Curveball Whiff Predictor

Zach Landry

2023-11-03

The goal of this exercise is to create two different models that predict the likelihood of a curveball resulting in a whiff using the provided data. A logistic regression and XGBoost model will be implemented to make these predictions, and their accuracies will be compared against each other. First, investigate the shape and the types of variables in the "PitchData.csv" data:

```
# Load tidyverse package
library(tidyverse)

# Load the data
pitch_data <- read.csv("PitchData.csv")

# Filter data to only include curveball pitch types (roughly 11% of total
observations)
pitch_data <- pitch_data %>% filter(Pitch_Type == "Curveball")

# Convert categorical variables from character to factor data types
pitch_data$Pitcher_Throws <- as.factor(pitch_data$Pitcher_Throws)
pitch_data$Batter_Hits <- as.factor(pitch_data$Batter_Hits)
pitch_data$Pitch_Outcome <- as.factor(pitch_data$Pitch_Outcome)
pitch_data$Pitch_Type <- as.factor(pitch_data$Pitch_Type)
```

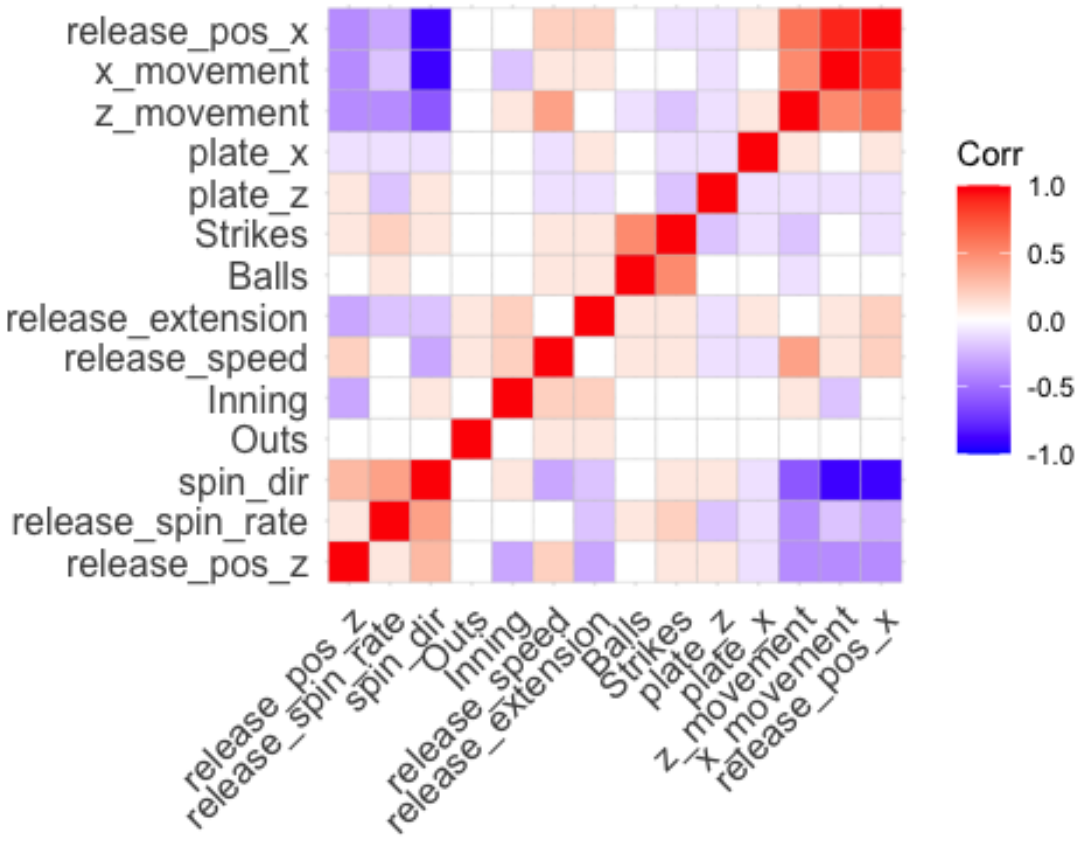
The filtered data including only curveballs has 2582 total observations and 24 variables – both categorical and numeric. There are 6 observations with missing spin rates, and it appears that some other observations have a value of 0 for `release_spin_rate`, `spin_dir`, `release_pos_z`, and `release_extension`. Since these values are not realistic and could skew the results of the model, they will be removed from the data. Additionally, the data requires a binary target variable, called `whiff`, to denote swing-and-misses (1) versus other outcomes (0).

```
# Remove observations with missing values
pitch_data <- na.omit(pitch_data)

# Count and remove observations with a value of 0 for spin rate... 131
observations removed
sum(pitch_data$release_spin_rate==0)
pitch_data <- pitch_data[!(pitch_data$release_spin_rate==0),]

summary(pitch_data)

# Create "whiff" target variable
```

```r
pitch_data$whiff <- ifelse(pitch_data$Pitch_Outcome=="StrikeSwinging", 1, 0)

whiff.percent <- sum(pitch_data$whiff) / nrow(pitch_data)
```

