Modeling In Play Probability

```r
# importing libraries
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(readxl)
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.2.0 --

## v broom        0.8.0     v rsample      0.1.1
## v dials        0.1.1     v tune         0.2.0
## v infer        1.0.2     v workflows    0.2.6
## v modeldata    0.1.1     v workflowsets 0.2.1
## v parsnip      0.2.1     v yardstick    1.0.0
## v recipes      0.2.0

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Learn how to get started at https://www.tidymodels.org/start/
```

```r
# importing data
pitches <- read.csv("Assessment_JQA_2022.csv")
```

1. Checking the data

```r
str(pitches)
```

```
## 'data.frame':    100000 obs. of  22 variables:
##  $ is_in_play           : int  1 1 1 1 1 0 1 1 1 1 ...
##  $ pitch_type           : chr  "SL" "CU" "FA" "SI" ...
```

```
##  $ three_plus           : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ batter_stance        : chr  "R" "R" "R" "R" ...
##  $ pitcher_throws       : chr  "L" "R" "L" "L" ...
##  $ strikes              : int  0 1 2 1 2 1 1 2 0 1 ...
##  $ balls                : int  1 1 2 1 1 2 2 1 0 1 ...
##  $ outs                 : int  2 2 0 1 1 0 1 0 2 2 ...
##  $ pitch_plate_location_x: num  -0.356 0.095 -0.556 -0.212 -0.248 -0.321 0.221 0.571 0.729 -0.368 ..
##  $ pitch_plate_location_z: num  1.84 2.14 2.52 3.11 3.44 ...
##  $ pitch_initial_speed   : num  87.9 81 91.8 92.2 83.1 90.6 94.4 85.6 81.9 89.2 ...
##  $ pitch_arc_break_x     : num  0.404 4.821 2.652 5.927 -1.289 ...
##  $ pitch_arc_break_z     : num  -6.28 -13.18 -3.52 -3.88 -9.83 ...
##  $ pitch_spin_rate       : num  1026 2257 2457 3510 478 ...
##  $ inning                : int  3 7 5 6 8 1 2 9 7 1 ...
##  $ top_of_inning         : int  1 1 0 0 0 1 1 1 1 0 ...
##  $ pitch_per_atbat       : int  2 3 5 3 7 4 4 4 1 3 ...
##  $ home_team_runs        : int  1 2 0 0 4 0 0 7 2 1 ...
##  $ away_team_runs        : int  0 3 1 0 2 0 0 0 7 0 ...
##  $ pitcher_mlb_id        : int  518516 501789 457918 519242 593576 502327 533167 533167 459987 59276
##  $ batter_mlb_id         : int  596115 449181 460099 592200 457803 543807 467793 643603 408252 51929
##  $ venue_city            : chr  "San Francisco" "Houston" "Baltimore" "St. Petersburg" ...
```

```r
# running through each categorical / factorial variable, as well as easy to check
# quantitative variables, for any abnormalities
unique(pitches$is_in_play)
```

```
## [1] 1 0
```

```r
# no apparent issues. What's the typical percentage of pitches put in play? Is this
# representative?

unique(pitches$pitch_type)
```

```
## [1] "SL" "CU" "FA" "SI" "CH" "FC"
```

```r
# all 5 pitch types represented

unique(pitches$three_plus)
```

```
## [1]  0  1 NA
```

```r
# NA's, look into this further

unique(pitches$batter_stance)
```

```
## [1] "R" "L" NA
```

```r
# NA's, look into this further

unique(pitches$pitcher_throws)
```

```
## [1] "L" "R" NA
```

```r
# NA's, look into this further
unique(pitches$strikes)
```

```
## [1]  0  1  2 NA
```

```r
# NA's, look into this further
unique(pitches$balls)
```

```
## [1]  1  2  0  3 NA
```

```r
# NA's, look into this further
unique(pitches$outs)
```

```
## [1] 2 0 1
```

