# 226 Project

Peter Karnchanapimonkul (phatk), Tyler Coleman (SUNET: colemant)

November 2019

## 1 Investigation of Data

### 1.1 Data Set Description and Background

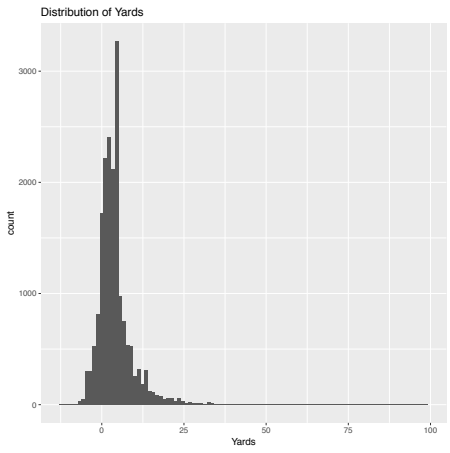
The data was collected through NFL Next-Gen-Stats, a data analytics platform that captured real time information about players during the 2017 and 2018 NFL football season, and aggregated on the Kaggle Wesbite for the NFL Big Data Bowl and is a current competition. Because of this, we did not have major concerns about the collection process and the completeness or accuracy of the data, although there are a few covariates and observations that are missing data. Specifically, the dataset contains 49 covariates to a play - some provide ID's and other characteristics of understanding the dataset. Examples include, the home and away team, season, quarter, rusher ID, field position offense and defense personnel, as well as many other covariates that are used to characterize each rushing play for each game.

The observations are for each rushing play covariates assicuated with each of the 22 players on the field (11 offense, 11 defense). We decided to only use the observations associated with the rushing player for each play, as each of the 22 covariate provided the same information except player ID and associated heights and weights, and we decided that this information is collinear with other information about the teams provided. Ultimately, in the training set with 80% of the data, we had 18,536 observations and 49 covariates of rushing player and overall team and setting characteristics.

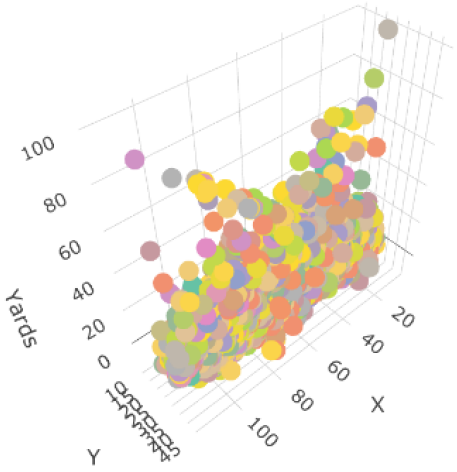
### 1.2 Continuous and Binary Response Variables

#### 1.2.1 Continuous Response Varaible

The continuous response variable that we are trying to predict is in line with the contest guidlines - "How many yards will an NFL player gain after receiving a handoff?" Whereas the competition calls for a probability density associated with values from {-99,99} yards, we decided to modify and predict the exact number of yards a player will get given covariates. Therefore, our predictions of $\hat{y}$ are yards for the play, and we will be comparing it to the true y yards on the play. This modification made it more straightforward to apply many of the techniques we learned in class. As shown below, many runs result in plays between 0 and 10 yards, with 9.5% resulting in negative yards and 11% resulting in more than 10 yards.




(a) Distribution Yards.




(b) X and Y Field Position vs. Yards: Colored by Team

### 1.2.2 Binary Response Variable

The binary response variable we are using for this prediction task is "Yes or no, if a run results in a new set of downs." This means, that if the yards gained on the play is $>=$ the distance required to get a first down then they convert the first down and the binary variable is a 1. If yards gained is $<$ distance to convert, then the response variable is a 0, as they did not get the first down. We thought this was a interesting take on the prediction of the continuous response variable of yards. In our data, there were 13,219 outcomes in which a first down was not reached and 3542 outcomes in which it was.

## 1.3 What questions can we answer with the data set

Beyond the questions asked in the previous section about first downs and yards gained, it is interesting to try and understand, what if any features of a play can give insight into the results of the play, and if these insights go with our intuitive understanding. Specifically, questions we look at through analyzing the data are: 1: If the results of a play given a set of characteristics is truly random (one draw to football is the idea that anything can happen) 2: Specific features of a play beforehand may give insight into the results and may provide a large advantage if the information is asymmetrical. 3: Idiosyncratic plays: examples of this are "is it true that a superstar player or characteristic make a true difference?", "are there some offensive formations that dominate a defensive formation?" as well as other similar questions.

## 1.4 Further Exploration of Data and Data Cleaning

### 1.4.1 Data Cleaning and Covariate Selection

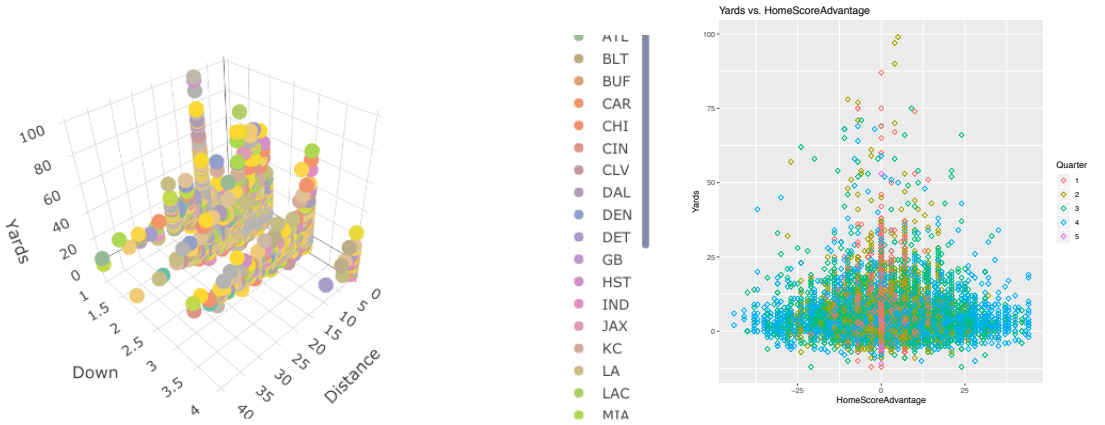
Beyond what is stated in section 1.1, it was necessary to clean the data further as there were several cells within GameWeather and Wind, that were either missing, or contained misspellings. Moreover,

there were certain 'special' offensive and defensive formations and personnel that only contained 1 or 2 observations, that would provide skewed estimates or caused errors in cross validation. In both of these cases we decided to omit the observation as we felt there were enough other observations to be represent an accurate representation of the population model and the missing data was random: however, this is definitely something to be considered. The final train set had 16,761 observation.

As for the covariates, we factored categorical variables such as Formation, Team Name, quarter ...etc and made sure to kept ordinality in specific cases such as defenders in the box. Other changes were required such as changing Age and GameClock to a usable format. Also, we dropped several of the columns as we believed they were collinear. An example of this include Stadium and Location: If we know the home team, they always play in the same stadium located in the same city. Besides these changes, we created HomeScoreAdvantage and TimeDifference between snap and handoff, from the data as we thought these covariates would give insights into plays.

### 1.4.2 Further Exploration

Two intuitive and interesting patterns we were able to discern were 1: Down, Distance, and Yards Gained - it appears fourth downs result in less yards. This supports an intuitive understanding as many teams only try to convert fourth downs when they are close to a first down. The defensive team knows this and so they stack the box to stop the run. Ultimately the resulting play has less yards, but the offensive team might only 'need' 2 yards. 2: When the HoemScoreAdvantage (Amount home team is winning or losing by) is large, then yards appear to be less. During blowouts teams may play more conservatively and take kneels for negative yards. Both graphs helped our understanding of the data and which covariates may be predictive of yards and are shown below. More exploration is shown in the appendix.




(a) Down and Distance vs Yards: Colored by team

(b) Difference In Scores vs. Yards Colored by quarter

# 2 Prediction

## 2.1 Regression Task

### 2.1.1 Baselines

For The baselines of the regression task, we decided that we would have two baselines to try to improve upon. The first is a regression against yards using only an intercept term, which always predicts the mean yards (4.2 yards). We also used the model with all covariates to serve as another baseline. Using 10-fold cross validation, the RMSE's are 6.4191 and 6.388, so only a slight improvement with all covariates.

### 2.1.2 Lasso and Ridge Regression

In an effort to further understand the covariates and potentially their ability to explain yards, we executed both Lasso and Ridge Regression. The RMSE's are 6.3349 and 6.3333 respectively. So, these both appeared to be an imporovement in RMSE within the training data. The graphs of the coefficeints against log lamda and also MSE against log Lambda are shown in the appendix.

### 2.1.3 Forward and Backward Stepwise

Lastly, we ran both forward and backward stepwise regression to try and find a balance between explainability and complexity. Forward stepwise, resulted in a very similar outcome to both the regression with intercept term, and Lasso and Ridge. Ultimately, there was not a reduction in AIC by adding any covariate, so the 'best' model was simply the intercept. However, the backward stepwise regression resulted in what we deemed an interpretable model with RMSE of 6.3614.

We decided that the **best model** to implement for the test will be the backwards stepwise regression. It resulted in a a lower RMSE than both of the baselines, and provided coefficients were understandable. The code to produce the model is shown in the appendix. However, even though within the train the RMSE is 6.3614, we expect it to approach the prediction with just the intercept in the test set of 6.4191.

## 2.2 Classification Task

### 2.2.1 Baseline

For the baseline of the classification task, we have a model that uses logistic regression with all covariates. Using 10-fold cross validation, the accuracy is 81.6 %.

## 2.3 Best Classifier

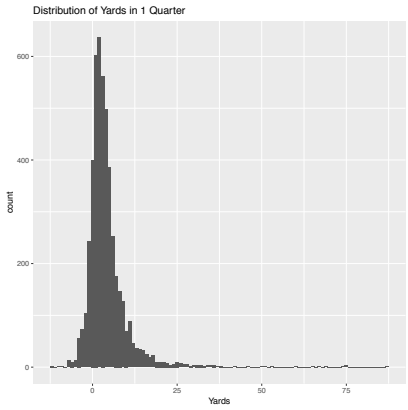
We explored two other classifiers: SVM and bagging. SVM or support vector machine works by choosing a hyperplane with the largest margin which categorizes new examples. Boosting trains bootstrap samples using decision trees in parallel and then average into a bagging classifier.

Out of all the classifiers, the classifier with the highest accuracy is SVM with a radial kernel, cost of 1, and gamma of 0.05 with accuracy of 81.7 %. Nonetheless, the increase in accuracy is less than one percent in the expense of run time and complexity. Hence, the best classifer is logistic regression with all covariates with accuracy of 81.4 %, and we expect it to perform similarly in the test error. Diagrams showing the accuracy for each model and confusion matrix for logistic regression are shown in appendix.

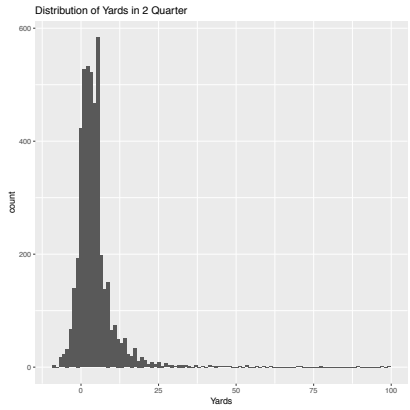
# 3 Appendix

## 3.1 Further Exploration

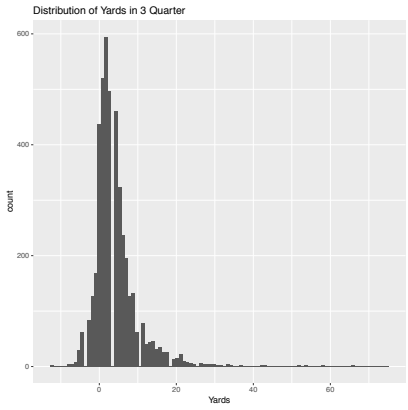
Besides the investigation of the data presented in the main piece of the project, we analyzed many different covariates to try to understand our data as well as possible. First we looked at the distribution of yards over each quarter to try to see if there was any correlation. The plots are shown below and appear to not have a pattern.