Next, explore the data and look for multicollinearity among the numerical feature variables: `Inning`, `Balls`, `Strikes`, `Outs`, `release_speed`, `x_movement`, `z_movement`, `release_spin_rate`, `spin_dir`, `release_pos_x`, `release_pos_z`, `release_extension`, `plate_x`, and `plate_z`. The relationship between whiffs and pitcher/batter handedness will be explored later in the analysis.

```r
# Select feature variables and create a new dataset with these
features <- c("Inning", "Balls", "Strikes", "Outs", "release_speed",
"x_movement", "z_movement", "release_spin_rate", "spin_dir", "release_pos_x",
"release_pos_z", "release_extension", "plate_x", "plate_z")
feature_data <- pitch_data %>% select(all_of(features))

# Correlation plot
#install.packages("ggcorrplot")
library(ggcorrplot)
corr <- round(cor(feature_data), 1)
ggcorrplot(corr, hc.order = TRUE) # organize by hierarchial clustering
```

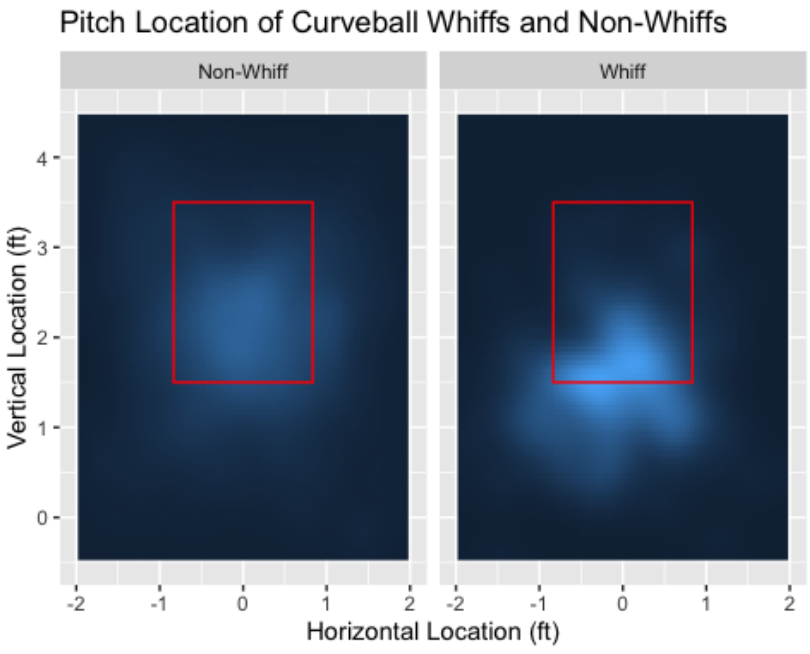


There appears to be a significant negative correlation between spin direction and vertical movement, horizontal movement, and horizontal release point. These relationships make intuitive sense given that a pitch's movement is influenced by its spin, and its spin direction is influenced by the release point. There is also a significant positive correlations between horizontal release point and horizontal movement, as well as moderate correlations between horizontal release point and vertical movement. Hence, `spin_dir` and `release_pos_x` can be dropped from the list of features since they have similar effects on the pitch's movement:

```
# Exclude release_pos_x and spin_dir
features <- features[!(features %in% c("release_pos_x", "spin_dir"))]
feature_data <- feature_data %>% select(all_of(features))
```