```r
# 0, 1, and 2 outs represented, all good here.
unique(pitches$inning)
```

```
##  [1]  3  7  5  6  8  1  2  9  4 12 11 10 13 14 16 18 17 15
```

```r
# no data points outside of the realm of normalcy
unique(pitches$top_of_inning)
```

```
## [1] 1 0
```

```r
# only 1's and 0's, all good
unique(pitches$pitch_per_atbat)
```

```
##  [1]   2   3   5   7   4   1   6   9  11   8  10  14  65  12  13  15  31  54  37
## [20] 100  27  18  21  92  69
```

```r
# some very obviously wrong data points here, will return to look at this further
unique(pitches$home_team_runs)
```

```
##  [1]  1  2  0  4  7  3  6  8  9  5 10 11 12 13 15 17 14 16
```

```r
# nothing out of the norm here
unique(pitches$away_team_runs)
```

```
##  [1]  0  3  1  2  7  4  5 13  6  9 10 11 12  8 14 15 16 20 18 17 21
```

```r
# nothing out of the norm here either

unique(pitches$venue_city)
```

```
##  [1] "San Francisco"  "Houston"        "Baltimore"      "St. Petersburg"
##  [5] "Cincinnati"     "Minneapolis"    "Anaheim"        "Arlington"
##  [9] "Oakland"        "Seattle"        "Pittsburgh"     "Detroit"
## [13] "Flushing"       "St. Louis"      "Cleveland"      "Atlanta"
## [17] "Chicago"        "Miami"          "Toronto"        "Bronx"
## [21] "San Diego"      "Washington"     "Los Angeles"    "Phoenix"
## [25] "Philadelphia"   "Boston"         "Milwaukee"      "Kansas City"
## [29] "Denver"
```

```r
# only 29 venues here
```

```r
summary(pitches$pitch_plate_location_x)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
## -8.01000 -0.47600 -0.02900 -0.02652  0.41700  2.78800       15
```

```r
summary(pitches$pitch_plate_location_z)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -1.777   1.741   2.248   2.235   2.743  11.988      15
```

```r
# there appear to be 15 na's

summary(pitches$pitch_initial_speed)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   45.50   84.00   89.00   88.01   92.60  108.30       8
```

```r
summary(pitches$pitch_arc_break_x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -7.2046 -1.6718  0.8292  0.4832  2.5668  7.2425      10
```

```r
# nothing tremendously unusual here / 8 and 10 na's respectively

summary(pitches$pitch_arc_break_z)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
## -26.6802  -8.1528  -5.9258  -6.5538  -4.1485  -0.3805       24
```

```r
# reveals at least one outlier on the negative side, 24 na's

summary(pitches$pitch_spin_rate)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.84 1190.68 1789.30 1682.83 2188.02 4388.31
```

```
# reveals some likely outliers in terms of both min and max RPM

summary(pitches$pitcher_mlb_id)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  112526  457918  519141  518854  573186  664641
```

```
