1 Introduction

Group 27 Members: Aidan Dowd (ajd7ma), Brandon Escobar-Campos (be9wd), Gray Goss (gwg3hs), and Chris Seo (rae5rg)

Question 1: What are the main factors that predict net offensive yards?

A football team is largely dependent on the amount of yards the offense secures. Having a greater amount of yards is beneficial for a team because it allows for a higher likelihood of scoring touchdowns and setting back the opposing team. Looking into what factors influence net offensive yards can be very applicable to NFL coaching staff, as well as lower level football teams.

We are interested in interpreting what variables can result in a significant prediction of net offensive yards.

Description of variables used in Question 1:

Variable	Description	Type
Net Offensive Yards (*net_off_yds)*	The sum of total pass and rush yards a team has accumulated throughout the regular season	Quantitative (Response variable)
Number of drives (*num_drive*)	The total number of drives a team has accumulated throughout the regular season	Quantitative
Total time of possession (*tot_pos_time*)	The total amount of time (in minutes) a team has had offensive possession throughout the regular season	Quantitative
Conference (*conf*)	The designated conference of an NFL team	Categorical - levels: NFC, AFC
Number of turnovers (*turnover_ct*)	The total number of turnovers a team's defense has accumulated throughout the regular season	Quantitative
Net penalty yards (*net_pen_yds*)	The total number of yards gained or lost during offensive possession throughout the regular season	Quantitative
Total Return Yards (*tot_return_yds*)	The total number of yards gained from punt and kickoff returns throughout the regular season	Quantitative

Question 2: What are the main factors that predict whether a team makes a pass or run play on 4th down?

4^{th} down plays can be the difference between winning and losing games. The decision to go for it on 4^{th} down is a big one, because if you don't convert it, the other team will take over the ball and you will have failed to score any points. Analytics has become a big part of this decision in the modern game, and we want to look at what makes a team decide whether to pass it or run it on 4^{th} down (binary response variable).

Description of variables used in Question 2:

Variable	Description	Type
Play Type (*play_type*)	Whether a 4th down play was a run or pass	Categorical - levels: pass, run
Seconds Remaining in the Game (*game_seconds_remaining*)	Number of seconds remaining the game	Quantitative
Yards to go until 1st Down (*ydstogo*)	Number of yards to convert 4th down	Quantitative
Score Differential (*score_differential*)	Score difference between the offense and defense at the time of 4th down	Quantitative
Yards to go to the Endzone (*yardline_100*)	Field position of the offense starting from their endzone, in yards	Quantitative
Proportion of Yards from Passing Plays (*pass_prop*)	Proportion of net offensive yards that come from passing plays noted at the time of the play	Quantitative

The original dataset is from *nflreadr*, using data from the 2017-2018 NFL season through the 2020-2021 NFL season. *nflreadr* is a minimal R package that downloads data repositories from *nflverse*, a set of packages exclusively about the NFL. It has its own website and can be found on GitHub. The dataset provides variables of each play that occurred throughout the entirety of NFL seasons. We aggregated the data to get variables which measured season totals to answer our first question. We chose to stop at the 2020-21 season because after this season, the total number of games increased from 16 to 17. We chose the 4 most recent seasons in particular because we wanted a large sample of data to work with while keeping the data relevant to the present.

2 Exploratory Data Analysis for Regression Question

In this section, we will analyze graphical summaries for the first question of the report.

The original dataset consisted of every play that occurred throughout each NFL season. As a result, filtering was required to only show regular season plays and aggregate observations based on the

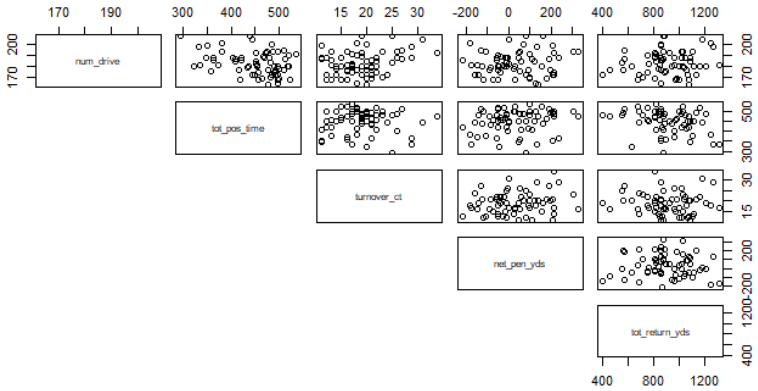
NFL team that had possession of the ball at that particular play. The only exception to this, is *turnover_ct* where the original data was filtered to only show plays where the results ended with a turnover and then aggregated based on the team that was defending that play.

Correlation Matrix of Explanatory Variables

	num_drive	tot_pos_time	turnover_ct	net_pen_yds	tot_return_yds
num_drive	1.00000000	-0.33542534	0.29494501	0.08260258	0.15832222
tot_pos_time	-0.33542534	1.00000000	0.02123313	0.17219276	-0.28082608
turnover_ct	0.29494501	0.02123313	1.00000000	0.18237164	-0.09152293
net_pen_yds	0.08260258	0.17219276	0.18237164	1.00000000	-0.02252895
tot_return_yds	0.15832222	-0.28082608	-0.09152293	-0.02252895	1.00000000

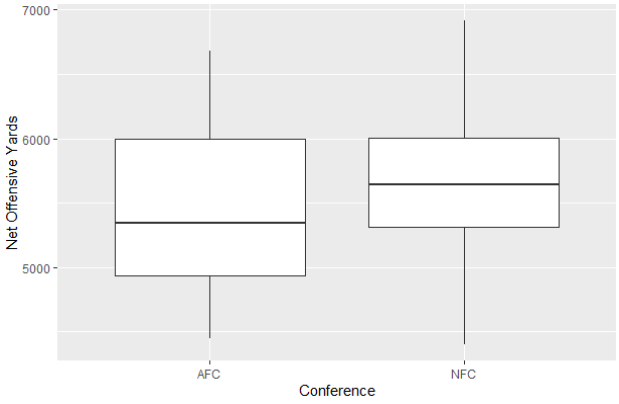
We created a correlation matrix between the aggregated explanatory variables to see if any of them were highly correlated to each other. Based on the correlation matrix above, there isn't any correlation that sticks out as having problems with multicollinearity. The highest correlation (in magnitude) shown is between *tot_pos_time* and *num_drive* (-0.3354). This relationship makes sense because typically the more time a team has the ball, the more drives that team will have. However, the relationship is not high enough to eliminate one of the two variables. None of the explanatory variables are especially correlated or uncorrelated to the other variables comparatively.

