|  |
| --- |
| ###PACKAGES### |
|  | library(pacman) |
|  | p\_load(rvest,tidyverse,janitor,tidymodels,broom, |
|  | rpart,rpart.plot) |
|  |  |
|  | final\_df$allstar= as.factor(final\_df$allstar) |
|  | ###DATA### |
|  | ##Read in data, use janitor::clean\_names for renaming columns |
|  |  |
|  | #background info |
|  | playerstats\_df <- read.csv("data/player\_data.csv") %>% clean\_names()%>%rename(player = name) |
|  |  |
|  | #physical characteristics |
|  | players\_df <- read.csv("data/Players.csv") %>% clean\_names() %>% |
|  | mutate(player = gsub(" +$","",gsub("[^[:alpha:]]"," ",player))) |
|  |  |
|  | #season stats |
|  | seasonstats\_df <- read.csv("data/Seasons\_Stats.csv") %>% |
|  | #Change year format to be more specific (from xxxx-yy to xxxx-yyyy) |
|  | mutate(season = paste(Year-1,Year,sep = "-")) %>% |
|  | #reorder columns for convenience |
|  | .[,c(1,2,54,3:53)] %>% clean\_names() %>% |
|  | #Only considering data after the 1999 season |
|  | filter(year>1999) %>% |
|  |  |
|  | #Clean up player names |
|  | mutate(player = gsub(" +$","",gsub("[^[:alpha:]]"," ",player))) |
|  |  |
|  | #Read in wikipedia scrapped data |
|  | #MVP data |
|  | mvp <- read\_html("https://en.wikipedia.org/wiki/NBA\_Most\_Valuable\_Player\_Award") %>% |
|  | html\_nodes(xpath = "//\*[@id='mw-content-text']/div[1]/table[4]") %>% |
|  | html\_table() %>% data.frame() %>% clean\_names() %>% |
|  | #clean up season and player variables |
|  | mutate(season = gsub("[^[:alnum:]]","-19",season), |
|  | player = gsub(" +$","",gsub("[^[:alpha:]]"," ",player))) %>% |
|  | #rename player variable to ensure smooth joining with other dfs |
|  | rename("year\_mvp" = "player") %>% |
|  | .[,c(1,2)] |
|  | mvp$season[45:65] <- gsub("^([0-9]{4})([^[:alnum:]]{1}[0-9]{2})([0-9]{2})$","\\1-20\\3",mvp$season[45:65]) |
|  | mvp$year\_mvp[44] <- "Karl Malone" |
|  | mvp$year\_mvp[56] <- "Derrick Rose" |
|  |  |
|  | #All-star selections |
|  | allstars <- read\_html("https://en.wikipedia.org/wiki/List\_of\_NBA\_All-Stars") %>% |
|  | html\_nodes(xpath = "//\*[@id='mw-content-text']/div[1]/table[2]") %>% |
|  | html\_table() %>% data.frame() %>% clean\_names() %>% |
|  | #clean up player variable, and prepare allstar selections for future use |
|  | mutate(player = gsub(" +$","",gsub("[^[:alpha:]]"," ",player)), |
|  | selections = strsplit(selections\_c,split = ";"), |
|  | selections\_c = NULL) %>% |
|  | .[,c(1,5)] |
|  | #manually fix minor issues from webscraping |
|  | allstars$player[1] <- "Kareem Abdul Jabbar" |
|  | allstars$player[19] <- "Hakeem Olajuwon" |
|  | allstars$player[430] <- "Metta World Peace" |
|  |  |
|  | #merging the player df's |
|  | df\_halffull <- left\_join(seasonstats\_df, players\_df,playerstats\_df, by = "player") |
|  |  |
|  | #create mvp column with TRUE/FALSE values |
|  | full\_df <- left\_join(df\_halffull, mvp, by ="season") %>% |
|  | mutate(mvp = (year\_mvp == player)) |
|  |  |
|  | #Create allstar column with TRUE/FALSE values |
|  | full\_df$allstar <- FALSE |
|  | for (i in 1:nrow(allstars)) { |
|  |  |
|  | #Pickup selections in their source format from webscrapped table |
|  | selections <- allstars$selections[[i]] |
|  |  |
|  | #Turn the format into yyyy:yyyy for further use |
|  | selections\_seq <- gsub("[^[:alnum:]]",":",gsub(" ","",selections)) |
|  |  |
|  | #put years selected into a useable format: a df |
