PML HR Probability Model 1

By Charlee Anne Rodriguez

Model 1: HR Probability Formula = (HRs last 20/20) x (Pitcher HR/9 / 9) x Ballpark Factor HR

**If batter has 0 HRs in the last 20 games, use batting average x total hits in the last 20 games\*\*\***

**MODEL 1 MODIFICATION NOTES:**

1. Consider barrel rate, launch angles, exit velocity for the new formula
   * Average Exit Velocity
   * Barrel Rates per Batted Baseball Events Brls/PA%
   * Launch Angle Sweet spot LA SwSp%
2. Consider if pitcher struggles with LHB or RHB

**Model 1 Modification #1:** HR Probability Formula = (HRs last 20/20) x (Pitcher HR/9 / 9) x Ballpark Factor HR x Barrel Rate Multiplier x LA SwSp Multiplier x Exit Velocity Adjustment

* + BRM Formula = 1 + (Brls/PA% / 10)
  + LASwSpM Formula = 1 if within 25 to 35 degrees otherwise 0.8
  + EVA = 1 + (Avg EV– 95) / 100

**August 2 Notes:**

As for pre-game results, model 1 versions 1 & 2 produced the same peak and low hitters. I cannot count Ozuna and Soler due to the pitcher data being off. Therefore I will test with other batters whom I did not place a bet on. **The model has a lot of emphasis on batter performance, will try to find ways to implement pitcher data a lot more.** I realized that there should be a success threshold to the model so I can see whether it is successful or not. Considering that there’s a Peak/Low hitters, I will average out who’s hitting the HR or not. I will add Peak/Mid/Low hitters, Peak will start at 10%+ for Model 1.1 and 20%+ for Model 1.2, Mid will start at 5%+ for model 1.1 and 10%+ for model 1.2 and Low will start at any place below that.

The results of low chance players being the highest percentage is a little funny so far, homeruns really are a chance play at this point of the program build. Although I am impressed that the total homerun hit percentage based on the total amount of picks being 46% is really good especially for the first test. I want to **cap the amount of players to 15 per test** run aka every day.

Model 1.2

print("Welcome to HR Probability Model 1")  
calculate = True  
while calculate:  
 batter\_hr = float(input("Enter the batter's HRs in the last 20 games: "))  
 pitcher\_hr = float(input("Enter the pitcher's total HRs in the most current season: "))  
 bp\_rate = float(input("Enter the ballpark HR rate in the most current season: "))  
 brls = float(input("Enter the batter's Barrel Rate: "))  
 angle = float(input("Enter the batter's Launch Angle Sweet-Spot: "))  
 avg\_ev = float(input("Enter the batter's Average EV: "))  
  
 #barrel rate multiplier  
 brm = 1 + (brls / 10)  
 print("BRM is: ", brm)  
  
 #launch angle multiplier  
 a = 25  
 b = 35  
 if min(a, b) < angle < max(a, b):  
 lam\_sw = 1.0  
 else:  
 lam\_sw = 0.8  
 print("LAM is: ",lam\_sw)  
  
 #exit velocity adjustment  
 eva = 1 + ((avg\_ev - 95) / 100)  
 print("EVA is: ", eva)  
  
 hr\_formula = (batter\_hr / 20) \* (pitcher\_hr / 9 / 9) \* bp\_rate \* brm \* eva \* lam\_sw \* 100  
  
 print("The HR Probability is: ", hr\_formula, "%")  
 more\_calc = (input("Do you want to continue? Y/N"))  
 if more\_calc != "Y":  
 print("Thank you for using HR Probability Model 1")  
 calculate = False

**August 3, 2024**

Model 1.2 Notes – Testing 15 players who’s odds range from +400 and over. I want to test Model 1 for almost this whole month while continuing to read on how to improve the statistics and program. Peak/Mid/Low can be adjusted depending on the probabilities coming out.

Fixes to the probability model:

* Ballpark HR factor = HR Ballpark/Overall Factor. Instead of just HR Ballpark
* Use doink find pitcher’s two most used types. Find the batter’s average for both of those pitch types and make an average out of that value. Batter Pitch Type Average (BPT Avg)

**August 3 Notes:**

To make this program a little fancier, I plan to have it ask for how many players we are inputting data for. I want it to ask for the name of each player and then the regular calculations proceed. Although I don’t want it to immediately print results the way it is doing it now, I want it to print results for all the players once all the information has been input. Basically store all the information and boom. I wanted to add more pitching information to the calculations and decided to do BPT Average, basically how well the player does against the pitcher’s top 2 most-used pitch types. I want to test this current formula for longer and as I said: **I want to figure out a better statistical approach with the data that is currently being used. There can be a better formula with this data being used.**

**Peak/Mid/Low seems to adjust according to the players I pick**, I decided to try the BPT average calc to the August 2 results and the chances still shift much much higher when the player is just insanely good aka Judge, Guerrero, Seager etc. I don’t think there’s anything wrong with adjusting the PML or else I wouldn’t have a PML if I set it to one thing. The best I can do is order it from greatest to least in terms of percentages.

