Identifying UCL Injury in MLB Pitchers Using Machine Learning and Single-Season Sabermetrics

Timothy I. Morales

Department of Statistical Science, Southern Methodist University

STAT 6341: Sports Analytics

Dr. Charles South

December 16, 2020

Abstract

Ulnar collateral ligament, UCL, reconstruction, also referred to as Tommy John surgery, is a procedure prevalent in the population of Major League Baseball, MLB, pitchers. Associated with long recovery times and decreases in general player performance, UCL injury can be harmful to not only the player, but also the franchise as a whole. With a significant increase in the incidence of UCL injury since 1990, there is a greater importance each year to identify individuals at risk of a UCL tear. This research uses machine learning methods including uniform manifold approximation projection, random forest, logistic regression, boosted tree models and model stacking techniques to use single season statistics and Statcast sabermetrics to classify individual MLB pitcher's individual seasons associated with UCL injury from the population of MLB individual pitching seasons from 2015 through 2019. The results show an ensemble model utilizing multiple random forests, a logistic regression and a boosted tree model is able to correctly recall 31.25% of individual seasons associated with UCL injury while maintaining a precision of 15.63%, roughly 8 times more accurate than a random guess on a test set designed to mimic the population of MLB pitchers in an individual season

Keywords: Model ensemble, Tommy John, UCL injury, machine learning.

Introduction

There is money in America's favorite pastime. Forbes reports that Major League Baseball, MLB, saw a record $10.7 billion in revenue in 2019. That $10.7 billion includes a $5.1 billion dollar television deal with Fox. The MLB also signed a $1 billion uniform deal with Nike for the 2020 season (Brown, 2019). With that huge revenue comes great costs, a $4.7 billion payroll to be exact (Brown, 2019). Fans are paying to watch the MLB, brands are paying to host the MLB, fashion companies are paying to dress the MLB, and teams want to make the most effective use of their payroll to maximize profits. Since all contracts in the MLB are guaranteed, player injury is at the forefront of mitigating payroll risks, especially injuries associated with long-term performance decreases.

Ulnar collateral ligament, UCL, reconstruction, more commonly known as Tommy John surgery, is notorious in the game of baseball. The sport, involving over-hand throwing motions, has also been associated with high risks of elbow injury. Over-hand throwing motions can produce extremely high levels of valgus stress in the elbow, eventually leading to instability and increased risk for UCL injury (Chen et al., 2001). Recent research has shown MLB pitchers are statistically the most at-risk group for UCL injury (Conte et al., 2015). In 2018, Fangraphs, a commonly used baseball database, reported that 86.7% of MLB games in the 2017 season included at least one pitcher who had undergone UCL surgery in the past (Roegele, 2018).

Recovery times from UCL reconstruction vary between 12 and 15 months and can span across multiple seasons. Not only does the surgery require long recovery times, recent research has found the surgery to be linked to performance decreases (Selley et al., 2019). These long recovery times as well as long-term performances decreases are concerning for the league, the

team, the individual players and the fans. With the afore mentioned billions of dollars going into player payroll each year, understanding and preventing injuries like UCL tears has huge monetary value.

Since 2010, there have been multiple studies centered around identifying key risk factors for UCL injury. There have also been studies designed around classifying at-risk players relative to a control group. In Erickson et. al. (2014), researchers found MLB players who attended high schools in warmer climates to be at a significantly higher risk of UCL injury. Keller et. al. (2015) found that individuals who throw a higher percent of fastballs were more likely to undergo the surgery relative to a control but found no difference in the effect of velocity.

Most similar to the research in this paper is Whiteside et al. (2016) where researchers used support vector machines and naïve bayes models utilizing variables like average spin rates and average velocity to classify individuals against a control group of equal size. Although Whiteside et. al. (2016) was able to classify individuals correctly 75% of the time, this number is an extreme overestimate of the model's actual predictive power. This 75% is the maximum average five-fold cross validation accuracy from testing over 4000 different models. Each fold was 41 or 42 individuals and had perfect class balance. Therefore, with no hold-out set to assess the optimal model's true performance, as well as test folds that are only 42 individuals with perfect class balance, these findings do not stem from a research design that mimics the practical use of a classification model for UCL injury. Roughly 1000 player pitch in the MLB each season and only 2% of those pitchers will undergo UCL injury within the following year.

This paper seeks to address some of the concerns within the previous research as well as build upon prior findings by using a design that more closely emulated the population of MLB pitchers. The classification models in this research also include the latest MLB Statcast information. This Statcast information, also referred to as sabermetrics, includes individual yearly pitcher spin rates, velocities, movements, and pitch counts across various pitch types. This new Statcast database was released by the MLB in 2015 and is publicly accessible through MLB.com's Baseball Savant library (Major League Baseball, 2020). Rhinehart (2020) published a thesis comparing the means of these Statcast variables between a control group and a pre-UCL injury group. Although multiple comparisons were made, each with an alpha of 0.05, the study found multiple significant difference between the groups and illustrates the potential these variables have in classification.

Through a more holistic approach, this paper uses single-season traditional counting statistics, the new single-season Statcast database, and personal information including height, weight, and birthplace to build a classification model that correctly predicts individual seasons associated with an individual who will undergo UCL reconstruction in the near future. This research uses methods including, SMOTE, random forest, neural networks, boosting models, logistic regression and model stacking to correctly classify the entire population of pitchers at risk between the 2015 and 2019 seasons.

Methods

Data Collection

The analysis involved data from three separate sources, the Lahman database, Jon Roegele's Tommy John surgery list, and the afore-mentioned MLB.com Baseball Savant

database. The Lahman (2020) database is an open-source collection of baseball counting statistics that is easily accessible through RStudio (2020) and is maintained by journalist, Sean Lahman. Single season pitching statistics and single season postseason pitching statistics were collected from the Lahman database for all pitchers who appeared in the 2015 through 2019 seasons. The Tommy John surgery list was obtained from the open-source list kept by Jon Roegele, a former writer for Fangraphs, a well-known baseball statistics database and website. Within the surgery list are all players publicly known to have undergone UCL surgery as well as the surgery date and multiple player IDs (Roegele, 2020).

The Lahman statistics were joined to the Tommy John statistics using Baseball Reference IDs. The Baseball Reference IDs for the Tommy John list were obtained from another former Fangraphs, author, Tanner Bell who publicly posts a baseball ID translation sheet on his website, SmartFantasyBaseball (Bell, 2020). Using the Fangraphs IDs prevalent in both the Tommy John list and the translation list, Baseball Reference IDs were added to the Tommy John list data. The Tommy John list was then merged with the Lahman data by Baseball Reference ID and year through the *left_join*( ) function in the *dplyr* package in Rstudio (Whickham, 2020).

The Tommy John list was collected on September 23, 2020. For all individuals who underwent surgery in 2020, the 2019 season statistics were marked as the season associated with the injury. For all other years, if a player underwent UCL surgery before March 31st, that player's previous season statistics were used and marked as the season associated with the UCL injury. For example, if Jacob deGrom underwent UCL surgery on March 31st, 2019, Jacob deGrom's 2018 statistics are flagged as the season associated with the UCL injury. If Jacob deGrom underwent UCL surgery on April 1st, 2019, Jacob deGrom's 2019 season statistics are flagged as the season associated with UCL injury. The March 31st cutoff for all seasons other

than 2020 is used so that the data and sabermetrics most closely aligned with the UCL injury are used to fit the model. Using a cutoff of roughly opening day will help amplify the signal, if any, within the sabermetric statistics. Those who are injured during the season will use that season's statistics rather than back tracing a full year.

