Modeling Called Strike Probability for NCAA data

```
#library management
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()
```

```
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/
```

```
library(readxl)
library(ggrepel)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(mgcv)
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
## This is mgcv 1.8-40. For overview type 'help("mgcv-package")'.
```

```
#base dataframe / to be updated with more data as needed. For now using 2019 data from big ten only
B1G_Baseball_Data_2019 <- read_excel("~/B1G Baseball Data 2019.xlsx")
data <- B1G_Baseball_Data_2019
```

```
#new dataframe management
endpa <- data %>% filter(LastPA == 1)
batters <- endpa %>% group_by(Batter)
pitchdata <- data %>% select(EventID, BatSide, Balls, Strikes, PitchType, PitchX, PitchY, PitchSide, Del
# interested specifically in pitches the umpire made a call on. Don't care about BIP
pitchestaken <- pitchdata %>% filter(PitchResult == "Ball" | PitchResult == "StrikeTaken")
pitchestaken <- pitchestaken %>% mutate(PitchResult = ifelse(PitchResult == "StrikeTaken", "Strike", "Ba
                                          Strike = ifelse(PitchResult == "Strike", 1,0),
                                        Ball = ifelse(PitchResult == "Ball",1,0),
                                        PitchType_Binary = ifelse(PitchType == "Fastball", 1, 0)) %>%
  mutate_at(6:7, as.numeric) %>% na.omit
```

```
## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion
```

```
pitchestaken <- pitchestaken %>% mutate(
  PitchResult = as.factor(case_when(
    PitchResult == "Strike" ~ "Strike",
    PitchResult == "Ball" ~ "Ball")),

    BatSide = ifelse(BatSide == "R", 1, 0),
    PitchSide = ifelse(PitchSide == "R", 1, 0),
    Delivery = ifelse(Delivery == "WindUp", 1, 0)) %>% na.omit()
```

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


model_recipe <-
  recipe(PitchResult ~ ., data = pitches_train) %>%
  update_role(EventID, PitchType, Strike, Ball, new_role = "ID") %>%
  step_normalize(all_predictors())

summary(model_recipe)
```

```
## # A tibble: 13 x 4
##    variable         type    role      source
##    <chr>            <chr>   <chr>     <chr>
##  1 EventID          nominal ID        original
##  2 BatSide          numeric predictor original
##  3 Balls            numeric predictor original
##  4 Strikes          numeric predictor original
##  5 PitchType        nominal ID        original
##  6 PitchX           numeric predictor original
##  7 PitchY           numeric predictor original
##  8 PitchSide        numeric predictor original
##  9 Delivery         numeric predictor original
## 10 Strike           numeric ID        original
## 11 Ball             numeric ID        original
## 12 PitchType_Binary numeric predictor original
## 13 PitchResult      nominal outcome   original
```

```
rf_mod <-
  rand_forest() %>%
  set_engine("ranger") %>%
  set_mode("classification")
```

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

```
rf_fit_strikeprob <-
  rf_workflow %>%
  fit(data = pitches_train)
```

```
rfpredict <- rf_fit_strikeprob %>% predict(new_data = pitches_train) %>%
  bind_cols(pitches_train)

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

```
metrics(rfpredict, PitchResult, .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.974
## 2 kap      binary         0.938
```

```
rftestpredict <- rf_fit_strikeprob %>% predict(new_data = pitches_test) %>%
  bind_cols(pitches_test)

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

metrics(rftestpredict, PitchResult, .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.967
## 2 kap      binary         0.920
```

```
rftestpredict %>%
  conf_mat(PitchResult, .pred_class)
```

```
##           Truth
## Prediction Ball Strike
##     Ball   3746    121
##     Strike   58   1478
```

```
heatmapdata <- rfpredict %>% select(PitchX, PitchY, .pred_Strike)
heatmapdata <- heatmapdata %>% filter(PitchX > -1.5 & PitchX < 1.5)
heatmapdata <- heatmapdata %>% filter(PitchY > 1 & PitchY < 4)
#heatmapdata <- heatmapdata %>% filter(.pred_Strike >.5)

library(MBA)
heatmapdata=heatmapdata[ order(heatmapdata[,1], heatmapdata[,2],heatmapdata[,3]), ]
```

```
## Warning in xtfrm.data.frame(x): cannot xtfrm data frames
```

```
## Warning in xtfrm.data.frame(x): cannot xtfrm data frames
```

```
## Warning in xtfrm.data.frame(x): cannot xtfrm data frames
```

```
mba.int <- mba.surf(heatmapdata, 300, 300, extend=T)$xyz.est
library(fields)
```

```
## Loading required package: spam
```

```
## Spam version 2.8-0 (2022-01-05) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
```

```
##
## Attaching package: 'spam'
```

```
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve
```

```
## Loading required package: viridis
```

