Ian Bogley
Jonas Bowman
Professor Rubin
Econ 524
10 March 2021

NBA All-Star Prediction

The aim of our project is to create a model that can predict the National Basketball Association (NBA) all-stars with a high degree of accuracy. To pursue this goal, we gathered data from various online sources and used created predictions using boosting, decision trees and other machine learning techniques

As stated above the goal of this project was to see if we could predict which players would be all-stars in any given NBA season. Understanding this question is important because many basketball historians and analysts believe that all-star voting is skewed because it is voted on by fans. Fan voting is believed to be less accurate as many fans may be biased towards players who have a more exciting style of play or who play for bigger market teams. Our goal is to test, this theory and determine if there are players in the data whose play merited a spot on the team but were ultimately snubbed. This is a prediction problem because we are not asking what causes someone to be an all-star, but rather who are the all-stars.

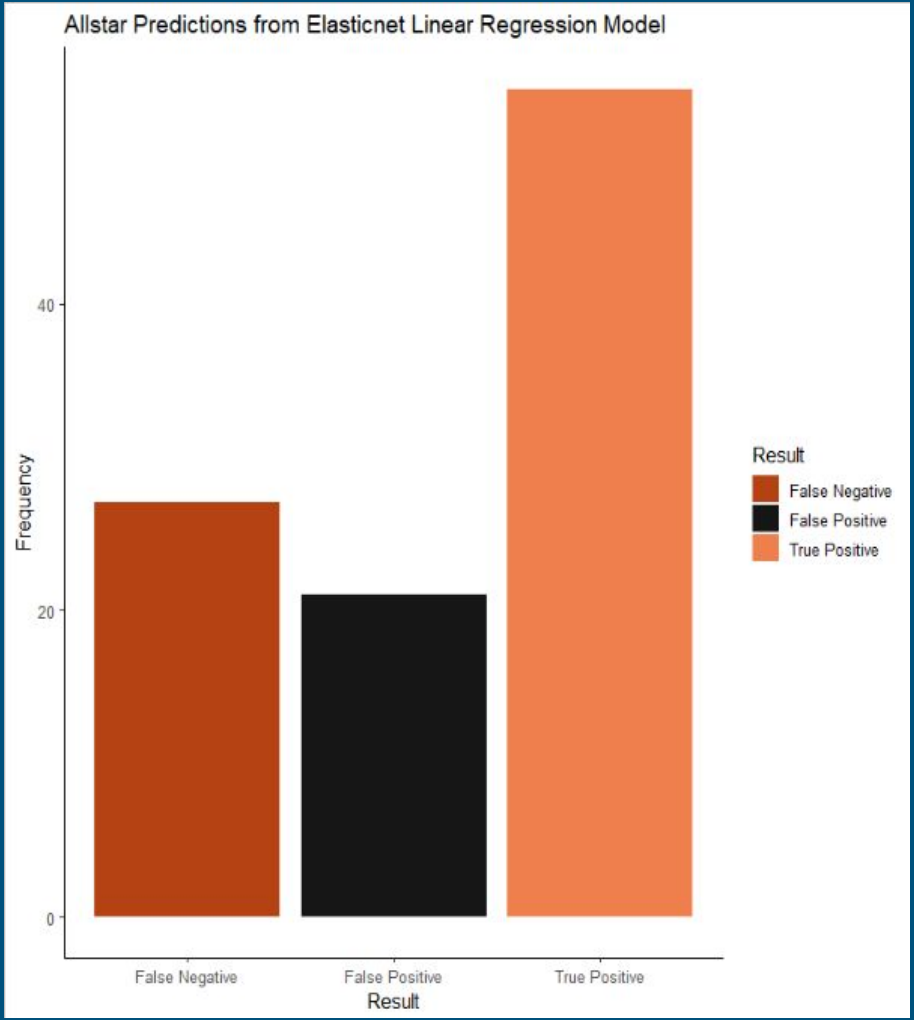
The data we gathered came from a Kaggle notebook called NBA player stats which we combined with data scraped from the web listing the all-stars and MVPs of each NBA season going back to the league's inception. In order to clean the data, we had to remove a few variables which were causing errors in our predictions and impute the mean and mode for numeric and nominal variables respectively. During the course of our work, we ran into challenges with the time scale of the data. The NBA has changed so much over the course of its existence that we decided to leave out every year before 2000 in order to get more accurate predictions. One major shortcoming of this data is the lack of all-star observations. The pool of all-star players is much smaller than the pool of non-all-star players.

