Forecasting Outcomes of Major League Baseball Games Using Machine Learning

EAS 499 Senior Capstone Thesis

University of Pennsylvania

An undergraduate thesis submitted in partial fulfillment of the requirements for the Bachelor of Applied Science in Computer and Information Science

Andrew Y. Cui[1]

Thesis Advisor: Dr. Shane T. Jensen[2]

Faculty Advisor: Dr. Lyle H. Ungar[3]

April 29, 2020

---

[1] Candidate for Bachelor of Applied Science, Computer and Information Science. Email: andrewc@seas.upenn.edu

[2] Professor of Statistics. Email: stjensen@wharton.upenn.edu

[3] Professor of Computer and Information Science. Email: ungar@cis.upenn.edu

Abstract

When two Major League Baseball (MLB) teams meet, what factors determine who will win? This apparently simple classification question, despite the gigabytes of detailed baseball data, has yet to be answered. Published public-domain work, applying models ranging from one-line Bayes classifiers to multi-layer neural nets, have been limited to classification accuracies of 55-60%. In this paper, we tackle this problem through a mix of new feature engineering and machine learning modeling. Using granular game-level data from Retrosheet and Sean Lahman's Baseball Database, we construct detailed game-by-game covariates for individual observations that we believe explain greater variance than the aggregate numbers used by existing literature. Our final model is a regularized logistic regression elastic net, with key features being percentage differences in on-base percentage (OBP), rest days, isolated power (ISO), and average baserunners allowed (WHIP) between our home and away teams.

Trained on the 2001-2015 seasons and tested on over 9,700 games during the 2016-2019 seasons, our model achieves a classification accuracy of 61.77% and AUC of 0.6706, exceeding or matching the strongest findings in the existing literature, and stronger results peaking at 64-65% accuracy when evaluated monthly. Further refinements in game-level pitcher covariates, team statistics such as errors, and Bayesian hyperparameter optimization are likely to strengthen our overall predictive accuracy and model lift. With continued improvement, this model has applications for baseball team strategy, sports predictions, and personal fan interest.

Keywords: sports analytics, baseball, machine learning, feature engineering, computer science

Note: All code associated with this thesis has been uploaded to GitHub and has been solely done for the purposes of EAS 499, Senior Capstone Thesis, Spring 2020. The repository can be found at https://github.com/andrew-cui/mlb-game-prediction

Table of Contents

1. Introduction and History

Every year (at least between 1995 and 2019), 30 Major League Baseball teams each play 162 games. Official game statistics have been kept in America since the mid-1800s, when Henry Chadwick developed systems for keeping records, modernizing the game towards its present-day form (Schiff, 2008). Player records evolved into statistics such as home runs, batting average and slugging percentage, which allowed fans and team managers to differentiate superior performance within the team of players. Even today, historically famous baseball achievements are interlocked with their quantitative measures: Babe Ruth's 14-WAR 59-home run monster; Ted Williams being the last man to hit .400 (.406, specifically); Steve Carlton's 30 complete games; Bob Gibson carrying an live-ball era low 1.12 season earned-run average (Tylicki, 2011).[4]

The inherent variance in baseball games, such as the players on the field, game conditions, and momentum, make it challenging to forecast the results of individual games. However, the sports analytics world provides us with multiple approaches to forecast games. The most basic models use aggregate statistics to compute values such as Pythagorean expected W-L percentage (Heumann, 2016), evaluating a team over the course of a season. On the other hand, the most complex models (and likely the proprietary ones used by teams) can update win posterior forecasts based on singular events in the course of a game. For instance, consider Fangraphs' win probability chart during Game 7 of the 2019 World Series, which evaluates win probability in greater detail or speed than the human brain can catch up with:

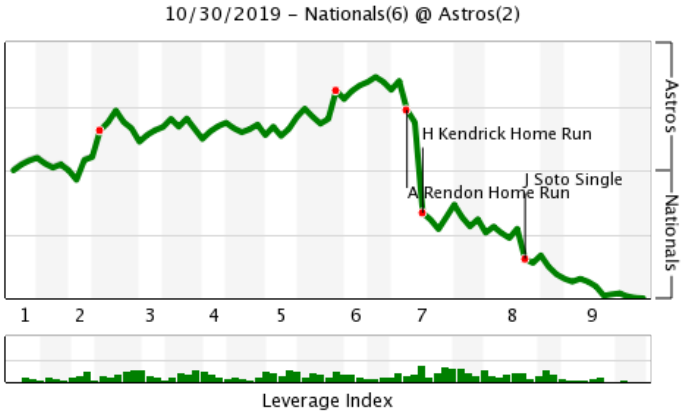

Figure 1: 2019 World Series, Game 7, Win Probability Graph (Fangraphs, n.d.)[5]

[4] Seasons ranked in the article as follows: 1. Ruth (1942), 5. Gibson (1968), 17. Williams (1941), 19. Carlton (1972)

[5] Y-axis ranges from 0 (Washington Nationals win) to 100% (Houston Astros win). We see a bias in favor of the home-team Astros at the beginning of the game, which grows as they take the lead, but rapidly vanishes when the Nationals hit a go-ahead home run (swinging from 68.7% Astros win to 34%), and progressing similarly as the game's, and Houston's, time runs out.

Due to the quantity of widely available baseball data, we are curious to create a model that sits in the middle of these extremes: machine learning-based approaches that incorporate game-specific data (such as roster ability or pitching) but provide prior predictions of which team will win a game.

In this paper, we seek to understand the game of baseball better through applying machine learning models, as the culmination of four years of study in computer science, engineering, and data science methodology. Through analyzing the existing literature about baseball prediction and machine learning models and using data from historical seasons (post-2000), we seek to construct a binary classifier that can predict, using only data available before a game is played, which of the two teams is more likely to win. In addition to understanding the modeling approaches, we apply practical data wrangling and feature engineering skills to create game-level covariates for each observation, which allow us to identify team strengths, hopefully in turn explaining part of the variance in baseball game results.

Note: we abbreviate many MLB statistics and terms in this paper. A full glossary is found in **Appendix 1**.

2. Literature Review

We begin by understanding the body of literature available for baseball analytics and specifically prediction of baseball game results. In order of increasing statistical and computational complexity, we discuss, in the context of game result classification (win/loss): aggregate evaluations of team performance such as Pythagorean W-L expectation; machine learning (ML) approaches including logistic regression and generalized linear models, tree-based classifiers such as random forests, support vector machines (SVMs/SVCs) and gradient-boosting ensemble classifiers; Bayesian and Markov-Chain statistics models; and artificial neural nets (ANNs). Several of these methods can be applied to tracking in-game events as well, achieving instantaneous predictions. While we mention examples of these latter models, they are meant to highlight the machine-learning techniques rather than focus on the sports-analytics question at hand, since our goal is to understand these methodologies as we build a model to predict games.

2.1 Aggregate predictions for a baseball season

Baseball's analytical revolution gained velocity and salience notably during the early 2000s, when the 2002 Oakland Athletics used statistical models for on-base percentage and expected run-scoring to discover undervalued players, winning 103 games[6] and qualifying for the playoffs with one of the lowest payrolls and least ability to sign top players (Brown et al., 2017). Since then, a spectrum of ways to predict baseball outcomes, with varying computational complexities, have emerged. On the basic end, we have backwards-looking forecasts based on aggregate team statistics. The most prominent is Bill James's Pythagorean W-L expectation; arguing that successful teams tend to outscore opponents, this computes an expected win percent based on the ratio of run-scoring and prevention (Heumann, 2016) as follows:[7]

$$P\% = \frac{RS^2}{RS^2 + RA^2}$$

This approach is intuitive to understand and compute and can be improved slightly through tuning the exponent hyperparameter, which through back-testing provides an RMSE-minimizing adjusted exponent of 1.83 instead of 2 (Davenport & Woolner, 1999; Heumann, 2016). More complicated analytical approaches include Davenport and Woolner (1999), who

[6] 103 wins in a season is no small feat; since MLB moved to the modern 162-game season, only 28 teams out of 1520 have won 103+ games, equivalent to a 98th percentile (Baseball Almanac, n.d.; Lokker, 2019). See **Appendix 2**.
[7] Note: many of the equations have been formatted externally in LaTeX

apply Poisson models to forecast win ratios, and Heumann (2016), who tests weighted pairwise Pythagorean W-Ls, factoring in the number of times two teams have played against each other. These refinements improve model flexibility, which means the model is able to adapt to a greater variety of potential function surfaces that map the input (runs scored and allowed) to the response (win-loss ratio), improving accuracy at a cost of interpretability (James et al., 2013). However, the models provide only incremental RMSE benefits: with the exponent set to 1.83, Heumann (2016) improves just 1.3% over James and 1.0% over Davenport and Woolner -- while these enhancements have solid theoretical approaches (from statistics, game theory, and baseball), the improvement in predictive accuracy is not meaningful enough to justify the slight loss of interpretability, which is why James' simple formula remains relevant.

Individual player forecasts are also used to project MLB seasons in advance, unlike the Pythagorean approach that requires existing data from the season. For instance, PECOTA[8] is a nearest-neighbor based proprietary method that identifies similar player attributes from baseball history, forecasting future performance through a combination of the comparable players and the individual's prior seasons (Koseler & Stephan, 2017; Druschel, 2016). Fangraphs' Steamer and ZiPS methods are alternative regressions of past performance, league performance, and advanced statistics such as BABIP, which are intended to isolate the signal from a player's own abilities in order to project their performance in the season (Druschel, 2016). However, while these projections have credibility -- Bailey et al. (2020) use PECOTA and batted-ball statistics to strengthen batting average predictions -- and are used in the industry widely, they still apply a Pythagorean approach to forecasting team results, which limits its usefulness for game-to-game predictions and can misvalue teams before a season starts despite having predictions for each player on the roster (Miller & Rogers, 2019).

The Pythagorean W-L is limited by the aggregated data (runs scored/allowed over multiple games combined) used in these metrics; the models lack the flexibility and parameters necessary to capture micro-level variations (in this case, individual games) that are important in further improving prediction accuracy, and as a result are mean-reverting by smoothing out variations in the underlying events/games (Park & Garcia, 1994). For instance, a team that wins many close games will be naturally mathematically overrated relative to their Pythagorean W-L, ignoring potential fundamental characteristics (such as good situational, or "clutch," pitching/hitting that allows teams to gain the slight edge) and increasing prediction error. Nonetheless, these approaches are important as the underlying strategy that our more complex machine-learning approaches build up from and highlight the importance of explanatory power -- particularly in communicating to fans and non-quantitative sports executives.

---

[8] "Player Empirical Comparison and Optimization Test Algorithm"

2.2 Game-level prediction strategies

Machine learning techniques are more mathematically complex and able to fit variations in baseball game outcomes better. Baseball's wealth of data makes the computational needs of ML models feasible, such as pitch-by-pitch data collected in games today allows for researchers to classify pitches thrown and forecast pitcher ERAs (Koseler & Stephan, 2017). Here, we want to predict the winner of games in advance. There are two ways to do this: first, by binary classification to identify the winner of a game based on features; second, regression-based analysis such as tracking the "performance" of a team over time and forecasting predictions, like Elo ratings (Koseler & Stephan 2017; Valero, 2016; Elo, 1978). The latter models, such as continuous responses like Elo, can be incorporated as independent variables as well.

In this case, we are primarily interested in understanding which team wins a baseball game (ignoring result components like the final score), which makes this a binary classification problem that considers the impact of features that can explain variance in the result of games. We limit the discussion of literature primarily to understanding binary classification of baseball games and series below; however, there still exists a breadth of machine learning applications to this problem, between sources from academic literature as well by sports writers. We outline the prominent examples below.

Valero (2016) compares four machine learning methods -- k-Nearest-Neighbors classifiers (KNN), support vector machines/classifiers (SVM/SVC), tree-based models (specifically, decision trees), and artificial neural networks (ANN) -- using prediction accuracy (proportion of accurately predicted games divided by the total games) to evaluate them, using aggregate features such as Pythagorean expectation, win percentage, OBP, and SLG percentage season-wide differences between the teams. Valero's features line up with research by Stick (2005) found through regression analysis that OPS and WHIP accounted for 26% of variance in all baseball productivity indexes alone; although Stick's findings were not in the context of predicting games specifically, they demonstrate the importance of factors like OPS. In Valero's (2016) analysis, the KNN model performed worst in game result classification, followed by pruned (to reduce overfitting) classification trees and artificial neural nets at 58% accuracy, and the SVM model had 59% accuracy.

These analyses outperform naïve baselines such as a guaranteed "home-team advantage" -- where the home team is always predicted to win -- that scored a 53.01% accuracy in the 2016 season (Elfrink & Bhulai, 2018). In another study, SVMs with 10-fold CV and varying kernel polynomial specifications (linear, quadratic, and cubic), predicted major league baseball playoff championship outcomes at similar accuracies of 57.1-67.9% (Tolbert & Trafalis, 2016). Tolbert and Trafalis (2016) likely selected SVMs due to their flexibility and predictive accuracy, as Valero (2016) found, while still maintaining some interpretability -- such as identifying which

attributes were most important in each model, even if without evaluations of how they influenced chances of winning. Despite the similarity in measures used, in comparing these models' accuracies, we must note that while Valero (2016) evaluated outcomes of individual games, Tolbert and Trafalis (2016) were predicting winners in the playoffs -- where "winning" was defined as winning the majority of games in a best-of-7 series in all cases, and variance in individual games is smoothed out more over a series.

Seeing the SVM and ANN have little difference, despite the latter's complexity (Valero, 2016), suggests an upper bound in the ability to predict baseball games from potentially two factors: first, quantitatively measured improvements in competitive balance through luxury taxes that reduce the separation between teams' skills, making contests more likely to go either way (Ajilore & Hendrickson, 2007); and second, variance inherent to sporting events since they depend on players' performances in a given day.

Meanwhile, Elfrink and Bhulai (2018) add less flexible methods such generalized linear regression models and logistic regression in addition to a tree-based random forest and gradient boosting, but despite targeting the same classification problem, have similar, but poorer, model classification accuracies of 53.75% to 55.52% compared to Valero (2016)'s 55.98% to 58.92%.[9] Comparably, Pharr (2019) used gradient boosting methods through LightGBM and XGBoost to find results of 59.3% and 60.7% respectively, even though he only ran a 30-day back test, where his strongest predictions occurred when only considering consensus models (where all models agree). Like FiveThirtyEight (2015), he uses an Elo-esque covariate, team and pitcher rest, and pitcher statistics; however, he also adds match differences such as the spread of errors between the home and away team (Pharr, 2019).

Jia et al. (2013) used adaptive boosting or AdaBoost, a similar boosting algorithm, in addition to logistic regressions and random forests, finding aggregate accuracies of around 56-60%. Like Pharr (2019), they also include team-level covariates such as team RBIs, batting average, and ERA; however, they choose to include the team's existing win percentage, similar to Valero (2016), whereas Pharr excludes it, and use aggregate season information instead of rolling statistics, which devalues "team momentum" (Jia et al., 2003). Interestingly, they chose to fit models over each month to test, arguing that baseball games are harder to forecast at the beginning of the season when there is less information about the team's skill, finding average accuracies of 52.7-57.6% in April and May, but of 59.7-64.1% by August and September (Jia et al., 2003).

[9] Both Valero (2016) and Elfrink and Bhulai (2018) consider results from the question as a regression problem as well. However, if the result is to predict which a team will win rather than a continuous response (such as the score), I see it more appropriate to use the binary classification situation.

Gradient boosting methods and random forests can sacrifice explanatory for predictive power; as is discussed below in MODELS, logistic regressions allow for log-odds coefficients that relate observable characteristics (ex: home team; team ERA; ERA difference) to their impact on the forecasted probability of winning that specific game observation, while decision trees' partitioning based on rules about the dataset characteristics (ex: ERA difference < 0.2) work like the way humans classify objects and events (James et al., 2013). Subsequently, these models are quantifiable and easier to explain to non-computer scientists, which is valuable when they need to be accepted for more than just quantitative measures.

