specific predictor variable against all other variables. The variable that reduces RSS the most is selected and added to the model. The most appropriate model is then selected after examining the Bayesian Information Criterion, or BIC (equation 3), for each model and selecting the minimum.

$$\text{BIC} = \frac{1}{n}(\text{RSS} + log(n)p\hat{\sigma}^2) \tag{3}$$

$\hat{\sigma}^2$ is an estimate of the variance of ϵ for each response measurement.[3]

2.1.3 Backward Selection

Backward selection is similar to forward selection except that it starts with the full linear model and proceeds backwards one variable at a time. The predictor variable removed at each step is the variable producing the lowest addition to RSS. BIC can also be used for model comparison using backward selection.[3]

By using selection and subset methods to reduce the number of predictor variables found in the model, the hope is to avoid overfitting the model to the training data set. The subset models can be found in section 3.

2.2 Linear Regression Using Shrinkage Methods

Ridge and Lasso regression are of a subclass of linear regression, and therefore will produce a final model estimate of the same form found in equation (1). However, both methods use a different minimization criterion than traditional least squares regression. The minimization criterion for ridge and lasso regression introduces a tuning parameter, λ, which penalizes large values of the β_j estimates. As λ increases, the flexibility of both ridge and lasso regression models decrease. In response, the model is more stable to changes in the training data set by decreasing the variance and increasing the bias.[3] Originally, λ was determined by plotting the ridge coefficients against values of λ and looking for values of λ where the coefficients appear to stabilize. A more modern alternative is to select a value for λ that results in the smallest cross-validation prediction error.[9]

2.2.1 Ridge Regression

Ridge regression uses the below minimization criterion to estimate coefficients.

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}{\beta_j}^2 \tag{4}$$

The addition of the second term $\lambda\sum_{j=1}^{p}{\beta_j}^2$, known as the penalty term, is a departure from the traditional least squares minimization. The penalty term in ridge regression will shrink all coefficients to zero at a rate

that depends on the size of λ, but it will not set any coefficients equal to zero. This means it will utilize each predictor variable in the estimation.[3] The estimated ridge regression models are shown in section 3.

2.2.2 Lasso Regression

Lasso regression is very similar to ridge regression with one minor difference in the penalty term.

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| \tag{5}$$

The utilization of $|\beta_j|$ instead of β_j^2 inside the penalty term allows for coefficients to equal zero, a feature not possible with ridge regression. Like selection methods, this feature can serve to reduce overfitting of the model to the training data.[3] The estimated lasso regression models are shown in section 3.

2.3 Performance Metrics

In order to compare the performance of models, some common metrics must be used. For prediction, the metric must be able to evaluate the closeness of a prediction to the actual response. Two commonly used methods for this are the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). The formula for calculating the two metrics is shown below.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2} \tag{6}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{f}(x_i)| \tag{7}$$

Where $\hat{f}(x_i)$ represents the prediction for the i^{th} observation. Models with lower values of these metrics are considered better performing. Although very similar, the squared term in the RMSE calculation tends to more heavily penalize larger prediction errors than the MAE, so having both metrics can help compare models.[3]

3 Illustrative examples with R

In the interest of conserving paper length certain code chunks were not included in the report. The full scripts for each example can be obtained by contacting the authors at nh59@students.uwf.edu, nss21@students.uwf.edu, or ajm104@students.uwf.edu.

3.1 Example 1: NCAA Division I Softball

3.1.1 Data

NCAA Division I softball data for the 2016/2017-2021/2022 seasons was collected from `https://stats.ncaa.org/`. A team's regular season winning percentage for the 2021/2022 season was established as the target variable for the analysis. Several predictor variables were gathered, as well, in two separate groups: Prior season team statistics and historical five season average team statistics. The prior season (2020/2021) team statistics include winning percentage, batting average [5], earned run average [6], fielding percentage [7], and home runs per game. The same statistics were averaged over the five seasons prior to the 2021/2022 season (2016/2017 - 2020/2021) as well. Only teams with five full years of historical seasons were considered for the analysis. As a whole, the data found on the website was extremely thorough and provided almost no missing

values, with the exception of the home run statistics. A few teams had missing seasons of home run totals. Those missing values were imputed using the mean home runs per game for a team, excluding that team's missing year(s).[1] Of the 1462 team and season combinations for the 2016/2017 - 2020/2021 seasons, only 8 had missing home run values, so the impact to the final result is minimal.

