Introduction

For this effort we will be examining a boxing data set that includes data on boxers competing in a boxing match and will attempt to predict the outcomes of the match

Data description

The dataset can be found at <https://www.kaggle.com/slonsky/boxing-bouts/home> and contains the following data. Because we are tying to predict the outcomes of the match before it commences, we will be eliminating decision related data from judges as there is not a way to tie historical data to fighters within this dataset.

NEED SOME FORMATTING HERE

* age, years
* height, cm
* reach, cm - reach from one hand to another
* stance, orthodox/southpaw
* weight, lbs
* won - number of past wins
* lost- number of past loses
* drawn - number of past draws
* kos - number of wins by knockout
* result - bout result - win\_A/win\_B/draw
* decision - type of judge decision. Possible values:
  + SD - splitted decision - Two judges have scored in favour of one boxer and the other judge has scored in favour of the other
  + MD - majority decision - Two judges have scored in favour of one boxer and the other judge has scored in favour of a draw
  + UD - unanimous decision - All three judges have scored in agreement
  + KO - knock out - A boxer is knocked down and the referee has counted to 10 before he can rise
  + TKO - technical knock out - The referee has stopped the fight due to a boxer being in no fit condition to continue
  + DQ - A boxer is disqualified by the referee and loses the bout when he repeatedly or severely fouls or infringes the rules
  + RTD - A boxer has retired between rounds
* judge[1,2,3] - judges scores for certain boxer (aka scorecards)

Logistic Regression Model Selection

Using trial and error (Forward, Backward, Stepwise, Variable selection the best selection criteria were identified using stepwise:

**PROC** **logistic** data= boxing;

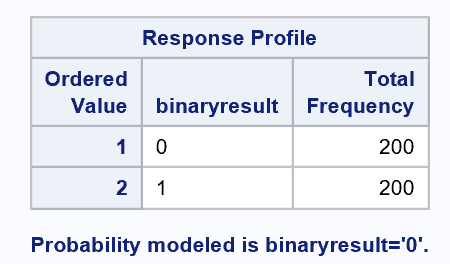
class Stance Over35AgeA Over35AgeB Over15lbA Over15lbB;

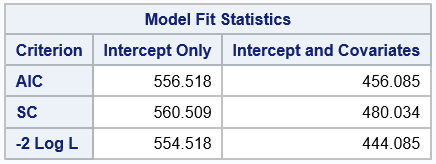
model binaryresult = age\_A age\_B height\_A height\_B reach\_A reach\_B weight\_A weight\_B won\_A won\_B lost\_A lost\_B kos\_A kos\_B AdvAgeA AdvHeightA AdvReachA AdvWgtA WinPA WinPB KoAPer KoBPer

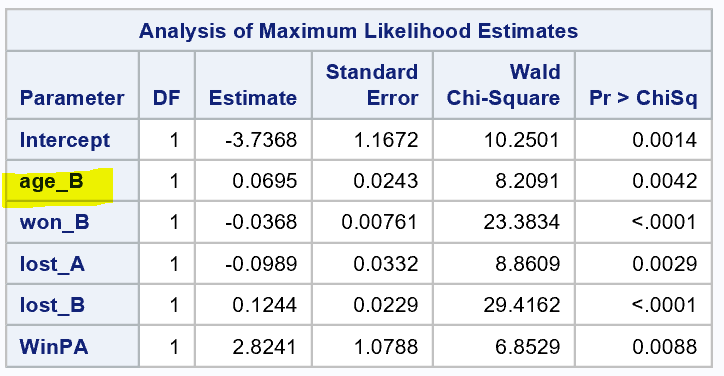
/ selection = stepwise;

output out=boxinglogregout predprobs=I p=probpreb;

**run**;







Goodness of Fit

Though the model that stepwise selection chose with lowest AIC and BIC fits, it doesn’t quite make sense that the stepwise selection includes age\_B but excludes age\_A. These variables have equal meaning and weight. If we simply swap a and b from left to right the result would be different for no logical reason.

This could be indicating that there is indeed a difference in the meaning of the fighter\_A and fighter\_B slots (i.e. challenger versus incumbent). However, we have no context or documentation to confirm this. It is more likely that there is interaction between the variables.

Trial and error shows that adding age\_A does not help. Removing age\_B gives us statistically significant coefficients but reduces fit.

Creating a new variable “AdvAgeA” as Age\_A – Age\_B accounts for both variables an allows us to model without an interaction term. Interestingly it actually has slightly better fit statistics than the interaction term. The fit statistics aren’t quite as good as what stepwise selection gave us but the model makes more sense.

|  |  |
| --- | --- |
| **Include age\_A and age\_B:** | **Exclude age\_A and age\_B:** |
|  |  |
|  |  |
| **Interaction age\_A\*age\_B** | **Cheat to represent interaction** AdvAgeA= Age\_A–Age\_B |
|  |  |
|  |  |

The Chosen Model

Common sense tells us that there is almost certainly interaction between age\_A and age\_B; however, **for the purpose of part 1 in this analysis we will not include an interaction term as instructed.**

It is also noted that stepwise selection included Won\_B but not Won\_A which would raise similar concerns as with age\_A and age\_B. This could be more evidence that there is interaction or there may be meaning to the slots A or B or. WinPA (percentage of prior wins for fighter A) was selected and probably accounts for this.

