```
library(baseballr)
library(xgboost)
library(stargazer)
library(MLmetrics)
library(pROC)
library(Metrics)
library(modelr)
library(pacman)
library(car)
library(caret)
library(vtable)
library(ggthemes)
library(readr)
library(purrr)
library(nnet)
library(rpart)
library(rpart.plot)
library(readxl)
library(patchwork)
library(tidyr)
library(tidyverse)

SavantData2015 <- read_csv("Documents/SAL 413/Professional
Work/MLB_Job_Apps/SavantData/SavantData2015.csv")
SavantData2016 <- read_csv("Documents/SAL 413/Professional
Work/MLB_Job_Apps/SavantData/SavantData2016.csv")
SavantData2017 <- read_csv("Documents/SAL 413/Professional
Work/MLB_Job_Apps/SavantData/SavantData2017.csv")
SavantData2018 <- read_csv("Documents/SAL 413/Professional
Work/MLB_Job_Apps/SavantData/SavantData2018.csv")
SavantData2019 <- read_csv("Documents/SAL 413/Professional
Work/MLB_Job_Apps/SavantData/SavantData2019.csv")
SavantData2020 <- read_csv("Documents/SAL 413/Professional
Work/MLB_Job_Apps/SavantData/SavantData2020.csv")
SavantData2021 <- read_csv("Documents/SAL 413/Professional
Work/MLB_Job_Apps/SavantData/SavantData2021.csv")
SavantData2022 <- read_csv("Documents/SAL 413/Professional
Work/MLB_Job_Apps/SavantData/SavantData2022.csv")
SavantData2023 <- read_csv("Documents/SAL 413/Professional
Work/MLB_Job_Apps/SavantData/SavantData2023.csv")
SavantData2024 <- read_csv("Documents/SAL 413/Professional
Work/MLB_Job_Apps/SavantData/SavantData2024.csv")

pacman::p_load(tidyverse, modelr, baseballr, gam, randomForest, caret, xgboost, mlr,
ggthemes, e1071, remotes, formatR, Boruta, stringi)

MLB_Data <- rbind(SavantData2015, SavantData2016, SavantData2017, SavantData2018,
SavantData2019, SavantData2020,
                  SavantData2021, SavantData2022, SavantData2023, SavantData2024) %>%
  filter(game_type == "R")

MLB_Data <- MLB_Data %>%
  separate(player_name, into = c("Last", "First"), sep = ", ", remove = FALSE) %>%
  mutate(player_name = paste(First, Last))

#FanGraphs Data: 2015-2024
fg_2024 <- fg_batter_leaders(startseason = 2024, endseason = 2024)
fg_2023 <- fg_batter_leaders(startseason = 2023, endseason = 2023)
fg_2022 <- fg_batter_leaders(startseason = 2022, endseason = 2022)
fg_2021 <- fg_batter_leaders(startseason = 2021, endseason = 2021)
fg_2020 <- fg_batter_leaders(startseason = 2020, endseason = 2020)
fg_2019 <- fg_batter_leaders(startseason = 2019, endseason = 2019)
fg_2018 <- fg_batter_leaders(startseason = 2018, endseason = 2018)
fg_2017 <- fg_batter_leaders(startseason = 2017, endseason = 2017)
```

```
fg_2016 <- fg_batter_leaders(startseason = 2016, endseason = 2016)
fg_2015 <- fg_batter_leaders(startseason = 2015, endseason = 2015)

fg_df <- rbind(fg_2015, fg_2016, fg_2017, fg_2018, fg_2019, fg_2020, fg_2021, fg_2022,
fg_2023, fg_2024, fill = TRUE)

#Steps 1 and 2: Calculate probability of Strike | Taken and Calculate expected runs given
strike and expected runs given ball
batter_data <- MLB_Data %>%
  select(-spin_dir, -spin_rate_deprecated, -break_angle_deprecated,
         -break_length_deprecated, -tfs_deprecated, -tfs_zulu_deprecated,
         -fielder_2, -umpire, -vx0, -vy0, -vz0, -ax, -ay, -az, -fielder_3, -fielder_4, -
fielder_5, -fielder_6, -fielder_7,
         -fielder_8, -fielder_9, -if_fielding_alignment, -of_fielding_alignment)

batter_data <- batter_data %>%
  run_expectancy_code(level = "pitch")

batter_data$description <- as.factor(batter_data$description)

batter_data <- batter_data %>%
  mutate(on_1b = ifelse(!is.na(on_1b), 1, 0), on_2b = ifelse(!is.na(on_2b),
                                                             1, 0), on_3b =
ifelse(!is.na(on_3b), 1, 0))

batter_data <- batter_data %>%
  mutate(outcome = case_when(
    description %in% c("swinging_strike", "swinging_strike_blocked", "foul_tip",
                       "bunt_foul_tip", "missed_bunt") ~ "Miss",
    strikes == 2 & description == "foul_bunt" ~ "Miss",
    description %in% c("foul", "foul_bunt") ~ "Foul",
    events == "single" ~ "Single",
    events == "double" ~ "Double",
    events == "triple" ~ "Triple",
    events == "home_run" ~ "Homerun",
    description == "ball" ~ "Ball",
    description == "called_strike" ~ "Called_Strike",
    TRUE ~ "Out"
  ))

batter_data <- batter_data %>% left_join(fg_df, by = c("player_name" = "PlayerName",
"game_year" = "Season"))

batter_data <- batter_data %>%
  mutate(barrel = ifelse(launch_angle <= 50 & launch_speed >= 98 & launch_speed * 1.5 -
launch_angle >= 117 & launch_speed +
                           launch_angle >= 124, 1, 0))

batter_data <- batter_data %>%
  select(-rDSV, -rBPTV, -rBTeamV, -rBTV, -xMLBAMID, -`K_pct+`, -`BB_pct+`, -CB_pct, -
CH_pct, -CT_pct, -BUH,
         -BUH_pct, -WPA, -WPA_LI, -WPA_minus, -WPA_plus, -Dollars, -bipCount, -rPPTV, -
IFH, -IFH_pct, -wLeague, -Offense,
         -Defense, -CH_pct, -Replacement, -SeasonMax, -SeasonMin, -TTO_pct, -Fielding, -
BaseRunning, -wBsR, -WAROld, -FBv, -CHv,
         -EBV, -ESV, -wKN, -wKN_C, -`pi_CH-X`, -`pi_CH-Z`, -pi_CH_pct, -pi_Contact_pct, -
`pi_CS-X`, -`pi_CS-Z`, -pi_CS_pct,
         -`pi_CU-X`, -`pi_CU-Z`, -pi_CU_pct, -`pi_FA-X`, -`pi_FA-Z`, -pi_FA_pct, -pi_vCH,
-pi_vCS, -pi_vCU, -pi_vFA, -pi_Pace,
         -pfx_wSI_C, -pfx_wSI, -wKN, -wKN_C, -pi_wKN, -wCB, -wCB_C, -WAROld, -Spd, -
hyper_speed, -KN_pct, -`pi_KN-X`, -`pi_KN-Z`,
         -pi_KN_pct, -`HRFB_pct+`, -HR_FB, -`BABIP+`, -`ISO+`, -`Soft_pct+`, -`Med_pct+`,
-`Hard_pct+`, -wCT, -wCT_C, -wLeague,
         -KNv, -rTV, -api_break_z_with_gravity, -api_break_x_arm, -api_break_x_batter_in,
-PH, -pLI, -phLI, -rDGV, -Hard, -Hard_pct,
```

```
        -HardHit, -HardHit_pct, -PPTV, -Cent_pct, -Cent_pct, -`Cent_pct+`, -LD_pct, -
`LD_pct+`,
        -`C+SwStr_pct`, -IFFB, -IFFB_pct, -CTv, -CPTV, -Cent, -Cent_pct, -`Cent_pct+`, -
CS, -CFraming, -fld_score,
        -SLv, -CTv, -RAR, -FB_pct1, -FB_pct, -`FB_pct+`, -`pi_FC-X`, -`pi_FC-Z`, -
pi_FC_pct, -pfx_wCH, -pfx_wFA_C, -pfx_wFA_C,
        -wFB, -XX_pct, -TPA, -Q, -`pfx_FO-X`, -`pfx_FO-Z`, -pfx_FO_pct, -pfx_vFO, -
pfx_wFO, -pfx_wFO_C, -pfx_vSC, -pfx_vSI,
        -pfx_vCH, -pfx_vCU, -pfx_vEP, -pfx_vEP, -pfx_vFA, -pfx_vFC, -pfx_vFO, -pfx_vKC, -
pfx_vSC, -pfx_vSL, -pfx_vSI,
        -pfx_vFS, -BU, -pfx_wKN, -pfx_wKN_C, -fld_score, -post_fld_score, -IFFB_pct, -
IFFB, -IFFB_pct, -pi_wXX, -pi_wXX_C,
        -pfx_wEP_C, -`pfx_CH-X`, -`pfx_CH-Z`, -pfx_CH_pct, -pfx_Contact_pct, -`pfx_CU-X`,
-`pfx_CU-Z`, -pfx_CU_pct,
        -`pfx_EP-X`, -`pfx_EP-Z`, -pfx_EP_pct, -`pfx_FA-X`, -`pfx_FA-Z`, -pfx_FA_pct, -
pfx_vFA, -rCPTV, -wSF, -wSF_C, -Clutch, -SL_pct
        , -`pi_SL-X`, -`pi_SL-Z`, -pi_SL_pct, -pi_vSL, -pi_wSL, -pi_wSL_C, -CBv, -wCH, -
wCH_C, -pi_wCH, -pi_wCH_C, -DGV, -wFB_C,
        -wCH_C, -wCH, -UBR, -pfx_SC_pct, -`pi_SB-X`, -`pi_SB-Z`,
        -pi_SB_pct, -pi_vSB, -pi_wSB, -pi_wSB_C, -REW, -IBB, -pfx_KN_pct, -`pfx_KN-X`, -
`pfx_KN-Z`, -pfx_vKN, -`pi_FS-X`,
        -`pi_FS-Z`, -pi_FS_pct, -pi_vFS, -pi_wFS_C, -`Oppo_pct+`, -`Pull_pct+`, -BPTV, -
`GB_pct+`, -XBR, -pi_XX_pct, -pi_vXX,
        -rCPTV, -`pi_FS-X`, -`pi_FS-Z`, -pi_FS_pct, -`pi_SB-X`, -`pi_SB-Z`, -pi_SB_pct, -
pi_wCH_C, -pi_wCH_C, -pi_wCS,
        -pi_wCS_C, -pi_Zone_pct, -pi_Swing_pct, -`pi_O-Swing_pct`, -`pi_Z-Swing_pct`, -
`pi_SI-X`, -`pi_SI-Z`, -pi_SI_pct, -pi_vFC,
