Pickoff, Then Takeoff?

Predicting Stolen Base Attempts After Pickoff Moves

Team 214

1. Abstract

As part of new Major League Baseball (MLB) rule changes (Castrovince, 2023), pitchers are limited to two disengagements (pickoff attempt or step-offs) per plate appearance before a third unsuccessful disengagement allows the runner to advance. This rule has heavily impacted baserunning philosophy and defensive strategies. This paper explores runner behavior after pickoff moves. Specifically, I built a tool that predicts pitch-by-pitch steal attempts, given that a pitcher has already thrown over at least once during the at-bat. **Figure 1** displays the tool's interactive Shiny App interface. The model is built using an XGBoost classifier trained and tested on pre-rule change MiLB player tracking data, with feature engineering designed to capture key elements of runner dynamics. The analysis highlights patterns and situations that indicate when a runner is likely to attempt a steal, providing decision-making insights for defenses.

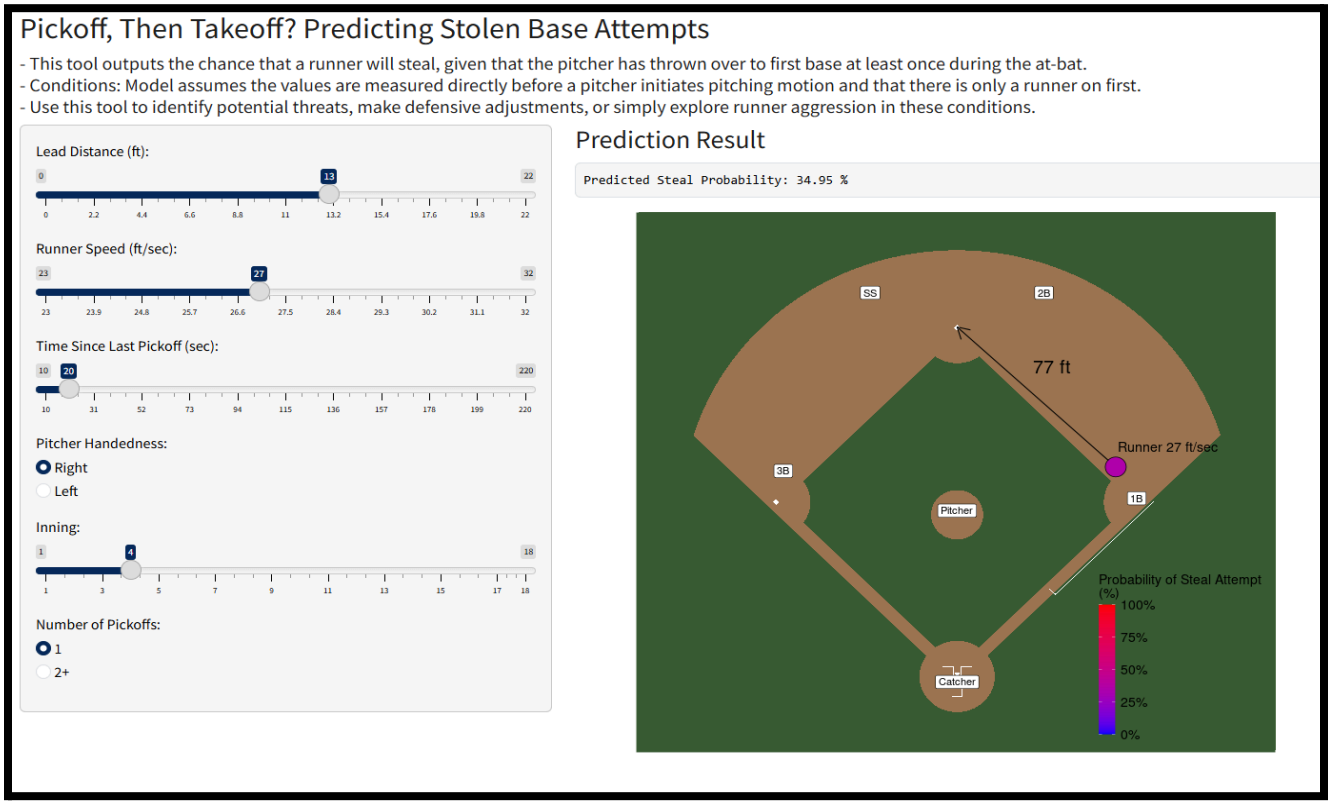

Figure 1: Shiny App interface. Adjustable parameters are along the sidebar, and the app displays a new predicted steal probability each time a parameter is adjusted. Predictions assume variables are measured directly before pitching delivery. If hyperlink is unavailable, URL: https://pickoffthentakeoff.shinyapps.io/pickoff-then-takeoff/

2. Introduction

Once 8-0 in the 3rd inning, the score is now tied in the bottom of the 6th. After giving up 8 unanswered runs, the Orioles need to get out of this inning to stay in the ballgame. With two outs and stolen base leader José Caballero on first, getting to second base could allow for a single to score a run. The Orioles are well aware of the threat. The pitcher, Seranthony Domínguez, tries a pickoff, but Caballero gets safely back to first.

Figure 2: Orioles defense performs a pitchout and gets José Caballero out on an attempted steal. After performing a pickoff move earlier in the at-bat, the Orioles seemingly predicted when he was going to steal.

This raises questions: given that a pickoff throw has occurred, what does Caballero do during the next pitch? Does he take a smaller lead and be less aggressive, afraid that the pitcher will throw over again? Or does he assume Domínguez won't throw over and call his bluff? Answering correctly could allow for the Orioles to act accordingly and get out of the inning. As **Figure 2** shows, they acted perfectly, keeping the game tied and illustrating the value in predicting stolen base attempts after pickoff moves.

I built an app that predicts steal attempts after pickoff throws. There are several existing projects that predict stolen base success. However, they do not attempt to predict *if* and *when* a runner is going to steal, nor do they focus on runner aggression *after* pickoff moves. My project fills in both of these gaps, and my tool considers scenarios where a pitcher has already thrown over, modeling situational runner aggression.

3. Data

The data for this project came from 274 anonymized Minor League Baseball (MiLB) games, with 5 teams and their players identifiable over 2 seasons. Sequential game events were organized by player positions, event types, player and ball locations, and timestamps. Ball tracking and player center-of-mass data indicated where a player or ball is on the baseball field throughout the duration of a play.

3.1. Steal Attempts After Pickoff Throws

I needed to find pickoff moves and stolen base attempts in the data. Since pickoff attempts were directly labeled in the data, I filtered for situations with a pickoff throw to first base, while second and third are vacant. I did not consider thethrow outcome, only when it occurred. I found ~1150 pickoff throws under these conditions.

