CITY UNIVERSITY OF NEW YORK
SCHOOL OF PROFESSIONAL STUDIES
Master of Science in Data Science

Predicting Major League Baseball Player Home Run Outcomes Using Recency-Based Data

By
Joseph Bochnik

Capstone Project

Capstone Professor: George Hagstrom

December 2025

Table of Contents

Abstract

This project builds a machine learning framework to predict daily Major League Baseball (MLB) home runs using a player's recent performance, pitcher trends, park factors, weather conditions, and sportsbook odds. Because it is widely debated whether hitters and pitchers go through hot and cold streaks, or if these streaks are just random fluctuations, the model focuses on short-term form rather than long-term, excluding season- and career-long stats, to see whether a streakiness effect can lead to profitable models. Data was collected through the SportsRadar API and stored in a MySQL database, combining player, pitcher, weather, and betting information. Several models, including Categorical Boost (CatBoost), Random Forest, XGBoost, LightGBM, and Neural Networks, were first tuned on all available data using TimeSeriesCV from sklearn, and then with a walk-forward backtesting approach to better reflect real-world prediction conditions. Model accuracy was evaluated using log loss, precision, recall, and simulated betting returns. Bets were selectively placed by computing the model's "edge" relative to the sportsbook's odds to increase profitability across two betting frameworks. Results show that while no model produced consistent full-season profitability, several demonstrated a clear predictive signal and generated meaningful returns under specific conditions. Profitable opportunities were found only at higher edge thresholds, in specific time windows of the season, and for particular players whom the models identified as systematically mispriced by sportsbooks. These findings show that although widespread profitability is difficult in such a low-frequency, high-variance outcome, focusing on recent performance, matchup context, weather effects, and sportsbook pricing can uncover targeted betting edges. Overall, this research demonstrates that machine learning models can beat sportsbook odds not broadly, but in narrow, well-defined situations.

Introduction

Baseball has long been regarded as America's pastime. With that title comes not only national attention but also decades of data collection—few sports rival baseball in terms of statistical prominence (Mordor Intelligence, 2018). Every pitch, swing, and play is recorded, producing a vast landscape of information that has fueled everything from player evaluation to predicting game outcomes and even informing in-game strategy, such as when to make a pitching substitution or call for a bunt or steal.

The modern era of baseball analytics traces back to Bill James and the development of sabermetrics in the late 1970s and 1980s, which popularized the use of advanced statistics to evaluate players beyond traditional metrics (James, 1982). Since then, sabermetrics has evolved into an essential part of the game, embraced by both front offices and researchers. With so much publicly available data, many have attempted to build predictive models using advanced machine learning techniques to forecast player outcomes and in-game events (Elfrink, 2018; Smith, 2016).

However, these models often rely on datasets stretching back several decades. While long-term data offers depth and volume, it may not fully represent the realities of the modern game. Baseball has evolved dramatically over the years, from training methods and ballpark dimensions to equipment and even the construction of the baseball itself, meaning that older data may not accurately reflect current conditions (Smith, 2016).

As a lifelong baseball fan who has closely followed the sport for over 15 years, I have also observed a phenomenon that traditional long-term datasets often struggle to capture: players' streakiness. Hitters frequently experience “hot” and “cold” stretches, where their performance fluctuates significantly over a matter of weeks or even days. Similarly, players can have breakout, All-Star caliber seasons one year, only to struggle to maintain a roster spot the next. For instance, Chris Davis of the Baltimore Orioles went from a 2015 All-Star slugger who signed a major contract to a player fighting to remain in the league by 2017. Such extreme short-term fluctuations raise an important question: can recent performance serve as a more accurate predictor of immediate outcomes than long-term historical averages?

Prior studies have acknowledged the challenge of accounting for randomness and momentum in baseball performance (Elfrink, 2018). Some researchers have experimented with weighting recent games or seasons more heavily to capture player form, but this project takes the idea further by focusing almost entirely on recent player performance as the primary predictive signal (Sun, Lin, & Tsai, 2022).

This project seeks to test that hypothesis. Specifically, it aims to build a predictive model that estimates the probability of a player hitting a home run on a given day using only recent performance data. In addition to player-level statistics, the model incorporates contextual variables such as weather conditions (e.g., humidity, temperature, and wind, which are known to influence ball flight distance), park factors, sports book prop odds, and pitcher/bullpen matchups (Ashoff, 2018; DraftKings MQP, 2021).

The ultimate goal is not only to create an accurate predictive model but also to evaluate its practical utility within the context of sports betting markets. By combining short-term player trends with contextual factors, the model aims to identify the most likely home run candidates for a given day. This recency-biased, matchup-based approach may better capture a player’s real-time form and thus offer both predictive accuracy and potential betting profitability that long-term historical models cannot achieve.

Literature Review

Baseball has long been one of the most data-rich sports, inspiring decades of research into player performance and predictive analytics. The field of sabermetrics, pioneered by Bill James, introduced the idea of using quantitative measures to evaluate players and strategies beyond traditional statistics such as batting average or runs batted in (James, 1982). Later research expanded these ideas through mathematical modeling to estimate game outcomes and player contributions. For example, Smith (2016) applied Markov chains to simulate baseball outcomes, demonstrating how probabilistic modeling can capture aspects of in-game performance. These foundational works established the basis for the modern relationship between data science and baseball analytics.

In recent years, researchers have increasingly applied machine learning methods to baseball prediction problems. Elfrink (2018) tested several algorithms using Major League Baseball data and found that while predictive performance improved over simple baselines, baseball's inherent randomness limited model accuracy. Sun, Lin, and Tsai (2022) used Long Short-Term Memory neural networks to predict player performance, showing that sequential models can leverage temporal dependencies, though they still rely primarily on season-long averages. The DraftKings Major Qualifying Project Team (2021) used large-scale player data to develop predictive tools for fantasy sports applications, illustrating how machine learning continues to influence decision-making in baseball-related industries.

Environmental and contextual factors have also received significant attention. Ashoff (2018) examined weather variables such as temperature, humidity, and wind, finding measurable impacts on batted-ball distance and home run rates. Similarly, Kagan and Atkinson (2004) showed that relative humidity alters the coefficient of restitution of baseballs, affecting how far they travel when hit. These findings demonstrate that external conditions can substantially influence offensive outcomes and therefore represent important predictors in home run modeling.

Beyond player statistics and environmental variables, researchers have explored psychological and behavioral dimensions of player performance. Raab, Gula, and Gigerenzer (2013) analyzed large datasets of hitting streaks and found evidence supporting the "hot hand" phenomenon, suggesting that short-term momentum may meaningfully affect a player's likelihood of success. Extending this idea, Bock, Maewal, and Gough (2012) analyzed 28 hitting streaks of 30 or more games across multiple MLB seasons and found statistical evidence that a "hot" hitter's success can positively influence teammates' batting performance, a phenomenon the authors described as "contagious hitting". Their findings supported the existence of a measurable "statistical contagion effect," suggesting that momentum and social influence may spread within lineups. Together, these studies provide empirical support for the existence of short-term performance streaks in baseball and underscore the relevance of recency effects in modeling player outcomes.

Despite the growing body of research, most predictive systems still depend on long-term or aggregated data, which may dilute the influence of recent player trends. Few studies have focused on daily home run prediction using only short-term performance indicators alongside contextual factors such as weather and betting market information. This gap highlights an opportunity to test whether a model emphasizing recent player and pitcher performance, combined with external contextual variables, can produce more accurate and practically relevant forecasts than traditional season-level approaches.

Data & Methodology

Data Collection

Data was collected from several sources. The majority of the information came from the SportsRadar API, which provided daily box score data for the 2025 Major League Baseball (MLB) season. Additional datasets included complete lists of players, venues, umpires, and sportsbook betting odds for player prop bets, along with game-level weather information. I also incorporated park factor data from the MLB Statcast website and merged it with the SportsRadar dataset to account for stadium-specific effects on hitting outcomes.

All data was initially stored as JSON files and then imported into a SQL database. Python scripts were used to collect and process the JSON data, inserting one row per hitter per game that contained each player's rolling averages for key statistics over the past 21 days. To prevent data leakage, only statistics from games played prior to each game date were used. A similar approach was applied to pitching data, where each row included the starting pitcher's rolling statistics and the team bullpen's rolling averages over recent games. Records without sufficient historical data to calculate rolling averages were first set to null and then later on all missing set to 0 for numeric and "Unknown" for categorical to show that no data was available for that player in the last 21 days. This was done to show that the player has not played in at least 21 days as I feel a 21-day absence is significant and I want my models to stay away from players like this due insufficient recent data for my recency-based models. I followed the same logic for starting pitchers and opponent team bullpen statistics.

After data ingestion, a consolidated dataset was generated in Python and exported as a CSV file. This master file contained one row per hitter per game and included the hitter's rolling averages, weather conditions, home plate umpire assignment, opposing pitcher and bullpen rolling averages, park factors, and player prop odds from sportsbooks.

Data Preparation

After all data was collected and stored in the SQL database, I prepared the dataset for modeling through a series of SQL and Python-based steps. The primary objective of this process was to ensure that each game-day record accurately reflected only information available before that day's games, avoiding any form of data leakage.

All player hitting and pitcher statistics were first stored in the database at the individual game level. Each game entry captured a player's full box score line, allowing for detailed performance tracking over time. From this raw data, Python scripts were used in conjunction with SQL queries to generate the training datasets. During this process, rolling averages and rate-based metrics were computed dynamically for each player using a 21-day lookback window. The queries aggregated all available stats from the 21 days prior to each game date meaning if only 5 games were available, then it would just use the last 5 games. If no qualifying data existed within that window, the resulting fields were recorded as null values (later set to 0 or "Unknown"). This was intentional, as the modeling framework is designed to recognize and handle recent performance gaps, reflecting the recency-biased nature of the approach and avoiding unreliable predictions for players with insufficient recent history.

In addition to the player and pitcher data, contextual features were merged into the main dataset, including weather variables (temperature, dew point, humidity, wind speed, wind direction, and roof type), venue park factors, and sportsbook player prop odds. To better quantify the impact of wind on batted-ball outcomes, I also collected the stadium bearing from home plate for each venue from MLB Statcast. Using this bearing and the forecasted wind direction, I created a derived feature indicating the wind effect categorizing whether the wind was blowing in, out, left, or right relative to home plate. This feature was designed to capture one of the most important weather-related influences on home run probability.

The final dataset contained one row per hitter per game, combining each player's recent offensive performance with contextual factors relevant to that day's matchup. Overall, the dataset included over 100 variables across hitting, pitching, weather, and betting domains. The target variable, *hit_hr*, was binary, indicating whether the player hit a home run in that game. Because home runs are relatively rare, the dataset was naturally imbalanced toward non-home run outcomes. Rather than oversampling or reweighting, I chose to keep the data in its original form to maintain realistic class proportions for evaluating model performance under real-world conditions.

Handling Missing Data

An initial review of the dataset revealed several columns with missing or incomplete values. As shown in Figure 1, the majority of missing data came from a small subset of features. Several aggregate columns such as *total_runs*, *total_hits*, *total_walks*, and similar "*total_*"* fields contained 100% missing values. These variables were remnants from earlier data pulls and did not contain any usable information, so they were removed entirely from the dataset prior to modeling.

Among the remaining columns, moderate levels of missingness (roughly 7–26%) were observed in several pitcher-related features (e.g., *sp_sum_hr, sp_sum_bb, sp_sum_er*) and in various sportsbook prop markets (e.g., *to_hit_first_home_run, to_record_4_or_more_hits, to_hit_2_or_more_home_runs*). These missing prop odds are common in real betting data and usually indicate that sportsbooks did not offer that particular market for a player often because the player was a longshot for the event. Rather than discarding these observations, missing prop odds were imputed with a value of 0 to explicitly capture the absence of a betting line. This allows the model to interpret the lack of available odds as a meaningful signal that the player was unlikely to achieve that outcome.

Other columns with minor missingness (less than 1%) included a handful of ballpark and weather-related factors such as *park_factor, stadium_orientation, and wind_relative*. Because the proportion of missing data was so small and upon further inspection due to games that were played at special sites; like the opening series in Tokyo and the series at the Little League World Series in Williamsport, these values were either set to 0 if they were numeric or "Unknown" if they were categorical. This will tell the model that the data was missing and should provide some useful information.

The column *to_hit_a_home_run* is the odds column from the sportsbook for that particular player to hit a home run on that day, had about 4500 missing rows. Because this feature is central to calculating profit and model performance, I decided to drop rows with missing home run odds. This was done because trying to impute the values or setting them as 0 would skew the betting simulations. Overall, it was just safer to leave these rows out as they most likely were missing due to sportsbooks not posting odds for that player on that day and therefore, they would not be viable bets anyway and should be excluded.

Overall, the missing data handling strategy balanced data retention with interpretability. Columns containing no usable data were dropped, while missing prop odds were explicitly encoded as zero to preserve informative absence. All other variables retained their null values, consistent with the model's recency-based framework that treats missing performance data as meaningful gaps in recent player activity.

