**Modeling Baserunning Decisions on Deflected Baseballs in Play**

**SMT Data Challenge**

Team 165

August 1st 2025

## Abstract

The research presented will focus on the baserunner’s decisions after a ball is deflected while in the field of play. The goal is to accurately identify whether a player should take an extra base upon a deflected ball. Real anonymized Minor League Baseball tracking data was cleaned, and new metrics were calculated in order to model when a player should advance. Five models were created using player and ball tracking data and the best model was selected. It was concluded that the Random Forest model was most accurate, reaching an elevated level of 72.39% accuracy. Further analysis was conducted to determine which metrics were most important in the decision for a player to advance. It was found that distance related factors were more important than sprint speeds and throwing speeds. Further research can still be done on this topic by exploring exit velocity and launch angle and their implications on creating deflections.

## 1. Introduction

The game of baseball is won on numerous fields, and decision-making is often done in rapid succession in the heat of the moment. One crucial aspect that is often neglected in the game is effective base running. While many fans tend to appreciate the power of a player like Aaron Judge, or the defensive wizardry of a player like Nolan Arenado, fans and players should have appreciation for the art of baserunning. This forgotten tool utilizes critical decision-making techniques and often dictates where the player will be upon the conclusion of the play, thus having significant implications on total runs scored. One strong indicator of an effective baserunner is their ability to make decisions once a ball is deflected after being put in play, which typically alters the path and velocity of the ball.

To answer when a runner should take the extra base on a deflection; we used play-by-play, game information data, and player-ball tracking to create a model capable of cultivating these decisions.

While watching the Boston Red Sox play against the Baltimore Orioles Ryan O’Hearn hit what should have been a double, but Red Sox outfielder Ceddanne Rafaela overthrew the cutoff causing the ball to hit O’Hearn in the back. This allowed Gunnar Henderson to score from third base and O’Hearn to attempt to move up to third. The Red Sox had a throwing error with the ball going out of play resulting in a Little League home run. After watching this game, it sparked interest when something else like this happened. So, for our project we are looking at plays when the ball deflects off someone in the field of play, analyzing when the baserunner(s) should advance a total of two bases, the one they are going to and an extra base, and creating a model to assist in these decisions.

Figure 1: This is a screenshot from MLB.com of the broadcast, click [here](https://www.mlb.com/video/ryan-o-hearn-doubles-7-on-a-sharp-line-drive-to-center-fielder-ceddanne-r?q=HomeTeamId%20%3D%20%5B111%5D%20AND%20AwayTeamId%20%3D%20%5B110%5D%20AND%20BattingTeamId%20%3D%20%5B110%5D%20AND%20PitchingTeamId%20%3D%20%5B111%5D%20AND%20Season%20%3D%20%5B2025%5D%20AND%20GameType%20%3D%20%5B%22REGULAR_SEASON%22%5D%20AND%20PlayerId%20%3D%3D%20%5B656811%5D%20AND%20HitResult%20%3D%20%5B%22Double%22%5D%20Order%20By%20Timestamp%20DESC&cp=MIXED&p=0) to view the play

**Figure 1**



## 2. Data

The data used in this project was provided by SportsMEDIA Technology (SMT) for the 2025 SMT Data Challenge. The data is two seasons of anonymized Minor League Baseball, and it includes game information (inning, players, game situation), game events (pitch thrown, ball in play, ball thrown, ball acquired, etc), and player and ball tracking data (position coordinates taken every 50 milliseconds for everyone on the field and the ball respectively). The data was very messy with many NA values and didn’t include any game state information. So, we had to make many assumptions about the data, mainly surrounding the game state. Many assumptions were able to be made about the game state thanks to David Awosoga and his animation code. It took the event data and output an animated version of the play that we could view see **Figures 4** and **5**. All NA values were imputed using a form of a group average value; for more information on how the NAs were imputed and metrics were calculated see [Appendix A](#_Appendix_A_–).

With the data provided, we filtered down to only deflection plays using the event code for ball deflection within game events. Then we removed any plays in which the ball deflects off the batter or a baserunner, as this typically results in a dead ball (hit batter or baserunner interference). After narrowing the data set, we utilized player and ball tracking, play-by-play, and game information data to create the metrics necessary for our model. For the player tracking data we only used the position players, batters and baserunners.

## 3. Analysis

### 3.1 Approach

We first wanted to focus on what variables to include in our model. We understood there was a limited number of hours for this project, so we completed only what was necessary for the model. Taking this approach, we looked at metrics for both baserunners and fielders that we thought would affect a baserunner being able to advance: running speed, throwing speed, distance to the base, and distance to the ball. A mix of these metrics were calculated for both the baserunners and fielders.

