How Far Will It Go? Predicting Hit Distance for MLB Players

Ian Curtis

2022-12-05

Introduction

All MLB teams have a private data analytics team devoted to running analyses and making predictions for that team; however, these analyses and results are not made public in an attempt to give the respective team an advantage. The goal of this project is to contribute to this public repository of MLB projects by predicting the distance a batted ball will travel when hit from a professional Major League Baseball (MLB) player and to find an effective algorithm to make these predictions.

One of the great aspects of baseball is the amount of random variables at play and the variation of day-to-day gameplay. One variable of interest is how far a batter will hit a ball. With a model that can predict how far a ball will travel, teams may be better able to predict where to place their position players and how to prepare these players for a ball. For instance, if a player is more likely to hit a ball in the outfield than the infield, a team's outfielders can be more prepared for a fly ball. At the same time, it may be useful for batters to know trends of how far balls are typically hit in certain situations in order to try and hit a ball somewhere else in the field.

To predict hit distance, I explore three algorithms: linear regression (implemented by hand with matrix algebra), decision trees, and neural networks. Each algorithm is used to train several models and prediction accuracies are compared.

In order to efficiently record and analyze hit distance, I divide up the field into distance categories (measured in feet from home plate):

- Zone 1: 0 - 70
- Zone 2: 71 - 140
- Zone 3: 141 - 210
- Zone 4: 211 - 280
- Zone 5: 281 - 350
- Zone 6: 351+

The regression algorithm predicts a numeric value for hit distance which is then assigned a zone based on the above categories while the decision tree and neural network predict the category itself. From my analyses, it appears that neural networks perform the best when predicting hit distance categories (81.14% accurate on test data with two predictors) followed by decision trees (73.22% accurate on test data with two predictors). Unfortunately, the linear regression assumptions were not met (some information on the analysis is provided nonetheless) indicating that linear regression is not the most useful model to use when predicting hit distance.

Related Work

Various other projects have been attempted in baseball analytics, usually with the goal of predicting pitch type or the outcome of a game. Huang and Li (2021) use various neural networks and Support Vector

Machines (SVM) to predict wins and losses of MLB games. Cserepy et al. also attempted to predict game outcomes but through simulations rather than an hard algorithm. Lee (2022) uses neural networks but for predicting a pitch type and where the pitch will land in the strike zone. Everman (date unknown) uses a handful of popular statistics to predict overall team performance.

Hung (2012) chose to use Principal Component Analysis to reduce data dimensions followed by a k-means clustering algorithm in order to evaluate certain batter metrics and skill set. Young et al. take a unique approach by using a neural network to predict which players might be inducted into the Hall of Fame. There appears to be support for using neural networks in baseball analysis as well as classifiers. Regression seems to be a fairly traditional approach to sports analysis and both Maszczyk et al. (javelin throws) and Karnuta et al. (baseball injuries) demonstrate the superiority of neural networks in this field.

Methods

Initial Data Source and Details

Data for this project was collected using the R package `baseballr` which allows for connection to the MLB Statcast (Baseball Savant) Database. The data analyzed in this project contained all pitches from the MLB 2022 regular season (i.e., no postseason) but was filtered down later on in the process (see EDA). Originally, I choose as many variables as I could that I thought might have an influence over the distance of a batted ball (`hit_distance`). These were:

- `game_date`: the specific date a game was played (YYYY-MM-DD)
- `events`: the result of a play (e.g., hit, foul, out)
- `pitch_type`: the type of pitch thrown by the pitcher (abbreviation)
- `release_speed`: the speed of the ball out of the pitcher's hand
- `stand`: the handedness of a batter
- `p_throws`: the handedness of a pitcher
- `balls`: the number of balls in the count at the time of the pitch
- `strikes`: the number of strikes in the count at the time of the pitch
- `outs_when_up`: the number of outs in the inning at the time of the pitch
- `inning`: the inning number the pitch occurred in
- `inning_topbot`: determiner of whether the inning was the top or the bottom
- `launch_speed`: the speed of the ball off of the bat
- `launch_angle`: the angle of the ball after being hit
- `release_spin_rate`: the spin rate of the baseball out of the pitcher's hand (RPM)
- `at_bat_number`: the number of the at bat in the game
- `pitch_number`: the number of the pitch in the at bat

In this original, "raw" data other variables were also selected to help the EDA process and to obtain more information about an observation:

- `description`: a brief indication of the result of the play (e.g., hit, out, foul)
- `events`: a detailed indication of the result of the play (e.g., type of hit, foul, type of out, etc.)
- `pitch_name`: the full name of the corresponding pitch (as abbreviated in `pitch_type`)
- `home_team`: an unused variable indicating the home team at the time of the pitch (to be used in the future work?)
- `dist_categ`: a variable added later on indicating the respective distance category for an observation based on its `hit_distance` value.

The script used to gather this data iterates over each day in the 2022 regular season and collects the above variables for each day. It then automatically removes any observations that may have missing value for `hit_distance` (as these would not hold any use for this project).

Exploratory Data Analysis

After collecting the raw data, I began analyzing variables individually and together through an exploratory data analysis (EDA) phase (`eda.Rmd`). This EDA led me to make a few changes in the data I was planning to use through transformation and preprocessing.