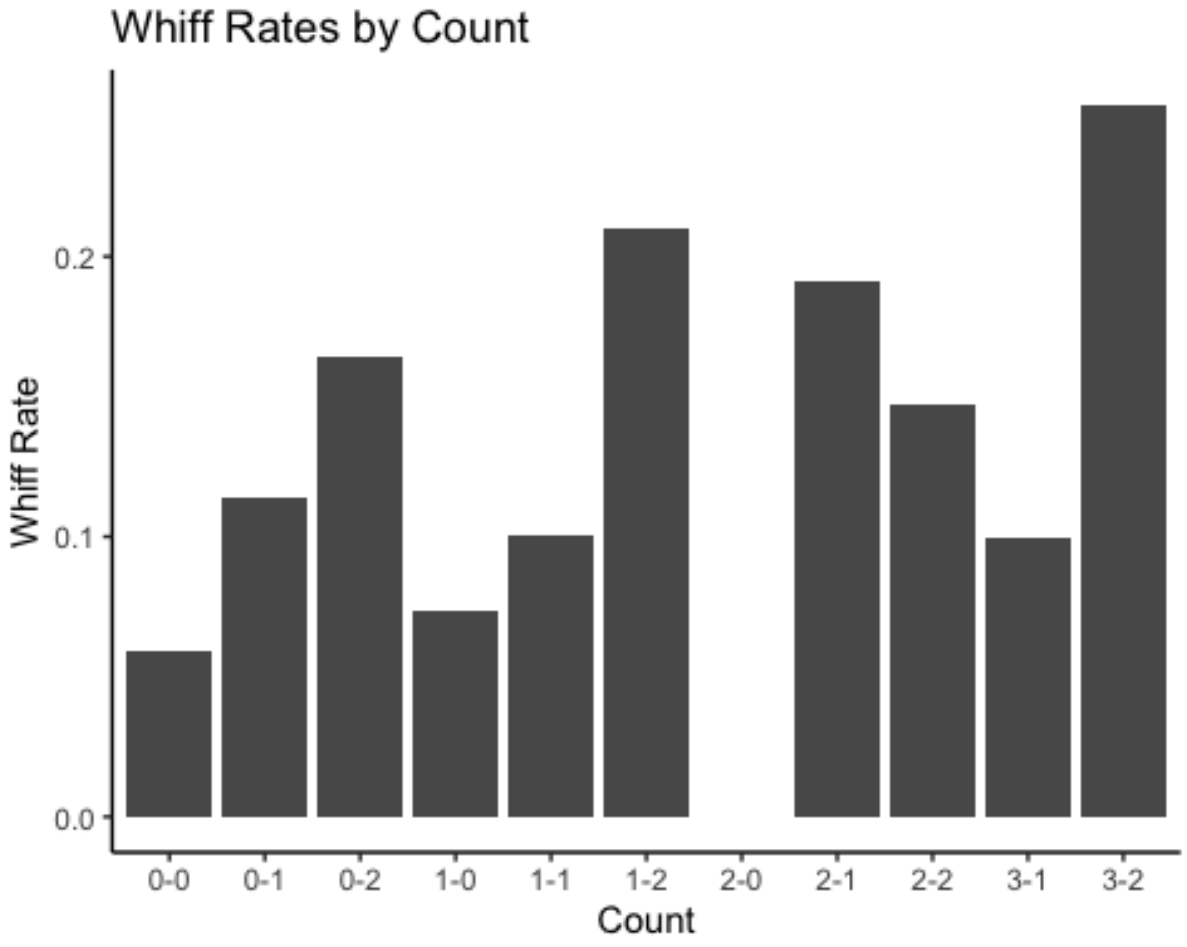
There are several factors that could influence a batter's propensity to swing and miss, namely the pitch's location, the game situation, and pitcher/batter handedness. Using the pitch data containing the target variable `whiff`, let's check for any relationships between whiffs and these factors, starting with pitch location:

```
# Density plot of curveball whiffs by pitch location
ggplot(pitch_data, aes(plate_x, plate_z))+
  stat_density_2d(aes(fill = ..density..), geom = "raster", contour = FALSE)+
  xlim(-2, 2)+
  ylim(-.5, 4.5)+
  theme(legend.position='none')+
  annotate("rect", xmin = -.8333, ymin = 1.5, xmax = .8333, ymax = 3.5, color
= "red", alpha = 0)+
  labs(title = "Pitch Location of Curveball Whiffs and Non-Whiffs", x =
"Horizontal Location (ft)", y = "Vertical Location (ft)")+
  facet_wrap(~whiff, labeller = as_labeller(c('0' = "Non-Whiff", '1' =
"Whiff")))
```