3.2 Model Variables

**3.2.1 Sprint and Throwing Speeds**

Sprint speeds were calculated for both the baserunners and fielders, in feet per second using a 95th percentile of their relative speeds at a given point in time while the ball was in play. Calculating sprint speeds while the ball is in play helps ensure players are running full speed and not walking around, while using the 95th percentile ensures we are getting as close as possible to a player’s true sprint speed. There are still some occasions when a ball is in play and players don’t have to run. For example, bases are empty, and the batter hits a single to right field, most of the players on the left side of the field don’t have to move or can walk. A similar process was used to calculate the throwing speeds for each fielder in miles per hour.

**3.2.2 Distance to the Base**

The distance to the next base gets more complicated. The fielders and the baserunners require two different distances to be calculated. For the fielders, we used a straight-line distance from point A to point B. However, baserunners have to follow the basepath, so a straight-line distance was calculated to the next base and then an additional 90 feet was added to account for the following base.

**3.2.3 Maximum Distance between Thrower and Ball**

The last variable we calculated was the maximum distance the ball bounced away from the fielder throwing the ball. It wasn’t always the case that the player the ball bounced off made the throw. Whether the ball gets away 2 feet or 20 feet makes a difference in the runner’s decision. To obtain the maximum distance, we calculated the distance between the ball and the player that threw it from the time of deflection until the ball was fielded, selecting the maximum.

3.3 Safe/Out Data

When outs occurred wasn’t given anywhere within the SMT Data. There were two options to get that information: calculate it ourselves or watch animations of each play and decide whether they were safe or out. We found that both of those options were outside the scope of what we could do. So, we did some further research and found a previous year’s project created by Atul Ventakish, Levon Sarian, and Ishan Kinikar. We used their code that calculated when outs occurred by looking at the game information data. The group used a process of elimination to determine when outs occurred, examining a combination of when runners and batters changed. The code only tells us whether an out occurred, not who was out, or how it happened. Under normal circumstances, we would be given the game state data. Our model is binary (safe/out) so, one of our assumptions was that anytime the code gave us an out, we assumed that the lead runner was out. Other assumptions were that the last pitch of an at bat dictated when an out occurred and that if a runner was on-base during one at bat and not the next, they were out.

3.4 Model Selection

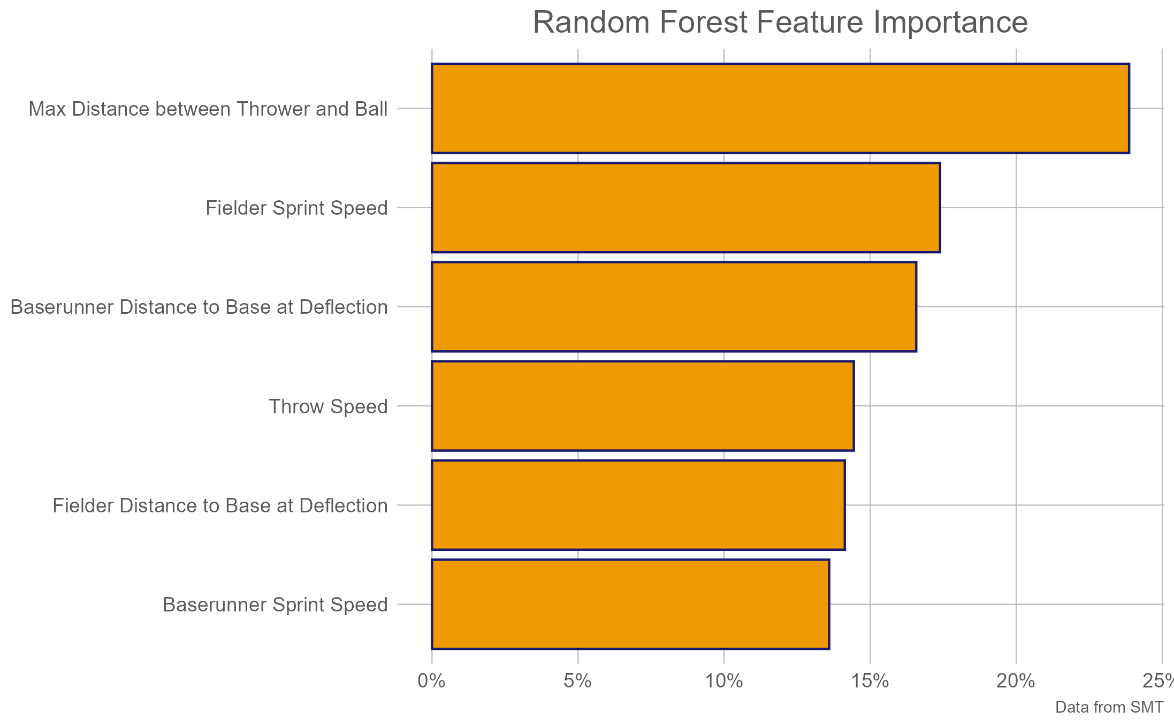
The modeling process was relatively easy compared to the cleaning and calculations. Overall, five different types of machine learning models were created to model the probability of a player advancing two bases (the one they are currently running towards and an extra base). The five were Logistic, Naive Bayes, Decision Trees, Random Forest, and xgBoost (organized from least to most complex). A Logistic model is a form of a linear regression (y=mx+b) and returns a probability. Naive Bayes is a classification model that predicts which category data points belong to, and the last three belong to the same family. Decision Trees use a tree structure with criteria at each point that help classify data points. A Random Forest utilizes multiple Decision Trees to help build a more accurate model. A xgBoost model is like a Random Forest, but it learns from the previous Decision Trees to improve accuracy as it builds more trees.