        -pi_vFC, -pi_vKN, -pi_vSI, -pi_wCU, -pi_wCU_C, -pi_wFA, -pi_wFA_C, -pi_wFC, -
pi_wFC_C, -pi_wFS, -pi_wKN_C, -pi_wSI, -pi_wSI_C,
        -`pi_XX-X`, -`pi_XX-Z`, -rFTeamV, -`pfx_FC-X`, -`pfx_FC-Z`, -pfx_FC_pct, -
`pfx_FS-X`, -`pfx_FS-Z`, -pfx_FS_pct, -`pfx_KC-X`,
        -`pfx_KC-Z`, -pfx_KC_pct, -pfx_Pace, -`pfx_SC-X`, -`pfx_SC-Z`, -`pfx_SI-X`, -
`pfx_SI-Z`, -pfx_SI_pct, -`pfx_SL-X`,
        -`pfx_SL-Z`, -pfx_SL_pct, -pfx_wCH_C, -pfx_wSC, -pfx_wSC_C, -pfx_wSL_C, -pfx_wSL,
-SFv, -wSL, -wSL_C, -GDPRuns, -`AVG+`,
        -DSV, -BTV, -TG, -wRAA, -SF_pct, -SF, -pfx_wFA, -pfx_wCU, -pfx_wCU_C, -pfx_wEP, -
pfx_wFA, -pfx_wFC, -pfx_wFC_C,-pfx_wFS,
        -pfx_wFS_C, -pfx_wFS, -pfx_wFS_C, -pfx_wKC, -pfx_wKC_C, -pfx_wEP)

batter_data[which(batter_data$launch_speed_angle == "null"),"launch_speed_angle"] <- NA
batter_data[which(batter_data$estimated_ba_using_speedangle == "null"),
"estimated_ba_using_speedangle"] <- NA
batter_data[which(batter_data$estimated_woba_using_speedangle == "null"),
"estimated_woba_using_speedangle"] <- NA
batter_data[which(batter_data$woba_value == "null"), "woba_value"] <- NA
batter_data[which(batter_data$iso_value == "null"), "iso_value"] <- NA

batter_data$launch_speed_angle <- as.numeric(batter_data$launch_speed_angle)
batter_data$estimated_ba_using_speedangle <-
as.numeric(batter_data$estimated_ba_using_speedangle)
batter_data$estimated_woba_using_speedangle <-
as.numeric(batter_data$estimated_woba_using_speedangle)
batter_data$woba_value <- as.numeric(batter_data$woba_value)
batter_data$iso_value <- as.numeric(batter_data$iso_value)

batter_data$pitch_id <- seq(1, nrow(batter_data))

grouped_stats <- batter_data %>%
  group_by(player_name, zone) %>%
  summarise(barrel_perc = mean(barrel, na.rm = T), avg_launch_angle = mean(launch_angle,
na.rm = T),
            avg_exit_velocity = mean(launch_speed, na.rm = T), avg_launch_speed_angle =
mean(launch_speed_angle, na.rm = T),
            xAVG = mean(estimated_ba_using_speedangle, na.rm = T), xwOBA =
mean(estimated_woba_using_speedangle, na.rm = T),
```

```
            wOBA = mean(woba_value, na.rm = T), ISO = mean(iso_value, na.rm = T))

grouped_pitching_stats <- batter_data %>%
  group_by(pitcher) %>%
  summarise(FF_spin_rate = mean(release_spin_rate[pitch_type == "FF"], na.rm = T),
            CU_spin_rate = mean(release_spin_rate[pitch_type == "CU"], na.rm = T),
            FF_velocity = mean(release_speed[pitch_type == "FF"], na.rm = T),
            BRK_velocity = mean(release_speed[pitch_type == "CU" | pitch_type == "SL"],
na.rm = T),
            overall_spin_rate = mean(release_spin_rate, na.rm = T), overall_velocity =
mean(release_speed, na.rm = T))

batter_data <- batter_data %>% inner_join(grouped_stats, by = c("player_name", "zone"))
batter_data <- batter_data %>% inner_join(grouped_pitching_stats, by = "pitcher")

#Takes Data
takes <- batter_data %>%
  filter(description == "ball" | description == "called_strike",
         !is.na(plate_x) & !is.na(plate_z)) %>%
  mutate(strike = ifelse(description == "called_strike", 1,
                         0), plate_x2 = plate_x^2, plate_z2 = plate_z^2, plate_x3 =
plate_x2 *
           plate_x, plate_z3 = plate_z2 * plate_z, plate_xz = plate_x *
           plate_z, plate_x2z = plate_x^2 * plate_z, plate_xz2 = plate_x *
           plate_z^2)

ball_next <- takes
ball_next <- ball_next %>% mutate(balls_new = ifelse(balls < 3, balls + 1, 0),
                                  strikes_new = ifelse(balls < 3, strikes, 0),
                                  outs_when_up_new = outs_when_up,
                                  on_1b_new = ifelse(balls == 3 & is.na(on_1b), 1, on_1b),
                                  on_2b_new = ifelse(balls == 3 & !is.na(on_1b), 1,
on_2b),
                                  on_3b_new = ifelse(balls == 3 & !is.na(on_1b) &
!is.na(on_2b), 1, on_3b)) %>%
  mutate(count_base_out_state_new = paste(balls_new, "-", strikes_new, ", ",
outs_when_up_new, " outs, ", ifelse(!is.na(.$on_1b_new), "1b", "_"),
ifelse(!is.na(.$on_2b_new), "2b", "_"), ifelse(!is.na(.$on_3b_new), "3b", "_")))

strike_next <- takes
strike_next <- strike_next %>% mutate(balls_new = ifelse(strikes < 2, balls, 0),
                                      strikes_new = ifelse(strikes < 2, strikes + 1, 0),
                                      outs_when_up_new = ifelse(strikes < 2, outs_when_up,
outs_when_up + 1),
                                      on_1b_new =  on_1b,
                                      on_2b_new = on_2b,
                                      on_3b_new = on_3b) %>%
  mutate(count_base_out_state_new = paste(balls_new, "-", strikes_new, ", ",
outs_when_up_new, " outs, ", ifelse(!is.na(.$on_1b_new), "1b", "_"),
ifelse(!is.na(.$on_2b_new), "2b", "_"), ifelse(!is.na(.$on_3b_new), "3b", "_")))

strike_next[which(strike_next$outs_when_up_new == 3), "avg_re.y"] <- 0

takes_ball <- ball_next %>%
  select(pitch_id, balls_new, strikes_new, outs_when_up_new, on_1b_new, on_2b_new,
         on_3b_new, count_base_out_state_new, next_avg_re, avg_re) %>%
  rename(count_base_out_state_ball = count_base_out_state_new, avg_re.ball = next_avg_re)

take_outcomes <- strike_next %>%
  rename(count_base_out_state_orig = count_base_out_state, count_base_out_state_strike =
count_base_out_state_new,
         avg_re.orig = avg_re, avg_re.strike = next_avg_re) %>% left_join(takes_ball, by =
"pitch_id")

takes <- take_outcomes %>% select(plate_x, plate_z, sz_top, sz_bot, strike, plate_x2 ,
```

```
plate_z2 , plate_xz , sz_top , sz_bot)

xgb_takes <- xgb.DMatrix(data = as.matrix(takes))
parameters <- list(objective = "binary:logistic", eta = .3, eval_metric = "auc")
train_takes <- takes %>% dplyr::select(plate_x, plate_z, sz_top, sz_bot, strike)
train_lab <- takes$strike
xgb.train_takes = xgb.DMatrix(data = as.matrix(train_takes[, -5]), label = train_lab)

strike_prob_mod <- xgb.train(params = parameters, data = xgb.train_takes, nrounds = 100,
watchlist = list(val=xgb.train_takes))

takes_preds <- predict(strike_prob_mod, xgb_takes)

takes_preds <- as.data.frame(takes_preds)

take_outcomes <- take_outcomes %>% bind_cols(takes_preds) %>% rename(strike_prob =
takes_preds)

set.seed(632764)

#Strike Prob Model
strike_prob_model <- glm(strike ~ plate_x + plate_z + plate_x2 + plate_z2 + plate_xz +
sz_top + sz_bot, data = takes, family = binomial(link = "logit"))
summary(strike_prob_model)
stargazer(strike_prob_model, type = "text")

take_outcomes <- take_outcomes %>%
  add_predictions(strike_prob_model, var = "strike_prob", type = "response")

take_outcomes <- take_outcomes %>% mutate(re_take = avg_re.ball * (1 - strike_prob) -
avg_re.strike * strike_prob)

#Create a strike heatmap to see the accuracy of the logistic model
x <- seq(-1.5, 1.5, length.out = 100)
y <- seq(0.5, 5, length.out = 100)

preds <- data.frame(plate_x = c(outer(x, y * 0 + 1)), plate_z = c(outer(x * 0 + 1, y)),
                    sz_top = mean(batter_data$sz_top, na.rm = T), sz_bot =
mean(batter_data$sz_bot,
                                                                          na.rm =
T))
preds <- preds %>%
  mutate(plate_x2 = plate_x^2, plate_z2 = plate_z^2, plate_x3 = plate_x2 *
           plate_x, plate_z3 = plate_z2 * plate_z, plate_xz = plate_x *
           plate_z, plate_x2z = plate_x^2 * plate_z, plate_xz2 = plate_x *
