Final Paper

Zach Brazil, Richard Charles, and Adam Kiehl

12/16/21

Introduction

In an attempt to predict the top fantasy point performers in the NFL, data from the 2018-2021 NFL regular seasons were scraped from pro-football-reference.com (Sports Reference (2021)), using 2018-2020 data as our training data while 2021 was used for testing. Variables that were deemed point-worthy statistics were used as the response variables while those remaining were used as predictors. Below is a table that shows what the response variables are, and the amount of fantasy points gained for each one. For example, one passing touchdown would net 4 fantasy points while one passing yard would net .04 fantasy points.

Stat	Pts
PassYds	0.04
PassTD	4.00
PassInt	-1.00
RushYds	0.10
RushTD	6.00
Rec	1.00
RecYds	0.10
RecTD	6.00
FL	-2.00

Using a wide array of models, each point-worthy statistic was used as a response variable and the model with the lowest mean-squared error for that specific statistic was used against the testing data for the most accurate predictions.

Passing Analysis

Our response variables, those that had some form of fantasy point value, were `PassYds` (Passing Yards), `PassTD` (Passing Touchdowns), `PassInt` (Passing Interceptions) and `FL` (Fumbles Lost). All predictors that weren't related to throwing the ball were removed from the training dataset. Out of the remaining variables, `CMP` (Completions) were used to filter out non-quarterbacks in the dataset. Typically because most other positions don't throw the ball or if they do it's extremely rare so we deemed it a safe assumption to make as far as removing those players from the training data. Other predictors were removed due to multicolinearity. In this instance, CAY (Completed Air Yards) and YAC (Yards After Catch) were perfectly correlated and when these two variables were added together we would get the total passing yards for the quarterback on each play. We avoided having bias in our data by removing `PassYAC`, thus avoiding perfect correlation between our predictors. Lastly, the predictor variable `FirstDPass` (First Down Pass) contained NA values so for easy intuition and to avoid errors in the future we changed each NA value to 0.

Cmp	PassAtt	PassYds	PassTD	PassInt	Sk	YdsLost	PassLng	Rate	FL	FirstDPass
21	30	255	2	0	3	8	52	118.1	0	13
20	34	153	0	1	2	8	27	57.6	1	6
24	32	354	1	1	2	12	57	108.1	0	14
17	27	90	0	1	1	7	15	53.0	0	4
4	7	36	0	1	1	7	10	31.5	0	2
24	35	220	0	1	3	26	39	73.5	1	11

FirstDPassPer	CAY	YACPerCmp	PassDrops	BadThrow	BadPer
39.4	75	8.6	1	5	18.5
16.7	87	3.3	4	5	15.2
41.2	205	6.2	0	2	6.9
14.3	41	2.9	0	5	19.2
25.0	37	-0.3	0	1	16.7
28.9	111	4.5	0	6	17.1

Models

Several models were used and each response variable was tested for train mean-squared error (MSE) in each model. These models include multiple linear regression (MLR), LASSO, Principal Component Regression/Analysis (PCR/PCA), Trees and Bagging. After each model was run, the resulting MSE was placed into a data-frame that stored all MSE values from every model for easy comparisons at the end to decide which model had the best train MSE for each respective response variable. The predictor variables were those seen in the above `train_data` data-frame minus the response variables (`PassYds`, `PassTD`, `PassInt`, `FL`).

Linear Regression

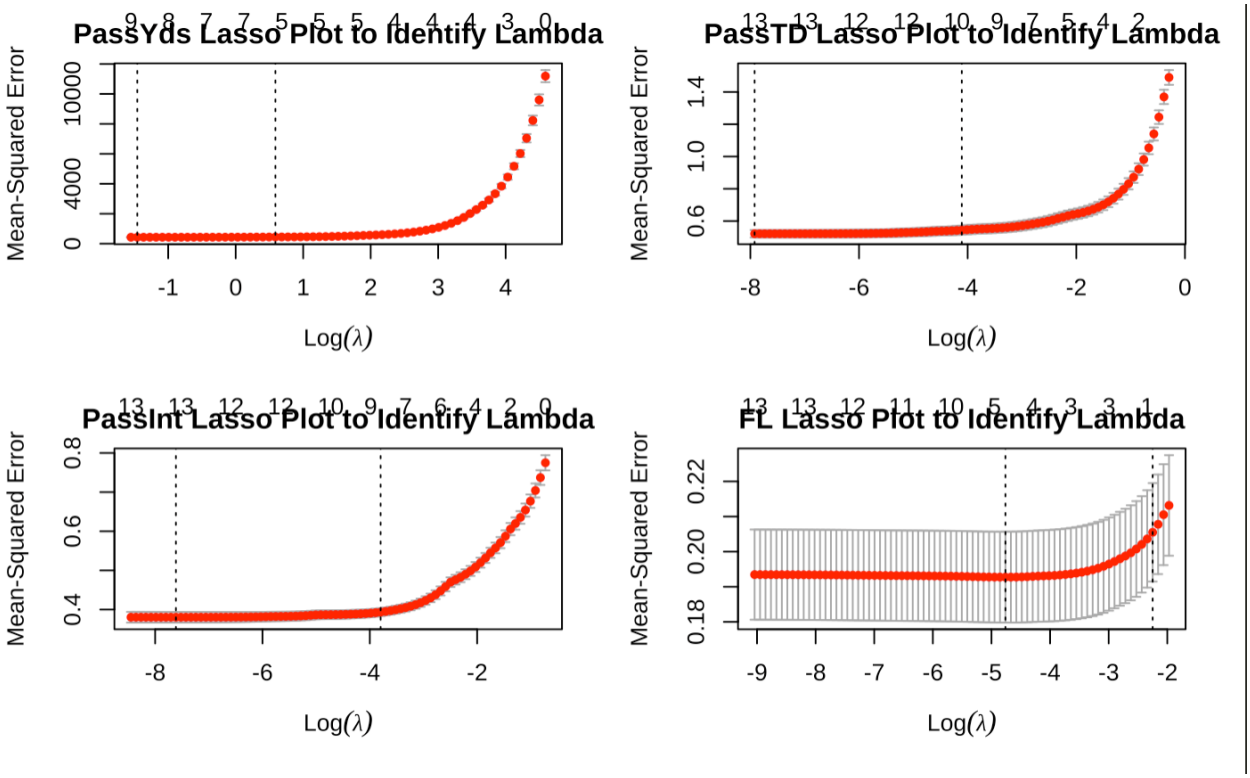
The first model used was Multiple Linear Regression. We split the data randomly into k-folds, 10 in this case and going forward, and used k-fold cross validation. We ran a for loop the length of k (10) holding out one fold and using the other nine as our training set for each of the linear regression models for each response variable. Four models were run with every iteration (one for each of the four response variables) until no folds were left to be tested on. We then predict the value of our response using whichever k-fold as our test set and calculate the MSE by subtracting the prediction value by the true value of the response and squaring the result. This was done for every model and the MSE values were stored into an empty data-frame and will be used later on. Below is the table of MSE from MLR.