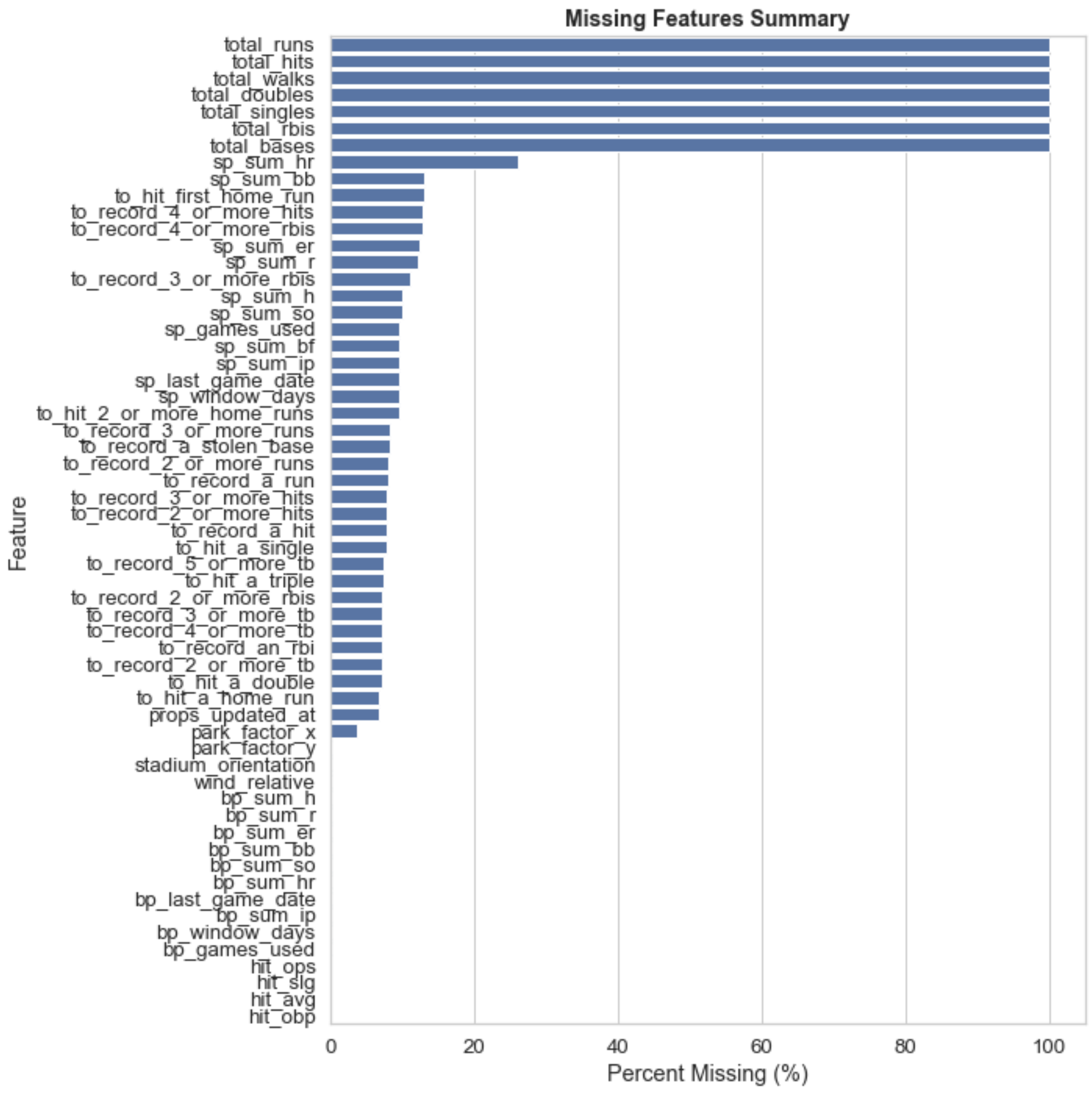

Figure 1: *Missing features in dataset by percentage.*

Data Exploration

Before building the predictive models, I conducted exploratory data analysis (EDA) to better understand the structure, quality, and relationships within the dataset. This step focused on identifying key patterns among the hitting, pitching, and contextual variables and on assessing the degree of class imbalance in the target variable, *hit_hr*.

The final dataset contained over 100 variables and covered every hitter-game instance from the 2025 MLB season. The target variable, *hit_hr*, was binary, indicating whether the player hit a home run in that game. As expected, the dataset was highly imbalanced, with home runs occurring in only a small percentage of player-game records. A frequency plot of *hit_hr* (Figure 2) clearly showed that the majority of observations corresponded to games where no home run was recorded. This imbalance was anticipated given that home runs are rare events in baseball and was taken into account during model evaluation.

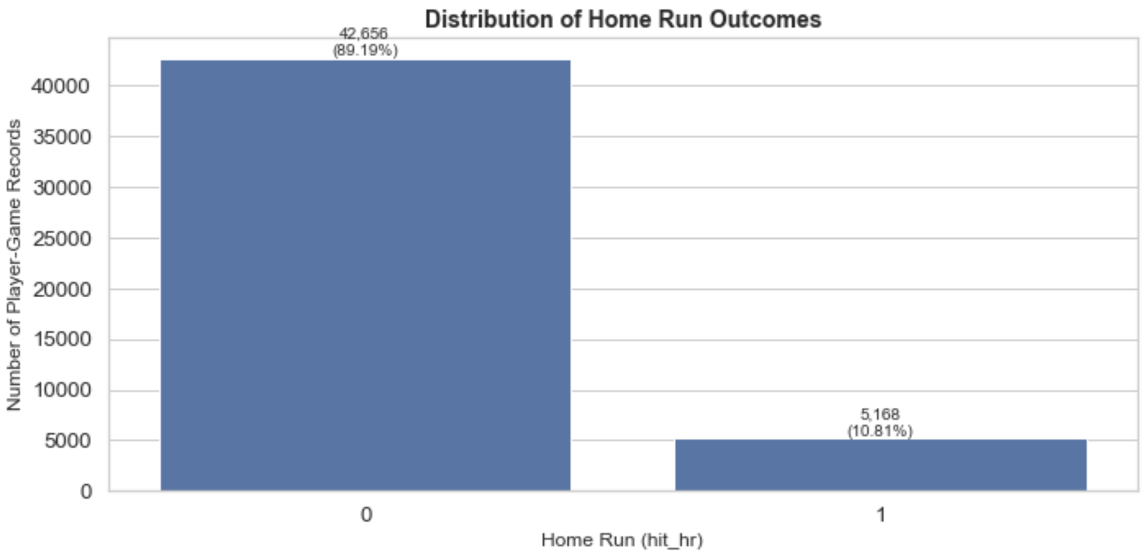

Figure 2: *Response class imbalance 89.19% No Home Run, 10.8% Yes Home Run.*

To explore the relationships among variables, I calculated pairwise correlations for the numeric features. A pairwise correlation matrix (Figure 3) was generated to examine linear relationships among all numeric features. You can see the interesting boxes that relate certain features. Several strong positive correlations were observed within logical feature groups; for example, among hitter performance metrics such as total bases, hits, and RBIs, and among pitcher statistics such as innings pitched, strikeouts, and earned runs. These internal clusters indicate consistency within related categories rather than redundancy across unrelated features. Prop-related variables (e.g., *to_record_a_run,to_record_2_or_more_hits*) also exhibited moderate correlations with underlying hitting performance measures, reflecting that sportsbook odds are influenced by recent player production. Weather and ballpark factors, by contrast, showed minimal correlation with player statistics, suggesting they contribute largely independent information to the model.

Overall, the correlation structure confirms that the dataset captures coherent relationships within each performance domain (hitting, pitching, props) without excessive multicollinearity across domains, supporting its suitability for predictive modeling.

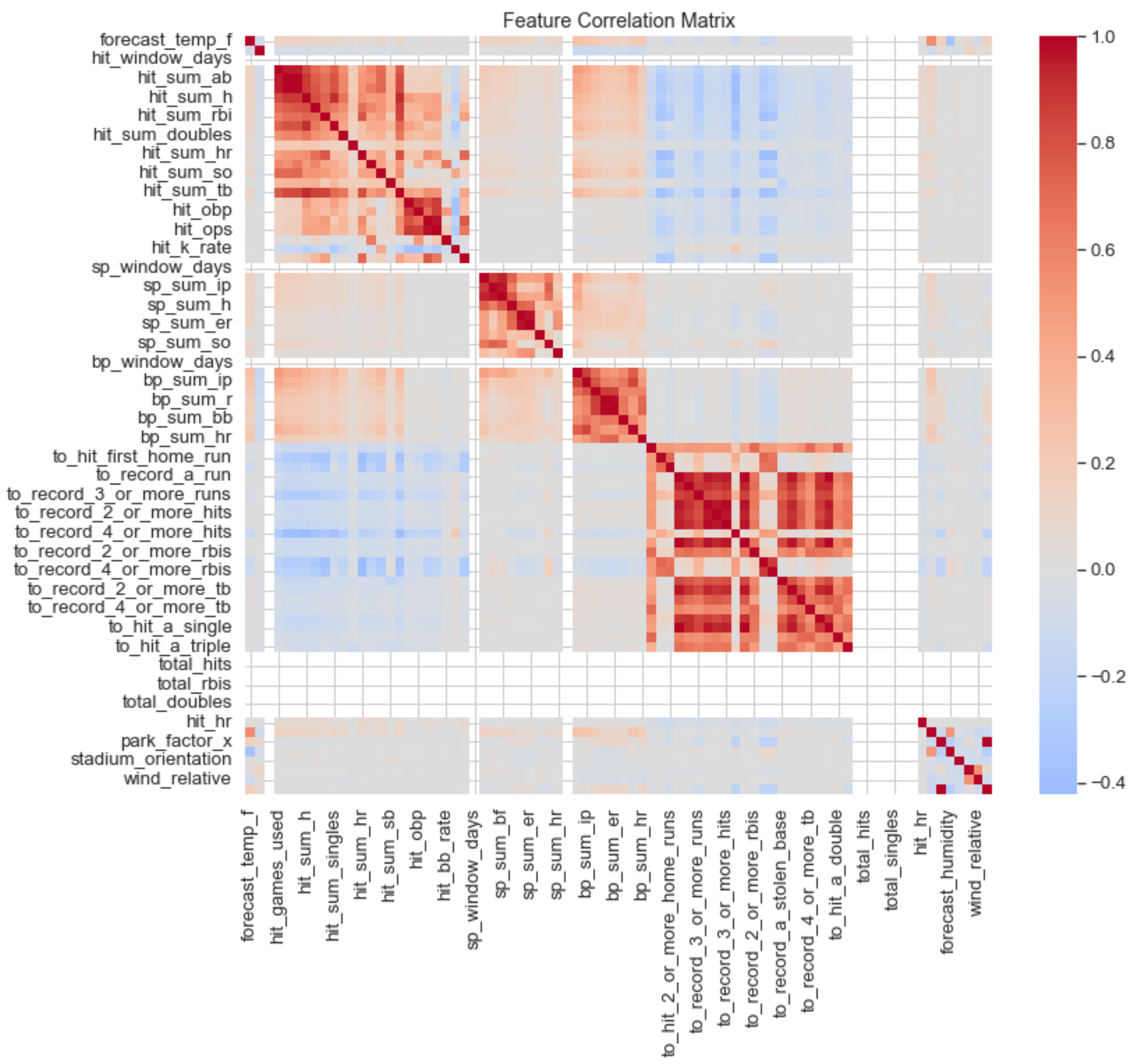

Figure 3: *Pairwise correlation matrix of all numeric features, illustrating relationships among hitting, pitching, and prop-based variables.*

Figure 4 shows the correlation between each feature and the response variable, *hit_hr*. None of the predictors show particularly strong correlation with the target, indicating that no single variable dominates the relationship. The most positively correlated features were *hit_sum_hr, hit_sum_tb*, and *hit_slg*, which makes intuitive sense since recent home run and power production should increase the likelihood of another. The most negatively correlated features included several prop-related variables such as *to_hit_2_or_more_home_runs* and *to_record_4_or_more_tb*, reflecting that players who have long odds in these props will be less likely to achieve them. Overall, these modest correlations highlight that home run prediction is a complex task influenced by a range of interacting factors, consistent with the rarity and variability of the event itself.

With the large number of features and the high correlation between related features feature reduction was applied, only keeping one of the highly correlated variables between variables correlated at least .9.