           plate_z^2) %>%
  add_predictions(strike_prob_model) %>%
  mutate(strike_prob = 1/(1 + exp(-1 * pred)))

zone_preds <- xgb.DMatrix(data = as.matrix(preds))

topKzone <- mean(MLB_Data$sz_top, na.rm = T)
botKzone <- mean(MLB_Data$sz_bot, na.rm = T)
inKzone <- -0.84
outKzone <- 0.84
kZone <- data.frame(x = c(inKzone, inKzone, outKzone, outKzone,
                          inKzone), y = c(botKzone, topKzone, topKzone, botKzone,
botKzone))

ggplot(kZone, aes(x, y)) +
  geom_tile(data = preds, aes(x = plate_x, y = plate_z, fill = strike_prob)) +
  scale_fill_gradient(low = "blue", high = "red") +
  geom_path(lwd = 1.5, col = "black") +
  coord_fixed() +
  ggtitle("Strike Probabilities: 2015-2024") +
```

```
  labs(fill = "Probabilities", y = "z") +
  theme(plot.title = element_text(hjust = 0.5))

#Step 3: Probabilities given swing (including player quality)
swings <- batter_data %>%
  filter(description != "ball" & description != "called_strike" &
           description != "blocked_ball" & description != "pitchout" &
           description != "hit_by_pitch")

set.seed(632764)
sample <- sample(1:length(swings$game_date), length(swings$game_date) *
                   0.8)
train <- swings[sample, ]
test <- swings[-sample, ]

train <- train %>%
  select(-xAVG.y, -ISO.y, -wOBA.y, -xwOBA.y) %>%
  rename(xAVG = xAVG.x, ISO = ISO.x, wOBA = wOBA.x, xwOBA = xwOBA.x)

test <- test %>%
  select(-xAVG.y, -ISO.y, -wOBA.y, -xwOBA.y) %>%
  rename(xAVG = xAVG.x, ISO = ISO.x, wOBA = wOBA.x, xwOBA = xwOBA.x)

train <- train %>%
  select(plate_x, plate_z, pfx_x, pfx_z, pitch_type, balls, strikes, outs_when_up,
release_spin_rate, stand, p_throws,
         effective_speed, release_speed, PA, AVG, OBP, SLG, barrel_perc,
avg_exit_velocity, avg_launch_angle, xAVG,
         xwOBA, wOBA, ISO, GB_FB, K_pct, overall_spin_rate, overall_velocity, outcome)

test <- test %>%
  select(plate_x, plate_z, pfx_x, pfx_z, pitch_type, balls, strikes, outs_when_up,
release_spin_rate, stand, p_throws,
         effective_speed, release_speed, PA, AVG, OBP, SLG, barrel_perc,
avg_exit_velocity, avg_launch_angle, xAVG,
         xwOBA, wOBA, ISO, GB_FB, K_pct, overall_spin_rate, overall_velocity, outcome)

train <- train %>% filter(!is.na(train$outcome))
test <- test %>% filter(!is.na(test$outcome))
train_lab <- as.factor(train$outcome)
test_lab <- as.factor(test$outcome)
final_lab <- levels(train_lab)

#Probabilities of Events
train$pitch_type <- as.numeric(as.factor(train$pitch_type)) -1
train$p_throws <- as.numeric(as.factor(train$p_throws)) - 1
train$stand <- as.numeric(as.factor(train$stand)) - 1
test$pitch_type <- as.numeric(as.factor(test$pitch_type)) - 1
test$p_throws <- as.numeric(as.factor(test$p_throws)) - 1
test$stand <- as.numeric(as.factor(test$stand)) - 1
train_lab <- as.numeric(as.factor(train_lab)) - 1
test_lab <- as.numeric(as.factor(test_lab)) - 1
xgb.train = xgb.DMatrix(data = as.matrix(train[, -29]), label = train_lab)
xgb.test = xgb.DMatrix(data = as.matrix(test[, -29]), label = test_lab)

params <- list(objective = "multi:softprob", num_class = 7, eta = 0.3,
               min_child_weight = 6.47, max_depth = 9, subsample = 0.837,
               colsample_bytree = 0.75, eval_metric = "merror", tree_method = "hist")
swing_mod <- xgb.train(params = params, data = xgb.train, nrounds = 100)
swing_mod

xgb.pred = predict(swing_mod, xgb.test, reshape = T)
xgb.pred <- matrix(xgb.pred, ncol = length(final_lab))
xgb.pred <- xgb.pred[1:length(test$outcome), ]
```

```
xgb.pred = as.data.frame(xgb.pred)
xgb.pred$label <- test$outcome
names(xgb.pred) <- levels(test$outcome)

swings_outcome_subset <- swings$outcome[-sample][1:nrow(xgb.pred)]
xgb.pred$label <- swings_outcome_subset

colnames(xgb.pred)[1:7] <- final_lab

table(xgb.pred$label)
colnames(xgb.pred) <- make.names(colnames(xgb.pred), unique = TRUE)
colnames(xgb.pred)
xgb.pred <- xgb.pred %>%
  select(-any_of(c("X")))

xgb_summary <- xgb.pred %>%
  group_by(label) %>%
  summarize(
    meanMiss = mean(Miss, na.rm = TRUE),
    meanOut = mean(Out, na.rm = TRUE),
    meanSingle = mean(Single, na.rm = TRUE),
    meanDouble = mean(Double, na.rm = TRUE),
    meanTriple = mean(Triple, na.rm = TRUE),
    meanHR = mean(Homerun, na.rm = TRUE),
    meanFoul = mean(Foul, na.rm = TRUE)
  )
print(xgb_summary)

xgb.pred %>%
  filter(xgb.pred$Homerun > 0.0779) %>%
  select(label) %>%
  table() %>%
  prop.table()

xgb.pred %>%
  select(label) %>%
  table() %>%
  prop.table()

#Step 4: Expected Runs Agnostic Player Quality
#Decision Tree for Take
take_outcomes <- take_outcomes %>% filter(!is.na(change_re))

set.seed(123)
dt_train_index <- createDataPartition(take_outcomes$change_re, p = 0.8, list = FALSE)
dt_train_data <- take_outcomes[dt_train_index, ]
dt_test_data <- take_outcomes[-dt_train_index, ]

dt_train_data <- dt_train_data %>% select(plate_x, plate_z, balls, strikes, change_re)
dt_test_data <- dt_test_data %>% select(plate_x, plate_z, balls, strikes, change_re)

tree_model_take <- rpart(change_re ~ .,
                         data = dt_train_data,
                         method = "anova",
                         control = rpart.control(minsplit = 10, cp = 0.01))

rpart.plot(tree_model_take, box.palette = "auto", nn = TRUE)

predictions <- predict(tree_model_take, dt_test_data, type = "vector")

rmse_val <- RMSE(predictions, dt_test_data$change_re)
r2_val <- R2(predictions, dt_test_data$change_re)
cat("RMSE:", round(rmse_val, 3), "\nR-squared:", round(r2_val, 3), "\n")

printcp(tree_model_take)
```

```
optimal_cp <- tree_model_take$cptable[which.min(tree_model_take$cptable[, "xerror"]),
"CP"]
pruned_tree <- prune(tree_model_take, cp = optimal_cp)

rpart.plot(pruned_tree)

take_outcomes$take_pred <- NA
take_outcomes$take_pred <- predict(tree_model_take, newdata = take_outcomes, type =
"vector")

control <- trainControl(method = "cv", number = 10)
tuned_tree <- train(change_re ~ .,
                    data = dt_train_data,
                    method = "rpart",
                    trControl = control,
                    tuneGrid = expand.grid(cp = seq(0.001, 0.05, 0.001)))

print(tuned_tree)

#Expected Run Scale Using Takes
x <- seq(-1.5, 1.5, length.out = 100)
y <- seq(0.5, 5, length.out = 100)

preds_takes <- data.frame(
  plate_x = c(outer(x, y * 0 + 1)),
  plate_z = c(outer(x * 0 + 1, y)),
  sz_top = mean(take_outcomes$sz_top, na.rm = TRUE),
  sz_bot = mean(take_outcomes$sz_bot, na.rm = TRUE)
)

preds_takes <- preds_takes %>%
  mutate(
    balls = sample(0:3, nrow(preds_takes), replace = TRUE),
    strikes = sample(0:2, nrow(preds_takes), replace = TRUE),
    plate_x2 = plate_x^2,
    plate_z2 = plate_z^2,
    plate_x3 = plate_x2 * plate_x,
    plate_z3 = plate_z2 * plate_z,
    plate_xz = plate_x * plate_z,
    plate_x2z = plate_x^2 * plate_z,
    plate_xz2 = plate_x * plate_z^2,

    balls_cat = case_when(
      balls == 0 ~ "0",
      balls == 1 ~ "1",
      balls == 2 ~ "2",
      balls == 3 ~ "3",
      TRUE ~ as.character(NA)
    ),

    strikes_cat = case_when(
      strikes == 0 ~ "0",
      strikes == 1 ~ "1",
      strikes == 2 ~ "2",
      TRUE ~ as.character(NA)
    )
  ) %>%
  add_predictions(pruned_tree) %>%
  mutate(change_re = pred)

preds_takes <- preds_takes %>%
