Pitch Prediction Code

Data Pull

Data Pull From PITCHf/x api

```
#This is going to set up a database for our data...takes a long time to run
#db <- src_sqlite("pitchfx.sqlite3", create = T)
#scrape(start = "2011-03-31", end = "2011-09-28", connect = db$con)
#con = dbConnect(RSQLite::SQLite(), dbname="pitchfx.sqlite3")
#combined = dbGetQuery(con,"SELECT * FROM atbat INNER JOIN pitch ON (atbat.num = pitch.num AND atbat.ur
```

```
pitch_data = read.csv("pitches",stringsAsFactors=FALSE) #Read in data
game_pitch_data = pitch_data[pitch_data$game_pk == 286874,]
```

Data Cleaning

Clean Up Pich Types

```
pitchTypeClean = function(x) {
  if (x == "FF"   |  x == "FC" |  x == "FA"  |  x == "SI"  |  x == "FS"  |  x == "FT" |  x == "SI" |  x
    return ("FB")
  }
  if (x == "CU"   |  x == "SL" |  x == "CB" |  x == "KN" |  x == "KC"){
    return("BB")
  }
  if (x == "CH"   |  x == "EP"){
   return("CH")
  }
  if (x == "UN"){
   return("unidentified")
  }
  if (x == "AB"){
    return("automatic ball")
  }
  else {
    return("")
  }
}
game_pitch_data$pitchLabel = as.vector(unlist(lapply(game_pitch_data$pitch_type,pitchTypeClean)))
pitch_data$pitchLabel = as.vector(unlist(lapply(pitch_data$pitch_type,pitchTypeClean)))

types = c("FB","BB","CH")

game_pitch_data = filter(game_pitch_data,pitchLabel %in% types)
pitch_data = filter(pitch_data,pitchLabel %in% types)
```

Visualize Pitch Distribution

```
barplot(table(pitch_data$pitchLabel),col="steelblue",ylab = "Count",main="Pitch Type Distribution Imbala
```

Moving Average Feature Engineering

```
mav <- function(x,n){
  stats::filter(x,rep(1/n,n), sides=1)
  }

mavback <- function(x,n){
  a<-mav(x,1)
  b<-mav(x,(n+1))
  c<-(1/n)*((n+1)*b - a)
  return(c)
}

library(zoo)

roll <- function(x, n) {
   if (length(x) <= n) NA
   else rollapply(x, list(-seq(n)), mean, fill = NA)
}

mavcols = c("game_pk","pitcher_id","x","y","start_speed","end_speed","sz_top","sz_bot","pfx_x","pfx_z",


pitch_data <- pitch_data %>%
  group_by(game_pk) %>%
  group_by(pitcher_id) %>%
  mutate(x_3_Avg = mavback(x,3),
         x_1_Avg = mavback(x,1),
         y_3_Avg = mavback(y,3),
         y_1_Avg = mavback(y,1),
         start_speed_3_Avg = mavback(start_speed,3),
         start_speed_1_Avg = mavback(start_speed,1),
         end_speed_3_Avg = mavback(end_speed,3),
         end_speed_1_Avg = mavback(end_speed,1),
         sz_top_3_Avg = mavback(sz_top,3),
         sz_top_1_Avg = mavback(sz_top,1),
         sz_bot_3_Avg = mavback(sz_bot,3),
         sz_bot_1_Avg = mavback(sz_bot,1),
         pfx_x_3_Avg = mavback(pfx_x,3),
         pfx_x_1_Avg = mavback(pfx_x,1),
         pfx_z_3_Avg = mavback(pfx_z,3),
         pfx_z_1_Avg = mavback(pfx_z,1),
         px_3_Avg = mavback(px,3),
         px_1_Avg = mavback(px,1),
         pz_3_Avg = mavback(pz,3),
         pz_1_Avg = mavback(pz,1),
         vx0_3_Avg = mavback(vx0,3),
         vx0_1_Avg = mavback(vx0,1),
         vy0_3_Avg = mavback(vy0,3),
         vy0_1_Avg = mavback(vy0,1),
         vz0_3_Avg = mavback(vz0,3),
         vz0_1_Avg = mavback(vz0,1),
         ax_3_Avg = mavback(ax,3),
         ax_1_Avg = mavback(ax,1),
         az_3_Avg = mavback(az,3),
         az_1_Avg = mavback(az,1),
         ay_3_Avg = mavback(ay,3),
         ay_1_Avg = mavback(ay,1),
         break_length_3_Avg = mavback(break_length,3),
         break_length_1_Avg = mavback(break_length,1),
         break_y_3_Avg = mavback(break_y,3),
         break_y_1_Avg = mavback(break_y,1),
         break_angle_3_Avg = mavback(break_angle,3),
         break_angle_1_Avg = mavback(break_angle,1),
         last_Zone = mavback(zone,1),
         spin_dir_3_Avg = mavback(spin_dir,3),
         spin_dir_1_Avg = mavback(spin_dir,1),
         spin_rate_3_Avg = mavback(spin_rate,3),
         spin_rate_1_Avg = mavback(spin_rate,1))

#test = pitch_data[,mavcolstest] %>% group_by(game_pk) %>% group_by(pitcher_id) %>% mutate_all(mavback(
```

```
#This will label each pitch with the data from the previous pitch in that specific count
# i.e. If the last pitch the pitcher threw when the count was 3-2 was 86 mph, start_speed_Avg_Count = 8

pitch_data = transform(pitch_data,count=paste0(balls,strikes))

pitch_data <- pitch_data %>%
  group_by(game_pk) %>%
  group_by(pitcher_id) %>%
  group_by(count) %>%
  mutate(x_1_Avg_count = mavback(x,1),
         y_1_Avg_count = mavback(y,1),
         start_speed_1_Avg_count = mavback(start_speed,1),
         end_speed_1_Avg_count = mavback(end_speed,1),
         sz_top_1_Avg_count = mavback(sz_top,1),
         sz_bot_1_Avg_count = mavback(sz_bot,1),
         pfx_x_1_Avg_count = mavback(pfx_x,1),
         pfx_z_1_Avg_count = mavback(pfx_z,1),
         px_1_Avg_count = mavback(px,1),
         pz_1_Avg_count = mavback(pz,1),
         vx0_1_Avg_count = mavback(vx0,1),
         vy0_1_Avg_count = mavback(vy0,1),
         vz0_1_Avgv = mavback(vz0,1),
         ax_1_Avg_count = mavback(ax,1),
         az_1_Avg_count = mavback(az,1),
         ay_1_Avg_count = mavback(ay,1),
         break_length_1_Avg_count = mavback(break_length,1),
         break_y_1_Avg_count = mavback(break_y,1),
         break_angle_1_Avg_count = mavback(break_angle,1),
         last_Zone_count = mavback(zone,1),
         spin_dir_1_Avg_count = mavback(spin_dir,1),
         spin_rate_1_Avg_count = mavback(spin_rate,1))
```