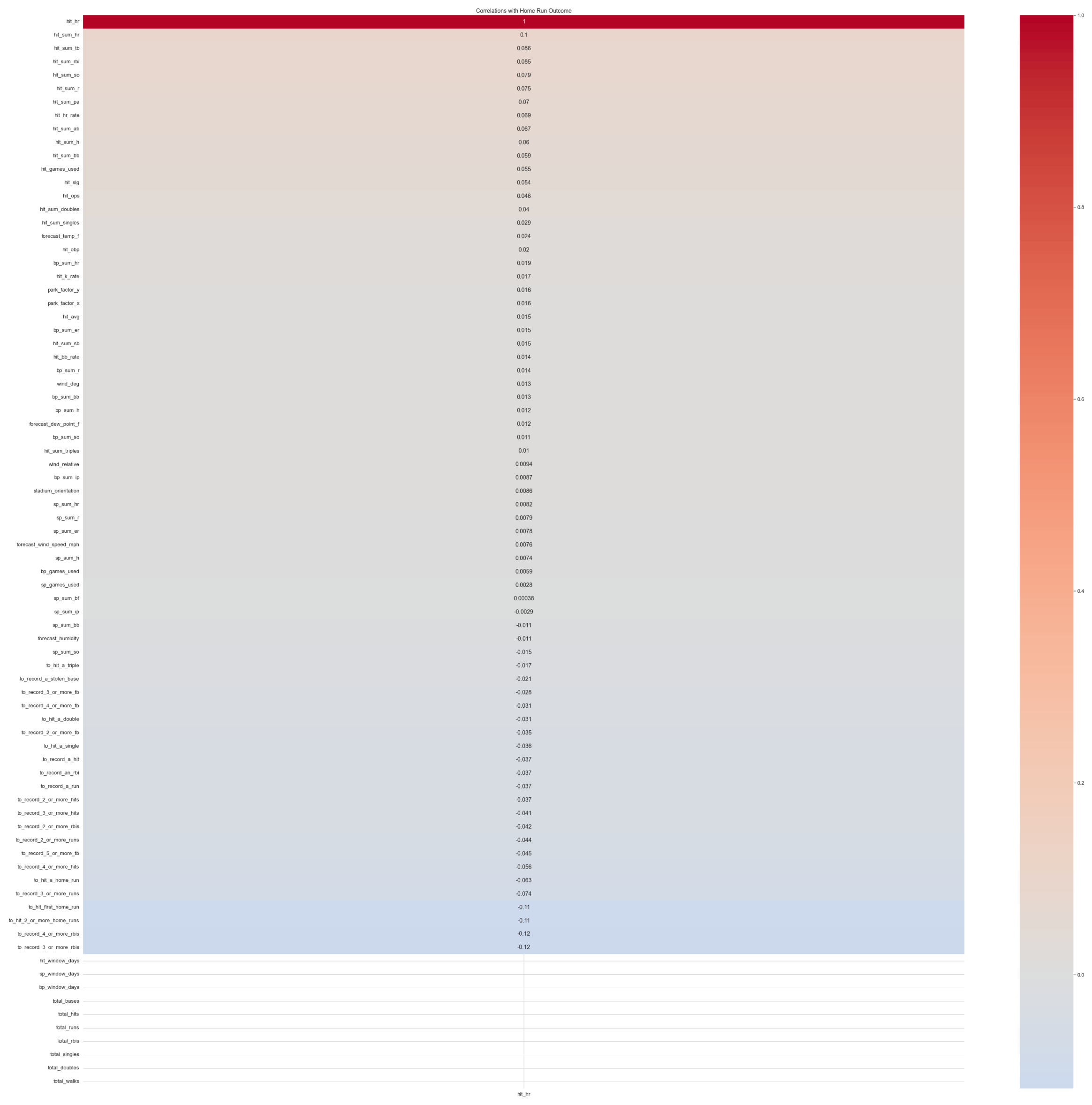

Figure 4: *Features correlation with the response (hit_hr).*

Distributions of key environmental and contextual features were visualized to assess skewness and variation. The weather-related variables exhibited realistic seasonal diversity, with *forecast_temp_f* roughly centered around 75°F and ranging from near 40°F to over 110°F, and *forecast_humidity* spanning a wide range from about 10% to nearly 100%. *Forecast_wind_speed_mph* and *forecast_dew_point_f* both displayed right-skewed distributions, indicating that most games were played under moderate wind and dew point conditions. The *park_factor* values were tightly clustered near 100, suggesting that most stadiums play close to league average in terms of offensive environment. Meanwhile, the distribution of *to_hit_a_home_run* odds was highly right-skewed, reflecting that the majority of players have long odds to hit a home run in a given game, while only a few top sluggers receive favorable prices.

Overall, the exploratory analysis confirmed that the dataset captured meaningful variation across both performance and contextual dimensions. The relationships observed aligned with baseball intuition stronger hitters and favorable weather conditions were associated with higher home run probabilities.

Feature Engineering

Before modeling, feature engineering was performed to add predictive information and reduce redundancy within the dataset. This included creating derived variables, such as hitter and pitcher home-run rates, weather-adjusted power indicators, and matchup interaction terms summarized in Table 1.

After generating these features, pairwise correlations were computed to identify variables that gave essentially the same information. Any feature with a correlation coefficient of 0.9 or higher with another feature was removed, leaving only one variable from each highly correlated pair. This step was performed for two main reasons.

First, removing redundant features simplifies the modeling space, which reduces noise, decreases the computational complexity of the algorithms, and improves overall model interpretability. Second, highly correlated features tend to tell the same statistical story, meaning that including both does not add meaningful information but can instead introduce issues such as multicollinearity. Multicollinearity can inflate variance in model coefficients, obscure feature importance rankings, and make it harder for certain algorithms to perform well. By retaining only one variable from each correlated pair, the final dataset retained predictive power while avoiding unnecessary duplication and potential modeling instability.

Feature Name	Description	Formula
hit_hr_rate_pa	HR rate per plate appearance.	HR / PA
hit_hr_rate_ab	HR rate per at bat.	HR /AB
hit_iso	Isolated power	SLG – AVG
hit_baip	Average on batted balls in play	(H − HR) / (AB − HR − SO + SB)
hit_xbh_rate	Extra base hit rate	(2B + 3B + HR) / AB
hit_bb_k_ratio	Walk to strikeout ratio	BB / SO
days_since_last_game	Days since last game played	Current Date – Last Game Date

hit_hot_short	Short term HR trend per game	HR window total / window days
hit_hot_pa	Short term HR per plate appearance	HR window total / PA window total
sp_hr9	SP Homeruns per 9 innings	HR / (IP / 9)
sp_bb9	SP walks per 9 innings	BB / (IP / 9)
sp_so9	SP Strikeouts per 9 innings	K / (IP / 9)
sp_HR_per_bf	SP HR per batter faced	HR / BF
bp_hr9	BP Homeruns per 9 innings	HR / (IP / 9)
bp_bb9	BP walks per 9 innings	BB / (IP /9)
bp_so9	BP SO per 9 innings	SO / (IP /9)
wind_out/in/left/right	Encoded wind direction	1 if wind blowing in that direction from home plate
temp_boost	Heat driven HR carry adjustment	max(0, temp-70) / 10
dew_boost	Humidity driven HR carry adjustment	max(0, dew_point – 60) / 10
matchup_power	hitter power * pitcher HR-proneness	ISO * SP_HR9
matchup_contact	hitter HR rate * SP walk rate	HR_rate_PA * (SP_BB9 + 1)
hot_weather_boost	recent HR streak * favorable weather	hot_short * (wind_out + temp_boost)

***Table 1**: Derived features with descriptions and formulas.*

After removing highly correlated variables a random forest was fit on the remaining features and their importance to the model was plotted. This gave the top 75 variables that the random forest found useful for predicting home runs. They can be found in Figure 5 below.

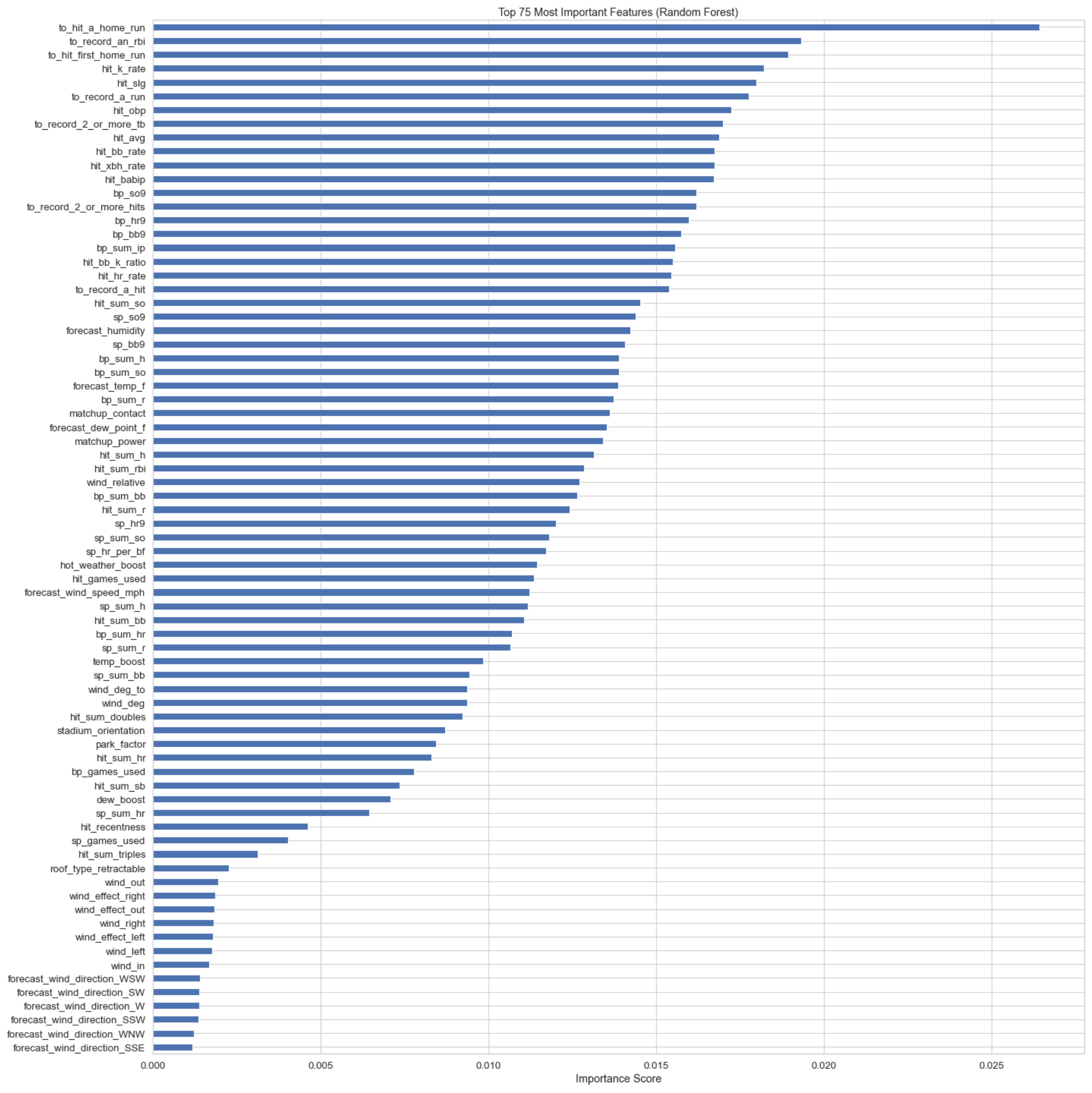

Figure 5: *Random Forest Variable Importance Plot top 75 features.*

The Random Forest model produced a ranked list of feature importances (Figure 5), providing insight into which variables contributed most to predicting daily home-run outcomes. To no surprise sportsbook-derived variables overwhelmingly dominated the top of the importance distribution. In particular, the primary bookmaker "HR yes/no" odds feature carried substantially more importance than any other individual predictor. This result is both expected and informative: sportsbook odds are generated using real-time information, expert modeling, injury data, park effects, and betting market behavior. Because sportsbooks fit a wide range of contextual information into a single probability, it is logical that these odds serve as an extremely strong predictor in the model.

Below the odds features, the importance scores show a natural and intuitive blend of baseball-driven variables, including recent batter performance metrics (e.g., HR rates, SLG, OPS, BB/K ratio), pitcher home-run susceptibility (HR/9, HR per batter faced), bullpen HR rates, and key weather factors (temperature, dew point, wind direction, and humidity). This mix of categories aligns closely with domain knowledge: a hitter's current form, the opposing pitcher's ability to limit home runs, and the environmental conditions of the ballpark all impact the probability of hitting a home run. It is notable that the Random Forest independently identified the same types of factors that one would logically assemble when analyzing baseball home runs, suggesting that the model is using features in a way that is consistent with baseball knowledge.

Another important observation from Figure 5 is the drop-off in importance after approximately the 50th feature. The top 50 features account for the vast majority of the model's explanatory power, while features ranked outside the top 50 contribute very marginal gains. This creates a natural cutoff point for feature selection: limiting the model to the top 50 predictors simplifies the modeling pipeline, reduces noise, and helps guard against overfitting, while still preserving nearly all of the predictive value captured by the full feature set.

Overall, the variable-importance distribution reinforces that home-run hitting is influenced by a combination of sportsbook expectations, batter power and form, pitcher quality, bullpen tendencies, and environmental conditions. The fact that the model's learned hierarchy of feature importance closely mirrors baseball logic provides another level of validation that the feature engineering process captured the right types of information for modeling home runs.

Model Development & Evaluation

Before developing and evaluating predictive models, it was important to choose a validation strategy that perfectly reflects how the model would be used in real life. A traditional random train–test split was not appropriate for this problem because the data is time-dependent. Using a standard split would risk mixing future games into the training data for earlier games, creating data leakage and possibly inflating performance. Additionally, many of the engineered features such as rolling averages and recent HR rates explicitly rely on the time ordering of games. Any break in that ordering compromises their meaning and compromises any results obtained.

Because of these constraints, it was necessary to bypass the typical "baseline model with a random split" workflow and instead go straight into evaluating and tuning models within a time-aware framework that prevents leakage. This ensured that every model was trained only on information that would have been available at the time of prediction, resulting in a more realistic and valid assessment of performance. Moving directly into a time-consistent tuning approach was therefore the most reliable choice for this project.

Model Selection

The models selected for this project were Categorical Boost (CatBoost), XGBoost, Random Forest, LightGBM, and a Multi-Layer Perceptron Neural Network. They were chosen to capture a range of modeling types, including boosting, bagging, and neural architectures, each of which offers different strengths for modeling nonlinear interactions, categorical and numeric features, and noisy real-world baseball data.

CatBoost was included for its powerful gradient-boosting framework and native handling of categorical variables. Its ordered boosting procedure also reduces target leakage, making it perfect for time-dependent, walk-forward prediction.