3.1.2 Train/Test Split

The data was split into train and test sets using a 70/30 split. The train split will be used to build the models, and the test split will be used to test the model performance by comparing RMSE and MAE values across models.

```
set.seed(123456)
rows <- sample(nrow(NCAA_4_MOD))
NCAA_4_MOD <- NCAA_4_MOD[rows, ]
split <- round(nrow(NCAA_4_MOD) * 0.7)
train <- NCAA_4_MOD[1:split, ] %>% select(!Team)
test <- NCAA_4_MOD[(split + 1):nrow(NCAA_4_MOD), ] %>% select(!Team)
x_train <- train %>% select(-c(WIN_PCT_2022)) %>% as.matrix()
x_test <- test %>% select(-c(WIN_PCT_2022)) %>% as.matrix()
y_train <- train %>% select(WIN_PCT_2022) %>% as.matrix()
y_test <- test %>% select(WIN_PCT_2022) %>% as.matrix()
```

3.1.3 Full Linear Model

The full linear model using all predictor variables was calculated by minimizing the RSS, as described in section 2.1. The final estimated model is shown below.

```
lm_mod <- lm(WIN_PCT_2022 ~ ., data = train)
```

Final Model for Full Linear Regression:
$Y_{\text{WinPct2022}} = 1.6761 + 0.0444X_{\text{WinPct2021}} + 0.0.6384X_{\text{BA2021}} - 0.0249X_{\text{ERA2021}} + 3.2685X_{\text{FP2021}} - 0.0140X_{\text{HR2021}} + 0.4110X_{\text{WinPct2017-2021}} - 0.4842X_{\text{BA2017-2021}} - 0.0205X_{\text{ERA2017-2021}} - 4.6142X_{\text{FP2017-2021}} + 0.0398X_{\text{HR2017-2021}}$

3.1.4 Forward Selection

The same minimization criterion used to calculate the full model was used for both forward and backward selection, with the addition of the selection algorithms described in sections 2.1.2 and 2.1.3.

```
forward_model <- regsubsets(WIN_PCT_2022 ~ ., data = train, nvmax = 10, method = "forward")
test_forward_coef <- unname(coef(forward_model, which.min(summary(forward_model)$bic)))
test_forward_coef_names <- names(coef(forward_model, which.min(summary(forward_model)$bic)))
test_forward <- test %>% mutate("(Intercept)" = 1) %>% select(all_of(test_forward_coef_names)
    )
```

By minimizing the BIC, the forward selection algorithm produced a final estimated model with two coefficients.

Final Model for Forward Selection:
$Y_{\text{WinPct2022}} = 0.4870 + 0.3820X_{\text{WinPct2021}} - 0.0470X_{\text{ERA2017-2021}}$

3.1.5 Backward Selection

```
backward_model <- regsubsets(WIN_PCT_2022 ~ ., data = train, nvmax = 10, method = "backward")
test_backward_coef <- unname(coef(backward_model, which.min(summary(backward_model)$bic)))
test_backward_coef_names <- names(coef(backward_model, which.min(summary(backward_model)$bic)
    ))
test_backward <- test %>% mutate("(Intercept)" = 1) %>% select(all_of(test_backward_coef_
    names))
```

Similar to the forward selection model, the model with the minimum BIC produced by backward selection contained only two coefficients.