The tradeoff and potential issue of using an early cutoff is running the risk of players standing out due to low amounts of playing time in a season rather than the actual risk factors. This issue was addressed by requiring pitchers to have a minimum of 30 batters faced in order for that pitcher's individual season to be included in the analysis. The 30 batters faced minimum requirement was adopted from the 10 innings pitched requirement used in Whiteside et. al. (2016). Batters faced was chosen rather than innings pitched due to the volatility in the duration of an inning. The results section also includes model prediction performance results for an additional test set that was created from the total data set by following the same methodology and pseudorandom seed but with the cutoff of July 31st rather than March 31st. The optimal model for predicting the March 31st cutoff data was also used to predict the July 31st test set in a form of sensitivity analysis. The date of July 31st was chosen as it is well past the mid-way point of the season. The additional results are to help check if the model is solely relying on low playing time for its classification prediction as well as to illustrate the importance of recency for accurate classification.

The MLB.com's Statcast variables did not include any ID information other than player names. Player name and year were used to merge the previously merged Tommy John list and Lahman data frame with the Statcast data frame. Overlapping variables including games pitched and appearances were used to cross reference the player names and insure that the merge was performed properly. This was again done using the *dplyr* package (Whickham, 2020). There

were 76 player names that did not overlap between the two datasets, most because of how certain naming conventions like “Jr.” were displayed. For these players, Baseball Reference IDs were collected from the Baseball Reference website itself, again using the aforementioned statistics for cross-reference (Sports Reference, 2020). With the Baseball Reference IDs, the data was merged for 70 out of the remaining 76 players. A total of six players were removed from the data because their identity could not be confirmed. Before enforcing a minimum of 30 batters faced, the data frame consisted of 3645 out of 3651 eligible major league pitchers.

Data Manipulation

The fully merged dataset was subset to only include individuals with over 30 batters faced, resulting in 3301 observations with each observation indicating an individual season for an individual pitcher. Of the 3301 observations, 69, or 2.09%, were marked with a flag variable to denote a season associated with a UCL injury. A postseason flag variable was created denoting if an individual recorded more one third of an inning pitched in the post season. A variable denoting warm birthplace was created with idea and execution from Keller et. al., (2015). The main difference between this variable and the variable used in Keller et. al., was this variable used birthplace rather than high school. All states within the United States denoted as warm in Keller et. al., were denoted as “warm” in the warm birthplace variable. For those born outside of the United States, individuals born in countries other than Lithuania, Germany and Canada were denoted “warm.” The variables pitches per game, pitches per out, pitches per batter faced and batters per inning were also feature engineered. All near zero variance variables were dropped other than postseason games started, postseason saves, postseason innings pitched,

postseason batters faced, and the UCL injury indicator using the *nearZeroVar( )* function from the *caret* package (Kuhn, 2020).

Missing data was an issue for many of the Statcast and postseason variables. Missing values for variables denoting counts or counting statistics were set to zero. The remaining variables with missing information were missing as a direct result of an individual not qualifying for a certain statistic. These variables where many individuals did not qualify related to individual pitch-type information such as spin rate, velocity, velocity difference, and break distance across the different classifications of pitches thrown by MLB pitchers. Since these variables were the main focus of the analysis and all of these variables contained some form of missing data, the variables were binned into quartiles using the *bin( )* function from the *OneR* package in R (Jouanne-Diedrich, 2017). The variables were binned into quartiles denoting if the individual during that individual season was in the top 25%, top 25 - 50%, top 50 - 75%, or bottom 25% of all individual seasons for all individual players between 2015 – 2019. Along with these four levels corresponding to quartiles, an additional level denoting the player did not qualify for that statistic was applied to players with missing information for that variable. This binning strategy allowed the research to include all individual players while also maintaining some of the variance present in the pitch-specific variables. These binned variables and all other factor variables including year, throwing arm, warm birthplace, league, side of plate for batting, and the postseason flag were one-hot encoded using the *step_dummy( )* function in the *recipes* package in R (Kuhn and Wickham, 2020).

After one hot encoding the data, the resulting data frame included 3301 observations and 337 variables. Using a stratified 75-25 train-test split based on the UCL flag variable, the data frame was split into a test set composed of 825 observations with 16 (1.9%) individual seasons

associated with a UCL injury as well as a training set composed of 2476 observations with 53 (2.14%) of the observations associated with a UCL injury. The 75-25 split was chosen as it created a test set of 825 observations, which replicates a population of a single season of MLB pitchers. According the Baseball Reference website, 1035 different individuals pitched in 2019 and 694 of those 1035 faced at least 30 batters (Sports Reference, 2020). Variables in the training and test set that were not one hot encoded were centered and scaled independently to a mean of 0 and standard deviation of 1 using the s*cale( )* function in the *base* package in RStudio (R Core Team, 2020).

With a training set featuring such extreme class imbalance and an ultimate goal of minority class recognition, a form of over-sampling referred to as Synthetic Minority Over Sampling Technique (SMOTE) was applied to the training set. Japkowicz (2000) found traditional methods of over-sampling with replacement do not significantly improve minority class recognition. SMOTE, a form of synthetic over-sampling, creates synthetic examples that are generated from neighboring minority class observations in the feature space (Chawla, 2002). This synthetic over-sampling helps generalize minority class recognition in machine learning models by forcing the classifier to make larger and more generalized decision regions in the feature space (Chawla, 2002). Setting the percentage over sampling parameter to 1000% and the percent under sampling parameter to 400% created a training set with 3727 observations, 2120 (78.43%) observations in the majority class and 583 (21.56%) observations in the minority class.

Data Visualization

The original training set, the synthetically over-sampled training set, and the test set were visualized in a two-dimension space to evaluate the overall structure of data. The main focus of

the dimension reduction and plotting was to visualize the difference between the majority and minority group across the varying data sets. With the goal of maintaining the structure within the data to observe potential clusters or grouping of observations that correspond to seasons associated with UCL injury, uniform manifold approximation projection (UMAP) was used. UMAP was chosen over t-SNE and other dimension reduction techniques for its superior run time performance and ability to arguably preserve more global structure (McInnes et. al., 2018). The UMAP visualizations were colored by the value of the UCL injury flag. Evaluating the structure of the data provided a better understanding of which model spaces would be most effective in classifying the minority class. The visualization also helped show how representative the structures of the training and over-sampled training sets were of the test set.

Data Models

After the initial visualization of the data, a random forest model was optimized using the over-sampled training data and the *ranger( )* function in the *ranger* package in R (Write and Ziegler, 2017). The number of samples to possibly split at each node (mtry) was able to take values of 5%, 15%, 25%, 33.33% or 40% of the 335 variables included in the model. The minimum node size was able to take a value of 1, 3, 5, or 10. Sampling could be either with or without replacement and be either 50%, 75% or 100% of the full data set. The classification threshold or probability needed to be denoted a season associated with UCL was also trained in the optimization and was able to take values of 0.1, 0.15, 0.3, 0.4 and 0.5. The optimal parameter space was recorded in terms of precision, recall, and a weighted sum of precision and recall. The weighted sum was calculating by summing 150% of the recall value with the precision. Each of the trees fit in optimization were optimized using the gini splitting rule and with 3350 trees.

