Home Run Predictor

Name: Saiteja Mukkamalla
NetID: srm319
Section: 2
GitHub Repo Link:
https://github.com/srm319/Intro-to-Data-Science-Project---Saiteja-Mukkamalla.git

1. Project Definition:

1.1 Project Statement: This main problem this project aims to solve is to estimate the number of home runs a Major League Baseball player will hit in the 2025 season through analysis of the player's hitting metrics which include exit velocity, launch angle, bat speed, and past performance from the 2024 season. Through the use of these statistics, we can create a model which can help track player development and trends in player performance in the future which can be beneficial for players themselves or even fans who are interested in the sport.

1.2 Plan: The main strategy is to create a predictive model based on player performance and the player's past data by using certain concepts in machine learning such as polynomial regression and time-series analysis. By using the different hitting metrics and past performance data from Statcast, a tracking tool which can analyze massive amounts of baseball data, we can create a model which can accurately predict an estimated number of home runs a player will hit in a season.

.

1.3 Relation to Course Material: The project relates to lectures since there will be a lot of data cleaning and data structuring while analyzing the data and we will be doing this through the use of pandas and numpy. We will have to sort data and handle missing values such as players with insufficient data or those who do not meet the requirement and this technique is something that we have learned in lectures and have implemented in our assignments as well.

2. Novelty and Importance

2.1 Importance: This project is important because it can help with player development by providing players the information they need to get better and which hitting metrics they should improve on in order to increase their number of home runs. It can also become an important aspect of scouting as MLB teams will look for players who are estimated to have higher home runs and can therefore have a greater impact on the team. Finally, this can be really helpful for fans as sports betting and fantasy teams are drastically on the rise. I am excited about the project because I find sports statistics interesting and I am curious to see if sports is all about the numbers or if there are other factors in play.

2.2 Excitement: I am very excited to work on this project because sports statistics is something that I am very interested in and there is a lot of potential in the sports market when it comes to analytics and helping improve performance. As an athlete myself, I know the importance of data and how it can drastically affect results. I feel that baseball is the perfect sport to incorporate this analysis because there are so many features about a batter such as their launch speed, launch angle, etc which all can be used to make predictions.

2.3 Existing Questions and Prior Work: There have been similar questions in the area such as how does launch angle and exit velocity impact home run probability on a single swing. The study proved that higher exit velocities and the optimal launch angle drastically increased the likelihood of a ball going for a home run. This study introduced a new metric called ‘Barrel Rate’. Other similar studies include how environmental factors such as stadium size, weather conditions and altitude affect home run probability. Overall, there have been many prior related works which analyze player performance.

3. Method

3.1 Data utilization: I collected data from Statcast which is a tool used to analyze player performance and it tracks every pitch in Major League Baseball. I used the 2024 season for this project and the data will include exit velocity, launch angle, batting averages as well as the number of home runs in the previous seasons. I collected this data through the pybaseball which is a popular python package which contains all of the numerical features of every pitch in the MLB. The data is stored in CSV format and has already been pre-processed into a tabular format which makes it very simple to analyze data and collect from. I decided to not create a SQL database like I mentioned in my proposal because the pybaseball library already has all the data from every pitch in the MLB and it is redundant to scrape data from the website itself and put it into a separate database. These websites also have security which does not allow for web scraping so that is the main reason I used the pybaseball library.

```
warnings.filterwarnings('ignore', category=FutureWarning)
cache.enable()
print("Initiating Download for all Statcast Data(will take some time due to amount of data)")
data = statcast(start_dt="2024-04-01", end_dt="2024-09-01")
```

```
Sample of Statcast Data
     pitch_type  game_date  release_speed  release_pos_x  release_pos_z  \
1674         FF 2024-09-01           87.8          -3.46           5.87
1716         SI 2024-09-01           90.0          -3.07           5.91
1766         FF 2024-09-01           87.1          -3.57            5.8
1781         FF 2024-09-01           87.3          -3.34           5.79
1835         SI 2024-09-01           86.7          -3.25           5.94

       player_name  batter  pitcher     events    description  ...  \
1674  Floro, Dylan  681624   571670  strikeout  called_strike  ...
1716  Floro, Dylan  681624   571670       None           ball  ...
1766  Floro, Dylan  681624   571670       None           foul  ...
1781  Floro, Dylan  681624   571670       None  called_strike  ...
1835  Floro, Dylan  500743   571670  field_out  hit_into_play  ...

      n_thruorder_pitcher  n_priorpa_thisgame_player_at_bat  \
1674                    1                                 2
1716                    1                                 2
1766                    1                                 2
1781                    1                                 2
1835                    1                                 0

      pitcher_days_since_prev_game  batter_days_since_prev_game  \
1674                             2                           15
1716                             2                           15
1766                             2                           15
1781                             2                           15
1835                             2                            1

      pitcher_days_until_next_game batter_days_until_next_game  \
1674                             6                           3
1716                             6                           3
1766                             6                           3
1781                             6                           3
1835                             6                           1

     api_break_z_with_gravity api_break_x_arm api_break_x_batter_in arm_angle
1674                     2.09            0.63                  0.63      33.9
1716                     2.39            1.41                  1.41      38.7
1766                     1.73             0.5                   0.5      33.5
1781                     1.88            0.77                  0.77      35.3
1835                     2.32            1.09                  1.09      37.6

[5 rows x 113 columns]
```