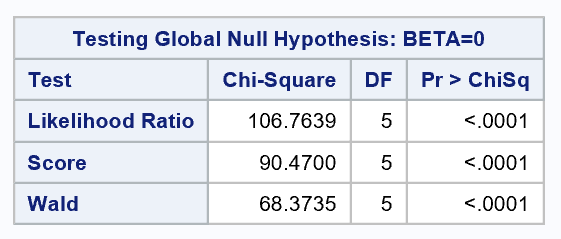
**PROC** **logistic** data= boxing;

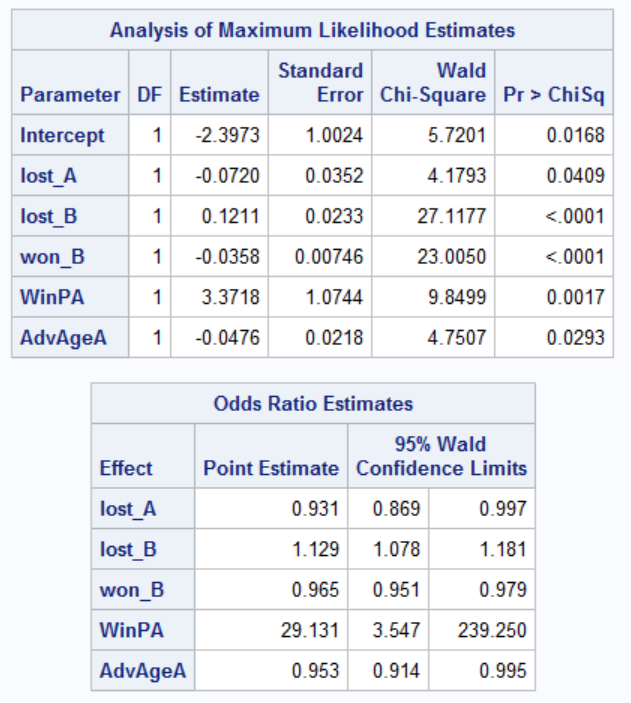
model binaryresult = lost\_A lost\_B won\_B WinPA AdvAgeA /LACKFIT CTABLE;

/\*output out=boxinglogregout predprobs=I p=probpreb;\*/

**run**;

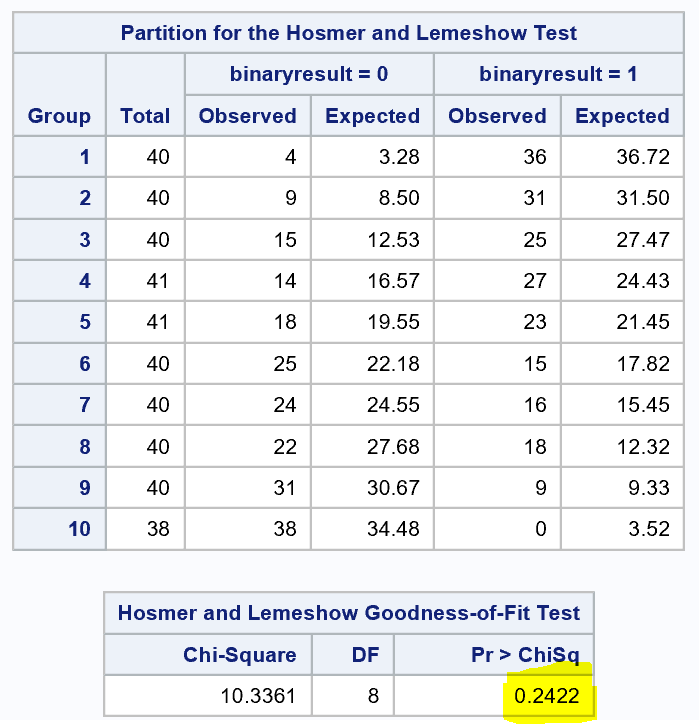
We can reject the null hypotheses that BETA=0. Our variables are statistically significant in predicting 0,1 Fighter A Wins versus Fighter B Wins





|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Actual | |
|  |  | Fighter A Wins (0) | Fighter B Wins (1) |
| Predicted | Fighter A Wins (0) | 144 | 56 |
| Fighter B Wins (1) | 63 | 137 |