For all models, a random sample was taken with 75% of the data being used to train the model and the other 25% to test. When looking at the accuracy of the predictions, all the models were within 10% of each other. We decided to use the model with the highest accuracy, which was the Random Forest with 72.39% accuracy. Across the different models the most important features are the distance the baserunner is to the next base and the maximum distance the ball is from the fielder. For more information on the modeling process see [Appendix B](#_Appendix_B_-_1).

## 4. Results

Over the course of the project, we analyzed player and ball tracking data aiming to predict when a player should advance on a deflected ball. After cleaning and modeling the data, we found that the best model was the Random Forest model. Overall, the most important factors to a baserunner’s decision to advance are the maximum distance between the ball and the thrower and the distance to the next base for both the runner and thrower at the time of deflection.

Figure 2: accuracy gained in percent that each variable contributes



**Figure 2**

**Figure 3**

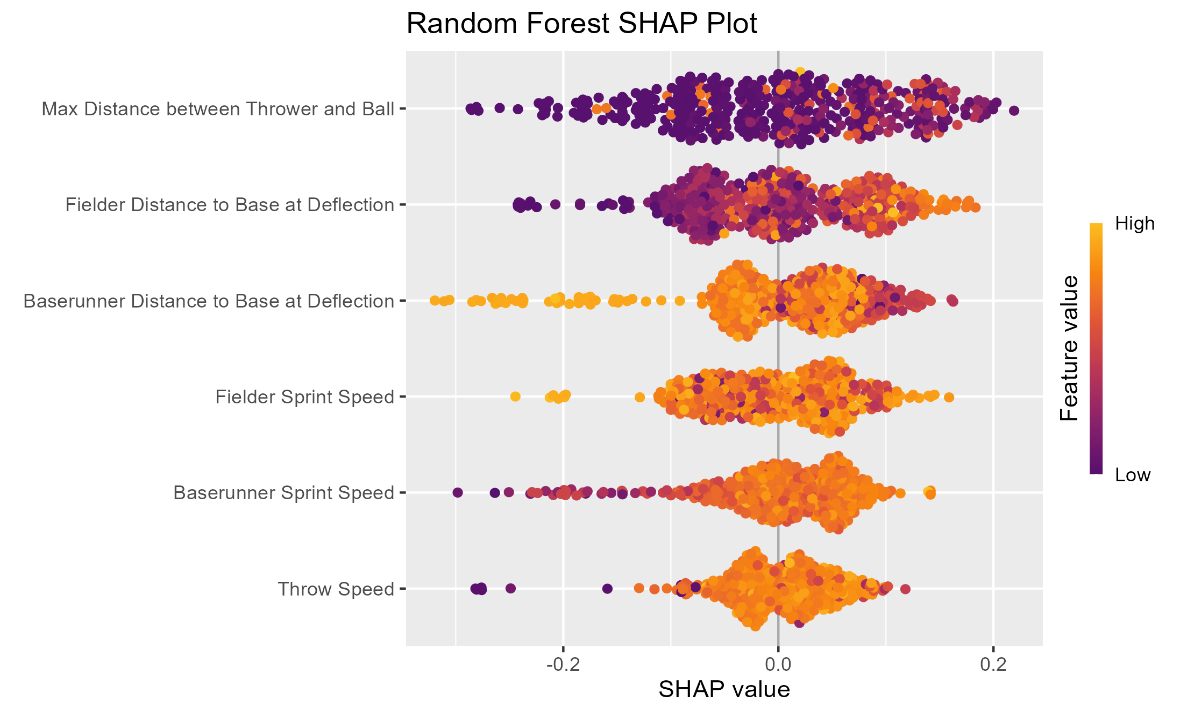


Figure 3: SHAP plot showing the breakdown of each variable’s effect on the probability of being safe

