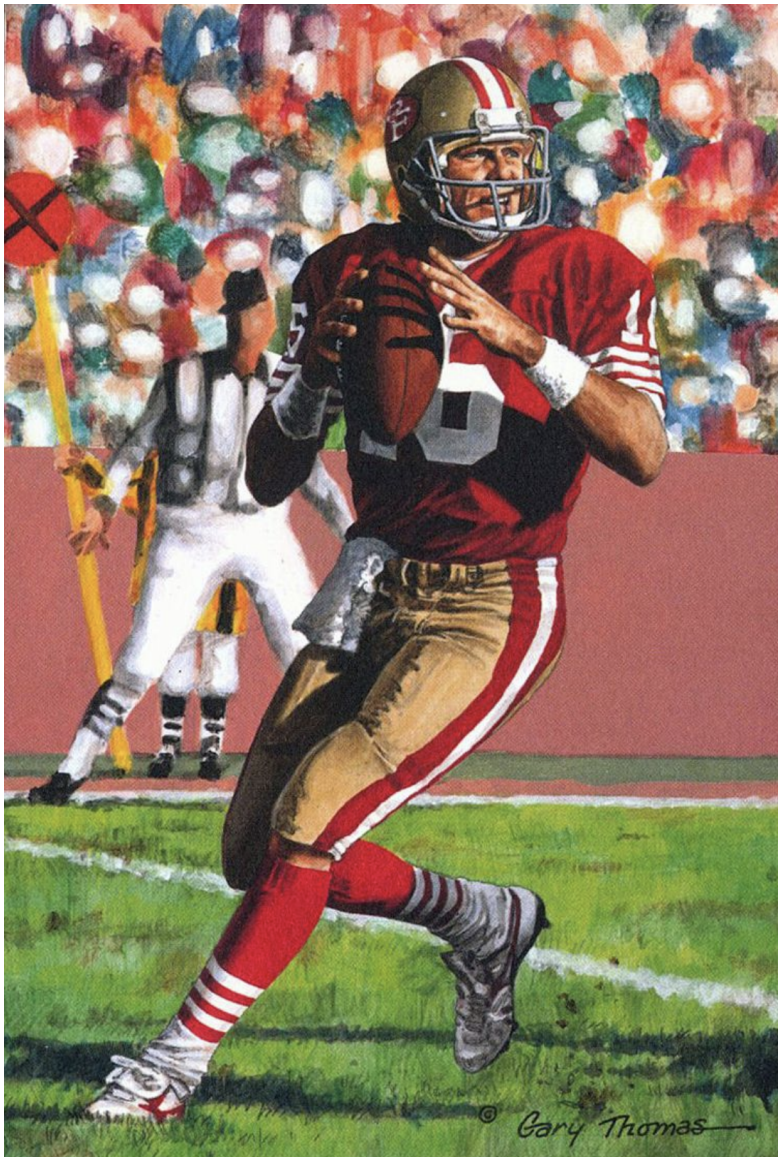
NFL 2021 Predictions:

Spread, Total, and Result

Team 1: Sam Moore, Garrett Mckinstry, Winnie Ren, Yiran Li, Tongyu Zhao



I. Data

A. Cleaning Summary

To clean and groom the data into a form that could be useful for projecting the outcome of a game, we first took the play by play data contained on the NFLfastR repository and aggregated it onto the game level using the "GAMEID" variable, mimicking the style of Dr. Mario's sample file. Then, we used the summarize() and sum() functions to count the total number of certain actions that we would use as predictors. Then, we split this new, game-by-game data set into 2, one containing the data for the home teams and one for the away teams, and then re-joined them back together based on the GAMEID and home/away cities. This allowed for each game to be contained in one row, allowing for the software to be able to properly use the data for predictive modeling. Finally, we renamed each of the variables to be slightly more descriptive and easier to understand. Overall, this data set contained data for both regular season and playoff games from the beginning of the 2010 season until the present. We called this data set "total_games," and this set was the basis for all of our models.

We also downloaded the offensive and defensive season stats for every team from the 2003 season until the present. These were obtained from the data repository provided, which in turn obtained them from Pro Football Reference. This contained basic statistics such as total yards, total touchdowns, rushing/passing yards, and TDs, etc. However, there were many additional, more advanced stats that were not included in the initial data sets that we wanted to use to help predict the outcome of games. These included interceptions, sacks, 3rd and 4th down conversions/conversion rates, and all kicking stats. In order to add these to the data set, we web scraped them from Pro Football Reference and joined them into the year-long data by team name and year. In order to address the case in which a team did not attempt a conversion/kick of any time, we set those rates, initially noted as NaN in R, to 0. This would avoid massive data loss, because in many games there are no 4th down or 2 point conversions attempted, and unless properly dealt with those entire games would be lost.

We primarily used this data to do out-of-sample testing for our models. We would take the data, divide it by the number of games played, and then use it as a method of validating our model selection.

Finally, we created a predictor set, which contained all of the per-game data for this season and was organized in the same manner as our "total_games" data set, with each variable used in prediction filled with the per-game stats for this season for the home and away teams.