|  | #player= player selected; selections = years selected for allstar game |
|  | years\_selected <- data.frame( |
|  | player = allstars$player[i], |
|  | selections = unlist(lapply(selections\_seq,function(x) {eval(parse(text = x))})) |
|  | ) |
|  |  |
|  | #turn values of full\_df's allstar column to true based on membership in years\_selected |
|  | full\_df[(full\_df$player == years\_selected$player[1] & |
|  | full\_df$year %in% years\_selected$selections),"allstar"] <- TRUE |
|  | } |
|  |  |
|  | #create final df with finalized variables |
|  | final\_df <- full\_df %>% |
|  | #create an age column |
|  | mutate(age = year-born) %>% |
|  | #reorder columns for convenience |
|  | .[,c(2:4,63:64,56:59,60,61,5:54)] %>% |
|  | select(-c(blanl,blank2)) %>% |
|  | mutate(mvp = as\_factor(as.numeric(mvp)), allstar = as\_factor(as.numeric(allstar))) |
|  |  |
|  | #for loop creating dummy variables for different positions |
|  | for (i in c("C","PG","PF","SG","SF")) { |
|  | eval(parse(text = paste("final\_df$",i,"<- as.numeric(grepl('",i,"',final\_df$pos))",sep = ""))) |
|  | } |
|  |  |
|  | ###Elasticnet Logistic Regression### |
|  | set.seed(8237) |
|  |  |
|  | #split df into training and testing sets by randomly choosing years, each as their own sample |
|  | #18 years possible, we will be using 80% (rounded up to 15 total years) for our training and the remainder for testing. |
|  | train\_years <- sample(unique(final\_df$year),15) %>% |
|  | sort() |
|  | test\_years <- unique(final\_df$year)[!unique(final\_df$year) %in% train\_years] %>% |
|  | sort() |
|  |  |
|  | final\_train <- final\_df[final\_df$year %in% train\_years,] |
|  | final\_test <- final\_df[final\_df$year %in% test\_years,] |
|  |  |
|  | #Use each year as it's own sample |
|  | final\_cv <- final\_train %>% group\_vfold\_cv(group = year) |
|  |  |
|  | #Create recipe and prepare it for use |
|  | final\_recipe <- final\_train %>% recipe(allstar ~ .) %>% |
|  | #Remove problematic variables |
|  | step\_rm(player) %>% step\_rm(collage) %>% step\_rm(pos) %>% |
|  | step\_rm(birth\_city) %>% step\_rm(birth\_state) %>% |
|  | step\_rm(season) %>% step\_rm(tm) %>% |
|  | #normalize numeric variables |
|  | step\_normalize(all\_predictors() & all\_numeric()) %>% |
|  | #turn categorical variables into dummy variables |
|  | step\_dummy(all\_predictors() & all\_nominal()) %>% |
|  | #impute categorical variables |
|  | step\_modeimpute(all\_predictors()&all\_nominal()) %>% |
|  | #impute numeric variables |
|  | step\_meanimpute(all\_predictors()&all\_numeric()) |
|  | final\_clean <- final\_recipe %>% prep() %>% juice() |
|  |  |
|  | #prepare elasticnet logit model |
|  | model\_en <- logistic\_reg(penalty = tune(), mixture = tune()) %>% |
|  | set\_engine("glmnet") |
|  |  |
|  | #prepare elasticnet logit workflow |
|  | workflow\_en = workflow() %>% |
|  | add\_model(model\_en) %>% |
|  | add\_recipe(final\_recipe) |
|  |  |
|  | #calculate the best models |
|  | cv\_en = workflow\_en %>% |
|  | tune\_grid( |
|  | final\_cv, |
|  | grid = grid\_regular(mixture(), penalty(), levels = 3), |
|  | metrics = metric\_set(accuracy) |
|  | ) |
|  |  |
|  | final\_en = workflow\_en %>% |
|  | finalize\_workflow(select\_best(cv\_en, 'accuracy')) |
|  |  |
|  | #Fitting the final model |
|  | final\_fit\_en = final\_en %>% fit(data = final\_train) |
|  |  |
