Results

Matthew Wankiewicz

2022-12-15

Question 1: Predict the chance of a pitch being put in play. Please use this model to predict the chance of each pitch in the "deploy.csv" file being put in play and return a csv with your predictions.

```r
deploy$estimate <- predict(model, newdata = deploy,
                           type = "response")
write_csv(deploy, "deploy.csv")
deploy %>%
  slice(1:10) %>%
  knitr::kable()
```

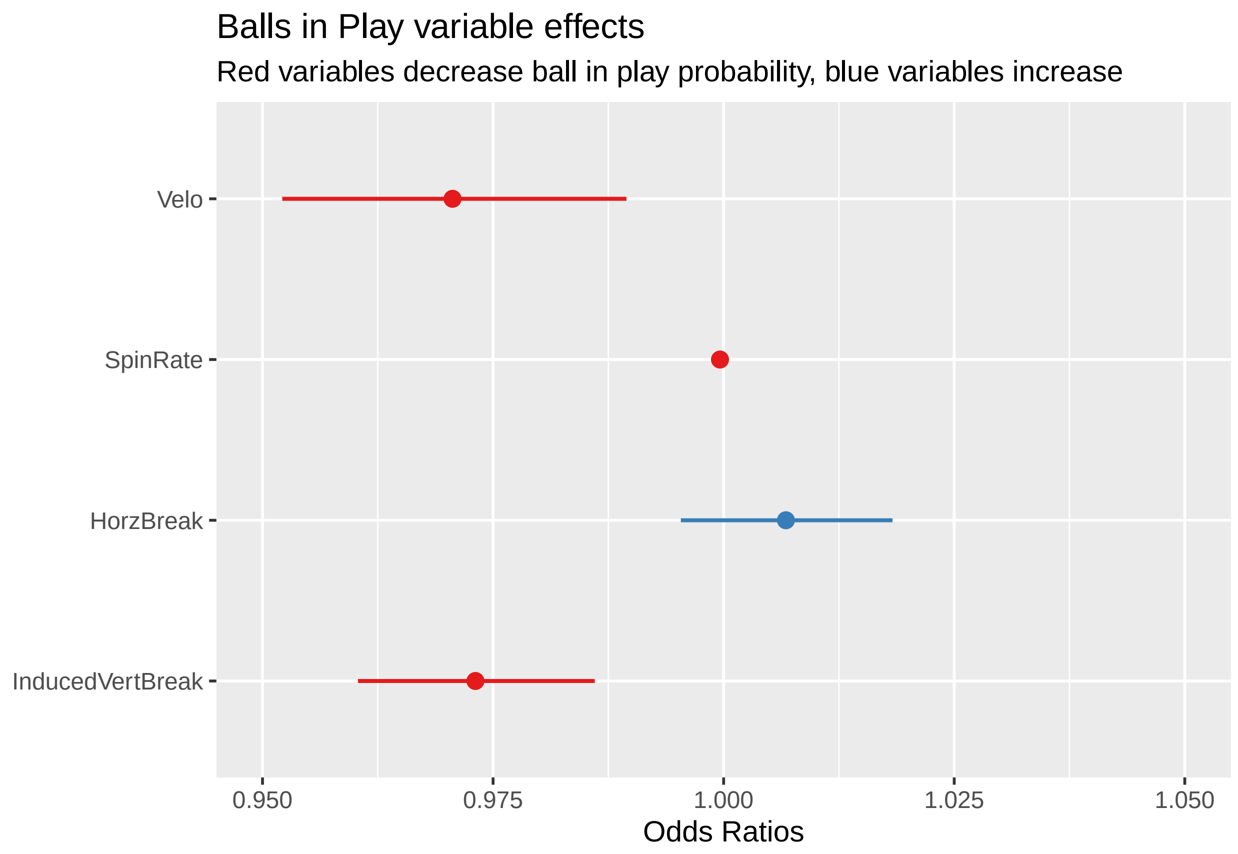
Velo	SpinRate	HorzBreak	InducedVertBreak	estimate
94.72	2375	3.10	18.15	0.4379868
95.25	2033	11.26	14.50	0.5062214
92.61	2389	11.00	21.93	0.4398209
94.94	2360	6.84	18.11	0.4443093
97.42	2214	16.70	13.38	0.4889004
95.98	2495	11.25	17.12	0.4374493
94.88	1998	15.13	15.22	0.5140464
92.73	2049	1.55	18.47	0.4800245
92.39	1955	18.15	7.25	0.5951540
95.77	1976	10.04	14.56	0.5055273

The `estimate` column in the `deploy.csv` file represents the estimates created by the model.

Question 2: In one paragraph, please explain your process and reasoning for any decisions you made in Question 1.