B. New Engineered Variable

For our engineered variable, we decided to create a new statistic called "conversion advantage," which calculated the 3rd down conversion advantage/disadvantage for the home team against the away team. 3rd down is often considered the most important down in football because it is often the deciding play as to whether or not a drive will continue or if a team will have to punt. In the red zone, a 3rd down conversion can mean the difference between scoring a

touchdown or having to settle for a field goal. We figured that whichever team has an advantage on 3rd downs throughout the game has a better chance to score more points and end up winning the game. The formula is below:

$$Conversion\ Advantage\ =\ \frac{3rd\ down\ percentage\ (Home)}{3rd\ down\ percentage\ (Away)}$$

A number greater than 1 would signify a home team advantage, while a number below 1 would signify the home team being worse on 3rd downs than their opponents.

C. Outside Data

As mentioned above, we had to web scrape lots of external data on seasonal stats for conversions, kicking, and some defensive stats from Pro Football Reference for our prediction and validating data sets. These are valuable parts of the game and are crucial to determining points and results. However, after doing some outside research, we chose to incorporate FiveThirtyEight's elo rating system for NFL teams. This rating relies heavily on individual players and their performances to assign a total skill rating to the team, including higher weights for skill positions such as QB, WR, and RB, generally the most impactful positions for offensive production. We were operating totally on the team level initially, and this rating, while a team rating, is able to add a bit of player performance impact into our models. Teams with injured QBs such as the NO Saints and the GB Packers suffered large elo hits for the time period in which we are predicting. Also, FiveThirtyEight had a data repository with elo ratings of every team after every game since the NFL-AFL merger, and we webscraped this data from 2010 into our game-by-game predicting set.

D. Variable Names

Below is a list of our variable names in our dataset:

Year, WEEK, AWAYCITY, HOMECITY, Home_Team, Total Yards_H, TD Pass_H, TD Rush_H, FGA_H, FGM_H, EPA_H, EPM_H, 2PCA_H, 2PCM_H, SFTY_H, Rush_Yards_H, Pass_Yards_H, RA_H, PA_H, Rush_YpA_H, Pass_YpA_H, INT_H, SACK_H, 3rd_PCT_H, 4th_PCT_H, COMP_H, COMP_PCT_H, FG_PCT_H, PEN_H, PEN_YDS_H, TO_H, HOME_PTS, Pass/Rush_Ratio_H, Away_Team, Total Yards_A, TD Pass_A, TD Rush_A, FGA_A, FGM_A, EPA_A, EPM_A, 2PCA_A, 2PCM_A, SFTY_A, Rush_Yards_A, Pass_Yards_A, RA_A, PA_A, Rush_YpA_A, Pass_YpA_A, INT_A, SACK_A, 3rd_PCT_A, 4th_PCT_A, COMP_A, COMP_PCT_A, FG_PCT_A, PEN_A, PEN_YDS_A, TO_A, AWAY_PTS, Pass/Rush_Ratio_A, elo_H, elo_A, Spread, Total, WINLOSS, CONV_ADV

Below are the full names of the abbreviations used in the variable name:
(Each of these variables will have _H or _A after it to indicate the Home or the Away team)

TD_Rush: Rushing Touchdowns Made
TD_Pass: Passing Touchdowns Made
EPA: Extra Points Attempted
EPM: Extra Points Made
2PCA: 2 Point Conversions Attempted
2PCM: 2 Point Conversions Attempted
SFTY: Safeties Made
Rush_Yards: Total Rush Yards Gained
Pass_Yards: Total Passed Yards Gained

RA: Rush Attempts
PA: Pass Attempts
Rush_YpA: Yards Per Rush Attempt
Pass_YpA: Yards Per Pass Attempt
INT: Number of Interceptions
SACK: Number of Sacks
3rd_PCT: 3rd Down Percentage
4th_PCT: 4th Down Percentage
COMP: Number of Completions
COMP_PCT: Completion Percentage
FGA: Field Goals Attempted

FG_PCT: Field Goal Percentage
FGM: Field Goals Made
PEN: Penalties
PEN_YDS: Number of yards given up through penalties
TO: Turnovers
Pass/Rush_Ratio: The ratio of pass attempts to rush attempts
ELO: Elo Rating (ranking system)
CONV_ADV: (Refer to Section about New Engineered Variable Above)

Spread = HOME_PTS - AWAY_PTS
Total = HOME_PTS + AWAY_PTS
WINLOSS = 1 = Home Wins, 0 = Home Loses
(If HOME_PTS = AWAY_PTS, we decided that WINLOSS = 1 since the Home Team typically has the advantage when games go into overtime.)

E. Dealing with NaNs and NAs in Dataset

As previously mentioned, there were cases where a team did not attempt a conversion or kick at any time during the game. In order to address the case in which a team did not attempt a conversion/kick of any time, we set those rates, initially noted as NaN in R, to 0. This would avoid massive data loss, because in many games there are no 4th down or 2 point conversions attempted, and unless properly dealt with those entire games would be lost. In addition, in cases where there were no 3rd downs attempted and made during a game, it resulted in *NA* values in the dataset when calculating the Conversion Advantage. Since we want to avoid *NA* values during modeling, we decided it was best to omit those observations in the final dataset used for modeling.