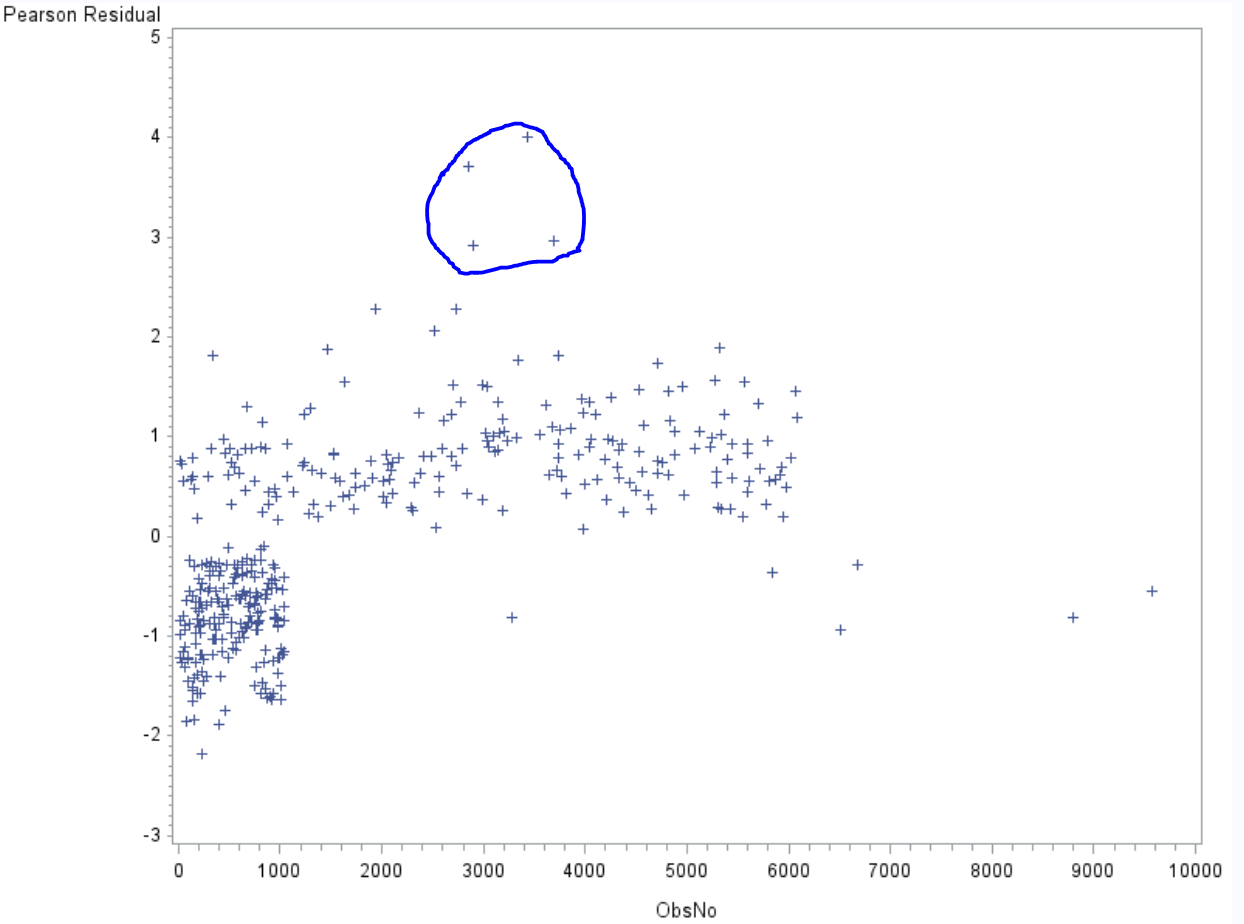
Using .5 accuracy because the negative impact of falsely predicting is equal. These are boxing matches, If this were cancer and not cancer and if treatment will have negative side effects then we would want to consider higher than .5

Use Hosmer Lemeshow because many continues variables.

Do not reject the null hypothesis p-value .2422; the model is a good fit.