**Figure 4**

Probability Safe: 85%

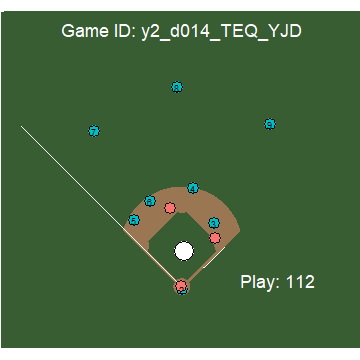


Figure 4: This is an example of a high probability of being safe. The ball deflects off the shortstop and into no man’s land allowing the runner to safely take the extra base

**Figure 5**

Probability Safe: 15%

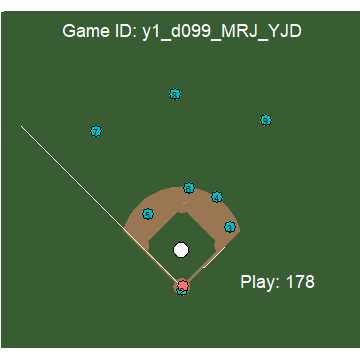


Figure 5: This is an example of a low probability of being safe. This is a routine single to right field. The right fielder momentarily bobbles the ball and quickly gets it in preventing the runner from taking the extra base.

## 5. Further Research

The intriguing aspect of a professional baseball game is the fact that not every ball in play has the exact same output. Outs can be made on the base paths, in the air, or on the ground. This ambiguity offers plenty more additional research in figuring out any trends that perhaps can influence the probability of a ball being deflected on the field of play. One intriguing element to consider is the launch angle of each ball put into play, which indicates the height of the ball from the ground.

Typically, we can infer that any baseball put in play on the ground has a launch angle between 0 and 6 degrees. A ball at a launch angle of 6 – 15 degrees is likely labeled as a low line drive, which could cause the ball to go into the outfield (Rapsodo, 2025). Baseballs with launch angles between 0 to 15 degrees would be the basis for future research. We want to figure out if launch angle directly impacts each play differently, also looking to see if specific position players tend to deflect the baseball more than others given the launch angle of the baseball at point of contact.

Lastly, one element to strongly consider as a crucial piece of causing deflections is exit velocity. Defined as the speed at which the ball comes off the bat at point of contact, exit velocity is a significant sign of an effective batter. Combined with a launch angle, these two elements play a significant role in determining the distance the ball travels (without taking environmental factors into account). We can make the logical assumption that a launch angle of 15 degrees with an EV of 90 MPH has a higher chance of being deflected on the field because of the low line drive – high EV classification. Taking into account the batter’s EV and launch angle for every deflected ball will be helpful for teams to consider when shifting the infield defense.

## Acknowledgements

A special thanks go out to Dr. Meredith Wills, Billy Fryer, and the rest of the SMT team for organizing this challenge. Dr. Wills and Mr. Fryer were instrumental in our success in modeling and explaining sports data. In addition, we would like to thank the following for their significant coding contributions for this challenge: David Awosoga (animation code used to visualize individual plays), Jordyn Geller (play-by-play description code), Atul Ventakish, Levon Sarian, and Ishan Kinikar (code that determined when outs occurred for each play).

## Appendix

### Appendix A – Imputing NA Values

As previously stated in the Data section, SMT’s data is very messy and includes a lot of NA values. That means that when calculating our metrics for our models we also encountered many NA values for each metric. Now there are three ways to deal with NA values. The first is to simply ignore the NA values and proceed as if they weren’t there. The second is to remove the rows that have those values. The third is to impute them with an average value. This third option is what we chose to do as we needed to make sure we had enough data for both each player and the entire set of players.

**Sprint Speeds**

With the player speed, the thought process was that the different positions have different characteristics. For example, you would expect a centerfielder to be faster than a first baseman. Which is why for imputing the player speed NAs we did it with an average speed by position. For each fielding position we followed the same process for the individual players (using a 95th percentile sprint speed). Then we utilized a simple case when statement to replace NAs with the corresponding value. When it came to doing the same for the batters and baserunners, we had to use the average for the entire data set as it was impossible to tell which position was batting or running. This was included in the same case when using a filter for batters and runners. Below are the respective values that replaced the NAs.