F. Dealing with Intercorrelated Variables and Perfect Predictors

We removed the variables Year, WEEK, AWAYCITY, HOMECITY, Home_Team, Away_Team, HOME_PTS, AWAY_PTS, as they were just extra information about each game that we thought were not useful for predicting. To find the best model for prediction, we were firstly concerned about the intercorrelated variables among our predictors, so we dropped the variable `Total Yards` for both the home team and away team to avoid the distraction of multicollinear variables in the model establishment. In addition, we needed to avoid perfect predictors for points, therefore the variables TD Pass, TD Rush, FGM, EPM, 2PCM, SFTY for

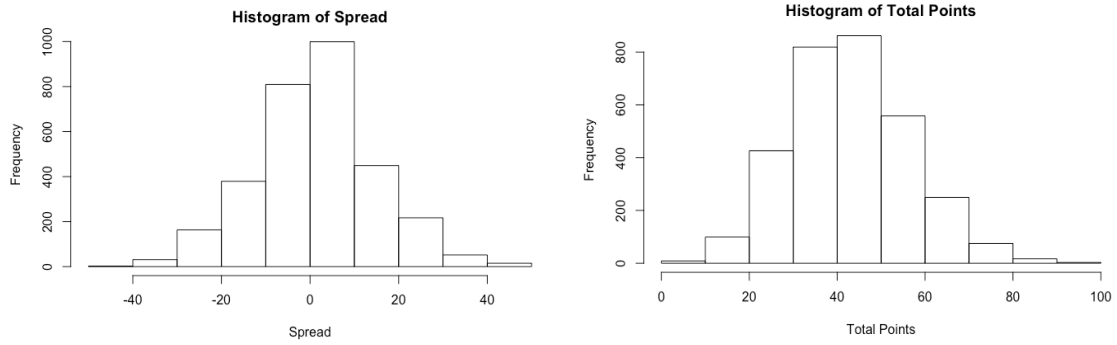
both teams were removed since the number of touchdowns, field goals, 2-point conversions, and safeties made are directly related to the number of points scored. As a result, we were left with the following variables to predict *Spread*, *Total*, and *WINLOSS*: FGA_H, EPA_H, 2PCA_H, Rush_Yards_H, Pass_Yards_H, RA_H, PA_H, Rush_YpA_H, Pass_YpA_H, INT_H, SACK_H, 3rd_PCT_H, 4th_PCT_H, COMP_H, COMP_PCT_H, FG_PCT_H, PEN_H, PEN_YDS_H, TO_H, Pass/Rush_Ratio_H, FGA_A, EPA_A, 2PCA_A, Rush_Yards_A, Pass_Yards_A, RA_A, PA_A, Rush_YpA_A, Pass_YpA_A, INT_A, SACK_A, 3rd_PCT_A, 4th_PCT_A, COMP_A, COMP_PCT_A, FG_PCT_A, PEN_A, PEN_YDS_A, TO_A, Pass/Rush_Ratio_A, elo_H, elo_A, CONV_ADV.

G. Splitting into Training and Testing Datasets for Modeling

After our dataset was cleaned so that it no longer contained any data entries NaN or NA values, our first step was to divide the data into training and testing datasets. We first set the seed to 406 after using a random number generator, in order to make sure that we have deterministic and reproducible models. We decided to randomly allot 75% of the full dataset for training and building our models. The other 25% of the full dataset was designated for testing our models. We made sure that our training and testing datasets did not overlap in order to prevent circular analysis or double-dipping. We had a total of 2982 observations, with 2236 observations used for training and 746 observations used for testing.

II. Methodology

Before we began testing our models we had to make sure the distribution of our response variables were normally distributed. Since *Spread* and *Total Points* are quantitative variables we created histograms to check their distribution.



The histograms shown above represent the distributions of *Spread* and *Total Points*. Both histograms appear to be relatively normal. Therefore, we did not need to seek out outliers to remove from our data. Since the variables are normally distributed, we believe we can make accurate models using this data.

A. Spread

Spread is the difference between the home team points and the away team points per game, so we wanted to establish a model based on the two teams' performance and points in the past and then predict *Spread* points in future games.

Methodology for Spread

We decided to build our model based on the statistics describing teams' performance and *Spread* points in games from 2010 to 2021. In the first iteration, we generated a linear regression model with all the initial variables(FGA_H, EPA_H, 2PCA_H, Rush_Yards_H, Pass_Yards_H, RA_H, PA_H, Rush_YpA_H, Pass_YpA_H, INT_H, SACK_H, 3rd_PCT_H, 4th_PCT_H, COMP_H, COMP_PCT_H, FG_PCT_H, PEN_H, PEN_YDS_H, TO_H, Pass/Rush_Ratio_H, FGA_A, EPA_A, 2PCA_A, Rush_Yards_A, Pass_Yards_A, RA_A, PA_A, Rush_YpA_A, Pass_YpA_A, INT_A, SACK_A , 3rd_PCT_A, 4th_PCT_A, COMP_A, COMP_PCT_A, FG_PCT_A, PEN_A, PEN_YDS_A, TO_A, Pass/Rush_Ratio_A, elo_H, elo_A) and the new engineered variable(CONV_ADV). Then we established the model on training data and found that the adjusted R-squared is 0.9193, which means that 91.93% of the variance for *Spread* points could be explained by the variables in the regression model. The adjusted R-squared is above 0.9 but not close to 1, so the linear regression is a good candidate but not a perfect model in prediction. Then, we predicted *Spread* in testing data based on the model and computed the root-mean-square deviation(RMSE) as 3.791178. We should figure out which model could decrease RMSE to the greatest extent.