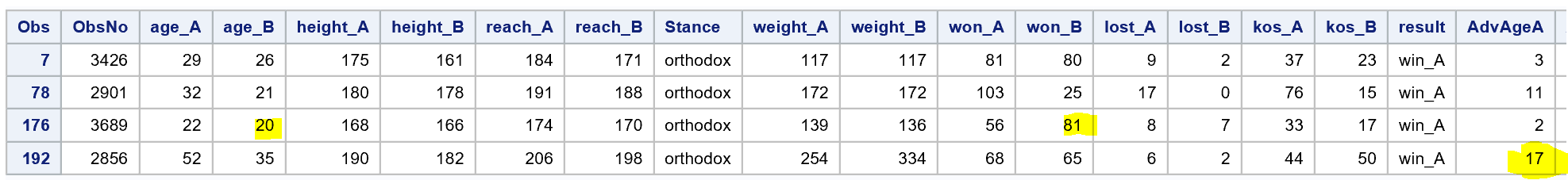
Residual Diagnostics

Something strange about these observations

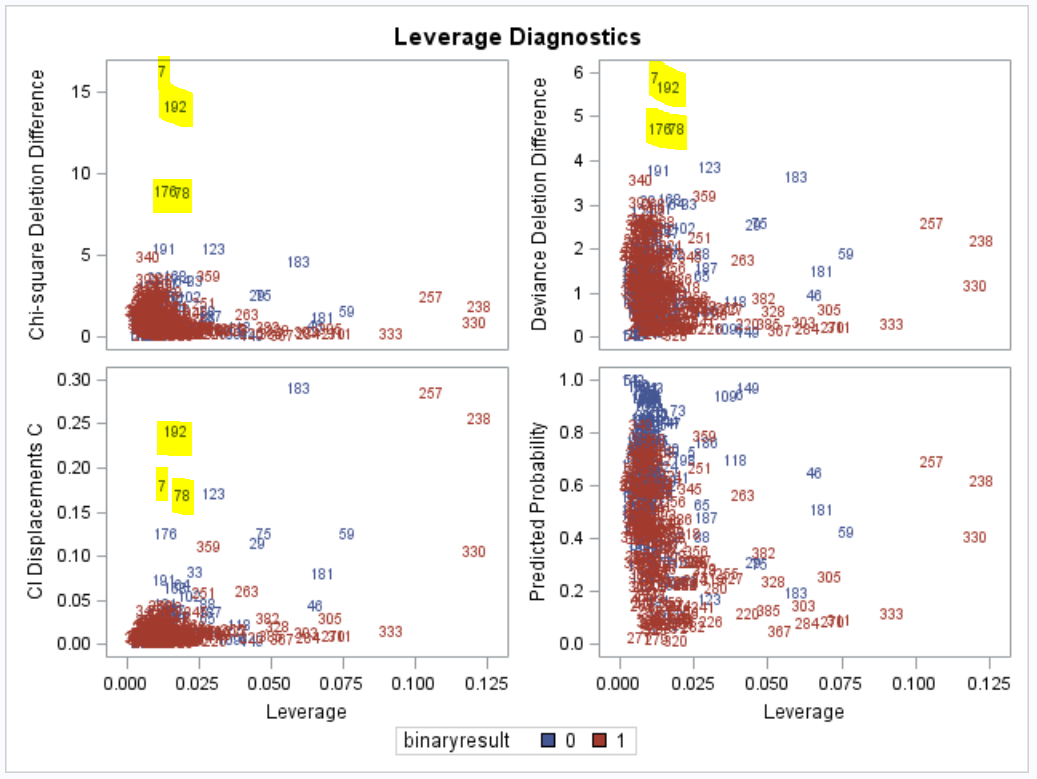


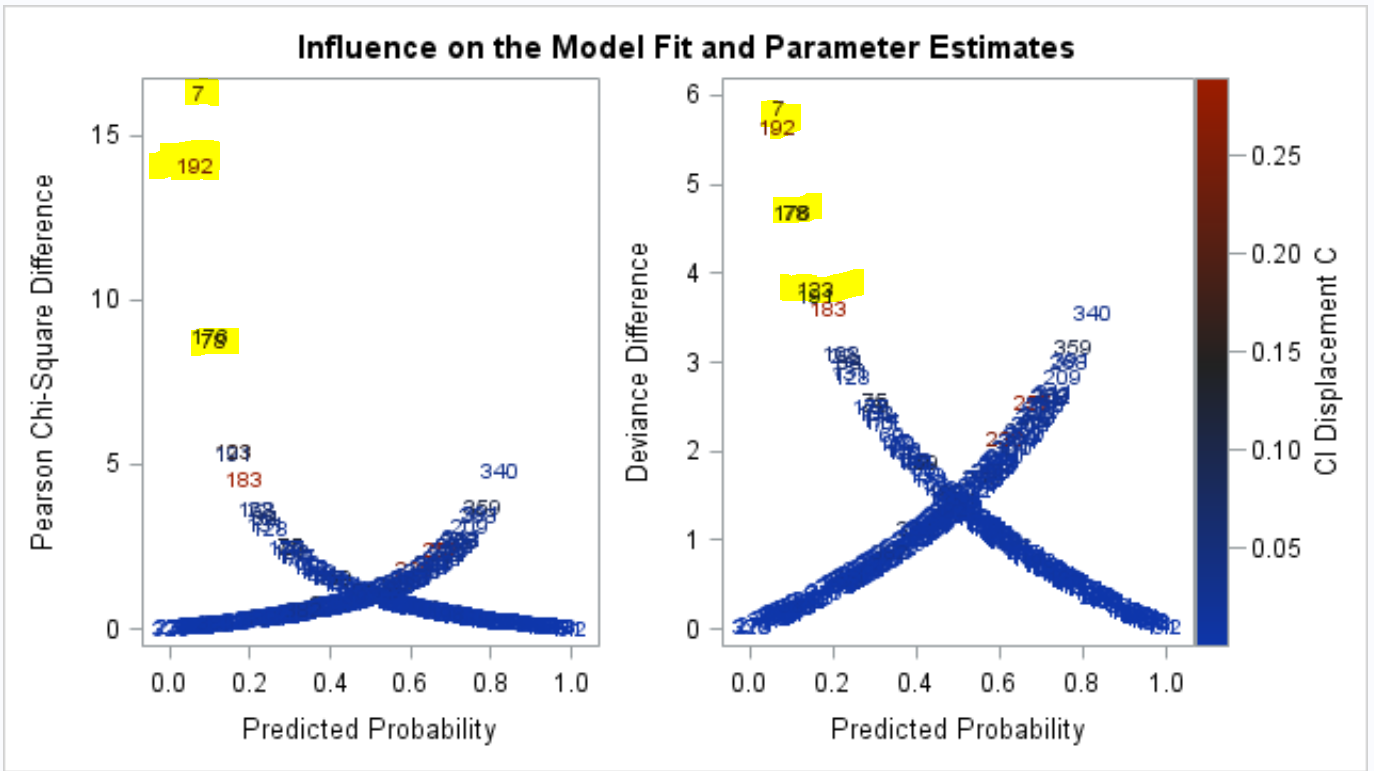
There are suspicious outliers;

* Observation No. 2856 age\_A=52 vs age\_B=35 seems extreme.
* Observation No. 3689 age\_B=20 and has had 88 fights total seems extreme given age.



Observation No. 7, 78, 176, 192 are standing out:





Try model without them and see if stats improve:

|  |  |
| --- | --- |
| With Outliers | Without Outliers |
|  |  |

Final Model

LDA Model

To compete with our logistic regression, a LDA model was developed with all continuous variables resulting in the following LD coeficients

> lda.fit<-lda(binaryresult ~ ., data=dfldatrain)

> lda.fit

Coefficients of linear discriminants:

LD1

age\_A 4.143902e-02

age\_B -9.640704e-02

height\_A -1.089003e-02

height\_B 3.356029e-02

reach\_A -2.122836e-02

reach\_B 4.180252e-03

weight\_A -6.058746e-05

weight\_B 3.002786e-03

won\_A -9.309674e-03

won\_B 3.156529e-02

lost\_A 7.708957e-02

lost\_B -8.440838e-02

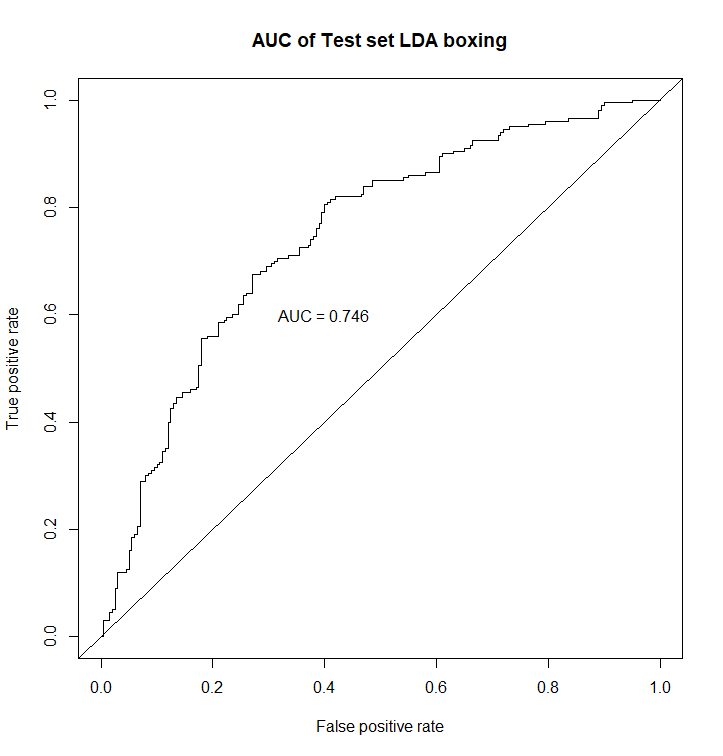
kos\_A -3.180615e-03

kos\_B -1.223608e-02

resulting in the following confusion matrix showing similar performance levels

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Actual (random forest CM) | |
|  |  | Fighter A Wins (0) | Fighter B Wins (1) |
| Predicted | Fighter A Wins (0) | 141 | 63 |
| Fighter B Wins (1) | 59 | 137 |

Examing the ROC curve shows that we are performing better than random predicitions, but are defiinitely not predicting with extremely high precision.



Random Forest

To compete with our logistic regression model, a random forest model was developed for our boxing data set.

To help improve our “mtry” settings of variable sampling, the “tuneRF” function was utilized to test different model performance at different levels

> bestmtry<-tuneRF(x = dftrain[,-27],y=dftrain[,27],stepFactor=1.5, ntree=400)

mtry = 5 OOB error = 30%

Searching left ...

mtry = 4 OOB error = 29.69%

0.01041667 0.05

Searching right ...

mtry = 7 OOB error = 30.56%

-0.01875 0.05

> print(bestmtry)

mtry OOBError

4.OOB 4 0.296875

5.OOB 5 0.300000

7.OOB 7 0.305625

This returned a lowest OOB eeror value for 4 mtry’s. multiple run’s were conducted with little variation so 4 mtry’s were utilized. Multiple iterations were run at different ntree levels as well but little variation was seen beyond 400 ntree’s, which may be asymptomatic of the small effective data size after cleaning the dataset.

> rf.box <- randomForest(x = dftrain[,-27],y = dftrain[,27],mtry=4,ntree = 400,importance = T)

> rf.box

Call:

randomForest(x = dftrain[, -27], y = dftrain[, 27], ntree = 400, mtry = 4, importance = T)