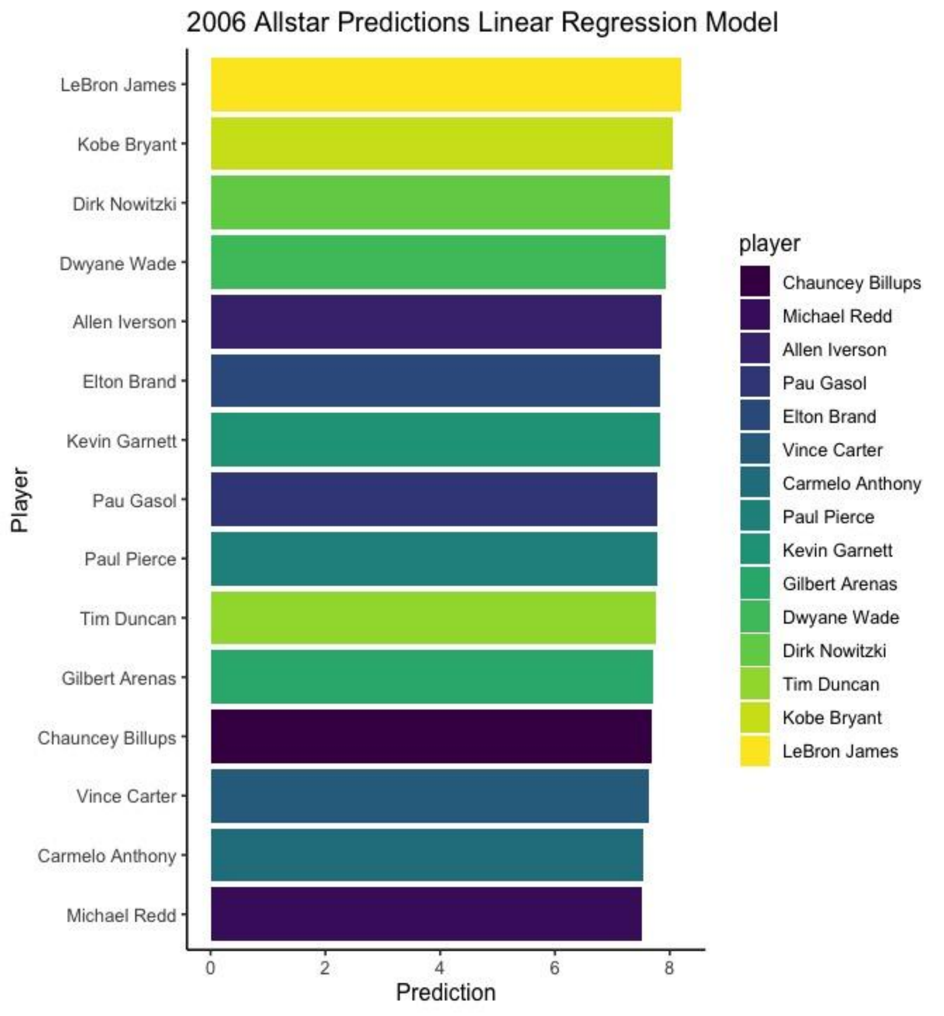
The models used include elastic net logistic regression, linear regression, decision tree, and boosting frameworks. The optimal parameters for the first method were a mixture of 1 (lasso model) and a penalty of 1*10^(-10).The linear regression model also used the same mixture and penalty. The decision tree split variables include steals, free throw percentage, minutes played, and field goal percentage. The best boosting model had 8 splits, a shrinkage parameter of 0.1, with 15 trees. For each of the models, we used accuracy as our metric for tuning. The only exception is the regular regression, which we tuned using root mean squared errors.