XGBoost was selected for its strong performance on tabular data and built-in regularization. It is widely used in industry due to its flexibility and interpretability through feature importance metrics, though it requires careful tuning to avoid overfitting.

Random Forest was chosen as a baseline ensemble model due to its generalization stability, ability to handle high-dimensional data, and clear feature importance signals. Its drawbacks include reduced interpretability and higher computational cost relative to simpler models.

LightGBM was selected for its speed and scalability. Its histogram-based splits and leaf-wise tree growth allow it to train efficiently while capturing complex patterns, though these same characteristics can increase the risk of overfitting if not tuned carefully.

Finally, a neural network (MLP) was included to provide a flexible nonlinear baseline capable of modeling complex decision boundaries. While powerful, neural networks require careful regularization, scaling, and hyperparameter tuning, and can be less interpretable than tree-based models.

Model Tuning Strategy

Log loss corresponds to the logarithmic scoring rule, which is a strictly proper scoring rule, meaning it incentivizes models to output well-calibrated probability estimates rather than just accurate classifications. Strictly proper scoring rules are recognized as the correct objective for probabilistic forecasting because they reward honest, calibrated assessments and heavily penalize overconfident incorrect predictions (Gneiting & Raftery, 2007). This makes log loss more suitable than metrics such as accuracy or AUC in a rare-event, probability-forecasting problem such as home-run prediction and is why it was chosen as the tuning metric for all models.

Hyperparameter search was performed using RandomizedSearchCV, which samples from the parameter space rather than exhaustively evaluating every possible combination like a grid search. This approach dramatically reduces computation time while still exploring a wide range of candidate configurations.

To avoid data leakage and preserve the temporal structure of the season, each tuning run used a TimeSeriesSplit (n_splits = 5) cross-validation scheme. Unlike standard k-fold cross-validation, TimeSeriesSplit ensures that each fold trains only on past data and evaluates on future data, closely shadowing how predictions would be made in practice. This framework provided a realistic, leak-free way to tune all models while respecting the chronological nature of the home run prediction problem.

Tuning CatBoost

CatBoost was tuned using a time-aware hyperparameter search designed to optimize probability calibration for daily home-run prediction. A RandomizedSearchCV combined with a 5-fold TimeSeriesSplit ensured that all tuning respected the chronological structure of the season and avoided data leakage. The search explored a wide range of learning rates, tree depths, regularization strengths, and boosting parameters, ultimately selecting a configuration that reflects a slow-learning but highly regularized model. The best model used a very low learning rate (0.01), a large number of boosting iterations (800), medium tree depth (6), and strong L2 regularization (30), along with increased sampling randomness through a high bagging temperature (5). This final set of hyperparameters shows that CatBoost benefited from small, stable learning and a high degree of regularization, which makes sense for the noisy and imbalanced nature of home-run prediction. Figure 6 below shows the tuned models metrics vs the base model where we can see the tuned model has a slightly better log loss (we are trying to minimize it) along with higher Precision than the base model.

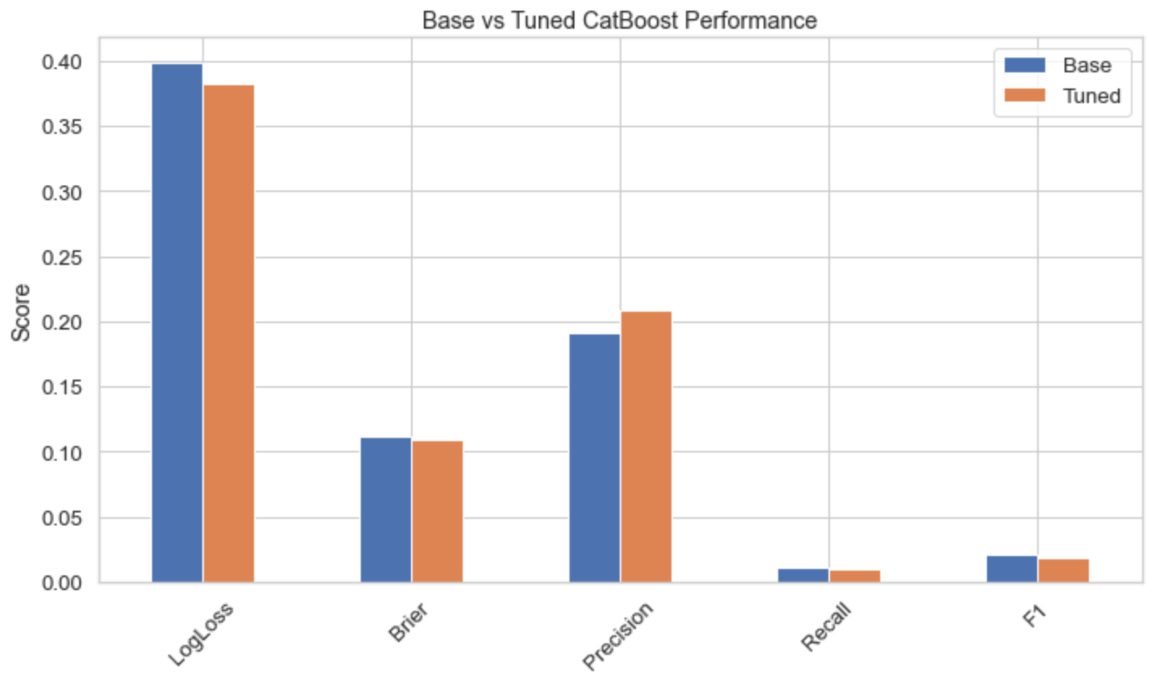

Figure 6: *CatBoost Tuned vs Base CatBoost Model metrics.*

Tuning XGBoost

XGBoost was tuned using the same time-aware procedure described in the CatBoost section. Within that framework, the tuning process explored a wide range of learning rates, tree depths, sampling fractions, and regularization strengths. The best-performing model favored a very low learning rate (0.01) paired with 400 boosting iterations and shallow trees (max_depth = 3), indicating that XGBoost benefited from gradual learning and strong constraints on model complexity. Additional parameters such as min_child_weight = 5, subsample = 0.9, colsample_bytree = 0.6, and both L1 and L2 regularization set to 1.0 further show a highly regularized model structure. Overall, the tuned XGBoost model settled on a configuration that emphasizes stability and generalization, which is consistent with the noisy and imbalanced nature of predicting daily home-run outcomes and also with the tuned CatBoost model’s structure.

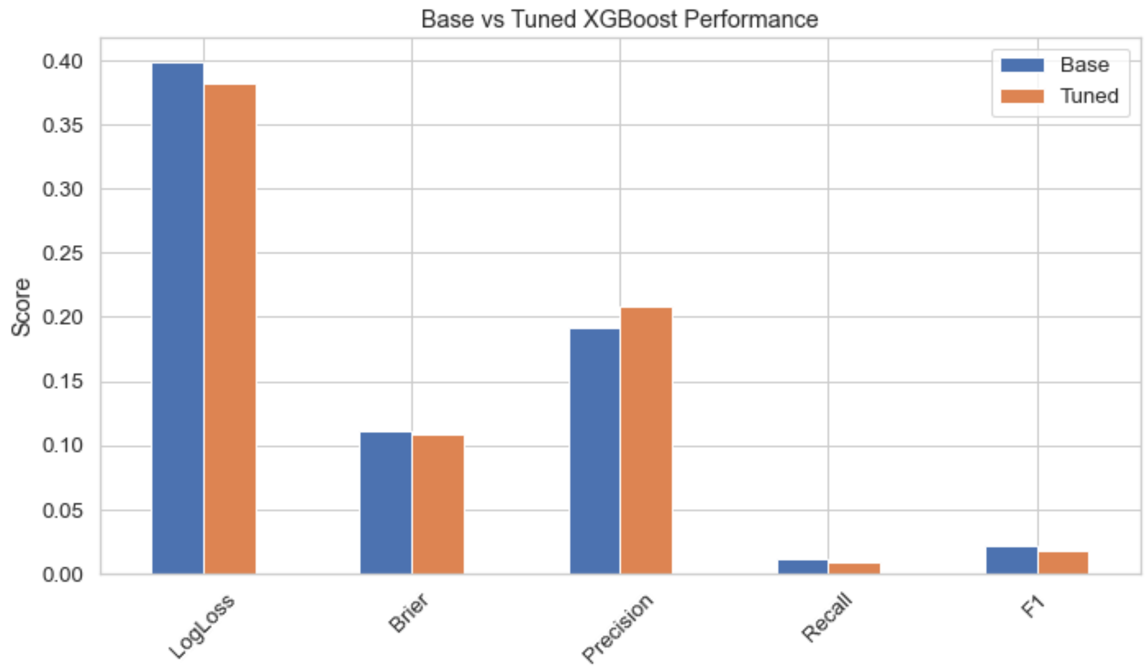

Figure 7: *XGBoost Tuned vs Base XGBoost Model metrics.*

Tuning Random Forest

Random Forest was tuned using the same time-aware RandomizedSearchCV approach described earlier. The model was configured with class_weight="balanced" to address the extreme class imbalance inherent in daily home-run prediction. The tuning process explored a wide range of tree counts, depths, sampling strategies, and minimum sample thresholds. The best-performing configuration consisted of 500 trees, no maximum depth constraint, min_samples_split = 10, min_samples_leaf = 2, and max_features = 'log2', with bootstrap disabled. This combination reflects a model that relies on deep, fully expanded trees but limits overfitting through split thresholds and strong feature subsampling. Using 'log2' as the feature selection method forces each split to consider only a small subset of predictors, which increases diversity among trees, especially when paired with bootstrap=False. Overall, the tuned Random Forest favors a high-capacity ensemble with controlled splitting behavior and aggressive feature randomness and as shown in Figure 8, actually has a slightly less optimal log loss than the base model.

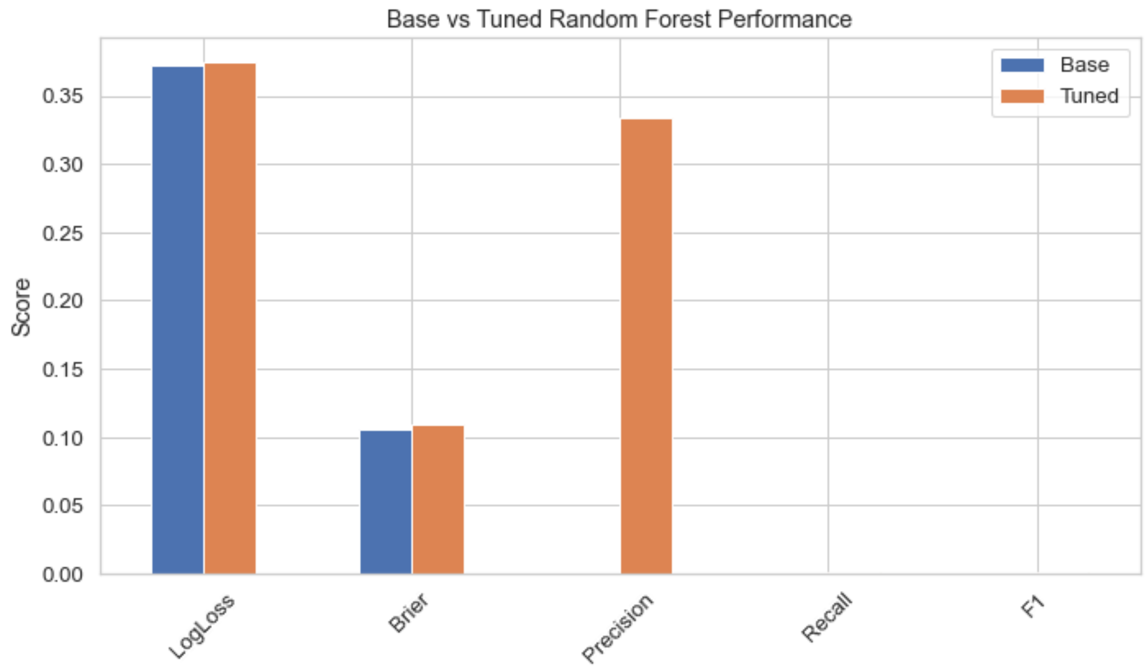

Figure 8: *Random Forest base vs Random Forest Tuned metrics.*

Tune LightGBM

LightGBM was tuned using the same time-aware RandomizedSearchCV procedure described earlier. The search explored a wide range of learning rates, leaf counts, sampling ratios, depth constraints, and regularization strengths. The best-performing configuration selected a low learning rate (0.01) with 300 boosting iterations, paired with num_leaves = 15 and max_depth = 7, indicating that LightGBM learned most effectively when its trees were relatively shallow and tightly constrained. Regularization parameters also contributed meaningfully: reg_alpha = 0.1 and reg_lambda = 0 produced the lowest log loss. The algorithm further favored subsample = 0.7 and colsample_bytree = 1.0, balancing row-level randomness with full feature availability at each split. Overall, the tuned LightGBM model reflects a strongly regularized, conservative boosting strategy, very similar to the other tuned models.