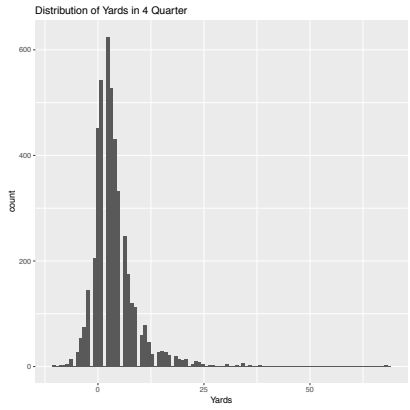

(a) Yards: 1st Quarter




(b) Yards: 2nd Quarter




(a) Yards: 3rd Quarter




(b) Yards: 4th Quarter

To answer the question of whether an offensive or defensive formation dominates another posed in section 1.3, we explored the data between offensive formation, defenders in the box and yards. Shown below it does not appear for any team, that one formation is 'better' than any other. We can see however, that some formations are run more than others.

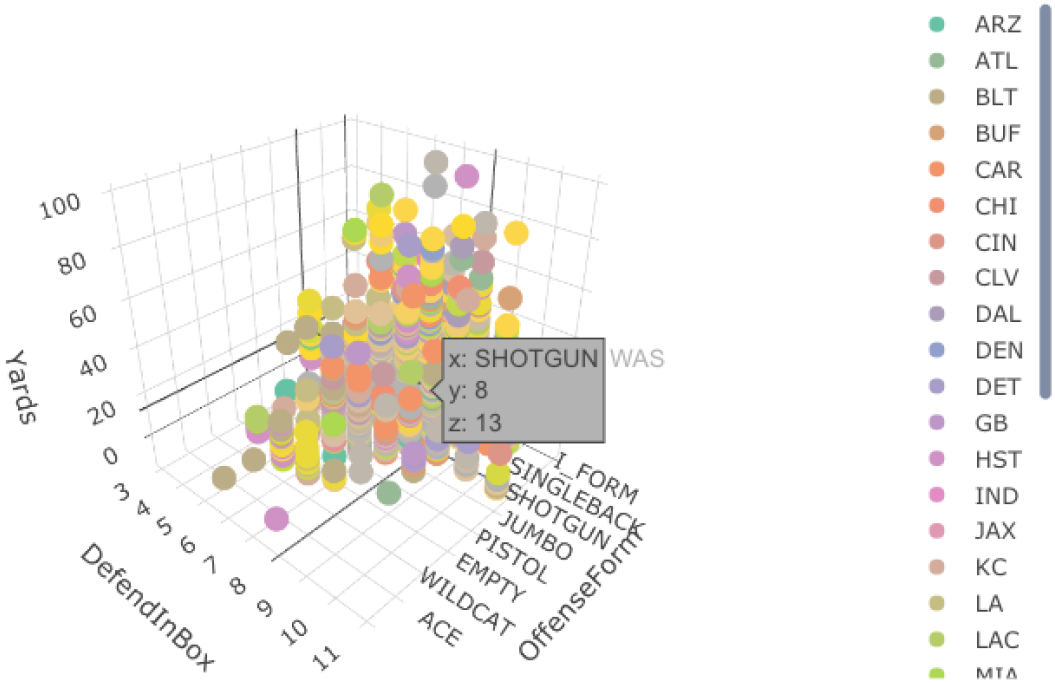



Figure 1: Offense vs. Defense vs. Yards

Lastly we looked at the variable of TimeDifference between snap and handoff we created relative to the number of yards gained. Again, although at first glance it seems that some time differences result in more yards, upon furhter inspection, it is the case that the proportion may be the same but the absolute number of plays that have a difference of 1-2 seconds is higher. Nevertheless, we thought this might be an important and unique covariate, and the graph is shown below.

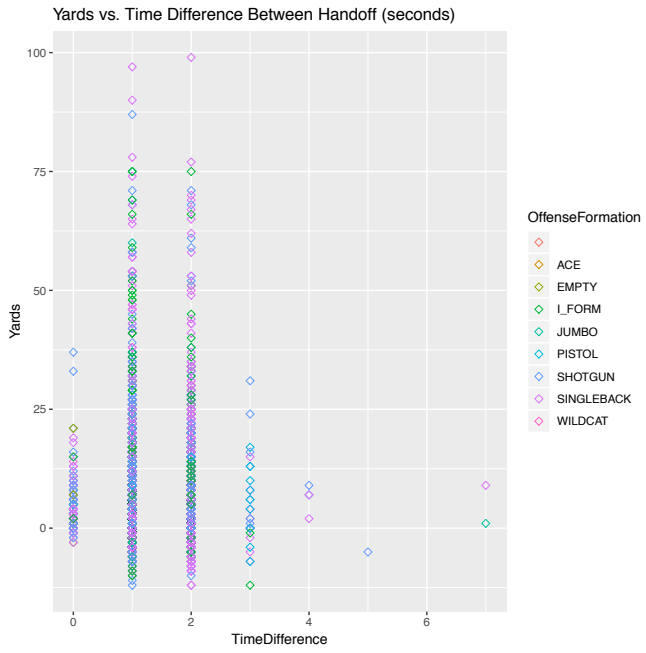



Figure 2: Time Difference vs. Yards

## 3.2 Prediction Continued

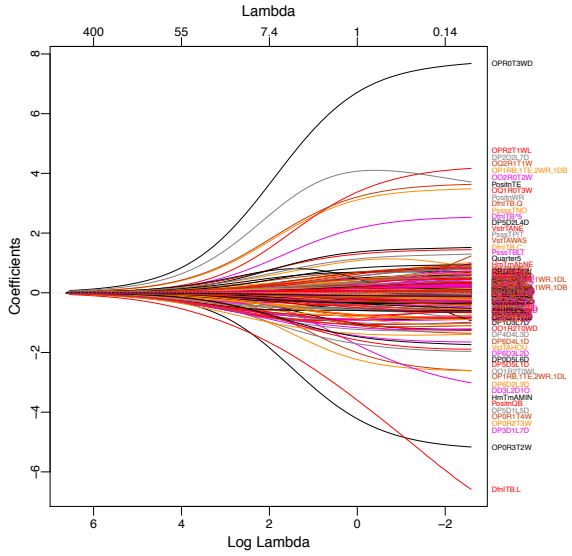
### 3.2.1 'Best' Model: Backward Stepwise Regression

```
Coefficients:
        (Intercept)          Season2018            Distance  DefendersInTheBox.L
           10.21528             0.29921             0.07749             -7.90479
DefendersInTheBox.Q  DefendersInTheBox.C  DefendersInTheBox^4  DefendersInTheBox^5
            1.73697            -1.43598             1.30967             -0.03050
DefendersInTheBox^6  DefendersInTheBox^7  DefendersInTheBox^8          PositionFB
            0.82678            -0.27303             0.33841             -5.53713
         PositionHB           PositionQB          PositionRB          PositionTE
           -5.10359            -6.39248            -5.09622             -3.26261
         PositionWR         Temperature            Humidity      TimeDifference
           -3.51122            -0.00532            -0.00470              0.14964
 HomeScoreAdvantage           PlayerAge
           -0.01002            -0.03798
```

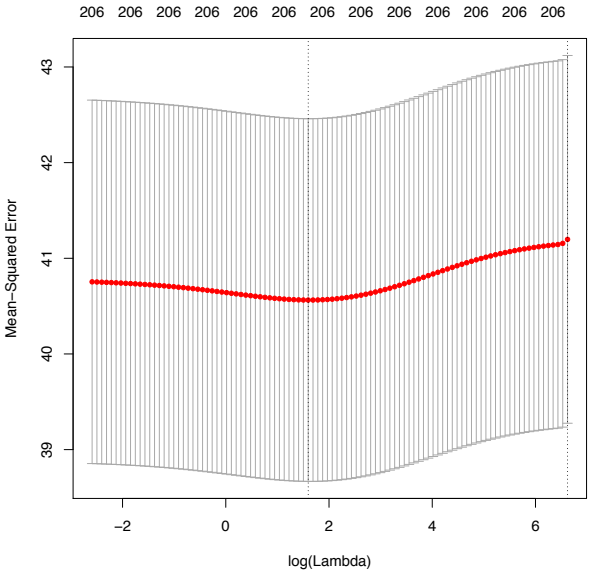
Figure 3: Coefficients of Backward Stepwise Regression

### 3.2.2 Lasso and Ridge Regression

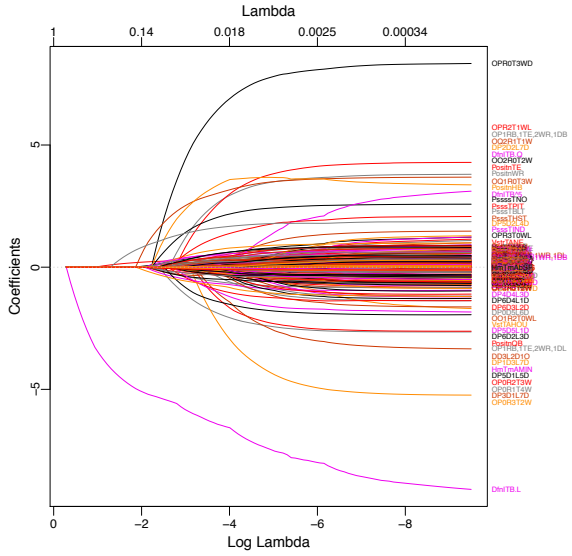
Although Lasso and Ridge seem to provide improvements in in sample Cross Validation, upon inspection of the covariates at the optimal values of lambda, it appears both models shrink the coefficients to basically 0, and predict small fluctuations around the mean. This time the intercept is 4.17 and there are small fluctuations per run. Nevertheless, the graphs shows that some covariates are nevertheless important, such as Offensive Personnel, Defenders in the Box, and Rushing Player. The graphs of shrinking the covariates and selection of optimal covariate coefficients and lamda are shown below.



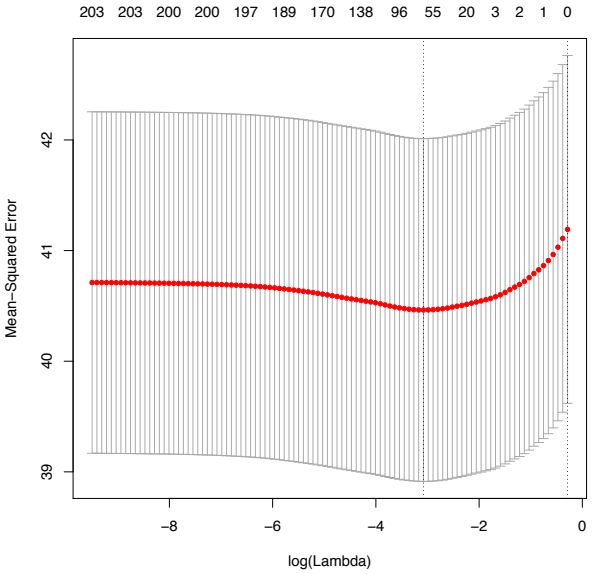

(a) Shrinkage Ridge




(b) Optimal LamdaRidge




(a) Shrinkage Lasso




(b) Optimal Lamda Lasso

#### 3.2.3 'Best' Model: Classification

```
Mean accuracy of 10-fold validation is:  0.81585
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1233   82
         1  226  134
```

Figure 4: Classification using Logistic Regression

| Classifiers | Accuracy |
|---|---|
| Logistic Regression | 81.58% |
| Bagging (Ada) | 81.25% |
| Support Vector Machine (radial) | 81.95% |