model	MSE
Pass Yards	419.9868249
Pass TD's	0.5127375
Pass Int	0.3738556
Fumbles Lost	0.1911845

Seeing the MSE for each model, `FL` had a decent MSE meanwhile the others left some to be desired, especially `PassYds`. However, passing yards in the NFL is an extremely variable statistic and is very difficult to predict.

LASSO

Each response variable was fit with a LASSO regression in hopes to lower MSE while shrinking insignificant predictors down to 0. The plots below show where the choice of lambda came from. Wherever the elbow was, was the value chosen.

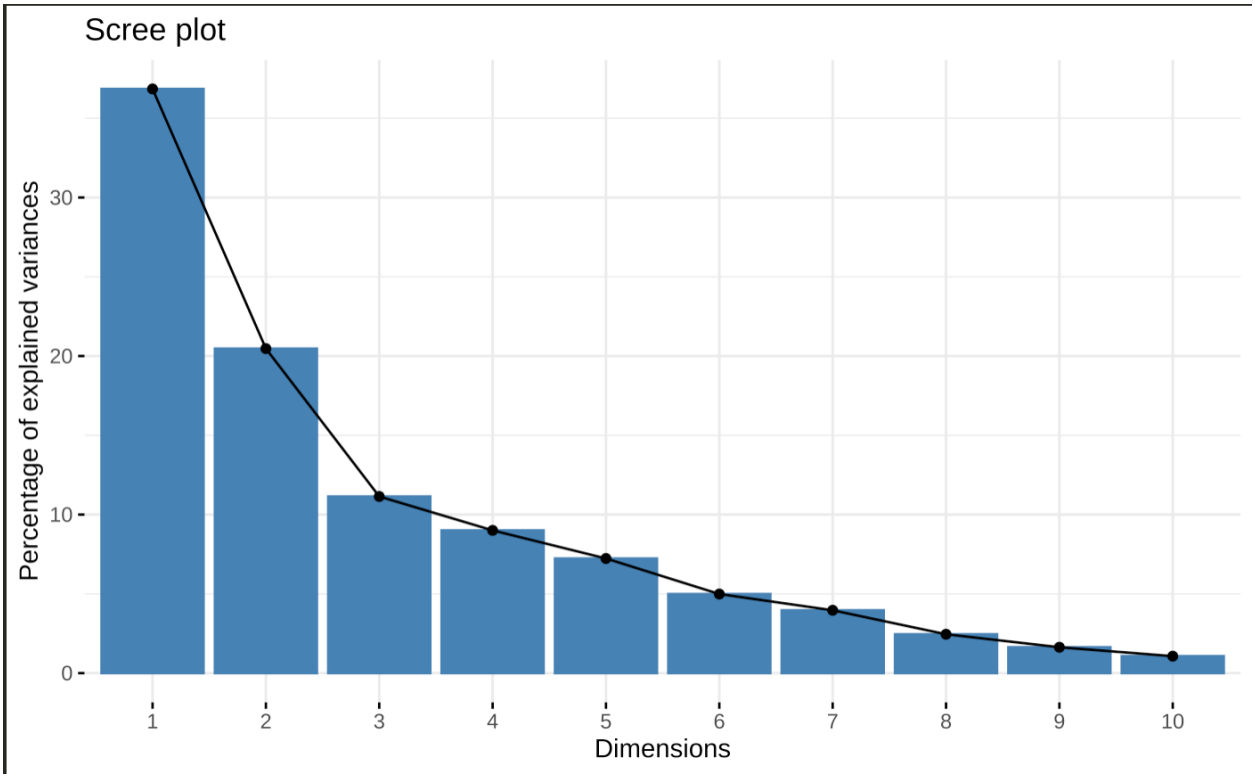


By randomly splitting the data into k-folds with k = 10, we can use one fold as the validation set. Each response is fitted, and the respective model is predicted against the validation set. The MSE is calculated for each k-fold and the mean is taken. The MSE was computed in a similar manner to that of Linear Regression. The resulting MSE values for each model are put into an empty data-frame that will be used later.

Model	MSE
PassYds	568.528
PassInt	0.423
PassTD	0.647
FL	0.192

PCA/PCR

Before we can use PCR to get some predictive analysis for our responses, we perform Principle Component Analysis on our data to get a better understanding of the overall structure of our training data. The resulting summary table from our PCA shows us how many components were necessary to explain at least 90% of our variance, and follow that up with a scree plot to indicate a which principle component is the best for explaining enough variance, meanwhile keeping the complexity of the model to a minimum.