The optimal parameters for the three models corresponding to optimal precision, recall, and the weighted sum, were again fit to the over-sampled training set but with 4000 trees. Oshiro et. al., (2012) shows that in a high dimension space, roughly 4000 trees assures the model has trained fully. The optimal random forests fit with 4000 trees were used to predict the test set. Confusion matrices were plotted using the *confusionMatrix( )* function in the *caret* package for both training and test set predictions (Kuhn, 2020). The *confusionMatrix( )* function also returns recall and precision metrics for the predictions.

In an effort to look at the performance of a classifier in the linear space, a logistic regression model was fit to the synthetically over-sampled training set using the *train( )* function inside the *caret* package (Kuhn, 2020). This model was then used to predict the test set. Training and test set results were again presented using the *confusionMatrix( )* function in the caret package.

A boosted tree model was optimized on the synthetically over-sampled training set using the *xgb.cv( )* function from the *xgboost* package in R (Chen et. al., 2020). The boosted tree model was optimized on binary classification error rate using a binary logistic objective function and 10-fold cross validation. The maximum number of iterations mirrored the number of trees allowed in the random forests, 4000. The early stopping rounds parameter was set to 50. The parameter space of the boosting model was optimized incrementally starting with the step size, which was able to take the values 0.005, 0.01, 0.05, 0.1, and 0.3. With the optimal step size, the maximum tree depth and minimum sum of an instance weight were optimized from the values 1, 3, 5, 7 and 5, 10, and 15 respectively. Using the optimal step size, tree depth, and sum of an instant weight, the subsampling, column subsampling per tree and column subsampling per node were optimized and each allowed to take values 0.5, 0.75 and 1. With the previous optimal

parameters, penalization was applied and optimized to help with general performance. Gamma was able to take a value of 0, 1, 10, or 100. Lambda and alpha were able to take values of 0, 0.001, 0.01, 0.1, 1, 10, or 100. Finally, with the optimized parameters, the step size was reevaluated and optimized from the same list values as the first optimization step. The optimal boosted tree model was fit to the over-sampled training data using the *xg.boost( )* function using the *xgboost* package (Chen et. al., 2020). This model was then used to predict the test set. Training and test set results were presented using the same *confusionMatrix( )* function as prior.

In an effort to see how the models would work together, a model ensemble/stacked model was fit using the *SuperLearner* package in R (Polley et. al., 2019). The three optimal random forests, the logistic regression and optimized boosted tree model were recreated to be compatible with the stacked model using the *create.Learner( )* function. The compatible models were combined into a stacked learner. Performance of the stacked model on the over-sampled training set was calculated and compared to the performance of the individual models using the *CV.SuperLeaner( )* function. The final stacked model was created using the *snowSuperLeaner( )* function and included the random forests, logistic regression and boosted tree model. The final stacked learner was used to predict the over-sampled training set. A list of probability threshold values needed for an observation to be classified as associated with UCL injury ranging from 0.05 to 0.95 by 0.05 were plotted against model recall and precision for the training set. The largest probability threshold associated with the minimal loss in recall performance in the over-sampled training set was noted. This maximum probability threshold was used in an effort to create the most precise model possible without compromising recall performance. The stacked learner was then used to predict the test set. A confusion matrix for the test set classifications

was created when using both the traditional 0.5 threshold as well as the noted maximum probability threshold.

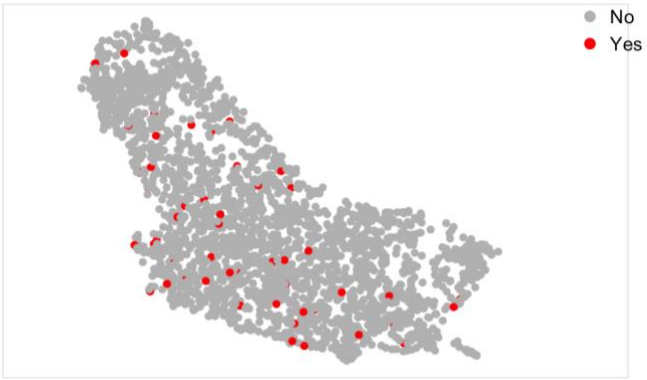
Finally, to address the concern surround the model continually selecting individuals with small amounts of playing time, a new test set was created using the same exact data manipulation procedures as prior, including SMOTE, in a form of sensitivity analysis. For the new test set, the cutoff date was set to July 31st instead of March 31st to ensure surgeries in the first half of the season were associated with the prior season's statistics. The July 31st test set was predicted using the stacked model trained on the original over-sampled training set. Performance on the two test sets was compared to evaluate if the stacked model was relying too heavily on low amounts of playing time as well as asses how signal carried over year to year.

Results

Results in the analysis are reported model by model in the order presented in the methods section. The positive class is denoted by a season associated with UCL surgery.

Figure 1

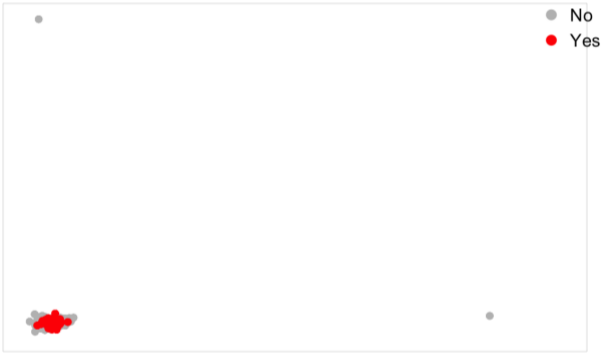
UMAP Training Set no SMOTE



Note: Red Denotes Season Associated with Surgery.

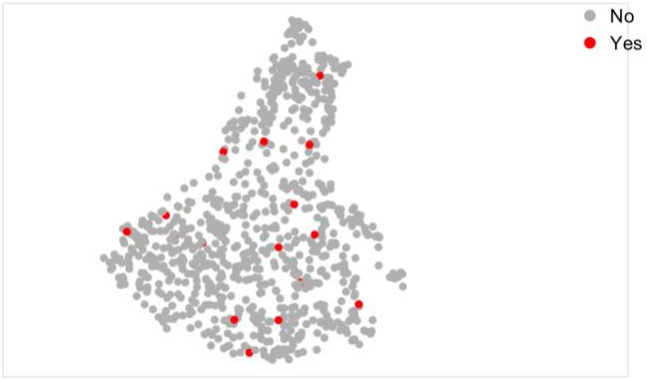
Figure 2

UMAP Training Set with SMOTE



Note: Red Denotes Season Associated with Surgery.

Figure 3
UMAP Test Set with SMOTE



Note: Red Denotes Season Associated with Surgery.

Random Forest

Table 1
Random Forrest Optimized on Precision Performance
Training Set with SMOTE Performance

		Predicted Class	
		Surgery	No Surgery
Actual Class	Surgery	500	83
	No Surgery	3	2117

Recall = *85.76%*
Precision = 96.23%

Table 2
Random Forrest Optimized on Precision Performance
Test Set Performance

		Predicted Class	
		Surgery	No Surgery
Actual Class	Surgery	6	10
	No Surgery	198	611

Recall = 37.50%
Precision = 2.941%
Minority Class = 1.94%

Note: Optimal Parameters. mtry = 134, Min Node Size = 1, Replace = True, Sample Fraction = 0.5, Probability Cutoff = 0.