Anecdotally, consider managerial decisions, like for Oakland manager Billy Beane as depicted in *Moneyball*, who needed to win the faith of his scouts and team ownership in order to trade for new players (Miller, 2011); lacking interpretable models would dampen his confidence and persuasiveness, rendering the models eventually useless in practical context, even if they are theoretically sound and accurate. Although predictive power is incredibly important, we must recognize that baseball prediction models may be used in contexts that demand more than just numerical excellence.

Bayesian statistics methods extend the basic conditional probability law of Bayes' Rule and treat parameters as random variables from probability distributions, which demands computational intensity but allows the resulting models to reduce assumptions about the data while making direct probability statements about parameter intervals and future sample predictions (Jensen, 2020). Soicher (2019) uses a simple Bayes classifier based on home vs. away-team status, which achieved a 55% accuracy despite its lack of features and outperformed rudimentary machine learning approaches, particularly at the beginning of the season lesser prior data hurts ML algorithm accuracy, as Jia et al. (2003) show.

Meanwhile, Yang and Swartz (2014) implement a hierarchical model comparing relative strength through comparing team records, batting averages and ERAs between opponents; their two-stage model accounts for greater variation in the data, computing season-long win-loss differences to within 2-3 raw percentage points for the majority of teams, but do not report individual game prediction accuracy, making it challenging to compare. Bukiet et al. (1997) apply Markov chain statistics to estimate conditional values of each offensive player to their team, which may be useful in a model that considers the impact of lineup order to game outcomes; however, they do not make this extension on their own.

Across these approaches, we see consistent themes. First, almost all the models focus on game-level features, believing they can help explain variance in the game-to-game results. Second, the game-level predictions emphasize the use of prior data only, to avoid data contamination (Valero, 2016). Finally, most models come in under 65% accuracy, with a baseline around 50-53%, indicating potential limitations in capturing baseball game variance.

2.3 Consensus and averaging methods in ensemble models

Several works (Elfrink & Bhulai, 2018; Valero, 2016; Pharr, 2019) use ensemble classification methods including boosted classifiers and random forests. Since ensembles use multiple individual classifiers, the eventual prediction from the model is often conducted by a majority vote (whichever label is predicted most often) by the individual classifiers (James et al., 2013).

However, for observations that are less obviously classified into either label (in the context of Major League Baseball, games where two teams are of similar skill levels), majority vote may not be the ideal solution. Pharr (2019) found an increase from 59.9% to 61.1% in boosted regression model accuracy when applying consensus instead of averaging ensemble methods, which require all ensemble learners to agree on the prediction (for example, that the home team will win) in order to make a classification. In both bagging and boosting models, averaging or majority-voting methods can hurt classification accuracy (Alzubi et al., 2018), as Pharr observed.

To remedy this, Alzubi et al. (2018) propose "consensus-based combining methods" (CCM) that iteratively adjusts each classifier's weights based on other classifiers until they converge to a consensus decision, and demonstrate superior accuracy of up to 2-5% when tested on standard pattern recognition datasets, like the Iris flower dataset (UCI, n.d.). However, Pharr (2019) combats this by only acting and placing bets on games when his models show consensus with stronger prior probabilities of winning, which makes the 9% of cases without predictions less relevant and omits needing to integrate Alzubi et al. (2018)'s approach.

For the purposes of our analysis, we are interested in understanding the entire set of game predictions across seasons, so the majority or averaging/majority classifications are less important. However, particularly in cases of potential sports bets (not the purpose of this paper, but a potential application for certain readers), mimicking Pharr (2019)'s use of consensus models may prove profitable, in line with considering model lift as we mention in **Discussion** later on.

2.4 Literature summary

In the prior section, we discuss examples of existing publications and their data science methodologies. To avoid making the assumption of prior knowledge about each of these models, but for brevity above, we include brief descriptions about the modeling approaches mentioned, with citations for machine learning and statistics textbooks that carry the mathematical derivations; while these are important to understanding the modeling approaches and conducting the analysis, the overview should be sufficient for comprehending this applied-science

paper. Afterwards, we aggregate information about the models, their strengths/weaknesses, and cited literature in "Comparison of analytical methods." The models discussed are (in order): generalized linear and logistic models (GLM), K-Nearest Neighbors (KNN), tree-based models (and random forests), gradient boosting methods (XGBoost), SVMs, and neural nets (ANN).

Several statistics and machine learning techniques from the field of sports analytics have been presented above but carry their own strengths and limitations. It is important to note the surprisingly limited literature in our problem of prior predictions for MLB games, similarly, noticed by Bunker and Susnjak (2019) who cite only Valero (2016) in a paper of key machine learning approaches in sports result predictions. Although online writers like Soicher (2019) and Pharr (2019) also document model examples, our research question demands further publicized study. It is important to note that many predictive strategies (ex: PECOTA, ZiPS, and any MLB-team-built models) will be proprietary for competitive advantage -- and are likely to be superior to the published works (Druschel, 2016).

However, despite the limited scope, we compare the analytical methods used to forecast baseball results. Since each model differs in statistical theory, use case, and interpretation of results, we can compare them using several components. These evaluations are also based off of Hastie et al. (2008) and shown on the next page. We evaluate:

- Flexibility: the degree to which a model can account for variation in the relationship between features and response variables within individual observations, where greater flexibility allows models to be predictive, at the risk of overfitting (James et al., 2013)

- Complexity: considers both the statistical (theory) and computational (operational) difficulty of running such models (subjectively evaluated, since is no purely objective measure)

- Interpretability (abb. "Interp."): how easy it is to communicate the modeling, such as identifying coefficients, or even just key features used in the prediction, which make the model useful in baseball context.

Table 1: Comparison of machine learning techniques in the literature

Method	Literature	Applications in Baseball	Flexibility	Complexity	Interp.
Pythagorean expectation	Davenport & Woolner, 1999; Heumann, 2016; PECOTA (in Druschel, 2016)	Computes an expected W-L% based on teams' run-scoring and run-prevention abilities	Low; aggregate data only	Minimal	High
Elo ratings	Elo, 1978; Glickman & Jones, 1999; FiveThirtyEight, 2015; Marcus, 2000; Hvattum & Arntzen, 2010	Tracks ratings as team skill levels that determine prior probabilities of winning a game; updates ratings based on the result and other team's rating	Medium; if covariates are used	Low	High
K-Nearest Neighbors	Valero, 2016; PECOTA (in Koseler & Stephan, 2017); ZiPS (in Druschel, 2016)	Computes conditional probability of a win or loss based on Bayes' Rule relative to the K nearest other observations by Euclidean distance	Medium; higher K can overfit	Variable; high dimensional data is costly	Low
Logistic regression (GLM)	Elfrink & Bhulai, 2018; Tait, 2014; Jia, Wong & Zeng, 2013	Computes a linear relationship between features to evaluate an output that can be a continuous response or a probability that can be classified into wins/losses	Medium	Low	High
Decision trees and random forests	Valero, 2016; Elfrink & Bhulai, 2018; Jia, Wong & Zeng, 2013	Recursively partitions data into sections based on information gain, classifying each section; random forests consider bootstrap subsets of features at each split to reduce overfitting	Medium	Medium	Medium/ high; lower for random forests
Gradient boosting	Elfrink & Bhulai, 2018; Pharr, 2019; Jia, Wong & Zeng, 2013	Combines a series of weakly predictive models, such as trees, by sequentially adjusting weights, improving the predictivity in aggregate. Ex: XGBoost	High; many weak learners	Medium	Medium; feature importance
Support vector machines	Valero, 2016; Tolbert & Trafalis, 2016	Creates polynomial kernel decision boundaries in the hyperspace of games, classifying as wins or losses	High	High; nonlinear kernels	Low
Bayesian models	Tait, 2014; Yang & Swartz, 2004; Soicher, 2019	Models individual outcomes as random variables, uses joint/conditional distributions to forecast posterior and predictive results using Bayes' Rule and MCMC	High	High	Medium
Artificial neural nets	Valero, 2016	Mimics the human brain's decision process through sequential layers of "neurons" with varying edge weights. Weights are trained through complex back-propagation for highest flexibility and accuracy	Very high	High; back-propagation needed to train	Negligible: "black box"

3. Motivation and Relevance

Baseball is a complicated sport and has a wealth of detailed game data. Moreover, "America's pastime" is popular and remains lucrative, with MLB teams constituting seven of the most valuable sports teams worldwide (Badenhausen, 2019). Consequently, there are many positive implications of studying baseball game prediction, and sabermetrics, through applying data science and computer science techniques as this paper does. they include:

1. **Academic value in studying complex modeling questions**: identifying and engineering the key components that can understand complicated, high-variance events such as sporting events is challenging but rewarding, as it contributes to our knowledge of data science applications.

2. **MLB long-term franchise management**: a model that explains variance in game outcomes can be leveraged to selectively sign or draft players who show strong performance among certain statistical categories, providing better return for team managers and success for the franchise; this is the "Moneyball" strategy that popularized sabermetrics (Miller, 2011).

3. **MLB game-to-game strategy**: baseball is constantly changing as teams look to increasingly analytical domains for a competitive edge. Identifying drivers of winning games can be used to change the way a team plays, such as the "opener" strategy re-invented in the last two seasons (McWilliams, 2019).

4. **Sports betting**: where legal, a model that can forecast results of games consistently better than average has potential to be profitable on Las Vegas sports gambling blocks, beating gambling spreads. Several papers, including Jia et al. (2013) and Pharr (2019) compare results to gambling outcomes; however, this is not a core goal of this thesis.

5. **Personal interest**: just as teams are becoming more analytical, so too are fans. Understanding the dynamics of how a team can win is intrinsically valuable to fans, who love the game. It allows fans to appreciate nuances in baseball strategy (ex: certain player substitutions or pinch-hitting) that improve odds of winning, deepening their relationship and interest in the sport. This was the primary motivation for this paper, and for many works written about baseball analytics from the **Literature Review**.

4. Models

4.1 Elo ratings with covariates

Baseball games are zero-sum events, played until one team definitively wins, and the other loses; therefore, Elo ratings have been used to track teams' relative strengths over time to build prior predictions for baseball game outcomes (FiveThirtyEight, 2015). Devised by Arpad Elo in 1978 to compare chess players, the Elo system has ratings that represent the comparable strengths for a given team or individual, which are updated due to the result of the match as follows, in this case for some team/individual A after playing a match against B (Elo, 1978; Glickman & Jones, 1999):

$$R_A^{t+1} = R_A^t + K(I(\text{A wins}) - E_A)$$

With the variables defined as follows:

R_A^t	A's prior rating
R_A^{t+1}	A's updated rating after the match
K	Scaling constant; determines how significantly to weight match outcomes
$I(\text{A wins})$	Binary response, 1 if team A wins; 0 if team A loses (no ties in baseball)
E_A	Elo logistic expectation for the outcome of a game vs. B (rating R_B), s.t.:

$$E_A = \frac{1}{1 + 100^{-(R_A - R_B)/400}}$$

The Elo update method and expectations are set formulas, but otherwise the formula is interpretable and flexible. First, E_A parallels the form of the logit function we commonly see in logistic regressions (James et al., 2013; Couronné et al., 2018), and due to scales of $E_A, S_A$for wins and losses (draws are impossible here), represents a probability of A winning the match; $E_A \approx 1$ indicates greater likelihood based on the difference between A and B's Elos.

Second, although this uses the same or a smaller number of variables as Pythagorean expectation formulas, the base Elo model allows us to make statements about individual matches that can be aggregated into a summary season-based prediction, providing us with similar insights but greater granularity (Davenport & Woolner, 1999; Heumann, 2016). Third, Elo allows us to introduce covariates that influence individual match outcomes, which allows us to integrate machine learning analysis through covariates that adjust teams' ratings; for instance, the best starting pitchers in the league will improve their team's odds of victory (FiveThirtyEight, 2015). For example, linear regressions or tree models could compute covariate weights and boundaries at which Elo adjustments are made, influencing the prior Elos similar to FiveThirtyEight's (2015) model, ideally improving accuracy (Valero, 2016).

Finally, Elo can be tracked over time, since it is a constantly updating mode, whereas a Pythagorean W-L system has to be reset every year (after all, there are no runs scored or against to begin with!),[10] adding to the reasons why it has been applied to chess (its original purpose), table tennis and soccer as well (Marcus, 2000; Hvattum & Arntzen, 2010).

4.2 GLM and Logistic Regression models

Generalized linear models are some of the mathematically simplest machine learning approaches, based on linear-combination relationships between observations, feature coefficients, and a response variable. One of the most popular linear models for classification problems is the logistic regression, which relates observations $X$with feature covariates $\boldsymbol{\beta}$to a response variable Y(in our analysis here a binary response variable using the logistic function, defined as (James et al., 2013):

$$P(Y_i = 1 \mid X_i) = \frac{\exp(\boldsymbol{X_i} \cdot \boldsymbol{\beta})}{1 + \exp(\boldsymbol{X_i} \cdot \boldsymbol{\beta})} \qquad \text{(logistic curve)}$$

$$\log\left[\frac{P(Y=1)}{P(Y=0)}\right] = \boldsymbol{X} \cdot \boldsymbol{\beta} \qquad \text{(log-odds)}$$

Logistic regression is popular because its variable coefficients can be translated into probabilities, where observation features and log-odds probabilities are directly correlated (James et al., 2013). Its simpler algebraic form, relative to other machine learning models, makes it convenient for any binary classification problem involving probabilities. For instance, Elfrink and Bhulai (2018) as well as Jia et al. (2013) both use logistic models to predict the result of MLB games.

Typically, linear models are suspect to overfitting on their own since they often use optimization strategies such as ordinary least squares (OLS) to identify coefficients best-suited for the training data (Brownlee, 2019). However, both cross-validation (including scikit-learn's LogisticRegressionCV package that we use in this paper) and regularization methods combat overfitting. Regularization methods either shrink the coefficient weights or remove certain features from consideration entirely, which reduce the risk over overfitting even at a slight cost to training dataset model accuracy (James et al., 2013). We use three of the most common regularization methods:

[10] In fact, FiveThirtyEight has used their model to do precisely that. As seen in this link, they have back-tested their Elo algorithm over the last 130 years, plotting every team's day-to-day Elo to identify franchises' peaks and relative strengths over time.

1. **Lasso regression**, known as a "L1" or linear penalty due to its exponent of 1 on the coefficients, which minimizes the equation's residual sum of squares like OLS, but adds a penalty term to make its objective function as follows (James et al., 2013; van Wieringen, n.d.):

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \Big[\sum_{i=1}^{n} (Y_i - \boldsymbol{X_i}\boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{k} |\beta_j| \Big]$$

   Here, the first term is RSS, residual sum of squares, for the logit model; the penalty term has a tuning parameter $\boldsymbol{\lambda}$ for all k coefficients $\boldsymbol{\beta_j}$ (van Wieringen, n.d.).