**The overall goal of this program is to consistently yield a high percentage of peak hitters actually hitting the homerun.** I know that there’s a lot of things to cover when it comes to predicting a homerun but I want there to be a consistently high percentage with the peak player picks. Having a high total % consistently would be also great overall.

**Model 1 Modification #2**:

HR Probability Formula = (HRs last 20/20) x (Pitcher HR/9 / 9) x Ballpark Factor HR x Barrel Rate Multiplier x LA SwSp Multiplier x Exit Velocity Adjustment x BPT Average

This day was a terrible day in general for homers but I want too adjust how I pick what batters I will use the numbers on. I’m also going to change a lot of the things I’m measuring. The formulas for all the multipliers have already been adjusted accordingly in the code as well

* Batter’s HRs in the last 10 games
* Pitcher’s HRs in the last 10 games
* BPT Average accounts for 3 pitch types instead of 2

The results for this day were terrible but it was an off day overall. Tomorrow should be a better day and again I will be more selective with the players I choose.

import numpy as np  
'''  
Model 1 uses a formula provided by ChatGPT but will be adjusted by me accordingly over time using following data:  
1. Batter HRs in the last 20 games  
2. Pitchers HRs against batter in the season  
3. Ballpark HR Rate  
4. Barrel Rate  
5. Launch Angle Sweet-Spot  
6. Average Exit Velocity  
7. Batter-Pitch Type Average  
  
HR Probability Formula = (Batter HRs / 20) x (Pitcher HRs / 9 / 9) x Ballpark HR Rate x   
BRM x EVA x LAM\_SW x  
  
This is the first model or version of this calculator and will be improved overtime  
date created: 8/2/2024  
latest modification date: 8/3/2024  
  
also refer to the partner word file for actual calculations and tests  
'''  
  
print("Welcome to HR Probability Model 1")  
calculate = True  
while calculate:  
 batter\_hr = float(input("Enter the batter's HRs in the last 10 games: "))  
 pitcher\_hr = float(input("Enter the pitcher's total HRs in the last 10 games: "))  
 bp\_rate = float(input("Enter the Ballpark HR rate in the most current season: "))  
 brls = float(input("Enter the batter's Barrel Rate: "))  
 angle = float(input("Enter the batter's Launch Angle Sweet-Spot: "))  
 avg\_ev = float(input("Enter the batter's Average EV: "))  
 type\_1, type\_2, type\_3 = map(float,input("Enter the batter's performance against the pitcher's top 3 pitch types: ").split())  
  
 #barrel rate multiplier  
 brm = 1 + brls  
 print("BRM is: ", brm)  
  
 #launch angle multiplier  
 a = 25  
 b = 35  
 if min(a, b) < angle < max(a, b):  
 lam\_sw = 1.1  
 else:  
 lam\_sw = 0.9  
 print("LAM is: ",lam\_sw)  
  
 #exit velocity adjustment  
 c = 90  
 d = 95  
 if avg\_ev >= 95:  
 eva = 1.2  
 elif avg\_ev >= 90:  
 eva = 1.1  
 else:  
 eva = 1.0  
 print("EVA is: ", eva)  
  
 #bpt average  
 bpt\_avg = (type\_1 + type\_2 + type\_3) / 3  
 print("BPT Avg is: ", bpt\_avg)  
  
 #main formula  
 hr\_formula = (batter\_hr / 10) \* (pitcher\_hr / 9 / 9) \* bp\_rate \* brm \* eva \* lam\_sw \* bpt\_avg \* 100  
  
 print("The HR Probability is: ", hr\_formula, "%")  
 more\_calc = (input("Do you want to continue? Y/N"))  
 if more\_calc != "Y":  
 print("Thank you for using HR Probability Model 1")  
 calculate = False

**August 4, 2024**

Modified a lot of the code with the changes that I’ve been wanting to implement such as asking for how many players to calculate and running the calculations that much and then printing it all at the end. Although I want to add a syntax where if the user input is wrong then they can try again because it becomes too tedious adding in numbers and then you fail one time then the whole code ends.

I’m tired of manually inputting the information so I’m going to learn how to do document scraping. I still want to do the research for the baseball stats manually. Until I create a program where the results are very good with the PML then I can go to web scraping as well.

