MSDS 6372 Project 2

Zackary Gill, Johnny Gipson, Satish Mylapore

1. Introduction

Every NFL season there are numerous games where field goals are a significant decider in who wins a game. The ability to predict whether a field goal will be successfully scored would help teams decide if a field goal should be attempted. Our team will first create an easily interpretable logistic model. Next we will use all tools at our disposal to create predictive models using PCA, LDA, Random Forests, and a complex logistic model. To accomplish this goal we have a dataset that we will use to determine what predictors are good and use a combination of the accuracy, sensitivity, and specificity to show the correctness of our models.

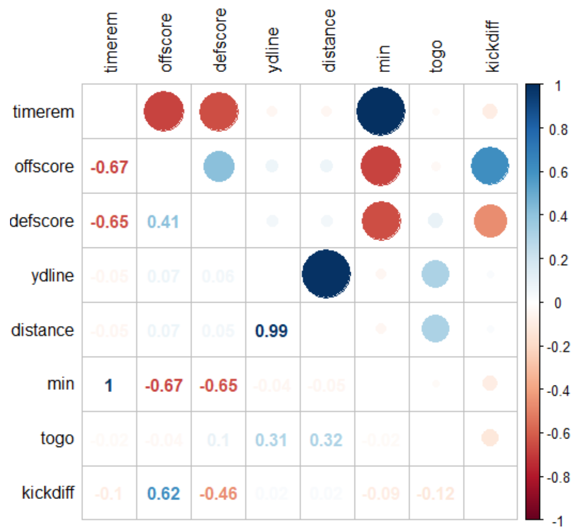
2. Data Description

The data set includes the results of all NFL regular season field goal attempts for the 2008 season. The original data set has 23 variables with 1039 rows. The data set has three binary outcomes for field goals (of which we will obviously use only one as a response variable). That variable 'GOOD' is represented by a 1 if the field goal was successful and a 0 if it missed. The source of the data set is from advancedfootballanalytics.com. The data set was obtained from the university of Florida data science website. To see a list of data variables and data description please refer to the appendix. For more information go to: http://users.stat.ufl.edu/~winner/datasets.html

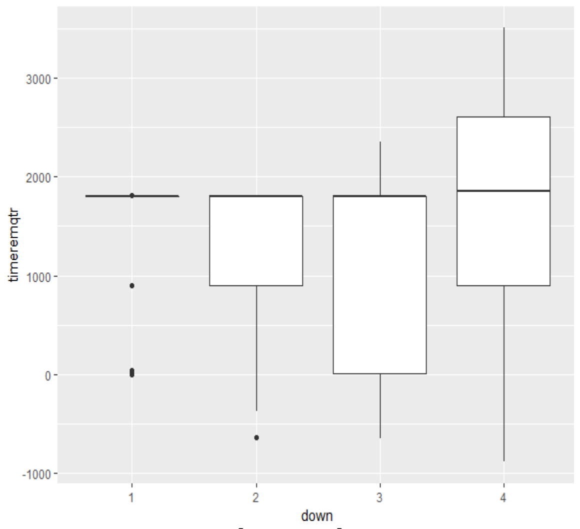
3. Exploratory Data Analysis

Using our football knowledge, usually a missed field goal is usually caused by the long distance and usually the time left remaining in the game. Investigating a plot of the two variables, we definitely see that the missed occur at all distance but have a heavy cluster between 40 plus yards but we also a lot field goals made in the 40 to 50 yd range (please refer to appendix).

In our data set in addition to our response variable 'GOOD', there are two variables 'Missed' and 'Blocked' that overlap in its indication. We have removed these. The dataset contained 2 rows that each contained 2 columns of missing data. Upon examining those rows we used information from pro-football-reference.com and our knowledge of football to determine the values of that missing data. For our analysis we decided that we did not want to limit the predictive capabilities to just the 2008 season and to those kickers, we removed these columns: 'GameDate', 'name', 'kicker', 'season', 'HomeTeam', 'AwayTeam', 'kickteam', and 'def'.



[CHART 3.1]



[CHART 3.2]

Looking at the correlation of continuous variables **[CHART 3.1]** we noticed that ‘ydline’ and ‘distance’ are nearly identical (0.99 correlation) and so we removed ‘ydline’. ‘timerem’, ‘min’, and ‘sec’ were correlated so we created a new variable ‘timeremqtr’ that indicates how much time is remaining in the quarter and removed ‘min’ and ‘sec’. ‘kickdiff’ is correlated with ‘offscore’ and ‘defscore’ so we kept ‘kickdiff’ and removed the other two.

Next we made tables of the categorical variables (qtr, down, homekick, and GOOD) in groupings of two. We discovered that ‘qtr’ had several groupings of data that had zero information and also had very few values during overtime. Because this would impede our ability to train/test we removed ‘qtr’.

Looking plots of continuous vs categorical (example in **[CHART 3.2]**), we came across three instances of multi-collinearity. They were between [timeremqtr, down], [timerem, down], and [togo, down]. Running a Kruskal-Wallis ANOVA test (nonparametric test, normality is not a requirement) resulted in there being definitive evidence of this (all p-values < 3.637e-07). Because ‘down’ was one of the culprits every time, that variable was removed.

Finally, a vif (variance inflation factor) was performed on the rest of the variables. The vif quantifies the severity of multicollinearity in regression analysis. As a rule of thumb is that if a VIF value exceeds the number 10 then we have multicollinearity. The rest of potential variables have values less than two (please refer to Appendix Table **[CHART 6.d.1]**) so there is no collinearity between the variables. At this point the cleaning portion of the data analysis was completed.

4. Objective 1

a. Problem Restatement

Every NFL season there are numerous games where field goals are a significant decider in who wins a game. The ability to predict whether a field goal will be successfully scored would help teams decide if a field goal should be attempted. Our team will create a logistic model to predict field goal success in the NFL. First, we build a simplistic logistic model. What do we mean by simplistic logistic model? The below equation will be used as a basis for our simplistic logistic model:

$$\ln\left(\frac{p(Good\ Field\ Goal)}{1-p(XGood\ Field\ Goal)}\right) = \beta_0 + \beta_1 X$$

We will be using model selection techniques to do our variable selection and these computer model selection will not take into complex terms like interaction or transformations. We will be using the model selection techniques of forward selection, stepwise, and LASSO for variable selection. We will be using estimator of relative quality such as AIC (Akaike Information Criteria), AUC, accuracy, sensitivity and specificity to determine what is the best model. Once a model is selected then an interpretation of the of the model regression coefficients including hypothesis testing and confidence intervals. We will also comment on the practical and statistical significance of the important factors. We will not be including interaction terms at this time.

b. Formation of Training and Test Set

For this objective, the typical way of doing random sampling of the data set to obtain train and test was not the best option. In below summary statistics table **[TABLE 4.b.1]**, you can see our categorical response ‘GOOD’ has a very small number of missed field goals (0’s) with respect to the number of successful field goals (1’s).