3.2 Data Structure and Features

The dataset includes the following numerical features for each player in the 2024 season:

- exit_velocity (average mph)
- launch_angle (average degrees)
- barrel_rate (percentage of barreled balls)
- bat_speed (average mph)
- past_home_runs (home runs in previous season)
- games_played (number of games played)

The target variable is home_runs, representing total home runs hit in a given season.

```
All Columns(features) in Dataset:
['pitch_type', 'game_date', 'release_speed', 'release_pos_x', 'release_pos_z', 'player_name', 'batter', 'pitcher', 'events', 'description',
'spin_dir', 'spin_rate_deprecated', 'break_angle_deprecated', 'break_length_deprecated', 'zone', 'des', 'game_type', 'stand', 'p_throws', 'ho
me_team', 'away_team', 'type', 'hit_location', 'bb_type', 'balls', 'strikes', 'game_year', 'pfx_x', 'pfx_z', 'plate_x', 'plate_z', 'on_3b',
'on_2b', 'on_1b', 'outs_when_up', 'inning', 'inning_topbot', 'hc_x', 'hc_y', 'tfs_deprecated', 'tfs_zulu_deprecated', 'umpire', 'sv_id', 'vx
0', 'vy0', 'vz0', 'ax', 'ay', 'az', 'sz_top', 'sz_bot', 'hit_distance_sc', 'launch_speed', 'launch_angle', 'effective_speed', 'release_spin_r
ate', 'release_extension', 'game_pk', 'fielder_2', 'fielder_3', 'fielder_4', 'fielder_5', 'fielder_6', 'fielder_7', 'fielder_8', 'fielder_9',
'release_pos_y', 'estimated_ba_using_speedangle', 'estimated_woba_using_speedangle', 'woba_value', 'woba_denom', 'babip_value', 'iso_value',
'launch_speed_angle', 'at_bat_number', 'pitch_number', 'pitch_name', 'home_score', 'away_score', 'bat_score', 'fld_score', 'post_away_score',
'post_home_score', 'post_bat_score', 'post_fld_score', 'if_fielding_alignment', 'of_fielding_alignment', 'spin_axis', 'delta_home_win_exp',
'delta_run_exp', 'bat_speed', 'swing_length', 'estimated_slg_using_speedangle', 'delta_pitcher_run_exp', 'hyper_speed', 'home_score_diff', 'b
at_score_diff', 'home_win_exp', 'bat_win_exp', 'age_pit_legacy', 'age_bat_legacy', 'age_pit', 'age_bat', 'n_thruorder_pitcher', 'n_priorpa_th
isgame_player_at_bat', 'pitcher_days_since_prev_game', 'batter_days_since_prev_game', 'pitcher_days_until_next_game', 'batter_days_until_next
_game', 'api_break_z_with_gravity', 'api_break_x_arm', 'api_break_x_batter_in', 'arm_angle']
```

