Refining Stuff+ for MLB: An Auto-Tuned Ensemble with Surrogate Run Value from Pitch-Level Data

John Davis and Michael Rabayda

***Abstract*—We present a framework that translates publicly available Hawkeye pitch-level data into a real-time, play-by-play simulation tool whose performance approaches that of proprietary Major League Baseball (MLB) "Stuff+" systems. Leveraging ˜4.6 million pitches (2018–2023) for training and ˜710 000 pitches (2024) for out-of-sample evaluation, we (i) engineer physics-based descriptors of velocity, movement and release geometry, (ii) denoise the Run Value target with a boosted-tree surrogate that predicts batted-ball outcomes from launch metrics, and (iii) model the resulting signal with an Optuna-tuned XGBoost regressor augmented by a residual LinearGAM. The ensemble is rescaled to the industry-standard Stuff+ index ($\mu = 100$, $\sigma = 10$) to enable intuitive interpretation. On unseen 2024 data the model explains 13.3% of pitch-by-pitch Run Value ($R^2 = 0.133$), more than doubling existing open-source baselines (3–7%) and matching the lower bound of front-office implementations. Because all features are pitch-intrinsic, the metric remains stable across seasons and can be computed with sub-second latency, supporting applications in injury-risk monitoring, in-game decision support, and cross-level talent identification.**

***Index Terms*—Machine Learning, Baseball Analytics, Stuff+, Run Value, XGBoost, LinearGAM.**

I. Introduction

In the early 2000's, Major League Baseball experienced a renaissance fueled by the introduction of sabermetrics into the sport. Sabermetrics is broadly defined as the empirical analysis of baseball, primarily through statistics, used to evaluate player performance and develop winning strategies. In short, each franchise began to invest heavily into deep statistical analysis in order to maximize effectiveness as an organization. The 2011 hit film "Moneyball" starring Brad Pitt and Jonah Hill highlights the rise of advanced data analysis for the 2002 Oakland Athletic's season. Based on a true story, this organization leveraged basic machine learning algorithms to simulate player results and transform an underfunded franchise into a playoff contender[1]. Since this time, machine learning within the scope of baseball has completely taken off. Each franchise spends millions of dollars a year to employ large teams of data scientists, build elite level ML pipelines and build out cloud infrastructure/distributed systems to house such data.

At the MLB level, each team has access to their own proprietary data to build out data. This data is mainly captured using an advanced camera tracking set up called Hawkeye systems. This high speed camera system is set up around the field and shoots every player and the ball / bats at 900 frames per second for every pitch. While this data is not readily available to the public, the MLB has their own API that releases a condensed version of this data in discrete form. Meaning instead of 900 frames per second, it has one row for each pitch and greatly reduced set of columns that are available for the public. Now, for this project, the goal was broadly to be able to simulate a pitch result based on this hawkeye pitch data alone. This processing of modeling pitch events based on only pitch level information is not new. Every single MLB team has some sort of proprietary algorithm that looks at pitch level characteristics like speed, movement and release and gives a grade to pitchers. This idea was formalized by Max Bay, a neuroscientist with a PhD from the University of Southern California [4]. Major league teams were so impressed with this idea that he was hired in a full time capacity by the Houston Astros to help lead data initiatives over the past several seasons. Going into this project, our group was well aware that we would not be able to fully match the model success of major league baseball franchises however we sought out to optimize what is possible using the concepts taught over the course of this past semester in machine learning.