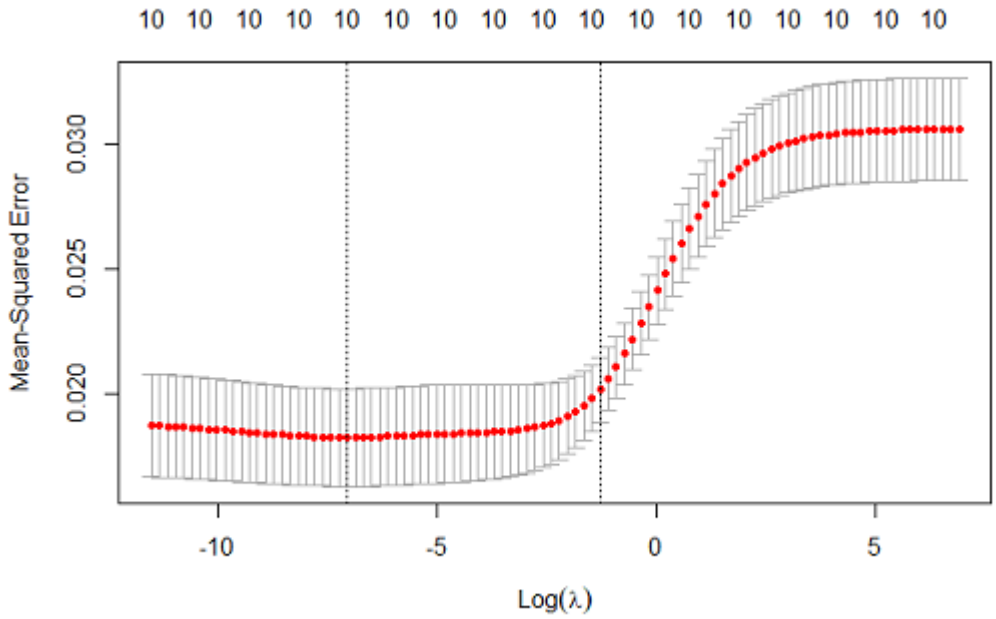
Final Model for Backward Selection:
$Y_{\text{WinPct2022}} = 0.4416 - 0.0433X_{\text{ERA2021}} + 0.4477X_{\text{WinPct2017-2021}}$

3.1.6 Ridge Regression

As stated above the regularization methods have an additional parameter, λ, that requires estimation. Several different λ values were tested, and the best value for each method was selected using 5-fold cross validation on the training data. The λ value which produced the lowest average RMSE was selected for each method separately.[3]

Selecting λ for Ridge Regression

```
myFolds <- createFolds(train$WIN_PCT_2022, k = 5)
folds_unlist <- unname(unlist(myFolds))
fold_ids <- c(rep(1,length(myFolds[[1]])),rep(2,length(myFolds[[2]])), rep(3,length(myFolds
    [[3]])), rep(4,length(myFolds[[4]])), rep(5,length(myFolds[[5]])))
fold_df <- as.data.frame(cbind(fold_ids, folds_unlist)) %>% arrange(folds_unlist)
myFoldID <- fold_df$fold_ids
lambdas_to_try <- 10^seq(-5, 3, length.out = 100)
cv_model_ridge <- cv.glmnet(x_train, y_train, alpha = 0, lambda = lambdas_to_try, foldid =
    myFoldID, standardize = FALSE)
lambda_ridge <- cv_model_ridge$lambda.min
plot(cv_model_ridge)
```



The above graph [8] shows the minimum value of $\log(\lambda)$ occurring at -7.0473, which translates to a value of .0009 for λ. This will be the value for λ used to test the performance of the ridge model on the test set.

As expected, the final model utilizes every available predictor variable. The coefficients are very similar to that of the full linear model from section 4.2, with the coefficients being pushed toward zero by the penalization term.

```
ridge_model <- glmnet(x_train, y_train, alpha = 0, lambda = lambda_ridge)
```

Final Model for Ridge Regression:

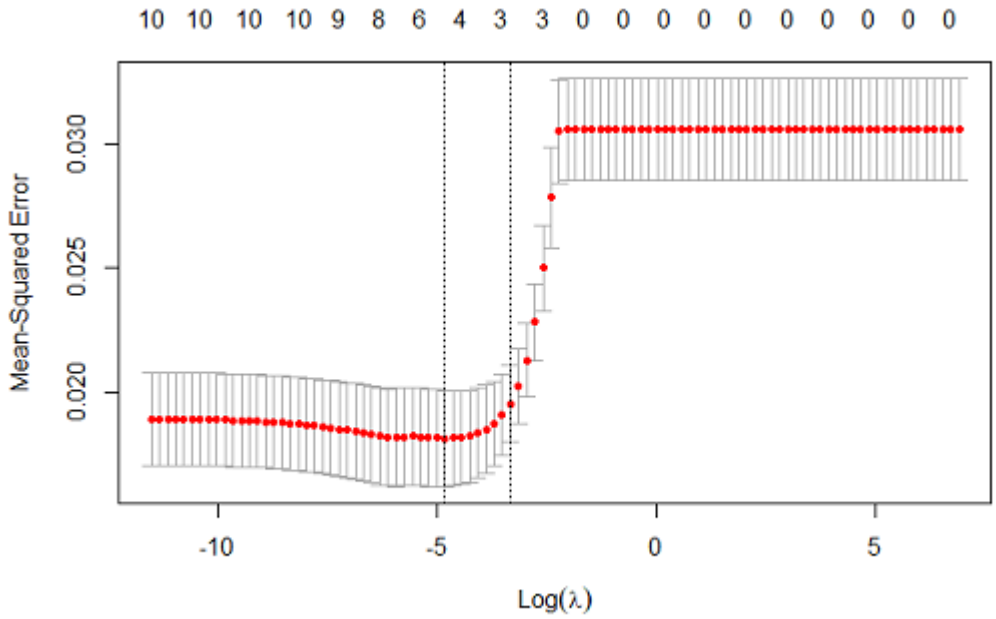
$Y_{\text{WinPct2022}} = 1.5515 + 0.0640X_{\text{WinPct2021}} + 0.5730X_{\text{BA2021}} - 0.0240X_{\text{ERA2021}} + 3.1499X_{\text{FP2021}} - 0.0136X_{\text{HR2021}} + 0.3649X_{\text{WinPct2017-2021}} - 0.3545X_{\text{BA2017-2021}} - 0.0226X_{\text{ERA2017-2021}} - 4.3658X_{\text{FP2017-2021}} + 0.0397X_{\text{HR2017-2021}}$

3.1.7 Lasso Regression

Selecting λ for Lasso Regression

The same cross validation folds and sample λ values were used to select the lasso regression penalization parameter (λ) as well.

```
cv_model_lasso <- cv.glmnet(x_train, y_train, alpha = 1, lambda = lambdas_to_try, foldid =
    myFoldID)
lambda_lasso <- cv_model_lasso$lambda.min
plot(cv_model_lasso)
```



The above graph [8] shows the minimum value of $\log(\lambda)$ occurring at -4.8145, which translates to a value of .0081 for λ. This will be the value for λ used to test the performance of the lasso model on the test set.

The lasso regression model ends up producing four coefficients, falling somewhere between the selection methods and the full model.

```
lasso_model <- glmnet(x_train, y_train, alpha = 1, lambda = lambda_lasso)
```

Final Model for Lasso Regression:

$Y_{\text{WinPct2022}} = -0.4936 + 0.1427X_{\text{WinPct2021}} - 0.0267X_{\text{ERA2021}} + .9069X_{\text{FP2021}} + 0.3109X_{\text{WinPct2017-2021}}$

3.1.8 Performance

The RMSE and MAE were calculated using the predictions of each method (full linear, forward selection, backward selection, ridge, and lasso) and the actual 2021/2022 winning percentage values found in the test set. The formulas found in the Methods section of this paper were used to calculate each metric.

Method	RMSE	MAE
Lasso Regression	0.1318	0.1028
Ridge Regression	0.1327	0.1041
Full Linear Model	0.1330	0.1045
Linear Model: Forward Selection	0.1344	0.1059
Linear Model: Backward Selection	0.1347	0.1038

The metrics perform similarly, with the two shrinkage methods appearing at the top. Lasso regression looks to be the best performer. However, the full linear model performs very similarly to the ridge regression model, and the two selection method models appear at the bottom, with the caveat that the backward selection model is actually second in performance using MAE.

3.2 Example 2: NCAA Division I Men's Soccer

3.2.1 Data

The data identified was Men's Soccer Division I data. The variable for prediction used was win percentage, averaged over the course of five seasons, from the 2017/2018 season through the most recent season 2021/2022. The predictor variableS used for modeling were number of yellow cards, number of red cards, shot accuracy as a percentage, number of shots on goal, average points per game, and average fouls per game. This data was compiled per season and summarized as a five season average to create the data set used for modeling. All data was acquired from `https://stats.ncaa.org/`.

3.2.2 Train/Test Split

The data was again split into train and test data sets similar to example 1.