3.3 Model: For this project, I used polynomial regression in order to predict the number of home runs a player hits since this is a supervised learning project. I chose to use this model instead of linear regression because it allows for non-linear relationships between certain hitting metrics and the predicted number of home runs. Since higher launch angles do not always lead to more home runs, polynomial regression is preferred since there is an optimal range which can be used for certain variables and therefore does not always have to be a linear relationship. This will help solve the problem and will work better than existing methods since it will be more precise and will help identify non-linear relationships. It is also much easier to understand compared to more advanced and detailed methods.

```
features = ['exit_velocity', 'launch_angle', 'barrel']

X = aggregatedDataframe[features]
y = aggregatedDataframe['home_runs'] / aggregatedDataframe['plate_appearances']

polyexpo = PolynomialFeatures(degree=2, include_bias=False)
X_poly = polyexpo.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2, random_state=42)
model = Lasso(alpha=0.1)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"\nModel Evaluations:")
print(f"MSE: {mse:.2f}")
print(f"MAE: {mae:.2f}")
print(f"R^2: {r2:.3f}")
```

3.4 Problem/Feature Space:

Since this is a regression problem where we want to observe the relationship between the player features and their home run output, we can use polynomial features of degree 2 because the relationship is likely non-linear and launch angles which are too high or too low will not impact home run probability drastically.

3.5 Implementation: First I collected the data from Statcast and created a dataset which is structured well and easy to follow. Next, I will handle the missing values and clean the dataset by removing any players who did not play enough games or those who do not have enough information. I will then create categories such as exit velocity, launch angle, past home runs, barrel rate, etc which will be used as the main features. For features such as launch angle, barrel rate and exit velocity, we can use higher order polynomials in order to account for the non-linear relationships and therefore increase the accuracy of the results. We can then split the data into training and testing sets and then create the polynomial features. Using Scikit-Learn, we can create the polynomial regression model and then use certain regularization techniques such as Lasso regression in order to clean up the results.

Steps

1. **Data Loading and Cleaning**: Handled missing values and removed NaN so that players with no data are not changing the results of the test. I also removed players who did not have any batting data so that most pitchers who are not hitting are not included.

2. **Features:** Created polynomial features of degree 2 for testing.

3. **Train-Test Split**: Split the dataset into a 80/20 with 20% in the training set to evaluate general applicability.

4. **Model Training**: Used Lasso regression to reduce overfitting and feature noise.

5. **Evaluation**: Calculated MSE, MAE, and R^2 metrics for evaluation of results and to check and measure results.