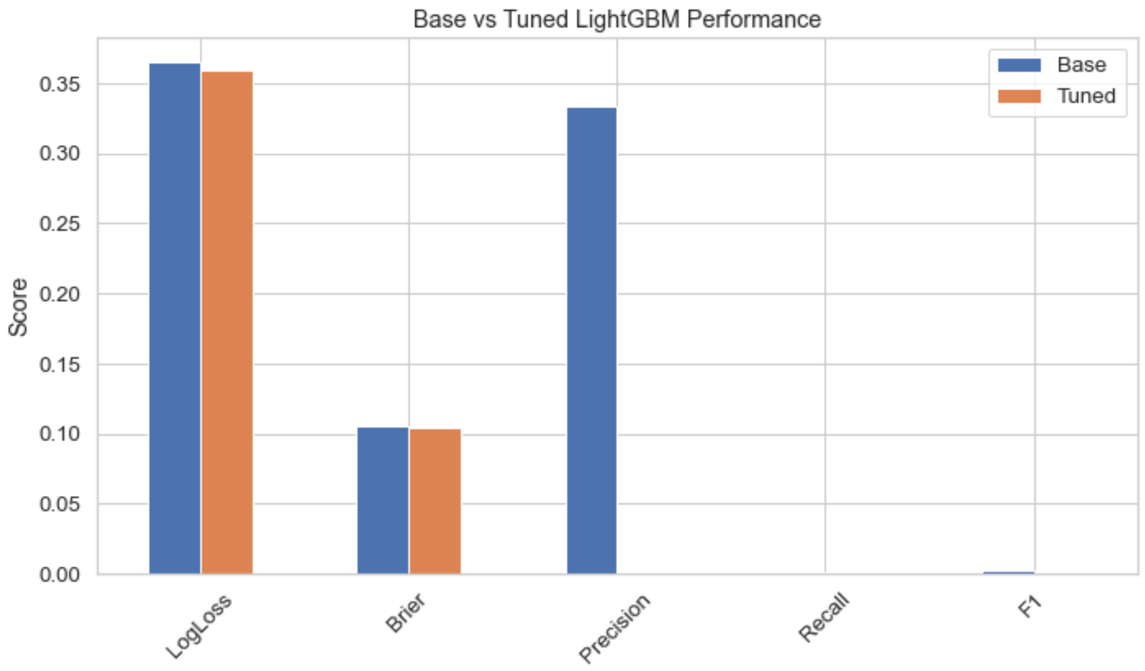

Figure 9: *LightGBM base vs LightGBM tuned metrics.*

Tuning Neural Network

The neural network (MLPClassifier) was tuned using the same time-aware RandomizedSearchCV framework described earlier. The search explored a variety of network depths, activation functions, learning rates, and L2 penalties. The best-performing model selected a three-layer architecture with 128, 64, and 32 neurons, using the tanh activation function, a low learning rate (0.0005), batch size of 64, and alpha = 0.001 for L2 regularization. This configuration suggests that the model benefited from a moderately deep architecture capable of capturing non-linear interactions, but required strong regularization and a small learning rate to remain stable given the noise and class imbalance inherent in daily home-run prediction.

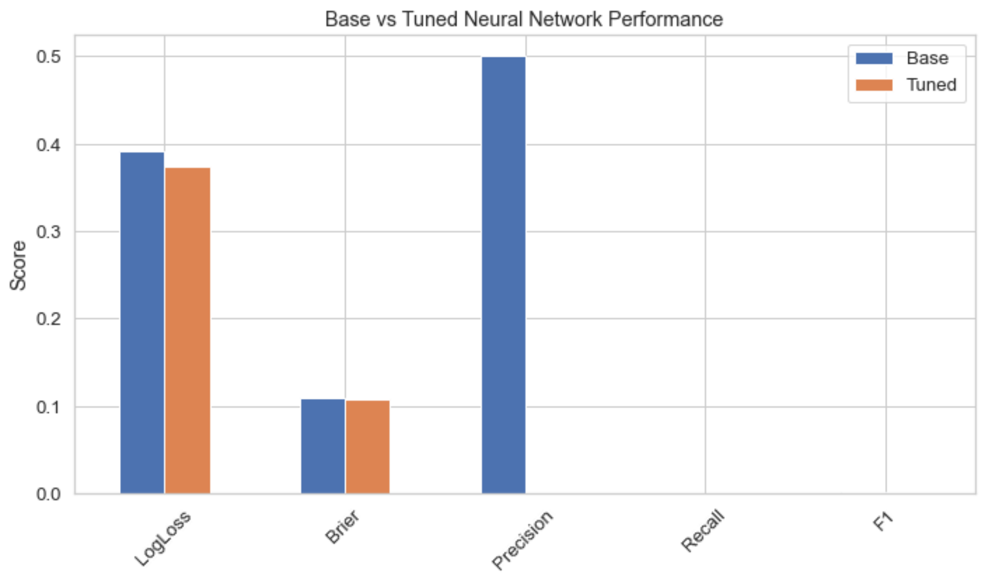

Figure 10: *Neural Network base vs Neural Network tuned metrics.*

Tuned Models Summary

Across all models, an obvious pattern emerged: the tuning process overwhelmingly favored highly regularized models. Tree-based models such as XGBoost and LightGBM, both chose shallow tree depths, smaller leaf sizes, and strong sampling or regularization constraints, indicating that the home-run prediction problem benefits from models that generalize cautiously. XGBoost and LightGBM, in particular, selected very low learning rates and conservative tree structures, reflecting the need for slow updates in a noisy and highly imbalanced problem. The Random Forest model represented a deviation from this pattern. Although it adopted moderate split constraints and aggressive feature subsampling (max_features='log2'), it also allowed trees to grow without a maximum depth limit, showing that: deeper trees may still capture beneficial interactions and not overfit as long as other forms of regularization are present. The neural network followed a similar trend toward caution, selecting a low learning rate and meaningful L2 regularization, despite employing a relatively deep three-layer architecture with 128, 64, and 32 neurons. Overall, the tuning results suggest that the underlying signal in daily home-run prediction is subtle and more often than not overshadowed by noise, making regularization crucial for most models.

Walk Forward Back Testing

To evaluate model performance in a way that accurately reflects real-world usage, a walk-forward back-testing framework was implemented. Unlike traditional evaluation methods, which rely on random or static train–test splits, the walk-forward approach keeps the chronological order of the MLB season. This prevents the model from ever training on information that would not have been available at the time of prediction, a form of data leakage that would undermine the validity of this project.

The walk-forward engine operated on a one-calendar-day-at-a-time basis. For each game date, the model was trained exclusively on all prior days (< game date) and then evaluated on the games occurring on that specific day. This process was repeated for every date in the dataset, resulting in a full season of sequential, leak-free predictions. For each day, the following information was generated and saved:

- Predicted probability that each player would hit a home run
- Actual outcome (hit HR vs. did not)
- Sportsbook odds for each player
- Implied probability derived from the odds
- Player Name and ID
- Model name

After generating raw model predictions, an additional step was performed: probability calibration using scikit-learn's CalibratedClassifierCV. Because tree-based models and neural networks often produce poorly calibrated probabilities, typically being overconfident or too conservative in rare-event settings such as home runs, each model's daily predictions were calibrated using an isotonic regression method. Calibration adjusts the predicted probabilities so that, over many predictions, a forecasted probability of *p* corresponds more closely to an actual event frequency of *p*. This step is crucial for this project because betting strategies rely directly on probability estimates. Any miscalibration, such as overestimating or underestimating home-run probabilities, would affect the computed betting edge and lead to misleading or unprofitable ROI results. The importance of probability calibration in decision-making systems is well-documented (Niculescu-Mizil & Caruana, 2005).

These daily prediction snapshots were added to a running dataset containing the model's forecasts for the entire season. Saving predictions at this level of granularity is crucial because it enables a wide range of simulations and analyses that can be done much faster than retraining and testing. For example, we can test any betting strategy across models in seconds with the saved predictions.

Because each day's predictions were produced strictly using past information, this procedure offers the closest possible approximation to real-world betting and forecasting conditions. This allows for the question of: "Is this model really profitable?" to be answered with confidence, as these are the exact conditions and data that the models would have seen, so the predictions and results can be used to gauge if the models are profitable or not.

Walk-forward evaluation is therefore the most reliable approach for this problem. It respects the temporal nature of baseball data, prevents leakage from future outcomes, and provides a realistic foundation for evaluating model performance in terms that matter for this problem.

Betting Simulations: True Edge vs Expected Value

To translate model predictions into actionable performance metrics, two betting simulation frameworks were implemented: True Edge-based betting and Expected Value (EV)- based betting. Both approaches leverage the model's calibrated probabilities and the sportsbook's implied probabilities, but each evaluates a different aspect of predictive quality and profitability. By examining model behavior under both strategies, it becomes possible to assess not only how accurate the model is, but also whether it generates a consistent and profitable betting edge.

True Edge Betting

True Edge betting evaluates the difference between the model's predicted probability and the sportsbook's implied probability:

$$True\ Edge = p^{\wedge}_{model} - p_{implied}$$

Where $p^{\wedge}_{model}$ is equal to the model's output probability for the player to hit a home run and $p_{implied}$ is the sports books implied probability for that player to hit a home run. The implied probability was calculated by dividing 1 by the sportsbook's odds or $\frac{1}{odds}$. This essentially tells us when our models think that a player has a better chance of hitting a home run than the sportsbook does. Because True Edge isolates the model's predictive advantage independently from payout structure, it is a direct measure of informational superiority over the market.

Expected Value Betting

The Expected Value (EV) ratio compares the model's predicted probability of a home run to the sportsbook's implied probability, expressed as**:**

$$EV\ Ratio = \frac{p^{\wedge}_{model}}{p_{implied}}$$

An EV ratio greater than 1.0 indicates that the model believes the bet has a positive expected return after taking the payout structure into account. This differs from True Edge, which measures only the raw difference between the model's probability and the sportsbook's probability. While True Edge focuses strictly on whether the model thinks the sportsbook is mispricing the outcome, the EV ratio incorporates both probability and payout, making it a more economic metric, which is essential for maximizing return on investment or ROI.

Betting Simulations

For both betting strategies, simulations were run across the entire 2025 season using a range of thresholds for each model. Whenever a prediction exceeded the threshold defined by either the True Edge or EV ratio strategy, a fixed $10 wager was placed on that player for that game. Winning bets added the appropriate profit based on the posted odds, while losing bets deducted the $10 stake. Throughout each simulation, cumulative profit, total number of bets, and return on investment (ROI) were tracked, and at the end of the season, these metrics were recorded for every model threshold combination. This framework provides a realistic assessment of how profitable each model would have been if deployed in real time during the 2025 MLB season.

Beyond full-season profitability, additional tools were developed to examine model performance across specific time windows and for individual players. These tools made it possible to determine whether certain models performed better during particular stretches of the season or whether sportsbooks consistently mispriced some players. Furthermore, the simulation framework also recorded whether a player had hit a home run in the previous game, allowing for an analysis of whether the models placed higher probabilities on players coming off a home run the previous game. This helps evaluate whether the models captured streakiness effects.

CatBoost Results

Table 2 below summarizes the CatBoost results for the True Edge simulation on the 2025 MLB season.

Threshold	Bets	Total Profit	ROI
0.01	2398	-4295.4	-17.91%
0.02	1288	-3563.0	-27.66%
0.03	788	-1916.0	-24.31%
0.05	372	-457.3	-12.29%
0.08	189	-200.3	-10.60%
0.10	144	110.7	7.69%
0.15	96	242.9	25.30%
0.2	61	158.2	25.93%
0.3	16	35.1	21.94%
0.4	6	0.9	1.50%
0.5	1	-10	-100%

Table 2: *True Edge betting simulation results for the CatBoost model, 2025 MLB season.*

The True Edge simulation results show a consistent pattern; CatBoost was unprofitable at all lower edge thresholds, but performance improved substantially as the edge increased. Thresholds from 0.01 to 0.08 produced losses (-10% to -27%), showing that small differences between the model's predicted probabilities and the sportsbook's implied probabilities were not profitable. Profitability began at a threshold of 0.10 (7.69% ROI), and the strongest returns occurred at thresholds of 0.15 (25.30% ROI) and 0.20 (25.93% ROI). However, these higher thresholds produced much fewer bets; 96 bets at 0.15 and 61 bets at 0.20 for the entire 2025 season, showing that large edges were profitable but were also relatively rare.

While the profitability at higher thresholds is encouraging, the reduced number of bets also introduces greater variance in the results. As the threshold increases, each individual outcome carries more weight, making ROI more sensitive to short-term fluctuations and luck. This is obvious at extreme thresholds such as 0.30–0.50, where only a handful of bets were placed, leading to unstable ROI outcomes (including a −100% result on a single bet at the 0.50 threshold). Overall, CatBoost demonstrated that it could identify high-confidence mispricing, but these opportunities were infrequent, and the reliability of returns decreased as sample size declined.

Table 3 below shows the results for the Expected Value betting simulation for the 2025 MLB season.

Threshold	Bets	Total Profit	ROI
1.05	3021	-5264.0	-17.42%
1.10	1930	-3992.0	-20.68%
1.20	958	-2805.3	-29.28%
1.30	601	-2019.3	-33.6%
1.40	403	-1371.5	-34.03%
1.50	286	-654.3	-22.88%
2.0	88	-1.9	-0.22%
2.5	27	31.5	11.67%
3.0	9	-12.0	-13.33%

Table 3: *Expected Value betting for the CatBoost model for the full 2025 season.*

The Expected Value simulations for CatBoost show that the model was consistently unprofitable across all lower EV ratio thresholds, with ROI values worsening as the threshold increased from 1.05 to 1.40. This suggests that CatBoost's modest probability advantages almost never translated into wagers strong enough to overcome the payout structure embedded in the odds. Profitability did not appear until very high EV ratios, where the number of bets taken dropped: the model achieved a positive ROI only at the 2.5 threshold (11.67% ROI on 27 bets). As with the True Edge strategy, the profitability at higher thresholds is accompanied by substantially smaller sample sizes, making results more volatile. Overall, CatBoost struggled to produce sustained positive EV opportunities, and meaningful profitability emerged only at rare, high-EV conditions, and the True Edge betting produced better, more profitable results.