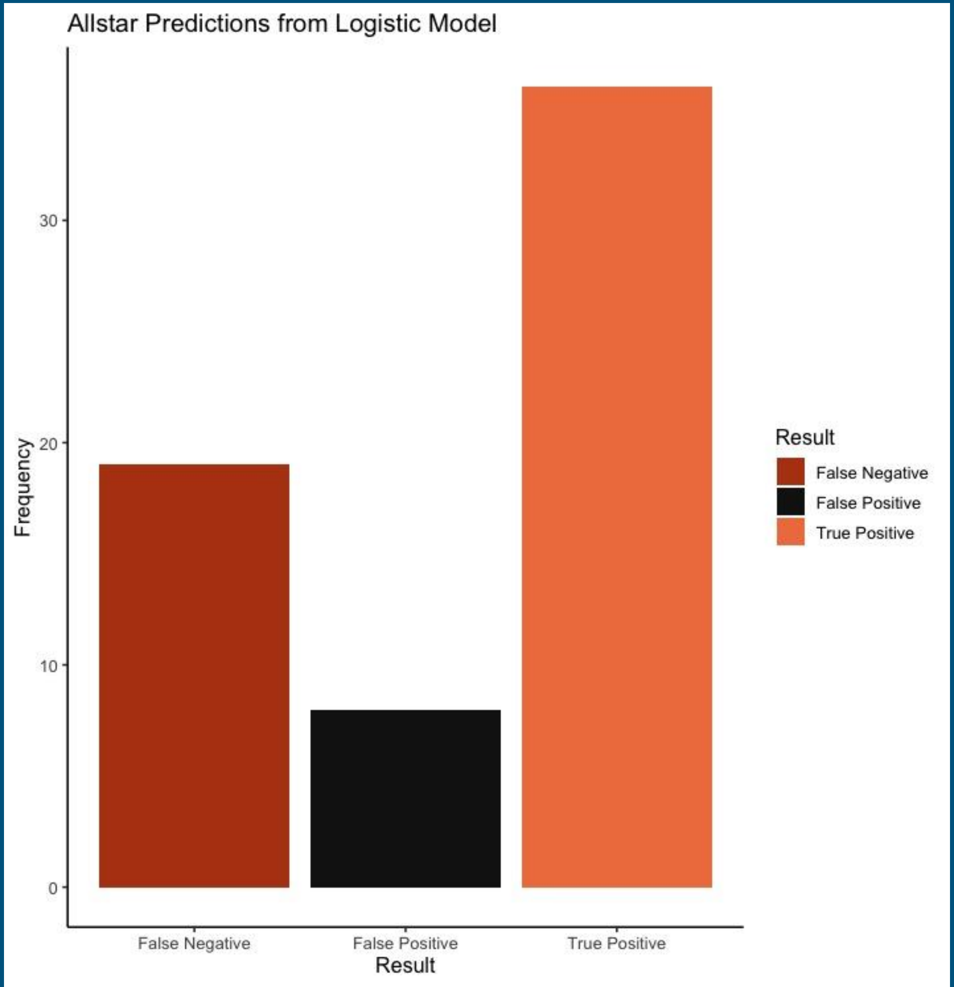
There are several ways to measure the success of our model. However, the focus of our metrics was on predicting true positives, as predicting true negatives is relatively easy given that most players in NBA history are not all-stars. Therefore, if true positives are weighted higher, the two options for determining model value are sensitivity and precision. For both sensitivity and precision, the logistic regression model scored the best (sensitivity: linreg= 66%, logistic = 72%, tree=59%,boost = 61%) (Precision: linreg=72%, logistic =81% ,tree = 73%, boost = 79%). One major aspect that limited our performance was the low variability of our outcome. Having so few positive outcome variables made it difficult to predict all-star, and even worse for MVPs. This was a big point of growth in our understanding, as the possible analysis that can be done on an outcome variable depends in large part on the variability of the outcome.