|  | #Predict onto the test data |
|  | y\_hat = final\_fit\_en %>% predict(new\_data = final\_test, type ="class") |
|  |  |
|  | cm\_logistic = conf\_mat( |
|  | data = tibble( |
|  | y\_hat = y\_hat %>% unlist(), |
|  | y = final\_test$allstar |
|  | ), |
|  | truth = y, estimate = y\_hat |
|  | ) |
|  |  |
|  | #view confusion matrix |
|  | cm\_logistic |
|  |  |
|  |  |
|  | #confusion matrix with probabilities |
|  |  |
|  | #Predict onto the test data |
|  | p\_hatlogit = final\_fit\_en %>% predict(new\_data = final\_test, type ="prob") |
|  |  |
|  | cm\_logisticp = conf\_mat( |
|  | data = tibble( |
|  | p\_hatlogit = p\_hatlogit$.pred\_2, |
|  | plogit = final\_test$allstar |
|  | ), |
|  | truth = plogit, estimate = p\_hatlogit |
|  | ) |
|  |  |
|  |  |
|  | #create logit dataframe |
|  | logit\_df = data.frame( |
|  | player = final\_test$player, |
|  | year = final\_test$year, |
|  | allstar = p\_hatlogit$.pred\_2 |
|  | ) |
|  |  |
|  |  |
|  | ##graph 2006 predictions |
|  | logit\_df%>% filter(allstar>.9, year==2006)%>%ggplot(aes(x=allstar, y= reorder(player,(allstar)),fill=player))+geom\_col()+theme\_classic(base\_size = 12)+scale\_fill\_viridis\_d() +xlab("Prediction") +ylab("Player")+ ggtitle("2006 Allstar Predictions Logistic Regression Model") |
|  |  |
|  | #logistic visual |
|  | cm\_logistic$table %>% as.data.frame() %>% |
|  | mutate(result = c("True Negative","False Positive","False Negative","True Positive")) %>% |
|  | ggplot(aes(x = result, y = Freq)) + |
|  | geom\_bar(stat = "identity") +xlab("Result")+ylab("Frequency") +ggtitle("Allstar Predictions from Logistic Model")+theme\_bw()+scale\_fill\_brewer() |
|  |  |
|  | #graph without true negatives |
|  | cm\_logistic$table %>% as.data.frame() %>% |
|  | mutate(result = c("True Negative","False Positive","False Negative","True Positive")) %>% |
|  | filter(result!= "True Negative")%>% |
|  | ggplot(aes(x = result, y = Freq, fill =result)) + |
|  | geom\_bar(stat = "identity") + xlab("Result")+ylab("Frequency") +ggtitle("Allstar Predictions from Logistic Model")+theme\_bw()+scale\_fill\_brewer() |
|  |  |
|  | #logistic |
|  | cm\_logistic$table %>% as.data.frame() %>% |
|  | mutate(result = c("True Negative","False Positive","False Negative","True Positive")) %>% |
|  | filter(result!= "True Negative")%>% |
|  | ggplot(aes(x = result, y = Freq, fill =result)) + labs(fill = "Result")+ |
|  | geom\_bar(stat = "identity") + xlab("Result")+ylab("Frequency") +ggtitle("Allstar Predictions from Logistic Model")+theme\_classic(base\_size = 12)+scale\_fill\_manual(values=c( "#B54213","#161616", "#EF7F4D" )) |
|  |  |
|  |  |
|  | ###Decision Tree### |
|  | #decision tree model |
|  | model\_tree <- decision\_tree( |
|  | mode = "classification", |
|  | cost\_complexity = tune(), |
|  | tree\_depth = tune(), |
|  | min\_n = 10 |
|  | ) %>% |
|  | set\_engine("rpart") |
|  |  |
|  | #setup decision tree workflow |
|  | tree\_workflow <- workflow() %>% |
|  | add\_model(model\_tree) %>% |
|  | add\_recipe(final\_recipe) |
|  |  |
|  | #tune models |
|  | tree\_cv\_fit <- tree\_workflow %>% tune\_grid( |
|  | final\_cv, |
|  | grid = expand\_grid( |
|  | cost\_complexity = seq(0,.15,by = .05), |
|  | tree\_depth = c(1,5,10) |
|  | ), |
|  | metrics = metric\_set(accuracy, roc\_auc) |
|  | ) |
|  |  |
|  | #select the best model based on accuracy |
|  | best\_flow\_tree <- tree\_workflow %>% |