At x principle components, approximately 90% of our variance is explained which does make for difficult intuition as we can't visualize x-dimensional data:

Afterwards, we calculate the PCA scores of each predictor variable to see which predictor has the greatest influence on each principle component. The graph is the best 2D representation of what is happening in several dimensions. It is very hard to conceptualize what is happening in PCA when more than three components are required to explain the majority of the variance.

	PC1	PC2	PC3
Cmp	-0.4084061	0.1746255	-0.0150215
PassAtt	-0.4360607	0.0574051	-0.0912329
Sk	-0.2295970	-0.2607229	0.5871378
YdsLost	-0.2187772	-0.2534627	0.6038598
PassLng	-0.1979753	0.2824285	-0.0055927
Rate	0.0444353	0.5103738	0.1746902
FirstDPass	-0.3827883	0.2661923	-0.0336993
FirstDPassPer	0.1752667	0.4076545	0.1077158
CAY	-0.3665943	0.2498426	-0.0332321
YACPerCmp	0.1041752	0.1822223	0.0563775
PassDrops	-0.2099962	-0.0290752	-0.0265771
BadThrow	-0.3372540	-0.1752095	-0.3027475
BadPer	-0.1448471	-0.3599068	-0.3770172

The matrix above informs which predictor explained the most variance along each principal component.

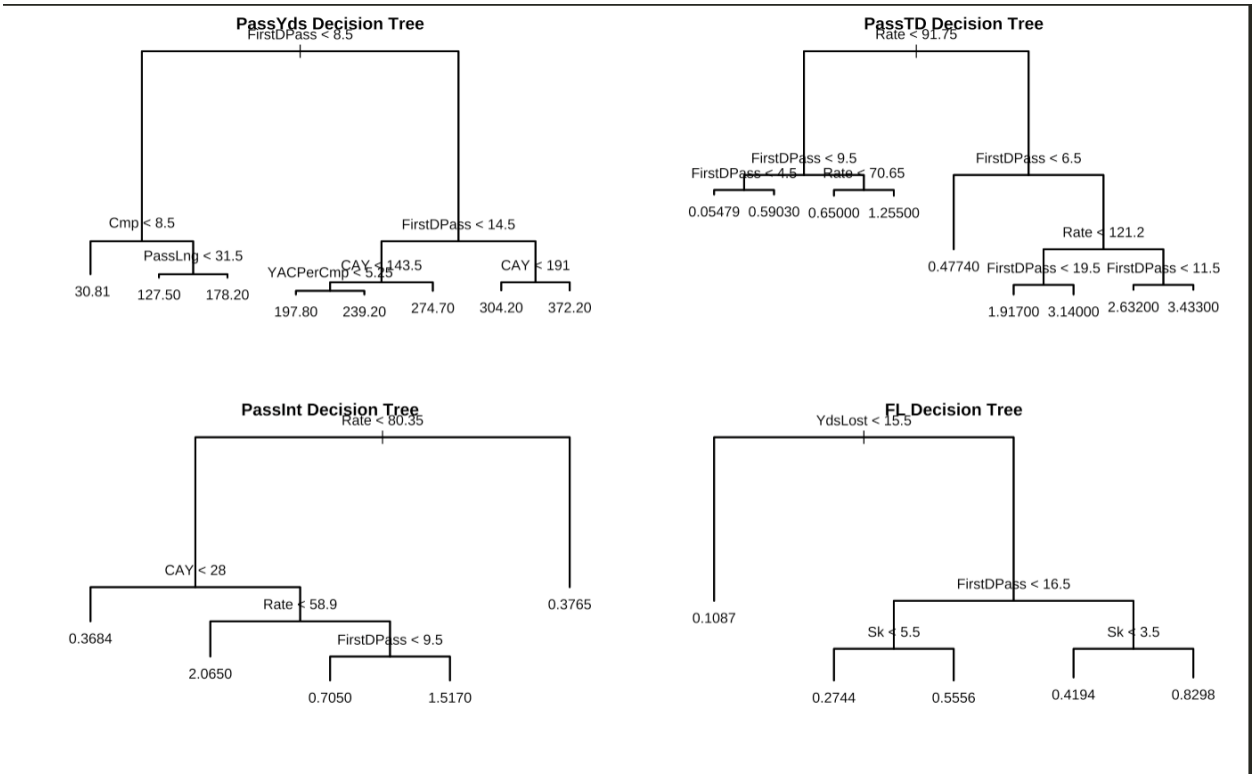
Now that we have our PCA scores, we can use these scores to train our PCR models. Again, our training data is randomly split into k-folds, where k = 10, every fold is held out once to be used as the testing data. We then calculate MSE of each by pulling the true value of each response from the validation set and subtracting the predicted value from it, squaring it and taking the mean. These values were again stored in an empty data-frame containing all MSE values for each response variable for their respective PCR model.

Model	MSE
PassYds	691.1332500
PassInt	0.6556087
PassTD	0.7882322
FL	0.1926117

Trees

The next method used were trees. First, we fit an unpruned tree model to each of our response variables and by using the `cv.tree()` function, were able to get the optimal size for each tree.

Once optimal tree sizes were indicated, we pruned each tree to their respective sized and using the `prune.tree()` function we were able to run our original tree model again but using only the optimal amount of nodes. The trees for each response are below and follow the same order graphically as listed above.



Afterwards, we finally begin splitting the data up using k-folds with k = 10. Each model is trained with their optimal tree size and the MSE is calculated similarly to the previous models and placed in an empty data-frame containing all MSE values for each response variable for their respective tree model.