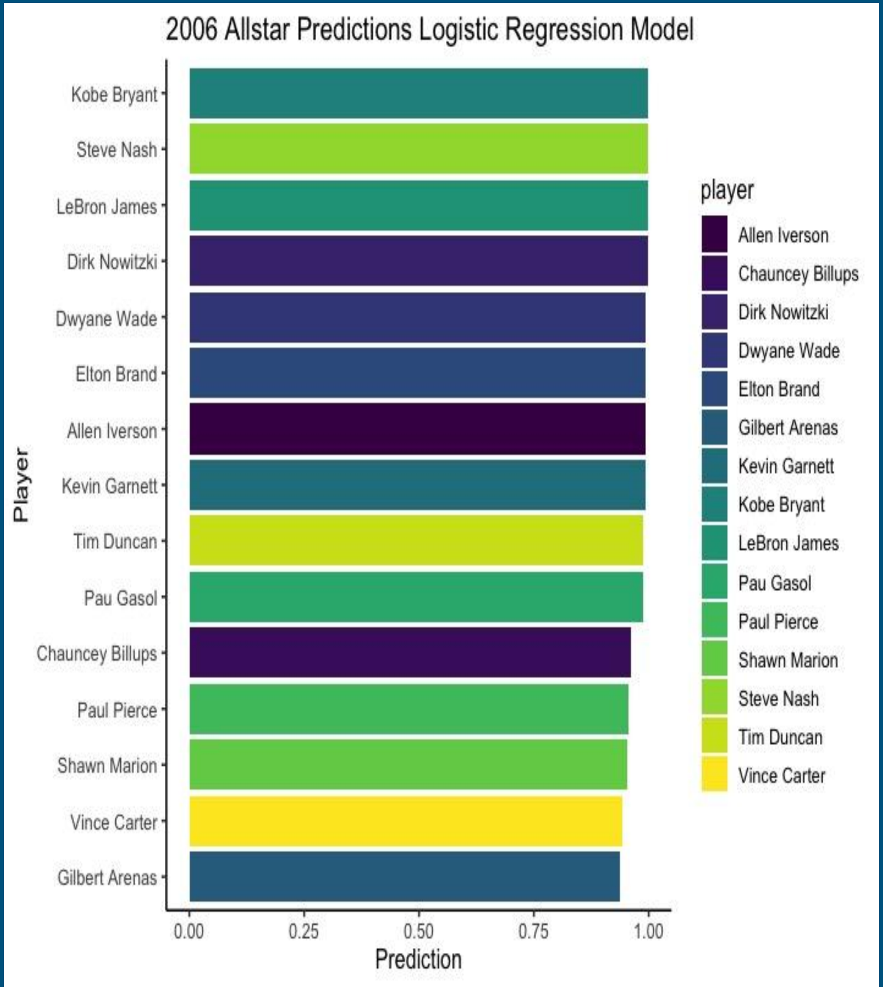
NBA All Star Predictions

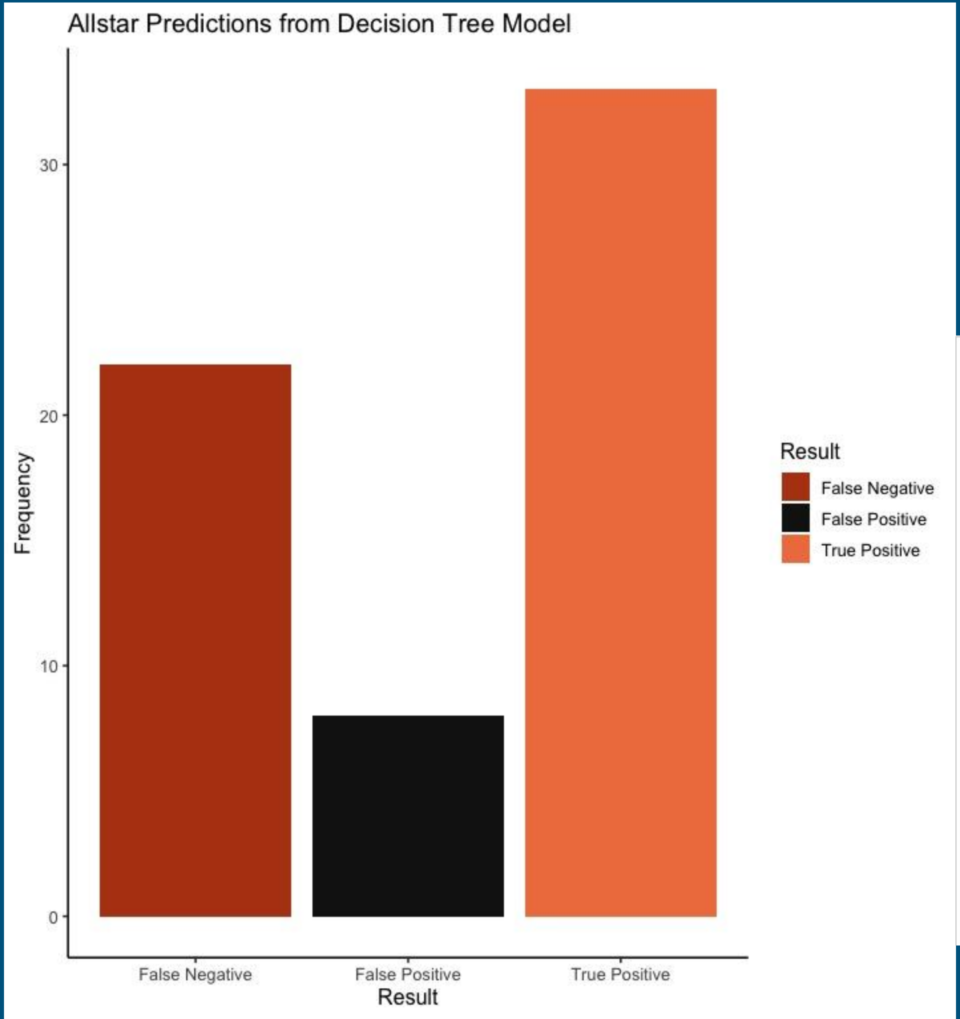
Ian Bogley
Jonas Bowman

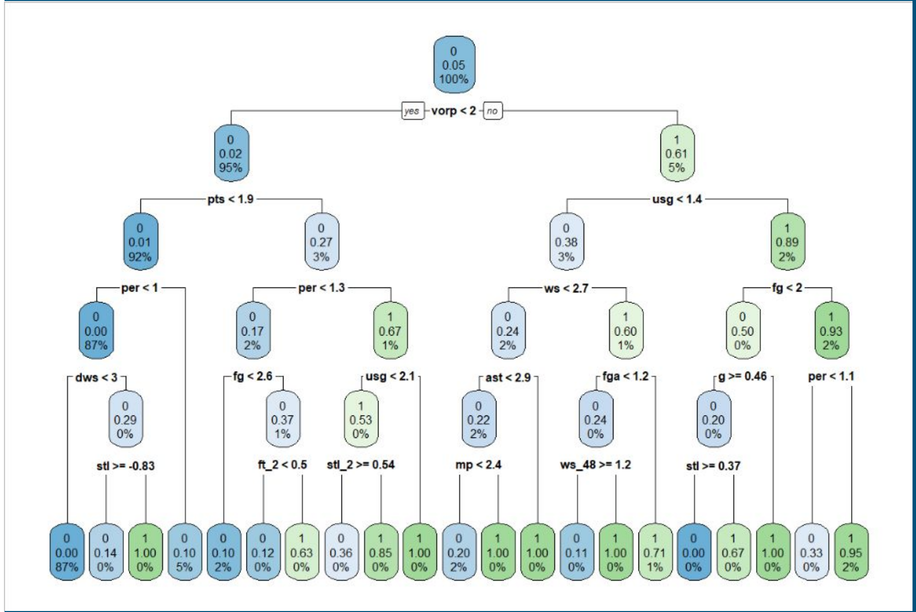

Allstar Predictions from Elasticnet Linear Regression Model
Frequency
0
20
40
False Negative
False Positive
True Positive
Result
Result
False Negative
False Positive
True Positive