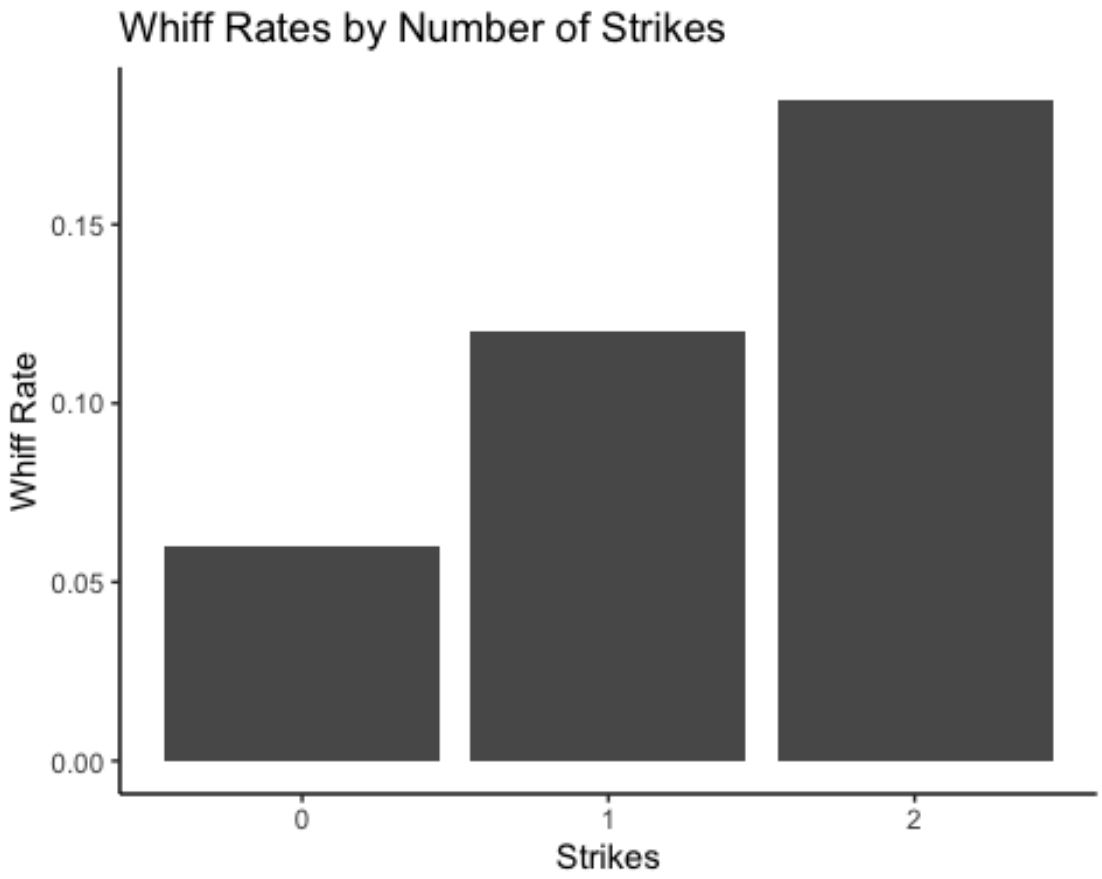


Looking at the 2D density plot above, it seems that more whiffs occur lower in the strike zone, hinting at the fact that pitch location can indeed influence a batter's propensity to whiff. Next, look at whiff rates based on `Count`, a variable that combines the `Balls` and `Strikes` for each pitch:

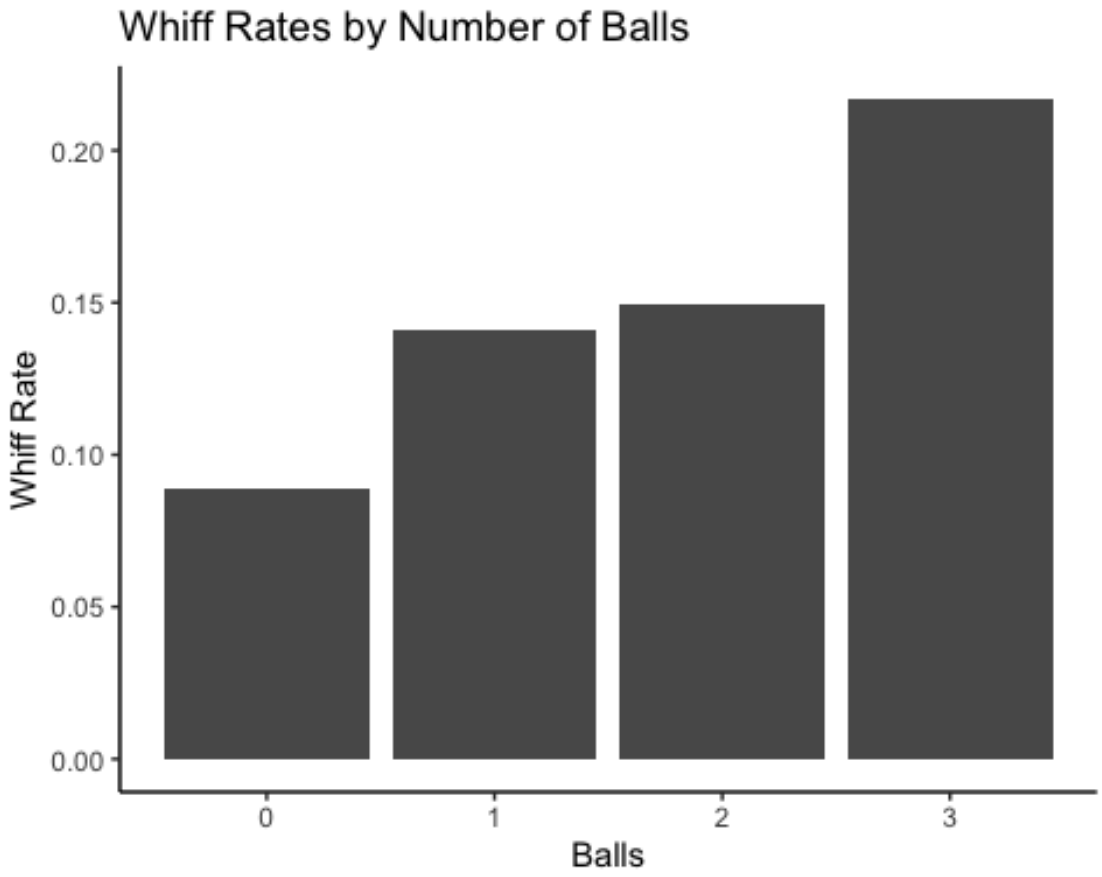
```
# Create new variable called "Count"
pitch_data$Count <- with(pitch_data, paste(Balls, Strikes, sep = "-"))
pitch_data$Count <- as.factor(pitch_data$Count) # convert from character to
factor data type
summary(pitch_data$Count)

# Create bar graph showing the whiff rate for each count
pitch_data %>%
  group_by(Count) %>%
  summarise(whiff_rate = mean(whiff)) %>%
  ggplot(aes(Count, whiff_rate))+
  geom_bar(stat = "identity")+
  theme_classic()+
  labs(title = "Whiff Rates by Count", x = "Count", y = "Whiff Rate")
```



```r
# Create bar graph showing the whiff rate by number of strikes
pitch_data %>%
  group_by(Strikes) %>%
  summarise(whiff_rate = mean(whiff)) %>%
  ggplot(aes(Strikes, whiff_rate))+
  geom_bar(stat = "identity")+
  theme_classic()+
  labs(title = "Whiff Rates by Number of Strikes", x = "Strikes", y = "Whiff
Rate")
```




```r
# Create bar graph showing the whiff rate by number of balls
pitch_data %>%
  group_by(Balls) %>%
  summarise(whiff_rate = mean(whiff)) %>%
  ggplot(aes(Balls, whiff_rate))+
  geom_bar(stat = "identity")+
  theme_classic()+
  labs(title = "Whiff Rates by Number of Balls", x = "Balls", y = "Whiff
Rate")
```