Summary Stats	**Togo**	**Distance**	**Homekick**	**Kickdiff**	**Timerem**	**Good**	**Timeremqtr**
Min	1.0	18.0		-41.0	-887.0		1.0
1st Quartile	4.0	28.0		-6	895.5		113.0

Medial	6.0	37.0		0	1808.0		326.0
Mean	6.7	36.7		0.4	1704.6		352.5
3rd Quartile	9	44.0		6.0	2556.5		579.5
Max	25	76.0		44.0	3507.0		900.0
0 (No)			525			139	
1 (Yes)			514			900	

[TABLE 4.b.1 – Summary Statistics]

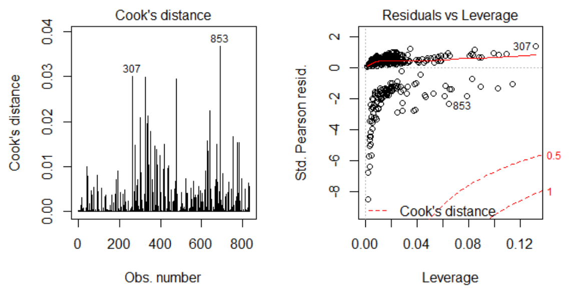
We initially went with making the number of 1's and 0's for GOOD in the training set equal, but because of the low number of 0's available in the data set this option gave poor results. The decision was made to split the data 80/20 (80% in the training, 20% in the test set) to ensure that there was enough data for training. For that 80/20 split we made 80% of the 0's be in the training set and 20% of them in the test set. This helped ensure that the model would consistently predict/interpret both the successful field goals and the misses.

c. Assumptions and Influential Points

Binary Dependent Observation: binary logistic regression requires the dependent variable to be binary as in our case the dependent variable field goal kick 'GOOD' is a binary (factor with 1/0).

Independent Observation: This data was collected by (advancedfootballanalytics.com) from the 2008 NFL season. Because of this we consider these observations to be independent.

Influential points



[CHART 4.c.2 – Cook's D and Residuals vs Leverage]

Two points have been observed with Cook's D value **[CHART 4.c.2]** higher than other values. However, not all outliers are influential observations. Data points 853 and 307 are still under 0.04 (less than 4/n = 0.004807692), hence these points are not considered as outliers. We also can see that in the data. We also see that in Residuals vs Leverage **[CHART 4.c.2]** shows no leverage on this dataset.

d. Model Selection – Forward, Stepwise, and LASSO

Forward Selection is a model selection technique that begins with just the intercept and then sequentially adds the effect that most improves the fit. The process terminates when no significant improvement can be obtained by adding variables. The forward selection process produces the following model.

$$\ln\left(\frac{p(Good\ Field\ Goal)}{1-p(XGood\ Field\ Goal)}\right) = 7.06 - 0.12\text{*distance} - 0.37965\text{*homekick1}$$

In the stepwise model selection, a variable is considered for addition to or subtraction from the set explanatory variables based on some prescribed criteria. In our stepwise model, the prescribed criteria was AIC. The stepwise model selection produce a model where the two predictors which were 'distance' and 'homekick1'.

$$\ln\left(\frac{p(Good\ Field\ Goal)}{1-p(XGood\ Field\ Goal)}\right) = 6.418 - 0.12\text{*distance} - 0.37965\text{*homekick1}$$

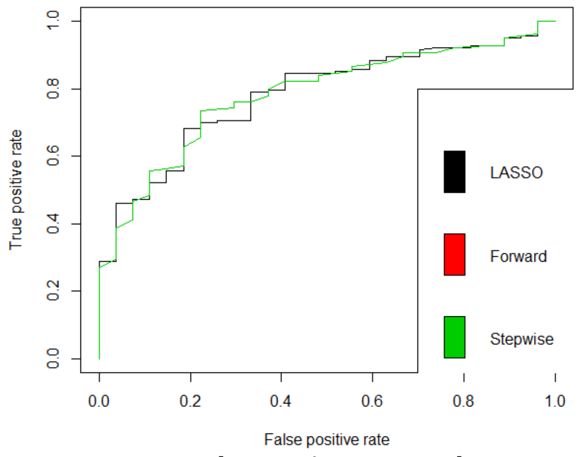
LASSO (least absolute shrinkage and selection operator) is a penalized regression model technique that is used to select a subset of variables. Using cross validation we computed a lambda of 0.009502046; the final LASSO model is:

$$\ln\left(\frac{p(Good\ Field\ Goal)}{1-p(XGood\ Field\ Goal)}\right) = 6.406 - 0.111*\text{distance} - 0.257*\text{homekick1} + 0.00029*\text{timeremqtr}$$

Model	AIC	AUC	Accuracy	Specificity	Sensitivity
Forward	554.48	0.782	0.8696	0.037	0.99
Stepwise	554.48	0.782	0.8696	0.037	0.99
LASSO		0.782	0.87	0.037	1.0

[TABLE 4.d.1 – Initial Model Results]

Diving into our models and looking at the confusion matrix, all of our models have great accuracy but poor specificity. Sensitivity is the true positive rate, when actually yes, how often does it predict yes? Our sensitivity is incredibly high. Specificity is the true negative rate, when it's actually no how often does it predict no? To gain more insight, the ROC curves were investigated and the area under the curve for all models was above 75% (see **[CHART 4.d.1]** and AUC in **[TABLE 4.d.1]**). Note that the stepwise and forward selection's ROC curves **[CHART 4.d.1]** are practically identical which is why you can't see the ROC curve for forward selection.



[CHART 4.d.1 – ROC Curves]

The curves look very good but we need to choose a new cut point to fix the low specificity. After choosing a new cut point value, in **[TABLE 4.2]** below you will see that sensitivity and specificity are much better balanced. Although accuracy took a hit, it was necessary to have the model be decently good at predicting both the successful and missed field goals. AUC was used to be the deciding factor on which model was best. Based on this metric LASSO was chosen as the best model.

Model	AIC	AUC	Accuracy	Specificity	Sensitivity
Forward	554.48	0.782	0.79	0.59	0.82
Stepwise	554.48	0.782	0.79	0.59	0.82
LASSO		0.787	0.8068	0.59	0.83889

[TABLE 4.d.2 – Model Results after cut point adjustment]

e. Interpretation

Based on the above model metrics, the most accurately predicting model is LASSO. To generate the confidence intervals for interpretation the model was run using the R function glm with the variables from LASSO's output. Because LASSO does not use a p-value for the decision of what parameter to select some of the confidence intervals from glm include zero.

The odds ratio for making a field goal for a 1 yard decrease in distance (with everything else held constant) is 1.117 with a 95% confidence interval of [1.099, 1.1617]. The odds ratio of making a field goal while away from home with everything else held constant is 1.293 times higher than when when at home (95% confidence interval [0.97, 2.33]). The odds ratio for making a field goal for a 1 second increase in time before the end of a quarter (with everything else held constant) is 1.00029 with a 95% confidence interval of [0.9997, 1.0013]. A person could consider the 'timeremqtr' variable for our LASSO model to be insignificant since the odds ratio is so close to one. However the variable does provide what our model needs to make a more accurate prediction for making or missing a field goal.

f. Conclusion

Using data from the 2008 NFL season we wished to create a model that will explain/predict whether a team will successfully score a field goal. Using logistic regression combined with three automatic selection techniques (forward, stepwise, and LASSO) we created models based on the explanatory variables resulting from the exploratory data analysis. These models included no interaction or polynomial terms. We found that of the three, the LASSO model provided the best prediction by a narrow margin (0.01 more Accuracy). In objective 2 we will be demonstrating a few different techniques: LDA, Random Forest, and another logistic model using interactions and polynomials. In objective 2's conclusion we will compare and contrast the resulting models from both objective 1 and 2.