|  | finalize\_workflow(select\_best(tree\_cv\_fit, metric = "accuracy")) %>% |
|  | fit(data = final\_train) |
|  |  |
|  | #pull the best decision tree model |
|  | best\_tree <- best\_flow\_tree %>% pull\_workflow\_fit() |
|  |  |
|  | #plot decision tree |
|  | best\_tree$fit %>% rpart.plot(cex = 1) |
|  | #Fitting the final model |
|  | final\_fit\_tree = best\_flow\_tree %>% fit(data = final\_train) |
|  |  |
|  | #Predict onto the test data |
|  | y\_hat\_tree = final\_fit\_tree %>% predict(new\_data = final\_test, type ="class") |
|  |  |
|  | #confusion matrix for decision tree |
|  | cm\_logistic\_tree = conf\_mat( |
|  | data = tibble( |
|  | y\_hat = y\_hat\_tree %>% unlist(), |
|  | y = final\_test$allstar |
|  | ), |
|  | truth = y, estimate = y\_hat |
|  | ) |
|  |  |
|  | #decision tree visuals |
|  | cm\_logistic\_tree$table %>% as.data.frame() %>% |
|  | mutate(result = c("True Negative","False Positive","False Negative","True Positive")) %>% |
|  | ggplot(aes(x = result, y = Freq)) + |
|  | geom\_bar(stat = "identity") + |
|  | geom\_text(aes(x = "False Negative",y = c(600,750,900,1050),label = paste(result,"s: ",Freq,sep = ""))) |
|  |  |
|  |  |
|  | cm\_logistic\_tree |
|  |  |
|  | #predict probabilities |
|  | p\_hat\_tree = final\_fit\_tree %>% predict(new\_data = final\_test, type ="prob") |
|  |  |
|  | #confusion matrix from estimates |
|  | cm\_logistic\_treep = conf\_mat( |
|  | data = tibble( |
|  | p\_hat = p\_hat\_tree %>% unlist(), |
|  | p = final\_test$allstar |
|  | ), |
|  | truth = p, estimate = p\_hat |
|  | ) |
|  |  |
|  |  |
|  | ##create a decision tree data frame |
|  | tree\_df = data.frame( |
|  | player = final\_test$player, |
|  | year = final\_test$year, |
|  | allstar = p\_hat\_tree |
|  | ) |
|  |  |
|  | ##graph 06 predictions |
|  |  |
|  | tree\_df%>% filter(allstar..pred\_2>0.9, year==2006)%>%ggplot(aes(x=allstar..pred\_2, y= reorder(player,(allstar..pred\_2)),fill=player))+geom\_col()+theme\_classic(base\_size = 12)+scale\_fill\_viridis\_d() +xlab("Prediction") +ylab("Player")+ ggtitle("2006 Allstar Predictions Decision Tree") |
|  |  |
|  |  |
|  | ##graph without true negatives |
|  | cm\_logistic\_tree$table %>% as.data.frame() %>% |
|  | mutate(result = c("True Negative","False Positive","False Negative","True Positive")) %>% |
|  | filter(result!= "True Negative")%>% |
|  | ggplot(aes(x = result, y = Freq, fill =result)) + |
|  | geom\_bar(stat = "identity") + xlab("Result")+ylab("Frequency") +ggtitle("Allstar Predictions from Decision Tree Model")+theme\_bw()+scale\_fill\_brewer() |
|  |  |
|  | ###Elasticnet Linear Regression### |
|  | #change allstar to numeric for compatibility with linear regression |
|  | final\_df$allstar <- as.numeric(final\_df$allstar)-1 |
|  | final\_train$allstar <- as.numeric(final\_train$allstar)-1 |
|  | final\_test$allstar <- as.numeric(final\_test$allstar)-1 |
|  |  |
|  | #rerun cv splits for updated outcome class |
|  | final\_cv2 <- final\_train %>% group\_vfold\_cv(group = year) |
|  |  |
|  | #rerun recipe in case it needs to update for new class of outcome variable |
|  | final\_recipe2 <- final\_train %>% recipe(allstar ~ .) %>% |
|  | #Remove problematic variables |
|  | step\_rm(player) %>% step\_rm(collage) %>% step\_rm(pos) %>% |
|  | step\_rm(birth\_city) %>% step\_rm(birth\_state) %>% |
|  | step\_rm(season) %>% step\_rm(tm) %>% |