Scatterplot of Quantitative Variables



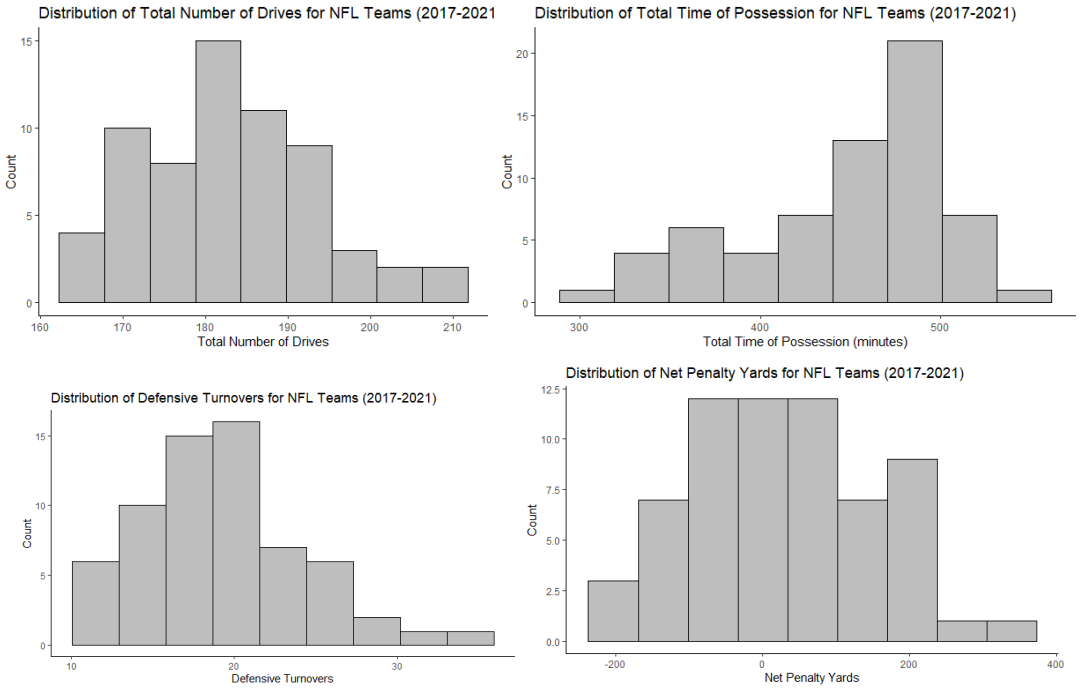
We created this scatterplot matrix to observe visually what was shown in the correlation matrix previously generated. As noted above, none of the explanatory variables for question 1 include any noticeable linear relationship. Some of the scatterplots appear to have clusters of data, while some seem to be more uniformly spread throughout the 2-dimensional space. The conclusions from this scatterplot matrix support that of the correlation matrix: There is no significant multicollinearity between the explanatory variables.

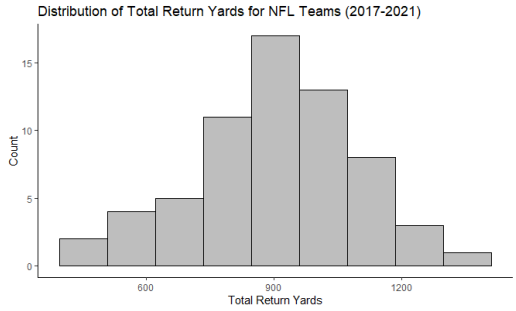
Boxplot of Net Offensive Yards by Conference



We included the boxplot above to explore the spreads of net offensive yards by conference. In the NFL, teams in one conference play more games against teams of their own conference than teams in the opposite conference. For this reason, it is worth exploring if one conference has stronger offenses (or weaker defenses) than the other. As it turns out, The NFC gets noticeably more net offensive yards than the AFC. The 25th percentile, median, and max for the NFC are all larger than the AFC. There are also no outliers, which in context makes sense as totaling over a full season (16 games) "smooths out" outlier offensive performances in individual games..

Distributions of Explanatory Variables

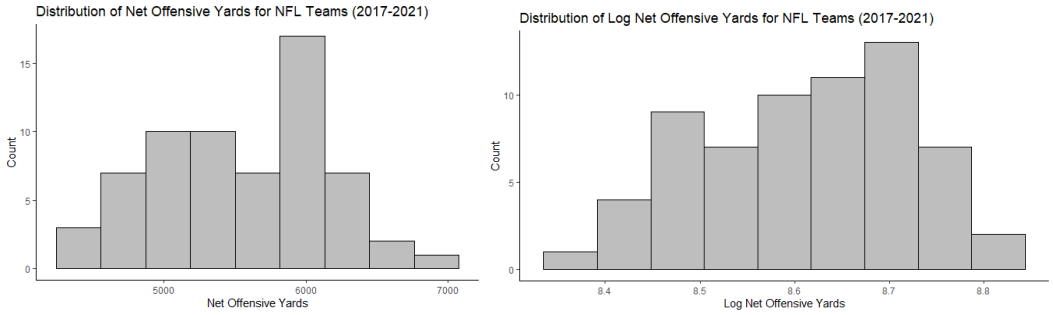




We looked into the histograms of the explanatory variables for each team to view the spreads, centers, and skewnesses of each variable to determine whether we should transform any of the variables, or remove any individual observations. One interesting difference to note is the differences in skewness between the number of drives and total time of possession. At first, one may reasonably suspect the shapes to be similar, as it makes sense that the more drives a team has, the more time they have the ball. However, the number of drives in a game actually increase when the possession time is shorter, and teams typically have roughly the same number of drives at the end of the game. So in reality, the differences in skewness make sense.