Visualize One Pitcher

```
rand_pitcher = sample(unique(pitch_data$pitcher_id),1,replace = FALSE)
viz_data = pitch_data[pitch_data$pitcher_id==rand_pitcher,]
```

```r
rand_game = sample(unique(viz_data$game_pk),1,replace=FALSE)
viz_data = viz_data[viz_data$game_pk==rand_game,]
viz_data = viz_data[,c("pitcher_id","game_pk","pitchLabel","vx0_3_Avg","spin_rate_3_Avg")]
x  = viz_data$vx0_3_Avg
y = viz_data$spin_rate_3_Avg
lab = viz_data$pitchLabel

ggplot(viz_data,aes(x=vx0_3_Avg,y=spin_rate_3_Avg)) +
  geom_point(aes(col=pitchLabel, shape=pitchLabel),size=4) +
  ggtitle("One Pitcher in One Game Type Distribution") +
  xlab("Avg Three Pitch X-Velocity") + ylab("Avg Three Pitch Spin Rate")
```

More Cleaning

```r
#This removes columns which would be unavalible at-bat (i.e. real-time data)

#remove runner data
pitch_data[ ,c(75:127)] = list(NULL)

pitch_data$first = pitch_data$on_1b > 0
pitch_data$second = pitch_data$on_2b > 0
pitch_data$third = pitch_data$on_3b > 0

pitch_data = pitch_data %>% replace_na(list(first = FALSE, second = FALSE, third = FALSE))

#remove current pitch stats
mavcols = c("x","y","start_speed","end_speed","sz_top","sz_bot","pfx_x","pfx_z","px","pz","vx0","vy0","

pitch_data[,mavcols] = list(NULL)#This removes columns which would be unavalible at-bat (i.e. real-time
pitch_data[,c(1,2)] = list(NULL)
pitch_data$pitchLabel =factor(pitch_data$pitchLabel)
pitch_data$date = as.Date(pitch_data$date)
pitch_data$pitchLabel =factor(pitch_data$pitchLabel)
pitch_data$count =factor(pitch_data$count)
pitch_data$b_height =factor(pitch_data$b_height)
pitch_data$inning =factor(pitch_data$inning)
pitch_data$stand =factor(pitch_data$stand)
pitch_data$first = factor(pitch_data$first)
pitch_data$second = factor(pitch_data$second)
pitch_data$third = factor(pitch_data$third)
```

Test Train Split

```r
index = round(length(unique(pitch_data$game_pk)) * 0.70)
games = unique(pitch_data$game_pk)
game_index = games[index]
game = pitch_data[pitch_data$game_pk == game_index,]
date = game$date[1] #2011-08-11
date
pitch_data$date = as.Date(pitch_data$date)
pitch_data_train = pitch_data[pitch_data$date <= date,]
pitch_data_test = pitch_data[pitch_data$date > date,]
```

Modeling and Analysis

Baseline Accuracy

```
fb = as.factor(replicate(214044,"FB"))
confusionMatrix(fb,test$pitchLabel)
```

Random Forest

```
rf.mod = randomForest(pitchLabel ~ . -uid -game_pk -year-date-team_id_b-team_id_p-pitcher_id,data=train
preds = predict(rf.mod,newdata = test)
confusionMatrix(preds,test$pitchLabel)
```