To predict steal attempts on a pitch-by-pitch basis, I filtered for pitches that occurred *after* a pickoff move, then determined which of these post-pickoff pitches were also steals by selecting pitches from the same at-bat as a pickoff move that sequentially occurred after the attempt. I then excluded any pitches resulting in a ball put into play. My final dataset consisted of pitches after pickoff moves, labeled by whether or not a steal occurred during the pitch. Of the ~1050 pitches in my dataset, 57 were steal attempts.

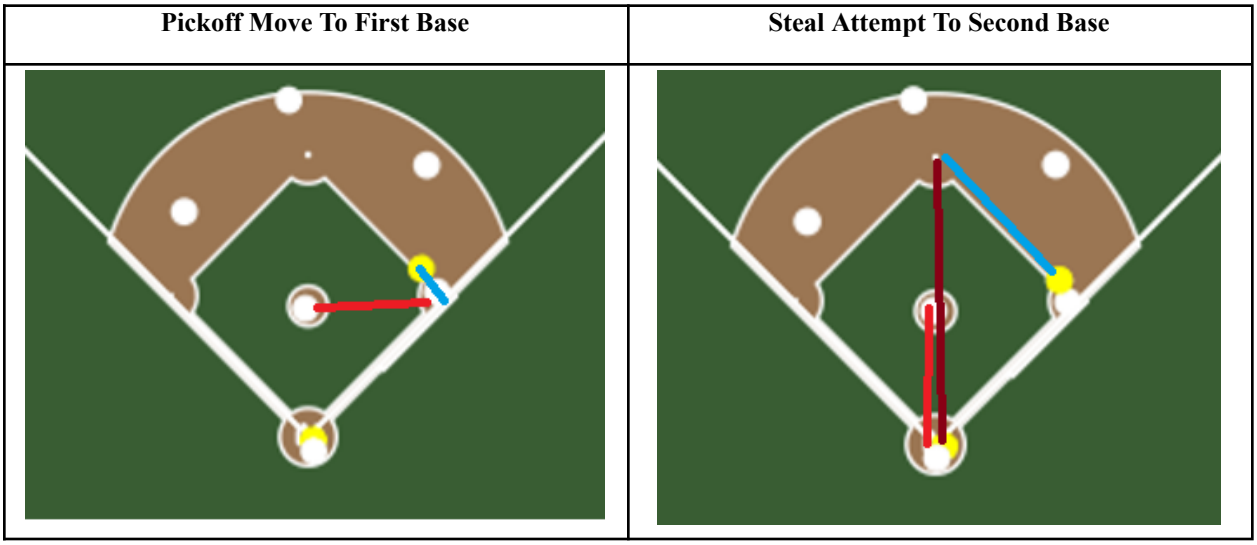

Figure 3: (left) Pitcher performs a pickoff throw to first base. The line in red depicts the pickoff throw, and the blue line is the runners direct back to the base. (right) Runner on first base attempts to steal second base during a pitch. The red lines show the ball path from pitch to catcher throwdown, and the blue line represents the runner's steal path. Both of these plays happened during the same at bat, with the steal attempt occurring on the very next pitch after the pickoff move.

Both plays depicted in **Figure 3** occur during the same at-bat, illustrating the scenario of focus.

4. Features

For my features, I calculated lead distance, sprint speed, time since last pickoff move, pitcher handedness, number of pickoffs, and game state.

4.1. Lead Distance

For lead distance, I calculated the runner's distance from the edge of first base at 1.2 seconds before the pitch release. Since player positional data tracks only a player's center of mass, it is impossible to measure the exact moment a pitcher initiates their delivery. I wanted to avoid measuring features *during* a pitch delivery, as steal attempts, secondary leads, etc. could all be taking place. At 1.2 seconds, a pitcher can still throw over, and defenders can adjust based on the runner's lead.

4.2. Sprint Speed

I emulated MLB's official sprint speed calculation: average sprint speed over 1-second running spurts during qualifying plays, such as running out a grounder or stealing a base (MLB, n.d.). I filtered to plays where the batter hits a ground ball, as these will frequently result in plays in which the runner is truly sprinting. I then calculated each runner's fastest 0.5-second spurt and defined their sprint speed as the average of those top-speed intervals.

4.3. Time Since Last Pickoff Move

I calculated how many seconds have passed since the last pickoff move. For example, if pitch *x* occurred 30 seconds after the last pickoff move, then the "time between" value is 30.

4.4. Pitcher Handedness

I classified handedness based on the ball's three-dimensional location at time of pitch release. If the ball's location is to the right of the center of the rubber, I classified the pitcher as right-handed, and the same logic for left-handed release points.

4.5. Game State (Inning)

My data does not implicitly provide count, score, outs, nor inning. However, some of these features can be calculated using other useful information present in the data. For example, I was able to calculate the inning number using a variable indicating whether the game was in the top or bottom of an inning and incrementing the inning by 1 after top/bottom cycles. Ideally, I would include outs, score, and count, but ultimately, I ended up only using the inning number.

4.6. Number of Pickoff Attempts

Since pitches and pickoff attempts were sequentially stamped, I was able to count the number of pickoff attempts that have occurred during the at-bat up until a given pitch. Each pitch is denoted by either a 1 or 2, indicating how many pickoffs have occurred earlier in the at-bat. If

only 1 pickoff has occurred, it is labeled as a 1. For more than that, I denoted the number of pickoff attempts as 2, regardless of how many actually occurred. I did this in light of the new MLB rule change that limits the number of pickoff attempts to 2, penalizing the 3rd attempt if unsuccessful. Since my data is from before the rule change, there were instances of 2+ pickoff attempts, and I did not want to model how runners behave after 3 or 4 pickoff attempts, as this condition is impossible in today's game.

5. Analysis

To create my tool, I evaluated two types of models: logistic regression and XGBoost. Given that steals happen very infrequently, I opted for an XGBoost model because it is effective in dealing with large class imbalances. I employed an 70/30 test/train split and utilized the previously discussed features as model inputs. Additionally, I tuned the model to understand the rarity of steals, increasing the prediction accuracy for when they actually do happen. Also, prioritized minimizing false negatives, instances where the model misses actual steal attempts.