Figure 5: Accuracy of each Classifiers

# MS&E 226 Project

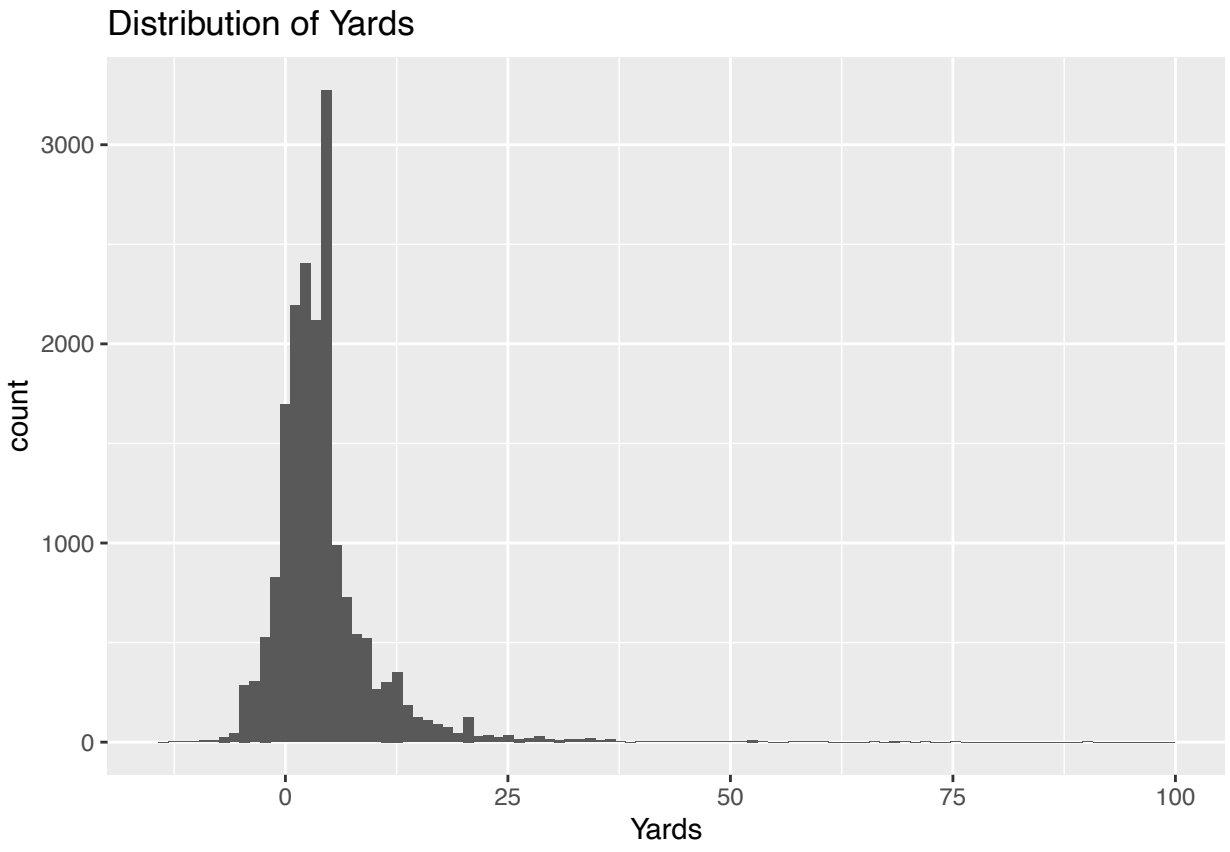
*Peter Karnchanapimonkul (SUNET: phatk), Tyler Coleman (SUNET: colemant)*

## IMPORT DATA AND SPLIT INTO TRAIN AND HOLDOUT

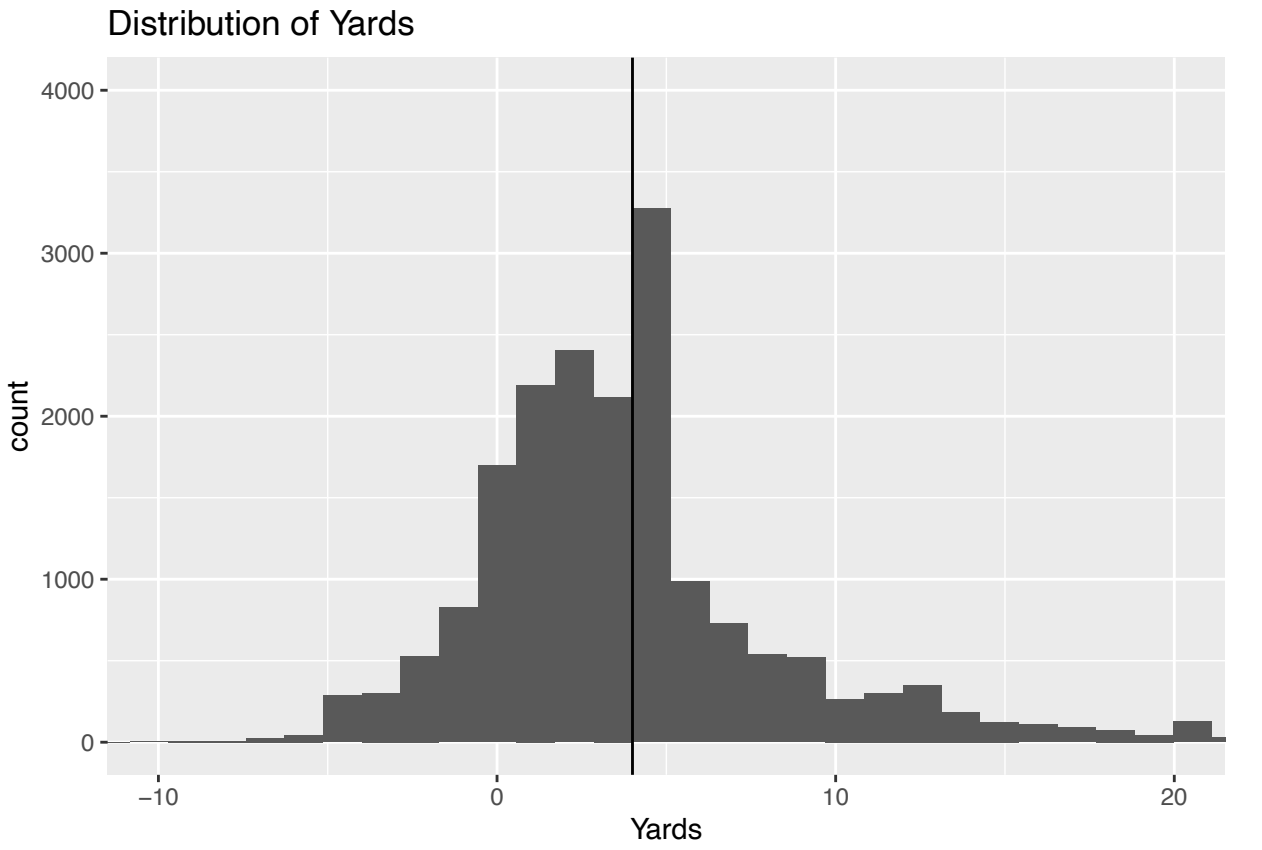
```
NFL_DATA <- read.csv(file = "train.csv", header = TRUE, sep=",")
NFL_DATA_Run_Observations <- NFL_DATA[(NFL_DATA$NflIdRusher == NFL_DATA$NflId)    , ]

# split 80:20 for training:test
set.seed(123)
training_data = sample(nrow(NFL_DATA_Run_Observations), size = nrow(NFL_DATA_Run_Observations) * 0.8)
NFL_DATA_Train = NFL_DATA_Run_Observations[training_data, ]
NFL_DATA_Holdout = NFL_DATA_Run_Observations[-training_data, ] # holdout is remaining indices
#View(NFL_DATA_Train)
```

## DATA EXPLORATION OF OUTCOME VARIABLE

```
# Explore distribution of the continuos response variable we are predicting. (Yards/carry)
ggplot(data = NFL_DATA_Train) +
  geom_histogram(mapping = aes(x = Yards), bins = 100) +
  ggtitle("Distribution of Yards")
```



```
# Zoom into specifically -10 to 20 yards
ggplot(data = NFL_DATA_Train) +
  geom_histogram(mapping = aes(x = Yards), bins = 100) +
  coord_cartesian(xlim=c(-10,20), ylim=c(0, 4000)) +
  geom_vline(xintercept = 4) +
  ggtitle("Distribution of Yards")
```




```
# 2, 3, 4, 5 yards in particular are the peaks

# What percentage of runs result in > 10 yards? 0.0956517
More_Than_10 = NFL_DATA_Train[(NFL_DATA_Train$Yards > 10), ]
Percentage_Over_10 =  nrow(More_Than_10) / nrow(NFL_DATA_Train)
Percentage_Over_10
```

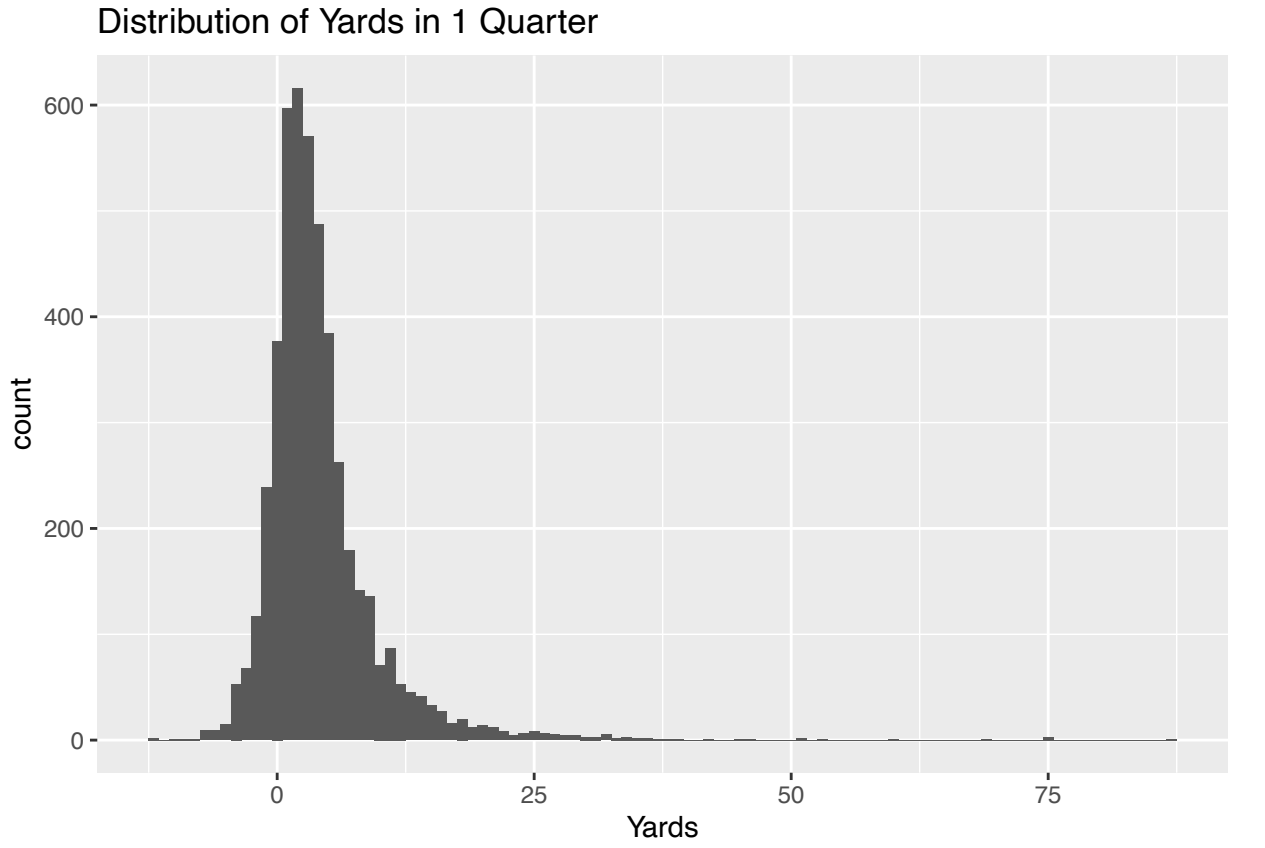
```
## [1] 0.095652
```

```
# What Percentage of runs is < 0 yards? 0.1101101
Less_Than_0 = NFL_DATA_Train[(NFL_DATA_Train$Yards < 0), ]
Percentage_under_0 =  nrow(Less_Than_0) / nrow(NFL_DATA_Train)
Percentage_under_0
```