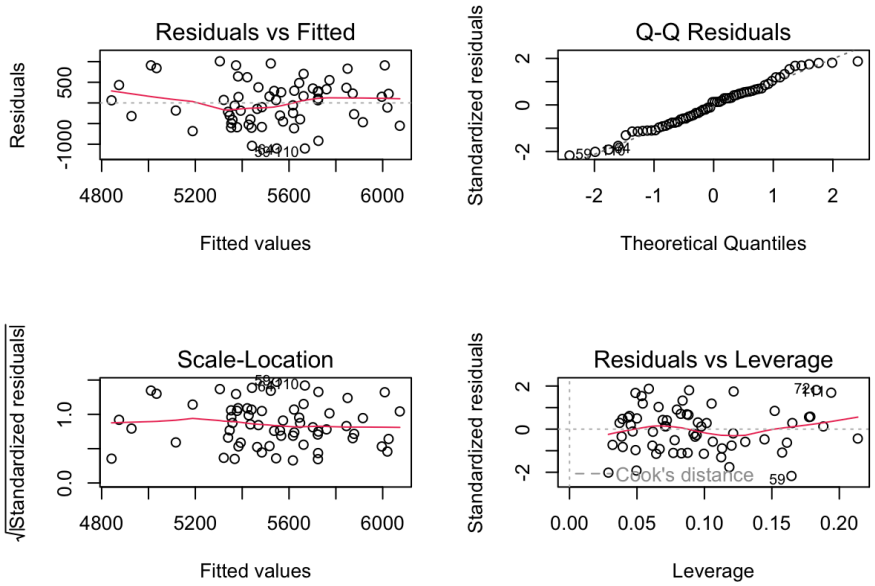
Overall, there does not appear to be any outliers with any of the observations' explanatory variable measurements. This is consistent with the scatterplot matrix above. All of the histograms above follow roughly normal distributions. For analysis, there will be no changes in the explanatory variables for ease of interpretation, unless we observe that the skewness of some of the variables results in poor predictive performance of our models (actual test MSE).

Distribution of the Response Variable



The first histogram shows the distribution of the non-transformed response variable, and the second shows the distribution of the natural log of the response variable. Although neither is perfectly normal, the transformed variable does not have a more normal-appearing distribution, so we elected to keep working with our non-transformed response variable (which can be interpreted as being roughly normal) to make our results more easily interpreted.

Checking for Regression Assumptions



All of the above 4 plots show that regression assumptions are met for the non-transformed version of the model for question 1. The Q-Q plot is roughly a straight line which proves normality of residuals. The two graphs on the left (Residuals vs Fitted and Scale-Location) prove that there is a consistent spread of residuals across all levels of the explanatory variables, indicating homoscedasticity. Finally, the Residuals vs Leverage plot shows that all observations are within Cook's distance, meaning there are no significant outliers that can influence our models.

3 Shrinkage Methods

Data Cleaning and Processing

In regard to data cleaning, the dataset had to be filtered by solely demonstrating plays in the regular season, to set an equal number of games (16) for each NFL team. Second, rows of data had to be grouped by the team in possession to represent the NFL team. Then, functions from dplyr and tidyverse were applied to find the values for each variable selected. For example, *net_off_yds* (response variable) was found by filtering the original dataset by regular season plays and by run or pass plays. Afterwards, the data was grouped by the team in possession to find the sum of yards gained for each pass or run by the team in possession throughout the season. Similarly, *num_drive* was calculated by finding the count of each drive an NFL team had possession of the ball. The only variable that required significant processing was *tot_pos_time*. In the original data set, time of possession was a character value. To use it for our model, it required changing it to a numeric value. Once it was changed to a numeric value, the procedure was similar in aggregating time of

possession values to each NFL team throughout the regular season. Due to the data spreading out to four seasons, the procedure was replicated with only a change of season.

Justification of Selected and Excluded Predictors

- *num_drive*: An offense can only accumulate offensive yards by being on the field. So, the total amount of times it's on the field is best represented by the total number of drives it has earned throughout the regular season.
- *tot_pos_time*: The longer an offense is on the field, the more opportunities it should have in earning net offensive yards. Total time of possession is the best variable to suit this description.
- *conf:* Teams are influenced by their own history and set of philosophies. Historically, NFC teams were known for its emphasis on the run game, while AFC teams emphasized the passing game. Generally, an offense that focuses more on passing than running will accumulate more offensive yards than a primarily offensive rushing team. So, seeing if conference relates to net offensive yards will help us understand the influence historical legacy can have on a team's offense success.
- *turnover_ct:* A team with a strong defense that earns a number of turnovers allows an offense to be on the field more repeatedly, helping the offense gain net offensive yards.
- *net_pen_yds*: Penalties can end drives or continue them. A penalty that costs a team a down or yards can result in the offense being restricted in the types of play they can call to get a first down, resulting in less offensive yards than without the penalty. Also, a penalty that advances the offense down the field allows the team a new set of downs and greater options to choose and gain net offensive yards.
- *tot_return_yds*: A high number of yards a team's special team earns can give great field advantage to an offense, allowing for a greater variety of plays to consider and benefit off of in yards accumulated. A low number of return yards can force an offense to be more stringent on what plays it conducts, due to its field position, and choose plays that might not be advantageous in earning a high number of offensive yards.