5. Objective 2

a. Introduction

In objective 1, a simple logistic model was created to predict the success of NFL field goal kicks based on a set of predictors using 2008 NFL data set. Using this simple model as a baseline, we will create complex logistic model using interaction terms and will also investigate competing models such as LDA and Random Forest. To investigate of clean data set further we will use Principle Component Analysis to gleam insight into the explanatory variables of data set and too see if we have any separation of principle components by the response. If the responses separate out, then predictive such as LDA, logistic, and random forest typically will predict well.

b. Principle Component Analysis as Explanatory Analysis

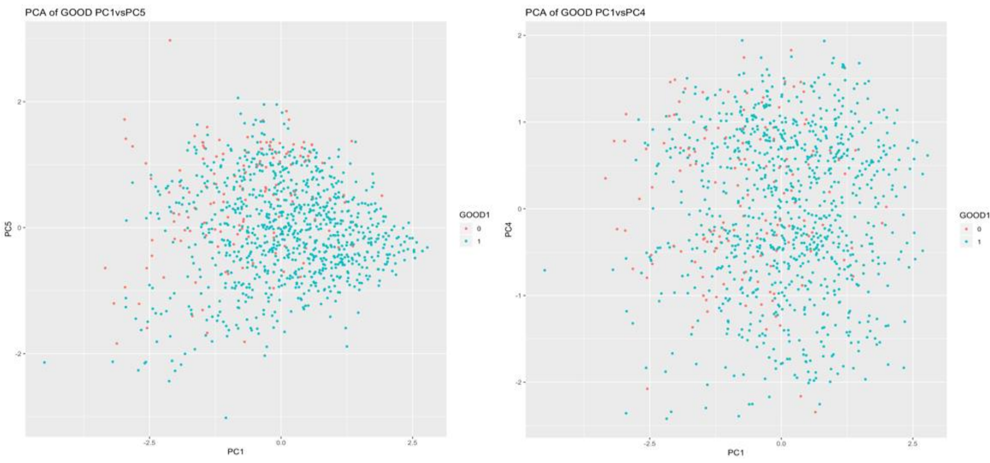
Principle Component Analysis was used as additional explanatory technique on our continuous variables of our clean data set. Why not us the entire data set? In objective 1 explanatory data analysis, we had already removed explanatory variables that we deem would not help us like the variable kicker. We wanted to make a truly predictive model, if kicker was included it would hamper our ability to predict the success of field goal's because the model would not be able to be used for new kickers. Second, our initial data set also contained a lot categorical explanatory variables. Those are removed for this PCA because we only want to look at the continuous variables for Linear Discriminate Analysis. If you look at the table below all of our continuous variables are weighted differently through our principle components which seem promising of some separation.

	PC1	**PC2**	**PC3**	**PC4**	**PC5**
togo	-0.64	0.3	-0.04	-0.07	-0.7
distance	-0.63	0.11	-0.42	0.02	0.65
kickdiff	-0.07	-0.57	-0.72	0.25	-0.3
timerem	0.25	0.61	-0.21	0.72	-0.02

timeremqtr	0.36	0.43	-0.51	-0.64	-0.03

[TABLE 5.b.1 – PCA Results]

Through the creation of scatter matrix of principle components (Appendix **[CHART 6.d.2]**) coded by our response there are few of graphs that have a separation. PC1 vs PC5 and PC1 vs PC4 **[CHART 5.b.1]** shows the most separation of the response.



[CHART 5.b.1 – PC1 vs PC5 and PC1 vs PC4]

There are no distinct clouds but there seems to be some areas of concentration of the different responses that gives some confidence that LDA and random can give us a decent model that will predict well.

c. Model Building Selection – Complex Logistic, LDA, and Random Forest

For the complex logistic model we decided that because the three automatic selection techniques had very little difference between their outcomes, we would use stepwise selection with the prescribed criteria being AIC. In this model we included all parameters, all 2-way interactions, and the square of each parameter. As seen below this result is exactly the same as the ones produced in Objective 1 for the forward and stepwise selections. See Appendix **[CHART 6.d.4]** for the ROC curve.

$$\ln\left(\frac{p(Good\ Field\ Goal)}{1-p(XGood\ Field\ Goal)}\right) = 7.0639 - 0.12*\text{distance} - 0.37965*\text{homekick1}$$

Linear discriminant analysis (LDA is a method used in machine learning to find a linear combination of features that separates two or more classes of objects or events. The resulting combination may be used as a linear classifier.

From our PCA explanatory work, the five continuous variables appear to create some separation between the responses of a good and bad field goal and LDA may be a good choice in prediction. Using LDA, the coefficients of linear discriminant output provides a linear combination our five continuous variables that will be used to form the LDA decision rule. In the table 4.5 are the coefficients for this data set.

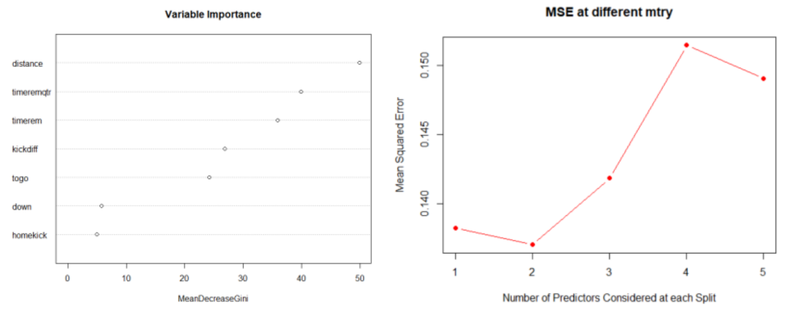
Variables	**togo**	**distance**	**kickdiff**	**timerem**	**timeremqtr**
LDA coeff	2.07e-2	-1.07e-1	-4.42e-3	2.35e-6	5.75e-4

[TABLE 5.c.1 – Initial Model Results]

If 2.07e-2*togo -1.07e-1*distance-4.42e-3*kickdiff+2.35e-6*timerem+5.75e-4*timeremqtr is large then the LDA classifier will predict the field goal is good and if it is small then it will predict a missed field goal. When we look at our confusion matrix, the LDA classifier model has an accuracy of 78%. The sensitivity and specificity is 78% and 70%, respectively. Our LDA does not perform as well as objective 1 models in terms of accuracy but it does perform better in terms of specificity and sensitivity. See Appendix **[CHART 6.d.3]** for the ROC curve.

Random forest is a method for classification that operates by construction a multitude of decision trees. Random Forest attempts to decorrelate trees by random sampling from the predictors equally not dependent correlation weight of predictors.

Random forest was executed with all the variables to identify the importance of the variables, this model was executed with 400 trees, we found these variables distance, timeremqtr, timerem, kickdiff, togo to effectively reduce the Gini impurities for Variable Importance **[CHART 5.c.1]** to predict good and bad field goal response, we also tried with 500, 1000 trees and the variable importance remains the same. Now, random forest was again executed to find the OOB error at different number of variables available for splitting at each tree node (mtry), we found that the lowest mean square oob error to be 0.1394231 at mtry=2 **[CHART 5.c.1]**. ROC curve was ran important variables with 400 trees and probability predictions using the test data to find the best cut point from the ROC curve (Appendix **[CHART 6.d.5]**) and found optimal cut point to be 0.88. With these optimal values random forest model was able to predict good and bad field goal with 69.57% accuracy with optimal specificity (0.74074) and sensitivity (0.68889).



[CHART 5.c.1]

Model Results

Model	AIC	AUC	Accuracy	Specificity	Sensitivity
LDA		0.775	0.7778	0.70370	0.78889
Complex	554.48	0.782	0.79	0.59	0.82
Random Forest		0.747	0.6957	0.74074	0.68889

[TABLE 5.c.2 – Model Results after cut point adjustment]

d. Conclusion

The goal of this project was to use the data from the 2008 NFL season to create a model that predicts if a field goal will be successful or not. For the modeling in objective 1 we did forward, stepwise, and LASSO selection techniques to create three models. In objective 2 the models were more complex and included interaction terms and polynomials. The resulting complex logistic model and the one using the random forest algorithm were no better at prediction than the models from objective 1.