Model	MSE
PassYds	1640.746
PassInt	0.510
PassTD	0.609
FL	0.201

The final method used for the passing statistics was bagging. Bagging provided by far the best results for some response variables which we will see. First, we started by once again splitting our data into k-folds with k = 10. Each response variable was run with each fold being used as the validation set once, and a bagging model was fit. Afterwards, using the `importance()` function, we could see which predictors were

significant to the response variable and based off these results and new bagging model was fit with the most significant predictors only, having the insignificant ones removed from the model altogether.

For `PassYds`, each of the other response variables were taken out of the model along with `PassDrops`, `BadThrow`, `BadPer`, `Rate`, `Sk`, `FirstDPassPer` and `YdsLost` as they weren't deemed significant.

For `PassInt`, each of the other response variables were taken out of the model along with `PassDrops`, `Sk`, `YACPerCmp`, `BadPer`, `PassAtt` and `YdsLost` as they weren't deemed significant.

For `PassTD`, each of the other response variables were taken out of the model along with `PassDrops`, `Sk`, `YACPerCmp`, `FirstDPassPer`, and `YdsLost` as they weren't deemed significant.

For `FL`, each of the other response variables were taken out of the model along with `PassDrops` as it wasn't deemed significant.

Model	MSE
PassYds	161.575
PassInt	0.305
PassTD	0.469
FL	0.212

Results

The resulting MSE values for each response variable were put into one matrix for easy comparisons so we could take the best model for each response and use that model to predict on the testing data set. The matrix is below:

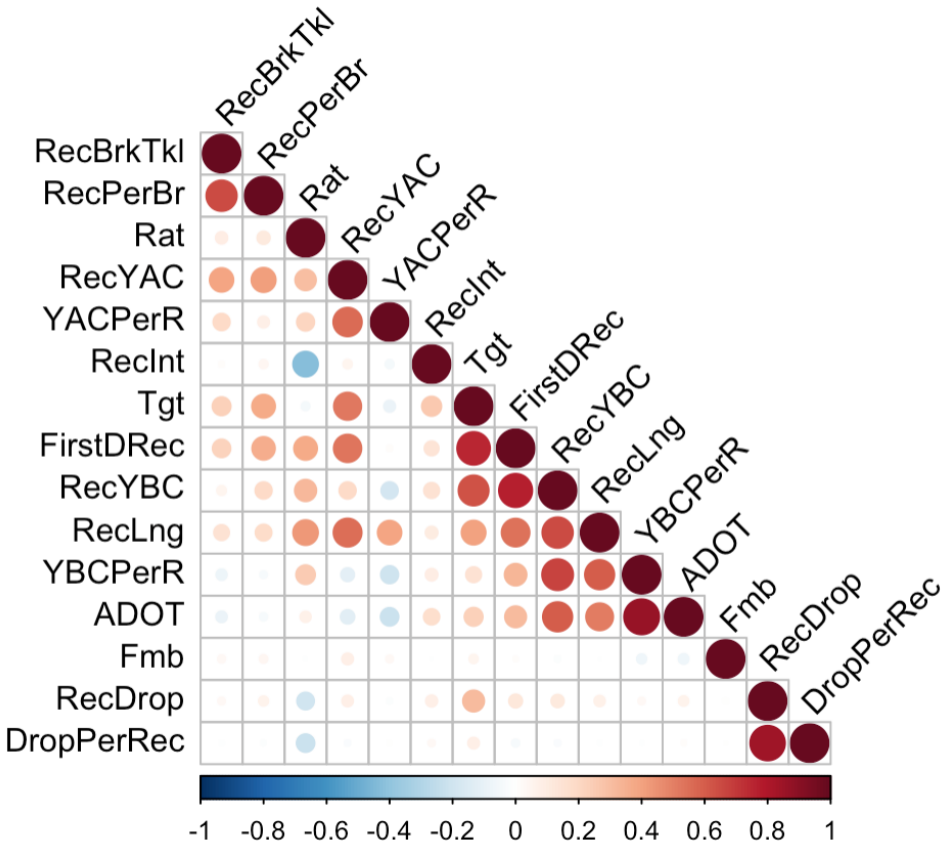
Model	MLR	LASSO	PCR	Tree	Bagging
PassYds	419.9868249	568.528	691.1332500	1640.746	161.575
PassInt	0.5127375	0.423	0.6556087	0.510	0.305
PassTD	0.3738556	0.647	0.7882322	0.609	0.469
FL	0.1911845	0.192	0.1926117	0.201	0.212

Receiving Analysis

In terms of receiving, players can be awarded points for a reception (`Rec`; worth 1 point), a receiving yard (`RecYds`; worth .1 points), and a receiving touchdown (`RecTD`; worth 6 points). The variables of interest were modeled with the full set of standard and advanced receiving metrics offered by pro-football-reference.com. These were `Tgt`, `RecLng`, `Fmb`, `FirstDRec`, `RecYBC`, `YBCPerR`, `RecYAC`, `YACPerR`, `ADOT`, `RecBrkTkl`, `RecPerBr`, `RecDrop`, `DropPerR`, `RecInt`, and `Rat`. The original training data set with 2018-2020 data was filtered to include only player-game observations with at least one reception. The resulting training set contained $n = 11,079$ observations and $p = 15$ predictors. Due to the large p utilized, the major goal of this analysis was to develop highly predictive models while maintaining a high degree of simplicity and interpretability. All models were validated using 10-fold cross-validation.