I realized that I need to exclude bunts, foul balls, and any hits with a hit distance less than 15 feet to meet the regression assumption of normality. There is a strong floor effect here; as `hit_distance` cannot be 0, a lot of small-valued distances (such as from bunts) appeared making the histogram of `hit_distance` not approximately normal (Figure 1).

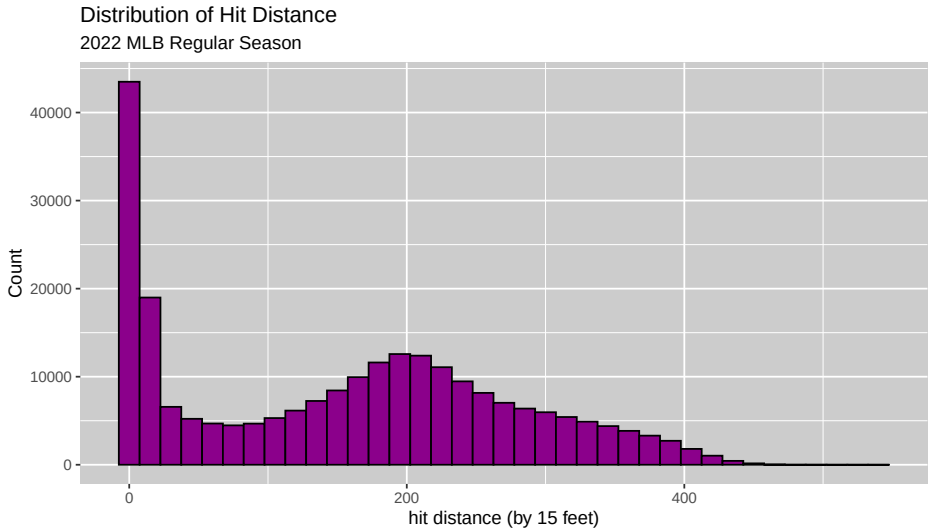


Figure 1: Distribution of Hit Distance

I am analyzing hit distance in an attempt to help prepare fielders for potential hits. Foul balls are not being considered as they were not in the field of play and may not accurately represent what in-play balls would do.

I also looked at the distribution of `dist_categ` (Figure 2).

From this figure we can see that some categories (namely, Zones 2, 3, and 6) have less observations that the others. It is unclear if this difference will play a role in how the algorithms will perform. Typically, we wish to see a roughly equal number of observations per category but since there are over 10,000 observations in each zone, this may not be as necessary.

From looking at `game_date`, I noticed that the number of observations per month differed (Figure 3).

Although it wouldn't be feasible to include each unique value of `game_date` in a model, I decided to turn each value of `game_date` into a `month` variable to include in further analyses (Figure 4).

In addition to these changes, I needed to consolidate of of the lesser-used `pitch_type`s into an "other" category while retaining the commonly used pitches. Those merged into "other" included CS, EP, FA, FS, KC, and KN (Figure 5).

I also decided to limit my analysis to only pitches in the first 9 innings of each game, to remove `at_bat_number` from the project, and to take the square root of `pitch_number` which was slightly right skewed.

For bivariate EDA, I generated a correlation matrix (Figure 6). For the most part, the variables are uncorrelated; however, `launch_angle` and `launch_speed` are both moderately correlated with `hit_distance`, which does make sense here as these variable occur after a ball has been hit and together can create some accurate predictions (Figure 7).