XGBoost Results

Table 4 below summarizes the full-season ROI results for the XGBoost model under the True Edge betting simulation.

Threshold	Bets	Total Profit	ROI
0.01	5815	-19369.5	-33.31%
0.02	2885	-11058.5	-38.33%
0.03	1221	-4416.5	-36.17%
0.05	205	-1094.5	-53.39%
0.08	39	-136.5	-35.00%
0.10	31	-136.5	-44.03%
0.15	22	-78.0	-35.45%
0.2	17	-28.0	-16.47%
0.3	3	-30.0	-100%
0.4	0	0	0%
0.5	0	0	0%

Table 4: *XGBoost True Edge betting simulation results for the 2025 MLB season.*

The True Edge simulation results show that XGBoost was not profitable at any threshold across the entire season. Although the model generated far more betting opportunities than CatBoost (over 5,800 bets at the 0.01 threshold), this higher volume translated into larger cumulative losses rather than improved performance. Losses persisted across all edge levels, from mild thresholds (−33% ROI at 0.01) to more conservative ones (−35% to −53% ROI in the 0.05–0.10 range). Even at very high thresholds, where only a handful of bets were placed, the model remained unprofitable, including a −100% result at the 0.30 threshold. These outcomes suggest that while XGBoost was more aggressive in identifying supposed edges, the majority of these signals were not actionable, and the model failed to produce profitable value against the sportsbook under the True Edge betting framework.

Threshold	Bets	Total Profit	ROI
1.05	7696	-24512.2	-31.85%
1.10	5883	-20126.5	-34.21%
1.20	3366	-13074.5	-38.84%
1.30	1825	-7090.5	-38.85%
1.40	1052	-4162.0	-39.56%
1.50	605	-2590.0	-42.81%
2.0	64	-318.0	-49.69%
2.5	14	2	1.43%
3.0	5	28	56.00%

Table 5: *XGBoost Expected Value betting simulation results for the 2025 MLB season.*

The Expected Value simulation results show that XGBoost was consistently unprofitable across nearly all EV ratio thresholds. Even at modest thresholds such as 1.05 to 1.40, where the model generated between 1,000 and 7,600 wagers, ROI values remained sharply negative, ranging from −31% to −40%. As the thresholds increased, the number of qualifying bets fell drastically. However, profitability did not improve: the model still produced losses at EV ratios of 2.0 and 2.5, achieving only a minimal profit at 2.5 (1.43% ROI on 14 bets). A single profitable outlier appeared at the 3.0 threshold (56% ROI), but this was based on only five wagers and therefore reflects high variance rather than meaningful predictive power. Overall, XGBoost failed to produce reliable positive EV signals, and the rare high-threshold wins occurred on sample sizes too small to be considered actionable. Overall, XGBoost produced far more bets than CatBoost, but these bets were unprofitable and led to more losses than gains.

Random Forest Results

Table 6 below presents the True Edge betting simulation results for the Random Forest model during the 2025 MLB season.

Threshold	Bets	Total Profit	ROI
0.01	3158	-8564.8	-27.12%
0.02	1621	-5172.8	-31.91%
0.03	782	-2642.8	-33.80%
0.05	248	-1207.5	-48.69%
0.08	81	-214.5	-26.48%
0.10	54	-166.5	-30.83%
0.15	28	-37.5	-13.39%
0.2	17	-4.0	-2.35%
0.3	3	72	240.00%
0.4	0	0	0%
0.5	0	0	0%

Table 6: *Random Forest True Edge betting simulation results for the 2025 MLB season.*

The True Edge simulation results show that Random Forest, like the other ensemble methods, was unprofitable across all lower and moderate edge thresholds. ROIs ranged from −27% to −49% for thresholds between 0.01 and 0.05, indicating that the model's smaller edge signals were too noisy to exploit. While losses decreased at higher thresholds, profitability did not emerge until the 0.30 threshold, where Random Forest produced a large positive ROI (240%). Still, this result was driven by only three bets, making it highly unstable and not indicative of consistent predictive value. Similar to all the other models, the model generated fewer high-edge opportunities as the threshold increased, leading to higher variance and less reliable performance metrics. Overall, Random Forest failed to produce actionable True Edge signals across the full season, with isolated high-threshold gains driven by extremely small sample sizes rather than systematic model strength.

Table 8 below shows the Expected Value Ratio betting simulation results for the Random Forest model during the 2025 MLB season.

Threshold	Bets	Total Profit	ROI
1.05	4250	-9578.8	-22.54%
1.10	3007	-8086.8	-26.89%
1.20	1565	-5646.5	-36.08%
1.30	835	-2397.5	-28.71%
1.40	462	-2392.5	-51.79%
1.50	269	-1323.5	-49.20%
2.0	35	106.0	30.29%
2.5	5	116.0	232.00%
3.0	1	45.0	450.00%

Table 8: *Random Forest Expected Value betting simulation results for the 2025 MLB season.*

Under the Expected Value betting strategy, Random Forest showed persistent losses across all lower and moderate EV ratio thresholds. From 1.05 to 1.50, the model generated a large number of bets ranging from 4250 bets at 1.05 down to 269 bets at 1.50, yet all produced negative ROI values between −22% and −52%. This indicates that Random Forest's probability estimates did not translate into profitable opportunities when incorporating the sportsbook's payout structure. Positive returns did not appear until the highest thresholds, with ROI improving at EV ≥ 2.0, including isolated high-percentage gains at 2.5 and 3.0. However, these profitable outcomes were based on extremely small sample sizes (5 bets and one bet, respectively), making them highly volatile and not representative of consistent model performance. Overall, Random Forest struggled to identify reliable EV wagers, with meaningful profitability appearing only in rare, high-threshold scenarios dominated by variance rather than predictive strength.

LightGBM Results

Table 9 below shows the True Edge betting simulation results for the LightGBM model during the 2025 MLB season.

Threshold	Bets	Total Profit	ROI
0.01	1574	-2959.8	-18.80%
0.02	749	-1159.4	-15.48%
0.03	389	-513.8	-13.21%
0.05	143	138.7	9.70%
0.08	61	112.3	18.41%
0.10	50	130.3	26.06%
0.15	29	38.2	13.17%
0.2	20	4	2.00%
0.3	6	-60.0	-100.00%
0.4	0	0	0%

0.5	0	0	0%

Table 9: *LightGBM True Edge betting simulation results for the 2025 MLB season.*

The True Edge simulation results show that LightGBM performed noticeably better than the other tree-based models while also being significantly more selective in the wagers it chose to make. Across almost all thresholds, LightGBM placed fewer bets than CatBoost, XGBoost, or Random Forest, indicating that it identified fewer edges overall but focused more on higher-confidence opportunities. While the model remained unprofitable at very low thresholds (0.01–0.03), the losses were smaller than those of the other models, and LightGBM transitioned to profitability much earlier, achieving positive ROI at the 0.05 threshold (9.70%), with increasing returns at 0.08 (18.41%) and 0.10 (26.06%). This pattern suggests that LightGBM produced fewer but stronger edge signals. As thresholds rose, the already limited number of bets decreased further, leading to higher variance at extreme levels, for example, a −100% ROI on only six bets at the 0.30 threshold. Overall, LightGBM demonstrated the most disciplined and effective True Edge performance among the models, combining selectivity with meaningful profitability at several mid-range thresholds.

Table 10 below summarizes the Expected Value Ratio betting simulation results for the LightGBM model during the 2025 MLB season.

Threshold	Bets	Total Profit	ROI
1.05	2136	-4689.4	-21.95%
1.10	1405	-2542.8	-18.10%
1.20	667	-1474.7	-22.11%
1.30	368	-959.9	-26.08%
1.40	227	-568.5	-25.04%
1.50	152	-64.0	-4.21%
2.0	25	-116.0	-46.40%
2.5	12	14.0	11.67%
3.0	3	-30.0	-100.00%

Table 10: *LightGBM Expected Value betting simulation results for the 2025 MLB season.*

Under the Expected Value betting framework, LightGBM showed mixed performance, with consistent losses at lower EV thresholds but signs of slight improvement as the model became more selective. At EV ratios between 1.05 and 1.40, LightGBM placed fewer bets than most of the other algorithms and produced ROIs ranging from −18% to −26%, indicating that its moderate-confidence signals were still not strong enough to outperform sportsbook pricing. Performance improved at higher thresholds, with losses narrowing at 1.50 (−4.21% ROI) and profitability emerging only at the 2.5 threshold, where the model generated an 11.67% ROI on 12 bets. As with the True Edge strategy, bet volume dropped sharply at higher EV levels, leading to greater variance, shown by a −100% ROI on only three bets at the 3.0 threshold. Overall, LightGBM demonstrated better calibration and selectivity than the other boosting models. Still, its EV-based profitability remained limited, but the True Edge betting for LightGBM proved to be the better option.

Neural Network Results

Table 11 below presents the True Edge betting simulation results for the neural network during the 2025 MLB season.

Threshold	Bets	Total Profit	ROI
0.01	7815	-25835.5	-33.06%
0.02	4815	-19131.0	-39.73%
0.03	2661	-9569.0	-35.96%
0.05	590	-2423.0	-41.07%
0.08	168	-174.0	-10.36%
0.10	129	-102.0	-7.91
0.15	75	187.0	24.93%
0.2	54	82.0	15.19%
0.3	24	35.0	14.58%
0.4	6	-60.0	-100.00%
0.5	4	-40.0	-100.00%

Table 11: *Neural Network True Edge betting simulation results for the 2025 MLB season.*

The True Edge simulation results show that the neural network performed poorly across all low and moderate edge thresholds, with substantial losses ranging from −33% to −41% ROI between 0.01 and 0.05. The model generated far more bets than any of the tree-based models, over 7,800 wagers at the 0.01 threshold, suggesting that the neural network produced a large number of overly optimistic edge estimates that the market did not support. Performance improved only at higher thresholds, where the model became more selective: profitability emerged at 0.15 (24.93% ROI), 0.20 (15.19% ROI), and 0.30 (14.58% ROI). However, these gains were based on relatively small sample sizes, and the model's performance quickly destabilized at extreme thresholds, with −100% ROI at both 0.40 and 0.50. Overall, the neural network struggled to generate reliable True Edge signals, offering occasional strong returns at higher thresholds but exhibiting high variance and poor calibration across most of the season.

Table 12 below summarizes the Expected Value Ratio betting simulation results for the neural network during the 2025 MLB season.

Threshold	Bets	Total Profit	ROI
1.05	9583	-29162.0	-30.43%
1.10	7882	-26482.0	-33.60%
1.20	5237	-19938.0	-38.07
1.30	3437	-13140.0	-38.23%

1.40	2261	-8751.0	-38.70%
1.50	1558	-7021.0	-45.06%
2.0	206	-576.0	-27.96%
2.5	66	-63.0	-9.55%
3.0	38	-105.0	-27.63%

Table 12: *Neural Network Expected Value betting simulation results for the 2025 MLB season.*

Under the Expected Value betting framework, the neural network performed poorly across all EV thresholds, with large negative ROIs persisting throughout the full range of 1.05 to 1.50. The model produced significantly more wagers than any other model, over 9,500 bets at the 1.05 threshold. Yet, these high-volume predictions translated into consistent losses, indicating that the neural network systematically overestimated edge strength relative to sportsbook pricing. Although bet volume decreased as thresholds increased, profitability did not improve: even at EV ≥ 2.0, the model remained unprofitable, and the modest reductions in losses at 2.5 and 3.0 still reflected negative ROIs rather than meaningful positive signals. These results suggest that the neural network's probability estimates were insufficiently calibrated for EV-based decision making, producing many false-positive opportunities and failing to generate reliable EV wagers at any threshold tested.

Ensemble Models

After evaluating the performance of each individual model, an ensemble approach was developed to determine whether combining multiple predictive signals could produce more stable and profitable betting outcomes. Ensembles are widely used in machine learning because they reduce variance, smooth out model-specific noise, and often improve predictive calibration. Based on the full-season evaluation of all models, only a subset demonstrated consistently meaningful predictive ability, particularly at higher edge thresholds. LightGBM and CatBoost were the strongest performers, each generating positive returns at multiple True Edge levels and showing better calibration than the other algorithms. Random Forest was also included with a small weight due to its diversity and tendency to reduce variance, despite its weaker standalone performance.

Two ensemble structures were constructed: a simple ensemble and a weighted ensemble. The simple ensemble averaged the predicted probabilities of the selected models equally, providing a straightforward baseline to assess whether model diversity alone improved predictive accuracy. In contrast, the weighted ensemble assigned greater influence to stronger models, primarily LightGBM (0.6), followed by CatBoost (0.3), with a minimal contribution from Random Forest (0.1). This weighting reflects the relative performance of the individual models during walk-forward evaluation and is designed to amplify reliable signals while suppressing noisier contributions.

The following sections present the simulation results for the True Edge and Expected Value betting strategies for both the simple and weighted ensembles. By comparing these ensemble outcomes to the individual model results, we can assess whether aggregating predictions provides a more robust and profitable approach to forecasting daily home-run probabilities.

