

Thou Shalt Not Steal (Bases):

When to Break Baseball's "11th Commandment"

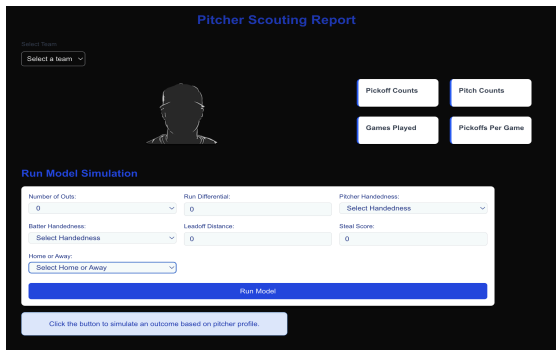
Team 90

Abstract

Identifying patterns in pickoff behavior transforms a traditionally instinctive part of baseball into a data-driven edge, allowing managers and teams to make smarter decisions about baserunning aggression, scoring opportunities, and overall game strategy. The goal of this project is to evaluate the intent of a pitcher to pickoff and thus give an advantage to baserunners. An application was developed to produce scouting reports for each pitcher on the three longitudinally identified teams in our data set.

These scouting reports include important pitching metrics such as the number of games played, pitches thrown, pickoffs attempted, and rate of pickoffs per game as well as a comparison to other pitchers at the same minor league level. These features had to be derived from low-level 2D player tracking and 3D ball tracking data. For users to check these derivations, the application has an interface to randomly sample plays from constructed data sets of pickoffs, steals, run scoring, and recorded out plays to visualize these plays with animated GIF images. The complete scouting report ultimately includes a model that simulates the probability of a pickoff on a play when given a set of game state circumstances and has the potential to produce a game-influencing advantage.

The images below are linked to the web application and inspiration for the project (left to right respectively). See Appendix A for details on using the app.



The screenshot shows a web application titled "Pitcher Scouting Report". It features a "Select a team" dropdown menu, a silhouette of a pitcher, and buttons for "Pickoff Counts", "Pitch Counts", "Games Played", and "Pickoffs Per Game". Below these is a "Run Model Simulation" section with input fields for "Number of Out", "Run Differential", "Pitcher Handedness", "Batter Handedness", "Leadoff Distance", "Steal Score", and "Home or Away". A "Run Model" button and a link to "Click the button to simulate an outcome based on pitcher profile" are also present.



I. Introduction

A pitcher and a runner have a relationship akin to dance partners. They are engaged in a tango full of intensity, deception, and mutual respect. The only check and balance that keeps the runner from bolting is the pickoff. However, if the runner knows when the pitcher will and will not pickoff, they gain control of the waltz, dictate the rhythm, and turn the pitcher's movements into predictable steps.

All team sports have unique situations that allow for head to head competition to occur. Basketball has isolation plays, soccer has penalty kicks, and hockey has faceoffs. There is something raw and captivating about two people—abiding by rules both written and unwritten—who attempt to find every edge to best their opponent.

I find the dance of the pitcher and the runner to be the most fascinating of all, especially due to its everchanging nature. With the MLB rule changes in 2023, specifically regarding the enlarged bases and disengagement limitations, the dance looks more different than ever. As a result, there is a heightened interest in baserunner dynamics within the analytics community and the larger baseball world. Data intensive projects can already determine ideal leadoff distance, evaluate a catcher's ability to stop the run, and even analyze sliding abilities of runners. However, there is no evidence of a project that aims to assess the probability that a pitcher will pick off on a given play.

I created an app that predicts the probability that a pitcher will attempt a pickoff at first base by isolating situations with a runner on first base and second base open. The application uses the data from (i) the number of outs, (ii) run differential, (iii) pitcher handedness, (iv) batter handedness, (v) runner leadoff distance (from first base), (vi) steal+, and (vii) if the pitcher is on the home team to inform its predictions. By surfacing the predicted probability of a pickoff attempt in real time, the application provides runners and coaches with a powerful, data-informed tool to calibrate baserunning aggression by turning instinct into strategy and giving teams a competitive edge on the basepaths.

II. Data

The data provided by SMT includes anonymized 2D player positioning and 3D ball tracking across MiLB.

There are four main play types used in this project and will be referred to throughout the paper: pickoffs, steals, run scoring, and recorded outs. Each of these play types can be visualized with animated GIFs in my application. Fig 1.1 shows a still frame from a pickoff GIF.

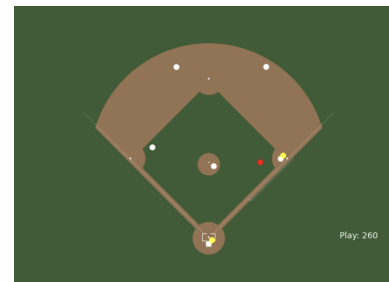


Fig. 1.1 Screenshot of a pickoff play from an animated gif

Pickoffs: Perhaps the most straightforward plays to procure from the data are pickoffs. The “game events” data have event codes for a pickoff, making it easy to tell which play of a given game resulted in a pickoff. Unfortunately, many other game dynamics require far more nuanced interpretation and complex modeling to extract meaningful insights.