|  | #normalize numeric variables |
|  | step\_normalize(all\_predictors() & all\_numeric()) %>% |
|  | #turn categorical variables into dummy variables |
|  | step\_dummy(all\_predictors() & all\_nominal()) %>% |
|  | #impute categorical variables |
|  | step\_modeimpute(all\_predictors()&all\_nominal()) %>% |
|  | #impute numeric variables |
|  | step\_meanimpute(all\_predictors()&all\_numeric()) |
|  |  |
|  | #Elasticnet regression |
|  | model\_en = linear\_reg(penalty = tune(), mixture = tune()) %>% set\_engine("glmnet") |
|  |  |
|  | #Define the workflow |
|  | workflow\_en = workflow() %>% |
|  | add\_model(model\_en) %>% |
|  | add\_recipe(final\_recipe2) |
|  |  |
|  |  |
|  | #run the model |
|  | cv\_enlinreg = workflow\_en %>% |
|  | tune\_grid( |
|  | final\_cv2, |
|  | grid = grid\_regular(mixture(), penalty(), levels = 5:5), |
|  | metrics = metric\_set(rmse) |
|  | ) |
|  |  |
|  | cv\_en %>% collect\_metrics() %>% arrange(mean) |
|  |  |
|  | #finalize |
|  | enlinreg\_final <- workflow\_enlinreg%>% |
|  | finalize\_workflow(select\_best(cv\_enlinreg, metric="rmse")) |
|  |  |
|  | #fit final model |
|  | enlinreg\_final\_fit = enlinreg\_final%>%fit(data=final\_train) |
|  |  |
|  | #predict onto test data |
|  | test\_hat = enlinreg\_final\_fit %>% predict(new\_data = final\_test) |
|  |  |
|  | head(test\_hat) |
|  |  |
|  |  |
|  | enlinreg\_df = data.frame( |
|  | player = final\_test$player, |
|  | year = final\_test$year, |
|  | allstar = test\_hat |
|  | ) |
|  |  |
|  | #view |
|  | enlinreg\_df |
|  |  |
|  |  |
|  | ##graph of 2006 predictions |
|  | enlinreg\_df$player = with(enlinreg\_df, reorder(player,.pred)) |
|  | enlinreg\_df%>% filter(.pred>1.65, year==2006)%>%ggplot(aes(x=.pred, y= reorder(player,(.pred)),fill=player))+geom\_col()+theme\_classic(base\_size = 12)+scale\_fill\_viridis\_d() +xlab("Prediction") +ylab("Player")+ ggtitle("2006 Allstar Predictions Boosted Model") |
|  |  |
|  |  |
|  | final\_test$allstar\_reg <- test\_hat$.pred |
|  |  |
|  |  |
|  | #Now, lets take the top 27 estimates for each year and guess those as our allstars |
|  |  |
|  |  |
|  | for (i in unique(final\_test$year)) { |
|  | #generate cutoff value such that 25 players are above it |
|  | cutoff <- final\_test[final\_test$year == i,] %>% select(allstar\_reg) %>% |
|  | pull() %>% |
|  | sort(decreasing = TRUE) %>% |
|  | .[25] |
|  |  |
|  | #get vector of allstar regression results |
|  | year\_allstar\_reg <- final\_test[final\_test$year == i,] %>% |
|  | select(allstar\_reg) %>% pull() |
|  |  |
|  | #turn allstar\_values into binary |
|  | year\_allstar\_pred <- ifelse(year\_allstar\_reg>=cutoff,1,0) |
|  |  |
|  | #if statement: add to allstar\_pred\_vec |
|  | if (exists("allstar\_pred\_vec")) { |
|  | allstar\_pred\_vec <- append(allstar\_pred\_vec,year\_allstar\_pred) |
|  | } |
|  | #if statement: create original allstar\_pred\_vec |
|  | if (!exists("allstar\_pred\_vec")) { |
|  | allstar\_pred\_vec <- year\_allstar\_pred |
|  | } |
|  | } |
|  |  |
|  | #add predictions to df |
|  | final\_test$allstar\_pred <- allstar\_pred\_vec |
|  |  |
|  | #make predictions |
|  | y\_hat\_tree <- final\_test$allstar\_pred |
|  |  |
|  | #confusion matrix |
|  | cm\_lin\_reg = conf\_mat( |
|  | data = tibble( |
|  | y\_hat = as.factor(y\_hat\_tree) %>% unlist(), |
|  | y = as.factor(final\_test$allstar) |