Table 3
Random Forrest Optimized on Recall Performance
Training Set Performance

		Predicted Class	
		Surgery	No Surgery
Actual Class	Surgery	581	2
	No Surgery	411	1709

Recall = 99.*66%*
Precision = 58.57*%*

Table 4
Random Forrest Optimized on Recall Performance
Test Set Performance

		Predicted Class	
		Surgery	No Surgery
Actual Class	Surgery	16	0
	No Surgery	789	20

Recall = 100*%*
Precision = 1.988*%*
Minority Class = 1.94*%*

Note: Optimal Parameters. Mtry = 16, Min Node Size = 3, Replace = False, Sample Fraction = 0.75, Probability Cutoff = 0.1

Table 5
Random Forrest Optimized on Weighted Sum Performance
Training Set with SMOTE Performance

		Predicted Class	
		Surgery	No Surgery
Actual Class	Surgery	557	26
	No Surgery	45	2075

Recall = 95.54*%*
Precision = 92.52*%*

Table 6
Random Forrest Optimized on Weight Sum Performance
Test Set Performance

		Predicted Class	
		Surgery	No Surgery
Actual Class	Surgery	11	5
	No Surgery	514	294

Recall = 68.75%
Precision = 2.095%
Minority Class = 1.94%

Note: Optimal Parameters. Mtry = 83, Min Node Size = 1, Replace = False, Sample Fraction = 0.75, Probability Cutoff = 0.3. Weighted Score = 1.5 x Recall + Precision.

Linear Model

Table 7
Logistic Regression
Training Set with SMOTE Performance

		Predicted Class	
		Surgery	No Surgery
Actual Class	Surgery	545	38
	No Surgery	23	2097

Recall = 93.48%
Precision = 95.95%

Table 8
Logistic Regression
Test Set Performance

		Predicted Class	
		Surgery	No Surgery
Actual Class	Surgery	3	13
	No Surgery	139	670

Recall = 18.75%
Precision = 2.1127%
Minority Class = 1.94%

Boosted Tree Models

Table 9
Boosted Tree Model
Training Set with SMOTE Performance

		Predicted Class	
		Surgery	No Surgery
Actual Class	Surgery	583	0
	No Surgery	0	2120

Recall = 100%
Precision = 100%

Table 10
Boosted Tree Model
Test Set Performance

		Predicted Class	
		Surgery	No Surgery
Actual Class	Surgery	4	12
	No Surgery	49	760

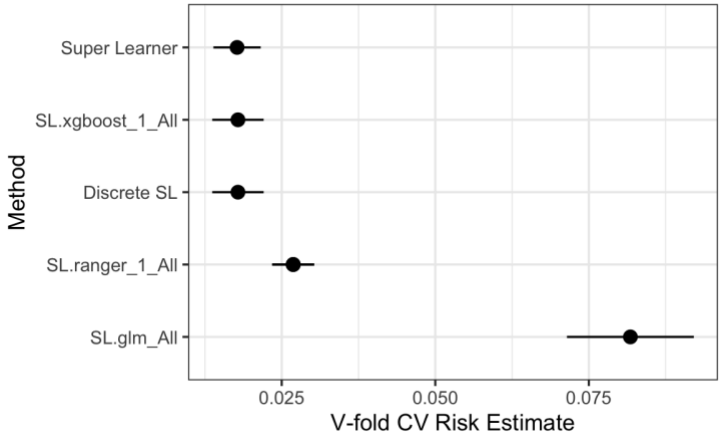
Recall = 25%
Precision = 7.547%
Minority Class = 1.94%

Note: Optimal parameters eta = 0.3, max tree depth = 3, min child weight = 5, subsample = 1, column subsample by tree = 0.5, column subsample by node = 0.75, gamma = 0, lambda = 0.01, alpha = 0.1

Stacked Model

Figure 4

Cross Validated Risk Estimates for Stacked Stacked Learner and Input Models



Note: Models sorted from lowest to highest risk estimate.

Table 11

Stacked Learner Coefficients and Risk

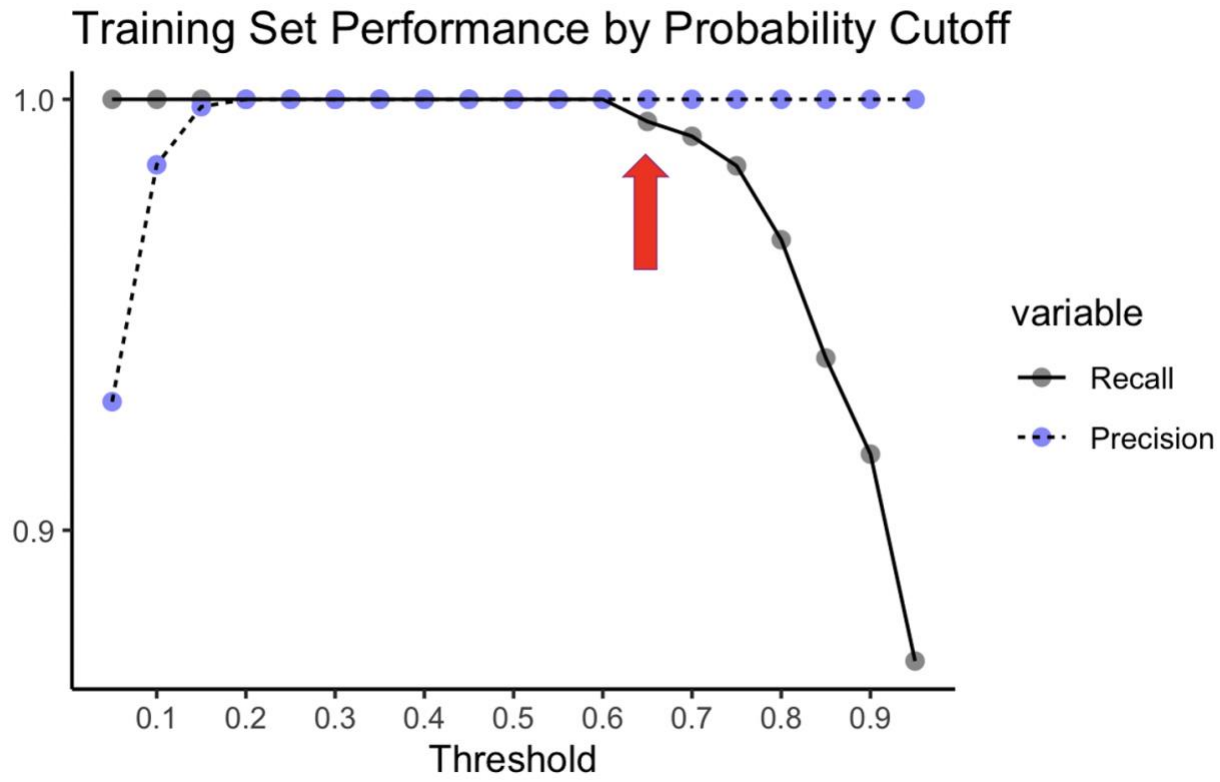
	Stacked Leaner Estimate	
	Risk	Coefficient
Precision Random Forest	0.02652	0.000
Recall Random Forest	0.02654	0.000
Weighted Sum Random Forest	0.02641	0.250
Boosted Tree Model	0.01746	0.750
Logistic Regression	0.08324	0.000

Table 12
Stacked Learner Probability Threshold 0.5
Train Set with SMOTE Performance

		Predicted Class	
		Surgery	No Surgery
Actual Class	Surgery	583	0
	No Surgery	0	2120

Recall = 100%
Precision = 100%

Figure 5
Training Set Probability Threshold's Effect on Recall and Sensitivity



Note: Arrow pointing to first decrease in recall at threshold value of 0.65. Recall still above 99.9%.