3.2.3 Full Linear Model

First, a linear regression model representing every predictor variable was created.

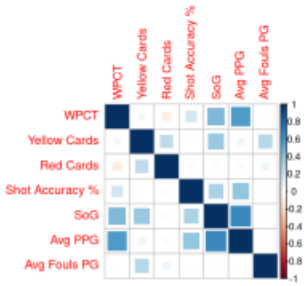
```
lm_train <- lm(mean_wpct ~ ., data = train)
lm_pred <- predict(lm_train, newdata = test, type = "response")
```

Final Model for Full Linear Regression:
$Y_{\text{WinPct}} = -0.1076 + 0.0008X_{\text{YellowCard}} - 0.0037X_{\text{RedCard}} + 0.1377X_{\text{ShotAcc}} + 0.0037X_{\text{SoG}} + 0.0428X_{\text{PPG}} + 0.0040X_{\text{FPG}}$

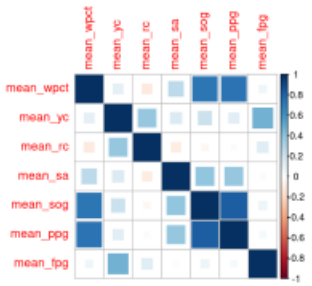
3.2.4 Subset Linear Model

A correlation plot was created using the original data before summarizing to identify as best as possible if any factors seem significantly influential on the win percentage variable. A correlation plot was created again, after the data had been summarized by team to 5 year averages for all variables. There was very little difference between the plots, both identifying shots on goal and points per game as influential predictor variables.

```
library(corrplot)
corr.df <- df.ms % >% select(-c(Team))
corrplot(cor(corr.df), method = "square")
raw_ms.df <- MensSoccer %>% select(-c(Team))
corrplot(cor(raw_ms.df), method = "square")
```



(a) Before Summary



(b) After Summary

A subset model was created using only the significant predictors as defined by the correlation plot.

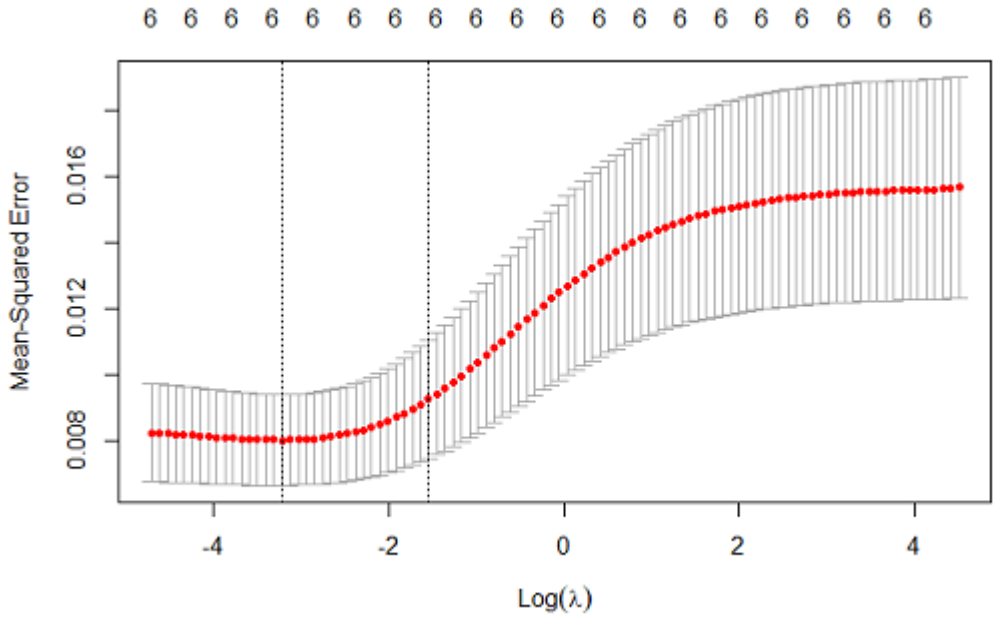
```
lm_corr <- lm(mean_wpct ~ mean_sog + mean_ppg, data = train )
```

Final Model for Subset Linear Regression:
$Y_{\text{WinPct}} = -0.0315 + 0.0042X_{\text{SoG}} + 0.0370X_{\text{PPG}}$