Feature Selection using Random Forest Importance

```
#This Plot Shows the 10 Most Important Variables
varImpPlot(rf.mod,type=2,n.var=10)
var_select = importance(rf.mod, type=2)

selected_feats = c("pitchLabel","b_height", "inning","count","pcount_pitcher","vx0_3_Avg","spin_rate_3_
                   "end_speed_3_Avg","vx0_1_Avg","vz0_3_Avg","vy0_3_Avg")
```

Modeling With Selected Features

New Training/Test Sets

```
train_selected = train[,selected_feats]
test_selected = test[,selected_feats]
```

Random Forest Deeper

```
loss <- matrix(c(0, 1, 1, 1,0,1,2,2,0), ncol=3)
cart = rpart(pitchLabel ~ . ,data=train_selected,parms=list(loss=loss))
preds = predict(cart,newdata = test_selected,type="class")
confusionMatrix(preds,test_selected$pitchLabel)
```

LDA

```
#Must Remove Height because of multivariate normal assumption
lda.mod = lda(pitchLabel ~ .-b_height,data=train_selected)
lda_preds = predict(lda.mod,newdata = test_selected,response="class")
confusionMatrix(lda_preds$class,test$pitchLabel)
```

Neural Net (basically useless)

```
## This ends up just making a baseline classifier :(

# Prep for Keras
trainX <- model.matrix(pitchLabel ~ . , data = train_balance) # conver to one-hot matrix
trainX = trainX[,-c(1:5) ]
trainY <- model.matrix(~ pitchLabel - 1, data = train_balance) # -1: no intercept term

testX <- model.matrix(pitchLabel ~ . , data = test)
testX = testX[,-c(1:5)]
testY <- model.matrix(~ pitchLabel - 1, data = test)
```

```r
nn_mod_1 <- keras_model_sequential()

nn_mod_1 %>%
  layer_dense(units = 100, activation = "relu", input_shape = c(126)) %>% # Adding the hidden layer
  layer_dense(units = 3, activation = "softmax") # adding the output layer
summary(nn_mod_1)

nn_mod_1 %>% compile(
  optimizer = "rmsprop",
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)

tic("Neural Net 1:")
training_history <- nn_mod_1 %>%
  fit(trainX, trainY,
      epochs = 20, validation_split = 0.2)
toc()


nn_preds = predict_classes(nn_mod_1,testX)
confusionMatrix(nn_preds,test$pitchLabel)
```

Final Model- a random forest of pitchers

```r
cf_mat = confusionMatrix(unique(pitch_data$pitchLabel),unique(pitch_data$pitchLabel))
cf_mat$table[1,1] = 0; cf_mat$table[2,2] = 0; cf_mat$table[3,3] = 0
confusion_table = cf_mat$table

n = length(unique(pitch_data$pitcher_id))
chosen_pitcher_ids = sample(unique(pitch_data$pitcher_id),n,replace = FALSE)

for (i in c(1:n)) {
  chosen_pitcher_id = chosen_pitcher_ids[i]

  one_pitcher = pitch_data[pitch_data$pitcher_id == chosen_pitcher_id,]

  index = round(length(unique(one_pitcher$game_pk)) * 0.70)

  games = unique(one_pitcher$game_pk)
  game_index = games[index]
  game = one_pitcher[one_pitcher$game_pk == game_index,]
  date = game$date[1]
  date
  one_pitch_data_train = one_pitcher[one_pitcher$date <= date,]
  one_pitch_data_test = one_pitcher[one_pitcher$date > date,]

  one_pitch_data_train = one_pitch_data_train[complete.cases(one_pitch_data_train),]
  one_pitch_data_test = one_pitch_data_test[complete.cases(one_pitch_data_test),]

  one_pitch_data_train$pitchLabel = factor(one_pitch_data_train$pitchLabel)
  #one_pitch_data_test$pitchLabel = factor(one_pitch_data_test$pitchLabel)
  if (nrow(one_pitch_data_train) < 200) {
    next
  }

  if (length(unique(one_pitch_data_train$pitchLabel))<2) {
    next
  }


  #rf.mod.one = randomForest(pitchLabel ~ . -uid -game_pk -year-date-team_id_b-team_id_p-pitcher_id,dat
  dont = c("pitchLabel","uid", "game_pk", "year","date","team_id_b","team_id_p","pitcher_id")

  tuneTrain = dplyr::select(one_pitch_data_train,-one_of(dont))
  rf.mod.one = tuneRF(tuneTrain, one_pitch_data_train[,c("pitchLabel")],ntreeTry=50, doBest = TRUE)

  tuneTest = dplyr::select(one_pitch_data_test,-one_of(dont))
  preds.one = predict(rf.mod.one,newdata = tuneTest)

  cf_mat = confusionMatrix(preds.one,one_pitch_data_test$pitchLabel)
  #print(cf_mat$table)
  confusion_table = confusion_table + cf_mat$table


}

confusion_table
sum(diag(confusion_table)/sum(confusion_table)) #0.6609412
#Other metrics determined by hand
```