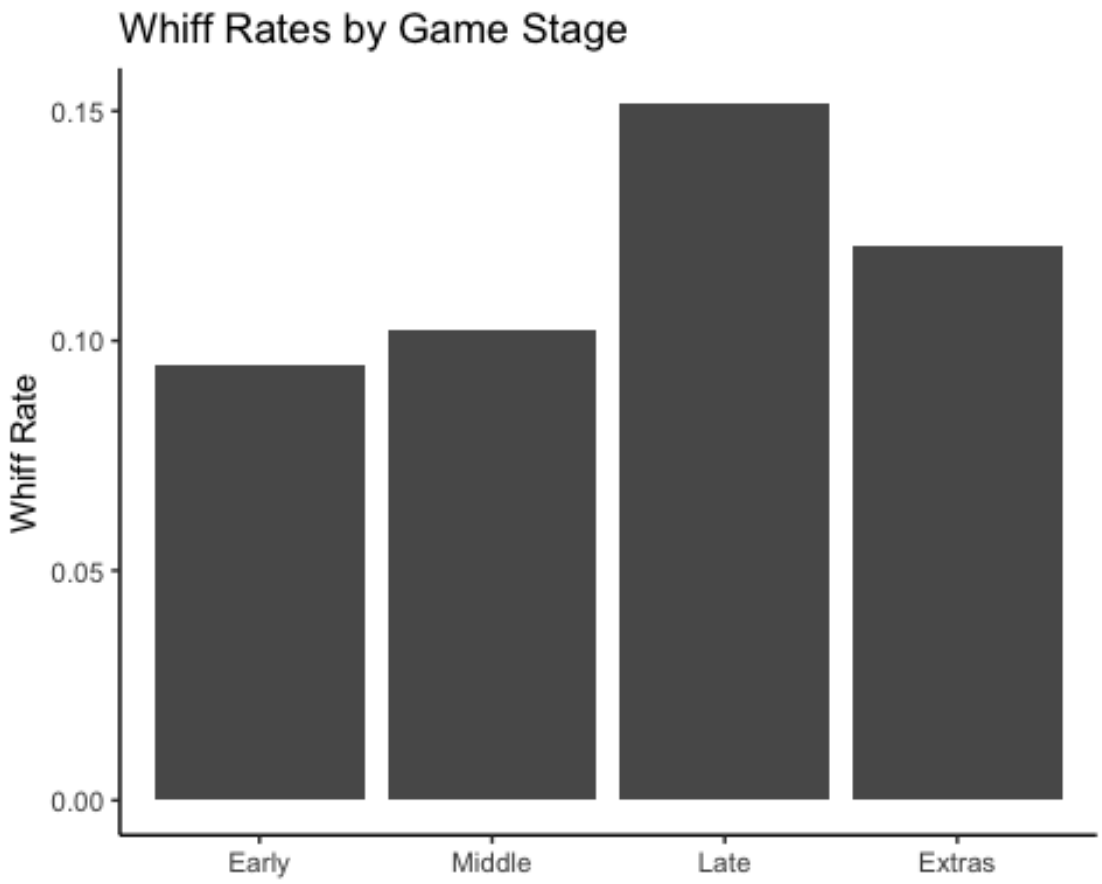


In early counts, it seems that whiff rate increases as the number of strikes increase, and the highest whiff rate occurs during a full count. The bar charts showing whiff rates by the number of strikes and balls indicate that more whiffs occur as both strikes and balls increase. It's interesting to see that no curveballs were thrown in 3-0 counts, and none of the 23 curveballs thrown in 2-0 counts resulted in whiffs.