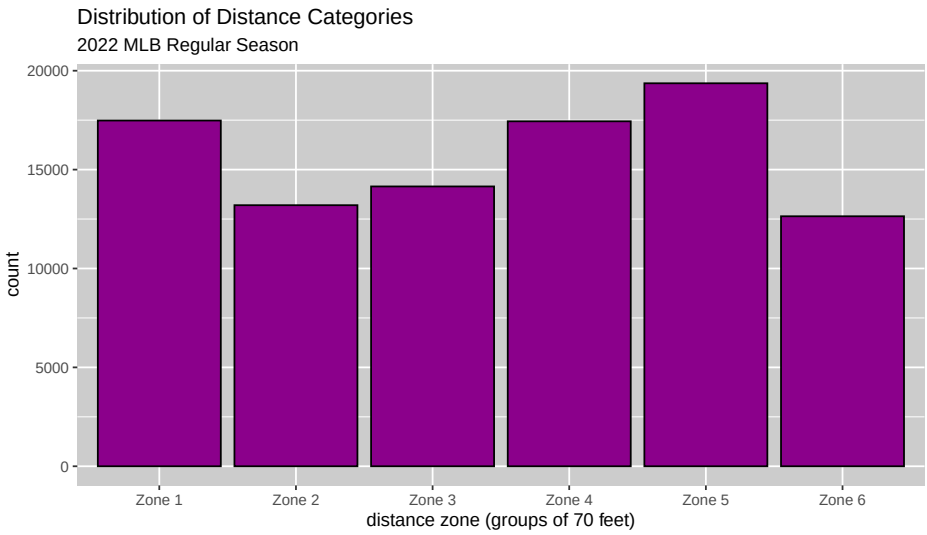


Figure 2: Distribution of Distance Category

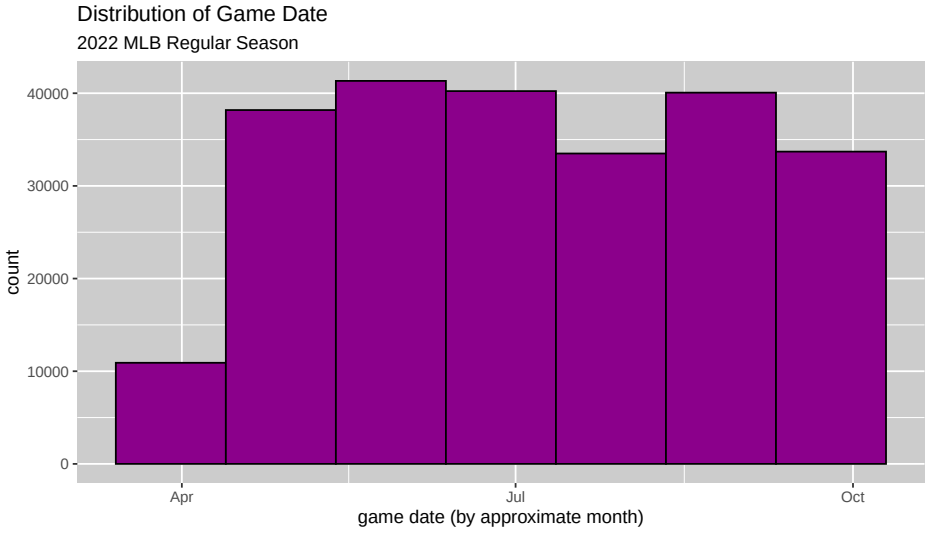


Figure 3: Distribution of Game Date

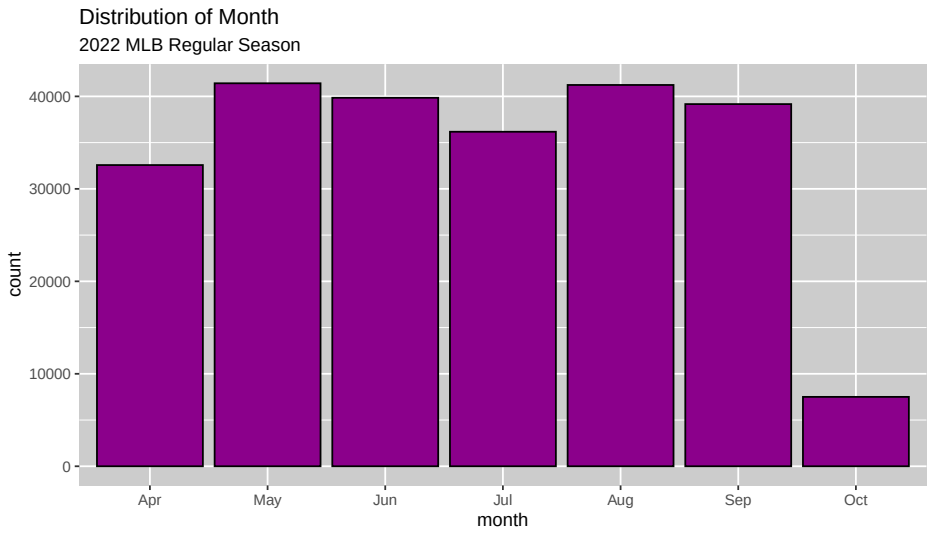


Figure 4: Distribution of Game Month

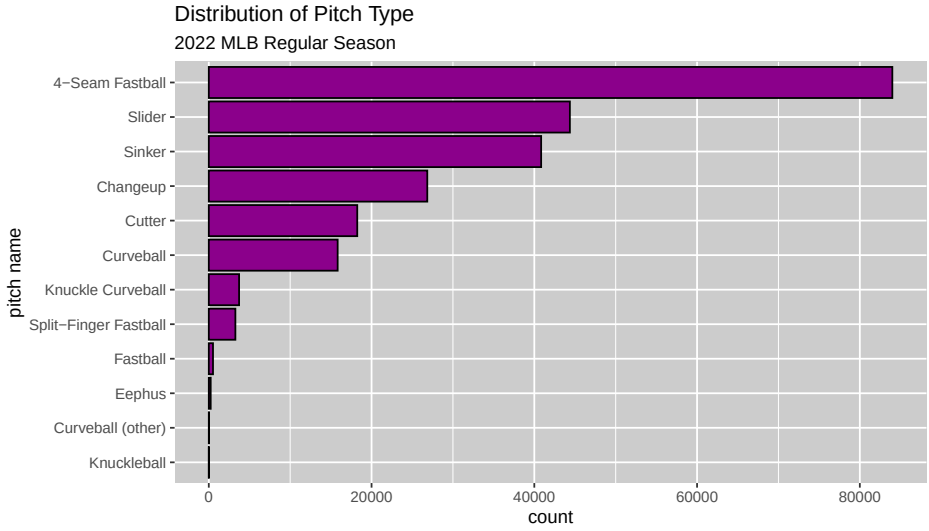


Figure 5: Distribution of Pitch Type

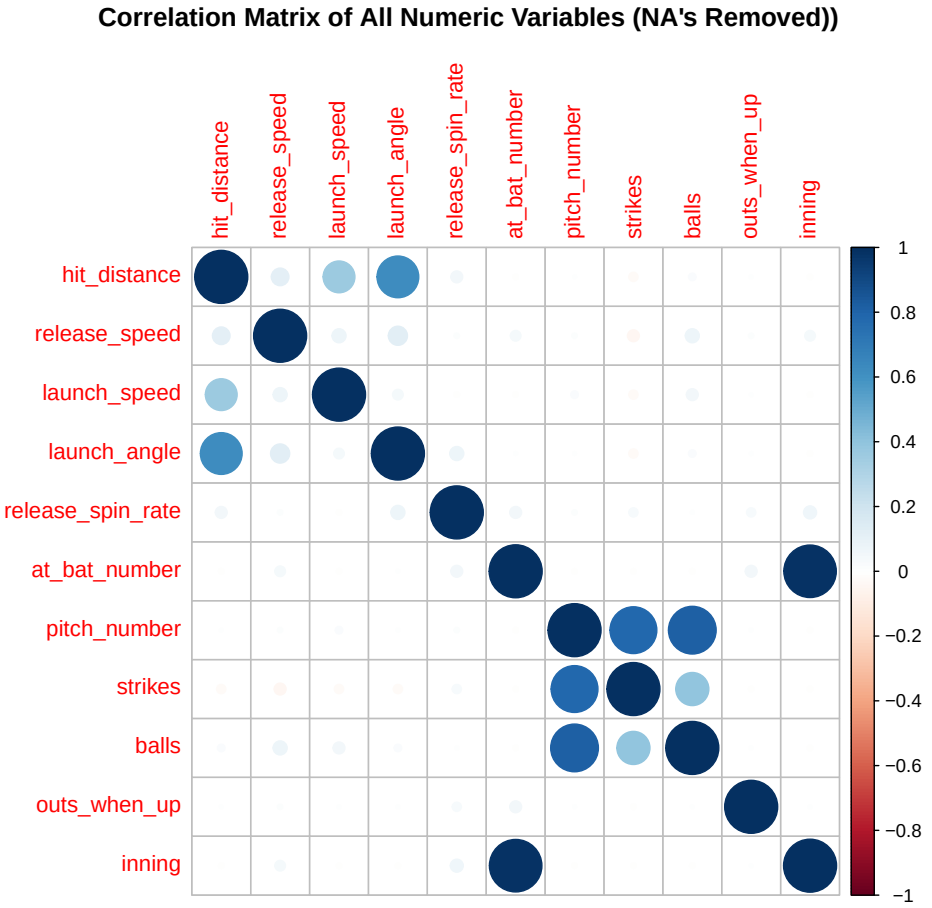


Figure 6: Correlation Matrix

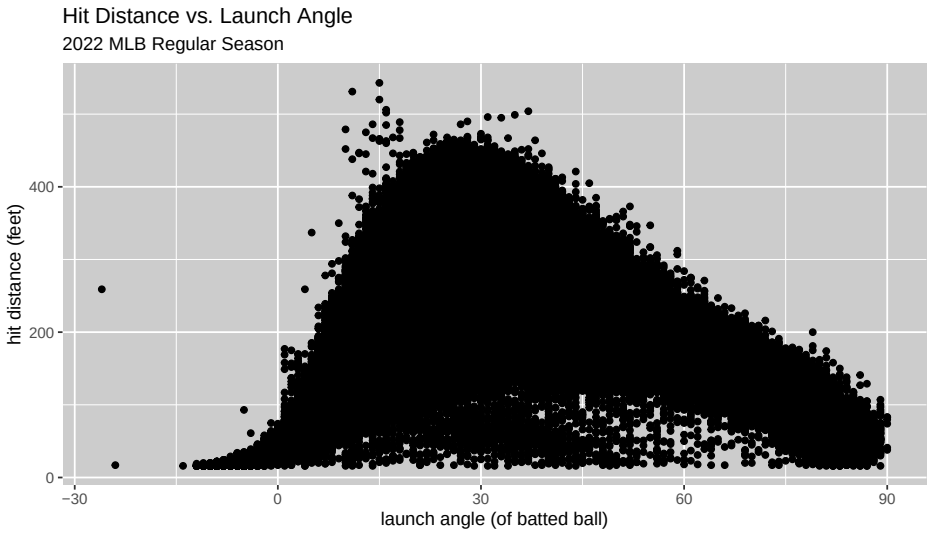


Figure 7: Comparing Hit Distance and Launch Angle

Not many other variables are correlated with `hit_distance` and this may indicate that they are not useful in predicting `hit_distance`. It is good to see that `balls` and `strikes` are correlated with `pitch_number` and that `inning` is coordinated with `at_bat_number`. This helps verify that the data was collected correctly.

Preprocessing

After completing EDA, I moved on to a preprocessing step where I gathered all the information from the EDA and performed some variable deletion, addition, and transforming to prepare the data for the upcoming model training.

The file `preprocessing.Rmd` contains the steps taken in the form of multiple datasets on which transformations were applied. `mlb_raw` consists of filters based on the results of the EDA. `mlb_transform` then builds on that by adding in new variables such as `month`, the square root of `pitch_number`, the new `pitch_type` (with the "other" category), and a factor variable for the field divisions.

`mlb_dummy` then takes that data and adds in dummy variables and select interaction variables for the regression and neural network models, removing any observations with missing values. Finally, the data is randomly split into a train and test data where 85% of the original observations are in the training set and the remaining 15% for the testing data. These datasets contain all possible variables that could be used by any of the three model in this project (not all algorithms will use all of the variables in the dataset). To avoid having to redesign and resplit the data for each model (this would mean that each model would be getting a different training set), the file writes the train and test files to disk in order to ensure that each algorithm receives the same training data.