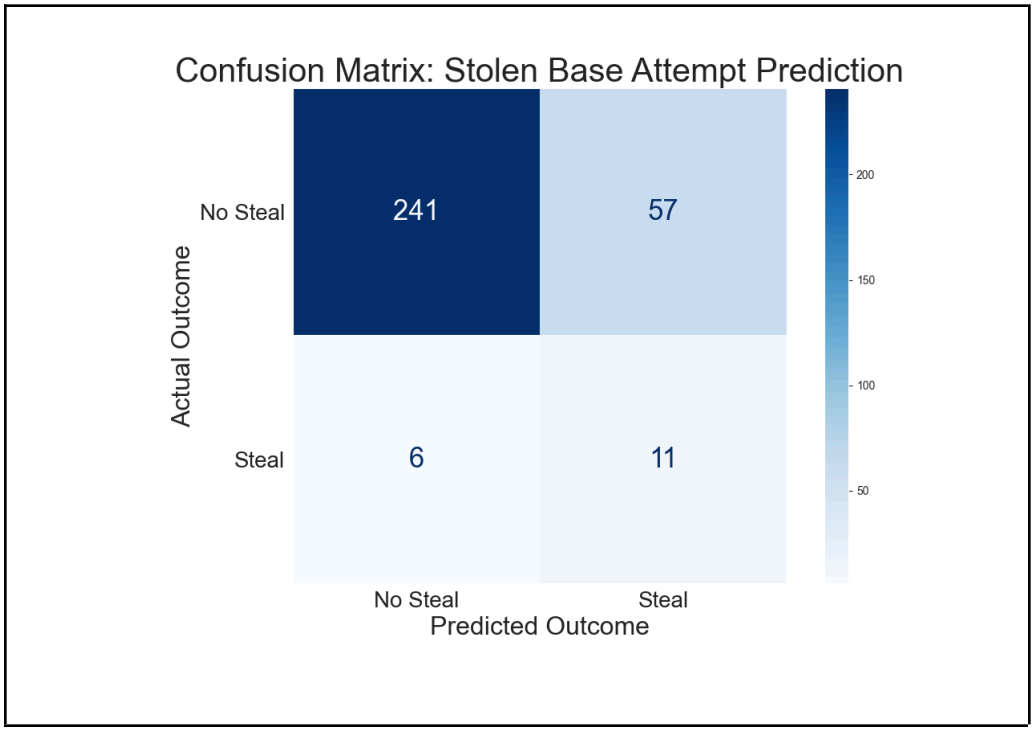

Figure 4: Confusion Matrix for stolen base attempt predictions on testing portion of dataset. Model accurately predicted 241 non-steals and 11 real steals. Model mislabeled 6 real steals as non-steals and 57 non-steals as real steals.

Figure 4 shows the predicted outcomes on the testing portion of the data. Essentially, the model was able to accurately predict **65%** of actual steal attempts and had **80%** overall accuracy. While it incorrectly predicted a steal would actually occur 57 times, it only missed 6 real steal attempts in a dataset with 315 pitches. This demonstrates that the model is sensitive and aware of actual steal attempts. Despite the high false positive rate, the results benefit the original goal of

the project: building a tool that identifies when a runner is likely going to steal following a pickoff attempt, providing valuable insight into baserunning decisions and subsequent defensive adjustments. Incorrectly thinking a steal will occur is far less damaging than missing real steal attempts, as a false alarm will – at worst – result in an unnecessary pitchout. This tradeoff promotes anticipation for aggressive baserunning while minimizing defensive blindsides.

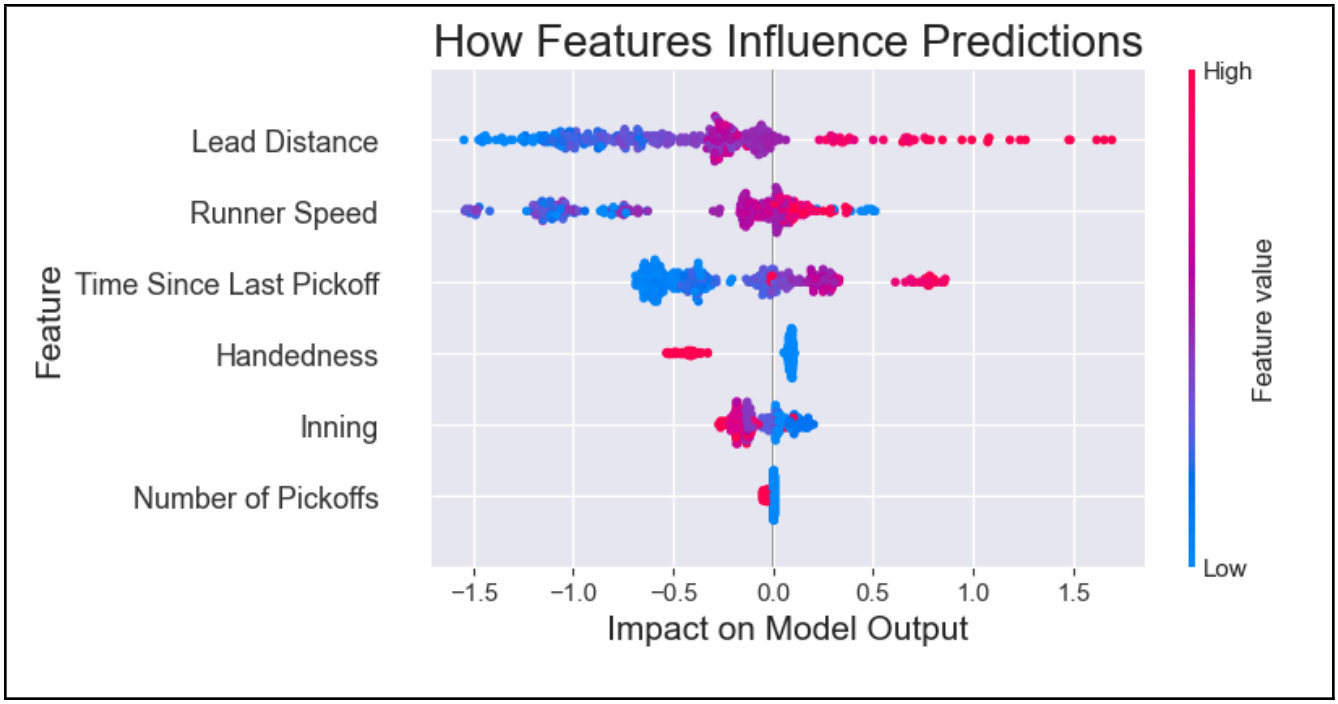

Figure 5: This SHAP summary plot illustrates how individual values impact the model's prediction decisions. Negative SHAP values push predictions toward lower steal probabilities, while positive values push them toward higher probabilities. The color bar indicates the raw feature values for each observation. Left-handed pitchers are denoted as 1, and right-handed pitchers are 0, and this is reflected in the heat colors.