Type of random forest: classification

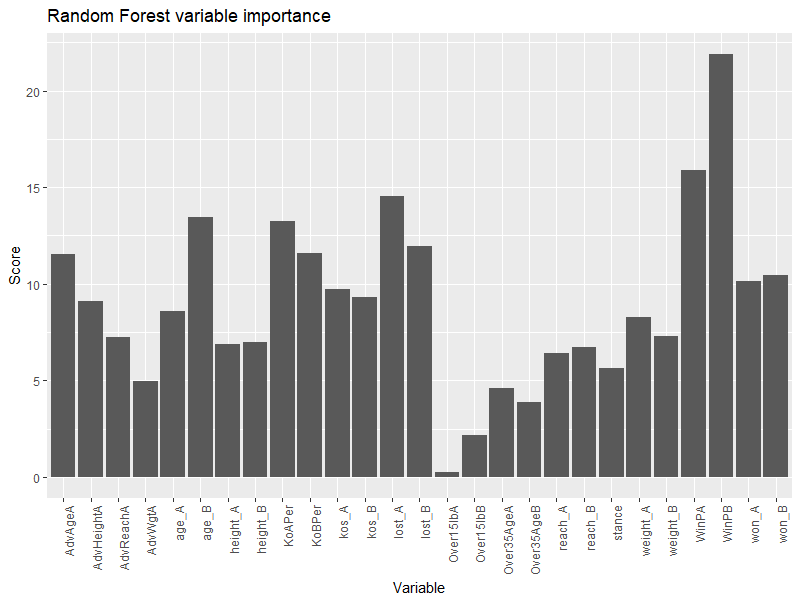
Number of trees: 400

No. of variables tried at each split: 4

OOB estimate of error rate: 29.31%

Model Characteristics

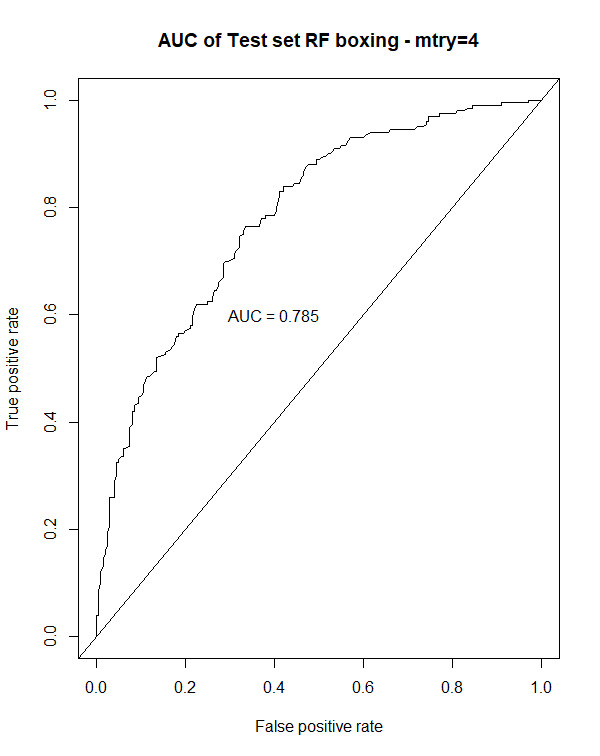
An examination of our random forest model shows that win percentage for each fight appear to have the highest scores for our model, but unfortunately there are no other standout predictors and our interpretability of a random forest model is somewhat limited.



Testing the dataset with a 80/20 split of our data shows the following confusion matrix and modest predictive performance

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Actual (random forest CM) | |
|  |  | Fighter A Wins (0) | Fighter B Wins (1) |
| Predicted | Fighter A Wins (0) | 138 | 62 |
| Fighter B Wins (1) | 59 | 141 |

This also yields the following ROC curve which is slightly more promising and appears to show a modestly good predictive power.



Conclusion

After examining logistic regression with and without additional variabels, LDA, and random forest modeling techniques, it appears that INSERT SOME TYPE OF MODEL SELECTION HERE AND JUSTIFICATION

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Metrics** | **Logistic Regression** | **Logistic Regression (complex)** | **LDA** | **Random Forest** |
| Accuracy |  |  |  |  |
| Sensitivity |  |  |  |  |
| Specificity |  |  |  |  |

SAS Code

%***web\_drop\_table***(WORK.BOXING);

FILENAME REFFILE "C:/Users/danie/Documents/GitHub/6372BoxingProject/train.csv";

**PROC** **IMPORT** DATAFILE=REFFILE

DBMS=CSV

OUT=WORK.BOXING;

GETNAMES=YES;

**RUN**;

**PROC** **PRINT** data=boxing;

ods graphics on;

**proc** **corr** data=boxing plots=matrix(histogram) PLOTS(MAXPOINTS=**9999**);

**run**;

ods graphics off;

**proc** **sgpanel** data=boxing;

panelby stance;

reg x=WinPA y=WinPB / group=binaryresult alpha = **.05** CLM CLI;

/\* Model Selection \*/

**PROC** **logistic** data= boxing;

class Stance Over35AgeA Over35AgeB Over15lbA Over15lbB;

model binaryresult = age\_A age\_B height\_A height\_B reach\_A reach\_B weight\_A weight\_B won\_A won\_B lost\_A lost\_B kos\_A kos\_B AdvAgeA AdvHeightA AdvReachA AdvWgtA WinPA WinPB KoAPer KoBPer

/ selection = stepwise;

output out=boxinglogregout predprobs=I p=probpreb;

**run**;

/\* Chosen Model\*/

**PROC** **logistic** data= boxing plots(only label)=(leverage dpc);

model binaryresult = lost\_A lost\_B won\_B WinPA AdvAgeA /LACKFIT CTABLE;

output out=boxinglogregout predprobs=I p=probpreb resdev=resdev reschi=pearres;

**run**;

**proc** **print** data=boxing;

where obsno = **3426**;

**run**;

/\* Candidate 1 remove age\_B \*/

/\* PROC logistic data= boxing; \*/

/\* model binaryresult = lost\_A lost\_B won\_B WinPA /LACKFIT CTABLE; \*/

/\* output out=boxinglogregout predprobs=I p=probpreb resdev=resdev reschi=pearres; \*/

/\* run; \*/

/\* \*/

/\* Candidate 1.5 add age A \*/

/\* PROC logistic data= boxing; \*/

/\* model binaryresult = age\_A age\_B lost\_A lost\_B won\_B WinPA /LACKFIT CTABLE; \*/

/\* output out=boxinglogregout predprobs=I p=probpreb resdev=resdev reschi=pearres; \*/

/\* run; \*/

/\* \*/

/\* Candidate 2 age A and B interaction \*/

/\* PROC logistic data= boxing; \*/

/\* model binaryresult = lost\_A lost\_B won\_B WinPA age\_A\*age\_B /LACKFIT CTABLE; \*/

/\* output out=boxinglogregout predprobs=I p=probpreb resdev=resdev reschi=pearres; \*/

/\* run; \*/

**proc** **gplot** data=boxinglogregout;

plot resdev\*obsno;

plot pearres\*obsno;

**run**; **quit**;

**proc** **print** data=boxinglogregout;

where pearres > **2.5**;

**run**;

/\* Remove outliers and run again \*/

**DATA** boxingRemovedOutliers;

SET boxing;

IF obsno = **3426** THEN DELETE;

IF obsno = **2901** THEN DELETE;

IF obsno = **3689** THEN DELETE;

IF obsno = **2856** THEN DELETE;

**RUN**;

**PROC** **logistic** data= boxingRemovedOutliers plots(only label)=(leverage dpc);

model binaryresult = lost\_A lost\_B won\_B WinPA AdvAgeA /LACKFIT CTABLE;

output out=boxinglogregoutRemovedOutliers predprobs=I p=probpreb resdev=resdev reschi=pearres;

**run**;

R Code

library(glmnet)

library(ROCR)

library(MASS)

library(ggplot2)

library(pheatmap)

library(randomForest)

library(dplyr)

library(caTools)

library(caret)

library(ROCR)

library(reshape)

#"Retrospective Study"

#Mention the transformation issues (i.e we removed the draw observations) we did in the data. Perhaps A Wins or doesn't win. MAybe include draws.