Simple Ensemble Results

The Simple Ensemble, which equally averaged the predictions of the selected models, showed mixed performance across both betting strategies, summarized in Table 13 (True Edge Simulation) and Table 14 (Expected Value Simulation). Under the True Edge framework, the ensemble was unprofitable at low and moderate thresholds, with ROIs between −19% and −26% from 0.01 to 0.05, indicating that averaging predictions did not eliminate the noise present in weaker model components. Profitability did not appear until higher thresholds, beginning at 0.08 (8.26% ROI), improving to 0.10 (34.44% ROI) and 0.15 (21.07% ROI), though these gains were based on smaller sample sizes. An extremely high ROI at the 0.20 threshold (397.5%) resulted from only two bets and reflects variance rather than reliable performance. A similar pattern emerged in the Expected Value simulations: large negative ROIs across EV ratios from 1.05 to 1.50 demonstrated that the ensemble's average probability estimates were not strong enough to overcome the sportsbook prices consistently. Positive returns occurred only at very high EV thresholds, 65.26% at EV 2.0 and 220% at EV 2.5, but again on very small numbers of wagers. Overall, the Simple Ensemble produced occasional profitable spikes at higher thresholds but lacked consistent profitability across the full season for a large sample size of bets.

Threshold	Bets	Total Profit	ROI
0.01	1421	-2835.6	-19.95%
0.02	682	-1559.6	-22.87%
0.03	404	-1036.1	-25.65%
0.05	152	-327.9	-21.57%
0.08	57	47.1	8.26%
0.10	34	117.1	34.44%
0.15	14	29.5	21.07%
0.2	2	79.5	397.50%
0.3	0	0	0.00%
0.4	0	0	0.00%
0.5	0	0	0.00%

Table 13: *Simple Ensemble True Edge betting simulation for 2025 MLB season*

Threshold	Bets	Total Profit	ROI
1.05	2037	-5408.6	-26.55%
1.10	1287	-3407.4	-26.48%
1.20	644	-1903.9	-29.56%

1.30	367	-1130.9	-30.81%
1.40	225	-1019.5	-45.31%
1.50	142	-409.5	-28.84%
2.0	19	124.0	65.26%
2.5	2	44.0	220.00%
3.0	0	0	0.00%

Table 14: *Simple Ensemble Expected Value betting simulation results for the 2025 MLB season.*

Weighted Ensemble Results

The Weighted Ensemble, which emphasized LightGBM and CatBoost while down-weighting Random Forest, showed improved performance compared to the Simple Ensemble but still demonstrated inconsistent profitability across thresholds. Under the True Edge strategy (Table 15), the ensemble reduced losses at lower thresholds and transitioned to positive returns at 0.08 (16.48% ROI) and 0.10 (17.62% ROI). These gains reflect the stronger contribution of LightGBM, whose calibrated predictions carried more influence in the weighted formulation. Profitability also appeared at the 0.20 threshold (142.14% ROI), though this was based on only seven bets and is therefore unstable. As with the simple ensemble, results at higher thresholds were dominated by low sample sizes, limiting their reliability.

In the Expected Value simulations (Table 16), the Weighted Ensemble again outperformed the Simple Ensemble but remained unprofitable across all moderate EV thresholds from 1.05 to 1.50, with ROIs between −8.93% and −32.34%. Positive returns emerged only at the highest EV ratios, where the model placed far fewer wagers, yielding 42.73% ROI at EV 2.0 and 168% ROI at EV 2.5, but on just 22 and 5 bets. These sharp increases in ROI reflect variance rather than consistent predictive strength. Overall, the Weighted Ensemble demonstrated clearer improvements over the Simple Ensemble, particularly in True Edge performance at mid-range thresholds, but still struggled to generate stable, scalable profitability across the full season.

Threshold	Bets	Total Profit	ROI
0.01	1338	-2922.8	-21.84%
0.02	657	-1485.0	-22.60%
0.03	372	-661.1	-17.77%
0.05	151	-62.1	-4.11%
0.08	61	100.5	16.48%
0.10	40	70.5	17.62%
0.15	19	6.7	3.53%
0.2	7	99.5	142.14%
0.3	0	0	0.00%
0.4	0	0	0.00%
0.5	0	0	0.00%

***Table 15:** Weighted Ensemble True Edge betting simulation for 2025 MLB season*

Threshold	Bets	Total Profit	ROI
1.05	1853	-3907.0	-21.08%
1.10	1184	-2714.1	-22.92%
1.20	579	-1872.5	-32.34%
1.30	335	-911.9	-27.22%
1.40	221	-578.3	-26.17%
1.50	135	-120.5	-8.93%
2.0	22	94.0	42.73%
2.5	5	84.0	168.00%
3.0	0	0	0.00%

***Table 16:** Weighted Ensemble Expected Value betting simulation results for the 2025 MLB season.*

Time Window Analysis

To examine whether model performance varied throughout the 2025 season, a time-window analysis was conducted focusing on the three strongest models: LightGBM, CatBoost, and the Weighted Ensemble. These models were selected because they were the only approaches that demonstrated meaningful profitability at higher edge thresholds and moderate acceptable losses at moderate and lower thresholds and therefore warranted deeper investigation. For each model, cumulative ROI was evaluated across key seasonal time periods: early season (April–May), midseason (June–July), and late season (August–September). This analysis provides insight into whether specific models were more effective in certain parts of the season and whether factors such as weather, roster changes, or shifting sportsbook pricing dynamics influenced predictive performance.

Time Window Analysis Results

A time-window analysis was conducted for the three strongest models: Weighted Ensemble, CatBoost, and LightGBM. Each model was evaluated at a representative True Edge threshold of 0.10. Figures 11, 12, and 13 display monthly ROI, bet count, and total profit for each model, revealing seasonal patterns and differences in model stability. Across all three figures, a common trend emerges: early-season months (March–April) and late-season months (August–September) were generally the most profitable, while midseason months (May–July) produced weaker and more inconsistent performance.

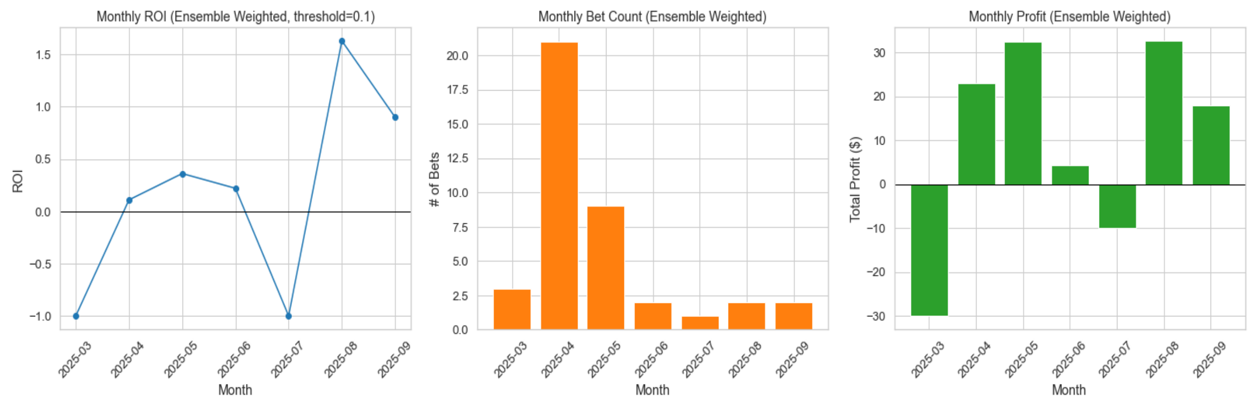

Figure 11: *Weighted Ensemble Monthly ROI, Bet Count, and Profit for True Edge 0.10.*

In Figure 11, the Weighted Ensemble shows strong profitability in April, May, and August, driven by a reasonable number of wagers in those months. March and July exhibit losses, though these months contained fewer bets, indicating that poor performance there may be partially attributable to variance rather than systematic model failure.

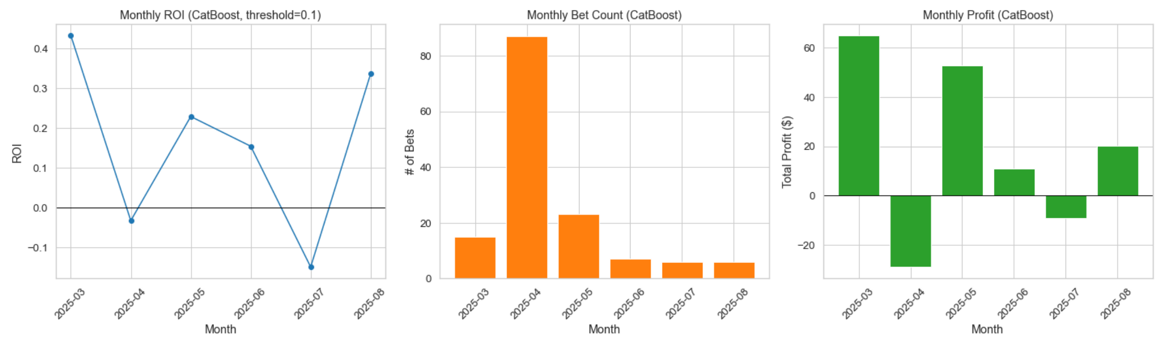

Figure 12: *CatBoost Monthly ROI, Bet Count, and Profit for True Edge 0.10.*

Figure 12 shows CatBoost displaying a similar structure: profitable early and late in the season, with April and May delivering the strongest profits. CatBoost placed more wagers than the ensemble, making its positive months more meaningful.

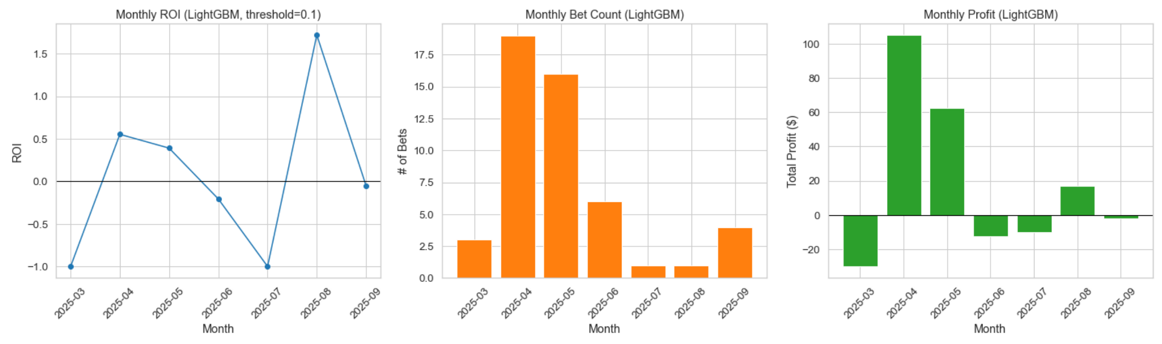

Figure 13: *LighGBM Monthly ROI, Bet Count, and Profit for True Edge 0.10.*

Among the three models, LightGBM (Figure 13) demonstrates the most precise pattern of strong, sustained performance. LightGBM's April and May results show both high ROI and substantial profit, supported by a relatively large number of bets, indicating genuine predictive strength rather than noise. Although June and July produce losses, the model rebounds sharply in August and maintains slightly negative results into September.

Player Specific Analysis

While aggregate model performance provides valuable insight into overall predictive ability, it can obscure meaningful variation at the individual player level. Because home-run hitting is highly concentrated among a small subset of power hitters and sportsbook pricing often differs substantially across player types (some players are always in the +200 range while others are always in the +1000 range), it is important to examine how the models performed for specific players. This analysis evaluates whether certain hitters were consistently profitable. By identifying which players the models predicted well or poorly, we gain a deeper understanding of where model-derived edges actually originated and whether the models captured true player-level signal or were influenced by noise. This perspective also helps assess the practical utility of the models for real-world betting, where bettors typically focus on individual players rather than season-long aggregates.

Player Specific Analysis Results

Player-specific profitability results reveal how differently each model allocated betting opportunities across the league and which hitters produced genuine positive returns. Across all three models, profitable outcomes tend to arise from players who were either:

1. Undervalued by sportsbooks despite strong underlying performance.
2. Lower-profile hitters whose odds remained high even during brief power surges.