Steals: This project uses stolen base plays for evaluating the stealing tendency of each runner in our dataset. To do this, both the game information and game events data are used to find plays with the following characteristics: a runner on first base with second base open, a pitch received by the catcher without contact from the batter, and the runner on first base appearing on second the following play. This method proved to be both effective and highly reliable in identifying true steal attempts.

Run scoring: The dataset does not provide information on whether a run gets scored or not, so the approach to find run scoring plays was to use the player position data to find plays in which baserunners came within two square feet of home plate. The critical unknown is that runners can also be within two square feet of the plate...and be out. Consequently, any play at the plate that is actually an out is assumed to be a run. Similarly, a situation could exist with a bases loaded, 6-4-3 double play that would end the inning, and as such, if the player on third runs a hard 90, the data would show him to have reached the plate and scored a run. The project continues with these scoring assumptions and relies on the relative infrequency of plays at the plate. This could imply an overestimation on the number of runs scored each game.

Recorded out: Lastly, it was critical for this project to find the number of outs at the start of each play. The identifying method was to find instances where, if the batter changed without reaching base or runner(s) disappeared without the inning ending, then each of those instances resulted in an out recorded. Accordingly, the number of outs on the following play were incremented. This method is more at the mercy of flawed data than the previous ones. Firstly, it is the method most reliant on the quality and consistency of data on consecutive plays (i.e. baserunners and batters). However, there are significant amounts of missing data in the set, so consecutive rows of data may not necessarily be consecutive plays in the game. Also, each play in the data is supposed to have a unique identification number, but there are many sections in the data with consecutive plays that all have the same ID number, which results in plays that say 15 outs were recorded. Finally, this method does not distinguish between outs and runs. If a baserunner is no longer on the basepath, the following play is deemed an out, although a run could have been scored instead. In this project, as many data inconsistencies as possible were resolved while still maintaining the remaining assumptions based on the premise that most of the player disappearances are likely due to outs.

III. Feature Engineering

Two features used in the model, outs and run differential, were computed by summarizing the number of outs and runs scored on each play as determined by the assumptions in the

previous section. Whether or not the pitcher is pitching at home is determined by whether it is the top or bottom of the inning as they are pitching (top of the inning implies the pitcher is home). Pitcher handedness is determined by using the 3D ball positioning data and finding from which side of the rubber the pitcher consistently released the ball. Similarly, batter handedness is determined using the player positional data and finding which box the batter stood in most consistently (while ignoring switch hitters). The first baserunner's lead distance was calculated by finding the distance of his center of mass to the front left corner of first base, three seconds before the pitch is released. Lastly, steal+ is a measure of a runner's tendency to steal. The score is higher relative to the more bases a runner has attempted to steal, and it is penalized for the repeated times a runner has been on base. This metric is normalized at 100, so the average baserunner has a steal+ of 100. If a runner has never stolen, he would have a steal+ of zero.

IV. Modeling

Four different classification models were tested (logistic regression, k-nearest neighbors, random forest classifier, and XGBoost classifier) to simulate the probability of a given play resulting in a pickoff. The "f1 score" was chosen to evaluate the quality of the models because it takes into account how many predicted pickoffs are actual pickoffs and how many actual pickoffs were correctly predicted to be pickoffs. The model with the highest f1 score was the random forest classifier, so that is used in the app's interactive model. See Appendix B for model results.

V. Results and Analysis

The model consistently performed well at identifying when a pickoff was not going to happen (true negatives), but struggled to accurately predict when a pickoff would occur (true positives). See Appendix C for the confusion matrix. While this might initially seem like a shortcoming, it's actually not a critical flaw in this particular use case. If a model has a low number of positive predictions (i.e. predicting a pickoff), it means the probabilities of pickoffs are generally low which makes baseball sense. Pickoffs make up a small percentage of plays in baseball, so this model does not and should not ever say that there is a 90% chance of a pickoff on play. Instead, it aims to report more marginal changes in the likelihood that should inform the level of aggressiveness of the baserunner. For example, consider the difference between a 5% and a 25% chance of a pickoff. Neither is likely, but one is significantly less likely than the other.

VI. Scouting Application

The heart of the project is a full stack scouting report application. See Appendix A for details on navigating the app and Appendix D for the technical background on its design.

Fundamentally, the application is a scouting report for pitchers on the three longitudinally identified teams in our dataset that compares their statistics to the other pitchers that we have data on. A user can select a team and a pitcher from that team, and a report is generated. This

report includes that handedness of the pitcher as well as a graph showing where they compare to the rest of the pitchers on four key metrics: number of pickoffs attempted, number of pitches thrown, number of games played, and average number of pickoffs per game. Figure 1.2 shows one of the four graphs displayed on each report.

The app also has an interactive pickoff prediction model.