#COnsider LDA for all 3

#Assumptions:

#Multivariate normal distribution for entire set of variables

#Univariate normal distribution on response

#Linear relationships between scores on Y and scores on X for all variables

#Uniform error variances for response (Y) across all values of X

##main file for boxing casestudy

setwd("~/GitHub/6372BoxingProject")

#read in data - may need to remove stance na values

df <- read.csv("data.csv", na.strings = c("", "NA"))

#####################filter out bad & unnecessary data ########################################################

#eliminate draw outcomes

df <- filter(df, result != "draw")

#eliminate draw history & judge data since won't function as a predictor

df <- df[, -c(15, 16, 20:26)]

#filter out missing data

df <- df[complete.cases(df[, c(1:6, 9, 10)]), ]

#filter out weird age values

df <- subset(df, subset = (df$age\_A >= 16 & df$age\_A <= 60))

df <- subset(df, subset = (df$age\_B >= 16 & df$age\_B <= 60))

#filter out weird reach values

df <- subset(df, subset = (df$reach\_A >= 100 & df$reach\_A <= 300))

df <- subset(df, subset = (df$reach\_B >= 100 & df$reach\_B <= 300))

#assign as factors

df$result <- factor(df$result)

##############################potential modeling factors for interaction effects ########################

#create age delta

df$AdvAgeA <- df$age\_A - df$age\_B

#create height delta

df$AdvHeightA <- df$height\_A - df$height\_B

#create reach delta

df$AdvReachA <- df$reach\_A - df$reach\_B

#create weight delta

df$AdvWgtA <- df$weight\_A - df$weight\_B

#over 35 age binary ###35 is the age limit for amateur boxing, some have argued limits should exist for pro's###

df$Over35AgeA <- ifelse(df$age\_A >= 35, 1, 0)

df$Over35AgeB <- ifelse(df$age\_B >= 35, 1, 0)

#over 15 lbs weight delta?

df$Over15lbA <- ifelse(df$AdvWgtA >= 15, 1, 0)

df$Over15lbB <- ifelse(df$AdvWgtA <= -15, 1, 0)

#win % for boxers

df$WinPA <- df$won\_A / (df$won\_A + df$lost\_A)

df$WinPB <- df$won\_B / (df$won\_B + df$lost\_B)

#KO per fight

df$KoAPer <- df$kos\_A / (df$won\_A + df$lost\_A)

df$KoBPer <- df$kos\_B / (df$won\_B + df$lost\_B)

#add binaryresult value: 0 means that A won. 1 means that B won

df$binaryresult <- ifelse(df$result == "win\_A", 0, 1) #doublecheck that binary result is correct #unique(df$binaryresult)

#Recategorize the stances that are NA.

df$stance\_A <- as.character(df$stance\_A)

df$stance\_B <- as.character(df$stance\_B)

df$stance\_A[is.na(df$stance\_A)] <- "Unknown"

df$stance\_B[is.na(df$stance\_B)] <- "Unknown"

#Check to make sure NA stances are recategorized #unique(df$stance\_A) #unique(df$stance\_B)

#Notcied that the stance is same for A and B in all observations to keep only A and re-label it

df$stance<-df$stance\_A

df<-df[c(-7,-8)]

#unique(df$stance)

#Remove any reminaing NA observations

df <- na.omit(df)

#nrow(df) result is 7135 observations

############################EDA#################################################

#Need to add histograms, box Plots, corr and cov matrices,

#DD: Clearly our data is overepresenting the scenario when a wins vs when b wins.

hist(df$binaryresult)

#Split df by category

df\_A\_wins <- filter(df, binaryresult == 0)

df\_B\_wins <- filter(df, binaryresult == 1)

#nrow(df\_A\_wins) shows 6094 obs where a won

#nrow(df\_B\_wins) shows 1041 obs where b won

set.seed(42)

#Randomly sample approx 20% of useable observations per category and combine back together

#dftrain <- rbind(sample\_n(df\_A\_wins, 1040), sample\_n(df\_B\_wins, 1040))

#creates randomsample

df\_A\_wins<-sample\_n(df\_A\_wins, 1000)

df\_B\_wins<-sample\_n(df\_B\_wins, 1000)

#creates split @80%

SplitAWin<-sample.split(df\_A\_wins$binaryresult,SplitRatio = 0.8)

SplitBWin<-sample.split(df\_B\_wins$binaryresult,SplitRatio = 0.8)

#creates training and test dataset

trainingA<-subset(df\_A\_wins,SplitAWin==TRUE)

trainingB<-subset(df\_B\_wins,SplitBWin==TRUE)

testA<-subset(df\_A\_wins,SplitAWin==FALSE)

testB<-subset(df\_B\_wins,SplitBWin==FALSE)