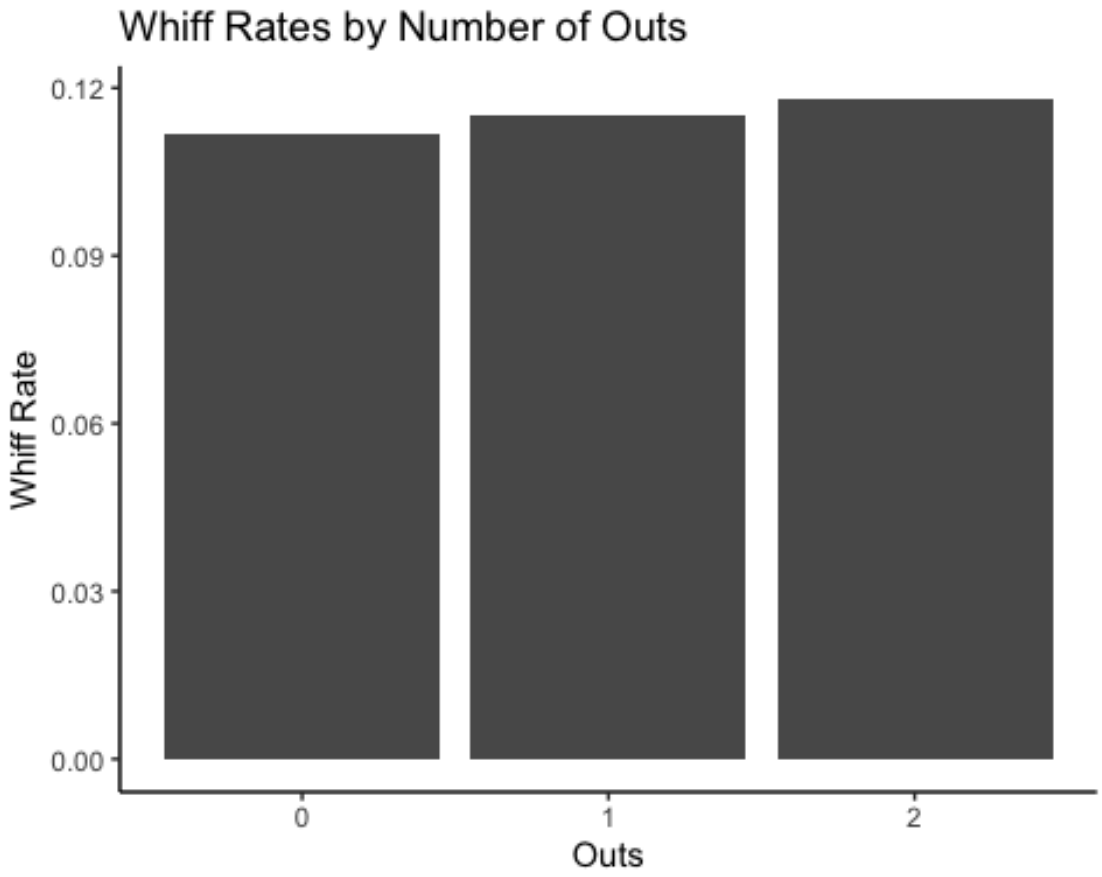
The bar charts below show a moderate association between whiff rates and inning, with a near 5% uptick in innings 7 through 9, which is likely due to the fact that relief pitching usually takes over at this stage in the game. These pitchers tend to have "nastier stuff" and generally only face a batter once, potentially leading to more whiffs. Unsurprisingly, the number of outs appears to have no effect on whiffs.

```
# Create bar graph showing the whiff rate by inning, which are grouped into
thirds
pitch_data %>%
  mutate(Stage = case_when(
    Inning %in% 1:3 ~ "Early",
    Inning %in% 4:6 ~ "Middle",
    Inning %in% 7:9 ~ "Late",
    Inning > 9 ~ "Extras"
  ), Stage = factor(Stage, levels = c("Early", "Middle", "Late", "Extras")))
%>%
  group_by(Stage) %>%
```

```r
  summarise(whiff_rate = mean(whiff)) %>%
  ggplot(aes(Stage, whiff_rate))+
  geom_bar(stat = "identity")+
  theme_classic()+
  labs(title = "Whiff Rates by Game Stage", x = NULL, y = "Whiff Rate")
```

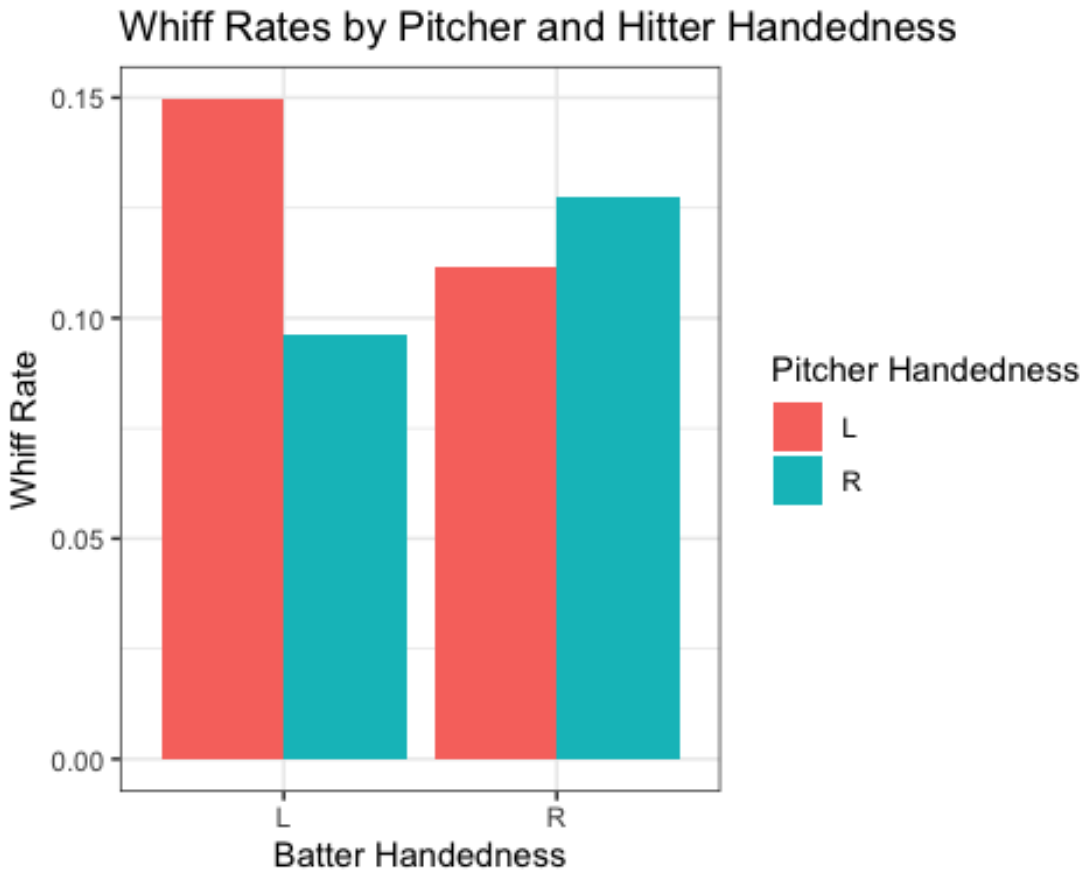



```r
# Create bar graph showing whiff rates by the number of outs
pitch_data %>%
  group_by(Outs) %>%
  summarise(whiff_rate = mean(whiff)) %>%
  ggplot(aes(Outs, whiff_rate))+
  geom_bar(stat = "identity")+
  theme_classic()+
  labs(title = "Whiff Rates by Number of Outs", x = "Outs", y = "Whiff Rate")
```