  filter(!is.na(plate_x) & !is.na(plate_z) & !is.na(change_re))

topKzone <- mean(take_outcomes$sz_top, na.rm = TRUE)
botKzone <- mean(take_outcomes$sz_bot, na.rm = TRUE)
```

```
inKzone <- -0.84
outKzone <- 0.84

kZone <- data.frame(
  x = c(inKzone, inKzone, outKzone, outKzone, inKzone),
  y = c(botKzone, topKzone, topKzone, botKzone, botKzone)
)

ggplot(kZone, aes(x, y)) +
  geom_tile(data = preds_takes, aes(x = plate_x, y = plate_z, fill = change_re),
            width = diff(range(preds_takes$plate_x))/length(x),
            height = diff(range(preds_takes$plate_z))/length(y)) +
  scale_fill_gradient(low = "blue", high = "red") +
  geom_path(lwd = 1.5, col = "black") +
  coord_fixed() +
  ggtitle("DRE for Pitches Taken: 2015-2024") +
  labs(fill = "DRE", y = "Vertical Distance [ft]", x = "Horizontal Distance [ft]") +
  theme(plot.title = element_text(hjust = 0.5))

#Swing Decision Tree
swings <- swings %>% filter(!is.na(change_re))

set.seed(123)
train_index_swing <- createDataPartition(swings$change_re, p = 0.8, list = FALSE)
train_data_swing <- swings[train_index_swing, ]
test_data_swing <- swings[-train_index_swing, ]

train_data_swing <- train_data_swing %>% select(plate_x, plate_z, balls, strikes,
change_re)
test_data_swing <- test_data_swing %>% select(plate_x, plate_z, balls, strikes, change_re)

tree_model_swing <- rpart(change_re ~ .,
                          data = train_data_swing,
                          method = "anova",
                          control = rpart.control(minsplit = 5, cp = 0.001))

rpart.plot(tree_model_swing, box.palette = "auto", nn = TRUE)

swing_predictions <- predict(tree_model_swing, test_data_swing, type = "vector")

swing_rmse_val <- RMSE(swing_predictions, test_data_swing$change_re)
swing_r2_val <- R2(swing_predictions, test_data_swing$change_re)
cat("RMSE:", round(swing_rmse_val, 3), "\nR-squared:", round(swing_r2_val, 3), "\n")

printcp(tree_model_swing)
optimal_cp_swing <- tree_model_swing$cptable[which.min(tree_model_swing$cptable[,
"xerror"]), "CP"]
swing_pruned_tree <- prune(tree_model_swing, cp = optimal_cp_swing)

rpart.plot(swing_pruned_tree)

control_swing <- trainControl(method = "cv", number = 10)
tuned_tree <- train(change_re ~ .,
                    data = dt_train_data,
                    method = "rpart",
                    trControl = control_swing,
                    tuneGrid = expand.grid(cp = seq(0.001, 0.05, 0.001)))

print(tuned_tree)

swings$swing_pred <- NA
swings$swing_pred <- predict(tree_model_swing, newdata = swings, type = "vector")

#Expected Run Scale Using Swings
x <- seq(-1.5, 1.5, length.out = 100)
```

```
y <- seq(0.5, 5, length.out = 100)

preds_strikes <- data.frame(
  plate_x = c(outer(x, y * 0 + 1)),
  plate_z = c(outer(x * 0 + 1, y)),
  sz_top = mean(swings$sz_top, na.rm = TRUE),
  sz_bot = mean(swings$sz_bot, na.rm = TRUE)
)

preds_strikes <- preds_strikes %>%
  mutate(
    balls = sample(0:3, nrow(preds_strikes), replace = TRUE),
    strikes = sample(0:2, nrow(preds_strikes), replace = TRUE),
    plate_x2 = plate_x^2,
    plate_z2 = plate_z^2,
    plate_x3 = plate_x2 * plate_x,
    plate_z3 = plate_z2 * plate_z,
    plate_xz = plate_x * plate_z,
    plate_x2z = plate_x^2 * plate_z,
    plate_xz2 = plate_x * plate_z^2,

    balls_cat = case_when(
      balls == 0 ~ "0",
      balls == 1 ~ "1",
      balls == 2 ~ "2",
      balls == 3 ~ "3",
      TRUE ~ as.character(NA)
    ),

    strikes_cat = case_when(
      strikes == 0 ~ "0",
      strikes == 1 ~ "1",
      strikes == 2 ~ "2",
      TRUE ~ as.character(NA)
    )
  ) %>%
  add_predictions(swing_pruned_tree) %>%
  mutate(change_re = pred)

preds_strikes <- preds_strikes %>%
  filter(!is.na(plate_x) & !is.na(plate_z) & !is.na(change_re))

topKzone <- mean(swings$sz_top, na.rm = TRUE)
botKzone <- mean(swings$sz_bot, na.rm = TRUE)
inKzone <- -0.84
outKzone <- 0.84

kZone <- data.frame(
  x = c(inKzone, inKzone, outKzone, outKzone, inKzone),
  y = c(botKzone, topKzone, topKzone, botKzone, botKzone)
)

ggplot(kZone, aes(x, y)) +
  geom_tile(data = preds_strikes, aes(x = plate_x, y = plate_z, fill = change_re),
            width = diff(range(preds_strikes$plate_x))/length(x),
            height = diff(range(preds_strikes$plate_z))/length(y)) +
  scale_fill_gradient(low = "blue", high = "red") +
  geom_path(lwd = 1.5, col = "black") +
  coord_fixed() +
  ggtitle("DRE for Pitches Swung At: 2015–2024") +
  labs(fill = "DRE", y = "Vertical Distance [ft]", x = "Horizontal Distance [ft]") +
  theme(plot.title = element_text(hjust = 0.5))

#Combined
combined_range <- range(c(preds_takes$change_re, preds_strikes$change_re), na.rm = TRUE)
```

```
p1 <- ggplot(kZone, aes(x, y)) +
  geom_tile(data = preds_takes, aes(x = plate_x, y = plate_z, fill = change_re),
            width = diff(range(preds_takes$plate_x)) / length(x),
            height = diff(range(preds_takes$plate_z)) / length(y),
            raster = TRUE) +
  scale_fill_gradient(low = "blue", high = "red", limits = combined_range) +
  geom_path(lwd = 1.5, col = "black") +
  coord_fixed() +
  ggtitle("DRE for Pitches Taken: 2015–2024") +
  labs(fill = "DRE", y = "Vertical Distance [ft]", x = "Horizontal Distance [ft]") +
  theme(plot.title = element_text(hjust = 0.5))

p2 <- ggplot(kZone, aes(x, y)) +
  geom_tile(data = preds_strikes, aes(x = plate_x, y = plate_z, fill = change_re),
            width = diff(range(preds_strikes$plate_x)) / length(x),
            height = diff(range(preds_strikes$plate_z)) / length(y),
            raster = TRUE) +
  scale_fill_gradient(low = "blue", high = "red", limits = combined_range) +
  geom_path(lwd = 1.5, col = "black") +
  coord_fixed() +
  ggtitle("DRE for Pitches Swung At: 2015–2024") +
  labs(fill = "DRE", y = "Vertical Distance [ft]", x = "Horizontal Distance [ft]") +
  theme(plot.title = element_text(hjust = 0.5))

(p1 + p2) + plot_layout(guides = "collect") & theme(legend.position = "right")

ggsave("dre_plot.png", plot = (p1 + p2) + plot_layout(guides = "collect") &
theme(legend.position = "right"),
       dpi = 300, width = 12, height = 6)

batter_data <- batter_data %>%
  left_join(take_outcomes %>% select(pitch_id, take_pred), by = "pitch_id") %>%
  left_join(swings %>% select(pitch_id, swing_pred), by = "pitch_id")

batter_data <- batter_data %>%
  mutate(
    swing_pred = if_else(
      is.na(swing_pred),
      take_pred - re24,
      swing_pred
    ),
    take_pred = if_else(
      is.na(take_pred),
      swing_pred - re24,
      take_pred
    )
  )

batter_data %>%
  select(pitch_id, outcome, re24, take_pred, swing_pred)

batter_data <- batter_data %>%
  mutate(
    obv = abs(take_pred - swing_pred)
  )

#Step 5: Probabilities of all Outcomes Given Swinging - Traded Df
traded_df <- read_csv("Documents/SAL 413/Professional
Work/MLB_Job_Apps/Thesis/traded_df.csv")

traded_df <- traded_df %>%
  mutate(outcome = case_when(
    description %in% c("swinging_strike", "swinging_strike_blocked", "foul_tip",
                       "bunt_foul_tip", "missed_bunt") ~ "Miss",
```

```
    strikes == 2 & description == "foul_bunt" ~ "Miss",
    description %in% c("foul", "foul_bunt") ~ "Foul",
    events == "single" ~ "Single",
    events == "double" ~ "Double",
    events == "triple" ~ "Triple",
    events == "home_run" ~ "Homerun",