As mentioned above, pitch level modeling has been prevalent in Major League Baseball for the last 5-7 years. Why is this the case? Simply put, baseball is a probabilistic sport. Many basic surface level stats have low signal in year to year progressions. Similarly, in pitching, location which is of utmost importance is largely a mystery still to this day to MLB teams. There is little carry over year to year in location metrics like strike% and teams are only beginning to leverage new computer vision technology in order to understand miss tendencies and further explore location related statistics. The case for pitch level data is that these algorithms are sticky year to year. The main cause in changes to these stats for a given player are age related decay or acute injury. This is important to note because in a probabilistic game with low signal, any stat that carries year to year will have immense value for organizations as a whole. These algorithms are mainly used to inform developmental initiatives as well as scout prospective talent. The magic of pitch level modeling is that if you can obtain data on any player from high school, college, minor and majors, you can deploy the algorithms

to see exactly where they stand in relation to the average MLB player. More recently, teams have invested in building streaming pipelines to ingest realtime hawkeye data, run it over the machine learning algorithm and receive real time low latency updates into how the pitcher is performing. Why does this matter? As injury rates continue to spike as well as teams try to maximize efficiency, it's important to understand exactly when a player reaches a fatigued state. By leveraging these pitch level algorithms in real time, monitoring drop off is paramount for informing when to keep a pitcher in the game or when to remove them.

II. Column Details

Before we can get into the details of the algorithms, there are some basic terms that the reader must understand:

A. Terminology

- **RunValue**: A metric that quantifies the impact of a specific event (like a pitch or batted ball) on the expected number of runs scored in a game. It is calculated as the difference in expected run value before and after the event, reflecting the net change in a team's scoring potential. It accounts for context, including the number of outs, base runners, and the inning, providing a more precise measure of a player's impact on the game.
- **TaggedPitchType**: In Baseball, you can throw a variety of different pitches that move in different ways. Each pitch is tagged based on movement to a column called "TaggedPitchType".
- **RelSpeed**: The velocity of which the ball leaves the pitcher's hand. Fastball's have a higher velocity compared to pitches that utilize spin like a slider or curveball.
- **SpinRate**: Measured in revolutions per second, this measures the amount of spin a ball has as it is thrown towards home by a pitcher.
- **InducedVerticalBreak**: This column measures the amount of vertical movement a pitcher imparts on baseball resisting or assisting gravity. A fastball will have positive InducedVerticalBreak and spin pitch like a curveball will likely have negative InducedVerticalBreak.
- **HorizontalBreak**: This column measures how much a pitch moves side to side from the pitcher's perspective. A positive Horizontal Break number will correspond with the pitch moving left to right while a negative will correspond with right to left.
- **ReleaseHeight**: This column measures the height of which the pitcher releases the ball towards home. Each player has unique mechanics, some release it lower and some release it higher.
- **ReleaseSide**: This column corresponds to how far right or left from the center of the mound a pitcher releases the ball. More right is associated with positive ReleaseSide and more left is associated with more negative release side. 0 is the center of the rubber.
- **Extension**: This column corresponds to how far the pitcher releases the ball towards home. More extension corresponds to being closer to the plate at the time of release.
- **Launch Angle**: This column corresponds to the angle at which the ball leaves the hitter's bat.
- **Exit Speed**: This column corresponds to the speed at which the ball leaves the hitter's bat.
- **Distance**: This column refers to the distance a batter hits the ball into play.
- **Bat Speed**: This column describes the speed at which the batter swings the bat. This column was open sourced by the MLB in 2024 as a part of their recent initiative to make more data publicly available.
- **Stuff+**: A learned metric that looks at how well a pitch should perform based on speed, movement and release characteristics. Normalized to 100, meaning 100 is league average, this model is used to inform pitch changes as well as pitch calling decisions [2].

III. Task

Develop a professional grade Stuff+ model trained on pitches from 2018–2023 that can explain 10–15% of RunValue for our 2024 data. Most open source models that you can find online only explain around 3–7% of Run Value, but better proprietary models can reach scores of 10–20% in some cases. Our goal is to improve upon the widely used XGBoost framework by engineering new features, refining our Run Value target, capturing residual patterns, and fine tuning the model parameters to create the best performance evaluator we can.