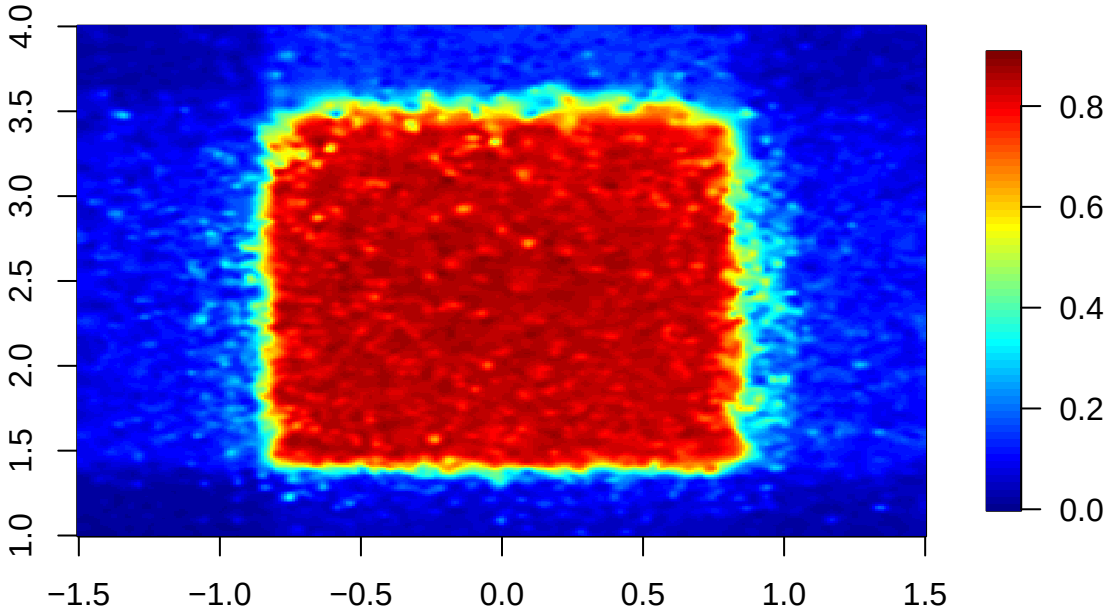
```
## Loading required package: viridisLite
```

```
##
## Attaching package: 'viridis'

## The following object is masked from 'package:scales':
##
##     viridis_pal

##
## Try help(fields) to get started.
```

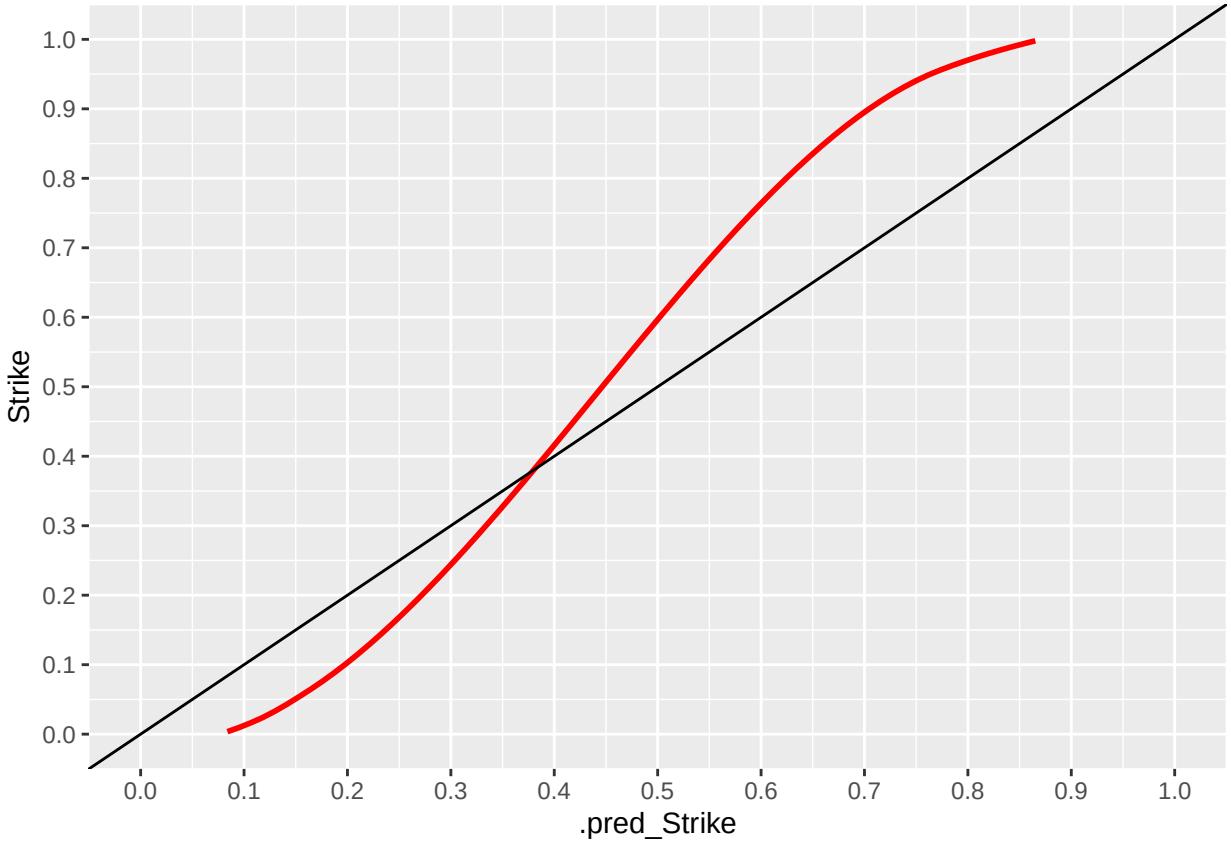
```
fields::image.plot(mba.int)
```




```
rfpredict %>%
arrange(.pred_Strike) %>%
ggplot(aes(x = .pred_Strike, y = Strike)) +
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_Strike, y = Strike), color = "red", se = F, method = "loess") +
# you can use stat_smooth in place of geom_smooth
geom_abline()
```

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

## Warning: Removed 11 rows containing missing values (geom_smooth).
```


Strike
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
.pred_Strike