**Table 1**

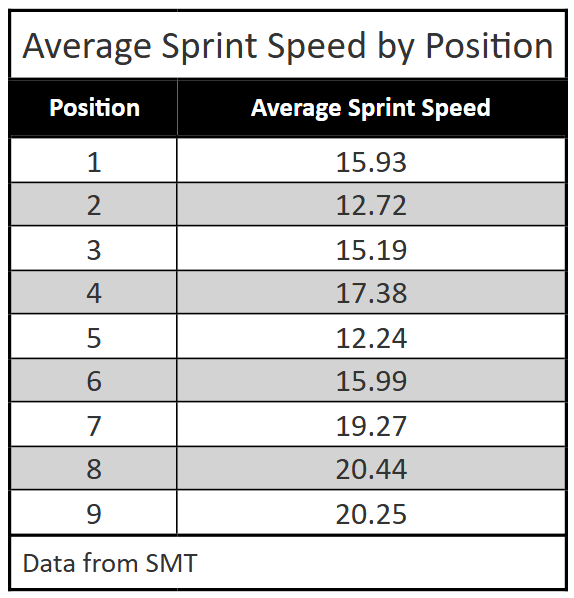


Table 1: Sprint Speed NA replacement values by position

**Throwing Speeds**

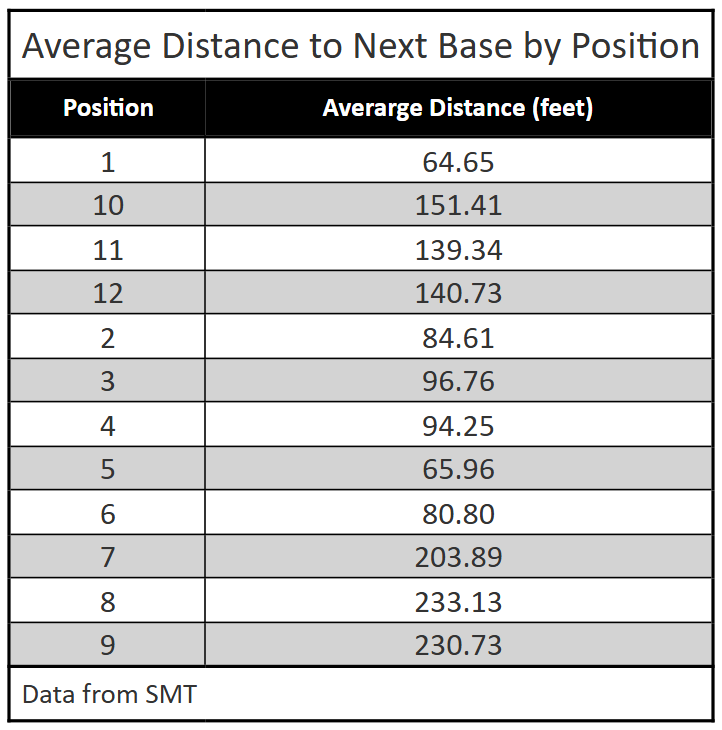
When looking at the throwing speeds, all the positions had similar speeds. So, we just used the mean of the data set since it was around the same as each position’s average.

**Distance to the Base**

For the distances to the next base, we found the average distance for each positional group. Also, we included the baserunners and batters as individual position groups (4 total, one for each possible position). Just like with the Sprint Speed NAs, a simple case when statement was used to impute the values in the data set. Below are the respective values that replaced the NAs.