IV. Data Processing

After calling the correct MLB API, the user is provided with .csv data for any range of specified dates since the inception of Hawkeye data systems. For this specific project, our training set will be all pitches thrown between the 2018 season and the 2023 season. The test/validation set will be all pitches thrown during the 2024 season. The main goal of our data preprocessing step was filtering out only the columns that were needed which are mentioned above and then map run values for each distinct pitch event using the delta_run_exp column provided by MLB. After this was done, we decided to take a different approach than what is considered the norm for many stuff algorithms. Most algorithms use static run values of each distinct play event meaning a single, a double, a groundout etc. all have a value associated with it. We identified correctly that using discrete run values for balls hit into play introduces excess noise into the model. The reason for this is that a pitcher can do his job meaning induce weak contact though the batter

can still get a hit which sways the run value falsely toward the batter. Conversely, a pitcher can give up hard contact which is not good and still get the batter out which falsely sways run value directionally toward the pitcher. To combat this, we built a predictor algorithm that estimates the run value of only balls in play given the **launch angle**, **exit speed** and **distance** of the hit. We used a boosted tree regression model which had an R squared of .37 meaning 37% of the variance in run value of balls in play can be explained by launch angle, exit speed and distance. After creating and deploying the model then concatenating it with non balls in play pitches, we noticed an increased signal in our final model.

We then created a few more variables to account for things like different handedness between the pitchers. Some pitchers throw with their right hand while others throw with their left hand. Our model had to be hand agnostic meaning any horizontal X axis stats had to be standardized before training the model. The 2 main columns that are affected by this are horizontal break and horizontal release. The next column added is called differential break which is the absolute value difference between the amount of movement backspin imparts on the baseball and the amount of side to side movement that the spin imparts on the baseball. This is only relevant for fastballs as the pitcher's goal is to either make the ball spin a lot on the Y axis (four seam fastball) or try to cut vertical movement and make it run on the X axis (sinker). Pitch's that have the same amount of vertical break as horizontal break are referred to as "deadzone". These pitches are generally much easier for a hitter to hit which is why we are measuring the distance each pitch is from this exact point of the same parts vertical break and horizontal break. As mentioned above, stuff algorithms look at speed, movement and release metrics. For fastball pitches, we used differential break as our movement feature. The last column we built was for non primary / fastball pitches. This column takes the differential speed between the average speed of the pitcher's fastball and average speed of the given pitch. One way a pitcher can deceive a hitter is by changing the speed of their pitches. We used this differential speed column as a way to measure this in the model.

As mentioned above, bat speed as a column was recently open sourced by MLB in 2024 for public use. We wanted to be able to include this in some form within our model as bat speed has a high correlation to positive results for the hitter. The inverse is also true meaning if the pitcher can slow down the bat of the hitter there is a directional sway in run value associated with that. Since not every pitch garners a swing, we had to build a secondary nested model for expected swing speed(xBatSpeed) based on pitch speed and movement. Obviously, this was done separated by pitch type as to what slows or speeds up a pitch changes based on what pitch is being thrown.

V. Building the Model

Now, we have everything we need to begin the larger modeling portion of this project. By now, we have split up the dataset into 8 distinct dataframe based on the pitch type column and have every column we need to be able to train this model. Each MLB organization has their own proprietary version of this algorithm however most are training using a simple decision tree regressor. We were well aware that if we wanted to match the R^2 of private models that had larger scopes of features and more data we would have to leverage different techniques than what is the status in modern day pitch modeling. Our research first brought us to a python library called Optuna which is a hyperparameter optimization framework. This library leverages a technique called Bayesian parameter optimizations which leverages past trial data to explore the hyperparameter space and suggest optimization hyperparameter combinations. The massive bonus of this library is that it supports parallel processing which means it runs rather quickly compared to other hyperparameter tuning algorithms. For our introductory algorithm,we leveraged an XgBoost regressor which is the industry standard for these models. Next, we also leveraged LinearGAM, a python model framework to capture residual patterns left over from XGBoost that might miss nonlinear relationships within the data. Our final predicted run value is a weighted ensemble between an XgBoost regression model and the LinearGAM model.