2. **Ridge regression**, known as a "L2" or quadratic penalty due to an exponent of 2 on the coefficients, similarly behaving like Lasso but being well-fit, but will not shrink coefficients exactly to 0 unlike Lasso (James et al., 2013; Hastie et al., 2008):

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \Big[\sum_{i=1}^{n} (Y_i - \boldsymbol{X_i}\boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{k} \beta_j^2 \Big]$$

3. **Elastic net**, which combines the lasso and ridge regularizations to optimize the following objective function (Zou & Hastie, 2005) -- for simplicity, on the right hand side we have used RSS, $L1(\lambda_1, \beta)$, $L2(\lambda_2, \beta)$ to indicate the aforementioned regression residual sum of squares, and each of the penalty terms from above (van Wieringen, n.d.; James et al., 2013):

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= \arg\min_{\boldsymbol{\beta}} \Big[\sum_{i=1}^{n} (Y_i - \boldsymbol{X_i}\boldsymbol{\beta})^2 + \lambda_1 \sum_{j=1}^{k} |\beta_j| + \lambda_2 \sum_{j=1}^{k} \beta_j^2 \Big] \\ &= \arg\min_{\boldsymbol{\beta}} (RSS + L1(\lambda_1, \boldsymbol{\beta}) + L2(\lambda_2, \boldsymbol{\beta})) \end{aligned}$$

4.3 K-Nearest Neighbor Classifiers (KNN)

KNN is a simpler classification approach designed to mimic what the decision boundaries on a native Bayes classifier would be like (James et al., 2013). In context of our baseball prediction problem, a KNN algorithm would identify, for each given game x_0, the K "closest" (Euclidean distance in a multi-dimensional plane) other observations, and averages their response values (if the team won or not) to estimate a Bayesian conditional probability for $x_0$being a win or loss for the team, selecting whichever result had the higher conditional probability (Hastie et al., 2008). Its mathematical probability can be computed as, for a binary problem, we can view the chance of being a home-team win, $P(Y = 1)$, for some observation can be derived (James et al., 2013):

$$P(Y = 1) = \frac{1}{K} \sum_{i=1}^{n} I(y_i = 1)$$

where y_i are the response variables for each of the K-nearest neighbors, by Euclidean distance.

However, since it takes in little prior information about the data, it is likely to overfit if the hyperparameter K is too large, due to the flexibility of this model (James et al., 2013). Furthermore, they can be computationally intensive for high-dimensional datasets; it has to evaluate distances for all observations and KNN lacks a direct objective function to optimize such as that of our linear and logistic models (Smola & Vishwanathan, 2008).

4.4 Decision Trees and Random Forests

Tree-based models recursively linearly partition the space of observations based on conditions in the data (for example, $hometeam = 1\ vs.\ 0;\ teamAVG\Delta > 0.040$[11]), evaluating different options via conditions like Gini index or entropy, eventually assigning a classification or regression value to each of the final partitions of observations (James et al., 2013).

The Gini index is a measure of variance across our classes, here our binary options that the outcome variable can take on as shown below, where $\hat{p}_{m,i}$ indicates the proportion of our observations in each partition that have outcome value i -- simplified for our two-class win/loss model (James et al., 2013). Meanwhile, entropy measures information gain, which reduces the amount of future depths and splits needed to partition tree elements (Sharma, 2020). These two methods behave similarly in practice, where both Gini and entropy will be very low if a partition's games are more homogeneous -- which makes sense when we have very low variance in that segment, and hence defining that partition does not help us define the overall tree much better (James et al., 2013).

$$\text{Gini } G = \hat{p}_{m,1}(1 - \hat{p}_{m,1}) + \hat{p}_{m,0}(1 - \hat{p}_{m,0}) \qquad \text{(only two classes)}$$

$$\text{Entropy } D = -\left[\hat{p}_{m,1} \log(\hat{p}_{m,1}) + \hat{p}_{m,0} \log(\hat{p}_{m,0})\right]$$

Trees have a unique advantage of being both flexible (since hyperparameters can be tuned to control how detailed the splits should be) and interpretable (since their classification methods are similar to how humans evaluate decisions or classify objects based on attributes)

---

[11] These are hypothetical examples. "teamAVGΔ" would be how much greater the predicted team in question's batting average is than the opposing team; positive difference being better for winning odds, all else equal.

(Hastie et al., 2008). Unfortunately, they tend to be less stable and potentially less accurate on new data, because they make binary linear splits within the data (James et al., 2013). Techniques such as cross-validation or pruning must be used to carefully reduce overfitting, since in theory, a tree could recursively partition all elements to get perfect training dataset accuracy (Hastie et al., 2008; Sharman, 2020).

Random forests, also called bootstrap random forests, are constructed through numerous decision trees, and build to improve accuracy while reducing overfitting, explaining their popularity (used by Elfrink and Bhulai (2018); Jia et al. (2013); Couronné et al., 2018, among others). At each split, each decision tree only evaluates a bootstrap-sample vector of predictors (typically a minority), which prevents trees from all using the same predictors; through this, random forest prediction error rates are empirically lower that trees, and the model is more robust to outliers in the data (Breiman, 2001; James et al., 2013).

As with most ensemble methods, such as bagged trees (which are also a collection of trees, but evaluate all predictors when making splits), the prediction of the forest as a whole depends on the predictions of each individual tree, of which there may over 100, typically through a majority vote (James et al., 2013). However, since each of the components of a random forest is a classifier in itself, methods such as consensus-based combining (CCM) can vary classifier weights until there is an eventual convergence, further improving classification accuracy (Alzubi et al., 2018).

Yet, one downside of random forests is explanatory power. Although the importance of features in random forests can be extracted (Gregorutti et al., 2013) -- and modern code packages include this ability -- the complex nature of random forests makes it more difficult to understand the decision mechanism for each classification. With many individual classifiers and different subsets of predictors in each case, although we can extract the most "important" features, it is not possible to understand a broader decision framework for classifying games in the way a linear model such as logistic regression, or the simpler but less stable decision tree provides.

4.5 Support Vector Machines/Classifiers (SVM/SVC)

SVMs are a non-linear extension of support vector classifiers (SVC) -- models that use hyperplanes that separate the observation space, creating a kernel to , while creating a "soft margin" of classification that allows errors (James et al., 2013). Mathematically, through adding a slack variable ξ in the optimization constraints, the underlying SVC models allow non-linearly separable data to be split by these hyperplanes through allowing for margin errors, or points within the classification margin range that will be intentionally misclassified (Smola &

Vishwanathan, 2008); by adding higher-order terms and polynomial kernels, a SVM is able to represent nonlinear classification boundaries that improve the model's flexibility in aggregate (James et al., 2013), which helped Valero (2016) find optimal performance with SVMs. Despite taking longer to train, SVMs are good at combating overfitting, and have also been used in basketball sports prediction (Haghighat et al., 2013).

However, the SVMs pay for the improvement in predictive power; the optimization increases sensitivity to scaling of inputs and reduces interpretability, as there are not direct evaluations of individual features' impact on the classification outcome (Hastie et al., 2008), as opposed to tree or regression based models which are more advantageous for explaining the components behind the classification.

4.6 Boosting models and XGBoost

Gradient boosting is an ensemble modeling strategy that combines a series of weak learners in sequence; this differs from random forests, which considers many weak trees in aggregate and intentionally lowers correlation through samples of predictors at each split (Natekin & Knoll, 2013; Breiman, 2001). By iteratively using many small, weaker models, the boosted trees tend to perform better, especially since their learning rates and tree hyperparameters can all be tuned for increased cross-validation accuracy, such as with tuning methods like grid search and randomized search -- which we eventually use (James et al., 2013; Bergstra & Bengio, 2012; Bergstra et al., 2011).

Specifically, we look at XGBoost for this analysis, which is faster yet accurate gradient boosting method for tree-based classifiers (Chen & Guestrin, 2016). XGBoost is popular among data science competitions due to its accuracy and implementation speed -- even though boosted methods tend to learn more slowly -- which makes it well-suited to large datasets and complex relationships between many features and outcome variables (Chen & Guestrin, 2016).

4.7 Artificial Neural Nets (ANN)

Much like SVMs, ANNs are powerful predictive models that sacrifice understandability for predictive accuracy. Using a collection of neurons that are designed to simulate the decision framework of a human brain, each ANN perceptron takes in multiple inputs (such as team batting average difference, starting pitcher ERAs, etc.), using a non-linear bias term and activation function (ex: ReLu, sigmoid) to activate certain neurons, and passing the relevant weights and outputs to further layers that eventually lead to the output, a classification of the

win or loss (Chauhan, 2019). To fit the model, neural networks use a technique called gradient descent (Hastie et al., 2008), which minimizes a loss function (error sum of squares between trained model output layer and actual classifications in this case) and iteratively do this from the output layers back through each layer of perceptrons, adjusting weights, in what is known as back-propagation (Li et al., 2012; Moawad, 2018).

The accuracy to which neural nets can be tuned through this back-propagation algorithm is NNs have become popular even despite their computational requirements (often needing GPUs). However, the multi-layered mathematics behind these algorithms create "black boxes" of computation, which makes it difficult to determine the inputs considered and their relative importance in determining the final classification; comparatively limited progress has been made to understand feature and information extraction, such as by Lee and Landgrebe (1997). We discuss ANNs in this section since they are a relevant classification model; however, since our goal is to have an interpretable model, we refrain from fitting one because it is harder to extract qualitative meaning from neural networks.

5. Methodology

5.1 Problem definition

Our goal is to make prior forecasts of win-loss results in MLB games. Since we are only interested in the win-loss result, rather than the score, this becomes a binary classification problem (Valero, 2016; Elfrink & Bhulai, 2018). Most MLB games are zero-sum because games are extended until one team wins,[12] and each game always has a home/away team, so we define our binary y-variable (named "outcome") similarly to Pharr (2019):

$$\text{outcome } y_i = \begin{cases} 1 & \text{if home team wins} \\ 0 & \text{if away team wins} \end{cases}$$

For our model, we will use team and player stats as features, such as rolling batting averages, which we believe will explain greater variance in computing the probability of being classified as a win or not for the home team (most models we explore allow for the latter). For this, we do have to assume that win determinants in baseball have not changed meaningfully over the period we are building our model on; namely, that the relationship between observations/covariates and outcome response is stationary.

This is not realistic over the course of baseball history. For instance, starting pitchers' average innings per start mattered far more in the 1960s, when complete games were frequent, than now, where the game's later innings are dominated by relief pitchers (S. T. Jensen, personal communication, February 3, 2020). As a result, we restrict our data to observations post-2000, which allows us to train and test on datasets from similar data-generating distributions of baseball play -- allowing us to be more accurate and make realistic predictions.[13]

5.2 Software stack

We used Python and Jupyter Notebook to compute the majority of this analysis. Primary packages used include pandas (the Python data analysis library), numpy (matrix

[12] In fact, after a 2007 rule change, ties can only occur at the end of the season if the game does not impact the postseason; there has only been one tie since, on September 29, 2016, when the Chicago Cubs (8 games ahead of any other team) drew against the Pittsburgh Pirates (9 games short of playoff qualification), where the outcome did not impact season standings (Baseball Reference, n.d.-c; Baseball Reference, n.d.-f).

[13] Note: recent changes like "openers" can impact this game, such as by devaluing starting pitchers (McWilliams, 2019). However, we ignore this because it is challenging to forecast in advance.

computation), matplotlib.pyplot (used for visualizations), xgboost (Extreme Gradient Boosting), scikit-learn (machine learning), scikit-plot (model visualization), statsmodels (statistical modeling and tests). Citation for their documentations are found in the Bibliography; we cite each package's documentation and reference the functions/models used with the package name.

5.3 Reducing overfitting: train/test splits and cross-validation

One key problem in machine learning is overfitting. This is also referred to in context of the bias-variance tradeoff: the more attuned our model is to characteristics of the training dataset, the lower our bias (difference in error), but similarly the greater our variance on fitting unseen datasets -- leading to higher error rates and "overfitting" (James et al., 2013). We simulate this in machine learning modeling through the use of training and testing datasets, where we only fit the model on the "training" portion of the dataset, and then see its performance on the remaining "test" dataset, which should confirm our models do not overfit. Standard training and testing splits randomly assign observations to either the training or testing data at common ratios of 80/20 or 75/25 (Shah, 2017).

Yet, this standard approach does not work as well for our classification problem. As we also discuss in **Analysis**, we want to avoid data contamination (Valero, 2016) and only train our model on both features/observations that it would have access to in advance of predicting a given game. This is stricter than just demanding independent samples. The rationale: if we used a standard training and testing data split, we could have games in the training set that are played before games in the testing set. Since we are using features that are customized to each day, this independent testing and training split would be statistically appropriate; however, this does not emulate how we could train models in practice, making it less valid (S. T. Jensen, personal communication, April 22, 2020). As a result, we train on the 2001-2015 seasons, and use the 2016-2019 datasets as hold-outs individually.

Cross-validation strategies help combat overfitting in the training process, where the model is evaluated on different samples of the training dataset to estimate an out-of-sample error for unseen data; low differences/biases between cross-validation and training dataset error, as well as low cross-validation accuracy variances, suggest lesser overfitting (Hastie et al., 2008). In this paper, we use k-fold cross validation to do this, where the training dataset is partitioned k ways, and for each "fold," our model trains on the remaining k - 1 partitions, evaluating that fold's cross-validation error as the testing error on the held out partition (Hastie et al., 2008). This gives us estimates of prediction error through the mean and variance of the fold estimates (James et al., 2013):

$$\text{CV Error} = \frac{1}{k}\sum_{i=1}^{k}\sum_{j=1}^{n} I(y_j \neq \hat{y}_j \mid \text{ fold } i)$$

We use 5-fold cross validation, scikit-learn's default CV setting for many algorithms. In fact, cross validation for the bulk of our models is performed automatically by our Python modeling packages, where models like LogisticRegressionCV or RandomForestClassifier automatically cross-validate parameters using a specified number of folds. Through this, we minimize our overfitting and improve eventual performance on our holdout data for evaluation.

5.4 Analysis roadmap; Code and GitHub repository

Table 2: Data science approach

Modeling Step	Relevant Thesis Section
1. Define the problem	Methodology
2. Identify models and testing metrics that will be used to compare them	Models; Model Evaluation
3. Acquire datasets and perform data cleaning	Preprocessing: Data Cleaning and Feature Engineering
4. Perform feature engineering and split data for test and training sets	Preprocessing: Data Cleaning and Feature Engineering
5. Train models on training datasets and tune hyperparameters to minimize cross-validation overfitting	Analysis
6. Evaluate models based on the testing metrics and select one or several best models to use	Results
7. Quantitatively and qualitatively interpret the model's features in context of the analytical problem	Discussion

The Python notebooks, some copies (when space permitting) of data files, Python scripts, and some charts' raw image files are included on a public GitHub repository under my name, username @andrew-cui. This code and associated work were done solely for the purposes of EAS 499, Senior Capstone Thesis, Spring 2020.

The link is here: https://github.com/andrew-cui/mlb-game-prediction

6. Model Evaluation

Machine learning is valuable for its ability to evaluate future observations, after "learning" the patterns in existing data, which is why we use hold-out and validation datasets to combat overfitting. However, since we have many variables of features, models, and patterns in the hold-out datasets, we must consider multiple methods to evaluate our classifier, where our best model should perform well across all of these metrics.

6.1 Prediction/Classification Accuracy

The most frequently used evaluation criteria, found in every single MLB game prediction paper or article in the literature review, is classification accuracy, defined as (Ferri et al., 2009):

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} I(y_i = \hat{y}_i)}{n} \times 100\%$$

This computes the rate of correctly predicted baseball games, both home team wins and losses. Due to its simple computation and clear meaning, prediction accuracy is used widely by cross-validation techniques, which compare accuracy across each validation set to minimize bias and variance relative to the training data accuracy (Hastie et al., 2008).

Since we are looking at the 2016-2019 seasons, which are individual entities, it makes most sense to compute prediction accuracy over each season individually. Although we do not expect drastic differences between seasons, this parallels our use of such a model in practice, and also gives us more data points to consider. In addition, both Soicher (2019) and Jia et al. (2013) describe time-dependence of game predictive accuracy, since earlier games with less data may be less predictive of a team's true quality. So, we also will evaluate our model on a monthly scale (only 1-2 seasons) to compare its results to Jia et al. (2013) as well as validate their findings.