SHAP provides the feature value and impact for each individual value, rather than aggregating each feature as a whole. **Figure 5** depicts a SHAP summary plot, which shows how each feature contributes to shifting a prediction away from the average prediction. Each dot on the plot represents a pitch, and dots are labeled by their respective feature value; a blue dot from the lead distance feature indicates the runner had a short lead during this pitch instance. A dot's horizontal position determines how much a feature pushes the prediction up or down relative to the average predicted probability.

Below, I detail how each feature influences steal predictions:

Lead Distance

- Instances where a runner has an exceptionally large lead results in high steal probabilities, and the contrary for extremely small leads.
- There is a cluster of observations in which the runner had a decently large lead, but they pulled the prediction lower than expected.

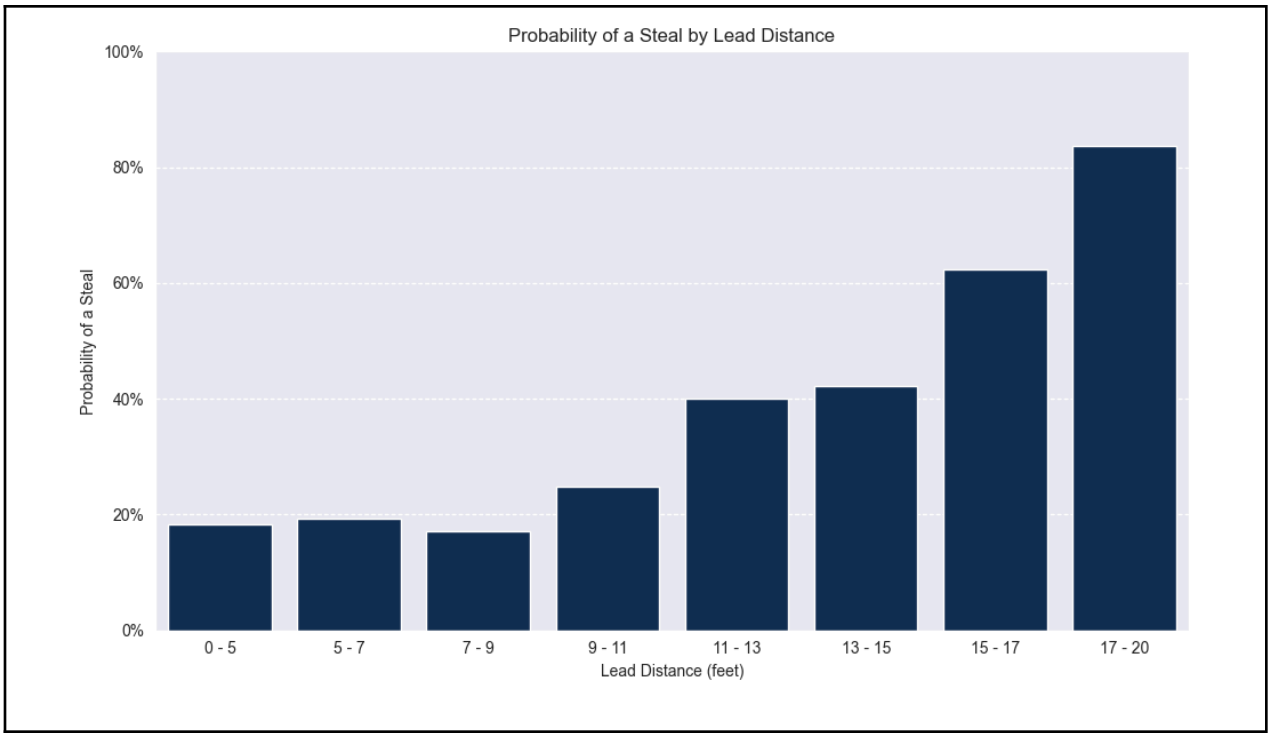

Figure 6: Lead Distance and Steal Attempt Probability

- **Figure 6** conveys the relationship between lead distances and the likelihood that a runner will steal, based on the model's predictions. There is a clear positive trend, indicating that furthering a runner's lead increases their chance of stealing, according to the model.
- Overall, smaller leads contribute to lower probabilities.

Runner Speed

- As expected, fast runners push predictions towards high steal probabilities, as the cluster of bright red skews positively.
- Interestingly, the observations that yield the highest predicted probabilities were all extremely slow players. This is likely caused by the runner's lead distance in these plays; if a slow player has an exceptionally large lead, then they are likely going to steal, especially since these are instances where there is only a runner on first base. Such leads are unusual for slow players and are strong signals of an impending steal attempt.

Time Since Last Pickoff

- This feature displays a clear, controlling dynamic. Instances where a pitcher recently performed a pickoff attempted push predictions towards lower steal probabilities, whereas instances where a significant amount of time has passed influences high predicted probabilities.

- This relationship suggests that pickoff moves have a “fear factor,” meaning that runners are less likely to steal shortly after a pickoff attempt, but then become increasingly more likely to steal as time goes on.
- For example, with the time since last pickoff variable set to 20 seconds, the predicted steal probability is 34.95%. Keeping all other variables constant and increasing the time since last pickoff value to 92 seconds, the probability is now 54.97%, visualized in **Figure 7** below.