3.2.5 Ridge Regression

Cross validation was also used to obtain the optimal value of λ for the ridge and lasso regression models in this example.

```
set.seed(1000)
rs <- sample(nrow(df.ms), 0.7*nrow(df.ms))
train <- df.ms[1: rs, ] %>% select(-c(Team))
test <- df.ms[rs: nrow(df.ms), ] %>% select(-c(Team))
x_train <- train %>% select(-c(mean_wpct)) %>% as.matrix ()
y_train <- train %>% select(mean_wpct) % >% as.matrix()
x_test <- test %>% select(-c(mean_wpct)) %>% as.matrix()
y_test <- test %>% select(mean_wpct) %>% as.matrix()
ridge_val_model <- cv.glmnet(x_train, y_train, alpha = 0)
plot(ridge_val_model)
```



The above graph shows the minimum value of $\log(\lambda)$ occurring at -3.1252, which translates to a value of .0439 for λ. This will be the value for λ used to test the performance of the lasso model on the test set.

```
ridge_lambda <- ridge_val_model$lambda.min
```

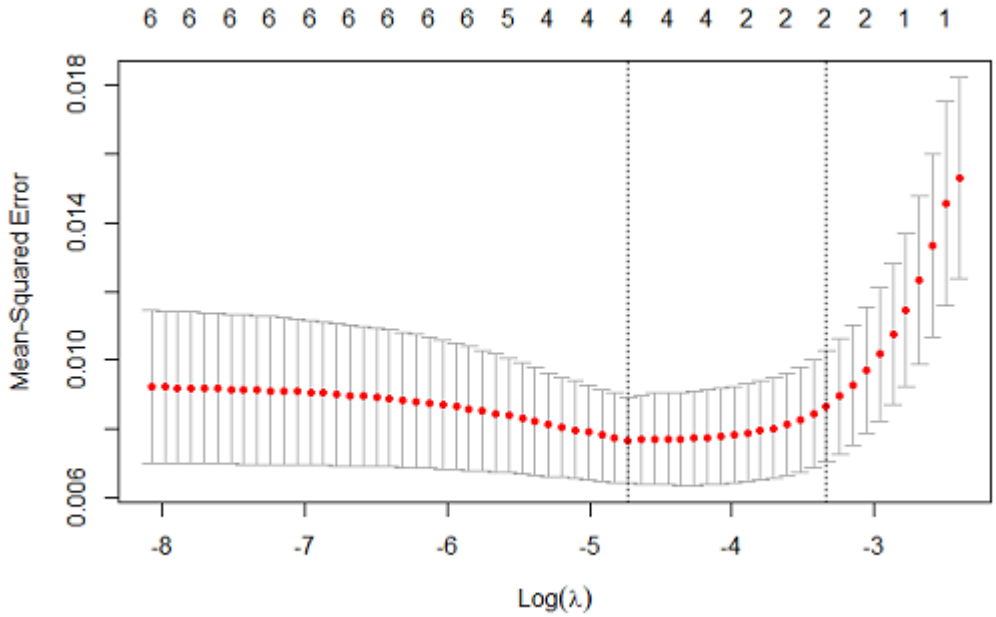
```
ridge_train <- glmnet(x_train, y_train, alpha = 0, lambda = ridge_lambda)
```

Final Model for Ridge Regression:

$Y_{\text{WinPct}} = -0.0090 + 0.0010X_{\text{YellowCard}} - 0.0248X_{\text{RedCard}} + 0.3029X_{\text{ShotAcc}} + 0.0027X_{\text{SoG}} + 0.0415X_{\text{PPG}} + 0.0028X_{\text{FPG}}$

3.2.6 Lasso Regression

```
lasso_val_model <- cv.glmnet(x_train, y_train, alpha = 1)
plot(lasso_val_model)
```



The above graph shows the minimum value of $\log(\lambda)$ occurring at -4.7301, which translates to a value of .0088 for λ. This will be the value for λ used to test the performance of the lasso model on the test set.

