MLB Pitching Classifier

Albert Wiryawan (netid@illinois.edu)

12/2/2020

Table of Contents

Abstract

In this data exploration, MLB data from the 2020 regular and post season is collected. From here, the statistical learning method of k nearest neighbors is used to generate a model to classify the different types of pitches as either SI, SL, CH, FF, CU, FC, FS. Two approaches are used in order to optimize this model. Firstly, in order to decide on a value of k for the model varying models were built for k = 1,10,100 where their percent errors were found to be 17.93%, 14.84%, and 16.87% respectively. As a result, the model's hyper parameter was chosen to be 10. Then, further improvement in the model was obtained by normalizing the data as to prevent unit and outlier effects. The model error from the normalized model was found to be 10.08 which results in an overall accuracy of 89.92%.

Introduction

In the game of baseball, the batter will guess the type of pitch by visually analyzing the movement of the ball. By doing so, the batting athlete can understand how to better hit the ball for the current situation of the field. For baseball athletes, this type of intuition is developed from years of practice. Utilizing data and statistical learning, however, a classifying model can be made to predict the pitch type given different attributes of a pitch. With the improvement of computers, data quality, and statistical understanding, baseball is a sport that is becoming more and more data driven. Being able to detect tendencies of a

pitch type given certain amounts of data can help train and better the baseball athletes of today.

Methods

In this data exploration we will be utilizing the statistical learning method entitled: K-nearest neighbors or knn. In order to create a classification model that is effective in characterizing the type of pitch it is based on the data given. To do this, several initial models will be trained on an attribute filtered version of the data set collected from pitches thrown during the 2020 regular season of MLB, varying the hyper parameter k. Once created, these models will be tested with the pitches that are thrown during the post season to obtain a metric for accuracy of the models.

Data

With an initial look at the data,

```
head(pitches_2020_regular)
count_na = function(x) {
  sum(is.na(x))
}
sapply(pitches_2020_regular, count_na)
sapply(pitches_2020_post, count_na)

na.omit(pitches_2020_post)
na.omit(pitches_2020_regular)
```

```
## # A tibble: 6 x 25
##   pitch_type game_date  release_speed release_pos_x release_pos_y
release_pos_z
##   <chr>      <date>             <dbl>         <dbl>         <dbl>
<dbl>
## 1 SI         2020-07-23         100           -1.53         -1.53
6.27
## 2 SI         2020-07-23          99.1         -1.47         -1.47
6.28
## 3 SL         2020-07-23          88.1         -1.66         -1.66
6.31
## 4 SI         2020-07-23          99.3         -1.22         -1.22
6.42
## 5 SI         2020-07-23          98.4         -1.44         -1.44
6.29
## 6 CH         2020-07-23          92.1         -1.17         -1.17
6.09
## # … with 19 more variables: player_name <chr>, batter <dbl>, pitcher
<dbl>,
```

```
## #   zone <dbl>, stand <chr>, p_throws <chr>, pfx_x <dbl>, pfx_z <dbl>,
## #   plate_x <dbl>, plate_z <dbl>, vx0 <dbl>, vy0 <dbl>, vz0 <dbl>, ax
<dbl>,
## #   ay <dbl>, az <dbl>, effective_speed <dbl>, release_spin_rate <dbl>,
## #   release_extension <dbl>

##        pitch_type         game_date     release_speed     release_pos_x
##                 0                 0                 0                 0
##     release_pos_y     release_pos_z       player_name            batter
##                 0                 0                 0                 0
##           pitcher              zone             stand          p_throws
##                 0                 0                 0                 0
##             pfx_x             pfx_z           plate_x           plate_z
##                 0                 0                 0                 0
##               vx0               vy0               vz0                ax
##                 0                 0                 0                 0
##                ay                az   effective_speed release_spin_rate
##                 0                 0                 0               294
## release_extension
##               515

##        pitch_type         game_date     release_speed     release_pos_x
##                 0                 0                 0                 0
##     release_pos_y     release_pos_z       player_name            batter
##                 0                 0                 0                 0
##           pitcher              zone             stand          p_throws
##                 0                 0                 0                 0
##             pfx_x             pfx_z           plate_x           plate_z
##                 0                 0                 0                 0
##               vx0               vy0               vz0                ax
##                 0                 0                 0                 0
##                ay                az   effective_speed release_spin_rate
##                 0                 0                 0                 0
## release_extension
##                13
```

First, we notice that the data set contains na values that must be dealt with in order to train the model. In addition, there is only a certain number of features from the data set that will help in the understanding of velocity, location, and movements the ball which is what we will be focusing on. As a result, we have to turn the list of pitch_types into factors of class to help in the classification.

```
pitches_reg = pitches_2020_regular[, c(1,4,5,6,13:24)]
pitches_post = pitches_2020_post[, c(1,4,5,6,13:24)]
pitches_reg$pitch_type = factor(pitches_reg$pitch_type, level= c("SI",
"SL", "CH", "FF", "CU", "FC", "FS"))
pitches_post$pitch_type = factor(pitches_post$pitch_type, level=
c("SI", "SL", "CH", "FF", "CU", "FC", "FS"))
```

Modeling

The models with varying tuning parameters of k are created varying from 1,10,100.

```
knn_model1 = knn3(pitch_type ~ ., data = pitches_reg, k = 10)
knn_model2 = knn3(pitch_type ~ ., data = pitches_reg, k = 100)
knn_model3 = knn3(pitch_type ~ ., data = pitches_reg, k = 1)
```

The lowest calculated error was found when k =10. So we will utilize this hyper parameter and utilize it to try and further improve the model that is currently in possession. To do this, we want to try and create a scaled model as the different collected data values can really drastically in magnitude due to units.

```
tst_pred_un1 = predict(knn_model1, pitches_post, type = "class")
tst_pred_un2 = predict(knn_model2, pitches_post, type = "class")
tst_pred_un3 = predict(knn_model3, pitches_post, type = "class")

calc_class_err = function(actual, predicted) {
  mean(actual != predicted)
}

error1 = calc_class_err(pitches_post$pitch_type, tst_pred_un1)
error2 = calc_class_err(pitches_post$pitch_type, tst_pred_un2)
error3 = calc_class_err(pitches_post$pitch_type, tst_pred_un3)
```

Results

Based on the models produced above the error in classification for the various different splits indicated by the hyper parameter value k, the values of error were found to be 17.93%, 14.84%, and 16.87% for k = 1, 10, 100 respectively. This trend shows that the k value of 10 is the best for this model. In order to further improve the accuracy of the model, however, the model has their parameters scaled due to the different potential magnitude in units. As a result, a scaled model is produced for a k-value of 10 and tested against the data set for pitches in the post season and obtained a 10.08% error rate. This further improvement in the model lead to the final obtained model, thus having an overall accuracy of 89.92% accuracy. ***

Discussion

In this exploration of k nearest neighbors, two approaches were taken in order to improve the model. First, the tuning of hyper parameters is first assessed by trying values of k distance. For the statistical learning method of knn for classification, the tuning parameter k resembles the amount of nearest neighbors chosen to compare an overall distance to see which class is closest. In this case, the tuning parameter k = 10 was chosen as it obtained the lowest percentage of classification error. In addition to this, the model with the chosen

hyper parameter is scaled in order to improve accuracy. This is done because the difference in the units of the collected data could present inaccuracies in the model. By, changing numeric attributes to a common scale, large distortion in data and outliers will no longer have a negative effect on the model.