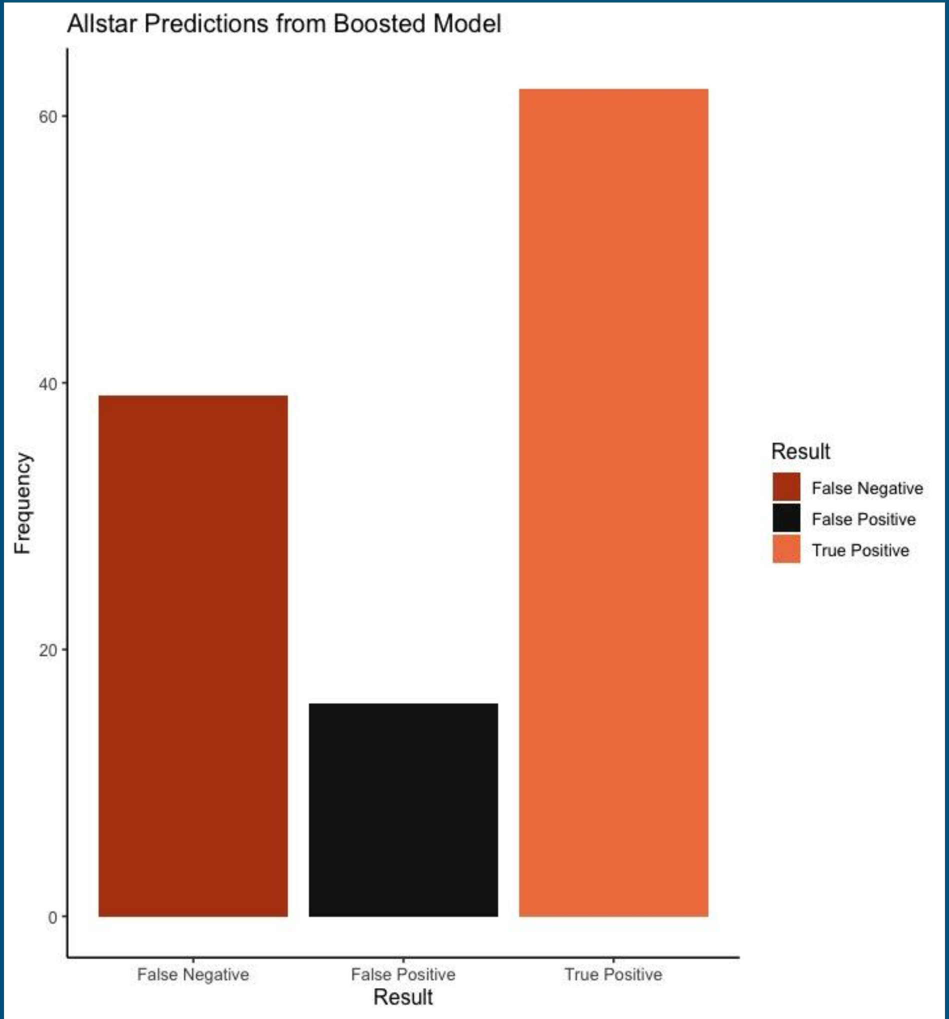

2006 Allstar Predictions Linear Regression Model
Player
LeBron James
Kobe Bryant
Dirk Nowitzki
Dwyane Wade
Allen Iverson
Elton Brand
Kevin Garnett
Pau Gasol
Paul Pierce
Tim Duncan
Gilbert Arenas
Chauncey Billups
Vince Carter
Carmelo Anthony
Michael Redd
0
2
4
6
8
Prediction
player
Chauncey Billups
Michael Redd
Allen Iverson
Pau Gasol
Elton Brand
Vince Carter
Carmelo Anthony
Paul Pierce
Kevin Garnett
Gilbert Arenas
Dwyane Wade
Dirk Nowitzki
Tim Duncan
Kobe Bryant
LeBron James