Linear Regression

As part of the final project requirements, a portion of the project was to be completed "by hand". For this piece, I chose to derive a linear regression algorithm through the necessary matrix algebra steps rather than use the `lm` function in R. Creating the algorithm in this way has helped me to appreciate the power of packages and functions as well as the simplicity of code (the `lm` function requires less lines of code than a full matrix algebra function).

The file `regression.Rmd` contains all of the code for the algorithm, models, and evaluation. The regression algorithm is run on all numeric, dummy, and interaction variables (and therefore does not need `pitch_number` or `pitch_type`). In total, there were six models evaluated:

1. The Full Model: contains all variables
2. Reduced 1: Contains all variables from the full model whose individual p-values are less than 0.05
3. Reduced 2: Contains all variables from Reduced 1 individual p-values are less than 0.05
4. Reduced 3: Contains select variables from the full model that I chose arbitrarily
5. Reduced 4: Contains all variables from Reduced 3 whose individual p-values are less than 0.05
6. Reduced 5: Contains variables from the full model specifically chosen based on the previous models. These variables were consistent over the reduced models.

Unfortunately, not all of the linear regression assumptions were met for these models. Some of the variables, as discovered in the EDA step, are not completely normally distributed. However, the most egregious assumption violation is seen in the residual plot (shown below in Figure 8 for the Reduced 5 model). Note that the following plot also is formed when the `lm` function is used.

This residual plot shows a lack of constant variance around 0 and also a lack of evidence towards normality as the plot does not show a random scatter of points. Thus, the results from this regression algorithm are not reliable and should not be used in further analysis. Testing accuracies are not included for this reason.

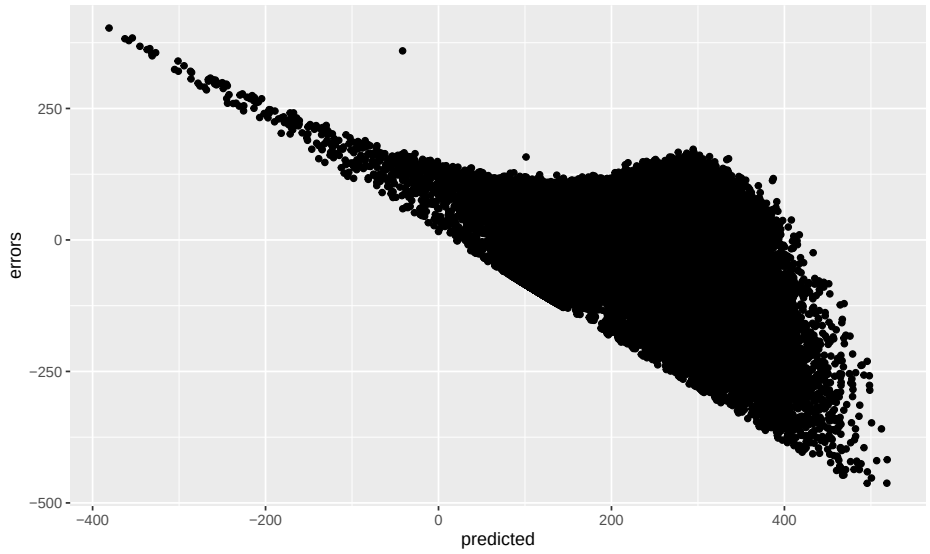


Figure 8: Residual Plot for Reduced 5

Decision Tree

The decision tree algorithm (`dtree.Rmd`) was implemented through the `rpart` package. The data used for this algorithm looked slightly different than that for regression. Since decision trees are designed to work very well with categorical data, no dummy variables were needed and interactions were also removed. Instead, each categorical/discrete variable was converted into a factor.

As model training proceeded, it was discovered that `launch_angle` and `launch_speed` were driving the prediction. This does make sense as these two variables occur after a ball has been hit and can largely determine how far a ball will travel. All in all, three models were attempted to determine the effects of these two variables on prediction:

1. The Full Model: Contains all variables
2. Reduced 1: Contains all variables except for `launch_speed` and `launch_angle`
3. Reduced 2: Contains only `launch_speed` and `launch_angle`

Each model uses a max depth of 5 (as recommended by Uzair & Jamil) and a complexity parameter (`cp`) set at 0.00001.

Neural Network

The neural network algorithm was implemented using the `nnet` package, as recommended by Mahdi et al. The data used to train the neural network consisted of all variables used in the linear regression, minus the interactions (thus the dummy variables were used). Before model training, all variables (except for `dist_categ`) were scaled to have a mean of 0 and unit variance. Similarly to the decision tree algorithm, it was discovered that `launch_angle` and `launch_speed` were controlling the predictions, which, again, does make sense in the context of hit distance. As a result, the same three models were chosen:

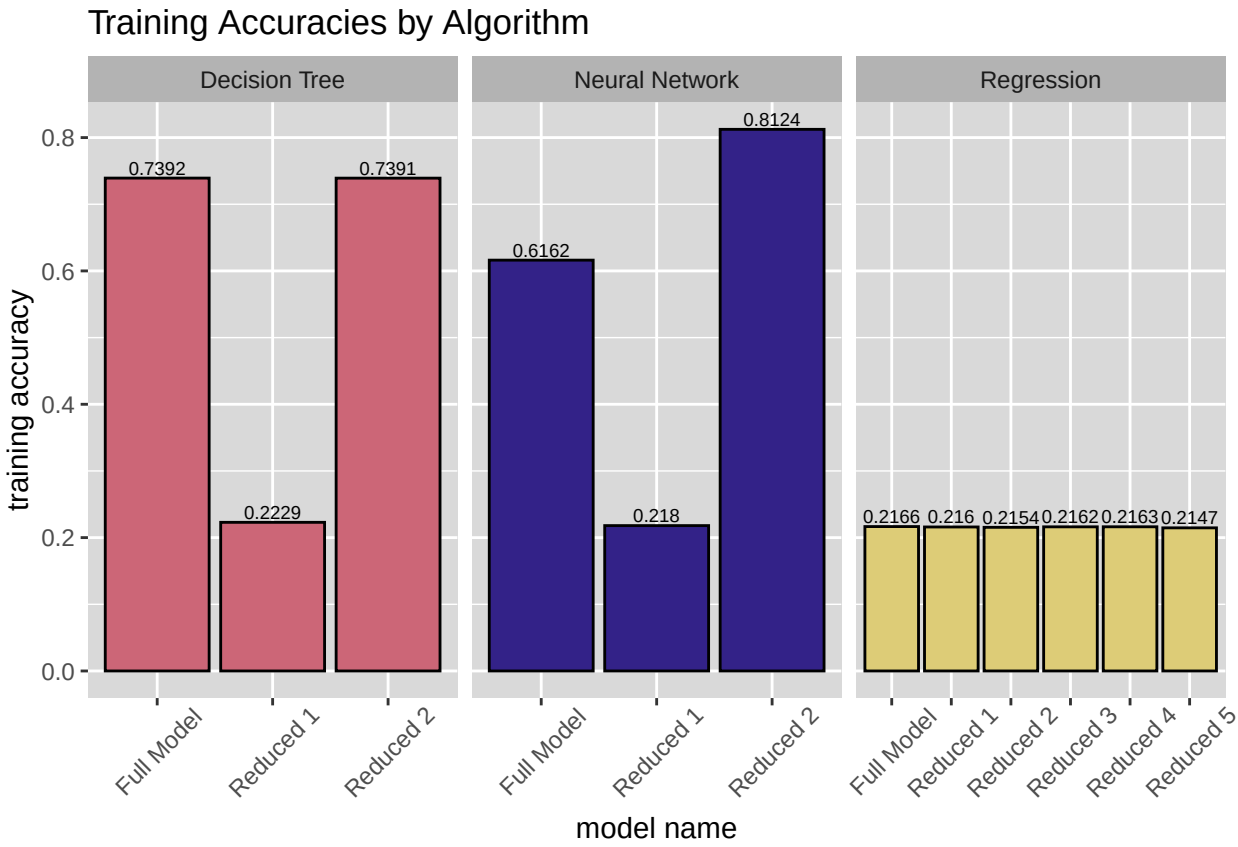
1. The Full Model: Contains all variables
2. Reduced 1: Contains all variables except for `launch_speed` and `launch_angle`
3. Reduced 2: Contains only `launch_speed` and `launch_angle`

All models use a hidden layer size of 5 and a decay value of 1.0e-7.

Results and Discussion

Training Set

Explorations of model performance on training data demonstrated interesting patterns. Below is a summary of model accuracies of predicting hit distance on the training data.