The model with the best overall predictive accuracy was the LASSO model. However the model that we would choose as the best predictive model overall is the LDA model. This model had only an approximate 2.2% decrease in overall accuracy over LASSO, but its combined sensitivity+specificity is approximately 6% greater. In the end based on these different models it appears that the success of a field goal is not reliant on a complex interaction between the variables.

In the future for improving the predictive accuracy of the models the best thing would be to gather more data. Most of the issues that we had during the initial modeling process were directly due to the low number of 0’s (misses) on our response variable. The prediction model was made with the minimal number of variables present in the 2008 data set. To continue to improve the prediction accuracy of our model for future seasons we would look to do exploratory data analysis on additional variables such as stadium type (open or enclosed), weather, turf conditions, and the opposing team percentage of blocking kicks. We believe that adding more features and data would be the best way to improve our model.

6. Appendix

a. R Code

```
#Read in the data
df.orig <- read.csv("data/nfl2008_fga.csv")
names(df.orig)

#Number of rows
length(df.orig$GameDate)

#Gets the number of NA's in the dataset
missing <- colSums(is.na(df.orig))
missing

#Put the data into a new dataframe to clean
df.clean <- df.orig

#DET vs GB information (about the NA)
#https://www.pro-football-reference.com/boxscores/200812280gnb.htm
#NYG vs ARI information (about the NA)
#https://www.pro-football-reference.com/boxscores/200811230crd.htm

#By logic and the websites, down has to be 1 and togo has to be 10
df.clean$down[df.clean$HomeTeam == "ARI" & df.clean$AwayTeam == "NYG" & df.clean$GameDate == 20081123 & df.clean$timerem ==
1811] <- 1
df.clean$togo[df.clean$HomeTeam == "ARI" & df.clean$AwayTeam == "NYG" & df.clean$GameDate == 20081123 & df.clean$timerem == 1811]
<- 10
df.clean$down[df.clean$HomeTeam == "GB" & df.clean$AwayTeam == "DET" & df.clean$GameDate == 20081228 & df.clean$timerem == 1807]
<- 1
df.clean$togo[df.clean$HomeTeam == "GB" & df.clean$AwayTeam == "DET" & df.clean$GameDate == 20081228 & df.clean$timerem == 1807]
<- 10

#Gets the number of NA's in the dataset
colSums(is.na(df.clean))

#Make a factor: qtr, down, GOOD, homekick
df.clean$qtr <- as.factor(df.clean$qtr)
df.clean$down <- as.factor(df.clean$down)
df.clean$GOOD <- as.factor(df.clean$GOOD)
df.clean$homekick <- as.factor(df.clean$homekick)

#Remove Missed and Blocked because they are also part of the response
df.clean$Missed <- NULL
df.clean$Blocked <- NULL

#GameDate won't help for future predictions
#Name and Kicker won't help for future predictions
#season is always 2008
df.clean$GameDate <- NULL
df.clean$name <- NULL
df.clean$kicker <- NULL
df.clean$season <- NULL

#We don't need AwayTeam or HomeTeam
df.clean$HomeTeam <- NULL
df.clean$AwayTeam <- NULL

#We don't care who the kick team is, just
#if the team is at home or away (homekick)
df.clean$kickteam <- NULL
df.clean$def <- NULL

pairs(df.clean, gap=0)

#Outliers visible in ydline vs distance
plot(df.clean$ydline, df.clean$distance)
#ydline is wrong on these
df.clean[df.clean$ydline > 50,]
#This is no problem because distance is fine and we are removing ydline
#ydline and distance are equivelent
df.clean$ydline <- NULL

#----------RUN CONTINUOUS CORRELATION SECTION AND COME BACK HERE

#timerem is correlated with qtr,min,sec
#Creating a new column "Time Remaining in Quarter" (timeremqtr)
df.clean$timeremqtr <- (df.clean$timerem - (4 - as.numeric(df.clean$qtr))*15*60)
#Don't need min, sec: they are redundant
df.clean$min <- NULL
df.clean$sec <- NULL

#kickdiff is correlated with offscore and defscore
#We will just keep the difference in score
df.clean$offscore <- NULL
df.clean$defscore <- NULL

xtabs(~qtr + down, data = df.clean)    #PROBLEM HERE, several 0's
xtabs(~qtr + homekick, data = df.clean)  #PROBLEM HERE qtr 5, few values
xtabs(~qtr + GOOD, data = df.clean)    #PROBLEM HERE qtr 5, few values
xtabs(~down + homekick, data = df.clean)
xtabs(~down + GOOD, data = df.clean)
xtabs(~homekick + GOOD, data = df.clean)

#Removing qtr because it is problematic, it does
#not have enough of some values, see xtabs above
df.clean$qtr <- NULL

pairs(df.clean, gap=0)

#-----------CORRELATION OF CONTINUOUS---------------------------------------------
library(corrplot)
df.clean.numeric <- df.clean[, sapply(df.clean, is.numeric)]
#Correlations of all numeric variables
df.clean.allcor <- cor(df.clean.numeric, use="pairwise.complete.obs")
#The cutoff point for correlation, currently randomly assigned
corr_amt <- 0.3

#Finds rows [A,B,123], [B,A,123] and removes them (duplicates)
rmCorDup <- function(res)
{
 rmrow <- numeric()
 len <- length(res$Var1)
 for( i in c(1:(len-1)) )
 {
  num <- i+1
  for(j in c(num:len))
  {
   if( (res[i,1] == res[j,2]) & (res[i,2] == res[j,1]) )
   {
    rmrow <- c(rmrow, j)
   }
  }
 }
 res <- res[-rmrow,]
 res
}

#Gets a list of all correlations with higher than 'corr_amt' of correlation
df.clean.highcor <- rmCorDup(subset(as.data.frame(as.table(df.clean.allcor)), (abs(Freq) > corr_amt) & (abs(Freq) < 1)))
df.clean.highcor
#Vector of the names of the columns with high correlation
df.clean.highcor.names <- unique( c(as.vector(df.clean.highcor$Var1), as.vector(df.clean.highcor$Var2)) )

#Creates a matrix of high correlation for the graphic
df.clean.highcor.matrix <- df.clean.allcor[df.clean.highcor.names, df.clean.highcor.names]
#Creates the high correlation graphic
corrplot.mixed(df.clean.highcor.matrix, tl.col="black", tl.pos = "lt")

#------CORRELATION OF MIXED------------------------------------------------
library(ggplot2)

df.clean.num1 <- df.clean[, sapply(df.clean, is.numeric)]
df.clean.cat1 <- df.clean[, sapply(df.clean, is.factor)]

for(i in c(1:ncol(df.clean.num1)) )
{
 for(j in c(1:ncol(df.clean.cat1)) )
 {
  plot <- ggplot(NULL, aes(df.clean.cat1[,j], df.clean.num1[,i])) +
   geom_boxplot() +
   xlab(names(df.clean.cat1)[j]) +
   ylab(names(df.clean.num1)[i])
  print(plot)
 }
}

#Most questionable ones from visual inspection
#timeremqtr, down
plot <- ggplot(df.clean, aes(down, timeremqtr)) +
 geom_boxplot() +
 xlab("down") +
 ylab("timeremqtr")
print(plot)
#Kruskal-Wallis ANOVA
fit <- kruskal.test(timeremqtr~down, data = df.clean)
fit

#timerem, down
plot <- ggplot(df.clean, aes(down, timerem)) +
 geom_boxplot() +
 xlab("down") +
 ylab("timeremqtr")
print(plot)
#Kruskal-Wallis ANOVA
fit <- kruskal.test(timerem~down, data = df.clean)
fit

#down, togo
plot <- ggplot(df.clean, aes(down, togo)) +
 geom_boxplot() +
 xlab("down") +
 ylab("timeremqtr")
print(plot)
```

```
fit <- kruskal.test(togo~down, data = df.clean)
fit

#Each of these ANOVAs indicate that the values are highly correlated