    description == "ball" ~ "Ball",
    description == "called_strike" ~ "Called_Strike",
    TRUE ~ "Out"
  ))

traded_df <- traded_df %>%
  separate(player_name, into = c("Last", "First"), sep = ", ", remove = FALSE) %>%
  mutate(player_name = paste(First, Last))

traded_df$description <- as.factor(traded_df$description)

traded_df <- traded_df %>%
  mutate(on_1b = ifelse(!is.na(on_1b), 1, 0), on_2b = ifelse(!is.na(on_2b),
                                                             1, 0), on_3b =
ifelse(!is.na(on_3b), 1, 0))

traded_df <- traded_df %>% left_join(fg_df, by = c("player_name" = "PlayerName",
"game_year" = "Season"))

traded_df <- traded_df %>%
  mutate(barrel = ifelse(launch_angle <= 50 & launch_speed >= 98 & launch_speed * 1.5 -
launch_angle >= 117 & launch_speed +
                           launch_angle >= 124, 1, 0))

traded_df <- traded_df %>%
  select(-rDSV, -rBPTV, -rBTeamV, -rBTV, -xMLBAMID, -`K_pct+`, -`BB_pct+`, -CB_pct, -
CH_pct, -CT_pct, -BUH,
         -BUH_pct, -WPA, -WPA_LI, -WPA_minus, -WPA_plus, -Dollars, -bipCount, -rPPTV, -
IFH, -IFH_pct, -wLeague, -Offense,
         -Defense, -CH_pct, -Replacement, -SeasonMax, -SeasonMin, -TTO_pct, -Fielding, -
BaseRunning, -wBsR, -WAROld, -FBv, -CHv,
         -EBV, -ESV, -wKN, -wKN_C, -`pi_CH-X`, -`pi_CH-Z`, -pi_CH_pct, -pi_Contact_pct, -
`pi_CS-X`, -`pi_CS-Z`, -pi_CS_pct,
         -`pi_CU-X`, -`pi_CU-Z`, -pi_CU_pct, -`pi_FA-X`, -`pi_FA-Z`, -pi_FA_pct, -pi_vCH,
-pi_vCS, -pi_vCU, -pi_vFA, -pi_Pace,
         -pfx_wSI_C, -pfx_wSI, -wKN, -wKN_C, -pi_wKN, -wCB, -wCB_C, -WAROld, -Spd, -
hyper_speed, -KN_pct, -`pi_KN-X`, -`pi_KN-Z`,
         -pi_KN_pct, -`HRFB_pct+`, -HR_FB, -`BABIP+`, -`ISO+`, -`Soft_pct+`, -`Med_pct+`,
-`Hard_pct+`, -wCT, -wCT_C, -wLeague,
         -KNv, -rTV, -api_break_z_with_gravity, -api_break_x_arm, -api_break_x_batter_in,
-PH, -pLI, -phLI, -rDGV, -Hard, -Hard_pct,
         -HardHit, -HardHit_pct, -PPTV, -Cent_pct, -Cent_pct, -`Cent_pct+`, -LD_pct, -
`LD_pct+`,
         -`C+SwStr_pct`, -IFFB, -IFFB_pct, -CTv, -CPTV, -Cent, -Cent_pct, -`Cent_pct+`, -
CS, -CFraming, -fld_score,
         -SLv, -CTv, -RAR, -FB_pct1, -FB_pct, -`FB_pct+`, -`pi_FC-X`, -`pi_FC-Z`, -
pi_FC_pct, -pfx_wCH, -pfx_wFA_C, -pfx_wFA_C,
         -wFB, -XX_pct, -TPA, -Q, -`pfx_FO-X`, -`pfx_FO-Z`, -pfx_FO_pct, -pfx_vFO, -
pfx_wFO, -pfx_wFO_C, -pfx_vSC, -pfx_vSI,
         -pfx_vCH, -pfx_vCU, -pfx_vEP, -pfx_vEP, -pfx_vFA, -pfx_vFC, -pfx_vFO, -pfx_vKC, -
pfx_vSC, -pfx_vSL, -pfx_vSI,
         -pfx_vFS, -BU, -pfx_wKN, -pfx_wKN_C, -fld_score, -post_fld_score, -IFFB_pct, -
IFFB, -IFFB_pct, -pi_wXX, -pi_wXX_C,
         -pfx_wEP_C, -`pfx_CH-X`, -`pfx_CH-Z`, -pfx_CH_pct, -pfx_Contact_pct, -`pfx_CU-X`,
-`pfx_CU-Z`, -pfx_CU_pct,
         -`pfx_EP-X`, -`pfx_EP-Z`, -pfx_EP_pct, -`pfx_FA-X`, -`pfx_FA-Z`, -pfx_FA_pct, -
pfx_vFA, -rCPTV, -wSF, -wSF_C, -Clutch,
         -SL_pct
         , -`pi_SL-X`, -`pi_SL-Z`, -pi_SL_pct, -pi_vSL, -pi_wSL, -pi_wSL_C, -CBv, -wCH, -
```

```
wCH_C, -pi_wCH, -pi_wCH_C, -DGV, -wFB_C,
         -wCH_C, -wCH, -UBR, -pfx_SC_pct, -`pi_SB-X`, -`pi_SB-Z`,
         -pi_SB_pct, -pi_vSB, -pi_wSB, -pi_wSB_C, -REW, -IBB, -pfx_KN_pct, -`pfx_KN-X`, -
`pfx_KN-Z`, -pfx_vKN, -`pi_FS-X`,
         -`pi_FS-Z`, -pi_FS_pct, -pi_vFS, -pi_wFS_C, -`Oppo_pct+`, -`Pull_pct+`, -BPTV, -
`GB_pct+`, -XBR, -pi_XX_pct, -pi_vXX,
         -rCPTV, -`pi_FS-X`, -`pi_FS-Z`, -pi_FS_pct, -`pi_SB-X`, -`pi_SB-Z`, -pi_SB_pct, -
pi_wCH_C, -pi_wCH_C, -pi_wCS,
         -pi_wCS_C, -pi_Zone_pct, -pi_Swing_pct, -`pi_O-Swing_pct`, -`pi_Z-Swing_pct`, -
`pi_SI-X`, -`pi_SI-Z`, -pi_SI_pct, -pi_vFC,
         -pi_vFC, -pi_vKN, -pi_vSI, -pi_wCU, -pi_wCU_C, -pi_wFA, -pi_wFA_C, -pi_wFC, -
pi_wFC_C, -pi_wFS, -pi_wKN_C, -pi_wSI,
         -pi_wSI_C,
         -`pi_XX-X`, -`pi_XX-Z`, -rFTeamV, -`pfx_FC-X`, -`pfx_FC-Z`, -pfx_FC_pct, -
`pfx_FS-X`, -`pfx_FS-Z`, -pfx_FS_pct, -`pfx_KC-X`,
         -`pfx_KC-Z`, -pfx_KC_pct, -pfx_Pace, -`pfx_SC-X`, -`pfx_SC-Z`, -`pfx_SI-X`, -
`pfx_SI-Z`, -pfx_SI_pct, -`pfx_SL-X`,
         -`pfx_SL-Z`, -pfx_SL_pct, -pfx_wCH_C, -pfx_wSC, -pfx_wSC_C, -pfx_wSL_C, -pfx_wSL,
-SFv, -wSL, -wSL_C, -GDPRuns, -`AVG+`,
         -DSV, -BTV, -TG, -wRAA, -SF_pct, -SF, -pfx_wFA, -pfx_wCU, -pfx_wCU_C, -pfx_wEP, -
pfx_wFA, -pfx_wFC, -pfx_wFC_C,-pfx_wFS,
         -pfx_wFS_C, -pfx_wFS, -pfx_wFS_C, -pfx_wKC, -pfx_wKC_C, -pfx_wEP)

traded_df[which(traded_df$launch_speed_angle == "null"),"launch_speed_angle"] <- NA
traded_df[which(traded_df$estimated_ba_using_speedangle == "null"),
"estimated_ba_using_speedangle"] <- NA
traded_df[which(traded_df$estimated_woba_using_speedangle == "null"),
"estimated_woba_using_speedangle"] <- NA
traded_df[which(traded_df$woba_value == "null"), "woba_value"] <- NA
traded_df[which(traded_df$iso_value == "null"), "iso_value"] <- NA

traded_df$launch_speed_angle <- as.numeric(traded_df$launch_speed_angle)
traded_df$estimated_ba_using_speedangle <-
as.numeric(traded_df$estimated_ba_using_speedangle)
traded_df$estimated_woba_using_speedangle <-
as.numeric(traded_df$estimated_woba_using_speedangle)
traded_df$woba_value <- as.numeric(traded_df$woba_value)
traded_df$iso_value <- as.numeric(traded_df$iso_value)

traded_df$pitch_id <- seq(1, nrow(traded_df))

grouped_stats <- traded_df %>%
  group_by(player_name, zone) %>%
  summarise(barrel_perc = mean(barrel, na.rm = T), avg_launch_angle = mean(launch_angle,
na.rm = T),
            avg_exit_velocity = mean(launch_speed, na.rm = T), avg_launch_speed_angle =
mean(launch_speed_angle, na.rm = T),
            xAVG = mean(estimated_ba_using_speedangle, na.rm = T), xwOBA =
mean(estimated_woba_using_speedangle, na.rm = T),
            wOBA = mean(woba_value, na.rm = T), ISO = mean(iso_value, na.rm = T))

grouped_pitching_stats <- traded_df %>%
  group_by(pitcher) %>%
  summarise(FF_spin_rate = mean(release_spin_rate[pitch_type == "FF"], na.rm = T),
            CU_spin_rate = mean(release_spin_rate[pitch_type == "CU"], na.rm = T),
            FF_velocity = mean(release_speed[pitch_type == "FF"], na.rm = T),
            BRK_velocity = mean(release_speed[pitch_type == "CU" | pitch_type == "SL"],
na.rm = T),
            overall_spin_rate = mean(release_spin_rate, na.rm = T), overall_velocity =
mean(release_speed, na.rm = T))

traded_df <- traded_df %>% inner_join(grouped_stats, by = c("player_name", "zone"))
traded_df <- traded_df %>% inner_join(grouped_pitching_stats, by = "pitcher")

#Takes Data
```

```
takes <- traded_df %>%
  filter(description == "ball" | description == "called_strike",
         !is.na(plate_x) & !is.na(plate_z)) %>%
  mutate(strike = ifelse(description == "called_strike", 1,
                         0), plate_x2 = plate_x^2, plate_z2 = plate_z^2, plate_x3 =
plate_x2 *
           plate_x, plate_z3 = plate_z2 * plate_z, plate_xz = plate_x *
           plate_z, plate_x2z = plate_x^2 * plate_z, plate_xz2 = plate_x *
           plate_z^2)

ball_next <- takes
ball_next <- ball_next %>% mutate(balls_new = ifelse(balls < 3, balls + 1, 0),
                                  strikes_new = ifelse(balls < 3, strikes, 0),
                                  outs_when_up_new = outs_when_up,
                                  on_1b_new = ifelse(balls == 3 & is.na(on_1b), 1, on_1b),
                                  on_2b_new = ifelse(balls == 3 & !is.na(on_1b), 1,