#creates train test

dftrain<-rbind(trainingA,trainingB)

dftest<-rbind(testA,testB)

#write.csv(dftrain, file = "dftrain.csv")

hist(dftrain$binaryresult)

hist(dftest$binaryresult)

############################PCA#################################################

#Choose continuous variables as x axis

boxing.x <-

subset(

dftrain,

select = c(

AdvAgeA,

AdvHeightA,

AdvReachA,

AdvWgtA,

WinPA,

WinPB,

KoAPer,

KoBPer

)

)

##Choose binary result as y axis

boxing.y<-subset(dftrain, select = c(binaryresult))

boxing.y <- as.factor(as.character(boxing.y))

#Scale x variables and run pcomp

pcresults <- prcomp(boxing.x, scale = TRUE)

#BiPlot shows that reach, weight and height aren't adding quite as much value as others

biplot(pcresults, scale = 0)

#Put PC Scores into dataframe

pcscores <- as.data.frame(pcresults$x)

#Combine the pc scores with variable with the df

pcscores$binaryresult<-boxing.y

pcscores<-data.frame(pcscores)

pceigen<-(pcresults$sdev)^2

pcprop<-pceigen/sum(pceigen)

pccumprop<-cumsum(pcprop)

plot(pcprop,type="l",main="Scree Plot",ylim=c(0,1),xlab="PC #",ylab="Proportion of Variation")

lines(pccumprop,lty=3)

#Combine everything into dftrain and ggplot

dftrain <- cbind(dftrain, pcresults$x)

#Need to think about this.

ggplot(dftrain, aes(PC1, PC2, col = result, fill = result)) +

stat\_ellipse(geom = "polygon", col = "black", alpha = 0.5) +

geom\_point(shape = 21, col = "black")

ggplot(dftrain, aes(PC2, PC3, col = result, fill = result)) +

stat\_ellipse(geom = "polygon", col = "black", alpha = 0.5) +

geom\_point(shape = 21, col = "black")

ggplot(dftrain, aes(PC1, PC3, col = result, fill = result)) +

stat\_ellipse(geom = "polygon", col = "black", alpha = 0.5) +

geom\_point(shape = 21, col = "black")

####begin random forest work#####

#drop pca & unnecessary data

dftrain<-dftrain[,c(-15,-30:-37)]

dftrain$binaryresult<-as.factor(dftrain$binaryresult)

dftrain$stance<-as.factor(dftrain$stance)

dftest<-dftest[,-15]

dftest$binaryresult<-as.factor(dftest$binaryresult)

dftest$stance<-as.factor(dftest$stance)

str(dftrain)

summary(dftrain)

colSums(is.na(dftrain)|dftrain == '')

set.seed(42)

#runs random forest

rf.box <- randomForest(x = dftrain[,-27],y = dftrain[,27],mtry=4,ntree = 400,importance = T)

rf.box

#performs OOB estimation

bestmtry<-tuneRF(x = dftrain[,-27],y=dftrain[,27],stepFactor=1.5, ntree=400)

print(bestmtry)

#use testdata

y\_pred <- predict(rf.box, newdata = dftest[-27])

#confusion matrix

cm <- table(dftest[, 27], y\_pred)

cm

#RF Accuracy

(138+141)/(400)

#RF Sensitivity

138/(59+138)

#RF Specificity

141/(141+62)

#creates table for randome forest variables

rf.imp<-varImp(rf.box)

rf.imp$variable<-row.names(rf.imp)

rf.imp<-rf.imp[,-1]

colnames(rf.imp)<-c("Score","Variable")

rf.imp$Variable<-as.factor(rf.imp$Variable)

#creates plot of variable importance

ggplot(rf.imp, aes(y=Score, x=Variable)) +

geom\_bar( stat="identity", show.legend = F) + ggtitle("Random Forest variable importance") +

theme(axis.text.x = element\_text(angle = 90, hjust = 1))

#barplot()

###end

#Go get the ROC

rf.pred<-predict(rf.box,newdata=dftest[,-27],type="prob")

pred <- prediction(rf.pred[,2], dftest$binaryresult)

roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")

auc.train <- performance(pred, measure = "auc")

auc.train <- auc.train@y.values

plot(roc.perf,main="AUC of Test set RF boxing - mtry=4")

abline(a=0, b= 1)

text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))

############################begin LDA#################################

#creates data for LDA removing correlated/created variables and class variable of stance

dfldatrain<-dftrain

dfldatrain<-dfldatrain[,c(-15:-26,-28)]

dfldatest<-dftest

dfldatest<-dfldatest[,c(-15:-26,-28)]

#fits lda

lda.fit<-lda(binaryresult ~ ., data=dfldatrain)

lda.fit

#predicts for lda

lda.pred <- predict(lda.fit, dfldatest)

names(lda.pred)

#makes confusion matrix

table(lda.pred$class, dfldatest$binaryresult)

#create lda roc

lpred <- prediction(lda.pred$posterior[,2], dfldatest$binaryresult)

roc.perf.lda = performance(lpred, measure = "tpr", x.measure = "fpr")

auc.train <- performance(lpred, measure = "auc")

auc.train <- auc.train@y.values

plot(roc.perf.lda,main="AUC of Test set LDA boxing")

abline(a=0, b= 1)

text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))

#LDA Accuracy

(114+137)/(400)

#LDA Sensitivity

141/(59+141)

#LDA Specificity

137/(137+59)