#(all p-values <3.637e-07)

#Because each of these are highly associated with 'down' we remove down
df.clean$down <- NULL

#------CORRELATION OF CATEGORICAL------------------------------------------
#MANTEL-HAENSZEL?

#pairs plot with GOOD colored into it
pairs(df.clean[,c(1:5, 7)], col = (as.numeric(df.clean$GOOD)+1) )


#------------ROC CURVE FUNCTION (EX in obj 2)----------------------
#Function that prints the ROC curve
#and returns the optimal cut value
printcutroc = function(thefit, df.pred, y)
{
 library(ROCR)
 pred <- prediction(df.pred, y)
 roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")
 auc.train <- performance(pred, measure = "auc")
 auc.train <- auc.train@y.values

 #Plot ROC
 plot(roc.perf)
 abline(a=0, b= 1) #Ref line indicating poor performance
 text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))

 #Function to get the optimal place to cut (instead of 0.5)
 opt.cut = function(perf, pred)
 {
  cut.ind = mapply(FUN=function(x, y, p)
  {
   d = (x - 0)^2 + (y-1)^2
   ind = which(d == min(d))
   c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
    cutoff = p[[ind]])
  }, perf@x.values, perf@y.values, pred@cutoffs)
 }

 #Returns the cut
 opt.cut(roc.perf, pred)[3,]
}
#============================OBJECTIVE 1====================================

library(caret)
library(car)

set.seed(123)
df.train.index <- createDataPartition(df.clean$GOOD, p = .8, list = FALSE)
df.train <- df.clean[df.train.index,]
df.test <- df.clean[-df.train.index,]

fit.full <- glm(GOOD ~togo + distance + homekick + kickdiff + timerem + timeremqtr +
        togo*distance + togo*homekick + togo*kickdiff + togo*timerem + togo*timeremqtr +
        distance*homekick + distance*kickdiff + distance*timerem + distance*timeremqtr +
        homekick*kickdiff + homekick*timerem + homekick*timeremqtr +
        kickdiff*timerem + kickdiff*timeremqtr +
        timerem*timeremqtr +
        togo*togo + distance*distance + homekick*homekick + kickdiff*kickdiff +
        timerem*timerem + timeremqtr*timeremqtr, data = df.train, family = binomial)
fit.min <- glm(GOOD ~1, data = df.train, family = binomial)
```

```
df.train.stepwise.model <- stepAIC(fit.min, direction = "both",
                scope=list(upper=fit.full, lower=fit.min), trace = FALSE)

#probabilities <- df.train.stepwise.model %>% predict(df.test, type = "response")
#predicted.classes <- ifelse(probabilities > 0.5, 1, 0)
# Model accuracy
#mean(predicted.classes==df.test$GOOD)

# Does the ROC curve plotting, gets the optimal cut point,
# cuts the prediction to match, and prints out the confusion matrix
df.train.step.pred <- predict(df.train.stepwise.model, df.train, type = "response")
df.step.cut <- printcutroc(df.train.stepwise.model, df.train.step.pred, df.train$GOOD) #custom function from cleaning file
df.train.step.pred <- as.factor(ifelse(df.train.step.pred >= df.step.cut, 1, 0))
confusionMatrix(df.train.step.pred, df.train$GOOD) #confusion matrix of the training

# Making prediction and confusion matrix on test data using df.cut
df.test.step.pred <- predict(df.train.stepwise.model, df.test, type = "response")
#printcutroc(df.train.stepwise.model, df.test.step.pred, df.test$GOOD) #custom function from cleaning file
df.test.step.pred <- as.factor(ifelse(df.test.step.pred >= df.step.cut, 1, 0))
confusionMatrix(df.test.step.pred, df.test$GOOD) #confusion matrix of the test data

summary(df.train.stepwise.model)

#### --------------- Libararies used ------------------
library(tidyverse)
library(caret)
library(glmnet)
library(ggplot2)
library(broom)
library(car)
library(corrplot)
library(pscl)
#library(pROC)
library(ROCR)
library(MASS)
library(survey)
library(selectiveInference)
library(factoextra)


#### ---------------SEED for reproducibility----------------
set.seed(123)

#####------------------splitting the data into train and test------------------------------
# Data is plit to 80/20 % by GOOD values into Train and Test.
df.train.index <- createDataPartition(df.clean$GOOD, p = .8, list = FALSE)
df.train <- df.clean[df.train.index,]
df.test <- df.clean[-df.train.index,]

#equal percentage of data split for train and test set.
xtabs(~GOOD+GOOD, data = df.train)
xtabs(~GOOD+GOOD, data = df.test)


# Train and test are equally split (randamizing at the GOOD feature level)
plot(df.train$GOOD)
plot(df.test$GOOD)

#####------------------Logistic regression diagnostics & Assumptions ---------------------------------

##------Building Model for Diagnostics-------
model <- glm(GOOD~., data = df.train, family = binomial)

#getting the predicted probablities of each observation.
probabilities <- predict(model, type = "response")
```

```
#assigning the pridiction classes at 50% probabilities to GOOD as 1's and 0's
predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")
#View(predicted.classes)

### ----- Linearity assumption-----

# Select only numeric predictors
#we will check the linear relationship between continuous predictor variables and the logit of the outcome.
#we will be visually inspecting the scatter plot between each predictor and the logit values.

diagData <- df.train %>% select_if(is.numeric)
pred <- colnames(diagData)

diagData <- diagData %>% mutate(logit = log(probabilities/(1-probabilities))) %>%
 gather(key = "predictors", value = "predictor.value", -logit)

ggplot(diagData, aes(logit, predictor.value))+
 geom_point(size = 0.5, alpha = 0.5) +
 geom_smooth(method = "loess") +
 theme_bw() +
 facet_wrap(~predictors, scales = "free_y")


### ----- Influential values-----

plot(model, which = 4, id.n = 3)

model.data <- augment(model) %>%
 mutate(index = 1:n())
model.data %>% top_n(3, .cooksd)
View(model.data %>% top_n(4, .cooksd))

# Visualizing the data with the standardized residuals
ggplot(model.data, aes(index, .std.resid)) +
 geom_point(aes(color = GOOD), alpha = .5) +
 theme_bw()

# removing the influential data points from training dataset
#df.train <- df.train[-c(118,212,615,661),] # not required for Zac's dataset
# repeat the process to see if the changes on removing the influential points

### ----- Multicollinearity --------------

vif(model)
#As a rule of thumb, a VIF value that exceeds 5 or 10 indicates a problematic amount of collinearity.
#In our data there is no collinearity: all variables have a value of VIF well below 5.

#### ---------------- Penalized Logistic Regression using LASSO Model regularization ----------------------------


# Additionnal data preparation
#training data
df.train.x <- model.matrix(GOOD~., data = df.train)[,-1] #remove the intercept data
df.train.y <- df.train$GOOD
#testing data
df.test.x <- model.matrix(GOOD~., data = df.test)[,-1] #remove the intercept data
df.test.y <- df.test$GOOD

#Identifying the best lambda value to peanilize the model for regularization using cross validation