summary(pitches$batter_mlb_id)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  120074  455104  518466  508475  571980  666560
```

```
# no NA's, players all accounted for
```

2. Building the First Model: Random Forest

```
# preparing predictors for modeling
pitches <- pitches %>% mutate(SL = ifelse(pitch_type == "SL", 1, 0),
                              CU = ifelse(pitch_type == "CU", 1, 0),
                              FA = ifelse(pitch_type == "FA", 1, 0),
                              SI = ifelse(pitch_type == "SI", 1, 0),
                              CH = ifelse(pitch_type == "CH", 1, 0),
                              FC = ifelse(pitch_type == "FC", 1, 0),
                              batter_stance = ifelse(batter_stance == "R", 1, 0),
                              pitcher_throws = ifelse(pitcher_throws == "R", 1, 0),
                              is_in_play = as.factor(case_when(is_in_play == 1 ~ "InPlay",
                                                               is_in_play == 0 ~ "Not")),
                              inPlay = ifelse(is_in_play == "InPlay", 1, 0)) %>% na.omit()
```

```
# splitting into testing and training data to more rigorously examine the model
set.seed(1234)
pitches_split <- initial_split(pitches, prop = .8)
pitches_train <- training(pitches_split)
pitches_test <- testing(pitches_split)

# creating a model recipe which we will use to train the model
# we exclude pitch type, as it has been encoded into new variables above, and inPlay as it
# is a duplicate of is_in_play which will only be used for model evaluation later
# we also exclude top_of_inning and venue_city as they seem unlikely to have a substantial
# effect
# also excluded pitcher and batter id's, despite likely usefulness, it is probable we would
# need significant quantities of data on each player to make a model with confidence
model_recipe <-
  recipe(is_in_play ~ ., data = pitches_train) %>%
  update_role(pitch_type, top_of_inning, pitcher_mlb_id, batter_mlb_id, venue_city, inPlay, new_role =
  step_normalize(all_predictors())

summary(model_recipe)
```

```
## # A tibble: 29 x 4
##    variable                type    role      source
##    <chr>                   <chr>   <chr>     <chr>
##  1 pitch_type              nominal ID        original
##  2 three_plus              numeric predictor original
##  3 batter_stance           numeric predictor original
##  4 pitcher_throws          numeric predictor original
##  5 strikes                 numeric predictor original
##  6 balls                   numeric predictor original
##  7 outs                    numeric predictor original
##  8 pitch_plate_location_x numeric predictor original
##  9 pitch_plate_location_z numeric predictor original
## 10 pitch_initial_speed     numeric predictor original
## # ... with 19 more rows
```