In the second iteration, we wanted to find the nonsignificant variables by utilizing both the forward method and the backward method in the previous linear regression model. In the stepwise model, the number of variable shrinks to 26(FGA_H, EPA_H, 2PCA_H, Rush_Yards_H, Pass_Yards_H, PA_H, Rush_YpA_H, INT_H, COMP_PCT_H, FG_PCT_H, PEN_H, PEN_YDS_H, FGA_A, EPA_A, 2PCA_A, RA_A, Rush_YpA_A, Pass_YpA_A, 3rd_PCT_A, FG_PCT_A, PEN_YDS_A, TO_A, elo_H). However, the adjusted R-squared only increases to 0.9197, and RMSE increases to 3.797858. Thus, we would not choose the stepwise linear regression model due to the larger RMSE and we still used variables declared in the first iteration in the following models.

The second and third types of models we chose were Ridge regression and LASSO regression, which are both used to analyze any data that suffer from multicollinearity for more accurate prediction. The most important step in building models in Ridge regression and LASSO regression is choosing the best λ to penalize the variable coefficients, so we test a sequence of powers from -3 to 2 with 0.1 increments on base 10 to figure out the best λ with the standardized values in each variable. By 10-fold cross-validation, we found the best λ in Ridge regression was 0.025 and that in LASSO regression was 0.012. Then, we constructed the Ridge regression model and LASSO regression model with their best λ and did prediction on testing data respectively. We found that RMSE in the Ridge regression was 3.780569 and RMSE in LASSO regression was 3.765236, which were slightly smaller than RMSE in the linear regression model,

so we thought the Ridge regression model and LASSO regression model were better than the linear regression model.

Also, we tried the K-Nearest Neighbor regression model for predicting *Spread*. This model predicts the values based on the average of K nearest neighbors. We first scaled all the predictive variables in the dataset because different numerical features may have different means and variances such that they are not intuitively comparable with each other or not compatible with some statistical learning models. Then, with different k values, we would construct different models, so we needed to find the best k with the smallest RMSE. The train function was set to test k values from 3 to 20, with increments of 2. The resulting KNN regression model for *Spread* had k = 17 with the corresponding RMSE equal to 6.397804. RMSE in the best KNN regression model is much larger than those in other models, so we would not consider the KNN regression model when predicting *Spread*.

The last model we built was the polynomial regression model, and we chose to add square and cubic terms to FGA_H, FGA_A, FG_PCT_H, and FG_PCT_A. With the changes, this regression model increases the adjusted R-squared from 0.9193 to 0.9301, which means more variations in *Spread* could be explained by the polynomial regression model. RMSE for this polynomial regression model is 3.661638, which is lower than RMSE from other models. Then, we also tried backward selection based on this polynomial regression model, which reduced RMSE to 3.659473 and increased the adjusted R-square to 0.9304. Thus, this backward stepwise polynomial regression model is the best one for predicting *Spread*.

Spread	**Model**	**RMSE**
	Linear Regression (Baseline)	3.791118
	Linear Regression (Forward & Backward selection)	3.797858
	Polynomial Regression (Baseline)	3.661638
	Polynomial Regression (Backward Selection)	**3.659473**
	Ridge Regression ($\lambda = 0.025$)	3.780569
	LASSO Regression ($\lambda = 0.012$)	3.765236
	KNN Regression (k = 17)	6.397804

Best Model for Spread

After building models with linear regression, Ridge regression, LASSO regression, KNN Regression, and polynomial regression, we determined that the best model for predicting *Spread* points was the backward stepwise polynomial regression model (last paragraph in Methodology part) as it minimized the RMSE to 3.659473 in the estimation of *Spread* for testing data. RMSE as 3.659473 is small, which means most predicted values are close to the real points; thus, we

could predict *Spread* points for future games based on this backward stepwise polynomial regression model.

Additionally, this backward stepwise polynomial regression model improves adjusted R-squared to 0.9304, so we are confident that our model could predict *Spread* points for NFL games between November 8 and November 29 and further learn about two teams' performance in each game. The reason for choosing to change the degrees of FGA_H, FGA_A, FG_PCT_H, and FG_PCT_A with the square and cubic terms was the importance of the four variables. FGA is field goal attempt and FG_PCT is field goal percentage, both of which are strong determinants in football, and they largely affect the points teams are gaining, so they are potential key predictors for points scored. Also, with backward selection, we discarded some insignificant variables and improved the accuracy of prediction. After our trial, it was true that the adjusted R-square was bigger than that in linear regression, and we got the minimized RMSE in this backward stepwise polynomial regression model. Thus, we could conclude that this backward stepwise polynomial regression model is best for the prediction of *Spread* points.