on_2b),
                                  on_3b_new = ifelse(balls == 3 & !is.na(on_1b) &
!is.na(on_2b), 1, on_3b)) %>%
  mutate(count_base_out_state_new = paste(balls_new, "-", strikes_new, ", ",
outs_when_up_new, " outs, ", ifelse(!is.na(.$on_1b_new), "1b", "_"),
ifelse(!is.na(.$on_2b_new), "2b", "_"), ifelse(!is.na(.$on_3b_new), "3b", "_")))

strike_next <- takes
strike_next <- strike_next %>% mutate(balls_new = ifelse(strikes < 2, balls, 0),
                                      strikes_new = ifelse(strikes < 2, strikes + 1, 0),
                                      outs_when_up_new = ifelse(strikes < 2, outs_when_up,
outs_when_up + 1),
                                      on_1b_new =  on_1b,
                                      on_2b_new = on_2b,
                                      on_3b_new = on_3b) %>%
  mutate(count_base_out_state_new = paste(balls_new, "-", strikes_new, ", ",
outs_when_up_new, " outs, ", ifelse(!is.na(.$on_1b_new), "1b", "_"),
ifelse(!is.na(.$on_2b_new), "2b", "_"), ifelse(!is.na(.$on_3b_new), "3b", "_")))

strike_next[which(strike_next$outs_when_up_new == 3), "avg_re.y"] <- 0

takes_ball <- ball_next %>%
  select(pitch_id, balls_new, strikes_new, outs_when_up_new, on_1b_new, on_2b_new,
         on_3b_new, count_base_out_state_new, next_avg_re, avg_re) %>%
  rename(count_base_out_state_ball = count_base_out_state_new, avg_re.ball = next_avg_re)

take_outcomes <- strike_next %>%
  rename(count_base_out_state_orig = count_base_out_state, count_base_out_state_strike =
count_base_out_state_new,
         avg_re.orig = avg_re, avg_re.strike = next_avg_re) %>% left_join(takes_ball, by =
"pitch_id")

takes <- take_outcomes %>% select(plate_x, plate_z, sz_top, sz_bot, strike, plate_x2 ,
plate_z2 , plate_xz , sz_top , sz_bot)

xgb_takes <- xgb.DMatrix(data = as.matrix(takes))
parameters <- list(objective = "binary:logistic", eta = .3, eval_metric = "auc")
train_takes <- takes %>% dplyr::select(plate_x, plate_z, sz_top, sz_bot, strike)
train_lab <- takes$strike
xgb.train_takes = xgb.DMatrix(data = as.matrix(train_takes[, -5]), label = train_lab)

strike_prob_mod <- xgb.train(params = parameters, data = xgb.train_takes, nrounds = 100,
watchlist = list(val=xgb.train_takes))

takes_preds <- predict(strike_prob_mod, xgb_takes)

takes_preds <- as.data.frame(takes_preds)

take_outcomes <- take_outcomes %>% bind_cols(takes_preds) %>% rename(strike_prob =
```

```
takes_preds)

set.seed(632764)

#Strike Prob Model
strike_prob_model <- glm(strike ~ plate_x + plate_z + plate_x2 + plate_z2 + plate_xz +
sz_top + sz_bot, data = takes, family = binomial(link = "logit"))
summary(strike_prob_model)
stargazer(strike_prob_model, type = "text")

take_outcomes <- take_outcomes %>%
  add_predictions(strike_prob_model, var = "strike_prob", type = "response")

take_outcomes <- take_outcomes %>% mutate(re_take = avg_re.ball * (1 - strike_prob) -
avg_re.strike * strike_prob)

#Create a strike heatmap to see the accuracy of the logistic model
x <- seq(-1.5, 1.5, length.out = 100)
y <- seq(0.5, 5, length.out = 100)

preds <- data.frame(plate_x = c(outer(x, y * 0 + 1)), plate_z = c(outer(x * 0 + 1, y)),
                    sz_top = mean(traded_df$sz_top, na.rm = T), sz_bot =
mean(traded_df$sz_bot,
                                                                              na.rm = T))
preds <- preds %>%
  mutate(plate_x2 = plate_x^2, plate_z2 = plate_z^2, plate_x3 = plate_x2 *
           plate_x, plate_z3 = plate_z2 * plate_z, plate_xz = plate_x *
           plate_z, plate_x2z = plate_x^2 * plate_z, plate_xz2 = plate_x *
           plate_z^2) %>%
  add_predictions(strike_prob_model) %>%
  mutate(strike_prob = 1/(1 + exp(-1 * pred)))

zone_preds <- xgb.DMatrix(data = as.matrix(preds))

topKzone <- mean(MLB_Data$sz_top, na.rm = T)
botKzone <- mean(MLB_Data$sz_bot, na.rm = T)
inKzone <- -0.84
outKzone <- 0.84
kZone <- data.frame(x = c(inKzone, inKzone, outKzone, outKzone,
                          inKzone), y = c(botKzone, topKzone, topKzone, botKzone,
botKzone))

ggplot(kZone, aes(x, y)) +
  geom_tile(data = preds, aes(x = plate_x, y = plate_z, fill = strike_prob)) +
  scale_fill_gradient(low = "blue", high = "red") +
  geom_path(lwd = 1.5, col = "black") +
  coord_fixed() +
  ggtitle("Strike Probabilities: 2015-2024") +
  labs(fill = "Probabilities", y = "z") +
  theme(plot.title = element_text(hjust = 0.5))

#Step 3: Probabilities given swing (including player quality)
swings <- traded_df %>%
  filter(description != "ball" & description != "called_strike" &
           description != "blocked_ball" & description != "pitchout" &
           description != "hit_by_pitch")

set.seed(632764)
sample <- sample(1:length(swings$game_date), length(swings$game_date) *
                   0.8)
train <- swings[sample, ]
test <- swings[-sample, ]

train <- train %>%
  select(-xAVG.y, -ISO.y, -wOBA.y, -xwOBA.y) %>%
```

```
  rename(xAVG = xAVG.x, ISO = ISO.x, wOBA = wOBA.x, xwOBA = xwOBA.x)

test <- test %>%
  select(-xAVG.y, -ISO.y, -wOBA.y, -xwOBA.y) %>%
  rename(xAVG = xAVG.x, ISO = ISO.x, wOBA = wOBA.x, xwOBA = xwOBA.x)

train <- train %>%
  select(plate_x, plate_z, pfx_x, pfx_z, pitch_type, balls, strikes, outs_when_up,
release_spin_rate, stand, p_throws,
         effective_speed, release_speed, PA, AVG, OBP, SLG, barrel_perc,
avg_exit_velocity, avg_launch_angle, xAVG,
         xwOBA, wOBA, ISO, GB_FB, K_pct, overall_spin_rate, overall_velocity, outcome)

test <- test %>%
  select(plate_x, plate_z, pfx_x, pfx_z, pitch_type, balls, strikes, outs_when_up,
release_spin_rate, stand, p_throws,
         effective_speed, release_speed, PA, AVG, OBP, SLG, barrel_perc,
avg_exit_velocity, avg_launch_angle, xAVG,
         xwOBA, wOBA, ISO, GB_FB, K_pct, overall_spin_rate, overall_velocity, outcome)

train <- train %>% filter(!is.na(train$outcome))
test <- test %>% filter(!is.na(test$outcome))
train_lab <- as.factor(train$outcome)
test_lab <- as.factor(test$outcome)
final_lab <- levels(train_lab)

#Probabilities of Events
train$pitch_type <- as.numeric(as.factor(train$pitch_type)) -1
train$p_throws <- as.numeric(as.factor(train$p_throws)) - 1
train$stand <- as.numeric(as.factor(train$stand)) - 1
test$pitch_type <- as.numeric(as.factor(test$pitch_type)) - 1
test$p_throws <- as.numeric(as.factor(test$p_throws)) - 1
test$stand <- as.numeric(as.factor(test$stand)) - 1
train_lab <- as.numeric(as.factor(train_lab)) - 1
test_lab <- as.numeric(as.factor(test_lab)) - 1
xgb.train = xgb.DMatrix(data = as.matrix(train[, -29]), label = train_lab)
xgb.test = xgb.DMatrix(data = as.matrix(test[, -29]), label = test_lab)

params <- list(objective = "multi:softprob", num_class = 7, eta = 0.3,
               min_child_weight = 6.47, max_depth = 9, subsample = 0.837,
               colsample_bytree = 0.75, eval_metric = "merror", tree_method = "hist")
swing_mod <- xgb.train(params = params, data = xgb.train, nrounds = 100)
swing_mod

xgb.pred = predict(swing_mod, xgb.test, reshape = T)
xgb.pred <- matrix(xgb.pred, ncol = length(final_lab))
xgb.pred <- xgb.pred[1:length(test$outcome), ]

xgb.pred = as.data.frame(xgb.pred)
xgb.pred$label <- test$outcome
names(xgb.pred) <- levels(test$outcome)

swings_outcome_subset <- swings$outcome[-sample][1:nrow(xgb.pred)]
xgb.pred$label <- swings_outcome_subset