```
# defining the type of model
rf_mod <-
  rand_forest() %>%
  set_engine("ranger") %>%
  set_mode("classification")
```

```
# defining model workflow
rf_workflow <-
  workflow() %>%
  add_model(rf_mod) %>%
  add_recipe(model_recipe)
```

```
# fitting the model
rf_fit_inplayprob <-
  rf_workflow %>%
  fit(data = pitches_train)
```

```
# creating dataframe with predictions and actual outcome included
rfpredict <- rf_fit_inplayprob %>% predict(new_data = pitches_train) %>%
  bind_cols(pitches_train)

rfpredict <- rf_fit_inplayprob %>% predict(new_data = pitches_train, type="prob") %>%
  bind_cols(rfpredict)
```

```
# assessing accuracy of model in training, predictably high, but overfitted. Will consult
# test data metrics for better understanding of predictive power.
metrics(rfpredict, is_in_play, .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.968
## 2 kap      binary         0.930
```

```
# applying the model to previously unseen data
rftestpredict <- rf_fit_inplayprob %>% predict(new_data = pitches_test) %>%
  bind_cols(pitches_test)
```

```r
rftestpredict <- rf_fit_inplayprob %>% predict(new_data = pitches_test, type="prob") %>%
  bind_cols(rftestpredict)

# accuracy metrics for previously unseen data. Some predictable drop-off in accuracy. Still,
# a solid 74%.
# the kappa is a moderate 0.41, showing that the model does a good deal better than chance,
# given the weights of our outcome variable present in the data
metrics(rftestpredict, is_in_play, .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.741
## 2 kap      binary         0.417
```

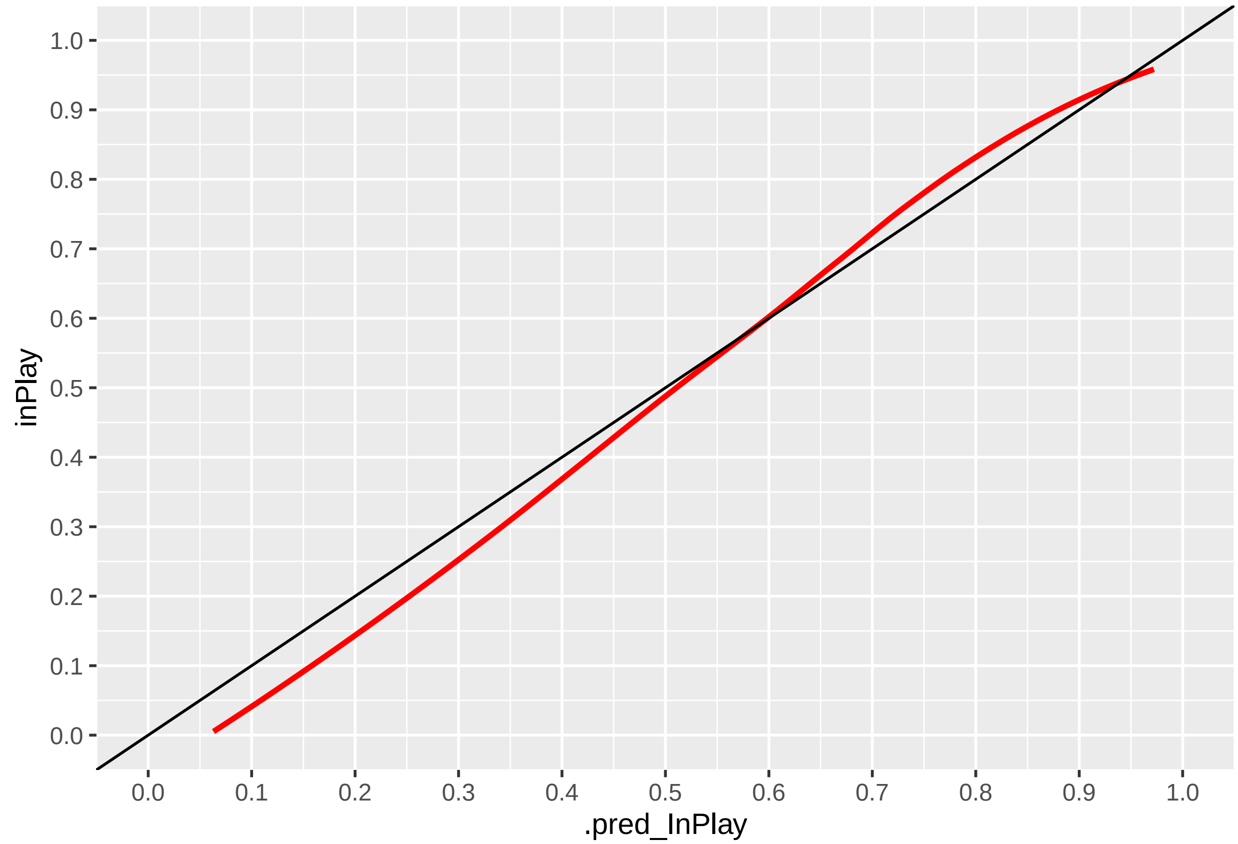
```r
# confusion matrix, looking for any patterns of errors. The vast majority of pitches are
# predicted to be put in play. When the model predicted pitches would not be in play, it
# still maintains strong accuracy.
rftestpredict %>%
  conf_mat(is_in_play, .pred_class)
```

```
##           Truth
## Prediction InPlay   Not
##     InPlay  10867  3723
##     Not      1414  3847
```

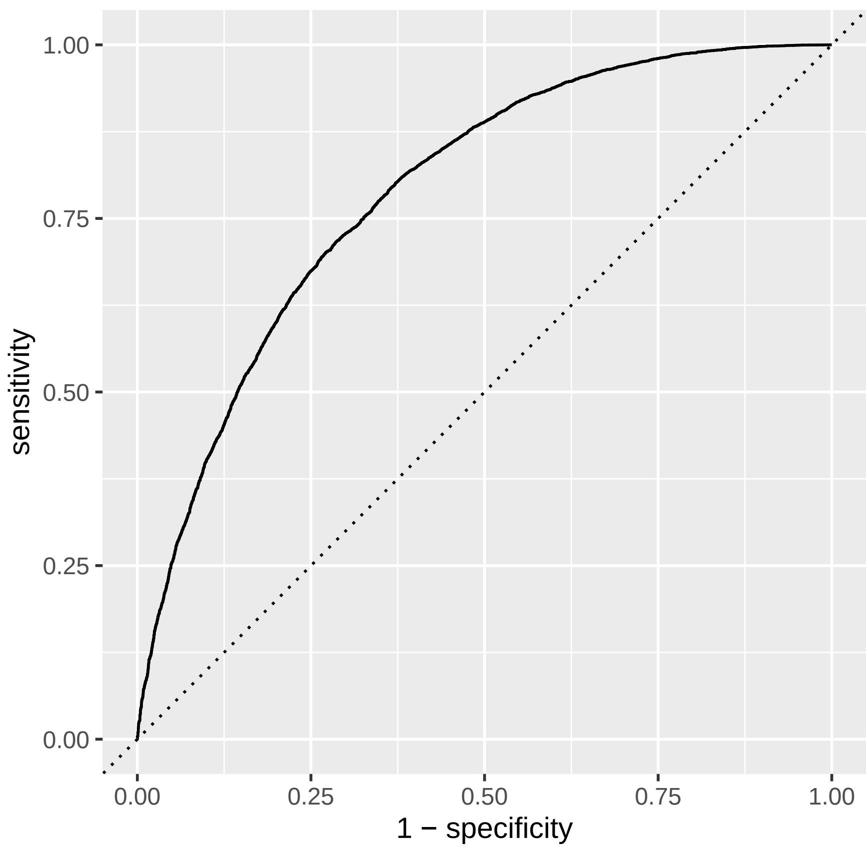
```r
# building a continuity plot, which shows that the random forest does an excellent job of
# approximating the probability of a ball being put in play at any given point
rftestpredict %>%
arrange(.pred_InPlay) %>%
ggplot(aes(x = .pred_InPlay, y = inPlay)) +
scale_y_continuous(limits = c(0, 1), breaks = seq(0, 1, by = 0.1)) +
scale_x_continuous(limits = c(0, 1), breaks = seq(0, 1, by = 0.1)) +
geom_smooth(aes(x = .pred_InPlay, y = inPlay), color = "red", se = F, method = "loess") +
geom_abline()
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 4 rows containing missing values (geom_smooth).