6.2 Brier score for weighted MCE

We want to reduce severity of inaccuracy. To do this, we use Brier scoring, which for each estimate, computes squared deviation from the outcome (Czakon, 2019; Roulston, 2007):

$$\text{Brier} = \begin{cases} \frac{1}{n} \sum_{i=1}^{n} \left[P(y_i = 1) - 1\right]^2 & \text{if home team wins} \\ \frac{1}{n} \sum_{i=1}^{n} \left[P(y_i = 0)\right]^2 & \text{if away team wins} \end{cases}$$

Brier scoring has been used by prediction engines such as the Good Judgement Open (Good Judgement Open, 2019). Smaller Brier scores are good, because they indicate that inaccurate evaluations were less severe (for example, a P(outcome = 1) probability of 0.56 is better than a probability of 0.95 if the home team actually lost, since the lower probability represented a lower prediction confidence). We can compare Brier scores directly, or evaluate them relative to a baseline (in our case, a model where the home team wins every game; a naïve classifier), using a Brier Skill Score (Roulston, 2007; DWD, n.d.):

$$BSS = \frac{BS_{baseline} - BS_{model}}{BS_{baseline}}$$

In this case, a positive Brier skill score close to 1 indicates a much lower Brier score for our model than the baseline, which would suggest our model is stronger (Roulston, 2007).

6.3 ROC Curve and AUC

Another popular metric for evaluating classification model quality is the ROC curve, or receiver operating characteristic -- which plots true positives (games where our model predicts the home team wins, which it does) against false positives (games where we predict the home team wins, but it loses) across different probability classification thresholds (Vuk & Curk, 2006). Note that in our modeling, we still use the standard 0.5 classification rule, since we are indifferent to wins and losses.[14]

The quantitative component of ROC is known as AUC, or "area under the curve," a metric that can be used to compare ROC curves of different models (James et al., 2013). We compare the ROC of our models to each other as well as a reference diagonal line with AUC = 0.5; the farther our model's ROC curve bows away from this reference line, and correspondingly the higher our AUC is, the better fit this model is for the data (James et al., 2013l; Vuk & Curk, 2016).

6.4 Lift

Elfrink and Bhulai (2018) highlight the value of having higher-confidence predictions be more likely to be accurately forecasted (such as probability that it is a win for the team); while they propose it in the context of betting, they do not explore this metric extensively. Inspired

[14] See **Analysis**.

by their discussion, we will examine events with a higher response probability (greater prior likelihood of a team winning) to see if they are indeed more likely to be accurately classified, known as the "lift" of the model (Vu et al., 2019; Vuk & Curk, 2006).

For games labeled as a high probability of a win, lift measures the ratio of actual wins. Ideally, a classification model would be more accurate for these high-likelihood events (Soicher, 2019), being up to several times better at classifying the top percentage of observations for a given outcome (Czakon, 2019). While some games in an MLB season will be very close and challenging to forecast in advance, being able to identify certain higher-probability outcomes improves the usefulness of our model even if we are capped at 55-60% accuracy due to improved competitive balance in the sport (Ajilore & Hendrickson, 2007) and game-idiosyncratic variance; a model with strong lift would be useful for betting on baseball games, making plays only on games forecasted as high-likelihood wins to maximize returns (Elfrink & Bhulai, 2018).

6.5 Summary

Through considering these metrics and others discussed in statistics and computer science literature, we use the following order of evaluating our models. We include qualitative consideration as discussed in our **Literature Review**

1. **Classification accuracy** scoring on our holdout "test" dataset, from 2016-2019, and monthly measures
2. **Brier scoring** to penalize models with lower lift (greater misclassification severity)
3. **ROC curves** of false positive vs. true positive classification rates, and **AUC** measures
4. **Interpretability of model** as a tie-breaker, since we want to understand the determinants of a baseball game, in addition to gaining prediction quality

The open-ended nature of this final characteristic gives us strong flexibility in how we choose to apply it. For instance, comparing models that have different measures of feature importance (regressions and direct coefficients on one hand; neural nets with little information on the other) can be useful. In addition, running our model over specific team-seasons or years would let us compare our forecasts to truth. Finally, we could use the model for individual games, bringing in personal knowledge about the sport to judge the sensibility of our predictions.

7. Preprocessing: data cleaning and feature engineering

7.1 Data acquisition

Our primary dataset -- including game logs and in-game events -- comes from Retrosheet's baseball database and includes all events from 2000 to 2019, inclusive, (Retrosheet, n.d.-c; Retrosheet, n.d.-d). Since each of 30 teams plays 162 games, and each game has two teams (home and away), this is a total of $20\ seasons * 81\ home\ games/team * 30\ teams = 48{,}600$games of data, at both game-level (game results) and event-level (individual hitting and pitching at-bats/outcomes) granularity.

Our secondary dataset comes from Sean Lahman's[15] baseball database (Lahman n.d.), specifically using his People and Pitching CSV datasets to compute covariates for our pitchers. The preprocessed datasets had a combined size of 291.9 MB. Although these datasets originally come with many features, a large portion of our work here involves data cleaning and feature engineering. As a result, we withhold more detailed descriptions of the columns until after this portion has been completed.

7.2 Data cleaning: game logs

The game log data (Retrosheet, n.d.-c) provides metadata (Retrosheet, n.d.-b) about the date, teams, result, and information about data quality. We preprocessed it as follows:

- Removed games that did not finish on the same day they started, since this is challenging to model, and it is a small proportion of the dataset[16]
- Combined records for teams that had name-changes (ex: the Florida Marlins and Miami Marlins), to leave us with 30 teams in total
- Created our response variable "outcome": 1 if home team wins; 0 if away wins

We are left with 48,556 games across 20 seasons, with 26 columns in game date, team names, starting pitchers, starting lineups, and our response variable. Our response variable mean was 53.87%, corresponding to the overall percentage of games won by the home team.

---

[15] Coincidentally, I discovered that Sean Lahman is an active reporter in my hometown of Rochester, NY.

[16] Official baseball scoring has these games played later but retroactively updated in the standings/statistics; however, since games might be played months later, the actual team on the field often differs, hence making it challenging for us to model. As less than 3/4 of a percent (32/48600) of the data, it is a reasonable omission.

7.3 Data cleaning: team/game-level data

We are interested in using team-level performance covariates as observed in the literature, where performance gaps between teams such as home-run differences should provide signals for win-loss outcomes (Valero, 2016; Pharr, 2019; Yang & Swartz, 2014). Using Retrosheet event files and metadata (Retrosheet n.d.-a; Retrosheet n.d.-d), we scraped each relevant at-bat and its corresponding statistical outcomes (since there is no native tabular format, necessitating manual data wrangling) across all seasons from 2000 to 2019.

We tracked at-bats, hits, total bases, walks, home runs, runs, and sacrifice flies at a player-game-level, so that we can compute rolling means for team performance such as AVG, OBP, and SLG. Similar to Valero (2016), we omit information about the actual game outcomes, such as the number of outs, home and away runs scored, game protest, or time of game, since they would contaminate the data we are modeling. Out of the remaining features though, since Retrosheet's comma-delimited text files include many edge cases, despite counting for stolen bases, sacrifice hits, and wild pitches, we miss the following scenarios that impact the aggregate statistics counted:

- Balks
- Double-plays that lead to a run scoring (coded differently by Retrosheet)
- Errors by the fielding team that lead to runs scoring (such as errant throws on a pickoff attempt)

In aggregate, our scraper performs reasonably accurately, extracting nearly 4.5M rows from the CSV structure across 600 files. For a large-scale, "big-data" wrangling problem, with a complicated text-encoding, is non-tabular, and physically large, these error rates are reasonably low (Ives et al., 2018), at less than 1-2% differences for the 2000 Los Angeles Angels' aggregate statistics (Baseball Reference n.d.-a) for instance:

Table 3: Evaluating data cleaning quality (ex: 2000 Los Angeles Angels)

	AB	**RS**	**H**	**HR**	**TB**	**BB**	**SF**
Cleaned Data	2767	447	797	130	1364	339	23
Actual	2742	447	788	130	1346	339	23
% Difference	0.91%	0	1.14%	0	1.34%	0	0

We also omit doubleheader games, since our eventual modeling iterates over dates, which becomes problematic to update information for if we have multiple entries on the same

date; however, this affects only 452 (~0.01%) entries in the dataset. Though tracking player-level rolling averages is most detailed -- and we have the foreign keys to do so in the tables -- it is computationally impractical. This is a limitation of our analysis, as instead, we aggregate by day and team to compute ~20,000 rows for each team's performances.

7.4 Data cleaning: pitching-level data

Retrosheet's event file dataset (Retrosheet, n.d.-c) includes event-level information that can be used to identify a game's starting pitchers and their statistics for the game; however, the metadata format (Retrosheet, n.d.-b) requires tabulating over each dataset while tracking the active pitchers to a $O(n^2)$complexity in addition to being algorithmically challenging to set up, which makes tracking game scores (FiveThirtyEight, 2015) harder to do even though they directly evaluate starting pitcher performance. Instead, we use a proxy for pitcher data, assuming their performance matches the prior season's average performance. For instance, for some statistic Y in year t, game observation i in the data, our prior estimate for the pitcher's performance is:

$$Y_i^t \sim N(\overline{Y}^{t-1}, 0)$$

Lahman's database (Lahman n.d.) includes pitcher aggregate statistics over each season, which we match to our Retrosheet database using the foreign key retroID (Lahm 2019). We select starters as pitchers that make more than 12 starts in a given season.[17] We kept summary statistics like ERA and WHIP (Stick, 2005), computed measures such as K/BB, K/9, and BAA, and added a feature for FIP (Valero, 2016), or Fielding-Independent Pitching.

Sabermetricians created FIP to evaluate MLB pitcher "quality," all else equal to teammates' defensive performance and random chance, arguing that a pitcher with ERA in excess of FIP may be "unlucky" and their ERA is expected to improve in time (Slowinski, 2010). We normalize FIP so each year's average is equal to the league ERA (Slowinski, 2010):

$$FIP = \frac{13 \cdot HR + 3 \cdot (BB + HBP) - 2 \cdot K}{IP} + C_{FIP} \text{ such that } \overline{FIP}^t = \overline{ERA}^t \text{ across the league}$$

[17] We compute the number of pitchers who make a given number of starts per year; finding a cutoff that leads to around 30 * 5 = 140-160 starting pitchers, since, conventionally, a team has five starting pitchers in its rotation. A graph can be found in the **Appendix 6**.

7.5 ELO model

Our first feature was inspired by Elo models (Elo 1978; Glickman & Jones 1999) such as FiveThirtyEight's baseball predictor (2015), which track a team's underlying "quality rating" based on wins and losses against opponents, where the ratings create their own, albeit limited, forecasts of probabilities of future matches. FiveThirtyEight's model is shown in **Appendix 4**.

Elo models are fantastic for understanding performance over time but were built to compare chess players when their primary statistics were wins and losses (Elo 1978) rather than the detailed data available in MLB predictions -- in fact, FiveThirtyEight's model includes very few features. This is a binary classification, so instead of Elo being used to determine the probabilities themselves, we instead use it to account for "momentum," hoping to capture some variance in pitching, relief pitchers, or streaks of good play that other covariates do not.

We start all Elos at 1500 (FiveThirtyEight, 2015) in 2000, and track over time, with hyperparameters tuned to reduce variance in teams as the season goes on. For future years, we start the Elo on the prior year's final Elo rating. Since our data is sorted by date and the teams in each game are mutually exclusive, we can run an $O(n)$ Elo algorithm. Our basic model, over 2001-2019, has an accuracy of 0.5442 with 95% proportion confidence interval (since each season has the same number of observations) of [0.528, 0.560]. This interval contains our baseline p = 0.5387 of home team wins.

7.6 Feature engineering

Using a team's season-wide statistical performance, as Jia et al. (2013), Valero (2016), and Tolbert and Trafalis (2016) do, provides signal, and would be appropriate if we restrict the data used to only games previously played. For example, tracking a game on August 1, 2018 using the entire 2018 season would not be appropriate because the team's aggregate statistics are influenced by nearly two months' data that is yet to come and would not be available if forecasting in practice, causing data contamination (Valero, 2016). Subsequently, we use 7-day rolling periods for each hitting statistic and set the prior values at the beginning of the season to be the team's average from the entire previous year.

For pitchers, as described, we use their previous year's average data, setting the information to be 0 in any cases where we do not have their previous year's data; this is a weak estimate, but our best possible since we do not have the game-level data that FiveThirtyEight (2015) uses to track game scores.

We also bring in our Elo momentum tracker, and include the number of rest days a team has (FiveThirtyEight, 2015), similar to how Valero (2016) uses the team's current

Pythagorean expectation to model team strength so far in the season, or how Yang and Swartz (2014) use a team's existing W-L record as a feature. In addition, while betting lines for baseball games can be strong predictive factors for the actual result (Pharr 2019), the predictive accuracy of diverse individuals in aggregate is an observed phenomenon known as the "wisdom of crowds" (Surowiecki 2004) -- hence, it is neither a novel signal nor a fundamental baseball component that impacts the game's outcome, so we do not include it.

We also create a feature for ISO (MLB n.d.), or isolated power to separate the impact of power from getting on base. We remove features including AVG and SLG (since ISO is a linear combination of the two), ERA (since FIP can be used to track this performance) and IP averages per start (due to low correlation). We use 2000 to set prior estimates for our features, so we exclude that year's games from the final dataset. Our final set of features has dimensions (46128, 9), with a tenth column for the outcome:

- Team hitting: OBP, ISO
- Team starting pitching: FIP, WHIP, K/9, HR/9, K/BB
- Additional metrics: ELO, rest days between games

For each statistic, we computed differences and percent differences between our feature statistics for the home and away teams, with the home team as our baseline. A positive value means a higher statistic (not necessarily good, such as FIP) for the home team.

$$Y_{DIFF} = Y_{HOME} - Y_{AWAY}$$

$$Y_{\%DIFF} = \frac{Y_{DIFF}}{Y_{HOME}} \times 100\%$$

We used the raw differences for models that require scaling, since it is complicated to interpret a model based on standard-scaled percentage differences between two statistics, and scaling puts all the feature values on the same mathematical scale. We used percent differences for scale-invariant algorithms since a more extreme difference, hence stronger signal -- while also avoiding scaling the data and add a layer of interpretation complexity.

8. Analysis

8.1 Exploratory data analysis

We begin by looking at our features, and some basic distributions of our data. Examining the distributions of several features as percentage differences between home and away teams,[18] we find that most appear normally distributed, with limited primarily seen in the ISO and K/BB graphs. This is promising for our analysis; data with a heavily skewed distribution could mask the signal from our features, particularly on the linear models such as our regularized logit regressions.

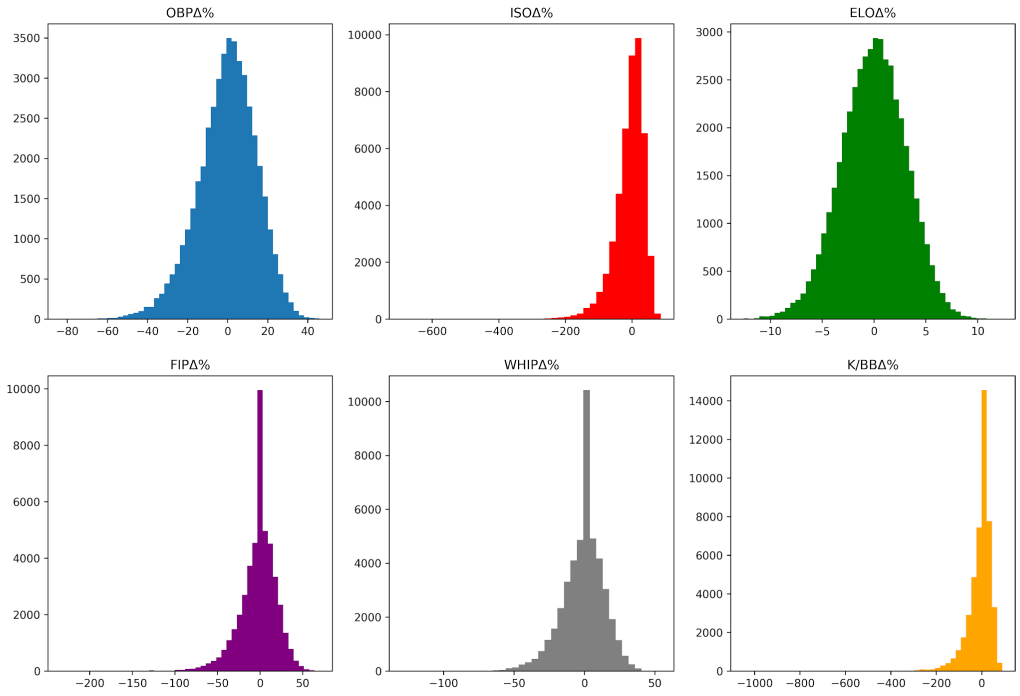

Figure 2: Histograms of % difference features

Our data shows correlation between factors such as HR9 and FIP, which is sensible since home runs are considered in the calculation of FIP. We have weak correlations with our response variable, with OBP and ISO the highest (r = 0.27, 0.19 respectively) -- indicating no single factor can predict games:

[18] We also use raw differences and standard-scale them in preprocess. We do not show these plots here; however, they appeared all normally distributed.