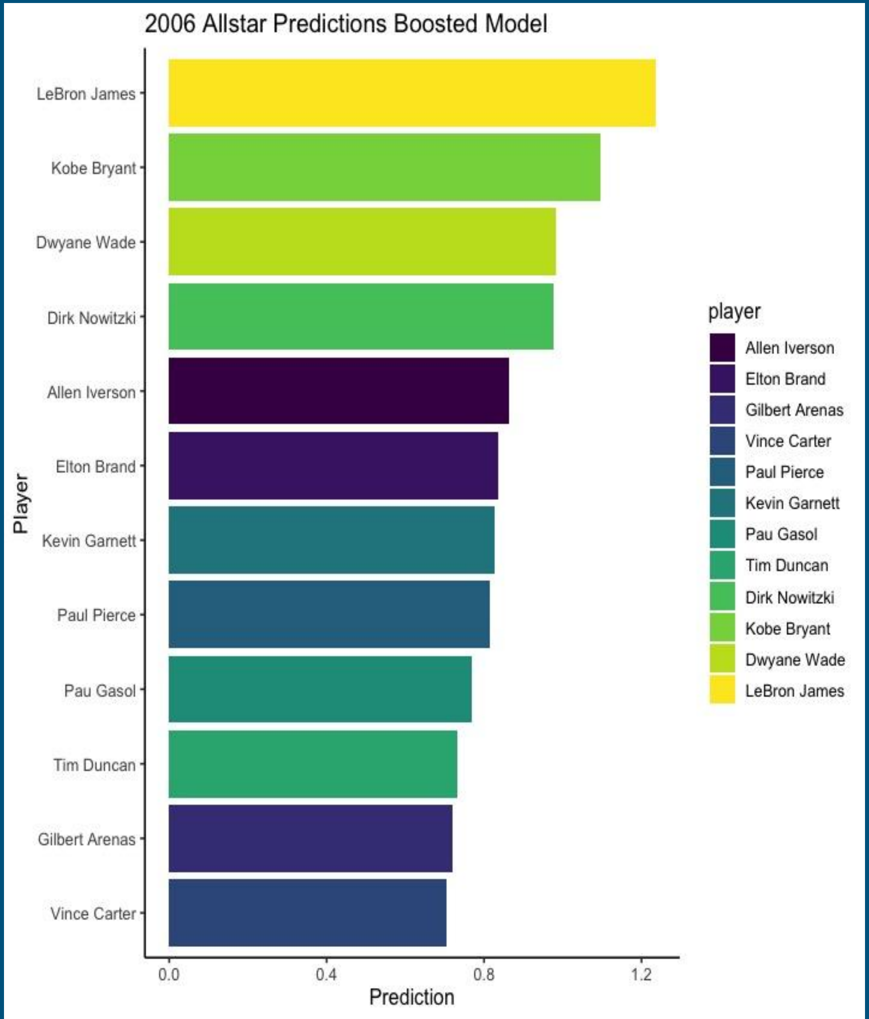

Allstar Predictions from Logistic Model
Frequency
30
20
10
0
False Negative
False Positive
True Positive
Result
Result
False Negative
False Positive
True Positive


2006 Allstar Predictions Logistic Regression Model
Player
Kobe Bryant
Steve Nash
LeBron James
Dirk Nowitzki
Dwyane Wade
Elton Brand
Allen Iverson
Kevin Garnett
Tim Duncan
Pau Gasol
Chauncey Billups
Paul Pierce
Shawn Marion
Vince Carter
Gilbert Arenas
0.00
0.25
0.50
0.75
1.00
Prediction
player
Allen Iverson
Chauncey Billups
Dirk Nowitzki
Dwyane Wade
Elton Brand
Gilbert Arenas
Kevin Garnett
Kobe Bryant
LeBron James
Pau Gasol
Paul Pierce
Shawn Marion
Steve Nash
Tim Duncan
Vince Carter


Allstar Predictions from Decision Tree Model
Frequency
0
10
20
30
False Negative
False Positive
True Positive
Result
Result
False Negative
False Positive
True Positive


0
0.05
100%
yes
vorp < 2
no
0
0.02
95%
1
0.61
5%
pts < 1.9
usg < 1.4
0
0.01
92%
0
0.27
3%
0
0.38
3%
1
0.89
2%
per < 1
per < 1.3
ws < 2.7
fg < 2
0
0.00
87%
0
0.17
2%
1
0.67
1%
0
0.24
2%
1
0.60
1%
0
0.50
0%
1
0.93
2%
dws < 3
fg < 2.6
usg < 2.1
ast < 2.9
fga < 1.2
g >= 0.46
per < 1.1
0
0.29
0%
0
0.37
1%
1
0.53
0%
0
0.22
2%
0
0.24
0%
0
0.20
0%
stl >= -0.83
ft_2 < 0.5
stl_2 >= 0.54
mp < 2.4
ws_48 >= 1.2
stl >= 0.37


Allstar Predictions from Boosted Model
Frequency
60
40
20
0
False Negative
False Positive
True Positive
Result
Result
False Negative
False Positive
True Positive


2006 Allstar Predictions Boosted Model
Player
LeBron James
Kobe Bryant
Dwyane Wade
Dirk Nowitzki
Allen Iverson
Elton Brand
Kevin Garnett
Paul Pierce
Pau Gasol
Tim Duncan
Gilbert Arenas
Vince Carter
0.0
0.4
0.8
1.2
Prediction
player
Allen Iverson
Elton Brand
Gilbert Arenas
Vince Carter
Paul Pierce
Kevin Garnett
Pau Gasol
Tim Duncan
Dirk Nowitzki
Kobe Bryant
Dwyane Wade
LeBron James

```
###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(ba
se_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()
```