Table 13
Stacked Learner Probability Threshold 0.5
Test Set Performance

		Predicted Class	
		Surgery	No Surgery
Actual Class	Surgery	5	11
	No Surgery	50	759

Recall = 31.25%
Precision = 9.806%
Minority Class = 1.94%

Note: Predicted probabilities above 0.5 predicted to be associated with UCL injury.

Table 14
Stacked Learner Probability Threshold 0.65
Test Set Performance

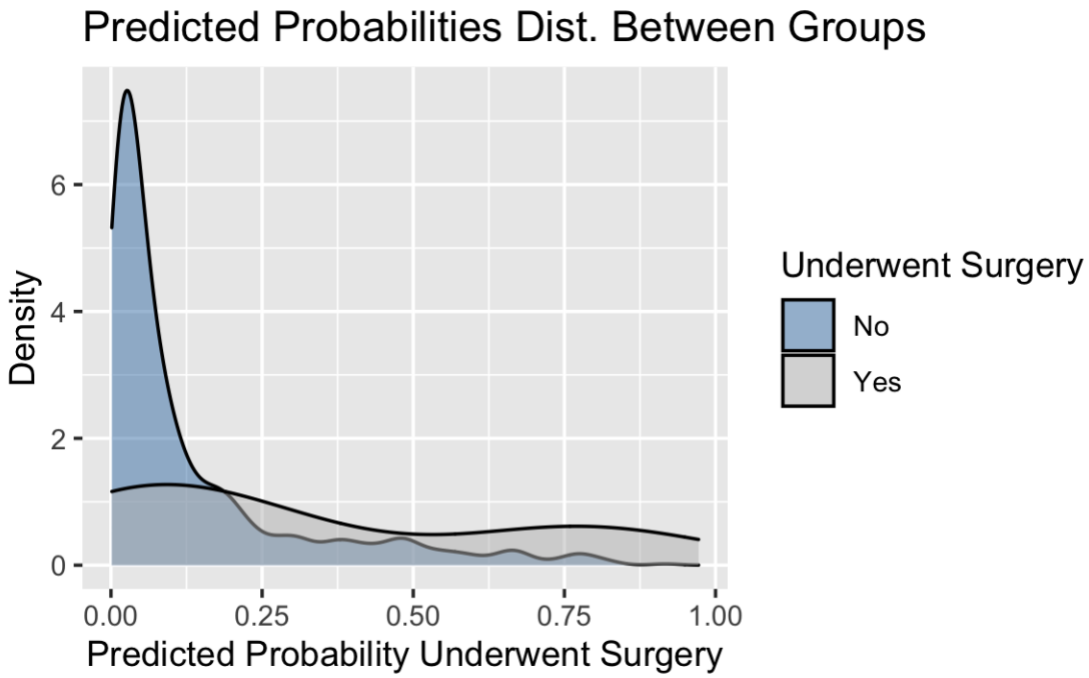
		Predicted Class	
		Surgery	No Surgery
Actual Class	Surgery	5	11
	No Surgery	27	782

Recall = 31.25%
Precision = 15.63%
Minority Class = 1.94%

Note: Predicted probabilities above 0.65 predicted to be associated with UCL injury.

Figure 6

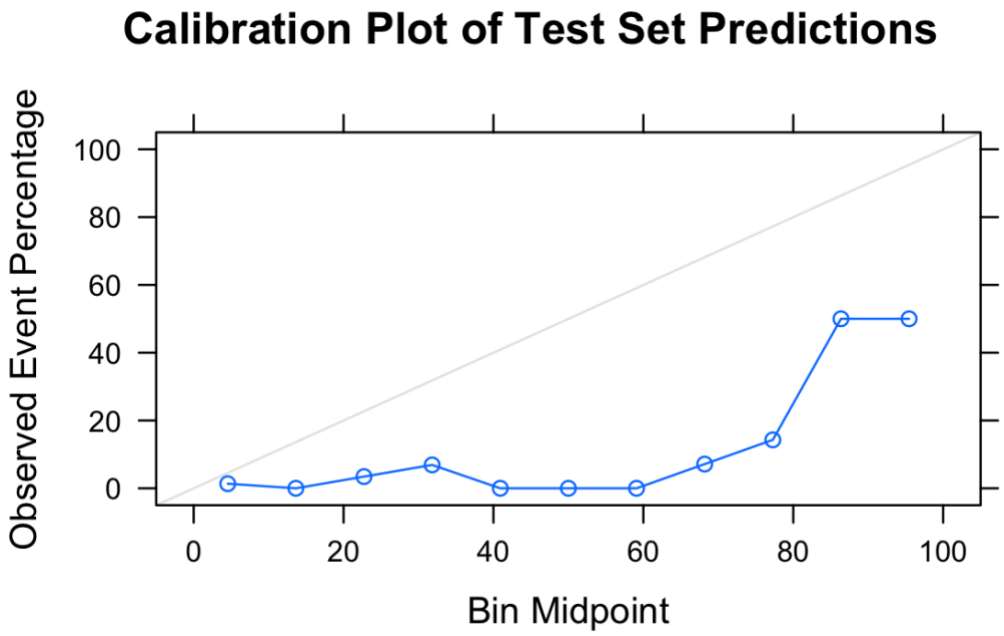
Distribution of Test Set Predicted Probabilities by Actual Class



Note: Grey region well above blue for high probabilities. Plot created using *ggplot2* package in R (Wickham, 2016)

Figure 7

Calibration of Test Set Predicted Probabilities



Note: Plot created using *caret* package (Kuhn, 2020)

Table 15
Model Comparison Test Set Performance

	Evaluation Metric	
	Recall	Precision
Precision Random Forest	37.50%	2.941%
Recall Random Forest	100%	1.988%
Weighted Sum Random Forest	68.75%	2.095%
Logistic Regression	18.75%	2.1127%
Boosted Model	25.00%	7.547%
Stacked Learner 0.5 Threshold	31.25%	9.806%
Stacked Learner 0.65 Threshold	31.25%	15.63%

Note: Precision around 1.94% denotes random guessing. (Minority class percentage)

July 31st Cutoff Sensitivity Analysis

Table 16
Stacked Learner Probability Threshold 0.5
Test Set with July 31st Cutoff Performance

		Predicted Class	
		Surgery	No Surgery
Actual Class	Surgery	5	17
	No Surgery	50	753

Recall = 22.727%
Precision = 9.091%
Minority Class = 2.67%

Note: This data does not have the same dimension as a different year cutoff was used. This test set would be the test set if the March 31st cutoff was instead July 31st.

Table 17
Stacked Learner Probability Threshold 0.65
Test Set with July 31st Cutoff Performance

		Predicted Class	
		Surgery	No Surgery
Actual Class	Surgery	4	18
	No Surgery	28	775

Recall = 18.18%
Precision = 12.50%
Minority Class = 2.67%

Note: This data does not have the same dimension as a different year cutoff was used. This test set would be the test set if the March 31st cutoff was instead July 31st.

Discussion

The UMAP visualization illustrates what appears to be a random spread of individual seasons that are associated with UCL injury. Without any distinct clusters, a clustering or outlier detection algorithm did not seem appropriate for this analysis. The UMAP also displays a different global structure for the training and test sets. The lack of similarity in global structure between the two sets is hypothesized to be a reason for the large difference in training and test performance throughout the analysis. The UMAP visualization of the training set that underwent SMOTE, shows how SMOTE changes the structure of the data entirely. The SMOTE appears to create a very homogenous dataset and may be another reason for the large differences in training and test performance.