Table 2: Distance to Next Base NA replacement values by position

**Table 2**

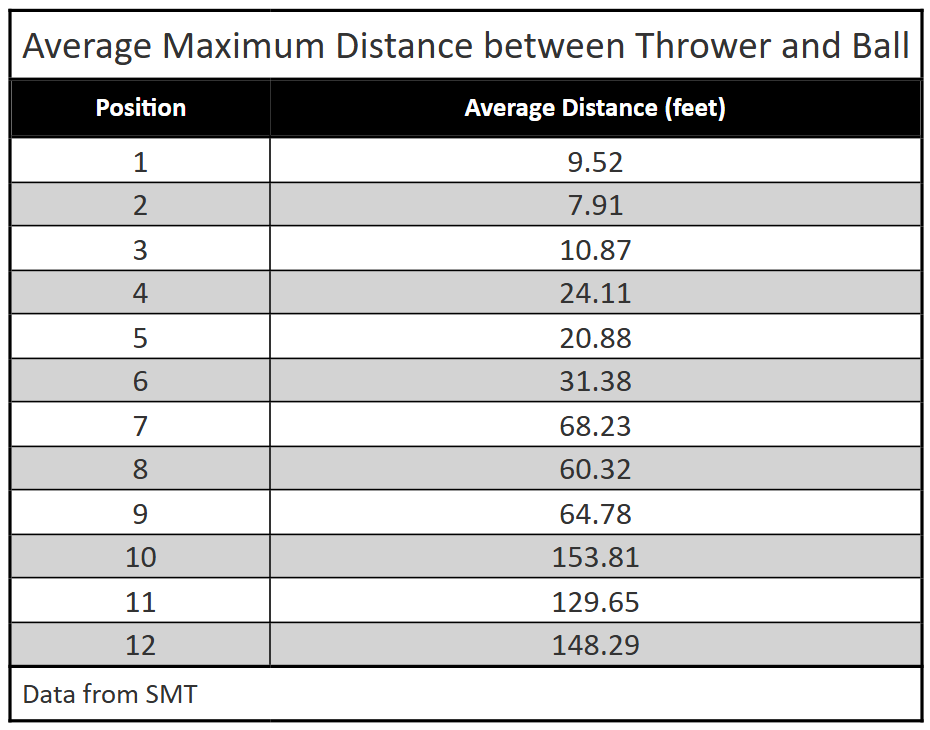


**Max Distance between Thrower and Ball**

Similar to Sprint Speed and Distance to Base we calculated the average distance between thrower and ball for each position, again excluding batters and baserunners. Then again, a simple case when statement was used to impute the values in the data set. Below are the respective values that replaced the NAs.

Table 3: Maximum Distance between Thrower and Ball NA replacement values by position

**Table 3**



## Appendix B - Modeling

Since the data we modeled is binary, we knew we needed to use classification models. We decided to use 5 types of models: Logistic Regression, Naive Bayes, Decision Trees, Random Forest, and xgBoost. Using training data of 75% each model was trained, which left the other 25% to test it. In **Table 4** is the accuracy of each model.

**Logistic Regression**

This model is the simplest and the one with the most interpretability. The results are easy to interpret, it is known exactly which variables are having which effect, and which variables are significant. This was our starting point and allowed us to double check our variables. With our Logistic Regression model, we first saw the significance and other metrics included in the model summary. We then went further to test accuracy and multicollinearity. Multicollinearity checks for correlation amongst the variables, within the Logistic Regression model an assumption is that the explanatory variables are not correlated. This test is a way to check this assumption by calculating the variance inflation factor (VIF) for each variable. After testing we found our model satisfied this assumption.

**Naïve Bayes**

This is one of the simpler classification models. Being a classification model, there isn’t a nice formula like with a Logistic Regression model. With this comes a lack of interpretability. The results are shown in a classification matrix where accuracy was calculated using the diagonal.

**Decision Trees, Random Forest, and xgBoost**

These three classification models are being grouped together as they all are a type of Decision Tree. Starting with the simplest, Decision Trees. This model has some interpretability as each node of the tree contains criteria that aid in the decision. We decided not to limit the number of branches as we wanted to allow for the highest accuracy possible, but this prevented us from being able to have a good visual of the tree as it was impossible to read the criteria. To calculate our model’s accuracy, we followed what we did with the Naïve Bayes model

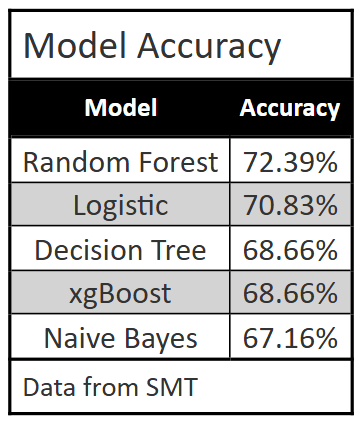
While the Random Forest model lacks interpretability it gains accuracy from the standard Decision Tree. This is done by creating multiple trees. For our model we started using 100 trees. Then we tried different numbers of trees to see the effect it had on the model. After some experimenting, we found that 500 was the best number for us as it optimized the accuracy, Kappa, p-value. The Random Forest model takes response variables that are of factor type. So, we had to change our Safe/Out data from numeric to factor type.

The xgBoost model expands on the Random Forest. It creates multiple trees and learns from the previous trees to gain accuracy. When it comes to interpretability they are about the same. To decide the number of rounds of boosting we looked at the root mean squared error (RMSE). Starting at 100 rounds of boosting we had a list of the RMSE after each round. We found the first occurrence of the minimum RMSE and used that as the number of rounds. For our model that was at 19 rounds of boosting. If we decided to go past 19 it could lead to overfitting the model. After optimizing the number of rounds we performed no additional tuning to the model.