colnames(xgb.pred)[1:7] <- final_lab

table(xgb.pred$label)
colnames(xgb.pred) <- make.names(colnames(xgb.pred), unique = TRUE)
colnames(xgb.pred)
xgb.pred <- xgb.pred %>%
  select(-any_of(c("X")))

xgb_summary <- xgb.pred %>%
  group_by(label) %>%
```

```
  summarize(
    meanMiss = mean(Miss, na.rm = TRUE),
    meanOut = mean(Out, na.rm = TRUE),
    meanSingle = mean(Single, na.rm = TRUE),
    meanDouble = mean(Double, na.rm = TRUE),
    meanTriple = mean(Triple, na.rm = TRUE),
    meanHR = mean(Homerun, na.rm = TRUE),
    meanFoul = mean(Foul, na.rm = TRUE)
  )
print(xgb_summary)

xgb.pred %>%
  filter(xgb.pred$Homerun > 0.0779) %>%
  select(label) %>%
  table() %>%
  prop.table()

xgb.pred %>%
  select(label) %>%
  table() %>%
  prop.table()

#Step 4: Expected Runs Agnostic Player Quality
#Decision Tree for Take
take_outcomes <- take_outcomes %>% filter(!is.na(change_re))

set.seed(123)
dt_train_index <- createDataPartition(take_outcomes$change_re, p = 0.8, list = FALSE)
dt_train_data <- take_outcomes[dt_train_index, ]
dt_test_data <- take_outcomes[-dt_train_index, ]

dt_train_data <- dt_train_data %>% select(plate_x, plate_z, balls, strikes, change_re)
dt_test_data <- dt_test_data %>% select(plate_x, plate_z, balls, strikes, change_re)

tree_model_take <- rpart(change_re ~ .,
                         data = dt_train_data,
                         method = "anova",
                         control = rpart.control(minsplit = 10, cp = 0.01))

rpart.plot(tree_model_take, box.palette = "auto", nn = TRUE)

predictions <- predict(tree_model_take, dt_test_data, type = "vector")

rmse_val <- RMSE(predictions, dt_test_data$change_re)
r2_val <- R2(predictions, dt_test_data$change_re)
cat("RMSE:", round(rmse_val, 3), "\nR-squared:", round(r2_val, 3), "\n")

printcp(tree_model_take)
optimal_cp <- tree_model_take$cptable[which.min(tree_model_take$cptable[, "xerror"]),
"CP"]
pruned_tree <- prune(tree_model_take, cp = optimal_cp)

rpart.plot(pruned_tree)

take_outcomes$take_pred <- NA
take_outcomes$take_pred <- predict(tree_model_take, newdata = take_outcomes, type =
"vector")

control <- trainControl(method = "cv", number = 10)
tuned_tree <- train(change_re ~ .,
                    data = dt_train_data,
                    method = "rpart",
                    trControl = control,
                    tuneGrid = expand.grid(cp = seq(0.001, 0.05, 0.001)))
```

```
print(tuned_tree)

#Expected Run Scale Using Takes
x <- seq(-1.5, 1.5, length.out = 100)
y <- seq(0.5, 5, length.out = 100)

preds_takes <- data.frame(
  plate_x = c(outer(x, y * 0 + 1)),
  plate_z = c(outer(x * 0 + 1, y)),
  sz_top = mean(take_outcomes$sz_top, na.rm = TRUE),
  sz_bot = mean(take_outcomes$sz_bot, na.rm = TRUE)
)

preds_takes <- preds_takes %>%
  mutate(
    balls = sample(0:3, nrow(preds_takes), replace = TRUE),
    strikes = sample(0:2, nrow(preds_takes), replace = TRUE),
    plate_x2 = plate_x^2,
    plate_z2 = plate_z^2,
    plate_x3 = plate_x2 * plate_x,
    plate_z3 = plate_z2 * plate_z,
    plate_xz = plate_x * plate_z,
    plate_x2z = plate_x^2 * plate_z,
    plate_xz2 = plate_x * plate_z^2,

    balls_cat = case_when(
      balls == 0 ~ "0",
      balls == 1 ~ "1",
      balls == 2 ~ "2",
      balls == 3 ~ "3",
      TRUE ~ as.character(NA)
    ),

    strikes_cat = case_when(
      strikes == 0 ~ "0",
      strikes == 1 ~ "1",
      strikes == 2 ~ "2",
      TRUE ~ as.character(NA)
    )
  ) %>%
  add_predictions(pruned_tree) %>%
  mutate(change_re = pred)

preds_takes <- preds_takes %>%
  filter(!is.na(plate_x) & !is.na(plate_z) & !is.na(change_re))

topKzone <- mean(take_outcomes$sz_top, na.rm = TRUE)
botKzone <- mean(take_outcomes$sz_bot, na.rm = TRUE)
inKzone <- -0.84
outKzone <- 0.84

kZone <- data.frame(
  x = c(inKzone, inKzone, outKzone, outKzone, inKzone),
  y = c(botKzone, topKzone, topKzone, botKzone, botKzone)
)

ggplot(kZone, aes(x, y)) +
  geom_tile(data = preds_takes, aes(x = plate_x, y = plate_z, fill = change_re),
            width = diff(range(preds_takes$plate_x))/length(x),
            height = diff(range(preds_takes$plate_z))/length(y)) +
  scale_fill_gradient(low = "blue", high = "red") +
  geom_path(lwd = 1.5, col = "black") +
  coord_fixed() +
  ggtitle("DRE for Pitches Taken: 2015–2024") +
  labs(fill = "DRE", y = "Vertical Distance [ft]", x = "Horizontal Distance [ft]") +
```

```
  theme(plot.title = element_text(hjust = 0.5))

#Swing Decision Tree
swings <- swings %>% filter(!is.na(change_re))

set.seed(123)
train_index_swing <- createDataPartition(swings$change_re, p = 0.8, list = FALSE)
train_data_swing <- swings[train_index_swing, ]
test_data_swing <- swings[-train_index_swing, ]

train_data_swing <- train_data_swing %>% select(plate_x, plate_z, balls, strikes,
change_re)
test_data_swing <- test_data_swing %>% select(plate_x, plate_z, balls, strikes, change_re)

tree_model_swing <- rpart(change_re ~ .,
                          data = train_data_swing,
                          method = "anova",
                          control = rpart.control(minsplit = 5, cp = 0.001))

rpart.plot(tree_model_swing, box.palette = "auto", nn = TRUE)

swing_predictions <- predict(tree_model_swing, test_data_swing, type = "vector")

swing_rmse_val <- RMSE(swing_predictions, test_data_swing$change_re)
swing_r2_val <- R2(swing_predictions, test_data_swing$change_re)
cat("RMSE:", round(swing_rmse_val, 3), "\nR-squared:", round(swing_r2_val, 3), "\n")

printcp(tree_model_swing)
optimal_cp_swing <- tree_model_swing$cptable[which.min(tree_model_swing$cptable[,
"xerror"]), "CP"]
swing_pruned_tree <- prune(tree_model_swing, cp = optimal_cp_swing)

rpart.plot(swing_pruned_tree)

control_swing <- trainControl(method = "cv", number = 10)
tuned_tree <- train(change_re ~ .,
                    data = dt_train_data,
                    method = "rpart",
                    trControl = control_swing,
                    tuneGrid = expand.grid(cp = seq(0.001, 0.05, 0.001)))

print(tuned_tree)

swings$swing_pred <- NA
swings$swing_pred <- predict(tree_model_swing, newdata = swings, type = "vector")

#Expected Run Scale Using Swings
x <- seq(-1.5, 1.5, length.out = 100)
y <- seq(0.5, 5, length.out = 100)

preds_strikes <- data.frame(
  plate_x = c(outer(x, y * 0 + 1)),
  plate_z = c(outer(x * 0 + 1, y)),
  sz_top = mean(swings$sz_top, na.rm = TRUE),
  sz_bot = mean(swings$sz_bot, na.rm = TRUE)
)

preds_strikes <- preds_strikes %>%
  mutate(
    balls = sample(0:3, nrow(preds_strikes), replace = TRUE),
    strikes = sample(0:2, nrow(preds_strikes), replace = TRUE),
    plate_x2 = plate_x^2,
    plate_z2 = plate_z^2,
    plate_x3 = plate_x2 * plate_x,
    plate_z3 = plate_z2 * plate_z,
```

```
    plate_xz = plate_x * plate_z,
    plate_x2z = plate_x^2 * plate_z,
    plate_xz2 = plate_x * plate_z^2,

    balls_cat = case_when(
      balls == 0 ~ "0",
      balls == 1 ~ "1",
      balls == 2 ~ "2",
      balls == 3 ~ "3",
      TRUE ~ as.character(NA)
    ),

    strikes_cat = case_when(
      strikes == 0 ~ "0",
      strikes == 1 ~ "1",
      strikes == 2 ~ "2",
      TRUE ~ as.character(NA)
    )
  ) %>%
  add_predictions(swing_pruned_tree) %>%
  mutate(change_re = pred)

preds_strikes <- preds_strikes %>%
  filter(!is.na(plate_x) & !is.na(plate_z) & !is.na(change_re))