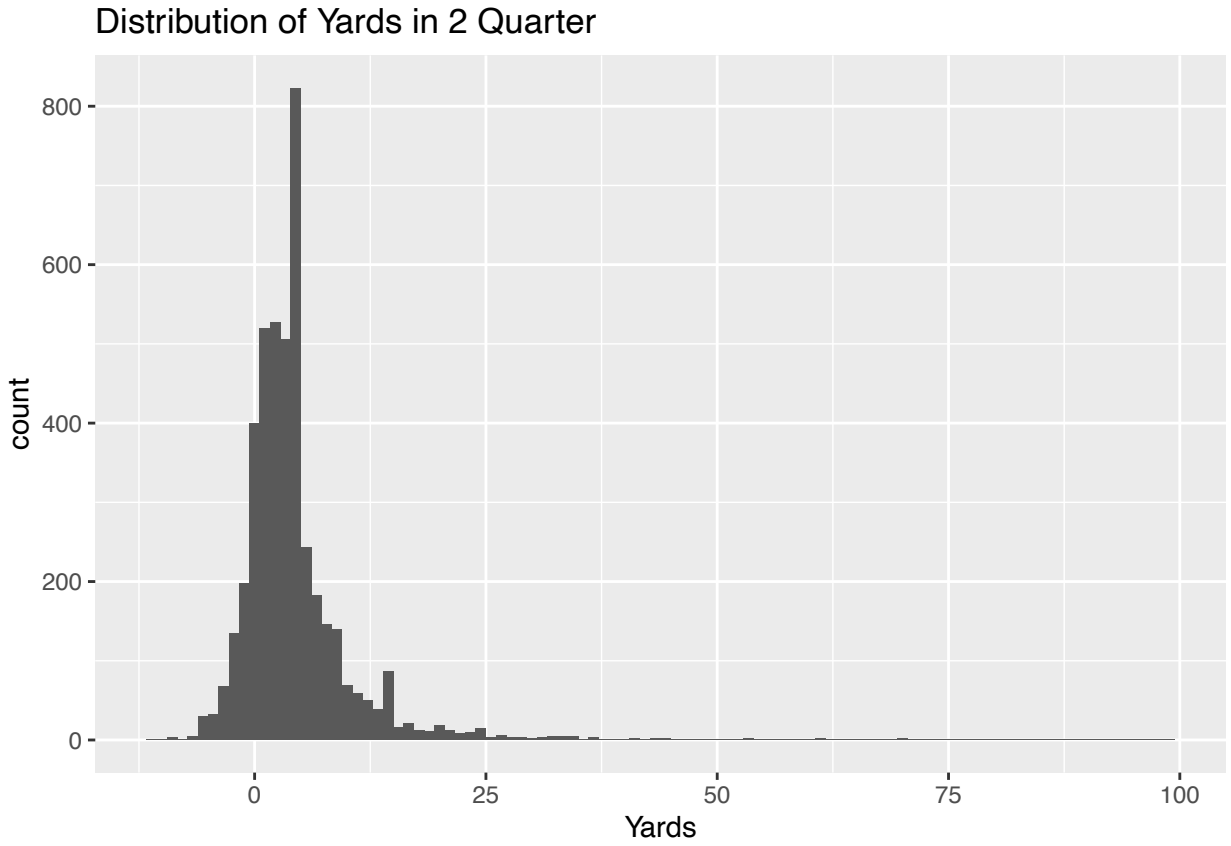
```
## [1] 0.11011
```

```
# See if there is difference in run yard by quarter

# FACTOR QUARTER
NFL_DATA_Train$Quarter = factor(NFL_DATA_Train$Quarter)

# First quarter
```

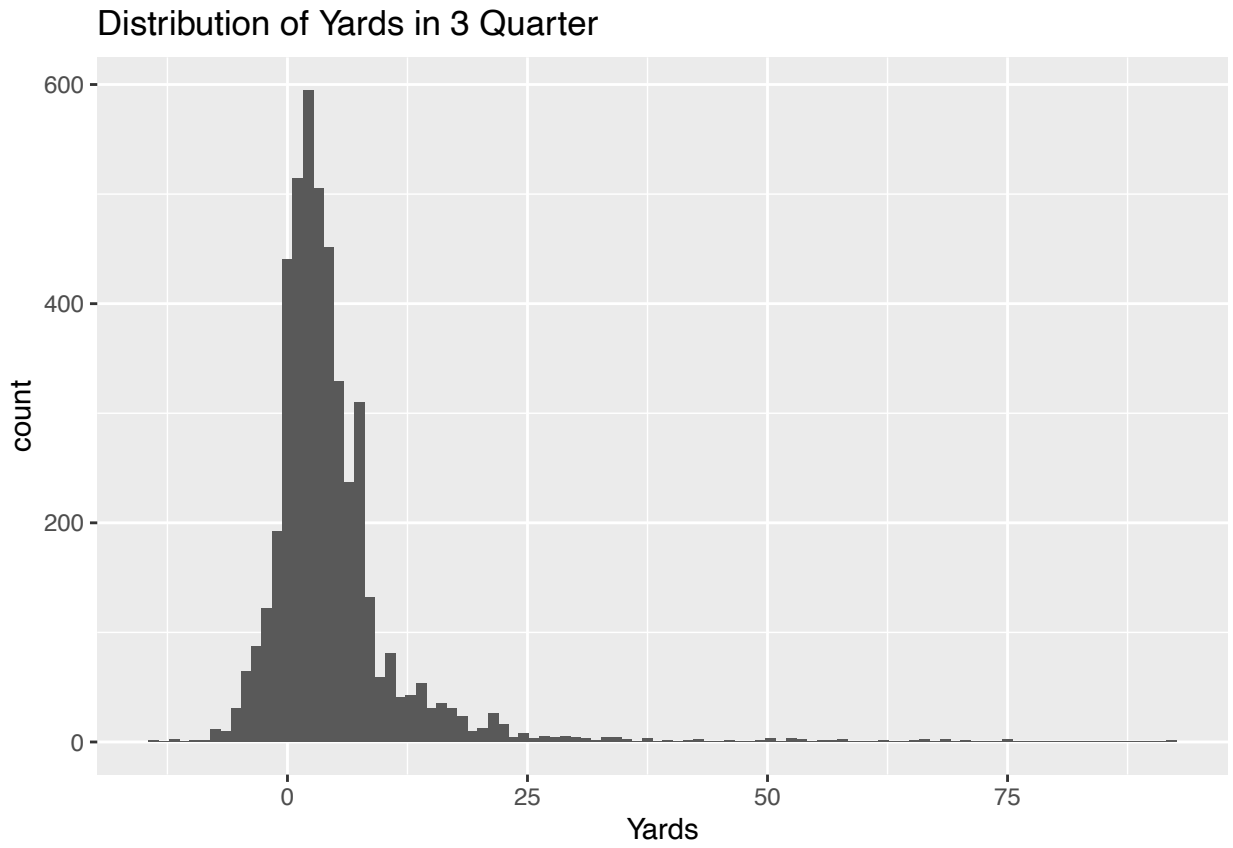
```r
ggplot(data = NFL_DATA_Train[(NFL_DATA_Train$Quarter == 1), ]) +
  geom_histogram(mapping = aes(x = Yards), bins = 100) +
  ggtitle("Distribution of Yards in 1 Quarter")
```



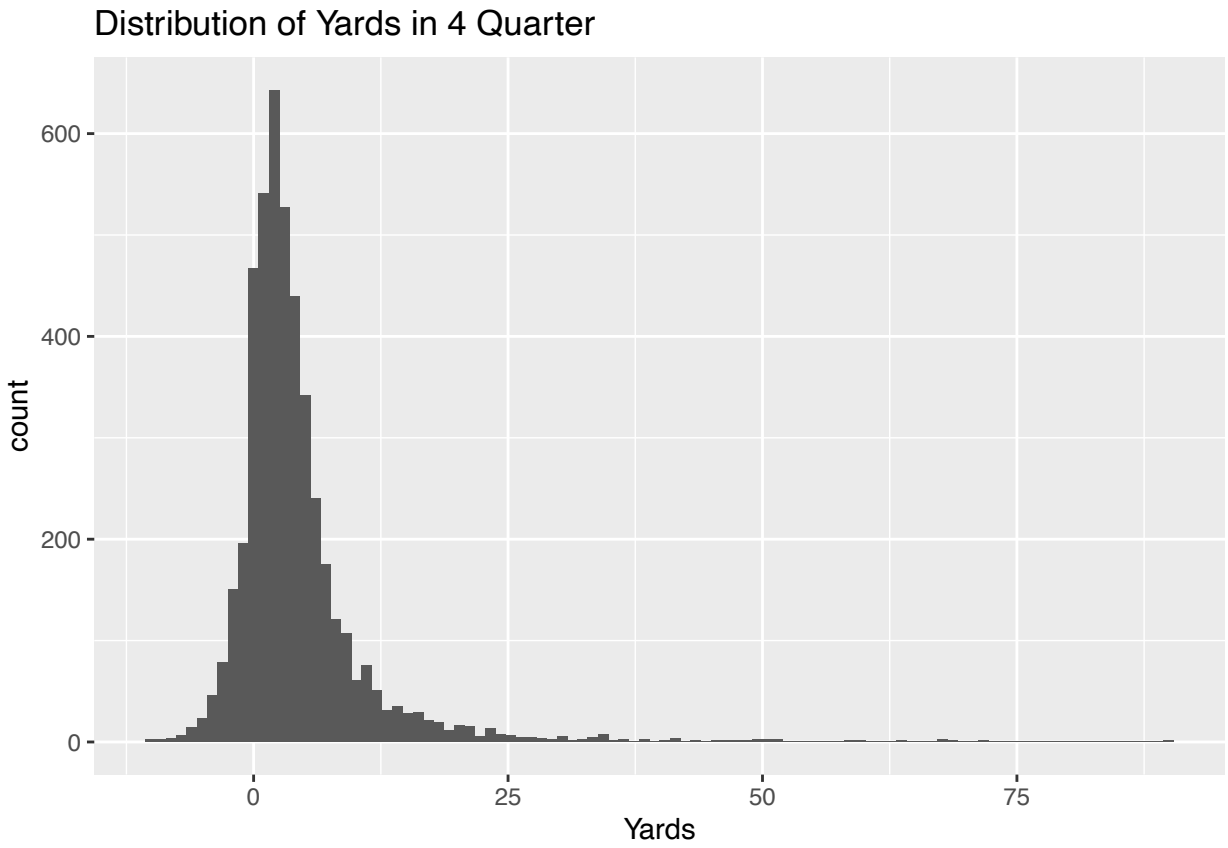

```r
# Second quarter
ggplot(data = NFL_DATA_Train[(NFL_DATA_Train$Quarter == 2), ]) +
  geom_histogram(mapping = aes(x = Yards), bins = 100) +
  ggtitle("Distribution of Yards in 2 Quarter")
```




```
# Third quarter
ggplot(data = NFL_DATA_Train[(NFL_DATA_Train$Quarter == 3), ]) +
  geom_histogram(mapping = aes(x = Yards), bins = 100) +
  ggtitle("Distribution of Yards in 3 Quarter")
```




```
# Fourth quarter
ggplot(data = NFL_DATA_Train[(NFL_DATA_Train$Quarter == 4), ]) +
  geom_histogram(mapping = aes(x = Yards), bins = 100) +
  ggtitle("Distribution of Yards in 4 Quarter")
```




# DATA EXPLORATION

```
# Plot 3d of X and Y position, and yards gotten
plot_ly(
  NFL_DATA_Train, x = ~X, y = ~Y, z = ~Yards, color = ~PossessionTeam) %>%
  add_markers() %>%
  layout(
    scene = list(xaxis = list(title = 'X'),
                 yaxis = list(title = 'Y'),
                 zaxis = list(title = 'Yards'))
  )
```

```
## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2 is 8
## Returning the palette you asked for with that many colors
```

```
## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2 is 8
## Returning the palette you asked for with that many colors
```

```
# Plot 3d of down and distance, and yards gotten
plot_ly(
  NFL_DATA_Train, x = ~Distance, y = ~Down, z = ~Yards, color = ~PossessionTeam) %>%
  add_markers() %>%
  layout(
    scene = list(xaxis = list(title = 'Distance'),
                 yaxis = list(title = 'Down'),
```

```
                  zaxis = list(title = 'Yards'))
)
```

```
## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2 is 8
## Returning the palette you asked for with that many colors
```

```
## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2 is 8
## Returning the palette you asked for with that many colors
```

# DATA MANIPULATION AND CLEANING

Adding the covariates that we want to include, and modififying the dataframe

```
NFL_DATA_TRAIN_Modified <- NFL_DATA_Train

# Function to take the difference in time from the dataframe
timeDifference <- function(time) {
  num <- gsub("[:]", "" , str_sub(time, 12, 19), perl=TRUE)
  hr <- ifelse(str_sub(num, 1, 2) == "00", 24, as.numeric(str_sub(num, 1, 2)))
  min <- as.numeric(str_sub(num, 3, 4))
  sec <- as.numeric(str_sub(num, 5, 6))
  newTime <- 3600*hr + 60 * min + sec
  return(newTime)
}
# Add Time_Difference between the snap and the handoff
NFL_DATA_TRAIN_Modified$TimeDifference <-
  timeDifference(NFL_DATA_TRAIN_Modified$TimeHandoff) - timeDifference(NFL_DATA_TRAIN_Modified$TimeSnap

# Add the Difference in Score by home score - visitor score
# Difference in Score (Pair with which team is winning (HomeScore-AwayScore))
NFL_DATA_TRAIN_Modified$HomeScoreAdvantage <-
  NFL_DATA_TRAIN_Modified$HomeScoreBeforePlay - NFL_DATA_TRAIN_Modified$VisitorScoreBeforePlay

# Add the age of the running player

# Change the birth dates to strings
NFL_DATA_TRAIN_Modified$PlayerBirthDate = as.character(NFL_DATA_TRAIN_Modified$PlayerBirthDate)
# Grab the Year for each of the running player
Birth_Year = str_sub(NFL_DATA_TRAIN_Modified$PlayerBirthDate, 7, 11)
# Grab Month of each running player
Birth_Month = str_sub(NFL_DATA_TRAIN_Modified$PlayerBirthDate, 1, 2)
# If Born in July (07) Have lived 5/12 of a year. ie (12 - (Birth_Month)) / 12
How_Much_Of_Year_Lived = (12 - as.numeric(Birth_Month)) / 12
Years_Lived = NFL_DATA_TRAIN_Modified$Season - as.numeric(Birth_Year)
Total_Years_Lived = Years_Lived + How_Much_Of_Year_Lived
NFL_DATA_TRAIN_Modified$PlayerAge = Total_Years_Lived

# Change HEIGHT to inches and continuous
Feet = as.numeric(str_sub(NFL_DATA_TRAIN_Modified$PlayerHeight, 1, 1))
Inches = as.numeric(str_sub(NFL_DATA_TRAIN_Modified$PlayerHeight, 3, 4))
Heights = (Feet * 12) + Inches
NFL_DATA_TRAIN_Modified$PlayerHeight = Heights
```

```
# Changes GAMECLOCK to Seconds.

NFL_DATA_TRAIN_Modified$GameClock = as.numeric(NFL_DATA_TRAIN_Modified$GameClock)

# FACTORING VARIABLES INTO CATEGORICAL

# Factor OFFENSE FORMATION
NFL_DATA_TRAIN_Modified$OffenseFormation = factor(NFL_DATA_TRAIN_Modified$OffenseFormation)

# DEFENDERS IN BOX (Need Categorical and Ordinal)
NFL_DATA_TRAIN_Modified$DefendersInTheBox = factor(NFL_DATA_TRAIN_Modified$DefendersInTheBox)
```