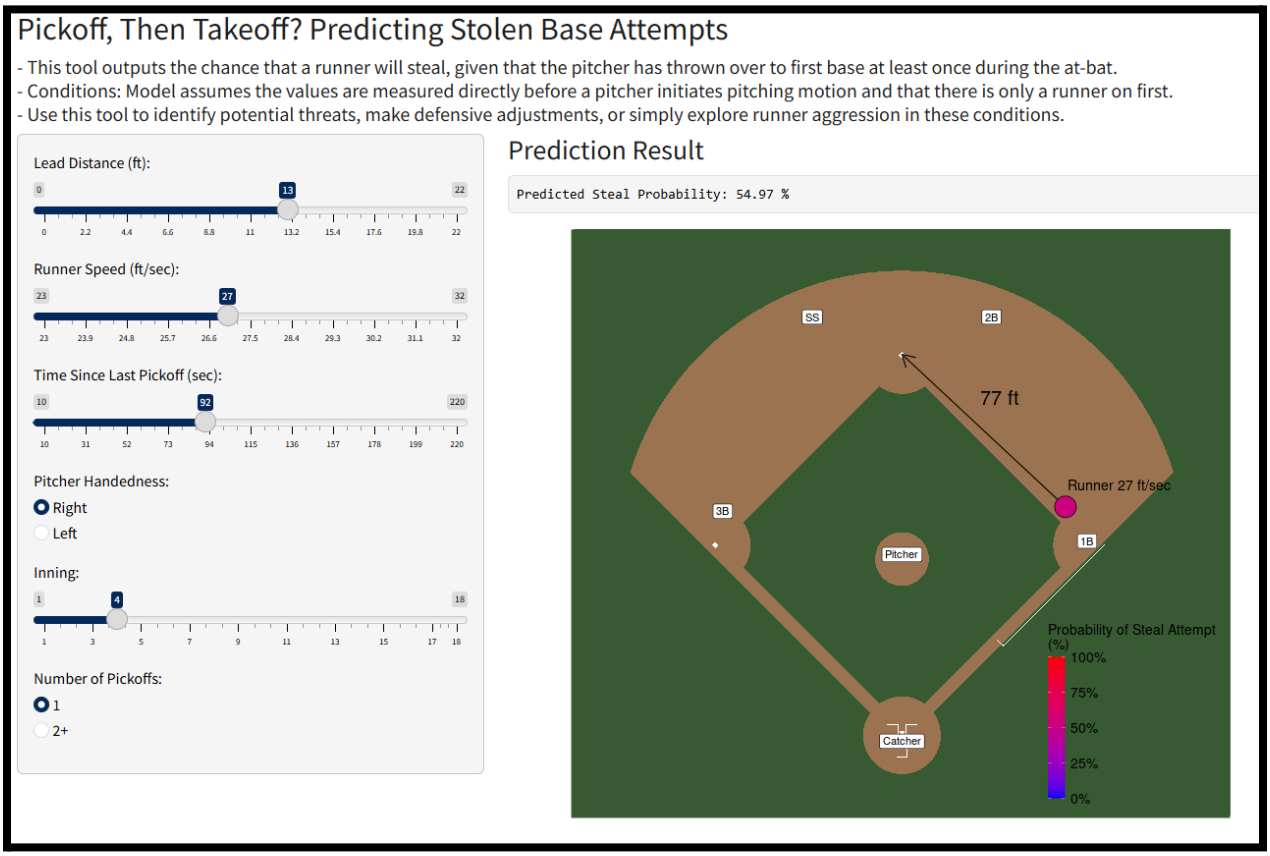

Figure 7: App display of new predicted steal probability after changing time since last pickoff value to 92 seconds, keeping all other variables constant. The predicted steal chance changed from 34.95% to 54.97%.

Handedness

- There is a binary split between left- and right-handedness. Left-handed pitchers are associated with small predicted probabilities, whereas right-handed pitchers influence the model toward larger predicted probabilities.

Inning

- Earlier innings are associated with higher predicted steal probabilities, while later innings correspond to lower probabilities. This may be because runners are more cautious about

being thrown out in crucial late-game situations, where baserunners are considered more valuable.
- Interestingly, baseball's unwritten rule of not stealing bases during blowout games could also impact late-game dynamics.

Number of Pickoff Moves

- This feature has almost no impact on model predictions, likely due to the time since the last pickoff feature, which captures a similar dynamic as the number of pickoff moves but is far more detailed and useful for the model.
- In today's game, I expect this feature to play a much larger role due to the rule change.

6. The Tool

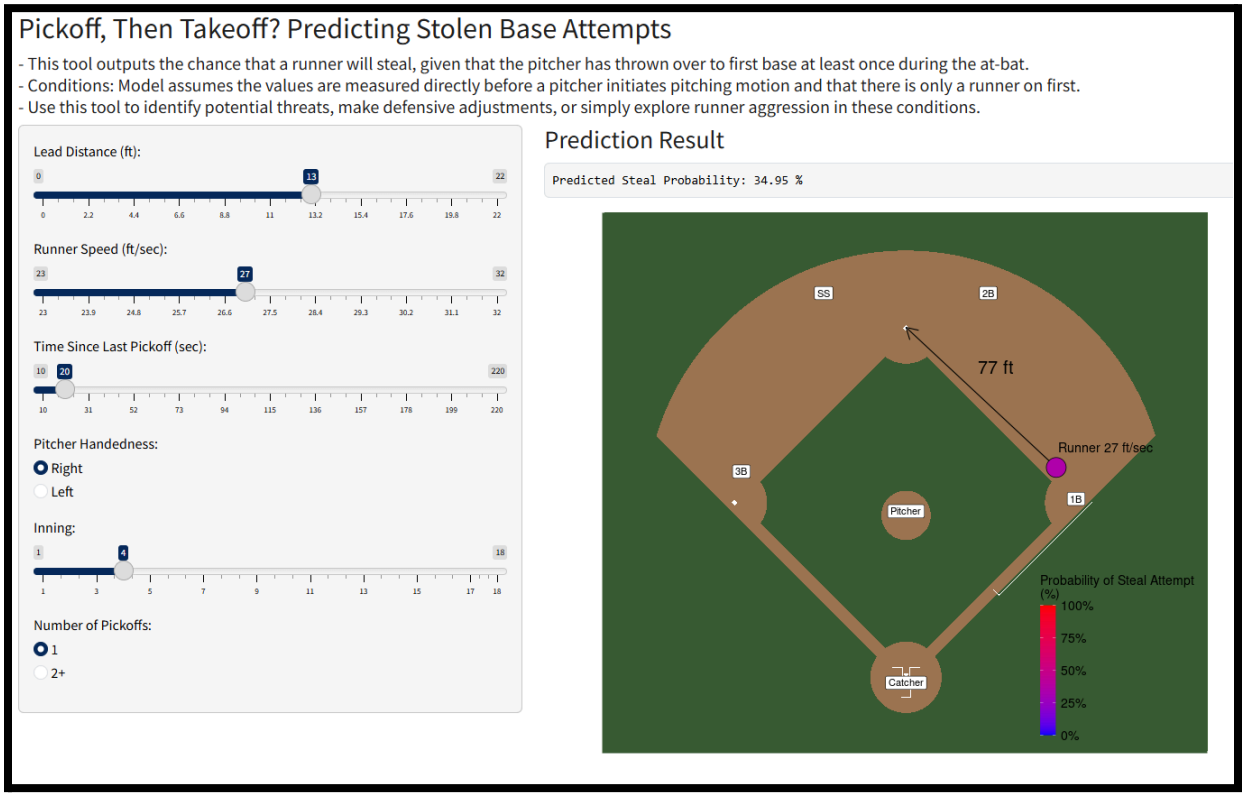

Figure 8: Shiny App interface. Adjustable parameters are along the sidebar, and the app displays a new predicted steal probability each time a parameter is adjusted. Predictions assume variables are measured directly before pitching delivery. If hyperlink is unavailable, URL: https://pickoffthentakeoff.shinyapps.io/pickoff-then-takeoff/

Figure 8 shows the app interface. Parameters can be adjusted on the left side, and the predicted steal probability is displayed each time a variable is adjusted. Ideally, this tool can be used to provide decision-making insights for in-game defenses or post-game to determine if the pitcher made the right decision after the fact. Additionally, by experimenting with the parameters, this tool can be used to explore runner aggression post-pickoff moves. Knowing the steal likelihood during specific hypothetical conditions can provide value for actual in-game circumstances. Regardless of its use, a tool like this can help coaches and players better understand runner tendencies after pickoff moves.