```




```
# building out a ROC-AUC plot, in case we want to fine tune the model to be better at
# avoiding false positives or false negatives
roc_data <- roc_curve(rftestpredict, truth = is_in_play, .pred_InPlay)
roc_data %>%
  ggplot(aes(x = 1 - specificity, y = sensitivity)) +
  geom_path() +
  geom_abline(lty = 3) +
  coord_equal()
```



3. Building a Second Model: Logistic Regression

```
# we now turn our attention to logistic regression
# I suspect the results will not be as strong, given the data is likely non-linear

# splitting data into testing and training data
log_split <- initial_split(pitches, prop = .8)
log_train <- training(log_split)
log_test <- testing(log_split)
```

```
# creating a model recipe. We utilize the same variables as our random forest.
log_recipe <-
  recipe(is_in_play ~ ., data = log_split) %>%
  update_role(pitch_type, top_of_inning, pitcher_mlb_id, batter_mlb_id, venue_city, inPlay, new_role =
  step_normalize(all_predictors())

summary(log_recipe)
```

```
## # A tibble: 29 x 4
##    variable                type     role       source
##    <chr>                   <chr>    <chr>      <chr>
##  1 pitch_type              nominal  ID         original
```

```
##   2 three_plus             numeric predictor original
##   3 batter_stance          numeric predictor original
##   4 pitcher_throws         numeric predictor original
##   5 strikes                numeric predictor original
##   6 balls                  numeric predictor original
##   7 outs                   numeric predictor original
##   8 pitch_plate_location_x numeric predictor original
##   9 pitch_plate_location_z numeric predictor original
## 10 pitch_initial_speed    numeric predictor original
## # ... with 19 more rows
```

```r
# setting model to logistic regression
log_mod <-
  logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")
```

```r
# setting workflow
log_workflow <-
  workflow() %>%
  add_model(log_mod) %>%
  add_recipe(log_recipe)
```

```r
# fitting the model to the training data
log_fit <-
  log_workflow %>%
  fit(data = log_train)
```

```r
# attaching the predictions to the actual dataset
trainpredict <- log_fit %>% predict(new_data = log_train) %>%
  bind_cols(log_train)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```r
trainpredict <- log_fit %>% predict(new_data = log_train, type="prob") %>%
  bind_cols(trainpredict)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```r
# assessing the accuracy of the model on training data, will do more detailed analysis on
# testing results
metrics(trainpredict, is_in_play, .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.670
## 2 kap      binary         0.224
```

```r
# confusion matrix
trainpredict %>%
  conf_mat(is_in_play, .pred_class)
```

```
##            Truth
## Prediction InPlay   Not
##     InPlay  43931 21053
##     Not      5140  9279
```