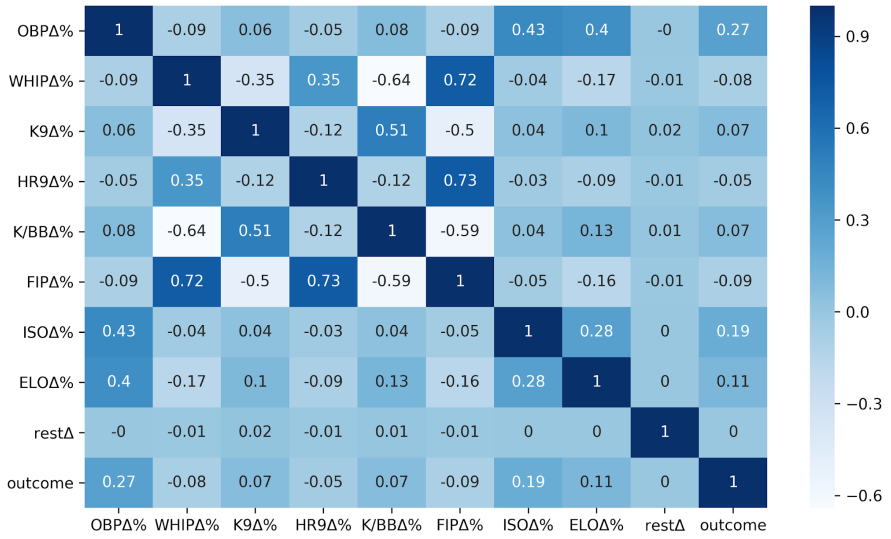

Figure 3: Correlation plot of % difference features

Summary statistics of our features using Δ% are seen here:

Table 4: Descriptive statistics for Δ% features (Δ% implied in feature names)

	OBP	**ISO**	**FIP**	**WHIP**	**K9**	**HR9**	**K/BB**	**Elo**
Mean	-0.361	-7.749	-2.283	-1.104	-3.551	-8.090	-10.008	-0.057
SD	14.753	46.718	22.182	14.868	46.676	54.387	54.387	3.150
Min	-83.206	-690.004	-233.875	-129.224	-269.458	-663.718	-1048.39	-12.587
50%	0.641	0.976	0	0	0	0	0	0.018
Max	45.785	86.971	69.886	54.189	73.482	88.837	90.411	12.373

We also use the differences themselves as features, but standard scaled so they fit a standard Normal distribution and allow us to work with scale-sensitive models (James et al., 2013); their summary statistics are shown here. The median for the pitching statistics is 0 due to two factors: 1), the normal distributions we saw in the histograms for the raw differences (although not plotted); 2) additional values 0 caused by our data wrangling, which occurred if both pitchers did not have any prior data.

These statistics' distribution looks less promising to analyze because almost all the means are around 0, where there would not be a strong signal with feature coefficients. However, it provides valuable insight into what baseball games look like in terms of statistical spreads. For instance, means all around 0 indicate that the "average" game has balanced out more (Ajilore & Hendrickson, 2007), even with the presence of outliers such as a 5.0 difference in FIP (indicating one pitcher is a top-line ace and the other is far below average).

Table 5: Descriptive statistics for Δ features (Δ implied in feature names)

	OBP	ISO	FIP	WHIP	K9	HR9	K/BB	Elo
Mean	0.002	0.001	-0.001	-0.001	0.002	~0	0.005	-0.113
SD	0.462	0.058	0.838	0.195	1.807	0.382	1.353	47.109
Min	-0.178	-0.275	-4.167	-1.155	-8.682	-2.028	-14.008	-176.251
50%	0.002	0.001	0	0	0	0	0	0.267
Max	0.196	0.215	5.000	0.999	9.586	2.208	13.964	193.659

8.2 Modeling approach

The most straightforward way to train our models is to randomly set a training and testing split, with cross-validation to reduce overfitting. However, in an actual MLB season, we do not have the luxury of randomly splitting up games, because they occur sequentially, but a train-test split could have us training on data from September 2019 even if our test samples include games from April 2002. Statistically, since all the feature values are game-specific and would not be affected by our approach here, there is no difference; however, a true baseball model would be fit on prior seasons and then applied to an entirely new season (S. T. Jensen, personal communication, April 22, 2020). Since we want to make our model applicable in an actual baseball setting, it is appropriate to behave as if we are in such a setting.

Lacking a 2020 season[19] due to COVID-19, we set our training sample for the overall models to be the 2001-2015 seasons, and our test sample to be 2016-2019. As part of our evaluation, we will also track performance over the months in a given season, as suggested by Jia et al. (2013). Our training dataset includes 36,417 games while our holdout/test dataset includes 9,711 games, a 79-21% split (approximately 80-20).

8.3 Models considered

We evaluated a series of machine learning models by training on the 2001-2015 seasons and evaluating their test accuracies each year over the 2016-2019 seasons separately. For consistency, we set random_state = 499 and used 5-fold cross-validation in testing, to combat

[19] Our initial goal was to train the model and live-track its performance during this season to see how it behaves on completely unexplored data.

overfitting. We fit the following models and tested against a "naïve" model where the home team always wins.

Logistic regression

- Lasso (L1) penalty regularization
- Ridge (L2) penalty regularization
- Elastic net (L1+L2) penalty regularization

Support vector machine

K-Nearest Neighbors classifier

Tree-based models and ensemble methods

- Decision tree classifier
- Random forest ensemble
- XGBoost classifier

Stochastic gradient descent classifier

We chose these models based off the ones included in the literature. The exception is the stochastic gradient descent classifier; the scikit-learn SGDClassifier implementation includes a loss function specification that can model either an SVM or logistic regression, increasing flexibility of our models. While we reviewed Bayesian models and neural nets briefly, we do not discuss them here. Certainly, a Bayesian model that can update its posterior distributions per game and capture greater variance (Jensen, 2020) or a complex neural net (Valero, 2016) would provide strong predictive power, but sacrifice interpretative ability (and as Valero (2016) finds, does not appear to provide a meaningful increase in predictive strength).

We use several classifiers that compute probabilities, both $P(y = 0)$ (away wins) and $P(y = 1) = 1 - P(Y = 0)$ (home wins), for each game. By default, these classifiers assign any probability less than 0.5 to a response value of 0, and any above 0.5 as a response value of 1. However, we have no interest in home teams specifically; rather, this framing is a byproduct of the data we have available. This indifference allows us to accept the default 0.5 classification rule across all models to which it applies.

8.3 Model tuning and evaluation

For simple models, particularly the logistic regression variants, hyperparameter tuning can be done manually through adjusting cross-validations or built-in packages such as LogisticRegressionCV. In each case, we ran the models, adjusted hyperparameters, and compared each iteration's cross-validation results to identify ideal parameters.

Meanwhile, for our tree-based ensemble models (random forests and XGBoost), since there are a set of continuous hyperparameters, we cannot do this tuning by hand. Our solution is automated hyperparameter optimization, which runs iterations quickly and produce the strongest set of hyperparameters through 5-fold cross-validation. Since our final dataset, after aggregation, has 9 features and is trained on 36,000 rows, its size is reasonable for a randomized search (implemented as RandomizedSearchCV) which runs many cross-validation iterations on randomly selected sets of hyperparameter values within specified ranges and is empirically more efficient than a standard grid-search approach (Sharma, 2018; Bergstra & Bengio, 2012). Moreover, the lower complexity of our engineered features, especially compared to the original data we began with, reduces the gain that a Bayesian method like SMBO or Python's hyperopt (Bergstra et al., 2011; Koehrsen, 2018; Sharma, 2018) would provide, since our models' objective functions are easier to evaluate.

The parameter grids are set through a combination of initial manual searches, adjusted depending on observed results -- such as for XGBoost's learning rate (Brownlee, 2016; Cambridge Spark, 2017; Jain 2016) -- and are meant to be excessive to give the randomized search a greater space to examine due to the feasible size of our data. Cross-validated randomized search results are as follows:

Table 6: Hyperparameter grid for RandomizedSearchCV

Model	**Parameter**	**Range**	**Selected**
DecisionTree	criterion	[Gini, entropy]	entropy
	max_depth	[1..5, 7]	3
	max_features	[auto, sqrt, log2, None]	None
RandomForest	criterion	[Gini, entropy]	entropy
	n_estimators	[30, 40, 50, 60]	60
	max_depth	[2, 3, 4]	4
XGBoost	max_depth	[3, 5]	3
	learning_rate	[0.001, 0.01, 0.025, 0.05, 0.08, 0.1, 0.15]	0.05
	n_estimators	[55, 100, 200, 300, 400]	300

9. Results

9.1 Comparing models with scaled and unscaled data

For comparing our models, we looked at four key factors discussed in **Model Evaluation**:

1. Model accuracy scoring on our holdout "test" dataset, from 2016-2019
2. Weighted Brier scoring to penalize models with lower lift (greater misclassification severity)
3. ROC curves of false positive vs. true positive classification rates, and AUC measures
4. Interpretability of coefficients as a tie-breaker, since we want to understand the determinants of a baseball game more than just predictive power

In general, the models using scaled-difference features performed worse in classification accuracy and Brier score, in addition to interpretability, than the unscaled-percentage-difference ones -- not a product of the scaling but rather the appropriateness of the models to the classification situation. For example, SVC and KNN models are less interpretable naturally than tree-based models like decision trees. This is in line with what Valero (2016) discovered when using KNN-classifiers.

Comparing the ranges here from our testing data, we see that the scale-invariant models almost universally perform better than the scale-sensitive models.

Table 7: Performance comparison between our models that required scaled data vs. models that are scale-invariant

Metric	Scaled models	Unscaled models	Best
Classification accuracy	53.2% - 60.5%	61.2% - 62.77%	**Unscaled models**
Brier score	0.234 - 0.605	0.227 - 0.230	**Unscaled models**
ROC/AUC	0.614 - 0.671	0.666 - 0.6706	Comparable
Interpretability	Low/Medium	Medium/High	**Unscaled models**

The exception to this rule is the stochastic gradient descent classifier (SGDClassifier), which had the highest AUC of all models at 0.6710 and a classification accuracy of 60.5% on the data to the best Brier score of 0.234 out of the scaled models. However, this was on a classifier fit with log-loss, which gives an underlying logistic regression. Since the unscaled logistic regression is more interpretable (directly comparing impacts of percent differences), and the unscaled logit models perform arguably better (mean classification accuracy 62.4%, AUC 0.6706,

and Brier score of 0.227), we discard it in favor of our logit models and other models fit on the unscaled dataset with percentage differences.

9.2 Selecting our final model

Out of these models, the three logistic regressions performed similarly well likely because our dataset did not have high dimensionality. As a result, we selected the Elastic Net out of these variations since it converged fastest empirically on smaller datasets, whereas the Lasso and Ridge did not, which would be problematic if we were training a model using a smaller set of observations (such as only 3-5 instead of 15 seasons). The benefits of the regularization do not appear to be as salient in this problem, since we do not have many feature variables (especially not in excess of our number of observations) for instance, but the regularization nonetheless helps us reduce overfitting (Zou & Hastie, 2005).

The random forest classifier similarly outperformed the decision tree on new data, since the bootstrap random samples and multiple classifiers help increase robustness to overfitting. Similarly, our gradient-boosted XGBoost, particularly after tuning our learning rate parameter through RandomSearchCV, had strong predictive power as expected. So, we make our final evaluation between the logistic regression with elastic net regularization, random forest classifier, and XGBoost classifier.

The majority of our models perform similarly on AUC from the ROC curves, which demonstrate why having multiple metrics to distinguish between them is important.

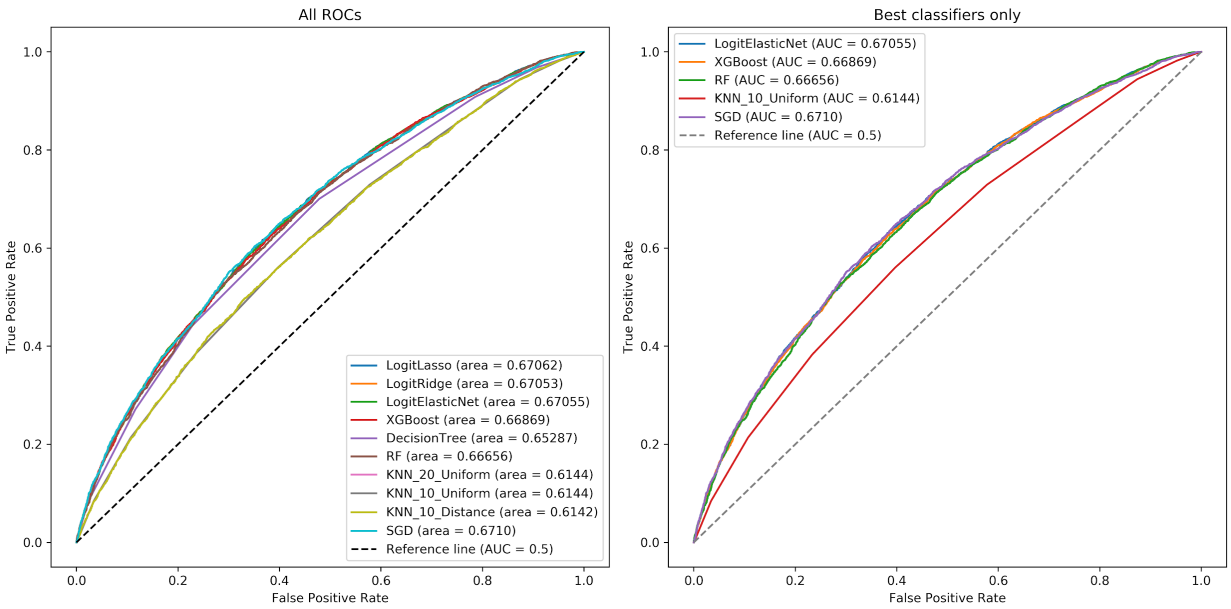

Figure 4: Model ROC curves and AUC comparisons

Out of the elastic net, random forest, and XGBoost, the Brier scores (respectively 0.2270, 0.2290, 0.2274) were very similar but slightly favoring the elastic net. For the sake of interpretability, we select the logistic regression with elastic net regularization as our final "best" model, because its coefficients can directly compute impacts of feature values on the prior forecasted probability of a team (in this case, our home team) winning. This makes it easiest to understand as well as explain to others -- particularly useful in a baseball front office, for instance -- without sacrificing predictive power.

Its mean classification accuracy is 61.77%. The hyperparameters used are:

Table 8: Logit elastic net model hyperparameters

C	CV	Penalty	L1 Ratio
0.01	5	elasticnet	0.5

C is an inverse regularization strength, so picking a lower C (default = 10) results in a stronger regularization and shrinkage of our parameters. We used 5-fold cross-validation in fitting the model, standard stopping criteria and appropriate solver ("saga" to use an elastic net), and an elastic-net mixing L1 ratio of 0.5. These were selected by manual tuning due to the lower number of combinations of hyperparameters.

9.3 Logit elastic net regression analysis

For this model, our coefficients are generally tiny, expected since the coefficients are multiplied against percentages and then used as exponents in the logit function and are additionally regularized.