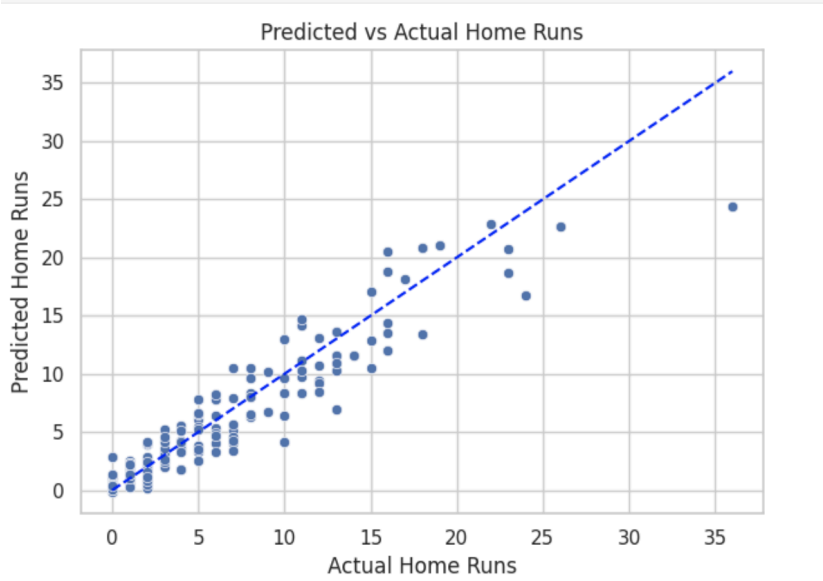
4. Results and Key Findings

4.1 Results of Model + Visualization:

```
Sample Predictions (Alphabetical):
                      Player  Actual HR rate  Actual HRs  Predicted HRs
151              Abbott, Cory        0.200000           7            4.0
144               Adam, Jason        0.187500           6            5.3
113                Adon, Joan        0.192308           5            6.1
35             Anderson, Grant        0.131579           5            3.6
127              Assad, Javier        0.142857           9           10.2
..                         ...             ...         ...            ...
86           Woodruff, Brandon        0.304348           7            3.4
67   Woods Richardson, Simeon        0.142857           1            1.4
72              Zastryzny, Rob        0.045455           1            2.2
38                Zerpa, Angel        0.178571           5            2.5
156             Zuñiga, Guillo        0.000000           0            0.4

[161 rows x 4 columns]
```

Above table shows sample predictions including # of HRs the past season and predictions for the 2025 season.



Above scatter plot shows comparisons between the actual # home runs in 2024 season to the predicted # home runs for 2025 season.

4.2 Evaluation Metrics:

- **Mean Squared Error (MSE)**: This calculates average squared difference between actual and predicted values. The value which I obtained was 0.01.

- **Mean Absolute Error (MAE)**: This is the average of differences between actual and predictions. The value which I obtained was 0.07.
- **R^2 Score**: Shows proportion of variance in the model. The value which I obtained was 0.436

```
Model Evaluations:
MSE: 0.01
MAE: 0.07
R^2: 0.436
```

5. Conclusion

5.1 Expectations vs Outcome: Before completing the project, I expected the predictions to be quite similar because there was enough data regarding launch speed and launch angle for multiple players that all of it can be used in order to come up with an accurate model. After looking at the results and the performance of the model, it was quite similar to what I expected at the start. The MSE and MAE were both very low which means that the predictions were accurate to a good level but the R^2 was only 0.4 which means that more than half of the variance is unexplained.

5.2 Differences from Proposal: One major difference I made was how I obtained the dataset at the beginning of the project. I first wanted to scrape the data from the website but there was too much data to do so and was very time consuming. Also the website does not allow for users to scrape their data and it is illegal to do so. Therefore, I decided to use the pybaseball library which gathers information from the Statcast data and records every single pitch in MLB. That made it much more convenient and since the data was already in CSV format and in tabular form, there was no need to create a SQL database.

5.3 Bottlenecks: One limitation was being able to access all the Statcast Data. There is so much data on every single pitch that it takes a long time to download for the user. Another issue is that there are only 4 features I took into account which can drastically affect the results. If more features were included like ballpark size or weather, then there could be improvements in the predictions. The issue with using all these features is that it is incredibly difficult to correlate all of these features and I was working alone which made it tougher to get all of the work done in the deadline.

5.4 Improvement and Summary: Overall, I was able to complete this work by myself and although it was difficult, I still got decent results from the model and with certain improvements in the code, there is potential for this to be massive when it comes to sports betting or player scouting. Those certain improvements can be taking in more features into account. For example, I only took into account some of the numerical features but there are other numerical features I could use such as weather because the ball travels farther in warmer weather. I could also take into account categorical features such as the type of ballpark they play in because all ballparks in MLB have different dimensions so the amount of home runs can change when a player plays at a certain stadium more times than others. At the end of the code, I was also able to create a cell which asks the user for a player name and then output of the code will then show the number of predicted home runs for that player in 2025 season.

```
inputPerson = input("Input a player name (e.g., Ohtani, Shohei): ")

player = aggregatedDataframe[aggregatedDataframe['player_name'].str.lower() == inputPerson.lower()]

if player.empty:
    print("Player is not in the Dataset! Please enter another player!")
else:
    inputX = player[['exit_velocity', 'launch_angle', 'barrel']]
    inputPolyX = poly.transform(inputX)  # Use already-fitted transformer
    predHrRate = model.predict(inputPolyX)[0]
    playerName = player['player_name'].values[0]
    numAppearances = player['plate_appearances'].values[0]
    predictedHRs = np.round(predHrRate * numAppearances, 1)
    print(f"\nPredicted Home Runs for {playerName}: {predictedHRs:.1f}")
```

```
Input a player name (e.g., Ohtani, Shohei): ↑↓ for history. Search history with c-↑/c-↓
```