VI. Algorithm Selection

Xgboost (extreme gradient boosting) Regression: supervised machine learning algorithm used to predict continuous values. Xgboost leverages ensemble learning by stacking layers of decision trees to make predictions on new data. Unlike traditional decision tree models, which can often suffer from either high bias or high variance, Xgboost leverages a technique called gradient boosting to iteratively train weak learner's (shallow decision trees) in a sequential manner. Each new tree is trained to correct the residual error of the past tree which iteratively increases the model accuracy. You can set a custom loss function for minimization which balances error on the training data to prevent overfitting. These algorithms are the standard for most baseball related machine learning projects due its speed and performance. The Xgboost library in python has numerous hyperparameters such as max_depth which is maximum tree depth and n_estimators which is the number of trees. As mentioned above, we used Optuna to effectively explore the hyperparameter space and boost model performance.

LinearGAM (generated additive model): supervised machine learning algorithm that extends linear models by allowing for nonlinear relationships between features and the target variables. Standard linear regression assumes linear relationships between the input and target variables. LinearGAM

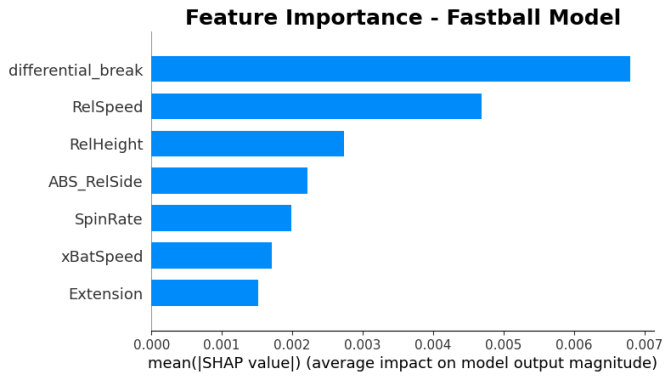

Fig. 1. Feature importance for the Fastball model, as measured by the mean absolute SHAP values. This plot highlights the relative contribution of each pitch characteristic to the Stuff$^+$ score, with differential_break and Release Speed emerging as the most impactful features. These attributes significantly influence pitch effectiveness by capturing key aspects of movement and velocity, aligning closely with physical metrics known to drive pitch success.

Top 10 MLB Pitchers by Stuff+ for the 2024 Season

Pitcher	Team	Stuff+	Run Value	ERA	K%
Clase, Emmanuel	CLE	131.55	-15.66	0.61	24.4%
Doval, Camilo	SF	120.45	-6.33	4.88	28.8%
Fairbanks, Pete	TB	116.24	0.09	3.57	23.8%
Jansen, Kenley	BOS	115.47	-5.01	3.29	28.4%
Thompson, Ryan	AZ	115.38	-5.41	3.26	19.1%
Sandlin, Nick	CLE	114.18	10.22	3.75	27.6%
Díaz, Alexis	CIN	114.16	-0.52	3.99	22.7%
Estrada, Jeremiah	SD	113.55	-11.01	2.95	37.3%
Uribe, Abner	MIL	113.43	0.79	6.91	21.2%
Anderson, Grant	TEX	113.24	2.77	8.1	24.2%

Fig. 2. Top 10 Pitchers by Stuff+ for the 2024 season (Minimum 500 pitches thrown). This table shows the top 10 pitchers in the league, with Stuff+ capturing the physical effectiveness of the pitch and Run Value reflecting the overall run prevention impact. Higher Stuff+ and more negative Run Values indicate superior performance.

leverages smooth, piecewise spline functions to capture non-linear / complex patterns in data. This becomes important in baseball modeling especially because features can have a non-monotonic relationship with a target variable which means that the direction of the relationship can change based on the value of the target variable rather than just increasing or decreasing as a whole. Each feature in a linearGAM model can have its own unique smooth function which allows for better interpretability within the model. LinearGAM's are robust to overfitting due to its use of regularization, penalizing overly complete spline functions. These models have started to gain traction within the baseball community due to its effectiveness in scenarios where you're trying to capture subtle, context dependent relationships in noisier data. In our project, we used LinearGAM to capture residual patterns that may have been missed by the xgBoost model, providing a complementary layer to correct any mistakes from the xgboost model.