```r
# attaching the prediction on previously unseen testing data
testpredict <- log_fit %>% predict(new_data = log_test) %>%
  bind_cols(log_test)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```r
testpredict <- log_fit %>% predict(new_data = log_test, type="prob") %>%
  bind_cols(testpredict)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```r
# metrics for test data. We see an accuracy of 67.7%, which seems good, except for the fact
# that the number of pitches put in play is around 65%, so we have not improved the model
# much beyond chance guessing. This is reflected in the Kappa of only 0.235. There is some
# predictive power, but significantly less than our random forest.
metrics(testpredict, is_in_play, .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.678
## 2 kap      binary         0.235
```

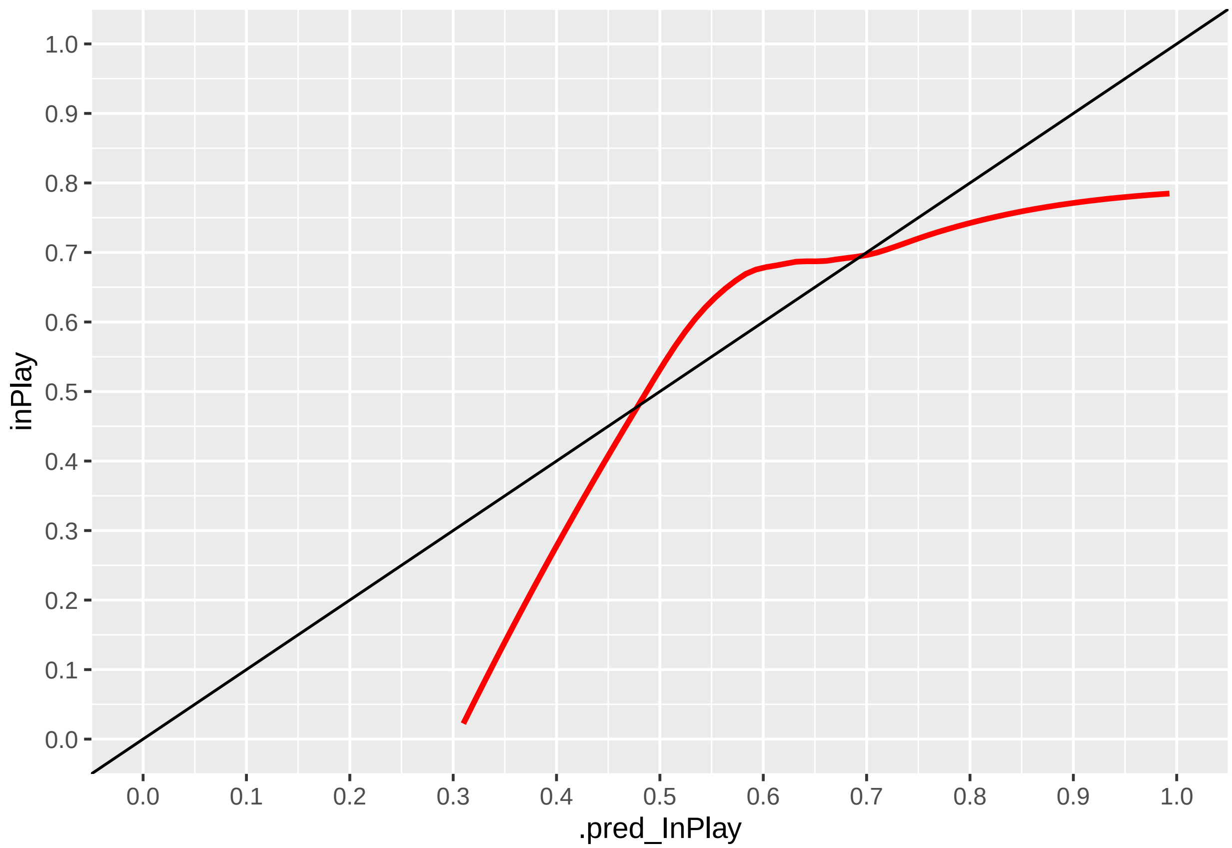
```r
# confusion matrix for the testing data
testpredict %>%
  conf_mat(is_in_play, .pred_class)
```

```
##            Truth
## Prediction InPlay   Not
##     InPlay  11121  5142
##     Not      1254  2334
```

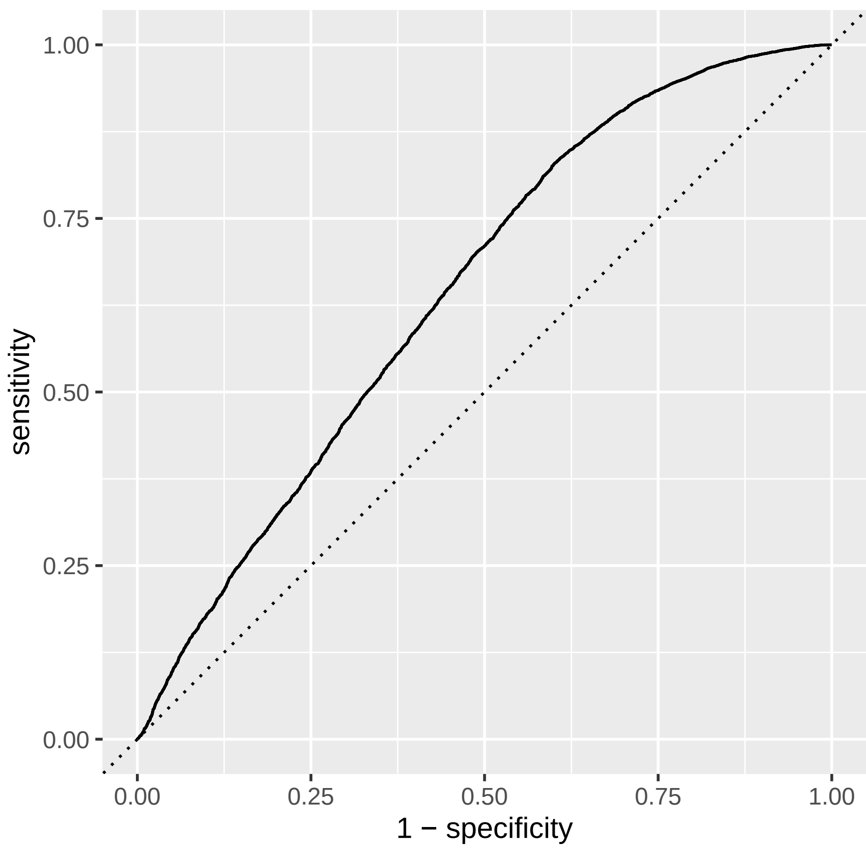
```r
# we get a calibration plot that indicates our prediction probabilities do not model to the
# data anywhere near as well as the random forest model
testpredict %>%
arrange(.pred_InPlay) %>%
ggplot(aes(x = .pred_InPlay, y = inPlay)) +
scale_y_continuous(limits = c(0, 1), breaks = seq(0, 1, by = 0.1)) +
scale_x_continuous(limits = c(0, 1), breaks = seq(0, 1, by = 0.1)) +
geom_smooth(aes(x = .pred_InPlay, y = inPlay), color = "red", se = F, method = "loess") +
geom_abline()
```

```
## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 9 rows containing missing values (geom_smooth).
```




```
# ROC-AUC plot, in case we want to hone the model to have better accuracy via avoiding false
# positives or false negatives
roc_data <- roc_curve(testpredict, truth = is_in_play, .pred_InPlay)
roc_data %>%
  ggplot(aes(x = 1 - specificity, y = sensitivity)) +