cv.lasso <- cv.glmnet(df.train.x, df.train.y , alpha = 1, family = "binomial")
plot(cv.lasso)
bestlambda <- cv.lasso$lambda.1se

coef(cv.lasso, cv.lasso$lambda.1se)
coef(cv.lasso, cv.lasso$lambda.min)
# based on the below values using the best lambda (lambda.1se) minimizes the complexity of the model to have only one feature (distance)
```

```r
# using next best lambda (lambda.min) removes two features (Kickdiff and timerem).
# Hence we will be using lambda.min to avoid high biasing.

##### -------- Computing final LASSO Model using min lambda ----------------

df.train.lasso.model <- glmnet(df.train.x, df.train.y , alpha = 1, family = "binomial", lambda = cv.lasso$lambda.min)

# Making prediction on test data
probabilities <- df.train.lasso.model %>% predict(newx = df.test.x)
predicted.classes <- ifelse(probabilities > 0.5, 1, 0)

# Model accuracy
mean(predicted.classes == df.test.y)

#Gets the coefficients of the lasso model
coef(df.train.lasso.model)

#---------------------------ADDED BY ZACK----------------------------
# Confusion matrix before the change in cut location
confusionMatrix(as.factor(as.vector(predicted.classes)), df.test$GOOD)

# Does the ROC curve plotting, gets the optimal cut point,
# cuts the prediction to match, and prints out the confusion matrix
df.train.lasso.pred <- predict(df.train.lasso.model, newx = df.train.x, type = "response")
df.lasso.cut <- printcutroc(df.train.lasso.model, df.train.lasso.pred, df.train$GOOD) #custom function from cleaning file
df.train.lasso.pred <- as.factor(ifelse(df.train.lasso.pred >= df.lasso.cut, 1, 0))
confusionMatrix(df.train.lasso.pred, df.train$GOOD) #confusion matrix of the training

#Prediction on the test set using the original cut point of 0.5
df.test.lasso.pred <- predict(df.train.lasso.model, newx = df.test.x, type = "response")
printcutroc(df.train.lasso.model, df.test.lasso.pred, df.test$GOOD) #ROC Curve
df.test.lasso.pred <- as.factor(ifelse(df.test.lasso.pred >= 0.5, 1, 0))
confusionMatrix(df.test.lasso.pred, df.test$GOOD) #confusion matrix of the test data on 0.5 cut point

# Making prediction and confusion matrix on test data using df.cut
df.test.lasso.pred <- predict(df.train.lasso.model, newx = df.test.x, type = "response")
df.test.lasso.pred <- as.factor(ifelse(df.test.lasso.pred >= df.lasso.cut, 1, 0))
confusionMatrix(df.test.lasso.pred, df.test$GOOD) #confusion matrix of the test data
#---------------------------END ADDED BY ZACK------------------------

####-------------CONFIDENCE INTERVAL of LASSO -----------------#########
df.train.lasso.model.ci <- glm(GOOD ~ distance + homekick + timeremqtr, data = df.train, family = binomial)
confint(df.train.lasso.model.ci)

df.train.lasso.model.ci <- glmnet(df.train.x, df.train.y ,alpha=1, family = "binomial", lambda = cv.lasso$lambda.min, standardize = FALSE)

lamb <- cv.lasso$lambda.min

beta_hat = coef(df.train.lasso.model.ci, s=(lamb/nrow(df.train.x)), exact=TRUE, x = df.train.x, y = df.train.y)

# apply(df.train.x, 2, as.numeric)
# sapply(df.train.x, as.numeric)
class(df.train.x) <- "numeric"
storage.mode(df.train.x) <- "numeric"

# apply(df.train.y, 2, as.numeric)
# sapply(df.train.y, as.numeric)
class(df.train.y) <- "numeric"
storage.mode(df.train.y) <- "numeric"

df.train.y <- ifelse(df.train.y == 2, 1, 0)

out = fixedLassoInf(df.train.x,df.train.y,beta_hat,df.train.lasso.model.ci$lambda,family="binomial")
out

#########----------------Stepwise/Forward Logistic Regression-----------------#####
```

```r
fit.full <- glm(GOOD ~., data = df.train, family = binomial)
fit.min <- glm(GOOD ~1, data = df.train, family = binomial)
df.train.forward.model <- stepAIC(fit.min, direction = "forward",
             scope=list(upper=fit.full, lower=fit.min), trace = FALSE)
df.train.stepwise.model <- stepAIC(fit.min, direction = "both",
              scope=list(upper=fit.full, lower=fit.min), trace = FALSE)


####---- Stepwise----
# Stepwise feature selection removed all the features as non significant and added only distance as a significant feature with model AIC = 571.2
summary(df.train.stepwise.model)

# Make predictions
probabilities <- df.train.stepwise.model %>% predict(df.test, type = "response")
predicted.classes <- ifelse(probabilities > 0.5, 1, 0)
# Model accuracy
mean(predicted.classes==df.test$GOOD)

#--------------------------ADDED BY ZACK---------------------------
# Confusion matrix before the change in cut location
confusionMatrix(as.factor(as.vector(predicted.classes)), df.test$GOOD)

# Does the ROC curve plotting, gets the optimal cut point,
# cuts the prediction to match, and prints out the confusion matrix
df.train.step.pred <- predict(df.train.stepwise.model, df.train, type = "response")
df.step.cut <- printcutroc(df.train.stepwise.model, df.train.step.pred, df.train$GOOD) #custom function from cleaning file
df.train.step.pred <- as.factor(ifelse(df.train.step.pred >= df.step.cut, 1, 0))
confusionMatrix(df.train.step.pred, df.train$GOOD) #confusion matrix of the training

#Making prediction on the test set using the original cut point of 0.5
df.test.step.pred <- predict(df.train.stepwise.model, df.test, type = "response")
printcutroc(df.train.stepwise.model, df.test.step.pred, df.test$GOOD) #custom function from cleaning file
df.test.step.pred <- as.factor(ifelse(df.test.step.pred >= 0.5, 1, 0))
confusionMatrix(df.test.step.pred, df.test$GOOD) #confusion matrix of the test data

# Making prediction and confusion matrix on test data using df.cut
df.test.step.pred <- predict(df.train.stepwise.model, df.test, type = "response")
df.test.step.pred <- as.factor(ifelse(df.test.step.pred >= df.step.cut, 1, 0))
confusionMatrix(df.test.step.pred, df.test$GOOD) #confusion matrix of the test data
#--------------------------END ADDED BY ZACK-----------------------

####---- Forward----
# Forward feature selection removed all the features as non significant and added only distance as a significant feature with model AIC = 577.62

summary(df.train.forward.model)

# Make predictions
probabilities <- df.train.forward.model %>% predict(df.test, type = "response")
predicted.classes <- ifelse(probabilities > 0.5, 1, 0)
# Model accuracy
mean(predicted.classes==df.test$GOOD)

#--------------------------ADDED BY ZACK---------------------------
# Confusion matrix before the change in cut location
confusionMatrix(as.factor(as.vector(predicted.classes)), df.test$GOOD)

# Does the ROC curve plotting, gets the optimal cut point,
# cuts the prediction to match, and prints out the confusion matrix
df.train.forw.pred <- predict(df.train.forward.model, df.train, type = "response")
df.forw.cut <- printcutroc(df.train.forward.model, df.train.forw.pred, df.train$GOOD) #custom function from cleaning file
df.train.forw.pred <- as.factor(ifelse(df.train.forw.pred >= df.forw.cut, 1, 0))
confusionMatrix(df.train.forw.pred, df.train$GOOD) #confusion matrix of the training