Regression Models

The first and most basic class of models considered in this analysis was multiple linear regression. One of the major assumptions of multiple linear regression forbids multicollinearity between predictors. To discover highly correlated predictors, a correlation matrix was used in conjunction with variance inflated factor (VIF) scores. Of particular interest are the strong linear relationships found between `Tgt` and `RecYBC`, and `FirstDRec` (0.75 and 0.77), `ADOT` and `YBCPerR` (0.86), and `RecLng` and several other predictors including `RecYAC` and `FirstDRec`. Additional high correlations were found between metrics and their averages per reception (like `RecYBC` and `YBCPerR`, for example).



Particularly high VIF scores were identified for `RecLng` (7.3), `RecYBC` (8.1), `YBCPerR` (6.8), and `RecYAC` (6.1). Removing `RecLng`, `FirstDRec`, `RecYBC`, `YBCPerR`, `DropPerR`, and `YACPerR` yielded reasonable cross-predictor correlations and VIF scores. A regression model was fit for each of `Rec`, `RecYds`, and `RecTD` using the nine remaining predictors. The one exception being for the `RecYds` model, `RecYBC` was included instead of `RecYAC` because of it's stronger relationship with the response (and both couldn't be included because $RecYds = RecYBC + RecYAC$). While the regression models performed decently with $MSE_{rec} = 0.51$, $MSE_{recYds} = 64.83$, and $MSE_{recTD} = 0.09$, the normality assumption is violated and so these models were not considered viable and were used merely as benchmarks.

Next, variable selection techniques were applied to the above regression framework to identify important predictors since all the predictors in each model were considered statistically significance at $\alpha = 0.05$. Since the assumptions of regression are not met for this problem, these analyses were performed merely for comparison with other selection techniques used throughout the project.

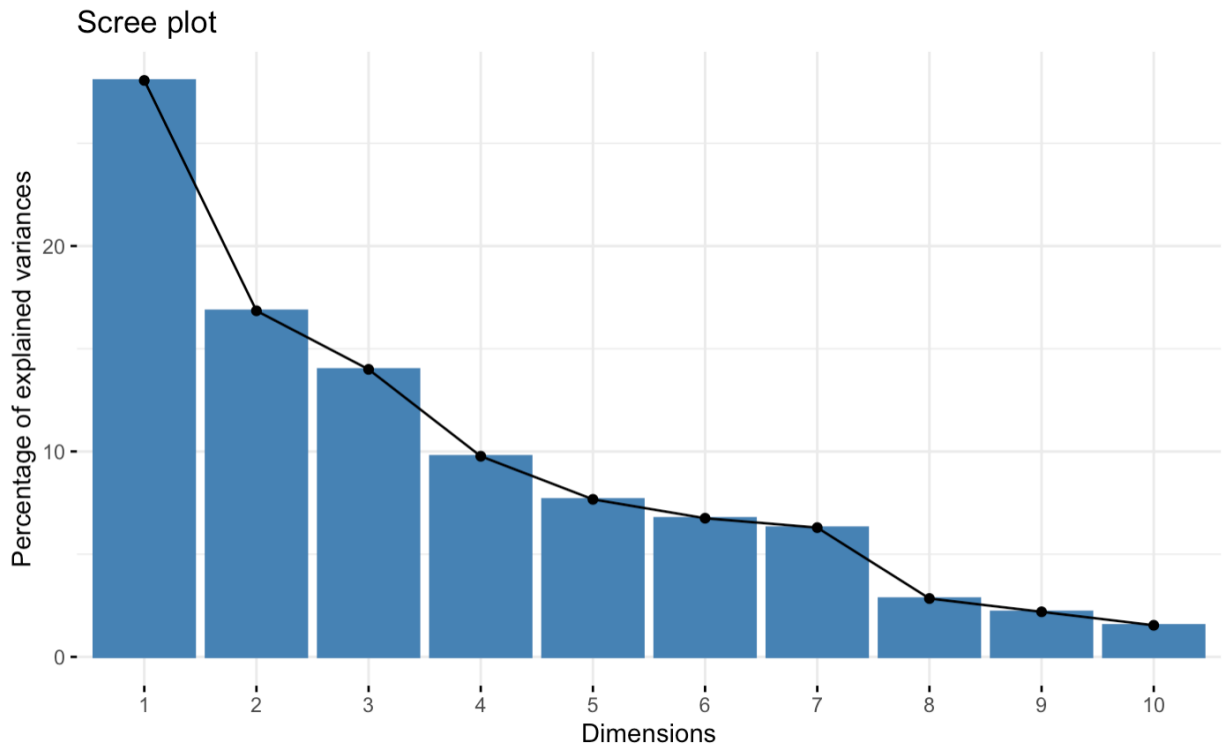
The first selection technique used was a best subset selection. Based on Mallows' Cp, the Bayesian information criterion (BIC), and adjusted R^2, best subset chose an optimal subset of predictors with the goal of predictive accuracy. The `Rec` model chose four predictors: `Tgt`, `ADOT`, `RecDrop`, and `Rat`. Rationally, a player's receptions must be equal to their targets less passes dropped and missed passes; therefore, $Rec \leq Tgt - RecDrop$, and `ADOT` and `Rat` both measure a quarterback's passing ability, predicting missed passes. The `RecYds` model chose three predictors: `Tgt`, `RecYBC`, and `Rat`. Receiving yards is here defined as a function of targets and the total yards gained by each reception with `Rat` simulating completion percentage to adjust targets to real receptions. The `RecTD` model also chose three predictors: `Tgt`, `RecInt`, and `Rat`. Strangely, `RecInt` is by far the most predictive variable in this model; perhaps it is suggesting that end zone passes are high variance and that they often result in touchdowns but also, conversely, interceptions. The best subset selection models performed relatively equivalent to the multiple linear regression models with $MSE_{rec} = 0.53$, $MSE_{recYds} = 71.77$, and $MSE_{recTD} = 0.11$. For the purposes of this study, the

interpretability of the best subset models outweighs the trade off in MSE when compared to the previous regression models.