It's undeniable that the effectiveness of an offense, in accumulating yards, is determined by the competence and skill of the starting quarterback. So, it would make sense to include some factor that helps determine the value of the quarterback (number of years played, whether they were nominated for the pro bowl in the previous season, or average passer rating). However, it was not included in the model because the original dataset didn't provide much information on the history of each quarterback that played for a team. Additionally, injuries are a natural part of the game, so providing a variable for a starting quarterback that may have gotten injured half way through the season wouldn't be so helpful in determining the number of net offensive yards a team produces. Another variable that isn't included is the proportion of games played indoors. The reason is because the original dataset had different values for the type of stadium a game was played in for

each NFL season. Additionally, we don't think it would have much influence on our response variable, even though playing in an indoor stadium is generally easier for an offense because they're not playing against the weather.

Value of Threshold for glmnet() function

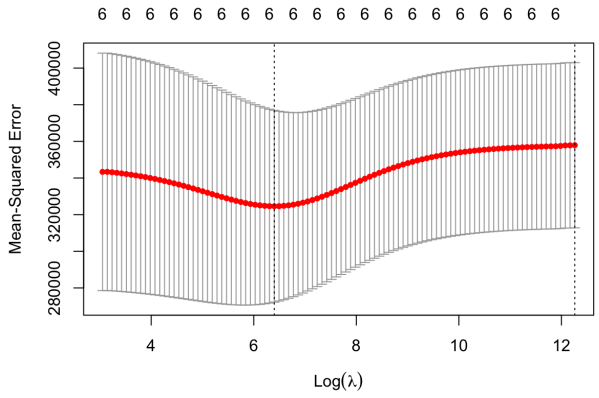
When the tuning parameter (λ) is equal to 0, the estimated coefficients between ridge and lasso regressions, and the estimated coefficients of ordinary least squares (OLS) are equal to each other. Using a threshold value of 10^{-22}, the coefficients from OLS and ridge are shown below:

		s0
(Intercept)	7056.36300555	7056.36300555
num_drive	-12.83496021	-12.83496021
tot_pos_time	1.63358307	1.63358307
confNFC	123.87458454	123.87458454
turnover_ct	18.14985420	18.14985420
net_pen_yds	0.03708206	0.03708206
tot_return_yds	-0.26661516	-0.26661516

Comparing the left column (OLS coefficients), with the right (ridge and lasso coefficients), we see that each row has an identical coefficient value. As a result, the threshold value of 10^{-22} is appropriate to use for the remainder of this section.

Ridge Regression

When conducting a 10-fold cross-validation on the training data, the value of the tuning parameter is 601.8521. The tuning parameter is generally high, which means the coefficients are being penalized heavily towards zero.



The tuning parameter that was calculated from a 10-fold cross-validation on the training data is shown, in the plot above, through the vertical dashed lines closest to the y-axis, on a log scale.

The number of predictors that are left in this model is 6. This is the same number of predictors that we started out with. This was expected because one of ridge regression's properties is that it doesn't remove any predictors from the model, regardless of how insignificant they are and how close they are to 0. The number of predictors that are still left in the model is shown by the number on top of the vertical dashed line closest to the y-axis.

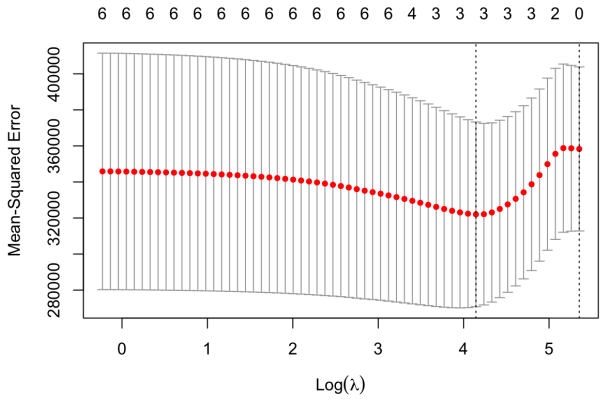
List of predictors that remain in the model after ridge regression:

- *num_drive*
- *tot_pos_time*
- *confNFC* (transformed from *conf* since glmnet() required categorical variables to be dummy variables - NFC = 1 and AFC = 0)
- *turnover_ct*
- *net_pen_yds*
- *tot_return_yds*

Using a tuning parameter of 601.8521, the actual test MSE is 340,206.6.

Lasso Regression

When conducting a 10-fold cross-validation on the training data, the value of the tuning parameter is 63.05097. It's not surprising that the tuning parameter for lasso is different from ridge regression because each is calculated differently and, therefore, have different ranges. For lasso, it's calculated by the sum of the absolute values of the coefficients, while ridge is calculated by the sum of squared coefficients.



The tuning parameter calculated using 10-fold cross-validation on the training data is represented on the plot above by the vertical dashed line on the left near 4 on the x-axis, on a log scale. Based

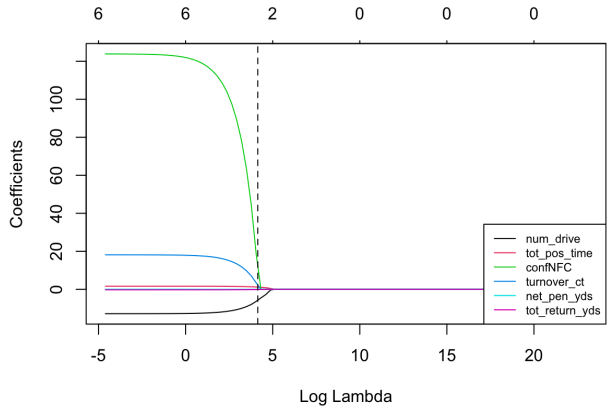
on the plot, and the vertical dashed line that represents the tuning parameter, there are 3 predictors left in the model.

```
(Intercept)    6494.384030
num_drive        -5.498560
tot_pos_time      1.322205
confNFC           .
turnover_ct       .
net_pen_yds       .
tot_return_yds   -0.611072
```

Based on the summary above, the three predictors that haven't been removed are:

- *num_drive*
- *tot_pos_time*
- *tot_return_yds*

The ridge plot below shows a visual representation of the effect the shrinkage penalty has had on the coefficients of the model:



confNFC shows to have the greatest shrinkage as it starts at a coefficient of 123.875 to eventually being removed from the model. The vertical dashed line in the ridge plot represents the tuning parameter calculated (63.5097) in log scale.