Ensemble Learning: Machine learning technique that combines a series of base models (weak learners) in order to create a more accurate and robust model. The main concept here is that each model will have some form of its own bias and variance, in combination with each other, they do a better job at reducing both and generalizing better on unseen data. In our project, we used an ensemble approach that combined the prediction from the xgboost model and the linearGAM to produce a balanced and accurate model.

VII. Scaling Results

Now that we have a predicted run value metric, there are certain steps we must take in order to make sure it matches the industry standard for Stuff+. Stuff+ or any model with a "+" at the end in baseball is scaled similarly to IQ which means that the mean is 100 and generally each standard deviation is 10. So that means that roughly 68% of all pitches thrown fall between 90 and 110 while 95% of all pitches thrown fall between 80 and 120 and so forth following a normal distribution. So, how did we scale our model to meet this standard? First, since a negative run value is better for the pitcher and we want the best pitcher to have the highest stuff+, we need to make the best pitch the most negative pitch and scale it so the worst possible pitch is zero. To do this, we subtract each point in the data by the maximal positive value (the worst pitch) to get run value scaled negative then we take the absolute value of this. After doing this process, we can scale the values to 100 by taking each individual value dividing by the mean of the given set and multiplying by 100. Lastly, we scaled each set meaning each subset of pitches separated by pitch type to have a mean of 100 and a standard deviation of 10.

VIII. Validating the Model

Now that we have a working model, we can test it over 2024 unseen data and test the R^2. As mentioned above our goal was to build a model that returned similar results to proprietary organizational models even without having access to the full scope of their data. Our intentional evaluation of the R^2 between run value and our Stuff+ algorithm returned .131 which is fantastic for any open source model. Baseball is a highly probabilistic sport and to be able to explain 13% of variance in a pitch event by just looking at openly available pitch level data is very good. We noticed that as we increased sample requirements meaning looked only at starters with over 500 pitches in a season, the R^2 continued to climb meaning that our model was performing exceedingly well as it saw large amounts of new data.

IX. Regression Results

Figure 3 confirms that our metric carries practical signal into unseen data. Trained on 4.6 million pitches from 2018–2023 and evaluated on 710 000 2024 pitches, the model produces a clear monotonic relationship: higher Stuff$^+$ scores are associated with increasingly negative Run Values (i.e. better run prevention). The fitted line yields $R^2 = 0.13$, mirroring the 13.3 % variance explained in the formal regression and more than doubling the 3–7 % range reported for previous open-source efforts. The tight confidence band highlights the

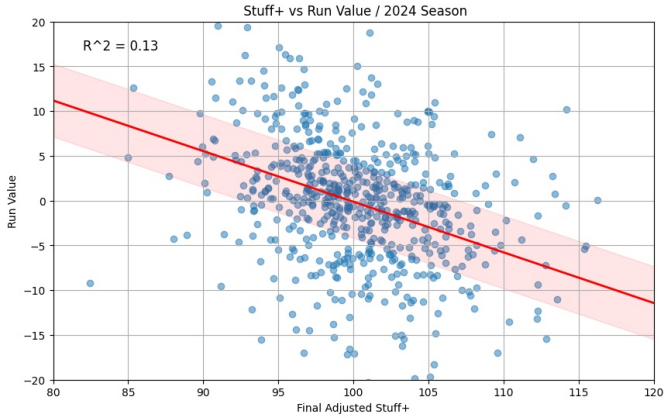

Fig. 3. Relationship between rescaled Stuff$^+$ and pitch-level Run Value on the 2024 hold-out set (≈710,000 pitches). The solid red line is the ordinary–least-squares fit, and the pink band is the 95 % confidence interval.