  geom_path() +
  geom_abline(lty = 3) +
  coord_equal()
```



4. Evalutating the Models and Next Steps

```
# When comparing the random forest model and the logistic regression model, it is very clear
# that the random forest is to be preferred for modeling on a new dataset. Both had solid
# accuracy ratings, with the random forest clocking a 74% overall accuracy on previously
# unseen data as opposed to the logistic regression's 68%. The Kappa for the random forest
# also comes in significantly better than the logistic regression, which was not surprising
# to me when I chose the two models. Had the data been more clearly linear, the logistic
# regression would have been a better comparison. The random forest really shines in its
# handling of non-linear data, making it optimal for this task. We also see that the random
# forest is much stronger at gauging estimated probability of any given pitch being put in
# play than the logistic regression, as we see via the two calibration plots.

# It is worth noting that we would be unlikely to include every single variable in any
# operational model, as some variables are likely to provide no real effect on our outcome.
# But, for purposes of simplicity, I have omitted variable selection and included almost
# every variable, with a few exceptions of variables that seemed highly unlikely to
# contribute any significant predictive power. Given more time and data, I would want to
# pursue further variable selection and testing, and we may be able to improve overall
# accuracy with these additional steps.

# Further, I wanted to pursue modeling the data via XGboost or SVM modeling, but the excess
# computing power seemed ill-suited for the task. Given more time or computing power, these
```

```
# methodologies would also be likely to boost our overall accuracy by a few percentage
# points. Lastly, we would want to consider the end goal for our model and the relative cost
# of false positives and negatives.
```