Using a tuning parameter of 63.05097, the test MSE is 346,943.4.

The actual test MSE for ordinary least squares (OLS) is 381,141.1.

Conclusion on Shrinkage Methods

Model:	Actual Test MSE:	Test Mean Error
Ordinary Least Squares	381,141.1	617.8

Regression		
Ridge Regression	340,206.6	583.5
Lasso Regression	346,943.4	588.8

The table above shows a brief summary of the actual test MSEs calculated based on the model. Overall, the models created didn't show to be effective in answering the question of interest (What are the main factors that predict net offensive yards?). This is evident by the high test MSE for all three methods. Our models showed to be ineffective in accurately predicting net offensive yards through unseen data.

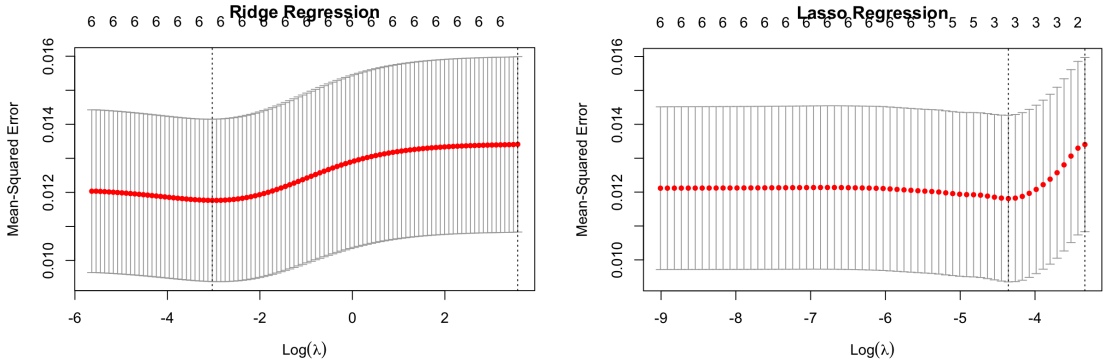
To understand why the test MSEs are high for the models presented, it's important to look back at the original data set and see possible influences. One possible explanation is that the response variable, *net_off_yds*, has an extremely high variance of 350,092 with a mean of 5,589.391. Looking at the Residuals vs. Leverage plot in the **EDA section**, there aren't any obvious outliers that pass Cook's distance, but even removing the observations (59, 72, and 111) that come close, the variance doesn't change much (346,434). One way to potentially reduce the variance is to consider a transformation to the response variable, given its right-tailed skewness (which is a second possible explanation for high test MSE for the models).

The reason a transformation wasn't applied to *net_off_yds* originally is because the original response variable passed the regression assumptions and the histogram of the log-transformed variable was relatively the same as the original variable. As a result, it was believed that keeping the original response variable would be reasonable, with the additional idea of wanting to maintain easy interpretation of the data and results. However, it now makes sense to look at how the OLS and shrinkage models will fare with a transformed response variable. Moreover, given transformation of the response variable, it also makes sense to log-transform the explanatory variables that showed skewness to compare the test MSEs for the original variables with the transformed variables.

The explanatory variables that will be log-transformed are: *tot_pos_time*, *num_drive*, and *turnover_ct*. The three variables were chosen because they show a skewness from the **EDA section**. Log-transformation will be applied because we're still concerned about interpretation and log-transformation is easier to interpret than other transformations, like squared or cubed root. The observations (59, 72, and 111) that were close to Cook's distance will also be removed for finding the updated test MSEs for the models.

Updated Ridge and Lasso Regression After Modifications

Threshold of 10^{-22} will still be used since the coefficients between ordinary least squares and ridge regression when the tuning parameter is 0 are the same.



The plot above and to the left is a graph showing the changes in MSE through changes in the tuning parameter, in log scale, in ridge regression. The plot above and to the right is the same thing except for lasso regression. For ridge and lasso, the tuning parameter associated with the lowest MSE is .0483 and .0128, respectively.

```
(Intercept)       1.029689e+01
confNFC           .
net_pen_yds       .
tot_return_yds   -7.692792e-05
log_tot_pos_time  3.904499e-02
log_num_drive    -3.526698e-01
log_turnover_ct   .
```

The summary above shows the coefficients that remain after lasso regression, which are the same coefficients as with the original lasso regression (*tot_return_yds*, *tot_pos_time*, and *num_drive*).

Table of Test MSE after Transformations of Skewed Variables and Removal of Potential Outliers

Model:	Actual Test MSE:
Ordinary Least Squares Regression	.009736549
Ridge Regression ($\lambda = .04828374$)	.009305033
Lasso Regression (λ =.01282458)	.00981849

Based on the table above, there has been a dramatic decrease in the test MSE. It's possible that the drastic decrease is the result of the updated models being incredibly more accurate than the

original. However, it could also be because the values we're dealing with are smaller in magnitude, influencing the magnitude of the actual test MSE to be smaller as well. This is evident by comparing the coefficient values for the original and modified lasso model. In regard to the performances between the three models, OLS actually performed better than lasso, which was not observed in the original model. Additionally, ridge performed the best out of all three, similar when dealing with the original data. The reason ridge had a better performance than lasso is because it was better to hold on to all variables, providing some influence to each of them, then removing some and providing greater influence to the remaining ones towards the response variable. This means that the variables in the model weren't very good to begin with when it came to predicting with unseen data.