After loading in the data, I found that there were some missing values in the `SpinRate` column. Instead of omitting them, I decided to replace them with the average spin rates in the data. I then did some exploratory data analysis to get more familiar with the data. After doing this, I split the data into a training and testing set and started to create models. The two types of models I focused on were logistic regression and random forest since we are predicting a binary response. After trying the two types of models, I settled on logistic regression because it was able to create estimates that made sense and were accurate in terms of the data, but it was also able to quantify the impact of each variable. I used step-wise selection and created a model that used all four variables and interactions between `Velo` and `InducedVertBreak` as well as `HorzBreak` and `InducedVertBreak`. Before going all-in with this model, I decided to balance the `InPlay` column to see if that gave better results as roughly 70% of the column were 0's. After doing this, I fit a model with all 4 variables as the predictors with no interactions. After testing this model and comparing it to the previous model, it led to lower accuracy and lower AUC, but it increased the sensitivity and negative predictive values. I felt that this model would be better suited because along with having some better performance metrics, it made many more predictions above the 0.5 threshold compared to the previous model (1221 vs 1). While the balanced model may not have had better accuracy, its other performance metrics paired with its simplicity make it the best model to use for this research question (the step-wise model has a higher accuracy because it predicted almost all 0's, meaning it would be 72% accurate if it just predicted 0 no matter what). Throughout the creation of the models, I wanted the model to be accurate, but also simple enough that anyone could easily understand it, allowing me to convey the results to the pitcher quickly and efficiently.

Question 3: In one or two sentences, please describe to the pitcher how these 4 variables affect the batter's ability to put the ball in play. You can also include one plot or table to show to the pitcher if you think it would help.



The most important variables for limiting the chance of a ball hit in play are velocity and induced vertical break, as higher values decrease the batter's chance of putting the ball in play. Higher spin rates decrease the chances but at a much lower factor while higher horizontal breaks increase the chance of a ball hit in play.

Question 4: In one or two sentences, please describe what you would see as the next steps with your model and/or results if you were in the analyst role and had another week to work on the question posed by the pitcher.

For the next steps, I would want to work with data that only includes swinging strikes and balls in play, or the count the pitch was thrown in, as this would give us a better idea of the impact each variable has on the ball going in play. I would also want to continue to test the model on different data sets and see if it still performs well.

Question 5: Please include any code (R, Python, or other) you used to answer the questions. This code doesn't need to be production quality or notated.