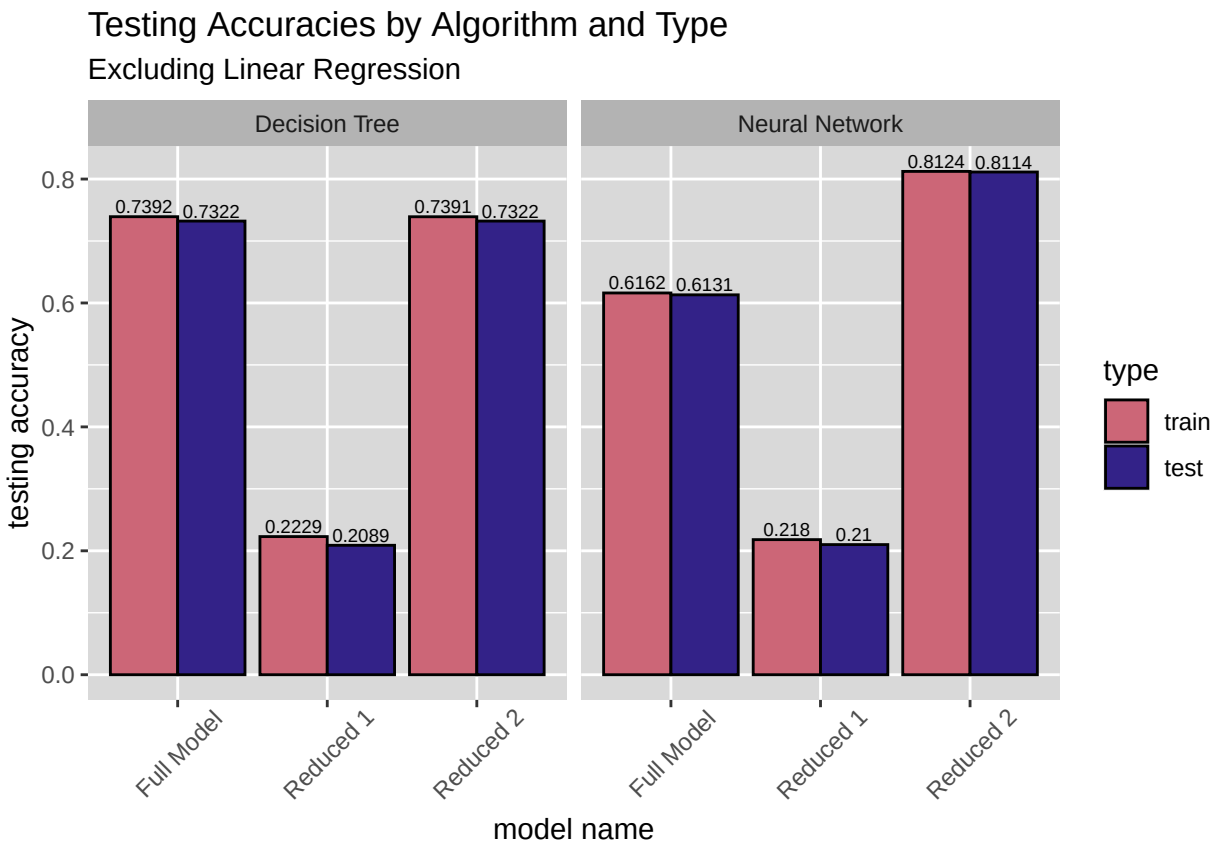
Overall, it appears that variables whose measurement occurs after a ball has been hit are the most accurate at predicting how far the ball will travel. This makes logical sense as the trajectory of a baseball is largely determined by the ball's angle and velocity and with just those two pieces of information, we can predict the ball's ultimate distance with high accuracy. For instance, as in the figure above, we can see that for both the decision tree and the neural network, the Reduced 1 model (which contains all variables except for `launch_speed` and `launch_angle`) have by far the lowest accuracy of the three models, at 0.223 and 0.218, respectively, and do not offer any productions for some categories demonstrating that the other variables do not provide a lot of information on hit distance on their own.

At the same time, the Reduced 2 models (those with only `launch_speed` and `launch_angle`) provide very accurate predictions, either with the same accuracy or higher than the full model. This also indicates that adding in extra predictor variables is not beneficial to promoting model accuracy. It is possible that other unused variables may provide this benefit (see Future Work).

The regression algorithm did was poor at predicting hit distance; even with invalid results the accuracies barely scrape past 20% success. Based on this project, it appears that linear regression is not a successful algorithm choice for predicting hit distance and that other algorithms offer more accurate and valid results.

Test Set

What follows is a summary of test and training accuracies for the decision tree and the neural network.



In general, the predictions for the testing data were almost exactly as accurate as those for the training data which is a good sign. Regardless of the model, it appears that decision trees and neural networks are reliable models for the prediction of hit distance (or, rather, distance zone), at least for the data used in this project. The testing results also demonstrate the pattern of `launch_speed` and `launch_angle` controlling the prediction.

As recommended by Mitchell et al., I have included a model card containing details on the models used in this project. This model card does not follow Mitchell et al.'s guidelines exactly but hopefully provides enough information to future researchers to use these algorithms and models. Emphasis is placed on the Reduced 2 models of the decision tree and neural network; regression is not considered.

Conclusion

Overall, this project provides useful information for future projects interested in predicting hit distance. Although linear regression may not be the ideal model solution for such a problem, decision trees and particularly neural networks show promise and have the potential to serve highly accurate predictions. With these two algorithms, the variables `launch_speed` and `launch_angle` appear to drive the prediction with a high accuracy coming from models using just those two variables as predictors. However, the accuracy for predictions made without these two variables may improve in future work with a different selection of predictor variables.

Nevertheless, this project had some limitations and would benefit from future work in the use of new variables and an expanded scope as described in the following section.

Limitations and Future Work

The main limitations for this project occur due to the scope of the data. At its current state, the data only consists of observations from the 2022 regular season (in the first 9 innings of each game). The results from this project would benefit from replication using data from previous seasons, postseasons, extra innings, and even using more or different explanatory variables (such as including foul balls, weather, home field advantage, or a team's winning streak). As mentioned in the results section , `launch_speed` and `launch_angle` drive the predictions. Since the measurement of these two variables occurs after the ball is hit, it would be useful to find a more accurate way to predict `hit_distance` with more advanced notice.

Additionally, the analysis would benefit from expanded detail in terms of predicting `hit_distance`. Currently, the data is aggregated: every pitch is treated the same regardless of home/away team, pitcher, batter, or ballpark. However, the setting in which a player finds themselves may in fact determine how far a ball is hit / can be hit. For example, a certain pitcher may be able to prevent balls from travelling very far or the dimensions of a ballpark could determine or limit the distance a ball could travel (e.g., Yankee Stadium's right field vs. Fenway Park's Green Monster). Future work may even examine the likelihood of hitting a home run given the ballpark, weather, and pitcher (among other potential variables).

Data and Software Availability

The files used for this project (code, scripts, and data) can be found on my GitHub repository (username `ian-curtis` and repository `mlb-hit-dist`, raw link https://github.com/ian-curtis/mlb-hit-dist). In particular, the raw data is found in the `data` folder under the name of `mlb_raw.csv` and `scrape.R` contains the code used to grab the data. This folder also contains the train and test data as well as a summary file of model performance. Files in the root directory are numbered according to the order in which they were completed, not in order of importance or preference.