Table 9: Regularized logit elastic net coefficients

OBPΔ%	ISOΔ%	FIPΔ%	WHIPΔ%	K9Δ%	HR9Δ%	K/BBΔ%	ELOΔ%	restΔ
0.0286	0.0065	-0.0004	-0.0046	0.00099	-0.00069	-0.00075	~0	-0.0106

However, we can only evaluate what the magnitude of the coefficients means through statistical tests, which we do using the statsmodels package.[20] Unfortunately, the statsmodels

[20] Surprisingly, scikit-learn, despite being the default for machine learning analysis in Python, does not have a single means to compute statistical significance of parameters.

package's elastic net regressions and summary tables are (somehow) not implemented, which means we have to examine a regression output of a regularized Lasso, a close comparison (given our earlier model analysis). However, this runs into the same issues that caused us to choose an elastic net; the Lasso logit regression does not converge fast enough, even after parameter tuning. Nonetheless, this proxy model's coefficients and hypothesis tests are below:

Table 10: Regression hypothesis tests for logit elastic net model

	β	**SE(β)**	**z**	**P(Z >|z|)**		**[0.025**	**0.975]**
OBPΔ%	0.0347	0.001	37.591	< 0.001	***	0.033	0.036
WHIPΔ%	-0.0059	0.001	-4.809	< 0.001	***	-0.008	-0.003
K9Δ%	0.0012	0.000	2.495	0.013	*	0.000	0.002
HR9Δ%	0.0003	0.000	0.738	0.461	n.s.	-0.001	0.001
K/BBΔ%	-0.0016	0.000	-5.329	< 0.001	***	-0.002	-0.001
FIPΔ%	-0.0069	0.001	-4.806	< 0.001	***	-0.010	-0.004
ISOΔ%	0.0039	0.000	14.235	< 0.001	***	0.003	0.004
ELOΔ%	-0.0112	0.004	-2.892	0.004	**	-0.019	-0.004
restΔ	0.0042	0.005	0.853	0.394	n.s.	-0.005	0.014

Although many of these small features are statistically significant, we see that OBP is by far the largest coefficient -- suggesting that the biggest determinant of a team winning is not the fancy pitching, or even home run hitters that draw crows to the ballpark, but rather the ability to, as Billy Beane says, "get on base" (Miller, 2011). We compare our top three models' performance on the test dataset (2016-2019) versus a baseline where the home team wins:

Table 11: Final evaluation of model performance

	Accuracy	**Brier Score**	**AUC**
Elastic Net	**61.77%**	**0.2278**	**0.6706**
Random Forest	61.48%	0.2383	0.6657
XGBoost	61.19%	0.2300	0.6687
Naïve	53.16%	0.4684	--
Best Model	Elastic Net	Elastic Net	Elastic Net

Our final model, the logistic regression with elastic net, outperforms a naïve predictor (assuming the home team always wins, accuracy 53.16%) by 8.61 raw percentage points, an increase of 16.2% in predictive power on the holdout dataset. Moreover, computing our Brier Skill Score (Roulston, 2007; DWD, n.d.) against the same naïve forecast's 0.4684 gives a Brier Skill Score of:

$$BSS = \frac{BS_{baseline} - BS_{model}}{BS_{baseline}} = 1 - \frac{BS_{model}}{BS_{baseline}} = 1 - \frac{0.2278}{0.4684} = 0.5137 > 0$$

A positive BSS indicates an improvement in predictive power (DWD, n.d.). Examining these statistics compared to the baseline further validate the strength of our model and viability in predicting baseball games, and selection of the elastic net.

10. Discussion

10.1 Feature importance

For our logistic regression, feature analysis can be evaluated by looking at the strength of the coefficients. Since all of our models are on a percentage basis, a greater coefficient magnitude indicates generally greater impact on the impact of the game (although we do have slight differences in the distribution of our statistics). For XGBoost, we use the "total_gain" metric out of XGBoost's five different options since we are looking for the aggregate prediction importance of each feature (Abu-Rmileh, 2019).

Plotting feature importance for our three models, we find that OBP is consistently the most important feature across all our models, with ISO typically coming in second, in line with Stick (2005) who demonstrates the importance of OPS in predicting productivity indexes.

Notably, the pitching covariates show similar but generally weak feature importance, which is sensible since they are aggregate numbers from the prior season; hence, this does not contradict Stick (2005). We would expect that, if we used live rolling game scores (FiveThirtyEight, 2015), we would see pitching increase in importance, since the information from that feature would be more relevant to each game.

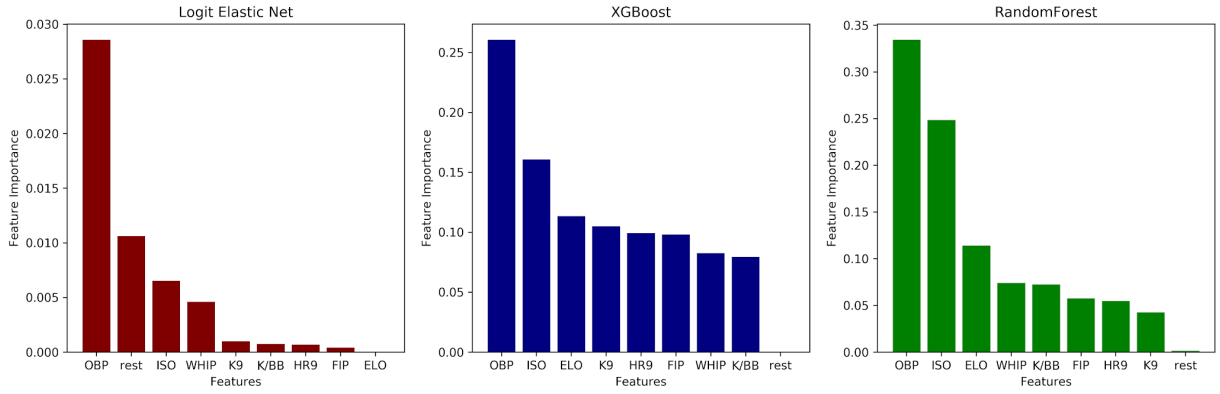

Figure 5: Feature importance for Elastic Net, XGBoost, and Random Forest Models

10.2 Lift

Another evaluation criterion we discussed was "lift" where a model should be better for the most extreme forecasted predictions (Czakon, 2019). We look at "lift" two ways, first through a standard lift chart, and second by bucketing our win probabilities and looking at the conditional chance of being a true positive.

On the left, our logit elastic net model shows around a 1.5-2x greater predictive strength for the most extreme 10-20% of games; on the right, all three models show positive correlation in conditional probability of being true positives, which means we could also act similarly to Pharr (2019) and count on our extreme values being more accurate in either direction.

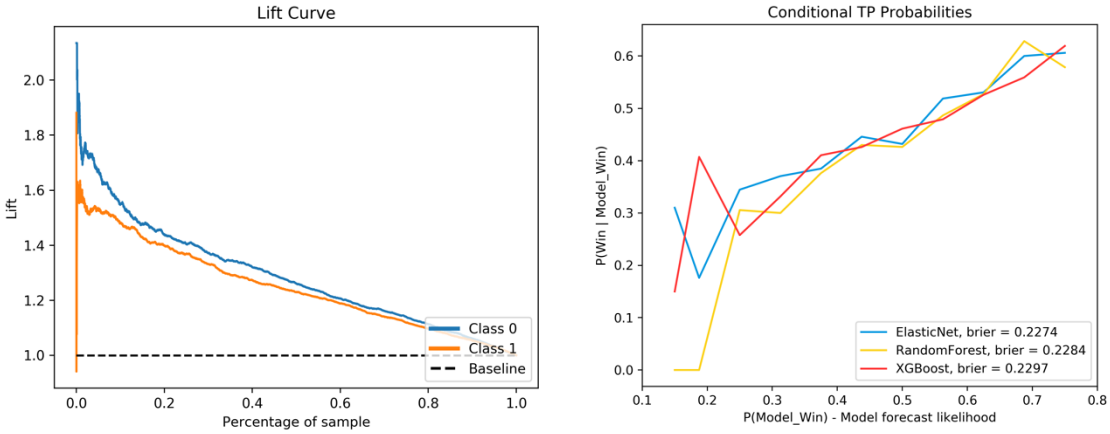

Figure 6: Lift curves and conditional TP plot

10.3 Monthly predictions

Jia et al. (2013) discussed how the beginning of a season is harder to predict due to a lack of data about a team's true performance, which is why they broke up their predictions by month. Similarly, for the 2019 season, we plotted predictive accuracy by month, omitting the few games played in March since we are looking at percentages, and the other months have 20-30 games apiece per team (March usually has no more than 1-3).

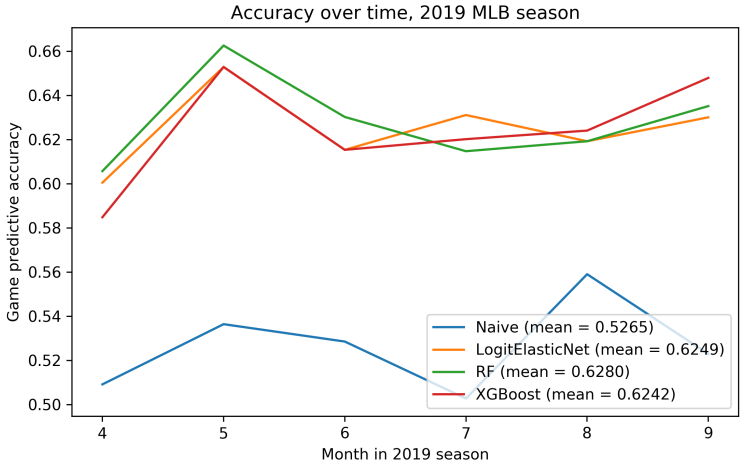

Figure 7: Monthly out-of-sample prediction accuracy (Jia et al., 2003)

This chart validates Jia et al. (2013)'s findings, as we do see a slight increase over time from April to September, an increase from 60.57% in April to 63.52% accuracy in September. However, while their models sat in the 53-57% range (hardly above the baseline) at the beginning of the year, ours begins much stronger, hence the smaller apparent increase. This suggests that our local covariates (7-day rolling periods) are a good fit, as they are more predictive than the season-to-date cumulative measures that Jia et al. (2013) among others use -- which fluctuate more at the beginning of the season.

10.4 Qualitative validation

Briefly, we can also qualitatively examine overlays of our model against prior team-seasons, because we can model individual team-years. Although our model's limitations (discussed below) prevent it from being a perfect fit, and we are only modeling the chance of winning, it stands that we could see some overlap between our probabilities of winning and the actual win-loss scores from Baseball Reference.

In each comparison, the top chart (green/red) is the actual win-loss margin for the team across the entire season, from Baseball Reference; the bottom (in team colors) is our model's forecasted probabilities of the team of interest winning, or $P(y = 1)$. We have to perform a slight adjustment of the model to calculate this, instead of just calculating home team win probabilities. Also, the likelihoods are only meant to compute the home team's expected probability of winning; they are not meant to line up with the margins of victory.

Case Study 1: The 2017 New York Yankees

I am an avid Yankees fan, and loved watching a young team that was not expected to make the playoffs nearly reach the World Series. Hence, this first comparison.

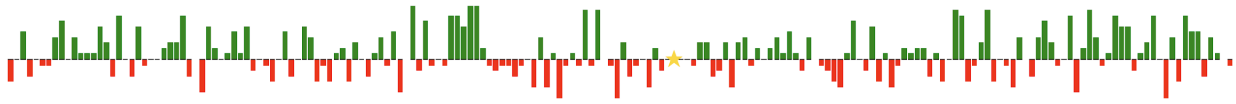

Figure 8a: 2017 New York Yankees W-L chart (Baseball Reference, n.d.-d)

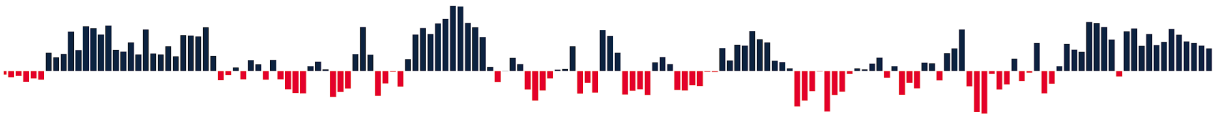

Figure 8b: 2017 New York Yankees model-predicted win probabilities

The model overemphasizes streaks, but generally predicts batches of results well. Since it does not rapidly adjust to game-to-game variations fast enough (likely due to the lack of pitching data, or just inherent variance in baseball), hot streaks for the team led to overestimated wins, with the model predicting 99 wins to New York's 91 -- but almost exactly nailing New York's 100-win Pythagorean W-L projection (Baseball Reference, n.d.-d).

Case Study 2: The 2002 Oakland Athletics

The Oakland A's in 2002 were the first poster child of sabermetrics, proving statistical understanding of the game rather than home runs and financial muscle were key to winning.

Figure 9a: 2002 Oakland Athletics W-L chart (Baseball Reference, n.d.-c)

Figure 9b: 2002 Oakland Athletics model-predicted win probabilities

This comparison makes our model look stellar, primarily due to the Oakland A's enjoyed a 20+ game win. However, looking more closely we see that our model also tracks the team's early struggles, midseason recovery, and faltering towards the playoffs relatively well, giving low probabilities of winning forecasted on the games around the all-star break (the star in the top chart) where Oakland alternated wins and losses by slim margins. Our model projects 104 wins, slightly more than Oakland's actual 103 and its Pythagorean expectation of 96 wins (Baseball-Reference, n.d.-b).

10.5 Limitations

There are several limitations in the analysis that we discuss in order of actionability.

1. We use an aggregate proxy from the pitcher's prior year performance, rather than live-tracking game scores (FiveThirtyEight, 2015)

As previously mentioned, we model a pitcher's performance on some statistic Y in year t, game observation i in the data as distributed $Y_i^t \sim N\left(\overline{Y^{t-1}}, 0\right)$. This simplified our analysis and allowed us to use some level of pitcher information in the models but lacks the granularity we hoped to get. Specifically, players have variance even across seasons, so a pitcher who has a considerably better or worse campaign (either due to young pitchers improving their game, older pitchers declining, injury, etc.) will be mis-modeled over the duration of the season. A rolling game score, or other metrics of pitching performance like the ones we used for hitting data would be most accurate in capturing variance and is far more likely to match professional teams' models.

2. We track team aggregate performance and do not consider day-to-day lineup changes

It is certainly reasonable that a lineup change (for example, replacing a minor-league substitute player with an MVP after a day of rest) would impact a team's chances of winning, which we could capture by including our player-level performance based on what players are in the lineup. Research such as Bukiet et al. (1997) demonstrates potential impacts of batting order optimality, although the complexity of their model makes it infeasible for us to integrate on a game-by-game setting (notably, their batting orders are season-wide and backwards-looking, which runs the risk of data contamination (Valero, 2016) if we take their findings and directly apply them); however, our aggregate features as previously mentioned can reduce variance and signal. Moreover, it is computationally infeasible to track rolling means across all games and over 4.5M instances of player data, given the limited computational resources at our disposal.

3. Variance in the game of baseball

All sports include variance, which makes them fun to watch and coveted to be a part of. As a result, there is a lot of variance due to in-game performance and events that impacts the win-loss outcome of a game, which we cannot capture with prior statistics. In fact, inconsistencies are why "consistent" players are regarded highly, and help explain why the models evaluated in other studies (Jia et al., 2013; Valero, 2016; Koseler & Stephan, 2017; Elfrink & Bhulai, 2018; Pharr 2019) all find predictive results for MLB games in the range of 55-61% predictive power (up to 63-64%, if we consider Jia et al. (2013)'s monthly analysis), which suggests a natural limit to how well baseball games can be forecasted in advance. In fact, our models appear to be at the top end of these accuracy measures.