The second selection technique used was LASSO regression. Based on a 10-fold cross-validation MSE, LASSO regression chose an optimal subset of predictors with the goal of predictive accuracy. The `Rec` model chose the four previous predictors along with `RecYAC` and `RecPerBr`, both with low predictive weight. The `RecYds` model chose the three previous predictors along with `RecBrkTkl` and `RecPerBr`; these both make sense as predictors seeing as receivers that can break tackles after the catch have more opportunities to gain yards. The `RecTD` model chose seven predictors with `RecInt` still standing out as the only highly predictive one. The LASSO regression models did not perform any better than the best subset selection models with $MSE_{rec} = 0.98$, $MSE_{recYds} = 153.48$, and $MSE_{recTD} = 0.09$, and were only less interpretable.

Principal Component Analysis

The next modeling technique applied was a principal component analysis with the objective of removing multicollinearity and identifying interpretable combinations of predictors. Using a scree plot, the first seven principal components were identified as important sources of variance explained with a cumulative 89.4% explained.



The first principal component was primarily defined by `Tgt`, `RecLng`, `FirstDRec`, and `RecYBC` and seems to represent some sort of aggregation of receptions and reception distance. The second principal component was primarily defined by `RecYAC`, `YACPerR`, and `RecBrkTkl` and represents a receiver performance after the ball is caught. The third principal component was negatively defined by `RecDrop` and `DropPerR` and represents the catching abilities of a receiver. The sixth principal component was dominated by the `Fmb` predictor. Despite the uninterpretability of the remaining principal components, they were included in the modeling process since they explain a significant amount of variance. Using 10-fold cross-validation, principal component regression models were fitted for the variables of interest and an optimal number of principal components was chosen for each. The `Rec` and `RecYds` models chose seven principal components

as would be suggested by the scree plot above. The `RecTD` model added an eighth principal component. While the principal component regression did not perform well for `Rec` and `RecTD`, with $MSE_{rec} = 0.61$, $MSE_{recYds} = 47.47$, and $MSE_{recTD} = 0.15$, it did yield intriguing predictive power for `RecYds`. Ultimately, principal components regression yielded some interesting results and predictor groupings but struggled with prediction and interpretation.

Tree-Based Models

The final class of models applied to the training data was tree-based models. Regression trees were fit for each variable of interest using all predictors and an optimal number of terminal nodes was chosen using cross-validation. For the `Rec` model, only `Tgt` and `Rat` were selected and eight terminal nodes were used. This differed somewhat from the regression subset previously chosen but it follows a similar concept by adjusting total targets by a proxy for completion percetnage in `Rat`. The `RecYds` model chose seven terminal nodes and selected `RecYBC`, `FirstDRec`, `RecLng` and `YACPerR`. While the regression subset was also dominated by `RecYBC`, the other three chosen predictors here were removed for multicollinearity in prior models but do make sense in context of the response. The `RecTD` model was the simplest of all, only using two terminal nodes and `Rat` for prediction. The metric `Rat` here measures a quarterback's performance when throwing to a given receiver and weights touchdowns heavily. This could explain why `Rat` was found to be strongly predictive of touchdowns. The regression tree models achieved $MSE_{rec} = 1.03$, $MSE_{recYds} = 191.37$, and $MSE_{recTD} = 0.10$. The interpretability of these results was much improved over principal components regression but the predictive power was too low for use in testing. Further tree-based methods were applied to improve accuracy.

Boosting was first applied using 100 trees and a 10-fold cross-validation tuned shrinkage factor. While interpretability was diminished due to the added complexity of the models, predictive accuracy was stronger across the board. The boosted models achieved $MSE_{rec} = 0.40$, $MSE_{recYds} = 50.25$, and $MSE_{recTD} = 0.09$. Next, bootstrap aggregation or 'bagging' was applied with the expectation that both predictive accuracy and interpretability would be improved. The models all selected the same important predictors as the regression tree models but performed significantly better predictively, achieving $MSE_{rec} = 0.15$, $MSE_{recYds} = 17.92$, and $MSE_{recTD} = 0.05$. A similar method, random forest, was also applied. For `RecYds` and `RecTD`, the random forest selected the same predictors as the bagged tree models but performed slightly worse. The `Rec` model, conversely, chose `Tgt`, `FirstDRec`, `RecYAC`, and `RecYBC` which makes intuitive sense as $FirstDRec \leq Rec \leq Tgt$. This model performed better than its bagged tree counterpart with $MSE_{rec} = 0.13$. The extensions to the standard regression tree proved to increase the predictive power of the tree models drastically.

Receiving Results

The ultimate goal of this analysis was to develop a model with high predictive accuracy but also reasonable interpretability. The multiple linear regression models along with the best subset selection models and the LASSO regression models provided a benchmark for interpretation and variable selection but violated assumptions prevent these models from being used for testing. The principal component analysis, while especially intriguing when predicting `RecYds`, included too many predictors and could only be vaguely interpreted. Building off the intuitive regression tree models developed, boosting, bagging, and random forest generated highly predictive tree-based models. The 10-fold cross-validation MSEs for each model are reproted below.

Model	MLR	Subset	LASSO	PCR	Tree	Boost	Bag	RF
Rec	0.513	0.532	0.976	0.607	1.029	0.398	0.153	0.131
RecYds	64.832	71.766	153.483	47.466	191.374	50.249	17.916	22.780
RecTD	0.094	0.107	0.095	0.153	0.104	0.086	0.047	0.057

To predict `Rec`, a final random forest model was fit using `Tgt`, `FirstDRec`, `RecYAC`, and `RecYBC`. To predict `RecYds`, a final bagged tree model was fit using `RecYBC`, `FirstDRec`, `RecLng` and `YACPerR`. To predict `RecTD`,

a final bagged tree model was fit using only `Rat`.