|  | ), |
|  | truth = y, estimate = y\_hat |
|  | ) |
|  |  |
|  | #lin reg visuals |
|  | cm\_lin\_reg$table %>% as.data.frame() %>% |
|  | mutate(result = c("True Negative","False Positive","False Negative","True Positive")) %>% |
|  | ggplot(aes(x = result, y = Freq)) + |
|  | geom\_bar(stat = "identity") + |
|  | geom\_text(aes(x = "False Negative",y = c(600,750,900,1050),label = paste(result,"s: ",Freq,sep = ""))) |
|  |  |
|  | cm\_lin\_reg$table %>% as.data.frame() %>% |
|  | mutate(result = c("True Negative","False Positive","False Negative","True Positive")) %>% |
|  | filter(result!= "True Negative")%>% |
|  | ggplot(aes(x = result, y = Freq, fill =result)) + labs(fill = "Result")+ |
|  | geom\_bar(stat = "identity") + xlab("Result")+ylab("Frequency") +ggtitle("Allstar Predictions from Elasticnet Linear Regression Model")+theme\_classic(base\_size = 12)+scale\_fill\_manual(values=c( "#B54213","#161616", "#EF7F4D" )) |
|  |  |
|  |  |
|  | ### Boost model### |
|  | #define boost model |
|  | allstar\_boost = boost\_tree( |
|  | mtry= NULL, |
|  | trees=10, |
|  | min\_n =NULL, |
|  | tree\_depth = tune(), |
|  | learn\_rate = tune() |
|  | ) %>% set\_engine( |
|  | engine = "xgboost") %>% |
|  | set\_mode("classification") |
|  | final\_test$allstar <- as.factor(final\_test$allstar) |
|  |  |
|  |  |
|  | #define workflow |
|  | allstar\_boost\_wf = |
|  | workflow()%>% add\_model(allstar\_boost)%>%add\_recipe(final\_recipe) |
|  |  |
|  | #run model |
|  | cv\_boost = allstar\_boost\_wf %>% |
|  | tune\_grid( |
|  | final\_cv, |
|  | grid = grid\_regular(tree\_depth(), learn\_rate(), levels = 5:5), |
|  | metrics = metric\_set(accuracy) |
|  | ) |
|  |  |
|  | #show the best model |
|  | cv\_boost %>% show\_best() |
|  |  |
|  | #finalize workflow and use it to predict onto test data |
|  | final\_boost = |
|  | allstar\_boost\_wf %>% |
|  | finalize\_workflow(select\_best(cv\_boost, "accuracy")) |
|  |  |
|  |  |
|  | #fit final model |
|  |  |
|  | final\_fit\_boost = final\_boost %>% fit(data =final\_train) |
|  |  |
|  |  |
|  | #predict onto test data |
|  | p\_hat = final\_fit\_boost %>% predict(new\_data = final\_test, type="class") |
|  |  |
|  | #confusion matrix |
|  | cm\_boost = conf\_mat( |
|  | data =tibble( |
|  | p\_hat = p\_hat %>% unlist(), |
|  | p = final\_test$allstar |
|  | ), |
|  | truth = p, estimate = p\_hat |
|  | ) |
|  | #view matrix |
|  | cm\_boost |
|  |  |
|  |  |
|  | head(p\_hat) |
|  |  |
|  |  |
|  | ##turn confusion matrix into a graph |
|  | cm\_boost$table %>% as.data.frame() %>% |
|  | mutate(result = c("True Negative","False Positive","False Negative","True Positive")) %>% |
|  | ggplot(aes(x = result, y = Freq, fill =result)) +theme\_bw()+scale\_fill\_brewer() +geom\_bar(stat = "identity") + xlab("Result")+ylab("Frequency") + |
|  | geom\_text(aes(x = "False Negative",y = c(600,750,900,1050),label = paste(result,"s: ",Freq,sep = ""))) |
|  |  |
|  | ##graph without true negatives |
|  | cm\_boost$table %>% as.data.frame() %>% |
|  | mutate(result = c("True Negative","False Positive","False Negative","True Positive")) %>% |
|  | filter(result!= "True Negative")%>% |
|  | ggplot(aes(x = result, y = Freq, fill =result)) + |
|  | geom\_bar(stat = "identity") + xlab("Result")+ylab("Frequency") +ggtitle("Allstar Predictions from Boosted Model")+theme\_bw()+scale\_fill\_brewer() |