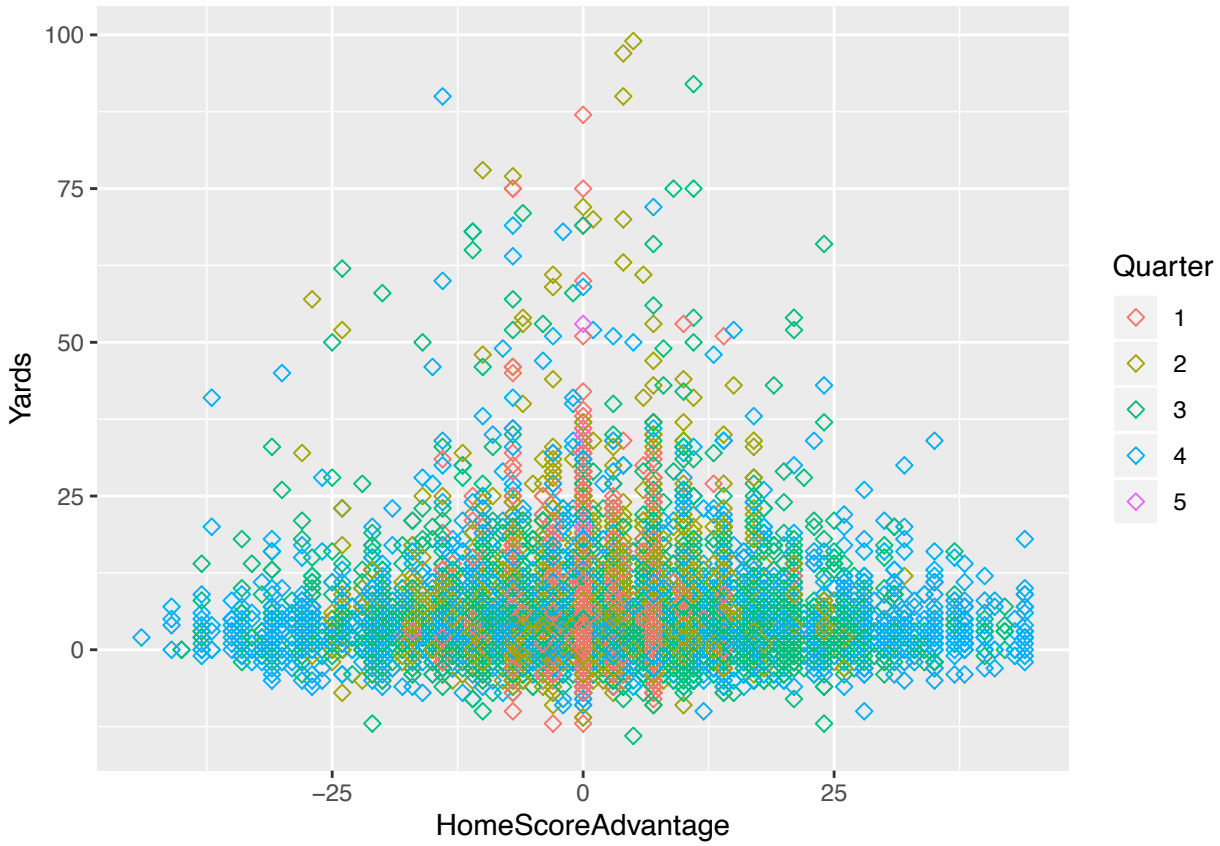
## MORE EXPLORATION WITH NEW/FACTORED COVARIATES (Still before Analyzing Techniques)

```
# Plot 3d of offense formation and defenders in box, and yards gotten. More defenders, less yards
# Might be ok, might only need 2 yards, same for 4th down
plot_ly(
  NFL_DATA_TRAIN_Modified, x = ~OffenseFormation, y = ~DefendersInTheBox, z = ~Yards, color = ~Possessi
  add_markers() %>%
  layout(
    scene = list(xaxis = list(title = 'OffenseForm'),
                 yaxis = list(title = 'DefendInBox'),
                 zaxis = list(title = 'Yards'))
)
```

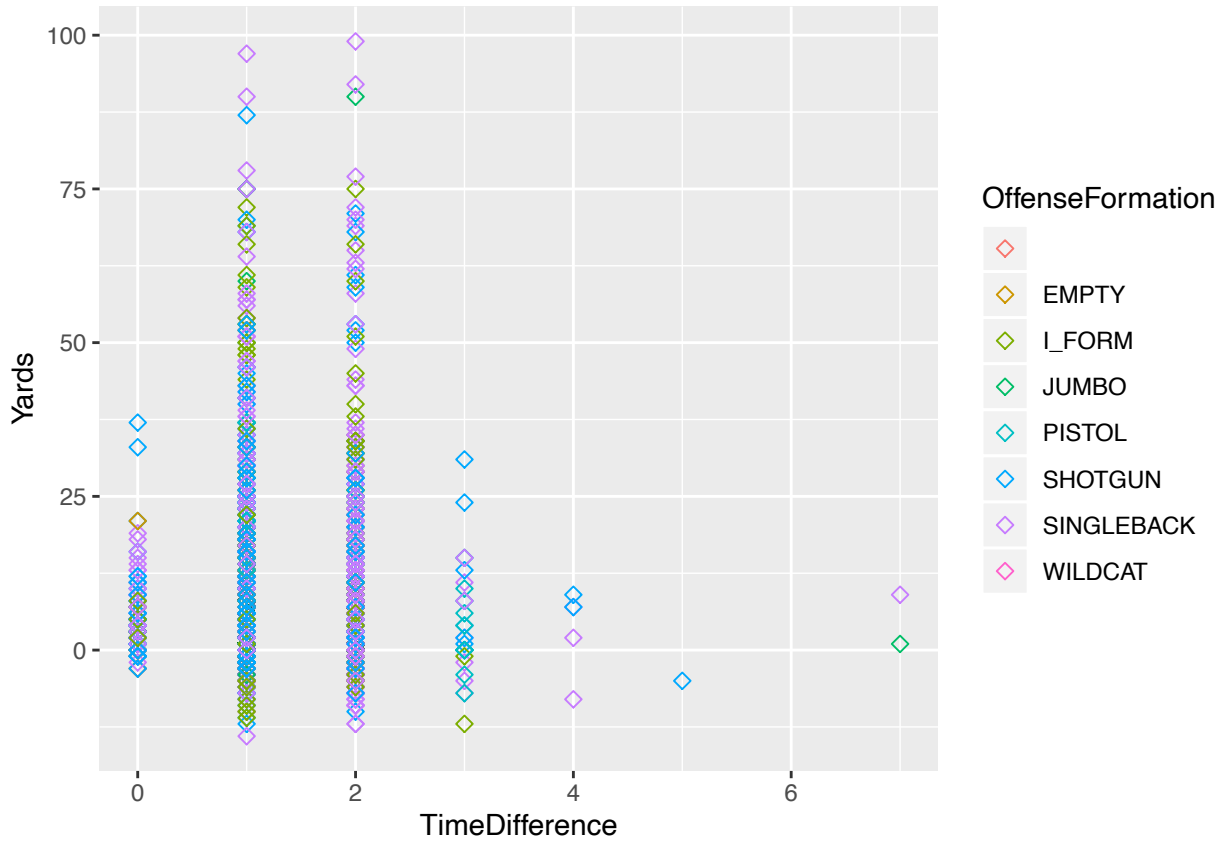
```
## Warning: Ignoring 2 observations
```

```
## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2 is 8
## Returning the palette you asked for with that many colors

## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2 is 8
## Returning the palette you asked for with that many colors
```

```
# Yards vs. Difference in Score, Color = Quarter
ggplot(NFL_DATA_TRAIN_Modified, aes(x=HomeScoreAdvantage, y=Yards, color = Quarter )) +
  geom_point(size=2, shape=23)
```




```
# Difference in score spreads out depending on the quarter (makes sense)
# Overtime looks to have highest average yards
# 4th quarter more run plays

# Time between handoff and yards, color = offense style
ggplot(NFL_DATA_TRAIN_Modified, aes(x=TimeDifference, y=Yards, color = OffenseFormation)) +
  geom_point(size=2, shape=23)
```




```
# GameClock, Quarter, Yards
plot_ly(
  NFL_DATA_TRAIN_Modified, x = ~GameClock, y = ~Quarter, z = ~Yards, color = ~factor(DefendersInTheBox)
  add_markers() %>%
  layout(
    scene = list(xaxis = list(title = 'Game Clock'),
                 yaxis = list(title = 'Quarter'),
                 zaxis = list(title = 'Yards'))
  )
```

```
## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2 is 8
## Returning the palette you asked for with that many colors

## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2 is 8
## Returning the palette you asked for with that many colors
```

# REFACTOR DEFENDER IN BOX TO INCLUDE ORDINALITY

```
# Leaving them unordered was for graphs above, to look good
NFL_DATA_TRAIN_Modified$DefendersInTheBox = factor(NFL_DATA_TRAIN_Modified$DefendersInTheBox, order = T
```

## SELECTION OF COVARIATES FOR ANALYSIS

```
# Drop columns that are collinear, or we think are not critical to our model
NFL_DATA_TRAIN_Filtered = select(NFL_DATA_TRAIN_Modified,
                                          -GameId, -PlayId, -Team, -S, -A, -Dis,
                                          -Orientation, -Dir, -DisplayName, -JerseyNumber,
                                          -YardLine, -FieldPosition, -HomeScoreBeforePlay,
                                          -VisitorScoreBeforePlay, -NflId, -TimeHandoff,
                                          -TimeSnap, -PlayerBirthDate, -PlayerCollegeName, -Location,
                                          -WindSpeed, -WindDirection, -StadiumType, -Turf,-GameWeather,
# Turf and stadium type all captured in stadium
# View(NFL_DATA_TRAIN_Filtered) # drop game weather, captured in Week, and Stadium. Also too many missi
```

## NOW THAT HAVE SELECTED COVARIATES. MORE DATA CLEANING, FACTORING, ETC...

```
# Need to count how many NA / Empty cells there are for each column
# summary(NFL_DATA_TRAIN_Filtered)  # changed empty to NA when reading in file
# GameWeather, Temperature, Humidy all have missing or NA data
# sum(is.na(NFL_DATA_TRAIN_Filtered$GameWeather))
# sum(NFL_DATA_TRAIN_Filtered$GameWeather == "")

# Factor the DOWNS, Ordinally
NFL_DATA_TRAIN_Filtered$Down = factor(as.numeric(NFL_DATA_TRAIN_Filtered$Down), order = TRUE, levels = 

# Player WEIGHT
NFL_DATA_TRAIN_Filtered$PlayerWeight = as.numeric(NFL_DATA_TRAIN_Filtered$PlayerWeight)

# Factor POSITION
NFL_DATA_TRAIN_Filtered$Position = factor(NFL_DATA_TRAIN_Filtered$Position)

# factor POSITION
NFL_DATA_TRAIN_Filtered$Position = factor(NFL_DATA_TRAIN_Filtered$Position)

# factor SEASON
NFL_DATA_TRAIN_Filtered$Season = factor(NFL_DATA_TRAIN_Filtered$Season)


# DATA CLEANING (REMOVING NA'S, and observations that happen less than 3 times)
# This was causing issues where one fold has a factor but another fold does not

# Need to delete a row within a column if there is just 1 special case. (Minimun 3 observations)

# DefensePersonnel (reduces observations by 11)
NFL_DATA_TRAIN_Filtered = NFL_DATA_TRAIN_Filtered[unsplit(table(NFL_DATA_TRAIN_Filtered$DefensePersonne

# Same for OffensePersonnel (reduces by 18 observations)
NFL_DATA_TRAIN_Filtered = NFL_DATA_TRAIN_Filtered[unsplit(table(NFL_DATA_TRAIN_Filtered$OffensePersonne

# Same for Position
```

```
NFL_DATA_TRAIN_Filtered = NFL_DATA_TRAIN_Filtered[unsplit(table(NFL_DATA_TRAIN_Filtered$Position), NFL_

# Same for Defenders In the Box
NFL_DATA_TRAIN_Filtered = NFL_DATA_TRAIN_Filtered[unsplit(table(NFL_DATA_TRAIN_Filtered$DefendersInTheB

# Same for Offense Formation
NFL_DATA_TRAIN_Filtered = NFL_DATA_TRAIN_Filtered[unsplit(table(NFL_DATA_TRAIN_Filtered$OffenseFormatio

# Need to remove NA Rows: Still 16,758 observations out of ~18,000
NFL_DATA_TRAIN_Filtered_Final <- na.omit(NFL_DATA_TRAIN_Filtered)
# View(NFL_DATA_TRAIN_Filtered_Final)
```