Overall, we decided to use the Random Forest model. It had the highest model accuracy. We were able to better interpret the model using visualizations (see **Figures 2** and **3** in [Results](#_4._Results)). They allowed us to see which variables were important to the model and how each variable individually affected the probability of the baserunner being safe.

Table 4: Accuracy in percent of the different models created

**Table 4**



## Sources

“Statcast Sprint Speed Leaderboard.” *Baseballsavant.com*, baseballsavant.mlb.com/leaderboard/sprint\_speed.

James, Gareth, et al. *An Introduction to Statistical Learning : With Applications in R*. Springer, 2013.

GeeksforGeeks. “XGBoost in R Programming.” *GeeksforGeeks*, 14 July 2020, [www.geeksforgeeks.org/r-language/xgboost-in-r-programming/](https://www.geeksforgeeks.org/r-language/xgboost-in-r-programming/). Accessed 17 July 2025.

KoshurAI. “Understanding Log Loss: A Comprehensive Guide with Code Examples.” *Medium*, 9 Jan. 2024, koshurai.medium.com/understanding-log-loss-a-comprehensive-guide-with-code-examples-c79cf5411426. Accessed 17 July 2025.

“RPubs - an R Intro to RandomForest.” *Rpubs.com*, rpubs.com/mdwybron/487157.

“Xgb.importance Function - RDocumentation.” *Rdocumentation.org*, 2017, [www.rdocumentation.org/packages/xgboost/versions/0.6-4/topics/xgb.importance](https://www.rdocumentation.org/packages/xgboost/versions/0.6-4/topics/xgb.importance).

“XGBoost R Tutorial — Xgboost 1.5.0 Documentation.” *Readthedocs.io*, 2021, xgboost.readthedocs.io/en/release\_1.5.0/R-package/xgboostPresentation.html. Accessed 23 July 2025.

“How to Cite R and R Packages.” *Ropensci.org*, ropensci.org/blog/2021/11/16/how-to-cite-r-and-r-packages/.

“GitHub - Atuav10/2024-SMT-Data-Challenge: Determining the Right Time to Pull a Pitcher - 2024 SMT Data Challenge Winning Project.” *GitHub*, 2024, github.com/Atuav10/2024-SMT-Data-Challenge. Accessed 23 July 2025.

“Stay-Or-Go/Modeling.R at Main · Billyfryer/Stay-Or-Go.” *GitHub*, 2015, github.com/billyfryer/stay-or-go/blob/main/Modeling.R.

Rapsodo. (2025, July). *Rapsodo Baseball Hitting Certification*. *Rapsodo Certification Course*.

“Boston Red Sox Schedule.” *MLB.com*, 2024, [www.mlb.com/redsox/schedule/2025-05](https://www.mlb.com/redsox/schedule/2025-05).

“Bee Swarm with SHAP Values for Random Forest in R.” *GeeksforGeeks*, 12 July 2024, [www.geeksforgeeks.org/machine-learning/bee-swarm-with-shap-values-for-random-forest-in-r/](https://www.geeksforgeeks.org/machine-learning/bee-swarm-with-shap-values-for-random-forest-in-r/). Accessed 28 July 2025.

“Option to Align the Table Header — Opt\_align\_table\_header.” *Rstudio.com*, 2020, gt.rstudio.com/reference/opt\_align\_table\_header.html. Accessed 28 July 2025.

“GitHub - ModelOriented/Shapviz: SHAP Plots in R.” *GitHub*, 17 July 2025, github.com/ModelOriented/shapviz. Accessed 28 July 2025.

Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). “Welcome to the tidyverse.” \_Journal of Open Source Software\_, \*4\*(43), 1686. doi:10.21105/joss.01686 <https://doi.org/10.21105/joss.01686>.

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Iannone R, Cheng J, Schloerke B, Hughes E, Lauer A, Seo J, Brevoort K, Roy O (2025). \_gt: Easily Create Presentation-Ready Display Tables\_. R package version 1.0.0, <https://CRAN.R-project.org/package=gt>.

Arnold J (2024). \_ggthemes: Extra Themes, Scales and Geoms for 'ggplot2'\_. R package version 5.1.0, <https://CRAN.R-project.org/package=ggthemes>.

Milborrow S (2024). \_rpart.plot: Plot 'rpart' Models: An Enhanced Version of 'plot.rpart'\_. R package version 3.1.2, <https://CRAN.R-project.org/package=rpart.plot>.