Results from the Best Model for Spread Points

Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	1.677e+00	2.737e+00	0.613	0.54006	
poly(FGA_H, 3)1	8.737e+01	5.687e+00	15.364	< 2e-16	***
poly(FGA_H, 3)2	3.093e+01	4.804e+00	6.438	1.48e-10	***
poly(FGA_H, 3)3	-2.601e+01	4.131e+00	-6.297	3.65e-10	***
EPA_H	5.616e+00	8.252e-02	68.050	< 2e-16	***
`2PCA_H`	5.470e+00	1.797e-01	30.441	< 2e-16	***
Rush_Yards_H	3.439e-02	7.459e-03	4.611	4.23e-06	***
Pass_Yards_H	1.541e-02	1.721e-03	8.957	< 2e-16	***
RA_H	-5.759e-02	3.530e-02	-1.632	0.10291	
PA_H	1.367e-01	5.137e-02	2.662	0.00783	**
Rush_YpA_H	-5.108e-01	2.001e-01	-2.553	0.01076	*
COMP_H	-2.289e-01	8.726e-02	-2.624	0.00876	**
COMP_PCT_H	8.569e+00	3.105e+00	2.760	0.00583	**
poly(FG_PCT_H, 3)1	7.674e+01	5.576e+00	13.764	< 2e-16	***
poly(FG_PCT_H, 3)2	2.822e+01	4.647e+00	6.073	1.47e-09	***
poly(FG_PCT_H, 3)3	3.535e+00	3.902e+00	0.906	0.36509	
PEN_H	-7.788e-02	5.276e-02	-1.476	0.14006	
PEN_YDS_H	1.055e-02	5.341e-03	1.975	0.04835	*
poly(FGA_A, 3)1	-1.181e+02	5.786e+00	-20.419	< 2e-16	***
poly(FGA_A, 3)2	-1.865e+01	4.752e+00	-3.925	8.94e-05	***
poly(FGA_A, 3)3	2.461e+01	4.154e+00	5.923	3.66e-09	***
EPA_A	-5.639e+00	8.892e-02	-63.410	< 2e-16	***
`2PCA_A`	-5.332e+00	1.886e-01	-28.270	< 2e-16	***
RA_A	-9.912e-02	1.678e-02	-5.908	4.01e-09	***
PA_A	-9.545e-02	1.199e-02	-7.959	2.74e-15	***
Rush_YpA_A	-2.836e-01	6.242e-02	-4.543	5.85e-06	***
Pass_YpA_A	-5.141e-01	5.591e-02	-9.194	< 2e-16	***
`3rd_PCT_A`	-2.144e+00	8.523e-01	-2.515	0.01196	*
poly(FG_PCT_A, 3)1	-5.535e+01	5.571e+00	-9.935	< 2e-16	***
poly(FG_PCT_A, 3)2	-3.895e+01	4.724e+00	-8.244	2.82e-16	***
poly(FG_PCT_A, 3)3	8.624e+00	3.865e+00	2.231	0.02576	*

```
PEN_YDS_A          -7.969e-03  2.893e-03  -2.755  0.00592 **
elo_H               1.231e-03  7.890e-04   1.560  0.11885
CONV_ADV   -3.141e-01  1.335e-01  -2.353  0.01870 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.568 on 2202 degrees of freedom
Multiple R-squared:  0.9314,     Adjusted R-squared:  0.9304
F-statistic: 906.5 on 33 and 2202 DF,  p-value: < 2.2e-16
```

B. Total

Total is the sum of home team points and the away team points per game. Similarly, we want to create a model to predict the total points in future NFL games. We will choose the lowest mean absolute error (MAE) as the criteria for model selection as our objective is to minimize the MAE in our predictions of *Total* points in future games.

Methodology for Total

For *Total* points, we first started building a linear regression model with all the predictors (FGA_H, EPA_H, 2PCA_H, Rush_Yards_H, Pass_Yards_H, RA_H, PA_H, Rush_YpA_H, Pass_YpA_H, INT_H, SACK_H, 3rd_PCT_H, 4th_PCT_H, COMP_H, COMP_PCT_H, FG_PCT_H, PEN_H, PEN_YDS_H, TO_H, Pass/Rush_Ratio_H, FGA_A, EPA_A, 2PCA_A, Rush_Yards_A, Pass_Yards_A, RA_A, PA_A, Rush_YpA_A, Pass_YpA_A, INT_A, SACK_A , 3rd_PCT_A, 4th_PCT_A, COMP_A, COMP_PCT_A, FG_PCT_A, PEN_A, PEN_YDS_A, TO_A, Pass/Rush_Ratio_A, elo_H, elo_A, CONV_ADV) using the training dataset. This baseline linear regression model resulted in an adjusted R-squared value of 0.9164 and a MAE of 3.130397. The adjusted R-squared value tells us that 91.64% of the data fit the baseline linear regression model.

Following this, we performed backward selection on the baseline linear regression model. Removing predictors from the model until all of the predictors are statistically significant resulted in an adjusted R-squared value of 0.9167 and MAE of 3.143422. The resulting adjusted R-squared value is only slightly higher than the baseline linear regression model, but has a slightly higher MAE.

It is important to consider a nonlinear regression model due to *Total* being a numeric variable that possibly could be highly skewed. Using predictors from the baseline linear regression model, we also added square terms FG_PCT_H, FG_PCT_A, FGA_H, FGA_A, EPA_H, and EPA_A. We also added cubic terms for FG_PCT_H and FG_PCT_A. This polynomial regression model resulted in the increased adjusted R-squared value of 0.9249. The mean absolute error of this polynomial regression model is 2.920052, which is lower than the error of the baseline and stepwise linear regression models built for *Total* points.