### BEGINNING OF ANALYSIS

## Regression with all Covariates

```
NFL_Train_Total_Model = lm(Yards ~ ., data=NFL_DATA_TRAIN_Filtered_Final)
NFL_Train_Total_Model.cv = cvFit(NFL_Train_Total_Model, data=NFL_DATA_TRAIN_Filtered_Final, y=NFL_DATA_
NFL_Train_Total_Model.cv # RMSE= 6.388  # May have collinearity
```

```
## 10-fold CV results:
##    CV
## 6.573
```

```
NFL_Train_Total_Model$coefficients
```

```
##                                (Intercept)
##                                 4.36400296
##                                          X
##                                 0.00136289
##                                          Y
##                                 0.01324105
##                                 Season2018
##                                 0.25591227
##                                   Quarter2
##                                 0.24042162
##                                   Quarter3
##                                 0.24208899
##                                   Quarter4
##                                 0.13308696
##                                   Quarter5
##                                 1.18448991
##                                  GameClock
##                                 0.00027695
##                           PossessionTeamATL
##                                 0.52279214
##                           PossessionTeamBLT
##                                 1.78413256
##                           PossessionTeamBUF
##                                 0.27577077
##                           PossessionTeamCAR
##                                 0.94340163
```

```
##                       PossessionTeamCHI
##                              0.28630036
##                       PossessionTeamCIN
##                             -1.03309945
##                       PossessionTeamCLV
##                              1.20458119
##                       PossessionTeamDAL
##                              0.76922709
##                       PossessionTeamDEN
##                              0.84305634
##                       PossessionTeamDET
##                              0.39927663
##                        PossessionTeamGB
##                              0.59790966
##                       PossessionTeamHST
##                              0.93271280
##                       PossessionTeamIND
##                              1.13432263
##                       PossessionTeamJAX
##                              0.64910649
##                        PossessionTeamKC
##                              0.74554290
##                        PossessionTeamLA
##                              0.88134301
##                       PossessionTeamLAC
##                              1.12016728
##                       PossessionTeamMIA
##                              0.66096948
##                       PossessionTeamMIN
##                              0.68265047
##                        PossessionTeamNE
##                              0.52885765
##                        PossessionTeamNO
##                              1.47683784
##                       PossessionTeamNYG
##                              0.73977295
##                       PossessionTeamNYJ
##                              0.47931667
##                       PossessionTeamOAK
##                              0.80688463
##                       PossessionTeamPHI
##                              0.81505089
##                       PossessionTeamPIT
##                              0.98387679
##                       PossessionTeamSEA
##                              0.00523551
##                        PossessionTeamSF
##                              1.09890839
##                        PossessionTeamTB
##                              0.03094075
##                       PossessionTeamTEN
##                              1.28413364
##                       PossessionTeamWAS
##                              0.14761205
```

```
##                                    Down.L
##                                -0.49376306
##                                    Down.Q
##                                -0.47586885
##                                    Down.C
##                                -0.24474125
##                                  Distance
##                                 0.06705364
##                      OffenseFormationI_FORM
##                                -0.45031458
##                       OffenseFormationJUMBO
##                                -0.15875945
##                      OffenseFormationPISTOL
##                                -0.33596008
##                     OffenseFormationSHOTGUN
##                                -0.38570103
##                  OffenseFormationSINGLEBACK
##                                -0.32342080
##                     OffenseFormationWILDCAT
##                                -0.56743232
##           OffensePersonnel0 RB, 2 TE, 3 WR
##                                 0.95596500
##           OffensePersonnel0 RB, 3 TE, 2 WR
##                                -2.03306037
##      OffensePersonnel1 RB, 0 TE, 3 WR,1 DB
##                                11.46310616
##           OffensePersonnel1 RB, 0 TE, 4 WR
##                                 2.78450477
##      OffensePersonnel1 RB, 1 TE, 2 WR,1 DB
##                                 7.04022435
##      OffensePersonnel1 RB, 1 TE, 2 WR,1 DL
##                                 1.62058401
##           OffensePersonnel1 RB, 1 TE, 3 WR
##                                 3.03476877
##      OffensePersonnel1 RB, 2 TE, 1 WR,1 DB
##                                 4.00775212
##      OffensePersonnel1 RB, 2 TE, 1 WR,1 DL
##                                 3.71319993
##      OffensePersonnel1 RB, 2 TE, 1 WR,1 LB
##                                 6.94399691
##           OffensePersonnel1 RB, 2 TE, 2 WR
##                                 2.98552445
##      OffensePersonnel1 RB, 3 TE, 0 WR,1 DL
##                                 2.99541932
##      OffensePersonnel1 RB, 3 TE, 0 WR,1 LB
##                                 5.06695308
##           OffensePersonnel1 RB, 3 TE, 1 WR
##                                 2.84947963
##           OffensePersonnel1 RB, 4 TE, 0 WR
##                                 2.87175018
##     OffensePersonnel2 QB, 1 RB, 0 TE, 3 WR
##                                 3.29414301
##     OffensePersonnel2 QB, 1 RB, 1 TE, 2 WR
##                                 3.30885850
```

```
##      OffensePersonnel2 QB, 1 RB, 2 TE, 1 WR
##                                  1.70215731
##      OffensePersonnel2 QB, 2 RB, 1 TE, 1 WR
##                                  7.03294294
##            OffensePersonnel2 RB, 0 TE, 3 WR
##                                  3.63855112
##            OffensePersonnel2 RB, 1 TE, 2 WR
##                                  3.38905662
##            OffensePersonnel2 RB, 2 TE, 1 WR
##                                  3.66270453
##            OffensePersonnel2 RB, 3 TE, 0 WR
##                                  2.43564786
##            OffensePersonnel3 RB, 0 TE, 2 WR
##                                  2.64966017
##            OffensePersonnel3 RB, 1 TE, 1 WR
##                                  2.70186985
##            OffensePersonnel3 RB, 2 TE, 0 WR
##                                  3.50371553
##      OffensePersonnel6 OL, 1 RB, 0 TE, 3 WR
##                                  3.58936284
## OffensePersonnel6 OL, 1 RB, 1 TE, 1 WR,1 DL
##                                  1.91550695
##      OffensePersonnel6 OL, 1 RB, 1 TE, 2 WR
##                                  2.38036923
## OffensePersonnel6 OL, 1 RB, 2 TE, 0 WR,1 DL
##                                  3.55171043
## OffensePersonnel6 OL, 1 RB, 2 TE, 0 WR,1 LB
##                                  4.07472385
##      OffensePersonnel6 OL, 1 RB, 2 TE, 1 WR
##                                  3.36056987
##      OffensePersonnel6 OL, 1 RB, 3 TE, 0 WR
##                                  3.57372879
##      OffensePersonnel6 OL, 2 RB, 0 TE, 2 WR
##                                  5.09858830
## OffensePersonnel6 OL, 2 RB, 1 TE, 0 WR,1 DL
##                                  3.01072723
##      OffensePersonnel6 OL, 2 RB, 1 TE, 1 WR
##                                  3.60057607
##      OffensePersonnel6 OL, 2 RB, 2 TE, 0 WR
##                                  4.06041580
##      OffensePersonnel7 OL, 1 RB, 0 TE, 2 WR
##                                  3.14554063
##      OffensePersonnel7 OL, 1 RB, 2 TE, 0 WR
##                                  3.62122212
##      OffensePersonnel7 OL, 2 RB, 0 TE, 1 WR
##                                  4.48123748
##                          DefendersInTheBox.L
##                                 -6.96017364
##                          DefendersInTheBox.Q
##                                  1.58359080
##                          DefendersInTheBox.C
##                                 -0.56595335
##                          DefendersInTheBox^4
##                                  0.22197237
```

```
##                          DefendersInTheBox^5
##                                   0.55012865
##                          DefendersInTheBox^6
##                                  -0.04065856
##                          DefendersInTheBox^7
##                                  -0.07257713
##                          DefendersInTheBox^8
##                                   0.06691407
##            DefensePersonnel0 DL, 5 LB, 6 DB
##                                  -1.90367142
##            DefensePersonnel1 DL, 3 LB, 7 DB
##                                  -1.63831555
##            DefensePersonnel1 DL, 4 LB, 6 DB
##                                  -1.64102659
##            DefensePersonnel1 DL, 5 LB, 5 DB
##                                  -0.43307818
##            DefensePersonnel2 DL, 2 LB, 7 DB
##                                   3.41441670
##            DefensePersonnel2 DL, 3 LB, 6 DB
##                                  -0.96370024
##            DefensePersonnel2 DL, 4 LB, 5 DB
##                                  -1.12173181
##            DefensePersonnel2 DL, 5 LB, 4 DB
##                                  -2.24849996
##            DefensePersonnel3 DL, 1 LB, 7 DB
##                                  -2.76827913
##            DefensePersonnel3 DL, 2 LB, 6 DB
##                                  -1.28256756
##            DefensePersonnel3 DL, 3 LB, 5 DB
##                                  -1.29077009
##            DefensePersonnel3 DL, 4 LB, 4 DB
##                                  -1.12210347
##            DefensePersonnel3 DL, 5 LB, 3 DB
##                                  -0.58904955
##            DefensePersonnel4 DL, 1 LB, 6 DB
##                                  -0.70220371
##            DefensePersonnel4 DL, 2 LB, 5 DB
##                                  -1.11749447
##            DefensePersonnel4 DL, 3 LB, 4 DB
##                                  -1.37702947
##            DefensePersonnel4 DL, 4 LB, 3 DB
##                                  -2.34898232
##            DefensePersonnel5 DL, 1 LB, 5 DB
##                                  -2.26531078
##            DefensePersonnel5 DL, 2 LB, 4 DB
##                                  -0.92526644
##      DefensePersonnel5 DL, 3 LB, 2 DB, 1 OL
##                                  -3.48683503
##            DefensePersonnel5 DL, 3 LB, 3 DB
##                                  -3.00113451
##            DefensePersonnel5 DL, 4 LB, 2 DB
##                                  -2.11957449
##            DefensePersonnel5 DL, 5 LB, 1 DB
##                                  -3.31938419
```

```
##            DefensePersonnel6 DL, 2 LB, 3 DB
##                                -3.18296116
##            DefensePersonnel6 DL, 3 LB, 2 DB
##                                -3.12072944
##            DefensePersonnel6 DL, 4 LB, 1 DB
##                                -3.00687073
##                           PlayDirectionright
##                                  -0.02558475
##                                 PlayerHeight
##                                  -0.02678516
##                                 PlayerWeight
##                                  -0.00018641
##                                   PositionFB
##                                    0.32109939
##                                   PositionHB
##                                    2.45225873
##                                   PositionQB
##                                  -0.60699802
##                                   PositionRB
##                                    0.48727282
##                                   PositionTE
##                                    2.33283997
##                                   PositionWR
##                                    2.64722612
##                              HomeTeamAbbrATL
##                                  -0.34076403
##                              HomeTeamAbbrBAL
##                                  -0.17950982
##                              HomeTeamAbbrBUF
##                                   0.26056064
##                              HomeTeamAbbrCAR
##                                   0.45528471
##                              HomeTeamAbbrCHI
##                                  -0.19392233
##                              HomeTeamAbbrCIN
##                                   0.35223469
##                              HomeTeamAbbrCLE
##                                  -0.03682803
##                              HomeTeamAbbrDAL
##                                   0.11077453
##                              HomeTeamAbbrDEN
##                                  -0.46466064
##                              HomeTeamAbbrDET
##                                  -0.06176218
##                               HomeTeamAbbrGB
##                                  -0.13296029
##                              HomeTeamAbbrHOU
##                                  -0.28582071
##                              HomeTeamAbbrIND
##                                  -0.16448220
##                              HomeTeamAbbrJAX
##                                   0.12239543
##                               HomeTeamAbbrKC
##                                  -0.18233362
```

```
##                          HomeTeamAbbrLA
##                              0.06118917
##                         HomeTeamAbbrLAC
##                             -0.20257936
##                         HomeTeamAbbrMIA
##                             -0.13993846
##                         HomeTeamAbbrMIN
##                              0.09406523
##                          HomeTeamAbbrNE
##                              0.83606442
##                          HomeTeamAbbrNO
##                             -0.14217071
##                         HomeTeamAbbrNYG
##                              0.17937087
##                         HomeTeamAbbrNYJ
##                              0.38781682
##                         HomeTeamAbbrOAK
##                              0.14923110
##                         HomeTeamAbbrPHI
##                              0.12023634
##                         HomeTeamAbbrPIT
##                             -0.00804839
##                         HomeTeamAbbrSEA
##                              0.37715297
##                          HomeTeamAbbrSF
##                             -0.31175042
##                          HomeTeamAbbrTB
##                              0.34331602
##                         HomeTeamAbbrTEN
##                              0.14974627
##                         HomeTeamAbbrWAS
##                              0.18317262
##                      VisitorTeamAbbrATL
##                              0.11926127
##                      VisitorTeamAbbrBAL
##                             -1.22593702
##                      VisitorTeamAbbrBUF
##                             -0.14853578
##                      VisitorTeamAbbrCAR
##                             -0.22602736
##                      VisitorTeamAbbrCHI
##                             -0.21665419
##                      VisitorTeamAbbrCIN
##                             -0.46257061
##                      VisitorTeamAbbrCLE
##                              0.02072874
##                      VisitorTeamAbbrDAL
##                             -0.05521551
##                      VisitorTeamAbbrDEN
##                             -0.01958155
##                      VisitorTeamAbbrDET
##                             -0.12727400
##                       VisitorTeamAbbrGB
##                              0.03796966
```

```
##                       VisitorTeamAbbrHOU
##                              -1.29096177
##                       VisitorTeamAbbrIND
##                              -0.51274503
##                       VisitorTeamAbbrJAX
##                               0.26487852
##                        VisitorTeamAbbrKC
##                              -0.04753699
##                        VisitorTeamAbbrLA
##                               0.65637090
##                       VisitorTeamAbbrLAC
##                              -0.41161468
##                       VisitorTeamAbbrMIA
##                              -0.03750437
##                       VisitorTeamAbbrMIN
##                              -0.28207140
##                        VisitorTeamAbbrNE
##                               0.32397861
##                        VisitorTeamAbbrNO
##                              -0.56852449
##                       VisitorTeamAbbrNYG
##                              -0.28675035
##                       VisitorTeamAbbrNYJ
##                              -0.38133070
##                       VisitorTeamAbbrOAK
##                              -0.02212047
##                       VisitorTeamAbbrPHI
##                              -0.08990205
##                       VisitorTeamAbbrPIT
##                              -0.52749840
##                       VisitorTeamAbbrSEA
##                               0.01376558
##                        VisitorTeamAbbrSF
##                              -0.13122065
##                        VisitorTeamAbbrTB
##                               0.07175978
##                       VisitorTeamAbbrTEN
##                              -0.52592300
##                       VisitorTeamAbbrWAS
##                               0.53538537
##                                     Week
##                              -0.01288763
##                             Temperature
##                              -0.00513192
##                                Humidity
##                              -0.00685989
##                          TimeDifference
##                               0.21244143
##                       HomeScoreAdvantage
##                              -0.00937183
##                                PlayerAge
##                              -0.05285723
```