7. Considerations

My tool falsely classifies actual non-steal attempts as steal attempts ~18% of the time, mainly due to the rarity and unpredictability of steals. If a team was able to use it in real time, there would be false alarms, leading to unnecessary defensive adjustments, like pitchouts or a change in the catcher's receiving position. Using game state data that included variables like outs, count, and score could improve prediction accuracy and allow teams to interpret

probabilities more strategically. For instance, in high-leverage situations, even a small change in predicted probability might warrant a defensive response.

In future iterations of this project, I would like to include factors like catcher pop time, average throw accuracy, pitcher delivery time, and other prior statistics that evaluate defensive ability. I would also like to look at niche situations where runners are on first and third, as pickoffs happen due to the fear of the double steal.

8. Acknowledgements

I am extremely grateful to Dr. Meredith Wills and Billy Fryer for their support throughout this project. I would also like to thank members of the Data Challenge's Discord for their collaborative help when answering questions.

9. Appendices

9.1. Appendix A: Feature Engineering

Steals were captured by filtering plays in which a pitcher throws a pitch to home, a catcher receives it, meaning the batter does not make contact, and the runner is significantly far away from first base (x coordinate < 30), indicating an advancement. From here, I filtered out instances where the ball gets away from the catcher, which would indicate a passed ball or wild pitch, not a stolen base.

Lead distance was calculated by taking the Euclidean distance between the runner's position and the edge of first base 1.2 seconds before time of pitch release. The coordinates for first base were calculated using trigonometry. The back corner of first base is 90 feet away from the back of home plate, at a 45 degree angle, so I could use sin and cos to find its respective x, y coordinates on the diamond. To find how far away the runner is from first, I needed to use the inner corner of first base, as this is the most representative of how far away a runner is from the base. In some cases, play timestamps only began at the moment of pitch release, preventing me from tracking the runner's position prior to that point. This affected ~300 pitches out of ~1050. Rather than removing these observations, I adjusted them by taking the lead distance at pitch release and subtracting the mean difference observed between these incomplete timestamps and normal observations. This approach allowed me to retain important data, especially steals.

Sprint speed was calculated by finding a particular player's average fastest 0.5-second spurts during plays where the ball hits the ground after contact. Although this filtering could lead to plays where it is a bloop, a blast, or some other outcome where the runner is not trying to sprint, it measured sprint speed at a pretty successful rate. The average sprint speed was 26.95 ft/sec, extremely similar to the MLB average of 27 ft/sec. Via checking by empirical rule and plotting the distribution, I can say that my average sprint speed calculation is very close to normally distributed, and the Central Limit Theorem (CLT) satisfies this assumption.

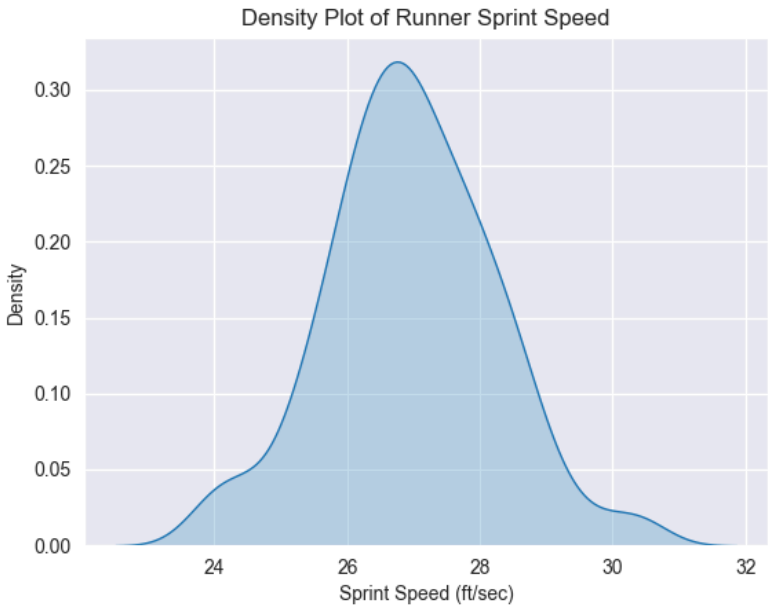

Figure 9: Density Plot of Runner Sprint Speed

I had plenty of rows where the runner ID was missing, so I substituted missing runner ID sprint speed with the mean sprint speed.

9.2. Appendix B: Modeling

Since I am dealing with a binary classification problem, I considered logistic regression, but ultimately opted for XGBoost. I had a severe class imbalance and dynamic, potentially non-linear relationships between my features and stolen base attempts. A logistic regression assumes that the effect of each predictor on predicted probability is constant, regardless of other predictors' values. This is not the case for my data, as my features have complex interactions with each other, such as instances where a slow player has a large lead. XGBoost naturally handles dynamic feature interactions better.

I used a grid search to tune parameters, with a focus on maximizing recall rather than overall accuracy. Because missing a steal attempt is worse than falsely predicting one, I prioritized capturing as many true steal attempts as possible, even at the cost of additional false positives. This approach lowers overall accuracy, as you can have extremely high accuracy just predicting "no steal" every time due to the rarity of steals, but it better suits the problem. I combined grid search results with manual tuning. I set the scale_pos_weight to be the ratio between no steals and steals, which helps with the class imbalance. I used 200 rounds of boosting with a low learning rate of 0.02, as well as a low max tree depth. High boosting rounds, low learning rates, and shallow trees gave me the best predictions. Steals are inherently difficult to predict, so tuning my model to generalize rather than overfit resulted in the best output.

When making predictions on the testing portion of the data, I was able to obtain an AUC score of 0.728, meaning the model performs significantly better than randomly guessing when a steal will occur (AUC=0.5).