Additionally, we also considered a Poisson regression model because Poisson regressions are good models for predicting counts. Since the total score is all positive values it is essentially a count variable. For the Poisson model, we did a baseline regression in which we used all of our predictors. Once we ran the regression and tested our model, we calculated that the MAE of the

Poisson regression to be 3.499497, which is higher than the MAE computed from the linear and polynomial regression models.

Similar to *Spread*, we also tried to build models with Ridge and LASSO regression. These two regression types are beneficial when modeling with data that potentially contains multicollinearity. To start, we needed to find the best λ value to adjust for the amount of coefficient shrinkage. We tested values of λ ranging from $\lambda = 10^2$ down to $\lambda = 10^{-3}$ by incrementing the power down by 0.1 to find the λ that minimizes the prediction error rate best while utilizing 10-fold cross-validation. The best λ for penalizing variable coefficients for Ridge and LASSO regression turned out to be 0.0501 and 0.0251, respectively. The respective MAE for Ridge and LASSO regression with the best λ value were 3.129131 and 3.115592. The mean absolute errors for Ridge and LASSO regression were both lower than the MAE for the linear regression models, but higher than the MAE computed for polynomial regression models.

We also tried the K-nearest neighbors regression model to predict To*tal.* Our intention was to find the best KNN regression model with the smallest MAE. After scaling the predictive variables, we used a train function that automatically performs 10-fold cross-validation. The train function was set to test k values from 3 to 20, in increments of 2. The resulting KNN regression model for *Total* had k = 15 with the corresponding MAE value equal to 7.162645. The MAE of the KNN regression model was more than double the MAE value of the baseline linear regression model, therefore, our best KNN regression model with k=17 will not be considered for the prediction of *Total.*

Total	**Model**	**MAE**
	Linear Regression (Baseline)	3.130397
	Linear Regression (Backward Selection)	3.143422
	Polynomial Regression (Baseline)	**2.920052**
	Polynomial Regression (Backward Selection)	2.928555
	Poisson Regression	3.499497
	Ridge Regression (λ = 0.0501)	3.129131
	LASSO Regression (λ= 0.0251)	3.115592
	KNN Regression (k = 15)	7.162645

Best Model for Total

After building the linear, polynomial, Poisson, Ridge, LASSO, and KNN regression models, we determined that the model that best minimized the MAE value was the baseline polynomial model without stepwise regression. The polynomial regression model minimized the

MAE to 2.920052 in the estimation for *Total* in the testing data. The adjusted R-squared value for the baseline polynomial regression model is 0.9249, which suggests that 92.49% of the data used for training fit the regression model. Since the MAE is small, it is a good indicator that the polynomial regression model can be used to predict *Total* points for the NFL games between November 8 to November 29.

Our polynomial regression model included second-degree polynomial terms for Field Goal Attempts, Field Goal Percentage, and Extra Points Attempted for the Home and Away teams. In addition, we also chose to add third-degree polynomial terms for the Field Goal Attempts for both teams. We chose FG_PCT_H, FG_PCT_A, FGA_H, FGA_A, EPA_H, and EPA_A as potential key predictors for *Total* points scored because of attempts at field goals and extra points are related to the points that can be scored during a game.