## What if we always predicted the mean of yards? (Just Intercept Term)

```
NFL_Train_Total_Model1 = lm(Yards ~ 1, data=NFL_DATA_TRAIN_Filtered_Final)
NFL_Train_Total_Model1.cv = cvFit(NFL_Train_Total_Model1, data=NFL_DATA_TRAIN_Filtered_Final, y=NFL_DATA
NFL_Train_Total_Model1.cv # RMSE= 6.4191
```

```
## 10-fold CV results:
##     CV
## 6.5943
```

## Forward Stepwise Regression

```
min_model = NFL_Train_Total_Model1
max_model = NFL_Train_Total_Model
stepwise_model = step(min_model, direction='forward', scope=max_model)
```

```
## Start:  AIC=63218
## Yards ~ 1
```

```
summary(stepwise_model)
```

```
##
## Call:
## lm(formula = Yards ~ 1, data = NFL_DATA_TRAIN_Filtered_Final)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -18.28  -3.28  -1.28   1.72  94.72
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)   4.2793     0.0509      84 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.59 on 16757 degrees of freedom
```

```
# Ultimately, this is saying that the extra info we gain is not worth the complexity
```

## Backward Stepwise Regerssion

```
backward_step = step(max_model, direction='backward')
backward_step
```

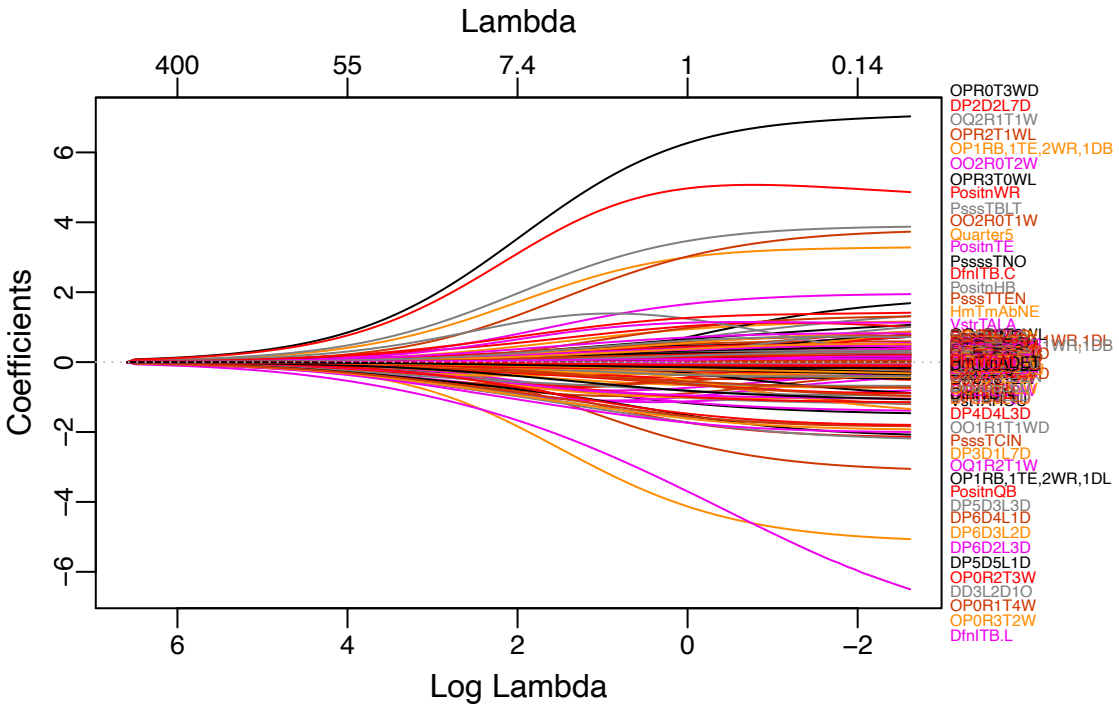
```
backward_step.cv = cvFit(backward_step, data=NFL_DATA_TRAIN_Filtered_Final, y=NFL_DATA_TRAIN_Filtered_F
backward_step.cv # RMSE= 6.5386 # Best Model so far
```

```
## 10-fold CV results:
##     CV
## 6.5386
```

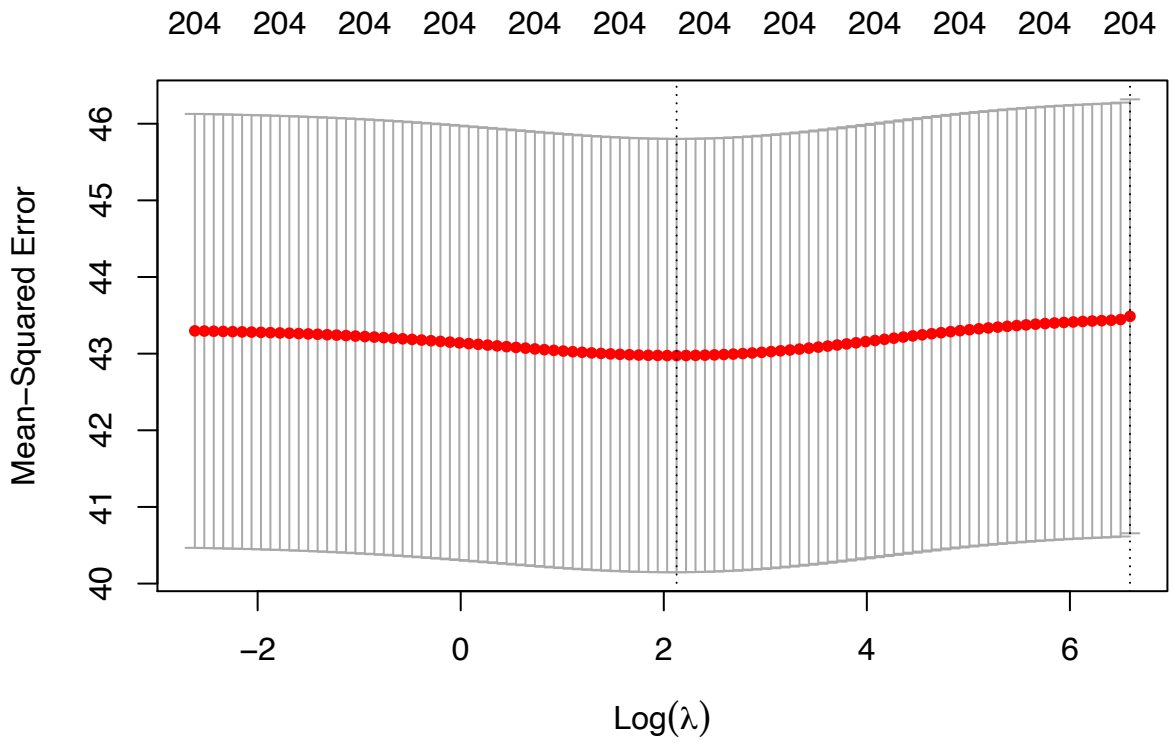
# Ridge Regression

```
library(glmnet)
## NORMALIZE Continouous Covariates # Will have to normalize defenders in box if we make continuous
Standardized_NFL_TRAIN = NFL_DATA_TRAIN_Filtered_Final
Standardized_NFL_TRAIN$X = scale(Standardized_NFL_TRAIN$X)
Standardized_NFL_TRAIN$Y = scale(Standardized_NFL_TRAIN$Y)
Standardized_NFL_TRAIN$GameClock = scale(Standardized_NFL_TRAIN$GameClock )
Standardized_NFL_TRAIN$Distance = scale(Standardized_NFL_TRAIN$Distance)
Standardized_NFL_TRAIN$PlayerHeight = scale(Standardized_NFL_TRAIN$PlayerHeight)
Standardized_NFL_TRAIN$PlayerWeight = scale(Standardized_NFL_TRAIN$PlayerWeight)
Standardized_NFL_TRAIN$Week = scale(Standardized_NFL_TRAIN$Week)
Standardized_NFL_TRAIN$Temperature = scale(Standardized_NFL_TRAIN$Temperature)
Standardized_NFL_TRAIN$Humidity = scale(Standardized_NFL_TRAIN$Humidity)
Standardized_NFL_TRAIN$TimeDifference = scale(Standardized_NFL_TRAIN$TimeDifference)
Standardized_NFL_TRAIN$HomeScoreAdvantage = scale(Standardized_NFL_TRAIN$HomeScoreAdvantage )
Standardized_NFL_TRAIN$PlayerAge = scale(Standardized_NFL_TRAIN$PlayerAge)

# Ridge Regression
# Ridge alpha = 0
x = model.matrix(Yards~. , Standardized_NFL_TRAIN)
y = Standardized_NFL_TRAIN$Yards
ridge_mod = glmnet(x, y, alpha = 0)
# install.packages("plotmo")
plot_glmnet(ridge_mod, label = TRUE)
```

```
## Warning in TeachingDemos::spread.labs(beta[iname, ncol(beta)], mindiff =
## 1.2 * : Maximum iterations reached
```




```
cvfit = cv.glmnet(x, y, alpha = 0)
plot(cvfit)
```




```
bestlam = cvfit$lambda.min # = 4.9384
```