The random forest models had issues with overfitting. Although all three optimized models had success on the over-sampled training set, none of the models were able to obtain a test set precision more than a single percentage point above the minority class percentage. With a precision extremely similar to the percentage of values within the minority class, the random forest models do not have any generalizable predictive power on their own. Arguably, the

random forest optimized on precision was the most effective model with a recall of 37.5% and precision just above the minority class percentage (2.941%).

The logistic regression arguably had the worst performance of all models. The UMAP suggested this may occur as there did not appear to be a linear decision boundary within the data. The logistic regression preformed well on the over-sampled training set but had low recall proportionally to other models (18.25%) as well as a precision slightly above what would be expected from random guessing (2.11%).

The boosted tree model was the only model with predictive power on its own. The boosted tree model was able to achieve perfect performance on the over-sampled training set, yet like the rest of the models displayed large differences in training and test set performance. Even with penalization, the boosted model overfits the SMOTE set. When used to predict the test set, the boosted model was able to recall 25% of the 16 seasons associated with UCL injury while maintaining a precision of 7.547%. Nearly four times more accurate than a random guess, the boosted model is able to catch some generalizable signal within the data.

When all models were combined into one model ensemble, the stacked learned had non-zero coefficients for only the random forest optimized on the weighted sum and the boosted tree model. The stacked model coefficients suggest it was composed of about 75% boosted model and 25% random forest optimized on the weighted sum. In the 10-fold cross validation risk estimates, the stacked learner narrowly outperformed the individual models as seen in Figure 4. The 10-fold cross validation risk estimates showed there to be a very small difference between the stacked learner and the boosted tree model, but large gains over the random forests and logistic regression.

When fit to the over-sampled training set, the stacked learner was able to reach perfect performance. Figure 5 shows the stacked learner is able to get perfect recall and precision on the over-sampled training set until a probability threshold of 0.65 when the recall drops to 99.4%. The 0.06% decrease in recall was seen as negligible when compared to the potential for even more precise test set results, thus 0.65 was chosen over 0.60. A plot of the test set precision and recall relative to the probability threshold can be found in the appendix and suggests a value above 0.65 but below 0.70 to give the optimal tradeoff in performance.

Table 13 shows the test set predictions for the stacked learner when using a probability threshold of 0.5. With a threshold of 0.5, the stacked learner achieved a recall of 31.25% while maintain a precision of 9.091%. When the threshold is set to 0.65, the stacked learner maintained the recall of 31.25% while boosting the precision to 15.63%.

By identifying nearly one third of the observations associated with UCL injury in the test set while maintaining a precision more than eight times more accurate than a random guess, the stacked learner with the optimal threshold of 0.65 is effective in a practical setting. When looking at Figure 6, the superimposed density plots of the stacked learner's test set predicted probabilities of the UCL injury and non-injury groups, there are distinguishable differences. The stacked learner predicts most of the seasons not associated with UCL injury to very low probabilities. Although not perfect, the stacked learner's predicted probabilities for the seasons associated with UCL injury appear to be higher than the predictions for seasons not associated to UCL injury. The calibration plot in Figure 7 shows similar findings. The calibration plot shows a visual increase in observed event percentage for bins associated with higher predicted probabilities.

When using the stacked learner to predict the test set that would have been created if the cutoff date for the associated UCL injury were to be change to July 31st, we see small drops in recall but relatively no change in precision. Although the precision is relatively the same, there is a change in the precision relative to the percentage of observations in the minority class. The stacked learner is able to recall around one fifth of the observations associated with UCL injury while maintaining a precision roughly four times greater than a random guess when averaging the results between the 0.5 and 0.65 thresholds. The stacked model does not completely lose its predictive power when predicting the July 31st test set, suggesting that the model is not solely classifying UCL injury based on observations with lower amounts of playing time. An argument can be made that the only reason the model is losing some of its recall and relative precision is that when using a later cutoff, the statistics and Statcast information are less relevant for some of the observations associated with UCL injury. In a data problem that is already associated with a low signal to noise ratio, using data from further in the past may be amplifying this issue.

Conclusion

Machine learning methods, especially tree-based models show promise in the space of identifying at-risk individuals for UCL injury from single season statistics and Statcast variables. Specifically, a model ensemble that is able to combine the unique predictive strengths of multiple tree-based models into one stacked learner resulted in the strongest performance. Although the results and predictions are not perfect, the cost of a false positive in this specific use case is very low. With an ability to correctly recall 31.25% of individual seasons associated with UCL injury and a precision eight times more accurate than a random guess, the stacked learner predictions can be used as an initial screening practice.

In practice, a team, doctor, or the league as a whole, could gather these readily available statistics with little issue. It is not too farfetched to assume the teams, doctors and the league already have all of the information needed on file since most of the information used in this analysis is published on the MLB's website. If the league was to pass every individual pitcher through the model, ideally sometime during the off-season or spring training, the research suggests this stacked learner model can help to identify at risk individuals and add to the conversation of which individuals should undergo preventative measures. Keep in mind, the model by no means should be the only component used in the decision. Any method of identifying MLB pitchers at risk of UCL injury has benefits at multiple levels. Identifying these individuals can save MLB players' careers, increase a team's return on investment, and increase the overall fan experience as fans want to watch their favorite players play.

When interpreting the results there are a few things the reader should keep in mind. From the data available, there was no way to identify when the UCL injury occurred and thus surgery date was used. When using surgery date, there is no way to know if the duration between sustaining the injury and the surgery date is consistent across individuals. Therefore, the cutoff date assigned as March 31st is inconsistent relative to when the injury occurred for individuals. The cutoff date was also an arbitrary decision and can be a topic of debate. The March 31st date was a way to obtaining the most recent information, but also offered risks.

In the future, this research is a starting point for using machine learning in combination with these new Statcast statistics to create models with a more feasible use case than research prior in the field of UCL injury for MLB pitchers. Future improvements will include either more data, as the population of individuals was strictly limited due to Statcast variables being release in 2015 or a way to over-sample the data that does not lead to such overfitting in the machine

learning tree-based spaces. Variable selection methods will also potentially improve results. LASSO penalized regression was considered for variable selection, but not deemed appropriate due to low test set performance seen in the linear space. A variable selection method that is more appropriate for the tree-based space may be helpful. Neural networks were also considered, and the results of a multi-layer perceptron (MLP) can be seen in the appendix. As more and more seasons of Statcast information become available, this multilayer perception may gain more feasibility. It is hypothesized the MLP struggled as a result of the ratio between samples size and minority class percentage.

An interesting continuation of this research would be to evaluate if large number of false positives stay true in the long-term. Some of these false positives may stem as a result of the how the data was structured with an absolute cutoff date. There is a chance the model may be identifying individuals who undergo UCL surgery, but that surgery is associated with a season later in the future. In that case, the prediction would be classified as a false positive, but in practice the positive prediction would be useful. This more downstream look was outside of the scope of this analysis. In the future, with more data methods like a survival analysis will be of use. Survival analysis was not used due to the extremely high number of parameters in those models.