Results from the Best Model for Total Points

Coefficients:	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	2.360e+01	3.940e+00	5.988	2.47e-09	***
poly(FGA_H, 3)1	1.108e+02	6.176e+00	17.934	< 2e-16	***
poly(FGA_H, 3)2	2.688e+01	4.722e+00	5.691	1.43e-08	***
poly(FGA_H, 3)3	-2.732e+01	4.214e+00	-6.482	1.11e-10	***
poly(EPA_H, 2)1	4.142e+02	7.321e+00	56.582	< 2e-16	***
poly(EPA_H, 2)2	-1.048e+01	3.854e+00	-2.718	0.006617	**
`2PCA_H`	6.063e+00	1.964e-01	30.869	< 2e-16	***
Rush_Yards_H	4.030e-03	8.001e-03	0.504	0.614549	
Pass_Yards_H	2.318e-02	6.185e-03	3.747	0.000183	***
RA_H	1.002e-01	3.905e-02	2.566	0.010351	*
PA_H	-8.480e-02	5.587e-02	-1.518	0.129240	
Rush_YpA_H	2.502e-01	2.127e-01	1.176	0.239762	
Pass_YpA_H	-3.133e-01	2.112e-01	-1.483	0.138176	
INT_H	1.181e-01	1.395e-01	0.847	0.397150	
SACK_H	1.280e-01	5.592e-02	2.289	0.022148	*
`3rd_PCT_H`	6.603e-01	9.607e-01	0.687	0.491963	
`4th_PCT_H`	3.178e-01	2.035e-01	1.562	0.118535	
COMP_H	5.881e-02	1.096e-01	0.537	0.591470	
COMP_PCT_H	2.113e-01	3.947e+00	0.054	0.957302	
poly(FG_PCT_H, 2)1	6.727e+01	5.728e+00	11.745	< 2e-16	***
poly(FG_PCT_H, 2)2	2.599e+01	4.807e+00	5.408	7.09e-08	***
PEN_H	-1.127e-01	5.488e-02	-2.053	0.040194	*
PEN_YDS_H	1.478e-02	5.546e-03	2.665	0.007753	**
TO_H	-6.679e-01	1.131e-01	-5.905	4.08e-09	***
`Pass/Rush_Ratio_H`	3.612e-01	5.313e-01	0.680	0.496677	
poly(FGA_A, 3)1	1.268e+02	6.277e+00	20.197	< 2e-16	***
poly(FGA_A, 3)2	1.788e+01	4.722e+00	3.786	0.000157	***
poly(FGA_A, 3)3	-2.136e+01	4.199e+00	-5.086	3.98e-07	***
poly(EPA_A, 2)1	3.982e+02	7.268e+00	54.783	< 2e-16	***
poly(EPA_A, 2)2	-6.375e+00	3.890e+00	-1.639	0.101431	
`2PCA_A`	6.048e+00	2.029e-01	29.809	< 2e-16	***
Rush_Yards_A	5.340e-03	7.908e-03	0.675	0.499560	
Pass_Yards_A	5.919e-03	6.565e-03	0.901	0.367429	
RA_A	7.355e-02	4.011e-02	1.833	0.066868	.
PA_A	1.803e-03	5.833e-02	0.031	0.975341	

```
Rush_YpA_A           5.467e-02  2.018e-01   0.271 0.786467
Pass_YpA_A           1.504e-01  2.273e-01   0.661 0.508415
INT_A                4.776e-02  1.335e-01   0.358 0.720446
SACK_A               1.228e-01  5.405e-02   2.272 0.023204 *
`3rd_PCT_A`          2.724e+00  1.119e+00   2.435 0.014974 *
`4th_PCT_A`          2.461e-01  2.022e-01   1.217 0.223713
COMP_A               7.999e-02  1.195e-01   0.669 0.503450
COMP_PCT_A           1.051e-01  4.357e+00   0.024 0.980753
poly(FG_PCT_A, 2)1   6.250e+01  5.706e+00  10.955  < 2e-16 ***
poly(FG_PCT_A, 2)2   3.176e+01  4.888e+00   6.498 1.01e-10 ***
PEN_A               -1.833e-02  5.891e-02  -0.311 0.755764
PEN_YDS_A            3.222e-03  6.135e-03   0.525 0.599568
TO_A                -4.971e-01  1.111e-01  -4.473 8.12e-06 ***
`Pass/Rush_Ratio_A`  7.163e-03  5.835e-01   0.012 0.990206
elo_H                6.995e-04  8.347e-04   0.838 0.402091
elo_A                1.487e-03  8.443e-04   1.761 0.078349 .
CONV_ADV             3.879e-01  1.759e-01   2.205 0.027555 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.681 on 2184 degrees of freedom
Multiple R-squared:  0.9266,     Adjusted R-squared:  0.9249
F-statistic: 540.9 on 51 and 2184 DF,  p-value: < 2.2e-16
```

C. Result

Result is the win/loss result of a team in the game. It is a binary variable that only has values 0 or 1. If the Home Team wins, *Result* = 1. If the Home Team loses, *Result* = 0.

Methodology for Result

Since the variable *Result* is binary, it is intuitive to first apply the logistic regression to predict the game result. The dataset we used to build the model was similar to that used in predicting Spread and Total. It has variables FGA_H, EPA_H, 2PCA_H, Rush_Yards_H, Pass_Yards_H, RA_H, PA_H, Rush_YpA_H, Pass_YpA_H, INT_H, SACK_H, 3rd_PCT_H, 4th_PCT_H, COMP_H, COMP_PCT_H, FG_PCT_H, PEN_H, PEN_YDS_H, TO_H, Pass/Rush_Ratio_H, FGA_A, EPA_A, 2PCA_A, Rush_Yards_A, Pass_Yards_A, RA_A, PA_A, Rush_YpA_A, Pass_YpA_A, INT_A, SACK_A, 3rd_PCT_A, 4th_PCT_A, COMP_A, COMP_PCT_A, FG_PCT_A, PEN_A, PEN_YDS_A, TO_A, Pass/Rush_Ratio_A, elo_H, elo_A, WINLOSS, CONV_ADV (engineered variable), with WINLOSS as the variable we want to predict, and the rest of the variables as the predictors. After splitting the training set and testing set, we built the logistic regression model on the training set that contains 75% of the total observations and tested the model on the testing set. If the predicted probability is greater than or equal to 0.5, we regard the outcome as 1, and 0 otherwise. As a result, this Logistic regression model resulted in an error rate of 8.579088%. From the summary report of the current Logistic regression model, we eliminated the predictors to only those that are significant to our predictions (with low p-values), and ran Logistic regression on these variables again. The error

rate didn't increase, but the AIC increased from 1032.8 to 1098.9. Therefore, the model that contains variables before the elimination is better for Logistic regression.

Following this, we performed Ridge regression and LASSO regression. Similar to *Spread* and *Total*, we chose Ridge and LASSO regression because they are beneficial when modeling with data that potentially contains multicollinearity. The only difference in predicting the *Result* is that the response variable is a class instead of a numeric value. Therefore, when calculating the error rate, we used the same method in logistic regression to decide the predicted class for *Result*. If the predicted probability is greater than or equal to 0.5, we regard the outcome as 1, and 0 otherwise. In Ridge regression, we found the best λ value for penalizing variable coefficients to be 0.01 by testing λ values with base 10 and powers from -3 to 2 decreasing in increments of 0.1. The best λ for LASSO regression is 0.00255196. Using the best λ values, we found the best error rates of the Ridge and LASSO regression for *Result* were 9.383378% and 9.115282%, respectively, which are both higher than that of Logistic regression.