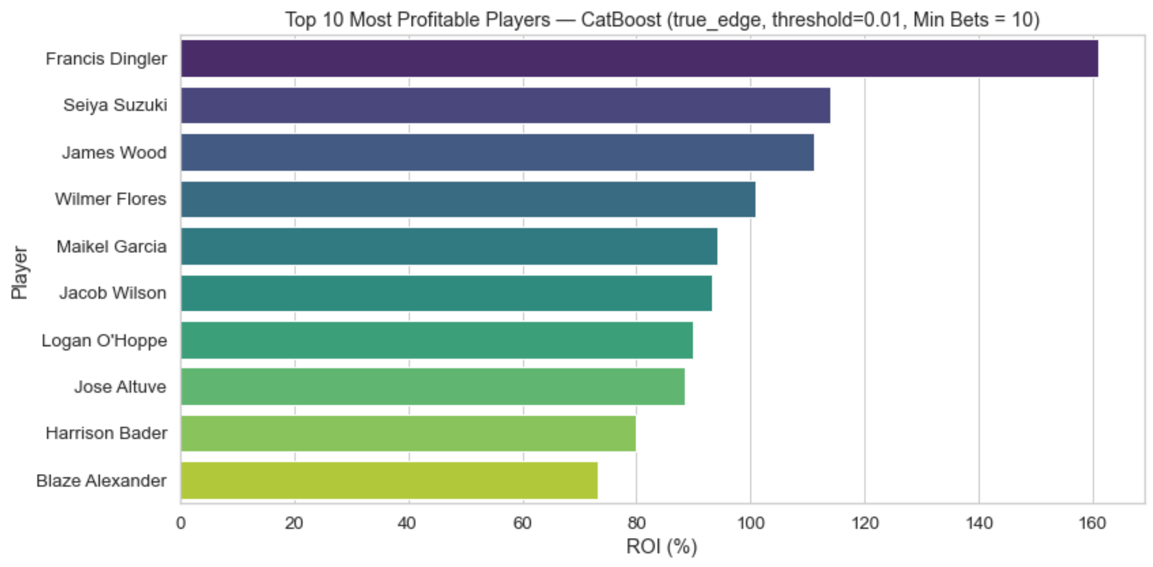

Figure 14: *CatBoost top 10 Most Profitable Players at True Edge 0.01.*

CatBoost (Figure 14) produced a wide range of extreme positive ROIs, highlighted by Francis Dingler, Seiya Suzuki, James Wood, and several mid-tier hitters who delivered exceptionally high returns often exceeding 100% ROI. Many of these players are not traditional elite home-run hitters, suggesting that CatBoost occasionally identified "hidden value" players that the sportsbooks had not priced correctly. However, the presence of unusually large ROIs, paired with relatively few wagers (minimum of 10) for some individuals, indicates that part of this outperformance may stem from small-sample variance rather than from a systematic predictive advantage.

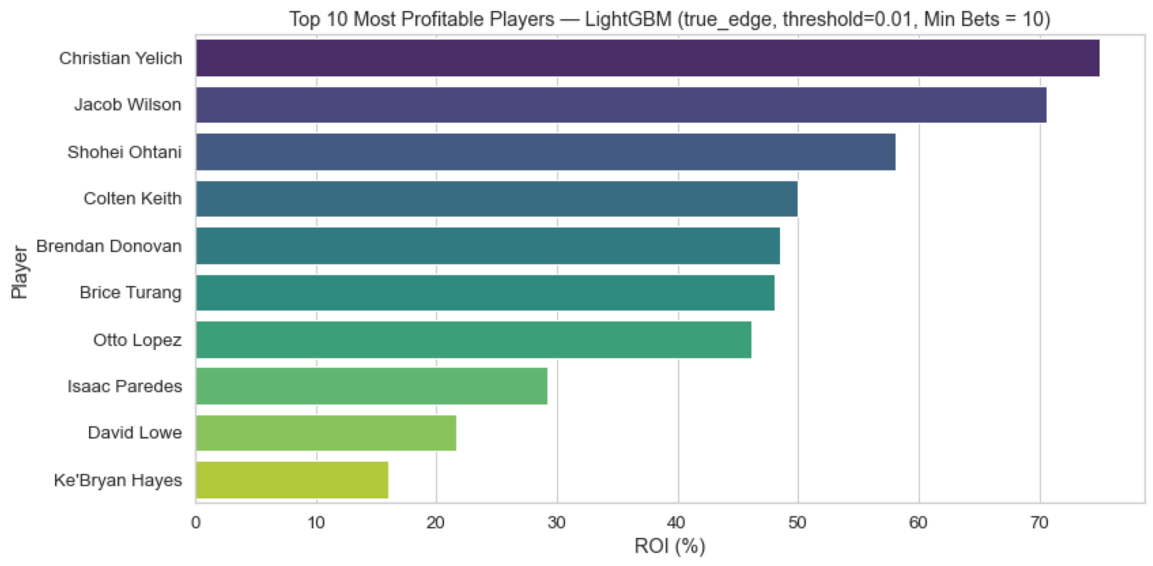

Figure 15: *LightGBM top 10 most profitable players at True Edge 0.01.*

LightGBM (Figure 15) generated a much more modest list of profitable players, with some showing only modest positive ROIs. The top returns came from established contributors such as Christian Yelich and Shohei Ohtani. These are players whose profiles align closely with consistent home-run production. Compared to CatBoost, LightGBM's player-level ROIs were less extreme, which points to its more conservative probabilities that led to more conservative bets and less volatility.

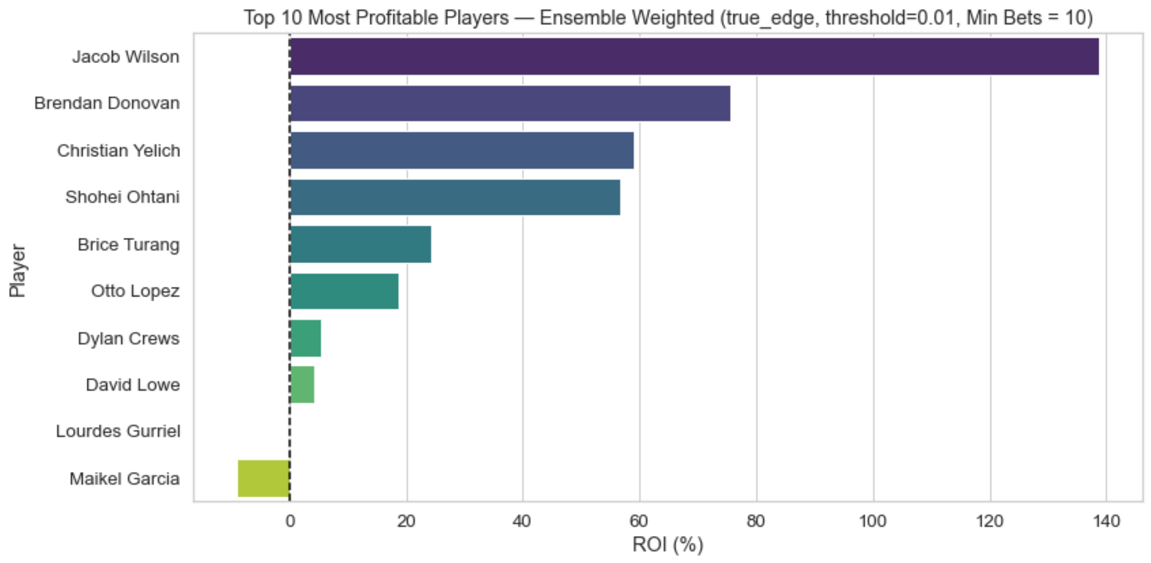

Figure 16: *Weighted Ensemble top 10 most profitable players at True Edge 0.01.*

The Weighted Ensemble (Figure 16) displays characteristics of both individual models. It identified several of LightGBM's profitable players (Yelich, Ohtani, Donovan) while also surfacing high-ROI opportunities such as Jacob Wilson, who delivered the strongest single-player return across all models. The ensemble's output appears more balanced than CatBoost's and less conservative than LightGBM's, indicating that combining model predictions allowed it to capture both steady producers and occasional overlooked value.

Notably, several players emerged as profitable across multiple models, including Jacob Wilson, Christian Yelich, Shohei Ohtani, Brendan Donovan, and Brice Turang. These repeated appearances suggest that the models independently detected consistent mispricing by sportsbooks rather than random noise. Cross-model agreement is particularly meaningful because it reflects underlying predictive consensus: when different algorithms identify the same players as profitable, it becomes far more likely that these hitters were genuinely undervalued throughout the season. As such, these players represent good examples of possible sustainable market inefficiencies.

Assessing the Impact of a Home Run the Day Before

An important question in both baseball analytics and sports betting is whether hitters experience short-term “streakiness,” particularly after hitting a home run. If players who homered the previous day are more likely to homer again, then a predictive model may overreact or underreact to recent events, leading to biased probability estimates. This is also a central question to this project: “Does streakiness affect player performance, and can it be captured in a predictive model?” To evaluate whether the models exhibited this type of recency bias, predicted home-run probabilities were compared for two groups of players: those who had hit a home run in the previous game and those who had not. For each model, the average predicted probability was calculated separately for both groups, and the difference between them was analyzed. This approach provides a straightforward way to assess whether the models implicitly assign higher home-run likelihoods to players coming off a strong performance, and whether such an effect reflects a meaningful signal or unwarranted overconfidence.

Model	Average Prob After HR	Average Prob After NO HR	Difference
CatBoost	0.15	0.11	0.04
LightGBM	0.14	0.11	0.03
Weighted Ensemble	0.14	0.11	0.03

Table 17: *Player predicted probabilities comparison for players who hit home runs the game before vs players who did not.*

The results of the streakiness analysis, shown in Table 17, indicate that all three models consistently assigned higher home-run probabilities to players who homered in the previous game. For CatBoost, the average predicted probability increased from 0.11 for players who did not homer the day before to 0.15 for those who did, a difference of 0.04. LightGBM and the Weighted Ensemble showed nearly identical behavior, each increasing their predicted probabilities from 0.11 to 0.14 following a prior-day home run. The remarkable similarity across models suggests a shared interpretation of recent performance: each model modestly elevates a hitter’s home-run likelihood after a successful prior game. This pattern implies that the models identify some short-term momentum or recency signal, whether driven by underlying changes in hitter form or simply correlated with increased odds-making attention. Importantly, the consistency across three independent modeling approaches indicates that this effect is unlikely to be random noise and may reflect a genuine, limited, predictive cue embedded in the data. This shows that while streakiness was captured in this form, it did not lead to profitable models, and shows that more than just recent stats, sportsbooks’ odds, and weather information are needed to actually beat the sportsbooks and build profitable betting models for home run props.

Conclusion

This project set out to evaluate whether home-run outcomes in Major League Baseball could be predicted using only recent player performance, contextual game information such as weather and park effects, and sportsbook odds. By restricting the feature set to short-term stats that capture players' current form, the goal was to assess whether recency-driven models could detect actionable signals in a highly volatile, low-frequency event such as home runs. The models were evaluated using a walk-forward procedure, probability calibration, and realistic betting simulations based on True Edge and Expected Value thresholds.

Overall, the results demonstrated that while the models were not profitable in aggregate, they did uncover meaningful predictive signal in the data. Certain models, most notably LightGBM and the Weighted Ensemble, showed consistent profitability at moderate True Edge thresholds, and several players emerged as repeat profitable opportunities across multiple modeling approaches, suggesting the presence of systematic sportsbook mispricing in specific cases for certain players. The time-window analysis further revealed that profitability tended to cluster during certain months of the season, such as early and late summer, indicating that temporal context plays an important role in the availability and strength of betting edges. Additionally, the streakiness analysis showed that all models modestly increased predicted home-run probabilities after a player hit a home run the previous day, reflecting a small but measurable recognition of short-term momentum effects.

While the majority of models did not achieve sustained full-season profitability, the collection of results highlights that targeted, model-driven betting strategies focused on specific players or certain stretches of the season may have practical value. More importantly, the study demonstrates that a predictive signal does exist, but extracting it consistently is challenging with the limited set of features available. Future work incorporating richer hitter-level information, including Statcast metrics such as exit velocity, launch angle, barrel rate, hard-hit percentage, and expected home run metrics, may substantially enhance predictive accuracy and betting profitability. These features capture the underlying swing quality and batted-ball skill that short-term box-score performance alone cannot fully represent.

Finally, although universal profitability remains elusive, this research demonstrates that machine learning approaches can identify pockets of inefficiency in MLB home-run markets. It lays a foundation for more advanced, data-rich modeling efforts in the future.

References

Ashoff, B. (2018). *Take me out to the ballgame: Weather's impact on Major League Baseball* (Master's thesis, Massachusetts Institute of Technology). https://hdl.handle.net/1721.1/118155

Bock, J. R., Maewal, A., & Gough, D. A. (2012). Hitting is contagious in baseball: Evidence from long hitting streaks. *PLOS ONE, 7*(12), e51367. https://doi.org/10.1371/journal.pone.0051367

DraftKings Major Qualifying Project Team. (2021). *Baseball data analysis for DraftKings* (Major Qualifying Project, Worcester Polytechnic Institute). Retrieved from https://digital.wpi.edu/

Elfrink, J. (2018). *Predicting Major League Baseball game outcomes using machine learning algorithms* (Master's thesis, Radboud University).

Gneiting, T., & Raftery, A. E. (2007). *Strictly proper scoring rules, prediction, and estimation*. Journal of the American Statistical Association, 102(477), 359–378. https://doi.org/10.1198/016214506000001437

James, B. (1982). *The Bill James Baseball Abstract.* Ballantine Books.

Kagan, D., & Atkinson, D. (2004). The coefficient of restitution of baseballs as a function of relative humidity. *The Physics Teacher, 42*(6), 330–333. https://doi.org/10.1119/1.1802051

Niculescu-Mizil, A., & Caruana, R. (2005). *Predicting good probabilities with supervised learning*. Proceedings of the 22nd International Conference on Machine Learning, 625–632.

Raab, T., Gula, B., & Gigerenzer, G. (2013). The hot hand in baseball revisited: New evidence from extensive data. *PLOS ONE, 8*(8), e71317. https://doi.org/10.1371/journal.pone.0071317

Smith, A. (2016). *Baseball and Markov chains: Using Markov models to predict baseball outcomes* (Master's report, University of Kansas).

Sun, H. C., Lin, T. Y., & Tsai, Y. L. (2022). Performance prediction in Major League Baseball by Long Short-Term Memory networks. *arXiv preprint arXiv:2206.09654.* https://arxiv.org/abs/2206.09654