# Makeing prediction on test data with original cut of 0.5
df.test.forw.pred <- predict(df.train.forward.model, df.test, type = "response")
```

```
printcutroc(df.train.forward.model, df.test.forw.pred, df.test$GOOD) #Print ROC Curve
df.test.forw.pred <- as.factor(ifelse(df.test.forw.pred >= 0.5, 1, 0))
confusionMatrix(df.test.forw.pred, df.test$GOOD) #confusion matrix of the test data

# Making prediction and confusion matrix on test data using df.cut
df.test.forw.pred <- predict(df.train.forward.model, df.test, type = "response")
df.test.forw.pred <- as.factor(ifelse(df.test.forw.pred >= df.forw.cut, 1, 0))
confusionMatrix(df.test.forw.pred, df.test$GOOD) #confusion matrix of the test data
#--------------------------END ADDED BY ZACK-----------------------

####-----------ROC----------------

# Considering the AUC values of all 3 models LASSO (AUC = 0.8299383), Stepwise (0.5720165), Forward (0.5667695), LASSO is a better
# model compared to stepwise or Forward feature selections.

# For Lasso Model
prob.lasso <- predict(df.train.lasso.model, newx =df.test.x, type="response")
pred.lasso <- prediction(prob.lasso, df.test.y)
perf.lasso <- performance(pred.lasso, measure = "tpr", x.measure = "fpr")

auc <- performance(pred.lasso, measure = "auc")
auc <- auc@y.values[[1]]
plot(perf.lasso, main="LASSO")
abline(a=0, b= 1)
text(x = .40, y = .6,paste("AUC = ", round(auc[[1]],3), sep = ""))
auc

# Forward
#prob.forward <- predict(df.train.forward.model, newx =df.test, type="response")
probabilities.forward <- df.train.forward.model %>% predict(df.test, type = "response")
pred.forward <- prediction(probabilities.forward, df.test.y)
perf.forward <- performance(pred.forward, measure = "tpr", x.measure = "fpr")
plot(perf.forward)

auc <- performance(pred.forward, measure = "auc")
auc <- auc@y.values[[1]]
abline(a=0, b= 1) #Ref line indicating poor performance
text(x = .40, y = .6,paste("AUC = ", round(auc[[1]],3), sep = ""))
auc

# Stepwise
#prob.forward <- predict(df.train.forward.model, newx =df.test, type="response")
probabilities.stepwise <- df.train.stepwise.model %>% predict(df.test, type = "response")
pred.stepwise <- prediction(probabilities.stepwise, df.test.y)
perf.stepwise <- performance(pred.stepwise, measure = "tpr", x.measure = "fpr")
plot(perf.stepwise)

auc <- performance(pred.stepwise, measure = "auc")
auc <- auc@y.values[[1]]
abline(a=0, b= 1) #Ref line indicating poor performance
text(x = .40, y = .6,paste("AUC = ", round(auc[[1]],3), sep = ""))
auc



######################## RANDOM FOREST ####################################


# we will use the same df.train and df.test

library(randomForest)

df.train.rf.model <- randomForest(GOOD~., data = df.train)

print(df.train.rf.model)
```

```
df.test.rf.pred <- predict(df.train.rf.model, type = "prob", newdata = df.test)
rf.pred <- prediction(df.test.rf.pred[,2], df.test$GOOD)
rf.perf = performance(rf.pred, measure = "tpr", x.measure = "fpr")

auc <- performance(rf.pred, measure = "auc")
auc <- auc@y.values[[1]]
auc

rf.important <- importance(df.train.rf.model)

varImpPlot(df.train.rf.model)

# looking at the importance plot distance, timeremqtr,timerem looks important, hence we wiil build model with these 3 predictors
########### ----------Updated-------- ##########
df.train.rf.model.updated <- randomForest(GOOD~distance+timeremqtr+timerem+kickdiff, data = df.train)

df.test.rf.pred.updated <- predict(df.train.rf.model.updated, type = "prob", newdata = df.test)
rf.pred.updated <- prediction(df.test.rf.pred.updated[,2], df.test$GOOD)
rf.perf.updated = performance(rf.pred.updated, measure = "tpr", x.measure = "fpr")

auc <- performance(rf.pred.updated, measure = "auc")
auc <- auc@y.values[[1]]
auc

#ROC Plots for just LASSO, Forward, and Stepwise
plot(perf.lasso, col=1)
plot(perf.forward,col=2, add=TRUE)
plot(perf.stepwise,col=3, add=TRUE)
legend(0.7, 0.8, legend = c("LASSO", "Forward", "Stepwise"), 1:3)

summary(df.clean)
table(df.clean$GOOD)
pairs(df.clean[,-6],col=df.clean$GOOD)
df.cleana<-df.clean[,-6]
df.cleanb<-df.cleana[,-3]
pairs(df.clean)
summary(df.cleanb)
var.raw<-apply(df.cleanb, 2,var)
sum(var.raw)
#[1] 1079767
sum(diag(cov(df.cleanb)))
#[1] 1079767
pc.results<-prcomp(df.cleanb, scale. = TRUE)
pc.scores<-pc.results$x
pairs(pc.scores)
cor(pc.scores)

pc.results$rotation

eigenvals<-(pc.results$sdev)^2
plot(1:5,eigenvals/sum(eigenvals),type="l",main="Scree Plot",ylab="Prop. Var. Explained")
cumulative.prop<-cumsum(eigenvals/sum(eigenvals))
plot(1:5,cumulative.prop,type="l",main="Cumulative proportion",ylim=c(0,1))
# looks like 4 variables princlple components explain 80% of the variance
##Creating data frame for PC with response added back in
pc.scoresdf <- data.frame(pc.scores)
pc.scoresdf$GOOD1 <- df.clean$GOOD

pairs(pc.scoresdf, col=pc.scoresdf$GOOD1)

##Graphing of PC vs PC
ggplot(data = pc.scoresdf, aes(x = PC1, y = PC2)) +
 geom_point(aes(col=GOOD1), size=1)+
 ggtitle("PCA of GOOD PC1vsPC2")

ggplot(data = pc.scoresdf, aes(x = PC2, y = PC3)) +
 geom_point(aes(col=GOOD1), size=1)+
 ggtitle("PCA of GOOD PC2vsPC3")

ggplot(data = pc.scoresdf, aes(x = PC3, y = PC4)) +
 geom_point(aes(col=GOOD1), size=1)+
 ggtitle("PCA of GOOD PC3vsPC4")

ggplot(data = pc.scoresdf, aes(x = PC4, y = PC5)) +
 geom_point(aes(col=GOOD1), size=1)+
 ggtitle("PCA of GOOD PC4vsPC5")


###Perform LDA
mylda <- lda(GOOD ~ ., data = df.train[,-3])
pred <- predict(mylda,newdata=df.test)$class
x <- table(pred,df.test$GOOD) # Creating a confusion matrix
ME <- (x[2,1]+x[1,2])/1000 #Missclassification Error
ME
1-ME #Calculating overall accuracy
confusionMatrix(table(pred,df.test$GOOD))


##-----NEED HELP------------Changing of cutpoints
# Does the ROC curve plotting, gets the optimal cut point,
# cuts the prediction to match, and prints out the confusion matrix
df.train.lda.pred <- predict(mylda, df.train, type = "response")$posterior[,2]
df.lda.cut <- printcutroc(mylda, df.train.lda.pred, df.train$GOOD) #custom function from cleaning file
df.train.lda.pred <- as.factor(ifelse(df.train.lda.pred >= df.lda.cut, 1, 0))
confusionMatrix(df.train.lda.pred, df.train$GOOD) #confusion matrix of the training

#Making prediction on the test set using the original cut point of 0.5
df.test.lda.pred <- predict(mylda, df.test, type = "response")$posterior[,2]
printcutroc(df.train.lda.model, df.test.lda.pred, df.test$GOOD) #custom function from cleaning file
df.test.lda.pred <- as.factor(ifelse(df.test.lda.pred >= 0.5, 1, 0))
confusionMatrix(df.test.lda.pred, df.test$GOOD) #confusion matrix of the test data

# Making prediction and confusion matrix on test data using df.cut
df.test.lda.pred <- predict(mylda, df.test, type = "response")$posterior[,2]
df.test.lda.pred <- as.factor(ifelse(df.test.lda.pred >= df.step.cut, 1, 0))
confusionMatrix(df.test.lda.pred, df.test$GOOD) #confusion matrix of the test data
```