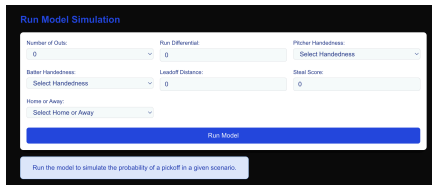


Fig. 1.3 Pickoff prediction model interface

The user can input the aforementioned game state information and run the model to simulate the probability that a pitcher picks off on that play.

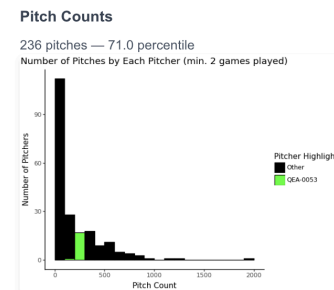


Fig.1.2 Total # of pitches thrown by pitcher QEA-0053 compared to all other pitchers in the dataset

Lastly, the application has several tabs to randomly sample plays from constructed data sets of pickoffs, steals, run scoring, and recorded out plays to visualize them and evaluate the assumptions made with animated GIF images.

VII. Future Work

Data issues aside, I want to improve how the model accounts for contextual nuance introduced by the new disengagement limitations and enlarged bases which are rules that have fundamentally shifted pitcher and runner behavior. Currently, the model doesn't explicitly track disengagement count per play which is a huge opportunity. If a pitcher has already used up both disengagements, he's far less likely to attempt a pickoff, and that should massively affect the pickoff probability. Similarly, runners may lead off more aggressively after those disengagements, increasing the pressure and changing the dynamics entirely. Including this rule-based context as a feature would likely increase the model's predictive power and real-world utility.

I would also like to add nuance to my evaluation of a runner's stealing ability. The steal+ statistic only scores on the basis of steal attempts and opportunity. A runner's sprint speed, ability to successfully steal bases (not just attempts), and history of getting picked off are excellent indicators of his true ability to cause disruption on the basepaths and these factors are likely to be more influential on a pitcher's intent to pickoff.

VIII. Acknowledgments

Thank you to SMT for hosting the Data Challenge and providing access to player tracking data. I'm grateful to Billy Fryer and Dr. Meredith Wills for their eager willingness, tireless joy, and drive to help me succeed, and to Dr. Debbie Stephens Stauffer for holding me accountable with

deadlines and countless proofreading run-throughs. Lastly, thanks to David Awosoga and the amazing Python animation code he developed for use in the Data Challenge.

IX. Citations

Vercel. “Next.Js Docs.” Next.Js, nextjs.org/docs. Accessed 1 Aug. 2025.

“FASTAPI.” FastAPI, fastapi.tiangolo.com/. Accessed 1 Aug. 2025.

User, GitHub. “Python API.” DuckDB, duckdb.org/docs/stable/clients/python/overview.html. Accessed 1 Aug. 2025.

“1. Supervised Learning.” Scikit, scikit-learn.org/stable/supervised_learning.html. Accessed 1 Aug. 2025.

Appendix A

Application Workflow: <https://smt-2025.vercel.app/>



Appendix B

Machine Learning Modeling Test Evaluation

Model Type	Accuracy	Precision	Recall	F1 Score
<i>Logistic Regression</i>	0.465	0.099	0.717	0.174
<i>Random Forest</i>	0.778	0.148	0.383	0.214
<i>K-Nearest Neighbors</i>	0.692	0.133	0.528	0.212
<i>XGBoost</i>	0.593	0.115	0.622	0.194

Appendix C

Random Forest Test Data Confusion Matrix:

1712 (TN)	396 (FN)
111 (FP)	69 (TP)

Appendix D

The three main stacks are a DuckDB database, a FastAPI backend, and a Next.js frontend. I chose DuckDB and the two frameworks due to familiarity, simplicity, and largely, efficiency.

DuckDB is my database engine, and it solved data acquisition issues from day one. Baseball data is not known to be classified as “Big Data,” but the player and ball tracking data is not your typical baseball data. The scale of the SMT data required an efficient solution to access it both for the app and for the exploratory data analysis. DuckDB allows you to load all of your data into a “persistent” database that is stored as a binary file. DuckDB is optimized for analytical queries, so once I loaded the SMT data into the database file, every query I wrote to

access the data took less than one second to execute. This dramatically sped up my analysis project, allowing me to have any information I desired in an instant.

FastAPI runs the backend API: it handles how data is processed and sent to the user. I chose it because it's both incredibly fast and intuitive to build with. It allows for easy creation of new endpoints and includes a clean interactive documentation interface for testing locally before deploying changes. It utilizes "asynchronous processing," which means I can serve complex predictions or data queries without blocking the app. That responsiveness matters when users are exploring multiple visualizations or simulating probabilities because they get near real-time feedback even with complex computations.

Next.js handles the frontend and web app framework. I wanted a simple introduction to using TypeScript and HTML, and Next.js provides an excellent framework and packages for this. With Next.js, I can serve pages quickly and efficiently, and preload heavy components only when needed, which is key for interactive elements like animated GIFs or scouting report visualizations. It also provides an intuitive file structure that allows me to quickly add pages and grow my application.

Together, these tools let me build a responsive, maintainable, and performant platform that makes it easy to work with deep analytics in real-time all without unnecessary overhead.