References

Bell Tanner, (2020). Player ID Map Excel Spreadsheet, SmartFantasyBaseball. https://www.smartfantasybaseball.com/tools/

Brown, M. (2019). MLB Sees Record $10.7 Billion In Revenues For 2019. *Forbes. https://www. forbes. com/sites/maurybrown/2019/12/21/mlb-sees-record-107-billion-inrevenues-for-2019/(accessed May 08, 2020)*.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321-357.

Chen, F. S., Rokito, A. S., & Jobe, F. W. (2001). Medial elbow problems in the overhead-throwing athlete. *JAAOS-Journal of the American Academy of Orthopaedic Surgeons*, *9*(2), 99-113.

Conte, S. A., Fleisig, G. S., Dines, J. S., Wilk, K. E., Aune, K. T., Patterson-Flynn, N., & ElAttrache, N. (2015). Prevalence of Ulnar Collateral Ligament Surgery in Professional Baseball Players. The American Journal of Sports Medicine, 43(7), 1764–1769. https://doi.org/10.1177/0363546515580792

Erickson, B. J., Harris, J. D., Tetreault, M., Bush-Joseph, C., Cohen, M., & Romeo, A. A. (2014). Is Tommy John surgery performed more frequently in Major League Baseball pitchers from warm weather areas?. *Orthopaedic journal of sports medicine*, *2*(10), 2325967114553916.

Greenwell, B., Boehmke, B., & Gray, B., (2020). vip: Variable Importance Plots. R package version 0.2.2. https://CRAN.R-project.org/package=vip

Japkowicz, N. (2000). The Class Imbalance Problem: Significance and Strategies. In Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI'2000): Special Track on Inductive Learning Las Vegas, Nevada.

Jouanne-Diedrich, H., (2017). OneR: One Rule Machine Learning Classification Algorithm with Enhancements. R package version 2.2. https://CRAN.R-project.org/package=OneR

Keller, R. A., Marshall, N. E., Guest, J. M., Okoroha, K. R., Jung, E. K., & Moutzouros, V. (2016). Major League Baseball pitch velocity and pitch type associated with risk of ulnar collateral ligament injury. *Journal of Shoulder and Elbow Surgery*, *25*(4), 671-675.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kuhn, M., (2020). caret: Classification and Regression Training. R package version 6.0-86. https://CRAN.R-project.org/package=caret

Kuhn, M., & Wickham, H., (2020). recipes: Preprocessing Tools to Create Design Matrices. R package version 0.1.14. https://CRAN.R-project.org/package=recipes

Lahman, S. (2020) Lahman's Baseball Database, 1871-2019, 2019 version, http://www.seanlahman. com/baseball-archive/statistics/

Major League Baseball. (2020, September 23). *Custome Leaderboard*. Baseball Savant. https://baseballsavant.mlb.com/leaderboard/custom?year=2019&type=pitcher&filter=&sort=4&sortDir=asc&min=q&selections=xba,xslg,xwoba,xobp,xiso,exit_velocity_avg,launch_angle_avg,barrel_batted_rate,&chart=false&x=xba&y=xba&r=no&chartType=beeswarm

McInnes, L., Healy, J., & Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (2012, July). How many trees in a random forest?. In *International workshop on machine learning and data mining in pattern recognition* (pp. 154-168). Springer, Berlin, Heidelberg.

Polley, E., LeDell, E., Kennedy, C., & Laan, M., (2019). SuperLearner: Super Learner Prediction. R package version 2.0-26. https://CRAN.Rproject.org/package=SuperLearner

R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rhinehart, E. (2020). Tommy John Surgery: Potential Risk Factors and Causes in Major League Pitchers.

Roegele, J. (2018, April 18). 10 Interesting Facts About Tommy John Surgery Hardball Times. https://tht.fangraphs.com/ten-interesting-facts-about-tommy-john-surgery/

Roegele, J. (2020, September 23). Tommu John Surgery List MLB Player Analys. https://docs.google.com/spreadsheets/d/1gQujXQQGOVNaiuwSN680Hq-FDVsCwvN-3AazykOBON0/edit#gid=0

RStudio Team (2020). RStudio: Integrated Development for R. RStudio, PBC, Boston, MA URL http://www.rstudio.com/.

Selley, R. S., Portney, D. A., Lawton, C. D., Shockley, M. D., Christian, R. A., Saltzman, M. D., & Hsu, W. K. (2019). Advanced baseball metrics indicate significant decline in MLB pitcher value after Tommy John surgery. *Orthopedics*, *42*(6), 349-354.

Sports Reference LLC (2020). 2019 MLB Standard Pitching. Baseball-Reference.com - Major League Statistics and Information. https://www.baseball-reference.com/leagues/MLB/2019-standard-pitching.shtml

Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, Mu Li, Junyuan Xie, Min Lin, Yifeng Geng & Yutian Li (2020). xgboost: Extreme Gradient Boosting. R package version 1.2.0.1. https://CRAN.R-project.org/package=xgboost

Whiteside, D., Martini, D. N., Lepley, A. S., Zernicke, R. F., & Goulet, G. C. (2016). Predictors of ulnar collateral ligament reconstruction in Major League Baseball pitchers. *The American journal of sports medicine*, *44*(9), 2202-2209.

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

Wickham, H., François, R., Henry, L. and Müller, K. (2020). dplyr: A Grammar of Data Manipulation. R package version 1.0.2. https://CRAN.R-project.org/package=dplyr

Wright, M., Ziegler, A., (2017). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. Journal of Statistical Software, 77(1), 1-17. doi:10.18637/jss.v077.i01

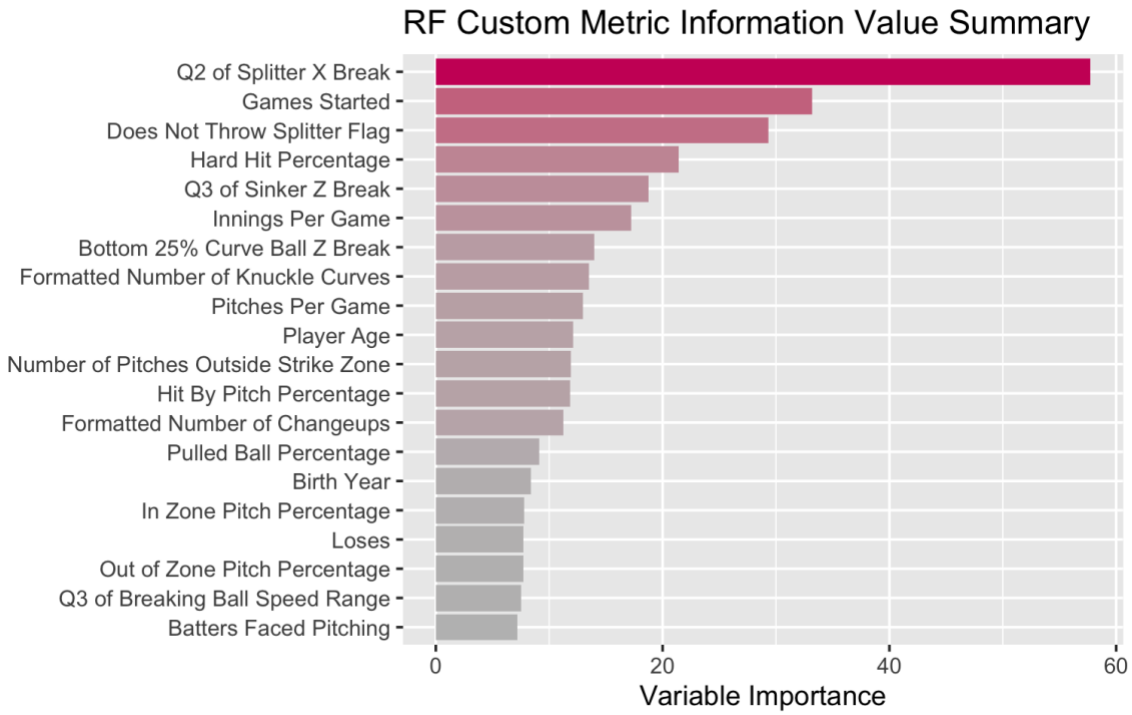
Appendix

Variable Importance

Variable importance plots are displayed for the two models that had coefficients greater than zero in the stacked learner.