Finally, we applied the supervised learning method k-nearest neighbor (KNN) on our dataset. We chose KNN because of its easy interpretation and low calculation time. Also, with the optimal k value, we can easily make a boundary of the two classes (win or loss) that clearly segregates them from each other. After scaling the data, we conducted 10-fold cross-validation and tested the error rate for each model with different k values from 3 to 20, with increments of 2. The best KNN model occurred when k = 20, with the corresponding error rate as 13.58149%.

Result	**Model**	**Error Rate**
	Logistic Regression	**8.579088%**
	Ridge Regression (λ= 0.01)	9.383378%
	LASSO Regression (λ= 0.00255196)	9.115282%
	KNN Classification (k = 20)	13.58149%

Best Model for Result

Comparing the error rate for each of the models we applied to predict the *Result*, Logistic regression has the best performance as its error rate is the lowest, with the value of 8.579088%. Therefore, when predicting the future game result, we can use Logistic regression on the predictor dataset and get our predictions. From the summary on the Logistic regression model on the training set, we found that FGA_H, EPA_H, 2PCA_H, FG_PCT_H, FGA_A, EPA_A, 2PCA_A, FG_PCT_A are the most significant predictors in predicting the *Result,* which makes sense as scoring the field goal is directly related to the points a team gets.

Results from the Best Model for Result

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
```

```
(Intercept)         -4.6234222  4.6254706  -1.000   0.3175
FGA_H                0.5761746  0.0943979   6.104 1.04e-09 ***
EPA_H                2.5810093  0.1883743  13.701  < 2e-16 ***
`2PCA_H`             2.1031769  0.2408106   8.734  < 2e-16 ***
Rush_Yards_H         0.0039645  0.0099130   0.400   0.6892
Pass_Yards_H         0.0028078  0.0064506   0.435   0.6634
RA_H                 0.0780708  0.0474055   1.647   0.0996 .
PA_H                 0.0132034  0.0698270   0.189   0.8500
Rush_YpA_H           0.1022642  0.2561450   0.399   0.6897
Pass_YpA_H           0.1349952  0.2232773   0.605   0.5454
INT_H                0.1920760  0.1526687   1.258   0.2083
SACK_H               0.0749674  0.0596876   1.256   0.2091
`3rd_PCT_H`          1.4488038  1.1658511   1.243   0.2140
`4th_PCT_H`          0.2613004  0.2149581   1.216   0.2241
COMP_H               0.0268560  0.1228063   0.219   0.8269
COMP_PCT_H           0.7984566  4.6464417   0.172   0.8636
FG_PCT_H             1.6590037  0.2640641   6.283 3.33e-10 ***
PEN_H                0.0118092  0.0568427   0.208   0.8354
PEN_YDS_H           -0.0030591  0.0057531  -0.532   0.5949
TO_H                -0.2093540  0.1182340  -1.771   0.0766 .
`Pass/Rush_Ratio_H` -0.5179156  0.6901743  -0.750   0.4530
FGA_A               -0.7093412  0.0949047  -7.474 7.76e-14 ***
EPA_A               -2.4385347  0.1865335 -13.073  < 2e-16 ***
`2PCA_A`            -2.5613858  0.2555225 -10.024  < 2e-16 ***
Rush_Yards_A        -0.0059534  0.0105891  -0.562   0.5740
Pass_Yards_A         0.0014656  0.0081531   0.180   0.8573
RA_A                -0.0187925  0.0521159  -0.361   0.7184
PA_A                 0.0181468  0.0672967   0.270   0.7874
Rush_YpA_A           0.0231194  0.2731436   0.085   0.9325
Pass_YpA_A          -0.4813641  0.3029654  -1.589   0.1121
INT_A                0.1608243  0.1481023   1.086   0.2775
SACK_A               0.0660742  0.0596863   1.107   0.2683
`3rd_PCT_A`         -2.5723694  1.3194586  -1.950   0.0512 .
`4th_PCT_A`          0.0593730  0.2101710   0.282   0.7776
COMP_A              -0.1211567  0.1396465  -0.868   0.3856
COMP_PCT_A           7.0530854  5.3045948   1.330   0.1836
FG_PCT_A            -1.7376945  0.2810149  -6.184 6.26e-10 ***
PEN_A                0.0204007  0.0624978   0.326   0.7441
PEN_YDS_A           -0.0063585  0.0064842  -0.981   0.3268
TO_A                -0.0348437  0.1210998  -0.288   0.7736
`Pass/Rush_Ratio_A` -0.4195397  0.9831086  -0.427   0.6696
elo_H                0.0017210  0.0009130   1.885   0.0594 .
elo_A               -0.0003427  0.0008990  -0.381   0.7030
CONV_ADV            -0.3400877  0.2509311  -1.355   0.1753
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3056.64  on 2235  degrees of freedom
Residual deviance:  944.83  on 2192  degrees of freedom
AIC: 1032.8

Number of Fisher Scoring iterations: 8
```

Confusion Matrix of Logistic Regression for *Result* on Testing Set

n = 746		**Predicted**	
		Loss(0)	**Win(1)**
Actual	**Loss(0)**	284	32
	Win(1)	32	398