stability of the effect across the large test sample, underscoring the metric's suitability for real-time monitoring and cross-season comparisons.

```
================================
Dep. Var: RunValue   R-squared: 0.133
Model: OLS           Adj. R-squared: 0.131
Method: LS Approx.   F-stat: 94.30
Date: 05/12/25       Prob (F-stat): 7.69e-21
Time: 08:53:30       Log-Lik: -2131.7
No. Obs: 619         AIC:  4267.
Df Resid: 617        BIC: 4276.
Df Model: 1
CovarType: nonrobust
================================
```

X. Error Analysis

Despite the impressive results achieved by our own model, it is essential to analyze the sources of error to identify areas for further improvement. It is important to understand these errors to not only highlight the limitations of our current approach but also provide guidance for future iterations of the model.

A. Data Leakage and Model Bias

One of the primary challenges in this project was accounting for data leakage in the context of our training and test set. The challenging part of building these algorithms using years of pitch level data is that although you correctly can split the data into training data and a separate set of testing data, the model can learn certain patterns in the data. What this means is that if a pitcher is present in the training set even from years prior though explicitly not known, the model may recognize them in the test set based on pitch level characteristics causing potential overfitting. One way to combat this is the use LOOCV (Leave one out cross validation) which holds one sample from the data to test on and avoid this issue. We experimented with this in our project however given the large amount of data, this was too computationally expensive to be able to fully implement. Similarly, XgBoost is known for its high accuracy however when the number of estimators (trees) is too large, these models can overfit as well. Fortunately, optuna has built in library functions to correct this.

B. Heteroscedasticity and Non-Stationarity

Another significant issue is the presence of heteroscedasticity in the run value data, where the variance of errors is not constant across different pitch types or contexts. This is particularly evident in certain pitch types such as sinkers, where we noticed that small changes in release side can have outsized effects on predicted pitch outcome. Additionally and more importantly, the MLB environment itself is not stationary. Pitch speeds, batter approaches, and ball composition are in a constant state of adjustment, introducing drift that can degrade model performance if not regularly accounted for. In short, new trends arise every year or two until the "market" stabilizes.

C. Outliers and Noisy Observations

Outliers are a significant source of error in our current approach. While our model captures the central tendency of pitch behavior well, extreme values such as extremely high or low velocity pitches or unusually large breaks can disproportionately influence the regression fit. Additionally, batted ball events, even after denoising, are inherently noisy due to factors like ballpark dimensions, defensive alignment, and weather conditions, which are not included in the feature set. In the future, some integration of park factors based on dimensions, altitude and weather can be used to increase model performance.

D. Recap

In summary, while our model demonstrates strong predictive power, addressing these error sources will be critical to further improving performance and ensuring robustness in deployment

XI. Further Work

Now that we have set a basis for being able to predict long run pitch outcomes using pitch level data alone, how exactly can we improve these model results in the future? Similar to current LLM discourse, many people have suggested that simply scaling up the model meaning increasing the number of parameters the model is exposed to with current architecture will deliver better results. Although this may certainly provide some benefit, new architecture and modeling techniques will

be needed to drive a significant increase in model performance. As mentioned above, baseball is a probabilistic sport. Similar to the stock market, these models are generally low signal. So how can we leverage new techniques in order to increase model accuracy?

A. Computer Vision

One of the most likely innovations will come from implementation of computer vision technologies to measure things like pitcher occlusion, the ability to hide the ball from the hitter. These technologies also allow for more advanced object tracking such as hitter bat tracking, catcher glove tracking and more. If we can begin to understand how certain pitches effect how the hitters bat moves, there may be signal there in terms of increasing pitch success.