4. Other minor limitations to take note of, but of lesser severity:

- Use of randomized search (Bergstra & Bengio, 2012) in tuning rather than Bayesian SMBO, which identifies the next hyperparameters to test based on the results of prior cross-validation tests (Koehrsen, 2018; Bergstra et al., 2011); this may select better hyperparameters choices

- Omission of fielding features such as errors, which are removed by FIP but impact the results of the game. Valero (2016) and Pharr (2019) both include differences in errors committed.

- statsmodels lacks functionality for elastic-net/regularized regression statistical test summaries, so we have to assume that, given similar coefficients, the statistical significance carriers over to our regularized model with shrunken weights. This is not fully rigorous.

Altogether, we have built a strong model that matches if not exceeds the top predictors found in the literature through applying a novel idea of rolling statistics. Extensions of our model include adding in these covariates, Bayesian models that adjust weights (similar to Yang and Swartz (2014)) and game distributions as the season goes on, or machine learning analysis to supplement our basic Elo model.

Furthermore, an interesting supplementary analysis could be to build a betting strategy and back-test on the betting lines. This would be not for money, as much for seeing if we beat the concept that Surowiecki calls the "wisdom of the crowd" (2004). For example, we could identify at what confidence cutoffs would our model be able to out-predict Las Vegas spreads, and place positive-EV bets even with the uneven betting lines (a game that has a heavy favorite, for instance, might be less profitable to bet on because the comparative payout is small), as Tait (2014) demonstrates with a Bayesian approach. Succeeding here would be interesting and potentially profitable validation of our modeling approach and decisions.

Conclusion

We set out to understand the factors that influence Major League Baseball game outcomes with a goal to build a predictive model for future games. Using data from over 45,000 games from the 2001-2019 seasons, we engineered detailed covariates for team abilities, such as day-by-day rolling means of hitting performance, creating far stronger granularity than the features seen in the literature. As a result, we achieved classification accuracies above 61% above three models, including our elastic net logistic regression's 61.77% on the 2016-2019 seasons, which exceeds the classification accuracies found by cited works in the literature: Valero (2016), Elfrink & Bhulai (2018), Pharr (2019), Jia et al. (2013), and Soicher (2019).

In addition, we evaluated our model's performance over each month in the 2019 season, confirming the hypothesis by Soicher (2019) and empirical results from Jia et al. (2013), peaking at 64-65% classification accuracy in the latter months of the 2019 season. Our model also demonstrates good lift, where the top decile of predictions for both home wins and losses are accurate around 2x as often as other lower-confidence predictions are. As a result, our model could be leveraged in a baseball front office, at the Las Vegas books, or at home, streaming the next MLB season, when it does resume post-COVID-19.

This analytical process was not perfect, between data wrangling issues and concerns about omitted feature bias. Moreover, there is the broader looming question about from sports analytics: how much variance in baseball game results can be actually captured by prior statistics and features like ours?

Nonetheless, through this our work demonstrates the importance of feature engineering in problems like this, where the aggregate statistics that are easy to pull off of a database -- and used by other works -- explain less variance in results. This traces back to an underrated computer science and problem-solving lesson I learned from Professor Steve Zdancewic on my first day of CIS 120: define and understand the problem before you start. We defined our classification problem, supplemented it with literature and background knowledge gained from watching 100+ MLB games annually, before applying the relevant data science techniques to extract our datasets. The machine learning models are still the pièce de résistance, but they are only possible because of the deep understanding from the earlier sections of this paper, through the ways of thinking and technical skills built up over four years at Penn.

Looking back, I am so grateful to have been given this opportunity by the University of Pennsylvania and the School of Engineering and Applied Sciences to study here, opening intellectual doors I did not know even existed back in 2016.

Thank you to everyone who has made this journey -- culminating with this thesis today -- both possible, and invaluable.

Acknowledgements

This paper is the culmination of four years of study at Penn, and there are many people and groups whose support has been invaluable to this process.

First and foremost, thank you to my thesis advisor, Dr. Shane Jensen, for your modeling ideas, support, and eagerness to talk all things statistics and baseball. It has been an honor and pleasure to work with you and learn from you this semester.

Thank you to the EAS 499 coordinators and team -- Dr. Max Mintz and Desirae Cesar -- for your hard work behind the scenes to make this capstone project, and opportunity to apply my academic knowledge to this field I am so passionate about, possible.

Thank you to my CIS 545 professors -- Dr. Zack Ives and Dr. Lyle Ungar -- for sparking my interest in data science sophomore spring. Dr. Ives -- for giving me the chance to TA for CIS 545, a phenomenal learning experience that helped me fall in love with teaching while also giving me confidence in my data science abilities. Dr. Ungar -- for your help in course advising, suggesting Dr. Jensen as an advisor, and of course, the pizza during in advising sessions.

Thank you to the CIS department's professors, department coordinators, and chairs, have taught me, given me opportunities to learn and teach, and fascinated me with new concepts across data science, internet and web programming, and computer systems.

Thank you to my friends and fellow classmates for inspiring me, supporting me in my tough moments, and pushing me to become my best self here. It's special relationships like these that make it even more bittersweet to realize we are graduating.

Of course, thank you to my family -- Mom, Dad, and Claire -- for your constant support and love. College has been a massive growing-up experience, but I would not have made it here without you all being there, from my shocked sprint down the stairs when I got admitted, to right now as I finish writing this last page under the same roof.

And finally, thank you to the University of Pennsylvania for an incredible four years in Philly. Penn has been my dream school since I stepped foot on campus. Thank you for taking the chance on me on March 31, 2016, and for filling these years with excitement, happiness, and memories. I am to be leaving, but I am proud to be graduating as a Penn alumni.

Appendix

1. Glossary of MLB Terms/Statistics

This is a list of common MLB terms and their definitions. Events that are relevant to pitchers/batters respectively can be identified through cross-matching with the glossary of statistics below. We also define the following statistics that are used extensively in the paper. These definitions are all written using our background knowledge of the sport, or formulas from Baseball Reference (n.d.-e), MLB (n.d.), and Clark (2016) when applicable.

Table 12: Definitions of common MLB terms

Term	Abb.	Definition
Hits	H	A ball put in play where the runner reaches base and the inning does not end.
At-bats	AB	A plate appearance that does not end in a walk or sacrifice hit.
Walks/base on balls	BB	When a pitcher throws four balls in an at-bat that are not called strikes, the batter advances to first base.
Hit by pitch	HBP	A thrown pitch that hits the batter. The batter is granted first base, but this is not counted as a walk.
Sacrifice fly/hit	SF/SH	A ball in play where the runner is called out, but a teammate currently on base either advances or scores.
Home run	HR	A ball that exits the field of play, scoring the hitter as well as all current baserunners.
Total bases	TB	A weighted sum of the bases a hitter reaches. 2B (double), 3B (triple), and HR (home run) are worth 2, 3, and 4 respectively.
Earned run	ER	For pitchers, a run that scores not due to errors or a wild pitch; the runner cannot have reached on an error or wild pitch either.
Innings pitched	IP	3 * the number of outs a pitcher records.

Table 13: Formulas for common MLB statistics

Term	Abb.	Formula	Measures
Batting average	AVG	H/AB	Batter's hitting ability
On-base percentage	OBP	$\frac{H+BB}{AB+BB+HBP+SF}$	Batter's ability to get on base, even without hits

Slugging percentage	SLG	TB/AB	Batter's power and contact hitting
Isolated power	ISO	$SLG - AVG$	Batter's power only
Earned-run avg.	ERA	$9 \cdot ER/IP$	Pitcher's ability to prevent runs
Fielding-independent pitching	FIP	See **7.4**	Pitcher's ability "independent" of team and chance. C_{FIP} calibrated so the league average FIP = average ERA
Strikeout/home run rate	K/9, HR/9	$9 \cdot K/IP$, $9 \cdot HR/IP$	Pitcher's strike out and home run rates
Walks/hits per inning pitched	WHIP	$(HR + BB)/IP$	Pitcher's ability to prevent baserunners (lower = better)
Strikeout-walk ratio	K/BB	K/BB	Pitcher's strikeout to walk ratio.

2. Computation of 2002 Oakland A's wins percentile in MLB history

Since MLB switched to a 162-game season in 1962, we found all team seasons with at least 100 wins (Baseball Almanac, n.d.) and subtracted the number of 103+ win-seasons before 1962 from the total (53; 102 wins is ranked 54th) to get a total of 28 instances. From Lokker (2019)'s article, we can compute the total number of 162-game team-seasons, excluding the strike-shortened 1994* season:

Table 14: Computation of the "Moneyball" Oakland A's percentile

First	Last	Years	Event	Teams	Seasons
1962	1968	7	162-game season	20	140
1969	1976	8	Expansion to 24 teams	24	192
1977	1992	16	American League expands to 14 teams	26	416
1993	1997	4*	National League expands to 14 teams	28	112
1998	2019	22	Expansion to 30 teams	30	660
Grand Total					1520
Number of seasons with 103 or less wins, $\lvert W \leq 103 \rvert$[21]					1504
Percentile, $P(W \leq 103)$					**98.15%**

[21] While there are 28 teams of 103 or more wins, percentiles are calculated as the proportion of occurrences "equal to or lesser than" the given value, so we exclude the 12 teams who also won 103 games.

3. Transcript from Moneyball
(Zaillian & Sorkin, 47-51; Miller, 2011)

Context: Billy Beane (BILLY) and Peter Brand (PETER) are leading a meeting with the Oakland Athletics' scouts, looking to replace players on their roster. Brand has engineered statistical models to forecast W-L records for the team; however, the old-school scouting staff is skeptical. While the scene is dramatized, it illustrates the conflict between computational modeling and intuition-based "knowledge of the game" that demands explanatory power in order to accept the models. This excerpt is scripted as follows (Zaillian & Sorkin, 47-51):

> "BILLY: 'He can get on base.'
> POLONI: 'Alright, so he walks a lot.'
> BILLY: 'He gets on base a lot, Rocco. Do I care if it's a walk or a hit?'
> PETER: 'You do not.'
> BILLY: 'I do not.'
> GRADY: 'Billy, I got 37 free agents who are better than those guys.'
> (...)
> POLONI: 'You're not buying into any of this Bill James bullsh*t are ya?'
> BILLY: 'This is the new direction of the Oakland A's.'"
>
> (Zaillian & Sorkin, 47-51; Miller, 2011)

4. FiveThirtyEight's ELO model for baseball
(FiveThirtyEight 2015)

According to FiveThirtyEight's website, their model tracks each team's underlying team Elo team rating over time. For each game, FiveThirtyEight computes prior likelihoods of winning by adding factors of home-field advantage (+24 points; baseline 1500), days of rest (+2.3 points per day, max of three days), and a penalty for travel ($min(-4, -0.31 * (miles\ traveled)^{1/3})$). They also factor in starting pitcher ratings based on rolling game scores. All this computes a "pregame team rating" that predicts the expected win/loss outcome and probabilities between the two teams. After each game, the Elos are adjusted accordingly (FiveThirtyEight 2015).

The full model can be found on their website, cited in the **Bibliography** and linked here: https://fivethirtyeight.com/features/how-our-mlb-predictions-work

5. Elo model for the Seattle Mariners' 2001 season

This is an example of our Elo model validated across the 2001 Seattle Mariners season. We have chosen this as the example since the Mariners set the all-time record with a 116-46 W-L, and we should see significant changes in their Elo over the course of the season. Accordingly, their Elo quickly spikes as high as 1600. This makes sense for a team that won over 70% of its games -- based on our Elo formula, a Elo of 1596 (the peak observed here around June) would have an expected ~75.12% chance of beating a team with an Elo of 1500, and a 90.5% chance of beating a team with an Elo of 1400.

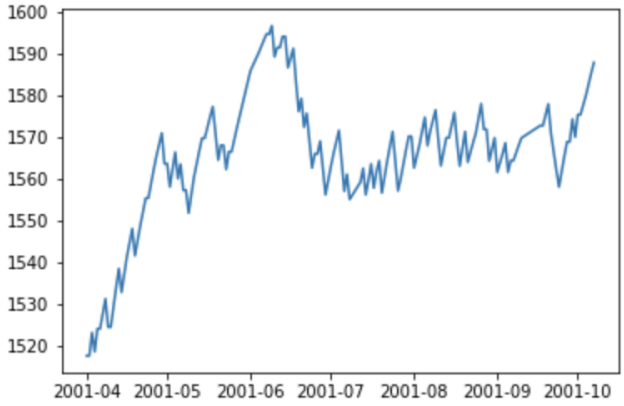

Figure 10: Elo for the Seattle Mariners, 2001

Although a 90.5% chance may be high, we observe from our data that most teams' worst Elo on any given day between March 2001 and October 2019 is at lowest just below 1400; hence, a peak-strength team in the modern era should be strong favorites against a team at its weakest in two decades, should this matchup ever occur. Summary statistics for several teams' Elos across the 19 seasons are shown below.

Table 15: Elo summary statistics

	ANA	ARI	ATL	BAL	BOS	CHA	CHN	CIN	CLE	COL
Mean	1517	1493	1508	1490	1525	1500	1497	1487	1506	1487
Min	1436	1393	1387	1390	1434	1423	1408	1406	1419	1422
Max	1596	1584	1592	1577	1601	1580	1595	1569	1614	1561

The team names here come from our game logs (Retrosheet, n.d.-c). Some of the abbreviations are no longer used since the data goes back to 2000; particularly "ANA" which is now "LAA."

6. Analysis and filtering of starting pitchers

From top to bottom in the graph, these are plotted as the number of pitchers in each year who make 8, 10, 12, 15, 20, and 25 starts in a given season. The standard 5 starting pitchers per team times 30 teams gives around 150 expected starting pitchers; due to injuries and roster changes, we expect there may be slightly more in a given season, setting our filter to a minimum of 12 starts per year for starting pitchers.

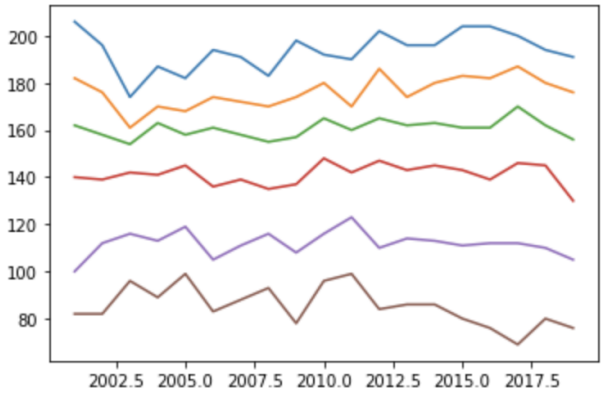

Figure 11: Number of pitchers (descending order) that made 8, 10, 12, 15, 20, and 25 starts per season

Bibliography

Notes and Disclaimers

All code was done in Python (www.python.org) using Jupyter notebooks (jupyter.org). The Git repository can be found at https://github.com/andrew-cui/mlb-game-prediction. However, there are several packages we used, some of which's documentation we reference. They are, with documentation cited below in our bibliography: pandas, numpy, matplotlib, scikit-learn, scikit-plot, statsmodels, xgboost. Our datasets came from Retrosheet game logs/event files (Retrosheet, n.d.-c; Retrosheet, n.d.-d) and Sean Lahman's database (Lahman, n.d.).

Bibliography

Abu-Rmileh, A. (2019, February 8). The Multiple faces of 'Feature importance' in XGBoost. Towards Data Science. https://towardsdatascience.com/be-careful-when-interpreting-your-features-importance-in-xgboost-6e16132588e7

Ajilore, O., & Hendrickson, J. (2007). The Impact of the Luxury Tax on Competitive Balance in Major League Baseball. Northern American Association of Sports Economists, 7(27), 1-13.

Alzubi, O., Alzubi, J., Tedmori, S., Rshaideh, H., & Almomani, O. (2018). Consensus-Based Combining Method for Classifier Ensembles. The International Arab Journal of Information Technology, 15(1), 76-86.

Bailey, S. R., Loeppky, J., & Swartz, T. B. (2020). The Prediction of Batting Averages in Major League Baseball. Stats, 3, 84-93.