Overall, it's a positive that the modifications showed a decrease in the actual test MSE for all three models, but there's an interesting observation to make the model even better for future analysis. Looking at the coefficients that were removed from lasso, for the original and modified data, all of them were not based on the direct involvement of the offense. There was a belief, within the group, that for an offense to succeed, the overall team had to succeed (defense and special team). However, with the coefficients in lasso, the variable that relates to the defense (*turnover_ct*) was removed and the variable for special teams (*tot_return_yds*) was still included, but with a significantly smaller magnitude than the other coefficients (original: -.611; modified: -7.692e-05). This indicates that the variables relating to other parts of the team weren't important in accurately predicting net offensive yards. The coefficients that did manage to stay in lasso were directly related to the offense (*tot_pos_time* and *num_drive*), with the exception of *return_yds*. In hindsight, this obviously makes sense, but it's interesting to see variables relating to other parts of the team not having much of an influence in the success of the offense (net offensive yards). This insight goes contrary to some of the most common ideas of how a good offense relies on its defense creating stops or special teams providing great field advantage.

The reason the performances of the models weren't great is because there weren't enough variables that included things explicitly about the offense. For future analysis, adding predictors like percentage of pass or run plays, number of injured offensive players, and average years of experience could improve our model's predictive abilities.

There weren't any significant challenges throughout creating the models. The only possible challenge was noticing an extremely high test MSE for the models with the original data and having to consider possible solutions like log-transformations and removing potential outliers.

4 Regression Tree

Data Cleaning and Processing

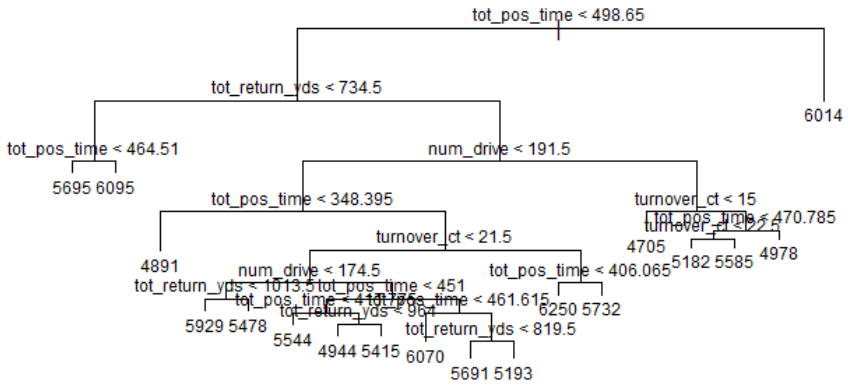
For the regression tree model, the final set of predictors that we decided to include were *net_off_yds, num_drive*, *tot_pos_time*, *conf*, *turnover_ct*, *net_pen_yds*, *tot_return_yds*. The reason for including these variables, and excluding others, are stated in the second part of the **Shrinkage Methods section**.

Summary() function of the Regression Tree Model

```
Regression tree:
tree::tree(formula = net_off_yds ~ ., data = tree_train)
Variables actually used in tree construction:
[1] "tot_pos_time"    "tot_return_yds" "num_drive"      "turnover_ct"
Number of terminal nodes:  18
Residual mean deviance:  178800 = 19670000 / 110
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-922.000 -227.400   -7.186    0.000  271.900  933.500
```

Based on the summary output shown above, the regression tree includes 18 terminal nodes and uses 4 variables (*tot_pos_time*, *tot_return_yds*, *num_drive*, and *turnover_ct*).

Graphical Regression Tree Model



When looking at the regression tree model, we can see that the tree does answer our question. Our question was if our variables are good predictors in net offensive yards. From the first split, teams with more than 498.65 minutes of possession time tend to have one of the higher amounts of yards in the model. Typically, when examining the plot, the top yards correlate to the *tot_pos_time* variable. This makes sense because teams with more time with the ball will have higher yards. Another statistic we found was that teams have higher offensive yards when *tot_return_yds* is less. This makes sense as the offense will have to compensate for yard gain the fewer yards returned on a kick off. We also noticed that teams with higher *num_drives* tend to have more yards as well. This variable also makes sense as teams have more drives they'll have more offensive yards. One

variable that may be misleading when looking at the plot is turnovers. Teams with less turnovers have fewer yards on average compared to the other variables. On both left and right sides of the tree, teams with less turnovers have less yards. This statistic doesn't go hand to hand as teams with less turnovers should have higher possession time. The regression tree model does its job of answering our question, and the only outlier or unreliable variable we had for predicting offensive net yards was turnovers.

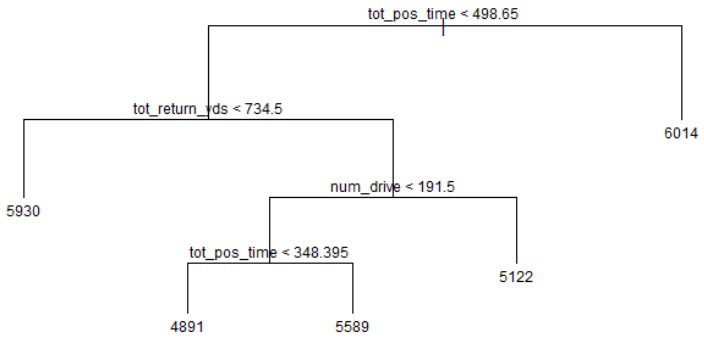
From the regression tree, the test MSE is 153,644.6.

Summary() function of Pruned Tree Model

```
Regression tree:
snip.tree(tree = tree_model, nodes = c(4L, 11L, 21L))
Variables actually used in tree construction:
[1] "tot_pos_time"    "tot_return_yds" "num_drive"
Number of terminal nodes:  5
Residual mean deviance:  255300 = 31400000 / 123
Distribution of residuals:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-1245.000  -295.400     5.866     0.000   306.500  1235.000
```

Based on the summary output shown above, the pruned tree includes 5 terminal nodes and uses 3 variables (*tot_pos_time*, *tot_return_yds*, and *num_drive*).