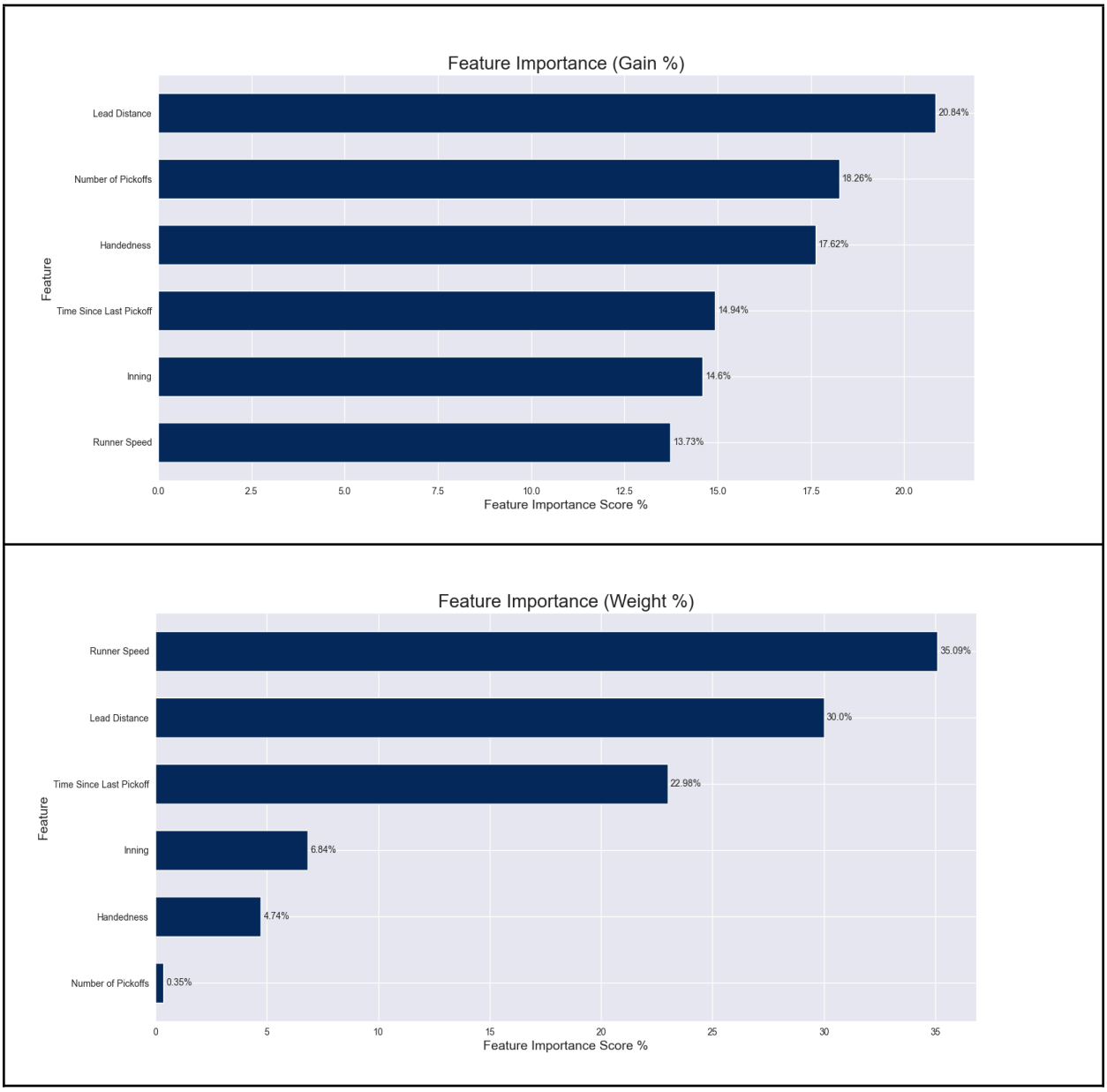

Figure 10: (Top) This feature importance plot depicts feature importance in terms of gain, which shows how useful each feature is for making accurate predictions. (Bottom) This feature importance plot depicts feature importance in terms of weight, which indicates how often the model uses the feature when making predictions.

Figure 10 illustrates the relative importance of each feature according to two different evaluation metrics: gain and weight. Gain shows how useful each feature is for making accurate predictions, whereas weight indicates how often the model uses the feature when making predictions. Comparing the plots shows several interesting dynamics. Sprint speed has a high weight but low gain, meaning that the model utilizes the feature very frequently, but it only adds

a small improvement in prediction accuracy. On the other hand, the number of pickoffs and handedness have low weights but high gains, suggesting that the model does not use the features often, but when it does, they make a very large impact. Lead distance and time since the last pickoff have high importance scores for both metrics, indicating that the model uses them frequently and they make large impacts.

9.3. Appendix C: Analysis

As mentioned previously, lead distance played a large role in predicting stolen base attempts for the model. The average lead distance for plays in which a stolen base attempt occurred is 13.4 feet, which is 2.46 feet further than the plays where a stolen base attempt did not occur.