b. Original Variables

GameDate
AwayTeam
HomeTeam
qtr (quarter, 5=overtime)
min (minutes remaining)
sec (seconds remaining, added to minutes)
kickteam (Team kicking field goal)
def   (Defending Team)
down
togo   (Yards to go for 1st down)
kicker   (ID #)
ydline   (yardline of kicking team)
name   (kicker's name)
distance (yards)
homekick (1 if kicker at Home, 0 if Away)
kickdiff (Kicking team lead +, or deficit -, prior to kick)
timerem (Time remaining in seconds, negative ==> overtime)

offscore (kicking team's score prior to kick)
defscore (defense " " " " ")
season (2008)
GOOD (1 if Success, 0 if Miss)
Missed (Missed, not blocked =1, 0 ow)
Blocked (1 if Blocked, 0 ow)

c. Missing Data

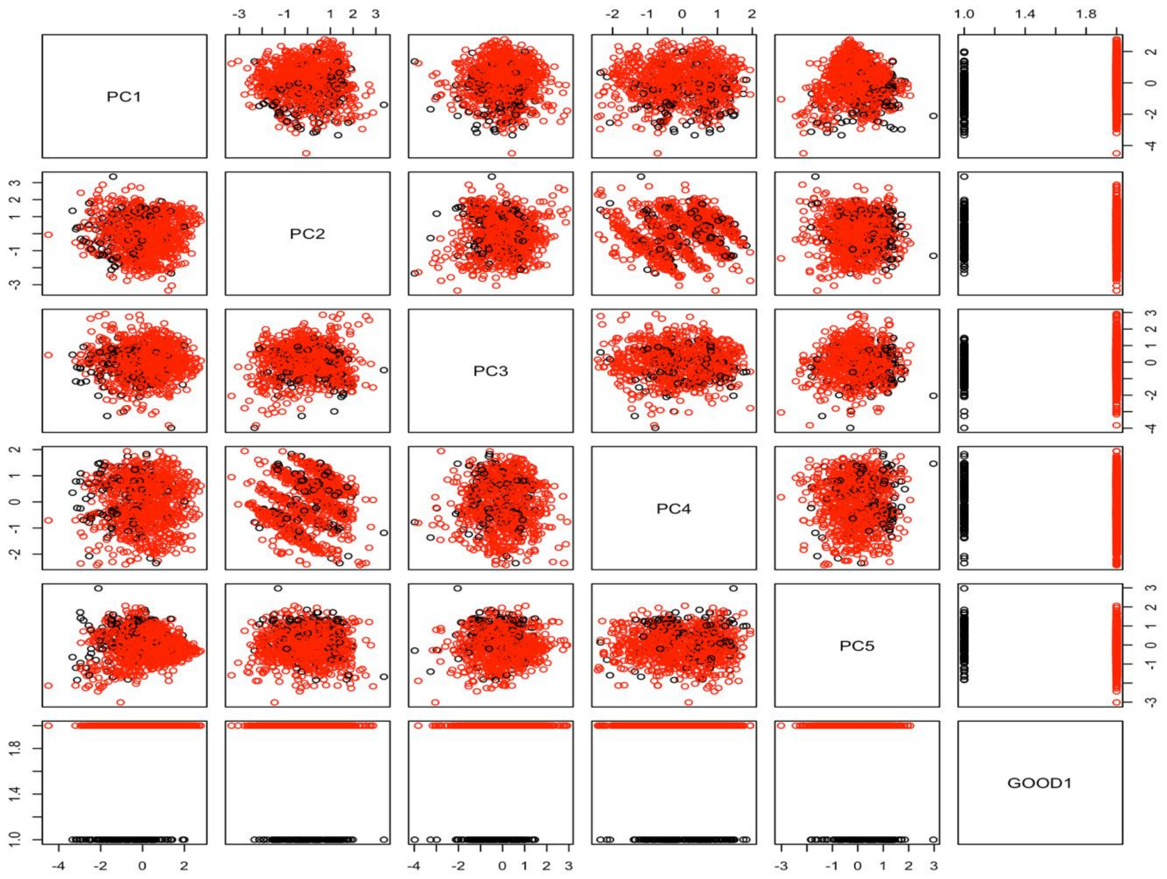
Detroit vs Green Bay: https://www.pro-football-reference.com/boxscores/200812280gnb.htm
New York Giants vs Arizona: https://www.pro-football-reference.com/boxscores/200811230crd.htm

d. Charts

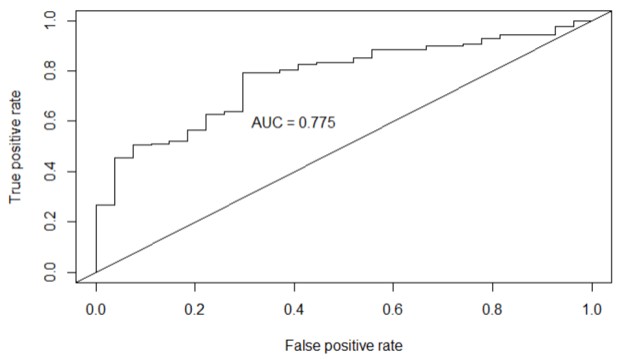
VIF Table

Variable	To go	Distance	Homekick	Kickdiff	Timerem	timeremqtr
VIF	1.08	1.05	1.05	1.07	1.03	1.05

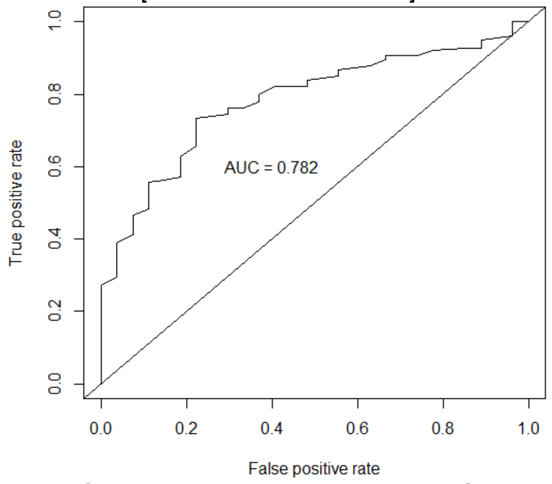
[CHART 6.d.1]



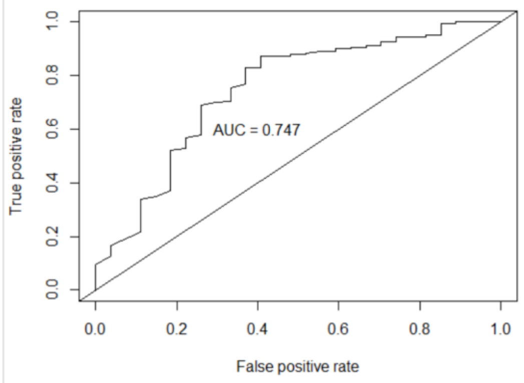
[CHART 6.d.2]



[CHART 6.d.3 – LDA ROC curve]



[CHART 6.d.4 – Complex Logistic ROC curve]



[CHART 6.d.5 – Random Forest ROC curve]