Pruned Tree Model



From the pruned tree model, the test MSE increased to 245,344.8.

Random Forest

```
Call:
 randomForest(formula = net_off_yds ~ ., data = tree_train, mtry = 4,      importance =
TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 350723.8
                    % var explained: -0.97
Random Forest Test MSE: 67625.45
               %IncMSE IncNodePurity
num_drive         5.80       7528920
tot_pos_time      7.38       9748757
conf              0.24       1325845
turnover_ct       1.88       7206475
net_pen_yds      -3.05       6114379
tot_return_yds    3.16       8804353
```

Predictors that were found important include *num_drive*, *tot_pos_time*, *conf*, *turnover_ct*, and r*tot_return_yds.* The test MSE is 67625.45

Conclusion

Model <chr>	Test_MSE <dbl>
Recursive Binary Split	153644.60
Pruned Tree	245344.80
Random Forest	67625.45

As seen from the summary and the plot of the regression tree, using binary recursive splitting gives us a regression tree with 18 terminal nodes, and 4 of our predictors were used in the process. The question of interest was what are the variables that predict net offensive yards. We decided to use variables that had stronger correlation with gaining yards, such as number of drives, possession time, conference, turnovers, net penalty yards, and total return yards. Looking at the pruned tree example, we can see that teams with a total possession time over 498.65 minutes per season resulted with the highest net offensive yards being roughly 6014. This makes sense as total possession time has correlation with a team's ability to gain net positive yards. The next split on the left is total return yards and teams with less than 734.5 yards that resulted with 5930 yards. This predictor makes sense as well because teams with less return yardage will have to make up for more yards to gain, therefore increasing the amount of net offensive yards. The next spit on the right is the total number of drives. When the number of drives is above 191.5, we see that teams will have 5122 yards. This correlation also makes sense because teams with more drives will have more opportunities to gain net positive yards. However, it's interesting that it was less yards than teams with less number of drives but more than 348.395 minutes of possession time being 5589. The final split is total possession time less than 348.395 minutes and it has 4891 yards being the least amount of yards in the total tree. This correlation makes sense as well meaning that less possession time will lead to the least amount of total net yards. Although these variables have correlation with net offensive yards, there was a problem when finding the MSE values for recursive binary, pruned tree, and random forest. Our MSE values were significantly high, suggesting that our variables are not great when finding which variables are the best predictors. Although these variables are related and have correlation in gaining offensive yards, in the overall game of football, there are many factors that have significance which can influence overall net

yards. From the random forest test, the only significant factors were *num_drive*, *tot_pos_time*, *conf*, *turnover_ct*, and *tot_return_yds*. They were the only variables with positive %IncMSE, but *conf* was also close to being insignificant with a value of .24. For the future, we may have to consider other variables that may have a lower MSE value, or either transform our current variables so that the MSE values can be lower. Some of the challenges we faced during regression trees were solutions to find variables with lower MSE values. We would have thought that variables with close relation to net yards would have had the lowest values, but that wasn't the case. Some outside variables can significantly impact yards gained, and finding a reliable variable was challenging in this case.

5 Classification Tree

Data Cleaning and Processing

Before fitting any type of classification tree, we had to do some data cleaning. We changed *tot_pass_yds* and *tot_rush_yds* into a proportion of the teams total yards gained up until that point to hopefully better predict whether a team would choose to run or pass. Because these would likely be correlated with each other, we dropped the *tot_rush_yds* proportion from the data, leaving us with 5 explanatory variables.

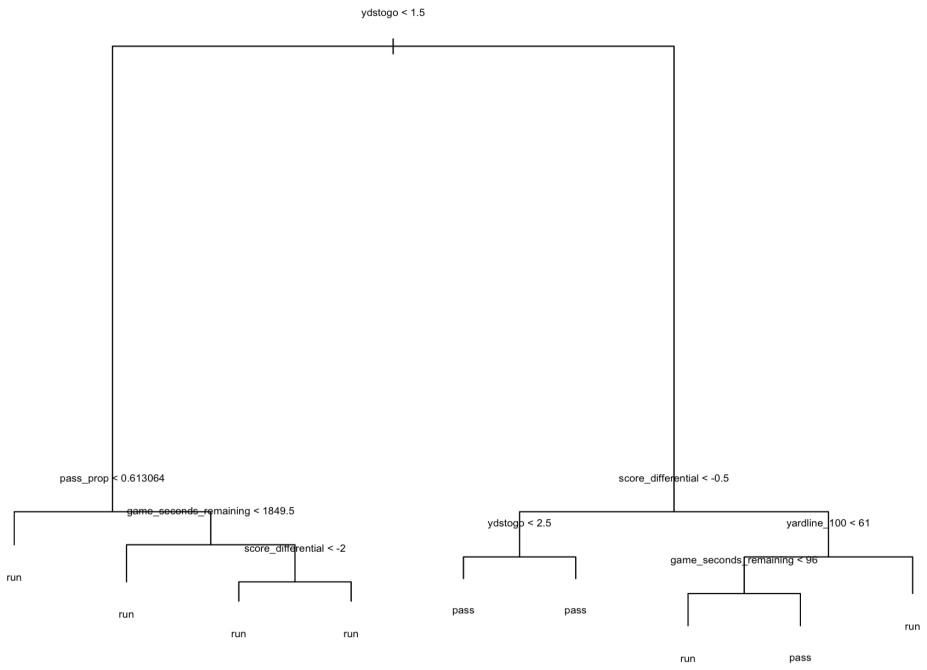
Variables used in the Model

- *play_type*
- *game_seconds_remaining*
- *ydstogo*
- *score_differential*
- *yardline_100*
- *pass_prop*