As in example 1, the lasso regression model ends up producing four coefficients, falling somewhere between the subset method and the full model.

```
lasso_lambda <- lasso_val_model.min
lasso_train <- glmnet(x_train, y_train, alpha = 1, lambda = lasso_lambda)
```

Final Model for Lasso Regression:

$Y_{\text{WinPct}} = -0.0067 - 0.0171X_{\text{RedCard}} + 0.1116X_{\text{ShotAcc}} + 0.0037X_{\text{SoG}} + 0.0337X_{\text{PPG}}$

3.2.7 Performance

The RMSE and MAE for each method were calculated exactly as in example 1.

Method	RMSE	MAE
Subset Linear Model	0.0900	0.0737
Full Linear Model	0.0901	0.0725
Lasso Regression	0.0917	0.0749
Ridge Regression	0.0932	0.0759

Again the metrics perform similarly. However, the shrinkage methods appear at the bottom of performance in this example, with lasso a slight improvement over ridge. The subset model is also a slight improvement over the full model. With the models having less coefficients performing better than their respective counterparts, this may hint at the model being overfit when all of the coefficients are used.

3.3 Conclusion

There is very little separation between models when looking at the entire picture of all models for both examples. Example 1 shows a prediction boost in both RMSE and MAE using the lasso regression model, while example 2 shows both shrinkage methods as the lowest performing models. The full linear model and the ridge regression model perform very similarly in both examples. This is likely because the λ value for the ridge model is extremely close to zero (more so in example 1), causing very little regularization. Both the full linear models and the ridge regression models produce coefficients that are counter intuitive. For example, using the NCAA softball data, both models show a negative coefficient in front of the prior season home runs per game, prior five season batting average and prior five season fielding percentage. In the NCAA Men's Soccer data a fouls per game produces a positive coefficient. Higher numbers in softball for home runs per game, batting average, and fielding percentage, and lower numbers in soccer of fouls per game are typically considered better, so this may be indicative of the training models being overfit. While the coefficient results for the models that don't utilize every available variable (forward selection, backward selection, subset, and lasso) don't appear to be counter intuitive, the result is not clear as to whether there is any increase in predictive performance on predicting win percentage in sports from using ridge and lasso regression.

3.3.1 Further Research

All models tested in the above analysis were linear with an identity link function. A future consideration of this predictive exercise would be to test out different model types. For example, since the target variable is a proportion, repeating the analysis with a binomial generalized linear model might better capture the true form of the underlying data. Or, in contrast to using a linear model form, using a random forest model or generalized additive model might provide more flexibility for prediction.

References

[1] David Campos, Shon Inouye, and Chester Ismay. *Dealing With Missing Data in R.* URL: https://app.datacamp.com/learn/courses/dealing-with-missing-data-in-r. (accessed: 07.15.2022).

[2] Danald E. Hilt and Donald W. Seegrist. "Ridge, a computer program for Calculating Ridge Regression Estimates". In: *USDA FOREST SERVICE RESEARCH NOTE* NE-236 (1977). DOI: https://www.nrs.fs.usda.gov/pubs/rn/rn_ne236.pdf.

[3] Gareth James et al. *An Introduction to Statistical Learning with Applications in R: Second Edition.* Springer Texts in Statistics. Springer, 2013. ISBN: 9781461471370.

[4] Ghadban Khalaf. "Improving the Ordinary Least Squares Estimator by Ridge Regression". In: *Open Access Library Journal* 9.e8738 (2022), pp. 1–8. DOI: 10.4236/oalib.1108738.

[5] MLB. *Batting Average (AVG).* URL: https://www.mlb.com/glossary/standard-stats/batting-average. (accessed: 07.15.2022).

[6] MLB. *Earned Run Average (ERA).* URL: https://www.mlb.com/glossary/standard-stats/earned-run-average. (accessed: 07.15.2022).

[7] MLB. *Fielding Percentage (FPCT).* URL: https://www.mlb.com/glossary/standard-stats/fielding-percentage. (accessed: 07.15.2022).

[8] Michał Oleszak. *Regularization Tutorial: Ridge, Lasso and Elastic Net.* URL: https://www.datacamp.com/tutorial/tutorial-ridge-lasso-elastic-net. (accessed: 07.15.2022).

[9] *Statistics - (Shrinkage—Regularization) of Regression Coefficients.* URL: https://datacadamia.com/data_mining/shrinkage. (accessed: 08.22.2022).