I’m also going to add where the program asks for the ball field instead so that I don’t have to keep putting that same data every single time. It’s the only data that never changes especially as of this season, so might as well.

Implemented a sorted output so I don’t have to manually rank the results from greatest to least.

Okay I got 0% today and so therefore I plan to only pick from the top 50 homerun hitters or if their barrel rate, exit velocity, and launch angles are good. Cycle through the names everyday. I understand that homeruns are by chance and basically anybody can hit them depending on the day but at least with the top hitters we can expect it more than usual.

print("Welcome to HR Probability Model 1")  
calc\_num = int(input("Enter how many players you want to calculate a probability for: "))  
  
# all the ballpark factors  
def bp\_factor(stadium):  
 bp\_factors = {  
 "American Family": 1.162,  
 "Angel": 1.109,  
 "Busch": 0.921,  
 "Chase": 0.830,  
 "Citi": 1.021,  
 "Citizens Bank": 1.139,  
 "Comerica": 0.857,  
 "Coors": 0.964,  
 "Dodger": 1.232,  
 "Fenway": 0.935,  
 "Global Life": 1.149,  
 "Great American": 1.212,  
 "Guarenteed": 1.000,  
 "Kauffman": 0.819,  
 "LoanDepot": 0.880,  
 "London": 0.679,  
 'Minute': 1.070,  
 'Nationals': 1.010,  
 'Oakland': 0.847,  
 'Oracle': 0.825,  
 'Oriole': 0.939,  
 'Petco': 1.052,  
 'PNC': 0.850,  
 'Progressive': 0.885,  
 'Rogers': 1.020,  
 'Target': 1.020,  
 'T-Mobile': 1.044,  
 'Tropicana': 1.010,  
 'Truist': 1.079,  
 'Wrigley': 0.949,  
 'Yankee': 1.202  
 }  
 bp\_num = float(bp\_factors.get(stadium))  
 return bp\_num  
  
# sorts the hr/player list  
def sort(list1, list2):  
 combine = list(zip(list1, list2))  
 combined\_sort = sorted(combine, key=lambda x: x[0], reverse=True)  
 list1sort, list2sort = zip(\*combined\_sort)  
 return list(list1sort), list(list2sort)  
  
def hr\_prob():  
 player\_list = []  
 hr\_list = []  
 for x in range(calc\_num):  
 player\_name = input("What is the name of the player?: ")  
 batter\_hr = float(input("Enter the batter's HRs in the last 10 games: "))  
 pitcher\_hr = float(input("Enter the pitcher's total HRs in the last 10 games: "))  
 bp = input("Enter the Ballpark for the BP factor: ")  
 brls = float(input("Enter the batter's Barrel Rate: "))  
 angle = float(input("Enter the batter's Launch Angle Sweet-Spot: "))  
 avg\_ev = float(input("Enter the batter's Average EV: "))  
 type\_1, type\_2, type\_3 = map(float, input("Enter the batter's performance against the pitcher's top 3 pitch types: ").split())  
  
 # calls function to get ballpark rate value via user input  
 bp\_rate = bp\_factor(bp)  
  
 # batter multiplier  
 bats\_mul = (batter\_hr / 10)  
  
 pitch\_mul = (pitcher\_hr / 9 / 9)  
 # barrel rate multiplier  
 brm = 1 + brls  
  
 # launch angle multiplier  
 a = 25  
 b = 35  
 if min(a, b) < angle < max(a, b):  
 lam\_sw = 1.1  
 else:  
 lam\_sw = 0.9  
  
 # exit velocity adjustment  
 c = 90  
 d = 95  
 if avg\_ev >= 95:  
 eva = 1.2  
 elif avg\_ev >= 90:  
 eva = 1.1  
 else:  
 eva = 1.0  
  
 # bpt average  
 bpt\_avg = (type\_1 + type\_2 + type\_3) / 3  
  
 # main formula  
 hr\_formula = round((bats\_mul \* pitch\_mul \* bp\_rate \* brm \* eva \* lam\_sw \* bpt\_avg \* 100), 3)  
  
 player\_list.append(player\_name)  
 hr\_list.append(hr\_formula)  
  
 hr\_sorted, player\_sorted = sort(hr\_list,player\_list)  
 print(pandas.DataFrame(hr\_sorted, player\_sorted))  
  
hr\_prob()

**August 5, 2024**

So I decided to **add a player dictionary** because I’m tired of manually adding the barrel rate, launch angle, and exit velocity information. There is definitely an easier way to do this such as scraping but for now I’ll stick to this. **I also only chose hitters from the top 50ish leaders in homeruns so we’ll see if we yield a higher percentage of players who actually hit.**

It actually took me 30 years to create a big dictionary that splits the information into variables. **I had to use an excel scrape considering that I already had the information for barrel rate, launch angle, and exit velocity.** The formatting is funky right now but we can fix it overtime. Web scraping felt too complicated to do and I’ll just do that once this model is actually doing its actual JOB.