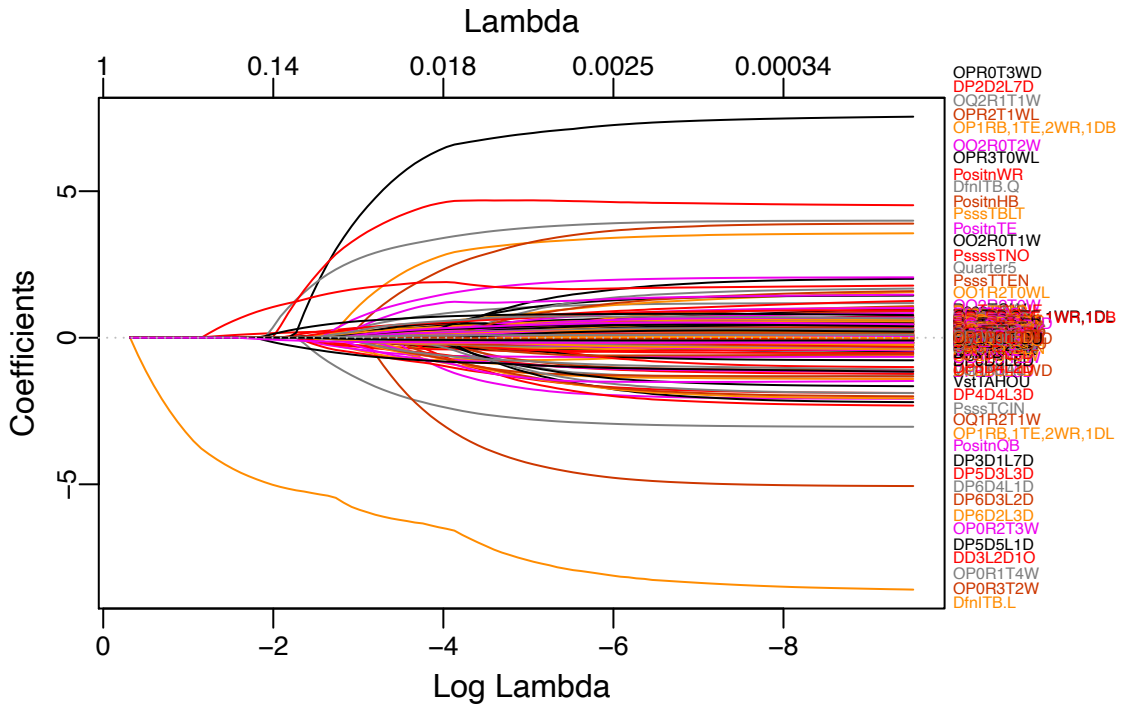
## Get coefficients when log lamda is 10.098

```
y_predicted <- predict(cvfit, s = bestlam, newx = x) # same x, in sample prediction
ridge_RMSE = sqrt(mean((y_predicted - y)^2))
# ridge_RMSE # = 6.3333
#coef(ridge_mod)[,4.9384] # Best is again basically forcing all the betas to 0. Just predict mean
```

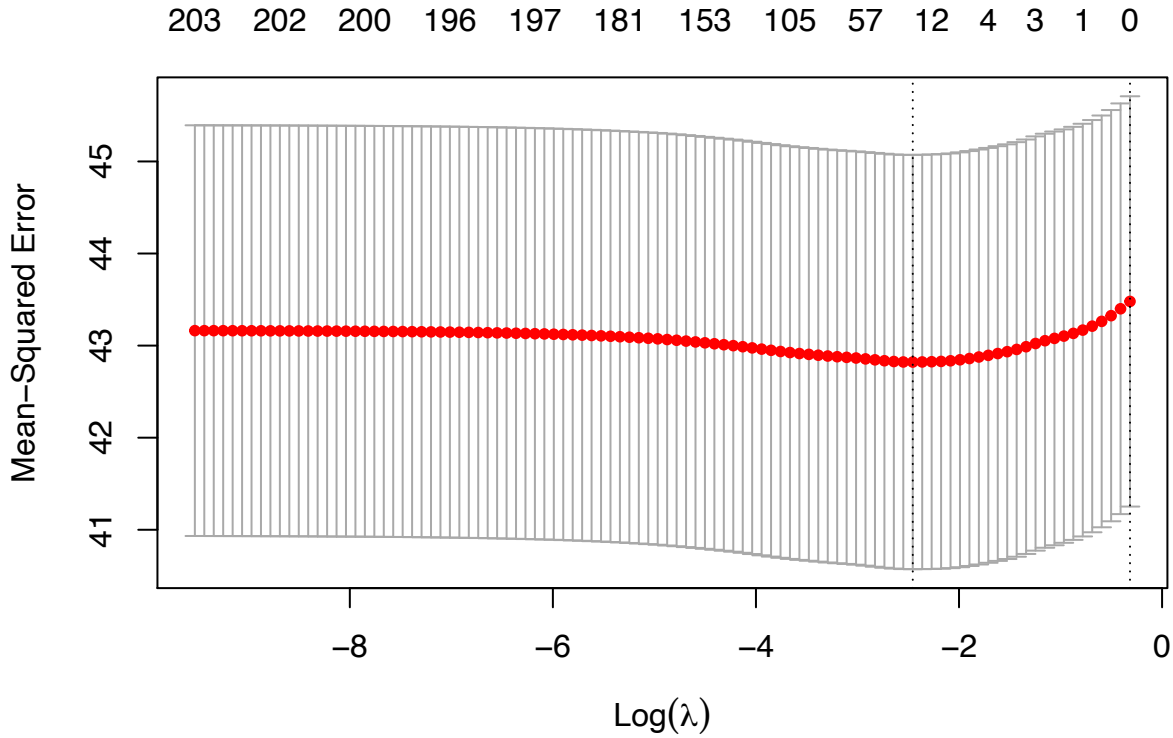
## Lasso Regression

```
lasso_mod= glmnet(x, y, alpha = 1)
# coef(lasso_mod)[,50]
plot_glmnet(lasso_mod, label = TRUE)
```

```
## Warning in TeachingDemos::spread.labs(beta[iname, ncol(beta)], mindiff =
## 1.2 * : Maximum iterations reached
```




```
cvfit_lasso = cv.glmnet(x, y, alpha = 1)
plot(cvfit_lasso)
```




```
bestlam_lasso = cvfit_lasso$lambda.min # = 0.046056
```

```
y_predicted_lasso <- predict(cvfit_lasso, s = bestlam_lasso, newx = x) # same x, in sample prediction
lasso_RMSE = sqrt(mean((y_predicted_lasso - y)^2))
lasso_RMSE # = 6.3349
```

```
## [1] 6.5326
```

```
out = glmnet(x, y, alpha = 1) # Fit ridge regression model on full dataset
#predict(out, type = "coefficients", s = bestlam_lasso)[1:80,] # Display coefficients using lambda chos
```

## Classification. Outcome Variable: NFL Yards >= distance. (Whether or not they get a first down)

### With all Covariates

```
NFL_Train_Total_Model_c <- NFL_DATA_TRAIN_Filtered_Final
NFL_Train_Total_Model_c$FirstDown <- ifelse(
  NFL_Train_Total_Model_c$Yards >= NFL_Train_Total_Model_c$Distance, 1, 0)
NFL_Train_Total_Model_c$FirstDown <-
  factor(NFL_Train_Total_Model_c$FirstDown)
NFL_Train_Total_Model_c <- select(NFL_Train_Total_Model_c, -Yards) #remove colinear response variable f
```

### Implementing Cross-Fold Validation for Classification

```
set.seed(122)
f <- createFolds(y=NFL_Train_Total_Model_c$FirstDown, k=10)
train_fold <- function (i) {
  NFL_Train_Total_Model_c[-unlist(f[i]),]
}

test_fold <- function (i) {
  NFL_Train_Total_Model_c[unlist(f[i]),]
}
```

### Baseline model

```
accuracyR <- c()

for (i in 1:10) {
  glm_Model = glm(FirstDown ~ ., data=train_fold(i),
                              family = binomial)
  predict_result <- predict(glm_Model, newdata=test_fold(i), type="response")
  predict_logit <- ifelse(predict_result > 0.5, 1, 0)
  t <- table(predict_logit, test_fold(i)$FirstDown)
  accuracyR[i] = (t[1,1]+t[2,2])/dim(test_fold(i))[1]
}
cat("Mean accuracy of 10-fold validation is: ", mean(accuracyR), "\n")
```

```
## Mean accuracy of 10-fold validation is:  0.81585
```

```
#print(mean(accuracyR)) #0.81585
confusionMatrix(test_fold(i)$FirstDown, factor(predict_logit))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1233   82
##          1  226  134
##
##                Accuracy : 0.816
##                  95% CI : (0.797, 0.834)
##     No Information Rate : 0.871
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.363
##
##  Mcnemar's Test P-Value : 0.000000000000000369
##
##             Sensitivity : 0.845
##             Specificity : 0.620
##          Pos Pred Value : 0.938
##          Neg Pred Value : 0.372
##              Prevalence : 0.871
##          Detection Rate : 0.736
##    Detection Prevalence : 0.785
##       Balanced Accuracy : 0.733
##
##        'Positive' Class : 0
##
```

Why you don't use in-prediction error - biased -lower error or higher accuracy

```
  glm_Model = glm(FirstDown ~ ., data=NFL_Train_Total_Model_c,
                                family = binomial)
  predict_result <- predict(glm_Model, newdata=NFL_Train_Total_Model_c, type="response")
  predict_logit <- ifelse(predict_result > 0.5, 1, 0)
  t <- table(predict_logit, NFL_Train_Total_Model_c$FirstDown)
  accuracyR = (t[1,1]+t[2,2])/dim(NFL_Train_Total_Model_c)[1] #0.82438
```

#Penalized Logistic Regression #Ridge error in prediction

## Lasso

```
lasso_mod= glmnet(x, y, alpha = 1)
```

## SVM

have to use parallelSVM since SVM from e1071 would time out.

```
accuracyR <- c()
for (i in 1:10) {
  parallel_svm = parallelSVM(FirstDown ~ ., data=train_fold(i),
                             numberCores=3,
                             kernel="radial", cost = 1, gamma = 0.05)
```

```
  predict_result <- predict(parallel_svm,
                            newdata=test_fold(i), type="response")
  t <- table(predict_result, test_fold(i)$FirstDown)
  accuracyR[i] = (t[1,1]+t[2,2])/dim(test_fold(i))[1]
}
cat("Mean accuracy of 10-fold validation is: ", mean(accuracyR))
```

```
## Mean accuracy of 10-fold validation is:  0.81328
```

```
#print(mean(accuracyR)) #0.81483
confusionMatrix(test_fold(i)$FirstDown, factor(predict_result))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1250   65
##          1  240  120
##
##                Accuracy : 0.818
##                  95% CI : (0.799, 0.836)
##     No Information Rate : 0.89
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.345
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.839
##             Specificity : 0.649
##          Pos Pred Value : 0.951
##          Neg Pred Value : 0.333
##              Prevalence : 0.890
##          Detection Rate : 0.746
##    Detection Prevalence : 0.785
##       Balanced Accuracy : 0.744
##
##        'Positive' Class : 0
##
```

# Ada

```
accuracyR <- c()
dat <- NULL
for (i in 1:10) {
  ada_fits <- ada(FirstDown ~ ., data = train_fold(i), iter = 50,
                  type="discrete", loss="exponential", nu=1)
  predict_ada <- predict(ada_fits, newdata=test_fold(i))
  t <- table(predict_ada, test_fold(i)$FirstDown)
  accuracyR[i] = (t[1,1]+t[2,2])/dim(test_fold(i))[1]
}
cat("Mean accuracy of 10-fold validation is: ", mean(accuracyR))
```

```
## Mean accuracy of 10-fold validation is:  0.81066
```

```
#print(mean(accuracyR)) #0.81251
confusionMatrix(test_fold(i)$FirstDown, factor(predict_ada))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1237   78
##          1  247  113
##
##                Accuracy : 0.806
##                  95% CI : (0.786, 0.825)
##     No Information Rate : 0.886
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.307
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.834
##             Specificity : 0.592
##          Pos Pred Value : 0.941
##          Neg Pred Value : 0.314
##              Prevalence : 0.886
##          Detection Rate : 0.739
##    Detection Prevalence : 0.785
##       Balanced Accuracy : 0.713
##
##        'Positive' Class : 0
##
```