Rushing Analysis

The rushing ability of a football team is a valuable component of a team that can sometimes go overlooked. When looking at a fantasy team, rushing points can make or break a season. The part of this project regarding rushing was one that certainly contributed to the validity of it. The first important part of rushing yard analysis was finding valuable predictors. The predictors that were expected to be of value were pulled and there were initially many. Before predictor significance was verified there were about 12 predictors. After verifying significance, and accounting for collinearity, the most significant predictors were `RushYds`, `RushTD`, and `FL`. One example of a predictor that was redundant was `FirstDRush`. This variable does not lead to fantasy points and relates closely with `RushYds`.

Models

Creating models for these predictors was an interesting process with helpful results. Obviously some models were more accurate than others, but it was beneficial to see how the same 3 predictors can be used in different models to produce different results. The models that were used to test rushing were Multiple Linear Regression, Lasso, PCR, Bagging, and Boosting. PCR ended up giving values that were not as useful for our rushing experimentation. This may have been able to be attributed to user error but when the team looked at the input and output the problem was not clear. Since the PCR values were not helpful they ended up not being used. Of all the models; multiple linear regression provided the best MSE values for `RushTD` and `FL` (Fumbles lost). Bagging gave the best value for `RushYds`. In the end, these models were the ones that were selected for final testing.

```
##     Model          MLR   LASSO          PCR   Boost     Bag
## 1 RushAtt  14.81595623  14.881 4.654656e-26  13.816  15.121
## 2 RushYds 168.65983529 177.579 1.077096e-25 170.551 118.468
## 3  RushTD   0.20393790   0.204 6.106743e-05   0.248   0.290
## 4      FL   0.08361117   0.084 3.365506e-04   0.091   0.107
```

Multiple Linear Regression

Multiple Linear Regression was the perfect place to start for analysis of the rushing data. In order to complete the multiple linear regression model the data was split randomly into k-folds. Once again k = 10 in this context. A for loop was run with each response variable. In this case the response variables were `RushTD`, `RushYds`, and `FL`. Four models were run with every iteration (one for each of the four response variables) until no folds were left to be tested on. Each from each model an MSE was calculated and that value was used later for assessing models. For some variables with this model, the MSE values were undesirable and did not seem vary helpful. The MSE values for `RushTD` and `FL` ended up being solid and when compared to later models, the values were best.

Bagging

Bagging, or bootstrap aggregation was another successful model for the rushing statistics. This model was done as a tree-based model following boosting and allowed for the boosting result to be more interpretable and useful in the analysis. The result of bagging in the rushing analysis led to one of the best MSE values and for the variable `RushYds` the MSE value was better than all the other models previously used. The MSE value for `RushYds` was solid ended up being used for final testing.

Fumbles Analysis

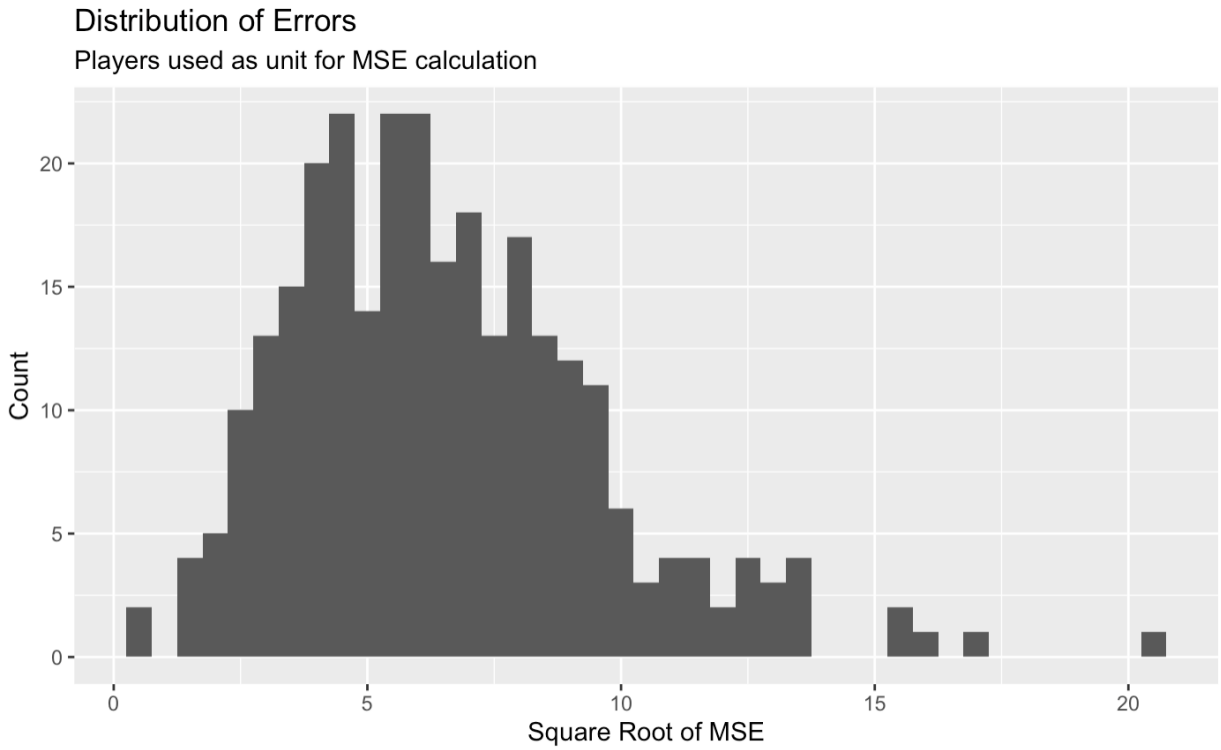
An important factor of building a fantasy football team is the number of fumbles lost. Fumbles lost is the count of fumbles a player has that occurs in a turnover. This is an important factor because a fumble lost results in a negative point for a fantasy player. Analyzing the fumbles lost variable was cool because models were being built for the same variable in multiple ways. There were 3 different groups of predictors for each model. Fumbles lost was predicted with receptions predictors, passing predictors, and rushing predictors. The best fumbles lost model for receptions predictors was a tree model with an MSE of .028. The best fumbles lost model for passing predictors was a MLR model with an MSE of .191. The best fumbles lost model for rushing predictors was a MLR model with an MSE of .084. After viewing all of the best models, it was determined that the best model way to predict fumbles lost is to use the tree model for receptions data.