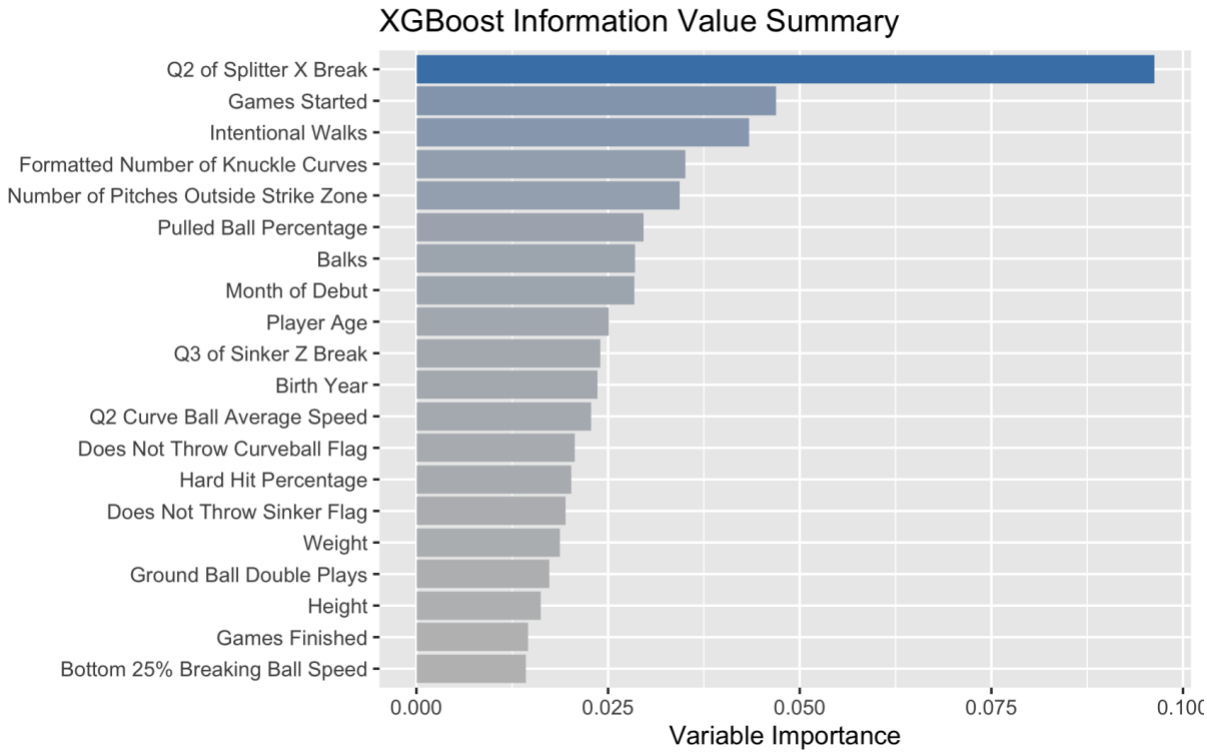
Figure 8

Variable Importance for Optimal Random Forest Using Weighted Sum



Note: Variable importance in the random forest is calculated in terms of average impurity decreases across nodes (Wright and Ziegler, 2017). Correlated variables may hinder certain importance.

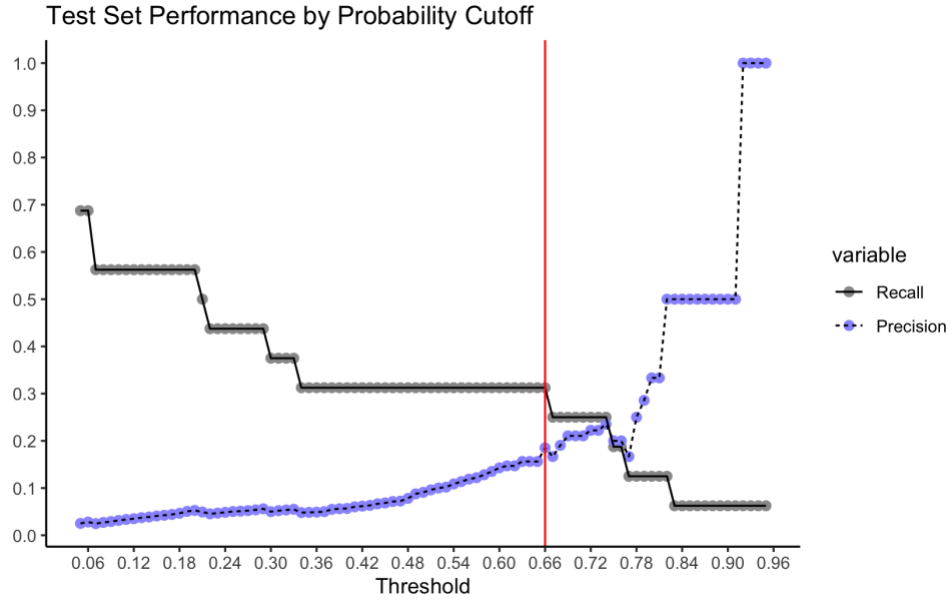
Figure 9
Variable Importance for Optimal Boosted Tree Model



Note: Variable importance in the boosted tree is calculated in terms of average impurity decreases across trees. Obtained using the *vip( )* function in the *vip* package in R (Greenwall et. al., 2020). Correlated variables may hinder certain importance.

Figure 10

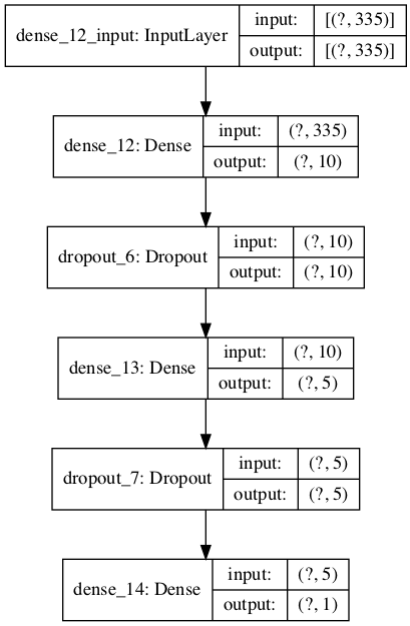
Training Set Probability Threshold's Effect on Recall and Sensitivity



Note: Redline denotes 0.66 optimal cutoff point test set.

Figure 11

Neural Network Architecture



Note: All activation functions are relu excluding final which is sigmoid. Optimizer is ADAM. Use binary cross entropy.

Table 18
Neural Network
Test Set Performance

		Predicted Class	
		Surgery	No Surgery
Actual Class	Surgery	0	16
	No Surgery	0	809

Recall = 0%
Precision = NA%
Minority Class = 2.67%

Note: Neural Network continually selects only majority class in test set.