B. Seam Effects

Another promising direction for improving pitch-level modeling is the incorporation of seam effects. Research by Driveline Baseball and Barton Smith has shown that the orientation and movement of the baseball seams can significantly influence the aerodynamic properties of a pitch, including Magnus force, wake deflection, and seam-shifted wake effects. These subtle aerodynamic changes can dramatically alter pitch movement, particularly for pitches like two-seam fastballs, sliders, and splitters that rely on asymmetric spin profiles [3]. Understanding and quantifying these effects could help refine model accuracy by better capturing pitch-specific flight dynamics. Future work should include machine learning techniques capable of detecting seam orientation and its influence on pitch trajectory.

C. Biomechanical Data

Incorporating biomechanics data offers another promising path for enhancing pitch-level modeling. Hawkeye systems has the ability to track joint angles, arm slot, stride length, hip-shoulder separation, and more which are all critical factors in pitch velocity, movement, and release consistency. Integrating these metrics can potentially reduce model error by accounting for previously unmeasured sources of variability. Additionally, combining these data with machine learning can identify subtle movement inefficiencies that may contribute to injury risk or performance inconsistency, further improving both predictive power and real-world applicability.

D. Recap of Further Work

The future of pitch modeling will require new techniques and technologies to be able to adjust to new league trends. Integration of computer vision technologies, a deeper understanding of the role seam effects plays into pitch success as well as the integration of biomechanics data all offer promise into a more robust and accurate model.

XII. Conclusion

This work demonstrates that publicly released Hawkeye pitch-tracking data, when paired with modern feature engineering and ensemble learning, can yield a play-level performance metric that rivals proprietary front-office solutions. By (i) replacing noisy discrete outcomes with a physics-informed surrogate Run Value for batted-balls, (ii) standardizing movement metrics across handedness, and (iii) stacking an Optuna-tuned XGBoost with a residual LinearGAM, we explain 13.3% of pitch-level Run Value on out-of-sample 2024 data, which is over twice the signal captured by existing open-source models. The resulting Stuff+ scale preserves interpretability (mean 100, SD 10) and delivers sub-second inference, enabling real-time bullpen management, injury-risk monitoring, and cross-level talent benchmarking.

Several limitations remain. The framework omits pitch-location, game-state, and biomechanical variables that franchises collect internally; integrating these features could further improve explanatory power. In addition, the boosted-tree surrogate for batted-ball Run Value assumes stationary launch-angle distributions that may drift with league-wide hitting trends. Future work will explore online re-training, uncertainty quantification for decision support, and reinforcement-learning extensions that optimize pitch sequencing rather than isolated pitch quality. We also plan to open-source a streaming implementation backed by Apache Kafka and Spark Structured Streaming to facilitate adoption by amateur and collegiate programs.

Ultimately, we show that near-proprietary insights can be achieved with transparent, reproducible methods, a step toward democratizing advanced baseball analytics beyond the walls of MLB front offices.

References

[1] M. Lewis, *Moneyball: The Art of Winning an Unfair Game*, W. W. Norton & Company, 2003.

[2] Rockland Peak Performance, "What is Stuff+ and How Can It Help You?" [Online]. Available: https://rocklandpeakperformance.com/what-is-stuff-and-how-can-it-help-you/. [Accessed: May 13, 2025].

[3] Driveline Baseball, "More Than What It Seams: An Introduction to Seam-Shifted Wakes and Their Effect on Sinkers." [Online]. Available: https://www.drivelinebaseball.com/2020/11/more-than-what-it-seams-an-introduction-to-seam-shifted-wakes-and-their-effect-on-si

[4] C. de Jesus, "Astros add analytics whiz Max Bay to R&D department,"*Houston Chronicle*,Feb. 21, 2024. [Online]. Available: https://www.chron.com/sports/astros/article/houston-astros-max-bay-r-d-18378678.php

[5] S. Magnusson, "Avoid the Dead Zone: An Extensive Analysis of the Relationship Between Fastball Stuff Characteristics and Utility Through Four Logistic Regression Models." [Online]. Available: https://www.seemagnus.com/blog-posts-test/avoid-the-dead-zone-an-extensive-analysis-of-the-relationship-between-fastball-stuff-ch