Badenhausen, K. (2019, July 22). The World's 50 Most Valuable Sports Teams 2019. Forbes. https://www.forbes.com/sites/kurtbadenhausen/2019/07/22/the-worlds-50-most-valuable-sports-teams-2019/#53f1f44c283d

Baseball Almanac. (n.d.). 100-Win Seasons in Major League Baseball. Baseball Almanac. https://www.baseball-almanac.com/recbooks/100-win-seasons-major-league-baseball.shtml

Baseball Reference. (n.d.-a). 2000 Anaheim Angels Batting Splits. Baseball Reference. https://www.baseball-reference.com/teams/split.cgi?t=b&team=ANA&year=2000

Baseball Reference. (n.d.-b). 2002 Oakland Athletics Statistics. Baseball Reference. https://www.baseball-reference.com/teams/OAK/2002.shtml

Baseball Reference. (n.d.-c). 2016 NL Team Statistics. Baseball Reference. https://www.baseball-reference.com/leagues/NL/2016.shtml

Baseball Reference. (n.d.-d). 2017 New York Yankees Statistics. Baseball Reference. https://www.baseball-reference.com/teams/NYY/2017.shtml

Baseball Reference. (n.d.-e). Baseball statistics. Baseball Reference. https://www.baseball-reference.com/bullpen/Baseball_statistics

Baseball Reference. (n.d.-f). Tie - BR Bullpen. Baseball Reference. https://www.baseball-reference.com/bullpen/Tie

Bergstra, J. S., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research 13, 281-305.

Bergstra, J. S., Bardenet, R., Bengio, Y. & Kégl, B. (2011). Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, et al. (Eds.), Proceedings of the 25th International Conference on Neural Information Processing Systems, NIPS '11 (pp. 2546-2554). Curran Associates. https://dl.acm.org/doi/10.5555/2986459.2986743

Breiman, L. (2001). Random Forests. Machine Learning, 45, 5-32.

Brown, D.T., Link, C.R., & Rubin, S. L. (2017). Moneyball After 10 Years: How Have Major League Baseball Salaries Adjusted? Journal of Sports Economics, 18(8), 771-786.

Brownlee, J. (2016, September 16). Tune Learning Rate for Gradient Boosting with XGBoost in Python. Machine Learning Mastery. https://machinelearningmastery.com/tune-learning-rate-for-gradient-boosting-with-xgboost-in-python

Brownlee, J. (2019, October 28). A Gentle Introduction to Logistic Regression With Maximum Likelihood Estimation. Machine Learning Mastery. https://machinelearningmastery.com/logistic-regression-with-maximum-likelihood-estimation

Bukiet, B., Harold, E. R., & Palacios, J. L. (1997). A Markov Chain Approach to Baseball. Operations Research, 45(1), 14-23.

Bunker, R., & Susnjak, T. (2019). The Application of Machine Learning Techniques for Predicting Results in Team Sport: A Review. Manuscript submitted for publication.

Cambridge Spark. (2017, October 9). Hyperparameter tuning in XGBoost. Cambridge Spark. https://blog.cambridgespark.com/hyperparameter-tuning-in-xgboost-4ff9100a3b2f

Chauhan, N. S. (2019, October 3). Introduction to Artificial Neural Networks (ANN). Towards Data Science. https://towardsdatascience.com/introduction-to-artificial-neural-networks-ann-1aea15775ef9

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Krishnapuram, B., Shah, M. (Eds./Chairs), Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16 (pp. 785-794). Association for Computing Machinery. https://dl.acm.org/doi/10.1145/2939672.2939785

Clark, J. T. (2016). Regression Analysis of Success in Major League Baseball (Publication No. 101) [Senior thesis, University of South Carolina]. https://scholarcommons.sc.edu/senior_theses/101

Couronné, R., Probst, P., & Boulesteix, A-L. (2018). Random forest versus logistic regression: a large-scale benchmark experiment. BMC Bioinformatics, 19(1), 270-284.

Czakon, Jakub. (2019, August 28). The ultimate guide to binary classification metrics. Towards Data Science. https://towardsdatascience.com/the-ultimate-guide-to-binary-classification-metrics-c25c3627dd0a#248c

Davenport, C., & Woolner, K. (1999, June 30). Revisiting the Pythagorean Theorem. Baseball Prospectus. https://legacy.baseballprospectus.com/article_legacy.php?articleid=342

Druschel, H. (2016, February 22). A guide to the projection systems. SBNation: Beyond the Box Score. https://www.beyondtheboxscore.com/2016/2/22/11079186/projections-marcel-pecota-zips-steamer-explained-guide-math-is-fun

DWD. (n.d.). Skill measure: Brier Skill Score. Deutscher Wetterdienst (German Meterological Service). https://www.dwd.de/EN/ourservices/seasonals_forecasts/forecast_reliability.html

Elfrink, T., & Bhulai, S. (2018). Predicting the outcomes of MLB games with a machine learning approach [Unpublished report]. Vrije Universiteit Amsterdam. https://www.math.vu.nl/~sbhulai/papers/paper-elfrink.pdf

Elo, A. E. (1978). The Rating of Chessplayers, Past and Present. Arpo Publishers.

Fangraphs. (n.d.). Scoreboard >> Wednesday, October 30, 2019. Fangraphs. https://www.fangraphs.com/scoreboard.aspx?date=2019-10-30

Ferri, C., Hernandez-Orallo, J., & Modroiu, R. (2009). An experimental comparison of performance measures for classification. Pattern Recognition Letters, 30, 27-38.

FiveThirtyEight. (2015). How Our MLB Predictions Work. FiveThirtyEight. Retrieved April 1, 2020, from https://fivethirtyeight.com/methodology/how-our-mlb-predictions-work

Glickman, M. E., & Jones, A. C. (1999). Rating the chess rating system. Chance, 12, 21-28.

Good Judgement Open. (2019, February 14). Frequently Asked Questions (FAQ): How are my forecasts scored for accuracy? Good Judgement Open. https://www.gjopen.com/faq

Gregorutti, B., Michel, B., & Saint-Pierre, P. (2013). Correlation and variable importance in random forests. Statistics and Computing, 27(3), 659-678.

Haghighat, M., Rastegari, H., & Nourafza, N. (2013). A Review of Data Mining Techniques for Result Prediction in Sports. Advances in Computer Science: an International Journal, 2(5.6), 7-12.

Hastie, T., Tibshirani, R., & Friedman, J. (2008). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer.

Heumann, J. (2016). An improvement to the baseball statistic "Pythagorean Wins." Journal of Sports Analytics, 2(1), 49-59.

Hunter, J., Dale, D., Firing, E., Droettboom, M., & the Matplotlib development team. (2020, March 19). Matplotlib, Release 3.2.1. https://matplotlib.org/Matplotlib.pdf

Hvattum, L. M., & Arntzen, H. (2010). Using ELO ratings for match result prediction in association football. International Journal of Forecasting, 26(3), 460-470.

Ives, Zachary G., Davidson, S. B., Ungar, L. H., Greenberg, C. (2018, September 10). CIS 545: Big Data Analytics - Representing and Manipulating Data. [PowerPoint slides]. University of Pennsylvania. https://drive.google.com/file/d/1w7d8tWuaB5RMoMqDAGX_Zk3KPbrfCOPr/view

Jain, A. (2016, March 1). Complete Guide to Parameter Tuning in XGBoost with codes in Python. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python

James, G., Witten, D., Hastie, T., & Tibshirani, R.. (2013). An Introduction to Statistical Learning - with Applications in R. Springer.

Jensen, S. T. (2020, January 16). STAT 442: Bayesian Statistics - Introduction to Bayesian Statistics [Course lecture]. University of Pennsylvania. https://canvas.upenn.edu/courses/1494969

Jia, R., Wong, C., & Zeng, D. (2013). Predicting the Major League Baseball Season [Unpublished project report; CS 229]. Stanford University.

Koehrsen, W. (2018, June 24). A Conceptual Explanation of Bayesian Hyperparameter Optimization for Machine Learning. Towards Data Science. https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f

Koseler, K., & Stephan, M. (2017). Machine Learning Applications in Baseball: A Systematic Literature Review. Applied Artificial Intelligence, 31(9-10), 745-763.

Lahman, S. (n.d.). Download Lahman's Baseball Database. Sean Lahman. http://seanlahman.com/baseball-archive/statistics

Lahman, S. (2019, May 2). Package 'Lahman.' CRAN: The Comprehensive R Archive Network. https://cran.r-project.org/web/packages/Lahman/Lahman.pdf

Lee, C., & Landgrebe, D. A. (1997). Decision Boundary Feature Extraction for Neural Networks. IEEE Transactions on Neural Networks, 8(1), 75-83.

Li, J., Cheng, J., Shi, J., & Huang, F. (2012). Brief Introduction of Back Propagation (BP) Neural Network Algorithm and Its Improvement. In D. Jin & S. Lin (Eds.), Advances in Computer Science and Information Engineering (2nd ed., pp. 553-558). Springer. https://doi.org/10.1007/978-3-642-30223-7_87

Lokker, B. (2020, April 6). History of MLB Expansion and Franchise Moves. HowTheyPlay. https://howtheyplay.com/team-sports/major-league-baseball-expansion-and-franchise-relocation.

Marcus, D. J. (2000). New table-tennis rating system. The Statistician, 50(2), 191-208.

McKinney, W., & the Pandas development team. (2020, March 18). pandas: power Python data analysis toolkit, Release 1.0.3. https://pandas.pydata.org/docs/pandas.pdf

McWilliams, J. (2019, February 27). The A's won't shy away from the opener and neither will the rest of baseball. The Athletic. https://theathletic.com/842534/2019/02/27/the-as-wont-shy-away-from-the-opener-and-neither-will-the-rest-of-baseball.

Miller, B. (Director). (2011). Moneyball [Film]. Columbia Pictures.

Miller, S., & Rogers, J. (2019, May 21). Why PECOTA projections were so wrong about the Cubs. ESPN. https://www.espn.com/mlb/story/_/id/26788792/why-pecota-projections-were-wrong-cubs

MLB. (n.d.). Isolated Power (ISO). Major League Baseball. http://m.mlb.com/glossary/advanced-stats/isolated-power

Moawad, A. (2018, February 1). Neural networks and back-propagation explained in a simple way. Medium: DataThings Analytics. https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e

Movieclips. [Movieclips]. (2019, July 28). Moneyball (2011) - The New Direction Scene (4/10) [Video]. YouTube. https://www.youtube.com/watch?v=AKXKkeLQWac

Nakano, R. S. (2018, August 19). Scikit-plot Documentation. https://readthedocs.org/projects/scikit-plot/downloads/pdf/stable

Natekin, A., & Knoll, A. (2013, December 4). Gradient boosting machines, a tutorial. Frontiers in Neurobiotics. https://www.frontiersin.org/articles/10.3389/fnbot.2013.00021/full

NumPy Reference, Release 1.17.0. (2019, July 26). https://docs.scipy.org/doc/numpy/numpy-ref-1.17.0.pdf

Park, W. I., & Garcia, P. (1994). Aggregate versus Disaggregate Analysis: Corn and Soybean Acreage Response in Illinois. Review of Agricultural Economics, 16(1), 17-26.

Pharr, R. D. (2019, August 3). Predicting MLB Game Outcomes with Machine Learning. Towards Data Science. https://towardsdatascience.com/predicting-mlb-game-outcomes-with-machine-learning-594eac9484e9

Retrosheet. (n.d.-a). The Event File. Retrosheet. https://www.retrosheet.org/eventfile.htm

Retrosheet. (n.d.-b). Game Logs Fields. Retrosheet. https://www.retrosheet.org/gamelogs/glfields.txt

Retrosheet. (n.d.-c). Game Logs, Updated December 10, 2019. Retrosheet. https://www.retrosheet.org/gamelogs/index.html

Retrosheet. (n.d.-d). Play-By-Play Data Files. Retrosheet. https://www.retrosheet.org/game.htm

Roulston, M. S. (2007). Performance targets and the Brier score. Meteorological Applications, 14, 185-194.

Schiff, A. (2008). Henry Chadwick. Society for American Baseball Research. https://sabr.org/bioproj/person/436e570c

Scikit-Learn User Guide, Release 0.22.2. (2020, March 4). https://scikit-learn.org/stable/_downloads/scikit-learn-docs.pdf

Shah, T. (2017, December 6). About Train, Validation and Test Sets in Machine Learning. Towards Data Science. https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

Sharma, A. (2018, November 3). Algorithms for hyperparameter optimisation in Python. Towards Data Science. https://towardsdatascience.com/algorithms-for-hyperparameter-optimisation-in-python-edda4bdb167

Sharma, N. (2020, April 1). Understanding the Mathematics Behind Decision Trees. Heartbeat by Fritz AI. https://heartbeat.fritz.ai/understanding-the-mathematics-behind-decision-trees-22d86d55906

Slowinski, S. (2010, February 15). FIP. Fangraphs. https://library.fangraphs.com/pitching/fip

Smola, A., & Vishwanathan, S. V. N. (2008). Introduction to Machine Learning. Cambridge University Press.

Soicher, P. (2019, May 12). Can you accurately predict MLB games based on home and away records? Towards Data Science. https://towardsdatascience.com/can-you-accurately-predict-mlb-games-based-on-home-and-away-records-8a9a919bad29

Statsmodels, v0.11.1. (2020) https://www.statsmodels.org/stable/index.html

Stick, J. D. (2005). A regression analysis of predictors on the productivity indices of Major League Baseball: 1985-2003 (Publication No. 1) [Doctoral dissertation, University of Nebraska]. https://digitalcommons.unl.edu/cehsdiss/1

Surowiecki, J. M. (2004). The Wisdom of Crowds. Doubleday.

Tait, J. R. (2014). Building a Predictive Model for Baseball Games (Publication No. 382) [Master's thesis, Minnesota State University]. https://cornerstone.lib.mnsu.edu/etds/382

Tolbert, B., & Trafalis, T. (2016). Predicting Major League Baseball Championship Winners through Data Mining. Athens Journal of Sports, 3(4), 239-252.

Tylicki, D. (2011, November 21). The 50 Greatest Individual Player Seasons in Baseball History. Bleacher Report. https://bleacherreport.com/articles/946387-the-50-greatest-individual-player-seasons-in-baseball-history

UCI. (n.d.). Iris Data Set. UCI Machine Learning Repository; Center for Machine Learning and Intelligent Systems. http://archive.ics.uci.edu/ml/datasets/iris

Valero, C. S. (2016). Predicting Win-Loss outcomes in MLB regular season games – A comparative study using data mining methods. International Journal of Computer Science in Sport, 15(2), 91-112.

van Wieringen, W. (n.d.). Lasso regression [PowerPoint slides]. VU University. http://www.few.vu.nl/~wvanwie/Courses/HighdimensionalDataAnalysis/WNvanWieringen_HDDA_Lecture56_LassoRegression_20182019.pdf

Vu, K., et al. (2019). The index lift in data mining has a close relationship with the association measure relative risk in epidemiological studies. BMC Medical Informatics and Decision Making, 19(112), 1-8.

Vuk, M., & Curk, T. (2006). ROC Curve, Lift Chart and Calibration Plot. Metodoloski zvezki, 3(1), 89-108.

XGBoost Python API Reference. (n.d.). https://xgboost.readthedocs.io/en/latest/python/python_api.html#module-xgboost.sklearn

XGBoost, Release 1.1.0-SNAPSHOT. (2020, April 27). https://buildmedia.readthedocs.org/media/pdf/xgboost/latest/xgboost.pdf

Yang, T. Y., & Swartz, T. (2014). A Two-Stage Bayesian Model for Predicting Winners in Major League Baseball. Journal of Data Science, 2, 61-73.

Zaillian, S. & Sorkin, A. K. (2011). Moneyball. The Internet Movie Script Database. https://www.imsdb.com/scripts/Moneyball.html

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society, 67(2): 301-320.