Lastly, look at the influence of pitcher and batter handedness on whiffs:

```
# Create bar chart showing whiff rates by pitcher/batter handedness
ggplot(pitch_data, aes(Batter_Hits, whiff, group = Pitcher_Throws, fill =
Pitcher_Throws))+
  geom_bar(stat = "summary", fun = "mean", position = "dodge")+
  theme_bw()+
  labs(title = "Whiff Rates by Pitcher and Hitter Handedness", x = "Batter
Handedness", y = "Whiff Rate", fill = "Pitcher Handedness")
```



As expected, same-handed curveballs have higher whiff rates than opposite-handed curveballs, and lefty-lefty matchups have higher whiff rates than righty-righty matchups. Since pitch location, count, inning, and handedness appear to affect whiffs, the associated variables will remain in the feature set, while `Outs` will be removed since this variable seems to have no effect on whiffs. The categorical variables `Pitcher_Throws` and `Batter_Hits` are also converted to binary indicator variables using one-hot encoding:

```
# Removing Outs from features and adding Pitcher_Throws and Batter_Hits
features <- features[!(features == "Outs")]
features <- append(features, c("Pitcher_Throws", "Batter_Hits"))

# Create data containing features and target variable
newdata <- pitch_data %>% select(features, whiff)
colnames(newdata)

# Encode numeric binary variables for Pitcher_Throws and Batter_Hits using
fastDummies package
#install.packages("fastDummies")
library(fastDummies)
newdata <- newdata %>%
  dummy_cols(select_columns = c("Pitcher_Throws", "Batter_Hits")) %>%
  select(-c("Pitcher_Throws", "Batter_Hits", "Pitcher_Throws_R",
"Batter_Hits_R"))
```

```r
summary(newdata)

# Changing to factors
newdata$whiff <- as.factor(newdata$whiff)
newdata$Pitcher_Throws_L <- as.factor(newdata$Pitcher_Throws_L)
newdata$Batter_Hits_L <- as.factor(newdata$Batter_Hits_L)
```

Next, the new dataset is split into training and test sets before model training:

```r
# Split data into training and test sets using 80-20 split
set.seed(42)
index <- sample(1:nrow(newdata), .8*nrow(newdata))
train <- newdata[index,]
test <- newdata[-index,]
```

The newly created training data is used to train the logistic regression model:

```r
# Train logistic regression model
lr <- glm(whiff ~ ., data = train, family = "binomial")
summary(lr)

lr.new <- glm(whiff ~ Balls*Strikes + x_movement*z_movement +
release_spin_rate + release_pos_z + plate_x*plate_z +
Pitcher_Throws_L*Batter_Hits_L, data = train, family = "binomial")
summary(lr.new) # best performing model
```