All code is also contained the `models.R` file. This contains data visualization, data cleaning and model creation/testing.

```r
library(tidyverse)
library(caret)
library(visdat)
library(lmtest)
library(pROC)
library(randomForest)
library(ROSE)

## read in and clean data ----------------------------------------------------
training <- read_csv("training.csv")

## change spin rate to numeric
training$SpinRate <- as.numeric(training$SpinRate)


## find missing values
colSums(is.na(training))

## impute missing values with average of column
training <- training %>%
  mutate(SpinRate = replace_na(SpinRate, mean(training$SpinRate, na.rm = T)))

## explore data ---------------------------------------------------------------

## characteristics of balls in play
training %>%
  filter(InPlay == 1) %>%
  summary()

## characteristics of balls not in play
training %>%
  filter(InPlay == 0) %>%
  summary()

### it appears that balls in play have slightly lower velocity, spin rates and
### higher Horizontal and Induced Vertical Breaks

vis_cor(training)
```

```r
cols <- c("Red", "Blue")
pairs(training,
      col = cols[as.factor(training$InPlay)])

training %>%
  ggplot(aes(InPlay)) +
  geom_bar() +
  labs(title = "Distribution of balls in play vs not in play")

### build models --------------------------------------------------------------

### create training and testing samples
set.seed(111)
sample <- sample(c(TRUE, FALSE), nrow(training), replace=TRUE, prob=c(.7,.3))
train  <- training[sample, ]
test   <- training[!sample, ]


## start with basic model - all four variables, no interactions

model1 <- glm(InPlay ~ ., data = train, family = binomial())

## get summary of the model
summary(model1)


## create estimates on testing set
test$estimate <- predict(model1, newdata = test,
                         type = "response")

test <- test %>%
  mutate(est_ip = ifelse(estimate > .5, 1, 0))

## create a confusion matrix for logistic model
confusionMatrix(reference = as.factor(test$InPlay), as.factor(test$est_ip), positive = "1")


#### while the model gives us a good idea of the importance of each variable,
#### it's estimates are very low, meaning we cannot look at its ROC curve, AUC and
#### and other performance metrics using a cutoff of 0.5


## create model including interactions between variables
model2 <- glm(InPlay ~ .^2,
             data = train, family = binomial())

## summary of the model
summary(model2)


## create estimates on testing set
test$estimate <- predict(model2, newdata = test,
                         type = "response")
```

```
test <- test %>%
  mutate(est_ip = ifelse(estimate > .5, 1, 0))

## create a confusion matrix for logisitic model
confusionMatrix(reference = as.factor(test$InPlay), as.factor(test$est_ip), positive = "1")

## ROC plot and AUC
roc(test$est_ip, test$InPlay, plot = T)


### this model is much more difficult to interpret than our initial model but
### it has very strong results (very high AUC). We can attempt to improve
### upon this model using step-wise selection to find a better overall model

## step-wise selection (AIC)
model_step <- step(model2, direction = "both")

## summary of new model
summary(model_step)


## create estimates on testing set
test$estimate <- predict(model_step, newdata = test,
                             type = "response")

test <- test %>%
  mutate(est_ip = ifelse(estimate > .5, 1, 0))

## create a confusion matrix for logisitic model
confusionMatrix(reference = as.factor(test$InPlay), as.factor(test$est_ip), positive = "1")

## ROC plot and AUC
roc(test$est_ip, test$InPlay, plot = T)

precision(test, as.factor(InPlay), as.factor(est_ip))

### our new model returns identical results to the more complex model. We will
### use the newer model as it is easier to interpret the impact of each variable
### on the chance a ball is put in play


## compare new model to null model and initial model

null_mod <- glm(InPlay ~ 1, data = train, family = binomial())

lrtest(null_mod, model_step)

lrtest(model1, model_step)


### both likelihood ratio tests yield p-values below 0.05, meaning that the final
### model we created has the best fit of the data so far
```

```
lrtest(model2, model_step)

### After testing the complex and step-wise models, we cannot conclude that there
### is a significant difference. Therefore, we will use the simpler one.

### random forest - determine if random forest yields better results compared to
### the logistic regression model


rf <- randomForest(as.factor(InPlay) ~ ., data = train)

print(rf)

preds <- predict(rf, newdata = test)

test$est_ip <- preds

confusionMatrix(reference = as.factor(test$InPlay), as.factor(test$est_ip), positive = "1")

## ROC plot and AUC
roc(test$est_ip, test$InPlay, plot = T)

### these results are not as strong as the logistic regression model, try to find optimal
### mtry value before settling on logistic
set.seed(111)
tuneRF(train[-1], as.factor(train$InPlay))


### optimal value is 1, lets try again
rf2 <- randomForest(as.factor(InPlay) ~ ., data = train, mtry = 1)

print(rf2)

preds <- predict(rf2, newdata = test)

test$est_ip <- preds

confusionMatrix(reference = as.factor(test$InPlay), as.factor(test$est_ip), positive = "1")

## ROC plot and AUC
roc(test$est_ip, test$InPlay, plot = T)

## find important variables
varImpPlot(rf2)

### results from the logistic regression are still the best, will now
### try re-sampling the response to see if that improves results

## re-sample -----------------------------------------------------------
set.seed(111)
over_sampled <- ovun.sample(InPlay ~ ., data = train, method = "both",
                            N = 6971)$data
```

```r
## re-attempt fitting logistic regression and random forest models


### logistic

model_samp <- glm(InPlay ~ ., data = over_sampled, family = binomial())

## summary of new model
summary(model_samp)


## create estimates on testing set
test$estimate <- predict(model_samp, newdata = test,
                         type = "response")

test <- test %>%
  mutate(est_ip = ifelse(estimate > .5, 1, 0))

## create a confusion matrix for logistic model
confusionMatrix(reference = as.factor(test$InPlay), as.factor(test$est_ip), positive = "1")

## ROC plot and AUC
roc(test$est_ip, test$InPlay, plot = T)

precision(test, as.factor(InPlay), as.factor(est_ip))

### while the AUC is much lower, this model allows us to have easy to interpret
### coefficients and is able to detect negative occurrences at a higher rate
### this model will likely be better to use compared to the step-wise model

### random forest
rf_samp <- randomForest(as.factor(InPlay) ~ ., data = over_sampled, mtry=1)

print(rf_samp)

preds <- predict(rf_samp, newdata = test)

test$est_ip <- preds

confusionMatrix(reference = as.factor(test$InPlay), as.factor(test$est_ip), positive = "1")

## ROC plot and AUC
roc(test$est_ip, test$InPlay, plot = T)

## find important variables
varImpPlot(rf_samp)

### we still have better results from our logistic regression model,
### this will be the model we use to create predictions

final_model <- model_samp

write_rds(final_model, "final_model.rds")
```

```r
## display coefficients of final model

sjPlot::plot_model(model_samp) +
  labs(subtitle = "Variables less than 1 decrease ball in play probability, greater than 1 increases") +
  scale_y_continuous(limits = c(0.9, 1.1))


### prep "deploy" files -------------------------------------------------------

deploy <- read_csv("deploy.csv")
deploy$SpinRate <- as.numeric(deploy$SpinRate)


## find missing values
colSums(is.na(deploy))

## impute missing values with average of column
deploy <- deploy %>%
  mutate(SpinRate = replace_na(SpinRate, mean(deploy$SpinRate, na.rm = T)))


write_csv(deploy, "deploy.csv")
```