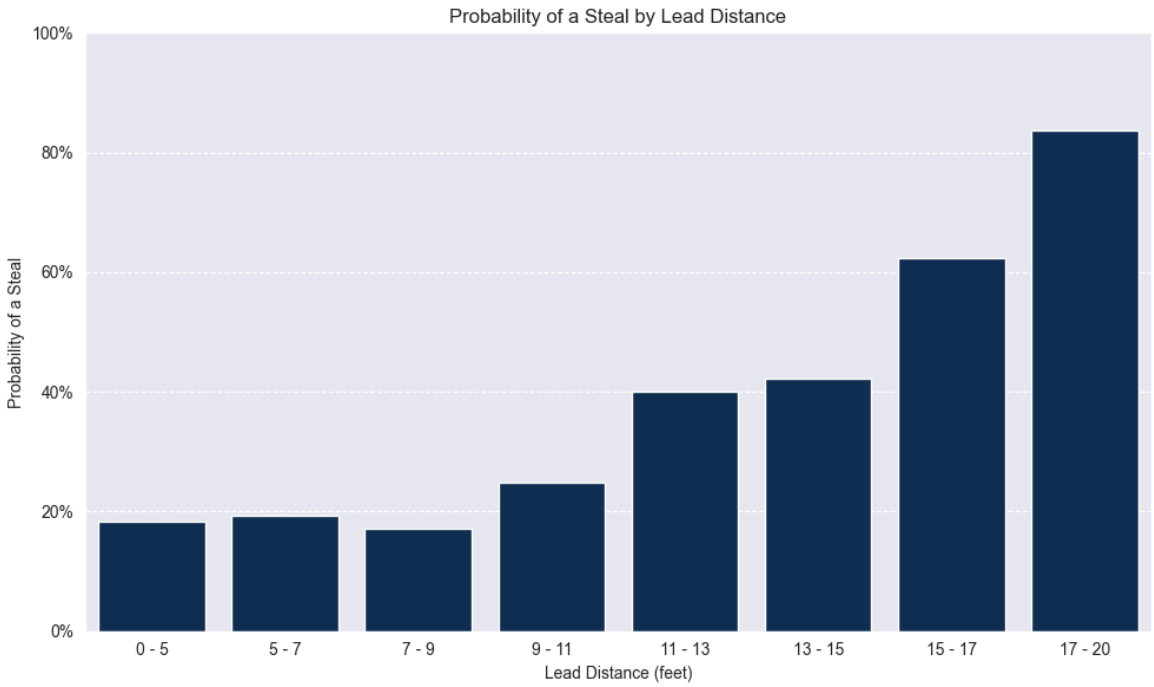

Figure 11: Lead Distance and Steal Attempt Probability

As seen above, **Figure 11** conveys the relationship between lead distances and the likelihood that a runner will steal, based on the model's predictions. There is a clear positive trend, indicating that furthering a runner's lead increases their chance of stealing.

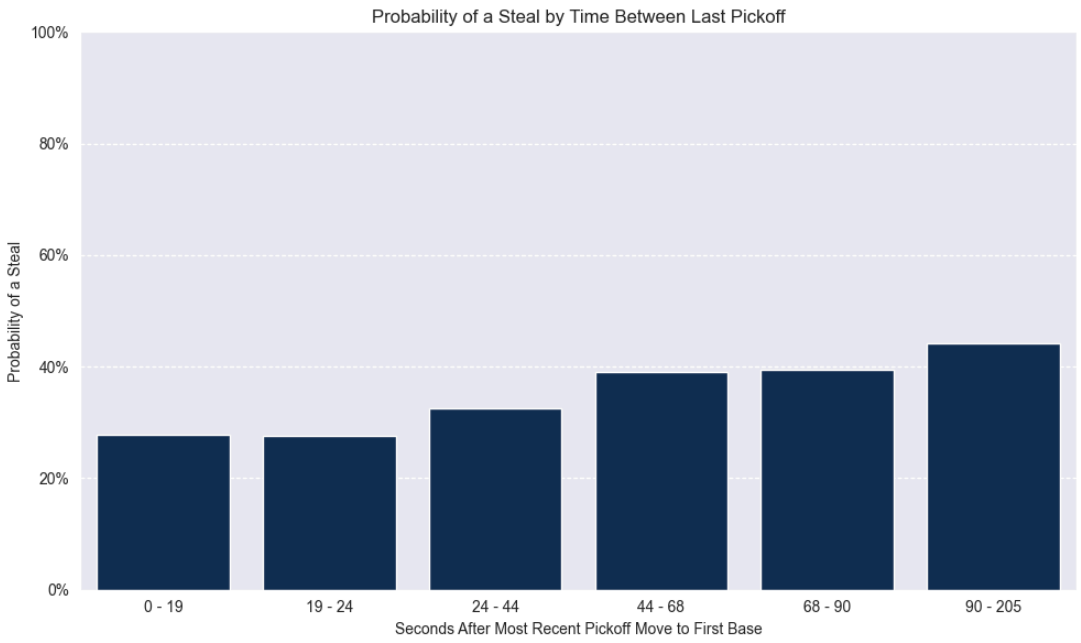

Figure 12: Time Between Last Pickoff and Steal Likelihood

Figure 12 depicts how proximity to a pickoff move impacts the model's predictions. It illustrates that recency to pickoff moves significantly impacts the predicted chance of an attempted steal. If a pitcher throws over, the model predicts the runner to be less likely to steal immediately after but increasingly likely to steal as the at bat goes on. This suggests that the recency of a pickoff deters runner aggression, but its "fear factor" effect severely fades over time.

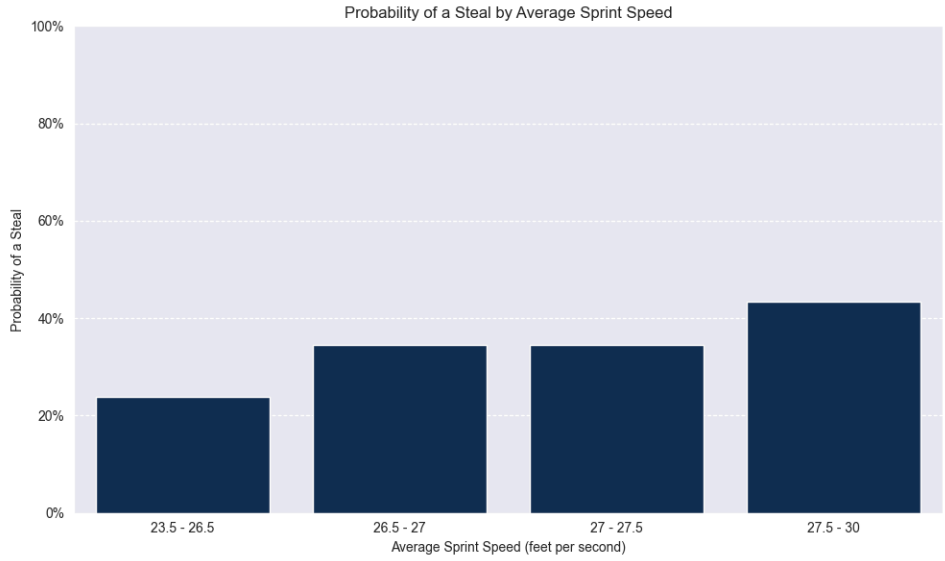

Figure 13: Sprint Speed and Steal Likelihood

As expected, **Figure 13** shows that the model predicts faster runners to be more likely to steal. Pitches involving runners in the top 25% of sprint speed (27.5–30 ft/sec) show the highest steal likelihood. Interestingly, runners in the middle 50% are – on average – similarly likely to steal, possibly indicating that – unless the runner is decently fast or particularly slow – then sprint speed alone might not strongly indicate steal intent.

10. References

Castrovince, A. (2023, February 13). *Pitch Timer, Shift Restrictions Among Announced Rule Changes For ’23*. MLB.com. https://www.mlb.com/news/mlb-2023-rule-changes-pitch-timer-larger-bases-shifts

Lundberg, S. M. & Lee, S.-I. (n.d.). *SHAP: SHapley Additive exPlanations*. SHAP Documentation. https://shap.readthedocs.io/en/latest/

Major League Baseball. (n.d.). *Sprint Speed (SS): Glossary.* MLB.com. https://www.mlb.com/glossary/statcast/sprint-speed