References

"Baseball Savant: Trending MLB Players, Statcast and Visualizations." Baseballsavant.Com, https://baseballsavant.mlb.com/. Accessed 17 Oct. 2022.

Cserepy, Nico, et al. "Predicting the Final Score of Major League Baseball Games." *Economics*, 2015, https://cs229.stanford.edu/proj2015/113_report.pdf.

Everman, Brad. "Analyzing Baseball Statistics Using Data Mining".

Hoeting, Jennifer. Neural Networks in R (Nnet Package). 2020, https://www.stat.colostate.edu/~jah/talks_public_html/isec2020/nnet.html.

Huang, Mei-Ling, and Yun-Zhi Li. "Use of Machine Learning and Deep Learning to Predict the Outcomes of Major League Baseball Matches." *Applied Sciences*, vol. 11, no. 10, 10, Jan. 2021, p. 4499. *www.mdpi.com*, https://doi.org/10.3390/app11104499.

Ibe. "Append Data Frames Together in a for Loop." *Stack Overflow*, 3 July 2018, https://stackoverflow.com/q/29402528.

Karnuta, Jaret M., et al. "Machine Learning Outperforms Regression Analysis to Predict Next-Season Major League Baseball Player Injuries: Epidemiology and Validation of 13,982 Player-Years From Performance and Injury Profile Trends, 2000-2017." *Orthopaedic Journal of Sports Medicine*, vol. 8, no. 11, Nov. 2020, p. 2325967120963046. SAGE Journals, https://doi.org/10.1177/2325967120963046.

Koseler, Kaan, and Matthew Stephan. "Machine Learning Applications in Baseball: A Systematic Literature Review." *Applied Artificial Intelligence*, vol. 31, no. 9–10, Nov. 2017, pp. 745–63. Taylor and Francis+NEJM, https://doi.org/10.1080/08839514.2018.1442991.

Lee, Jae Sik. "Prediction of Pitch Type and Location in Baseball Using Ensemble Model of Deep Neural Networks." *Journal of Sports Analytics*, vol. 8, no. 2, Jan. 2022, pp. 115–26. content.iospress.com, https://doi.org/10.3233/JSA-200559.

Mahdi, Salsabila, et al. "A Review of R Neural Network Packages (with NNbenchmark): Accuracy and Ease of Use." *The R Journal*. Zotero, https://www.inmodelia.com/exemples/2021-0103-RJournal-SM-AV-CD-PK-JN.pdf.

Maszczyk, Adam, et al. "Application of Neural and Regression Models in Sports Results Prediction." *Procedia - Social and Behavioral Sciences*, vol. 117, Mar. 2014, pp. 482–87. ScienceDirect, https://doi.org/10.1016/j.sbspro.2014.02.249.

Mitchell, Margaret, et al. "Model Cards for Model Reporting." *Proceedings of the Conference on Fairness, Accountability, and Transparency*, ACM, 2019, pp. 220–29. DOI.org (Crossref), https://doi.org/10.1145/3287560.3287596.

Neural Networks in R (Nnet Package). https://www.stat.colostate.edu/~jah/talks_public_html/isec2020/nnet.html. Accessed 29 Nov. 2022.

"Ordinal Independent Variables." *SAGE Research Methods Foundations*, SAGE Publications Ltd, 2020. DOI.org (Crossref), https://doi.org/10.4135/9781526421036938055.

Petti, Bill, et al. *Baseballr: Acquiring and Analyzing Baseball Data*. 1.3.0, 9 Sept. 2022. R-Packages, https://CRAN.R-project.org/package=baseballr.

Riederer, Yihui Xie, Christophe Dervieux, Emily. 5.4 Control the Size of Plots/Images. *R Markdown Cookbook*. bookdown.org, https://bookdown.org/yihui/rmarkdown-cookbook/figure-size.html. Accessed 29 Nov. 2022.

Ripley, Brian, and William Venables. *Nnet: Feed-Forward Neural Networks and Multinomial Log-Linear Models*. 7.3-18, 28 Sept. 2022. R-Packages, https://CRAN.R-project.org/package=nnet.

Therneau, Terry, et al. *Rpart: Recursive Partitioning and Regression Trees*. 4.1.16, 24 Jan. 2022. R-Packages, https://CRAN.R-project.org/package=rpart.

Thomas, Gregor. "Answer to 'Append Data Frames Together in a for Loop.'" *Stack Overflow*, 2 Apr. 2015, https://stackoverflow.com/a/29419402.

Tung, David D. "Data Mining Career Batting Performances in Baseball (Preprint)." *Journal of Data Science*, 2012, p. 30.

Uzair, Muhammad, and Noreen Jamil. "Effects of Hidden Layers on the Efficiency of Neural Networks." *2020 IEEE 23rd International Multitopic Conference (INMIC)*, 2020, pp. 1–6. IEEE Xplore, https://doi.org/10.1109/INMIC50486.2020.9318195.

Young, William A., et al. "Determining Hall of Fame Status for Major League Baseball Using an Artificial Neural Network." *Journal of Quantitative Analysis in Sports*, vol. 4, no. 4, Oct. 2008. www.degruyter.com, https://doi.org/10.2202/1559-0410.1131.