The first model returned an AIC value of 1307.5 with `Strikes`, `release_pos_z`, and `plate_z` significant with p-values less than 0.01 and `x_movement` significant with a p-value of approximately 0.06. The revised logistic regression model excludes insignificant variables and includes interaction variables for `Balls*Strikes`, `x_movement*z_movement`, `plate_x*plate_z`, and `Pitcher_Throws_L*Batter_Hits_L`. This model has a slightly lower AIC value with similar significant variables, including `plate_x` and the interaction term, at the .01 level. The model's predictions were about 67% and 65% accurate on the training and test data, respectively.

```r
# Make predictions on train and test data
lr.predictions.train <- predict(lr.new, train, type = "response")
lr.predictions.test <- predict(lr.new, test, type = "response")

# Converting to binary variable if predicted whiff percentage is greater than
league average
train.pred <- ifelse(lr.predictions.train > whiff.percent, 1, 0)
test.pred <- ifelse(lr.predictions.test > whiff.percent, 1, 0)

# Confusion matrix with training predictions vs observed values
lr.conf.matrix.train <- table(train.pred, train$whiff)
lr.conf.matrix.train
```

```
##
## train.pred    0    1
##          0 1156   69
##          1  573  158
```

```
lr.pt.train <- prop.table(lr.conf.matrix.train) # proportion table
lr.train.accuracy <- lr.pt.train[1,1] + lr.pt.train[2,2] # 67.1% accurate
training data
paste("Logistic Regression Training Accuracy:", lr.train.accuracy)
```

```
## [1] "Logistic Regression Training Accuracy: 0.671779141104294"
```

```
# Confusion matrix with test predictions vs observed values
lr.conf.matrix.test <- table(test.pred, test$whiff)
lr.conf.matrix.test
```

```
##
## test.pred   0   1
##         0 282  17
##         1 153  37
```

```
lr.pt.test <- prop.table(lr.conf.matrix.test) # proportion table
lr.test.accuracy <- lr.pt.test[1,1] + lr.pt.test[2,2] # 65.2% accurate test
data
paste("Logistic Regression Test Accuracy:", lr.test.accuracy)
```

```
## [1] "Logistic Regression Test Accuracy: 0.652351738241309"
```

Next, the features and target variables are split from the training and test sets before converting to a 'DMatrix' object to be fed to the model. Both a boosted and non-boosted model are evaluated before making predictions, and ultimately the boosted model performed better.

```
library(xgboost)

# Separate training features and target
features_train <- train %>% select(-whiff)
features_train$Pitcher_Throws_L <-
as.numeric(features_train$Pitcher_Throws_L)
features_train$Batter_Hits_L <- as.numeric(features_train$Batter_Hits_L)
target_train <- as.numeric(as.character(train$whiff))

# Separate test features and target
features_test <- test %>% select(-whiff)
features_test$Pitcher_Throws_L <- as.numeric(features_test$Pitcher_Throws_L)
features_test$Batter_Hits_L <- as.numeric(features_test$Batter_Hits_L)
target_test <- as.numeric(as.character(test$whiff))

# Convert training and test data to 'DMatrix' object
dtrain <- xgb.DMatrix(data = as.matrix(features_train), label = target_train)
dtest <- xgb.DMatrix(data = as.matrix(features_test), label = target_test)
```

```r
# Parameter to measure learning progress
watchlist <- list(train=dtrain, test=dtest)

# Initial XGBoost model
xgb_new <- xgb.train(
  data=dtrain,
  max.depth=10,
  eta=1,
  nthread = 3,
  nrounds=20,
  watchlist=watchlist,
  eval.metric = "logloss",
  objective = "binary:logistic")

# Apply linear boosting
boosted <- xgb.train(
  data=dtrain,
  booster = "gblinear",
  nthread = 3,
  nrounds=20,
  watchlist=watchlist,
  eval.metric = "logloss",
  objective = "binary:logistic")

# Make predictions on training and test data
train.predictions <- predict(boosted, dtrain, type = "class")
train.pred.binary <- as.numeric(train.predictions > whiff.percent)
confusion.matrix.train <- table(train.pred.binary, target_train)
confusion.matrix.train
```

```
##                  target_train
## train.pred.binary    0    1
##                 0 1140   72
##                 1  589  155
```

```r
pt.train <- prop.table(confusion.matrix.train)
xgb.train.accuracy <- pt.train[1,1] + pt.train[2,2] # 66.1% accuracy
paste("XGBoost Training Accuracy:", xgb.train.accuracy)
```

```
## [1] "XGBoost Training Accuracy: 0.662065439672802"
```

```r
predictions <- predict(boosted, dtest, type = "class")
pred_binary <- as.numeric(predictions > whiff.percent)
confusion.matrix <- table(pred_binary, target_test)
confusion.matrix
```

```
##            target_test
## pred_binary   0   1
##           0 283  13
##           1 152  41
```

```
pt <- prop.table(confusion.matrix)
xgb.test.accuracy <- pt[1,1] + pt[2,2] # 65.6% accuracy
paste("XGBoost Test Accuracy:", xgb.test.accuracy)

## [1] "XGBoost Test Accuracy: 0.662576687116564"
```

The XGBoost model performed slightly better than the logistic regression model, but the accuracy results are surprisingly similar. Looking at the feature importance bar chart, it appears that `plate_z` has by far the biggest negative impact on `whiff`, while `Strikes` and pitcher handedness carry the most positive weight for influencing whiffs. In other words, the higher the vertical location of a curveball, the lower the chance of a whiff. Conversely, a higher number of strikes result in a higher probability of a whiff.

```
# Build feature importance data table
importance <- xgb.importance(feature_names = colnames(dtrain), model =
boosted)

# Plot the feature importance
xgb.plot.importance(importance, xlab = "Weight", main = "Feature Importance
for XGBoost Model")
```