Recursive Binary Splitting

Summary

```
Classification tree:
tree::tree(formula = play_type ~ ., data = train)
Number of terminal nodes:  9
Residual mean deviance:  0.6807 = 275 / 404
Misclassification error rate: 0.1453 = 60 / 413
```



As you can see by the summary and the plot of the classification tree, using binary recursive splitting gives us a classification tree with 9 terminal nodes, and all five of our predictors were used in the process. This tree answers our question of interest because it helps us get down to predicting whether or not a team will choose to run or pass. The primary split uses the *ydstogo* variable. If the yards to go is less than 1.5, then the tree branches left, and if the yards to go is greater than 1.5, then the tree branches right. All of the splits on the left side result in a run regardless, whereas 3 of the 5 splits on the right result in a pass. This makes sense, as if there is only 1 yard to go to get a first down, most teams will elect to do the simple thing and run the ball as you're more likely to pick up a yard, whereas the decision becomes more complicated with more yards to gain in only one play. The next split on the right involves the *score_differential* variable, and it is split at -0.5, which is basically whether a team is winning or losing. At this split we can see that if a team is losing and has a yards to go greater than 1.5, then based on the decision tree the team will choose to pass, which is logical sense passing is better for trying to gain as many yards as possible while trying to catch up. The branches on the far right using *yardline_100* and *game_seconds_remaining* make a little less sense, so perhaps those will be pruned in the future. This could possibly be because there are very few fourth down plays that occur when *yardline_100* > 61, so this split could be thrown off a bit.

Below is a confusion matrix for the predictions made by our classification tree in comparison to our test response variable. The overall accuracy for the test is 78.45%.

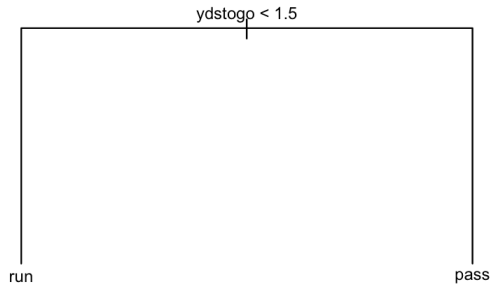
```
          tree.pred.test
y.test pass run
  pass  205  61
  run    28 119
```

As you can see by the confusion matrix, the false positive rate is 33.89% and the false negative rate is 12.02%. Having experimented with different thresholds, the .5 threshold is fine as lowering it makes it much less effective at predicting the choice to pass correctly, and raising the threshold decreases the accuracy of predicting both outcomes.

Pruning Tree

When electing to prune the tree, we are left with a best size of 2 terminal nodes, with the one split being whether or not the *ydstogo* variable is greater than or less than 1.5.

```
Classification tree:
snip.tree(tree = tree.class.train, nodes = 2:3)
Variables actually used in tree construction:
[1] "ydstogo"
Number of terminal nodes:  2
Residual mean deviance:  0.8783 = 361 / 411
Misclassification error rate: 0.1671 = 69 / 413
```



As previously explained, this makes sense as the easiest way to pick up 1 yard is to run it, whereas picking up 2 yards or more is a bit trickier to do by running it when only given one play to do so, so many teams will elect to throw it. The overall test accuracy for this test is 78.45%, which is exactly the same as the test accuracy from the tree created by binary recursive splitting.

```
          tree.pred.prune
y.test pass run
  pass  208  58
  run    31 116
```

The false positive rate given by the predictions from this pruned tree is 33.33%, which is just slightly lower than that of the binary recursive splitting tree test. However, the false negative rate for this test is 12.97% which is slightly higher. Based on this examination, it is unclear whether or not this tree is actually better, especially when you compare the results from the confusion matrix with the misclassification error rate, which is higher for the pruned tree as suggested by the summary outputs for each tree.

Random Forests

```
Call:
 randomForest(formula = play_type ~ ., data = train, mtry = 5,      importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 5

        OOB estimate of  error rate: 16.71%
```

In the random forest classification test, the *ydstogo* variable was once again the most important predictor, with the others well behind all at about the same level.

```
Confusion matrix:
     pass run class.error
pass  234  31   0.1169811
run    38 110   0.2567568
```

The overall accuracy for this test was 78.2%, which is just a little bit lower than the previous sections. The false positive rate is 21.98% which is much lower than the previous classification trees. However, the false negative rate is 13.97% which is a little higher. I don't think the threshold needs to be adjusted, as adjusting it had similar effects on the FPR and FNR in this test as in the previous test

Conclusion

	Binary Recursive Splitting	**Pruning**	**Random Forest**
Terminal Nodes	**9**	**2**	**5**
Variables Used	**All**	**Yards to Go Only**	**All**
Overall Accuracy	**78.45%**	**78.45%**	**78.2%**
FPR	**33.89%**	**33.33%**	**21.98%**
FNR	**12.02%**	**12.97%**	**13.97%**

The model does a decent job of helping us predict whether a team will run or pass on 4th down based on our predictors. As explained in the binary recursive splitting section, the data kind of lends itself to a decision tree, as all of the predictors go into the actual decision to go for it on 4th down. If there is a lot of time left in the game and a team is winning, they will be less likely to go for it logically, and if the yards to go is shorter, then it will be easier to get a first down, so it is more likely that a team will go for it. All of this makes sense logically, so it is not a surprise that the decision trees all do a decent job of making predictions. The most interesting thing that we found was how important *ydstogo* was. This does make sense, but I guess we figured that the other predictors would be similarly important, especially game_seconds_remaining. There were no

glaring challenges that we were presented with in this part. The only real challenge came when we changed the *tot_pass_yds* and *tot_rush_yds* to proportions, we had to decide if we wanted to drop one of the variables or not. We decided to drop the proportion of rush yards because we figured that there would be multicollinearity if we left both in.