Therneau T, Atkinson B (2025). \_rpart: Recursive Partitioning and Regression Trees\_. R package version 4.1.24, <https://CRAN.R-project.org/package=rpart>.

Liaw A, Wiener M (2002). “Classification and Regression by randomForest.” \_R News\_, \*2\*(3), 18-22. <https://CRAN.R-project.org/doc/Rnews/>.

Chen T, He T, Benesty M, Khotilovich V, Tang Y, Cho H, Chen K, Mitchell R, Cano I, Zhou T, Li M, Xie J, Lin M, Geng Y, Li Y, Yuan J (2025). \_xgboost: Extreme Gradient Boosting\_. R package version 1.7.9.1, <https://CRAN.R-project.org/package=xgboost>.

Lüdecke et al., (2021). performance: An R Package for Assessment, Comparison and Testing of Statistical Models. Journal of Open Source Software, 6(60), 3139. https://doi.org/10.21105/joss.03139

Richardson N, Cook I, Crane N, Dunnington D, François R, Keane J, Moldovan-Grünfeld D, Ooms J, Wujciak-Jens J, Apache Arrow (2025). arrow: Integration to 'Apache' 'Arrow'\_. R package version 19.0.1.1, <https://CRAN.R-project.org/package=arrow>.

Drucker R (2024). \_sportyR: Plot Scaled 'ggplot' Representations of Sports Playing Surfaces\_. R package version 2.2.2, <https://CRAN.R-project.org/package=sportyR>.

Pedersen T, Robinson D (2024). \_gganimate: A Grammar of Animated Graphics\_. R package version 1.0.9, <https://CRAN.R-project.org/package=gganimate>.

Borchers H (2023). \_pracma: Practical Numerical Math Functions\_. R package version 2.4.4, <https://CRAN.R-project.org/package=pracma>.

Chang W (2025). \_webshot2: Take Screenshots of Web Pages\_. R package version 0.1.2, <https://CRAN.R-project.org/package=webshot2>.

Mock T (2025). \_gtExtras: Extending 'gt' for Beautiful HTML Tables\_. R package version 0.6.0, <https://CRAN.R-project.org/package=gtExtras>.

Tuszynski J (2024). \_caTools: Tools: Moving Window Statistics, GIF, Base64, ROC AUC, etc\_. R package version 1.18.3, <https://CRAN.R-project.org/package=caTools>.

Kuhn, M. (2008). Building Predictive Models in R Using the caret Package. Journal of Statistical Software, 28(5), 1–26. https://doi.org/10.18637/jss.v028.i05

Meyer D, Dimitriadou E, Hornik K, Weingessel A, Leisch F (2024). \_e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien\_. R package version 1.7-16, <https://CRAN.R-project.org/package=e1071>.

Kassambara A (2023). \_ggpubr: 'ggplot2' Based Publication Ready Plots\_. R package version 0.6.0, <https://CRAN.R-project.org/package=ggpubr>.

“Shap Plots for Random Forest Models.” *Stack Overflow*, 21 Dec. 2020, stackoverflow.com/questions/65391767/shap-plots-for-random-forest-models.

Mayer M (2025). \_shapviz: SHAP Visualizations\_. doi:10.32614/CRAN.package.shapviz <https://doi.org/10.32614/CRAN.package.shapviz>, R package version 0.10.2, <https://CRAN.R-project.org/package=shapviz>.

Mayer M, Watson D (2025). \_kernelshap: Kernel SHAP\_. doi:10.32614/CRAN.package.kernelshap <https://doi.org/10.32614/CRAN.package.kernelshap>, R package version 0.9.0, <https://CRAN.R-project.org/package=kernelshap>.

“Ryan O’Hearn’s Little League Home Run.” *MLB.com*, 25 May 2025, www.mlb.com/video/ryan-o-hearn-doubles-7-on-a-sharp-line-drive-to-center-fielder-ceddanne-r?q=HomeTeamId%20%3D%20%5B111%5D%20AND%20AwayTeamId%20%3D%20%5B110%5D%20AND%20BattingTeamId%20%3D%20%5B110%5D%20AND%20PitchingTeamId%20%3D%20%5B111%5D%20AND%20Season%20%3D%20%5B2025%5D%20AND%20GameType%20%3D%20%5B%22REGULAR\_SEASON%22%5D%20AND%20PlayerId%20%3D%3D%20%5B656811%5D%20AND%20HitResult%20%3D%20%5B%22Double%22%5D%20Order%20By%20Timestamp%20DESC&cp=MIXED&p=0. Accessed 15 July 2025.