Results

This project's ultimate objective was to develop a pre-season ranking of players for use in a fantasy draft. The models developed were designed to accept a set of advanced statistics and return an estimate of the fantasy performance associated with that stat line. Since models were not developed to predict individual players' progressions over time, a constant "typical" game performance was estimated for each player for use in prediction. To this end, each player's advanced training metrics were averaged across their last 17 games played (only players with at least 17 games in our training set were included in this validation step). These averages were then supplied to the models to obtain a "typical" fantasy performance for that player. Predictions were rounded to the nearest integer since the response was discrete. This value was then compared to that players' 2021 season fantasy performance for testing, and MSE was calculated per player.

	Player	Pos	fanPts	MSE
223	Lamar Jackson	QB	30.00	121.60538
65	Christian McCaffrey	RB	26.14	151.05817
222	Kyler Murray	QB	25.56	67.35529
282	Patrick Mahomes	QB	24.14	98.37873
307	Russell Wilson	QB	23.64	87.11782
192	Josh Allen	QB	22.54	71.40160
77	Dak Prescott	QB	22.46	67.77287
339	Travis Kelce	WR	22.40	87.56667
134	Gardner Minshew II	QB	21.80	232.68500
107	Derek Carr	QB	21.46	45.44943
4	Aaron Rodgers	QB	21.42	62.46978

The above results concur strongly with the true top fantasy performers in the NFL. Positional top ten rankings were compared to ESPN's 2021 preseason fantasy rankings (Entertainment and Sports Programming Network (2021)) and seven each of QBs, WRs, and RBs were found in common. The discrepancies can be explained by player age, career progression (whether a starting job was earned for 2021), and simple over or under performance. MSE was calculated for each player and square rooted for direct comparison to `fanPts`. The median square rooted MSE for all players was 6.1 while the standard deviation of `fanPts` in the testing set was 8.1. Median was here used as a measure of central tendency because of the outliers in the histogram of MSE.



Decomposing `fanPts` MSE into model-by-model MSE revealed major sources of prediction error. Specifically, yards were difficult to predict across the board seeing as that was the highest variance of all the variables of interest. The `PassYds` model performed by far the worst and the error contained therein could explain the slight overvaluation of quarterbacks distinguishable in the ranking results.

Stat	MSE
PassYds	1562.32
PassTD	0.17
PassInt	0.11
RushYds	266.61
RushTD	0.13
Rec	3.93
RecYds	650.02
RecTD	0.24
FL	0.05
fanPts	52.04

Discussion

The rankings produced ultimately have enough basis in reality to be practically useful as a fantasy draft tool, but should they be? Obviously the project presented here can not perfectly predict the fantasy season. While this project had solid models that accounted for a lot of details of the game of fantasy football, there are still some parts that should be perfected in the future given more time and resources. Some of the problems with the solution were that this project does not account for matchup strength. There is only one performance estimate per player and it would be helpful to see how difficult a players upcoming season will be along with their stats. Another problem with the solution was that this solution does not account for players with less than 17 games. Not including players with less than 17 games can be detrimental to the models because it does not allow for all potential performers to be accounted for. There could be some

players that perform at a high level i.e. starting rookies or players that are returning after an injury and they are not presented as assets in the solution Another problem with the solution is that with the data used, it is difficult to account for expectation of injury properly. An important part of fantasy is having players that are reliable and healthy. When players get injured often or are playing though an injury that has potential to deteriorate; it is important to account for that and be ready to replace them with a backup if necessary. Another problem with this solution is that a continuous distribution for truly discrete variables is assumed. An example of this is how `Rec`, `TD`, `FL`, and other variables are assumed to be continuous when they are not. This can of course lead to some inaccuracies.

This solution is not without it's strengths and there are certainly some advantages to it's implementation. One advantage of the solution is that predictions can be made in a rolling fashion. This means that the training data can be updated and the last 17 games for all players can be accounted for repeatedly. Another advantage of this solution is that it uses a wide variety of predictor variables. When creating models to solve a problem it is crucial to have a myriad of predictors that allow for a variety of variables to be accounted for. By including so many predictors, the project was able to reach the accuracy it did. A big strength of the project that gives this group pride is the interpretability of the project. When there the project was started the data was in a format that was not user friendly and was sometimes difficult to understand. This project took the abstract data and put it in a format that tells a story. It is accessible and easy to explain. The visualizations and tables are a fantastic aid in getting across the points of the project. Overall this project was fun and fulfilling to work on and is one that would even be fun to continue working on as the group's skillsets continue to develop.

References

Entertainment and Sports Programming Network. 2021. "Fantasy Football Rankings for the 2021 Season." 2021. https://www.espn.com/fantasy/football/story/_/id/32056077/fantasy-football-rankings-2021-season.

Sports Reference. 2021. "Pro Football Reference." 2021. https://www.pro-football-reference.com.