**Picking the highest hitters of the MLB has yielded some crazy numbers, they’re actually above 10%**. DOINK, the website that I use to get the batter vs. pitcher information has a great tool that lets me see the matchups between lefties RHP LHP etc. and every combo possible. I used that feature for all the BPTA for these picks.

**If the percentages tend to keep going up by double digits and so on, I’m going to divide the PML range exactly by 3, so Low is 0-33%, Mid is 34-66%, and Peak is 67-100%**

# Retrieve the player's stats if the player exists in the dictionary  
if pl\_name in player\_info:  
 stats = player\_info[pl\_name]  
# Assign each stat to a variable, with default values for missing stats  
 barrel\_rate = stats.get("br\_rate", stats.get("barrel\_rate"))  
 launch\_angle = stats.get("la", stats.get("launch\_angle"))  
 average\_velocity = stats.get("avg\_ev", stats.get("average\_velocity"))  
 return barrel\_rate, launch\_angle, average\_velocity

This is the most important part of my player\_info function so I’m only going to paste this part, the dictionary is too long.

/

/

/

/

/

/

/

/

/

UPDATED CODE (does not include the ballpark and player\_info function cause they’re just long dictionaries and I already showcased the sort AND ballpark function in the last entry):

#calculation  
def hr\_prob():  
 player\_list = []  
 hr\_list = []  
 for x in range(calc\_num):  
 player\_name = input("What is the name of the player?: ")  
 batter\_hr = float(input("Enter the batter's HRs in the last 10 games: "))  
 pitcher\_hr = float(input("Enter the pitcher's total HRs in the last 10 games: "))  
 bp = input("Enter the Ballpark for the BP factor: ")  
 brls, angle, avg\_ev = player\_info(player\_name)  
 type\_1, type\_2, type\_3 = map(float, input("Enter the batter's performance against the pitcher's top 3 pitch types: ").split())  
  
 # calls function to get ballpark rate value via user input  
 bp\_rate = bp\_factor(bp)  
  
 # batter multiplier  
 bats\_mul = (batter\_hr / 10)  
  
 pitch\_mul = (pitcher\_hr / 9 / 9)  
 # barrel rate multiplier  
 brm = 1 + brls  
  
 # launch angle multiplier  
 a = 25  
 b = 35  
 if min(a, b) < angle < max(a, b):  
 lam\_sw = 1.1  
 else:  
 lam\_sw = 0.9  
  
 # exit velocity adjustment  
 c = 90  
 d = 95  
 if avg\_ev >= 95:  
 eva = 1.2  
 elif avg\_ev >= 90:  
 eva = 1.1  
 else:  
 eva = 1.0  
  
 # bpt average  
 bpt\_avg = (type\_1 + type\_2 + type\_3) / 3  
  
 # main formula  
 hr\_formula = round((bats\_mul \* pitch\_mul \* bp\_rate \* brm \* eva \* lam\_sw \* bpt\_avg \* 100), 3)  
  
 player\_list.append(player\_name)  
 hr\_list.append(hr\_formula)  
  
 hr\_sorted, player\_sorted = sort(hr\_list,player\_list)  
 print(pandas.DataFrame(hr\_sorted, player\_sorted))  
  
hr\_prob()

Updated code doesn’t ask for 3 pieces of information anymore: BR rate, Launch Angle, and Avg EV. Instead it splits 3 variables that calls for the player\_info function that includes all the information per the player’s last name. If the last name is repeated or similar then you type out the whole name. Accents on player last name were removed just to make it simpler. There was also no need to actually use .split since the player\_info function already returned 3 pieces of information in that order.

EXAMPLE OF THE PLAYER INFO DICTIONARY:

player\_info = {  
 "Abrams": {  
 "barrel\_rate": 88.8,  
 "launch\_angle": 4.9,  
 "average\_velocity": 34.9  
 },  
 "Abreu": {  
 "barrel\_rate": 90.8,  
 "launch\_angle": 7.1,  
 "average\_velocity": 34.5  
 },  
 "Adames": {  
 "barrel\_rate": 88.6,  
 "launch\_angle": 7.1,  
 "average\_velocity": 35.4  
 },  
 "Adell": {  
 "barrel\_rate": 89.8,  
 "launch\_angle": 7.1,  
 "average\_velocity": 30  
 },…