topKzone <- mean(swings$sz_top, na.rm = TRUE)
botKzone <- mean(swings$sz_bot, na.rm = TRUE)
inKzone <- -0.84
outKzone <- 0.84

kZone <- data.frame(
  x = c(inKzone, inKzone, outKzone, outKzone, inKzone),
  y = c(botKzone, topKzone, topKzone, botKzone, botKzone)
)

ggplot(kZone, aes(x, y)) +
  geom_tile(data = preds_strikes, aes(x = plate_x, y = plate_z, fill = change_re),
            width = diff(range(preds_strikes$plate_x))/length(x),
            height = diff(range(preds_strikes$plate_z))/length(y)) +
  scale_fill_gradient(low = "blue", high = "red") +
  geom_path(lwd = 1.5, col = "black") +
  coord_fixed() +
  ggtitle("DRE for Pitches Swung At: 2015–2024") +
  labs(fill = "DRE", y = "Vertical Distance [ft]", x = "Horizontal Distance [ft]") +
  theme(plot.title = element_text(hjust = 0.5))

#Combined
combined_range <- range(c(preds_takes$change_re, preds_strikes$change_re), na.rm = TRUE)

p1 <- ggplot(kZone, aes(x, y)) +
  geom_tile(data = preds_takes, aes(x = plate_x, y = plate_z, fill = change_re),
            width = diff(range(preds_takes$plate_x)) / length(x),
            height = diff(range(preds_takes$plate_z)) / length(y),
            raster = TRUE) +
  scale_fill_gradient(low = "blue", high = "red", limits = combined_range) +
  geom_path(lwd = 1.5, col = "black") +
  coord_fixed() +
  ggtitle("DRE for Pitches Taken: 2015–2024") +
  labs(fill = "DRE", y = "Vertical Distance [ft]", x = "Horizontal Distance [ft]") +
  theme(plot.title = element_text(hjust = 0.5))

p2 <- ggplot(kZone, aes(x, y)) +
  geom_tile(data = preds_strikes, aes(x = plate_x, y = plate_z, fill = change_re),
            width = diff(range(preds_strikes$plate_x)) / length(x),
            height = diff(range(preds_strikes$plate_z)) / length(y),
```

```
            raster = TRUE) +
  scale_fill_gradient(low = "blue", high = "red", limits = combined_range) +
  geom_path(lwd = 1.5, col = "black") +
  coord_fixed() +
  ggtitle("DRE for Pitches Swung At: 2015–2024") +
  labs(fill = "DRE", y = "Vertical Distance [ft]", x = "Horizontal Distance [ft]") +
  theme(plot.title = element_text(hjust = 0.5))

(p1 + p2) + plot_layout(guides = "collect") & theme(legend.position = "right")

ggsave("dre_plot.png", plot = (p1 + p2) + plot_layout(guides = "collect") &
theme(legend.position = "right"),
       dpi = 300, width = 12, height = 6)

traded_df <- traded_df %>%
  left_join(take_outcomes %>% select(pitch_id, take_pred), by = "pitch_id") %>%
  left_join(swings %>% select(pitch_id, swing_pred), by = "pitch_id")

traded_df <- traded_df %>%
  mutate(
    swing_pred = if_else(
      is.na(swing_pred),
      take_pred - re24,
      swing_pred
    ),
    take_pred = if_else(
      is.na(take_pred),
      swing_pred - re24,
      take_pred
    ))

traded_df %>%
  select(pitch_id, outcome, re24, take_pred, swing_pred, obv)

traded_df <- traded_df %>%
  mutate(
    obv = abs(take_pred - swing_pred)
  )

traded_df <- traded_df %>%
  mutate(
    obv_percentiles = percent_rank(obv),
    obv_final = case_when(
      outcome %in% c("Ball", "Called_Strike") & take_pred > swing_pred ~ 1 -
obv_percentiles,
      outcome %in% c("Double", "Foul", "Homerun", "Miss", "Out", "Single", "Triple") &
swing_pred > take_pred ~ 1 - obv_percentiles,
      TRUE ~ -obv_percentiles
    )
  )

traded_summary <- traded_df %>%
  group_by(player_name, game_year, team) %>%
  summarise(
    mean_obv_final = mean(obv_final, na.rm = TRUE),
    first_game_date = min(game_date),
    .groups = "drop"
  )

traded_diff <- traded_summary %>%
  group_by(player_name, game_year) %>%
  arrange(first_game_date) %>%
  summarise(
    obv_diff = last(mean_obv_final) - first(mean_obv_final),
    first_team = first(team),
```

```
    last_team = last(team),
    first_team_mean = first(mean_obv_final),
    last_team_mean = last(mean_obv_final),
    .groups = "drop"
  ) %>%
  mutate()


set.seed(632764)
sample_traded <- sample(1:length(traded_df$game_date), length(traded_df$game_date) * 0.8)
train_traded <- traded_df[sample_traded, ]
test_traded <- traded_df[-sample_traded, ]

train_traded <- train_traded %>%
  select(-xAVG.y, -ISO.y, -wOBA.y, -xwOBA.y) %>%
  rename(xAVG = xAVG.x, ISO = ISO.x, wOBA = wOBA.x, xwOBA = xwOBA.x)

test_traded <- test_traded %>%
  select(-xAVG.y, -ISO.y, -wOBA.y, -xwOBA.y) %>%
  rename(xAVG = xAVG.x, ISO = ISO.x, wOBA = wOBA.x, xwOBA = xwOBA.x)

train_traded <- train_traded %>%
  select(plate_x, plate_z, pfx_x, pfx_z, pitch_type, balls, strikes,
         outs_when_up, release_spin_rate, stand, p_throws, effective_speed,
         release_speed, PA, AVG, OBP, SLG, barrel_perc, avg_exit_velocity,
         avg_launch_angle, xAVG, xwOBA, wOBA, ISO, GB_FB, K_pct, overall_spin_rate,
         overall_velocity, obv, `Z-Contact_pct`, `O-Contact_pct`, EV)

test_traded <- test_traded %>%
  select(plate_x, plate_z, pfx_x, pfx_z, pitch_type, balls, strikes,
         outs_when_up, release_spin_rate, stand, p_throws, effective_speed,
         release_speed, PA, AVG, OBP, SLG, barrel_perc, avg_exit_velocity,
         avg_launch_angle, xAVG, xwOBA, wOBA, ISO, GB_FB, K_pct, overall_spin_rate,
         overall_velocity, obv, `Z-Contact_pct`, `O-Contact_pct`, EV)

train_traded <- train_traded %>% filter(!is.na(obv))
test_traded <- test_traded %>% filter(!is.na(obv))

train_lab <- train_traded$obv
test_lab <- test_traded$obv

train_traded <- train_traded %>% select(-obv)
test_traded <- test_traded %>% select(-obv)

train_traded$pitch_type <- as.numeric(as.factor(train_traded$pitch_type)) -1
train_traded$p_throws <- as.numeric(as.factor(train_traded$p_throws)) - 1
train_traded$stand <- as.numeric(as.factor(train_traded$stand)) - 1
test_traded$pitch_type <- as.numeric(as.factor(test_traded$pitch_type)) - 1
test_traded$p_throws <- as.numeric(as.factor(test_traded$p_throws)) - 1
test_traded$stand <- as.numeric(as.factor(test_traded$stand)) - 1

xgb.train = xgb.DMatrix(data = as.matrix(train_traded), label = train_lab)
xgb.test = xgb.DMatrix(data = as.matrix(test_traded), label = test_lab)

params <- list(
  objective = "reg:squarederror",
  eta = 0.3,
  min_child_weight = 6.47,
  max_depth = 9,
  subsample = 0.837,
  colsample_bytree = 0.75,
  eval_metric = "rmse",
  tree_method = "hist"
)
```

```
obv_mod <- xgb.train(params = params, data = xgb.train, nrounds = 100)

xgb.pred = predict(obv_mod, xgb.test)

xgb.pred <- data.frame(prediction = xgb.pred, actual = test_lab)

xgb_summary <- xgb.pred %>%
  summarize(
    mean_pred = mean(prediction, na.rm = TRUE),
    mean_actual = mean(actual, na.rm = TRUE),
    rmse = sqrt(mean((prediction - actual)^2, na.rm = TRUE))
  )

print(xgb_summary)

summary(traded_diff$obv_diff)

ggplot(traded_diff, aes(x = first_team_mean, y = last_team_mean)) +
  geom_point(alpha = 0.6) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +
  geom_vline(xintercept = mean(traded_diff$first_team_mean, na.rm = TRUE),
             linetype = "dotted", color = "blue") +
  geom_hline(yintercept = mean(traded_diff$last_team_mean, na.rm = TRUE),
             linetype = "dotted", color = "blue") +
  labs(
    title = "Obvious Score: First Team vs. Last Team",
    x = "First Team Mean Obvious Score",
    y = "Last Team Mean Obvious Score"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```