CSC 2515 Final Project

Marcus Uhthoff
MScAC - Data Science
University of Toronto
marcus.uhthoff@mail.utoronto.ca

Steven Tsai
MI - Information System & Design
University of Toronto
hungyi.tsai@mail.utoronto.ca

1 Abstract

In this paper, we apply machine learning techniques to predict the outcomes of Major League Baseball (MLB) games. This paper aims to take a novel approach to the task by addressing two potentially erroneous assumptions predominant in other attempts on this subject. Other approaches have used team-level statistics and treated all historical games equally. By recording time weighted averages of player specific statistics over a rolling window of games our methodology aims to incorporate recent performance trends and individual player contributions to improve predictive accuracy. We evaluate four models Logistic Regression, Support Vector Machines (SVM), XGBoost, and Neural Networks performance on this proposed approach. Results demonstrate that while player-level data does not offer improved predictive performance to team-level data, the use of weighted averages can improve accuracy across models.

2 Introduction

2.1 Background

The world of sports is an ideal domain for the application of Machine Learning. Indeed, today, there are already countless companies within the industry leveraging machine learning. This includes sports teams seeking to gain competitive advantages over competitors, the rapidly growing sports-betting platforms setting odds on everything from game outcomes to the color of the gatorade a team will have on the bench, broadcasting companies creating more in-depth analysis of the games, and even fans.

Sports naturally lends itself well to machine learning, as at its core it can be thought of as a microcosm of many real world systems. While sports are governed by set rules, the games are naturally extremely chaotic with high levels of uncertainty. Where there are countless variables that determine the outcome of any single action or event. Furthermore, there are rich, detailed, and structured datasets that stretch back years.

2.1.1 Major League Baseball

In this paper, we will focus specifically on **Major League Baseball**, which is the highest level of professional baseball in North America. We chose this sport and league for a few reasons.

MLB has the most games played per year of all the professional sports leagues in North America[1]. Additionally, the league has invested heavily to implement Statcast, an automated system to collect thousands of datapoints from every single game.

[1] there are 2,430 MLB games played each year, compared to 1,312 in the NHL, 1,230 in the NBA, and 272 in the NFL

We believe the sport of baseball also lends itself well to machine learning as the game is very structured and consists of repeated events (e.g., innings, at-bats, and pitches). The statistics can largely be grouped into two categories, offense and defense. Where offensive stats are derived from a player's batting performance and defensive stats can be characterized by the pitcher's performance. This creates a much more defined set of quantifiable statistics then other sports where there is much more interaction between players and difficult to measure events.

2.2 Prior work

While predicting the outcome of sports may not be a space with as much academic investment as other areas, there is still a rich history of attempting to predict outcomes of events and competitions. In fact, there is evidence of 'sports' betting (which inherently requires an individual to make a prediction on the outcome of an event) as far back as 4000 BC by ancient Egyptians [2].

More recently we have seen attempts to predict the outcome of sports by implementing contemporary ML methods [5] [3] [1]. One major driver for this is the sports betting industry, however there are usecases beyond this. Indeed, sports are a great environment to test new ML models as they generally are very rule based, produce repeatable trials, are complex, and have existing infrastructure to collect and process the data. A few examples of prior work in sports predictions are:

1. **Exploring and Selecting Features to Predict the Next Outcomes of MLB Games**: A 2022 paper comparing potential models and features to predict the outcome of MLB games found that of the tested models a support vector machine (SVM) with feature engineering produced the best results (65% accuracy) [5].
2. **Predicting the outcomes of MLB games with a machine learning approach**: A 2018 paper attempting to predict the outcome of MLB games using in season statistics and recent events (e.g., weather). The team achieved an accuracy of 55% using a XGBoost [1].
3. **Machine Learning in Football Betting: Prediction of Match Results Based on Player Characteristics**: a 2020 paper predicted the outcome of European soccer (football) leagues using a collection of models. They measured the performance of their model by measuring how much profit the model would have achieved betting on games. Their final solution achieved a 1.58% return [6].
4. **Exploiting sports-betting market using machine learning**: A 2019 paper predicting NBA outcomes using convolutional neural networks and logistic regression. The team applied a novel approach of encouraging the model to decouple with bookkeepers' odds rather than just predicting outcome. The goal of this approach was to create a more 'profitable' model that identified instances it believed the the odds were incorrect, and so would have a higher expected value on a bet[4].

In fact, Horvat and Job created a study analyzing over 100 applications of ML to sports, as might be expected given the inherent randomness of baseball, in their review, baseball had the lowest maximum achieved accuracy [3].

2.3 Problem Definition

While there has been research in the space as evidenced in the previous section. These approaches have largely taken a 'naive' approach to the challenge. Choosing to rely on the models and statistical approaches to identify the key features, learn the problem, and make predictions. While this is a reasonable approach, and removes potential bias of the researcher, it also means that there is potentially contextual/ domain knowledge that is not leveraged.

In our estimation, the biggest two shortcomings of the previous work are:

1. Previous models have taken team level statistics as the input rather than player level statistics. This has a potential major flaw when looking at pitching statistics, as teams will rotate between up to 5 pitchers for every game. Additionally, only measuring the average batting performance of the team does not capture the variance between individuals performance.

2. Previous work created the feature space for a given game as the simple average of all the previous games for the given team in that season. This means that on the 162nd game of a season, the statistics of the first game is weighted equally to the 161st game. Which we believe is potentially an incorrect assumption, as we expect more recent games to have a higher correlation to the outcome of a given game due to factors including injuries, player form, different rosters due to trades, and more generally the 'momentum' of the team.

And so through this paper we propose building our dataset and models without making these 2 generalizations.

3 Approach

3.1 Dataset

To collect the necessary data we leveraged the statcast API. This API connects to the MLB's Statcast system which captures extremely detailed pitch level data from every single game played.

From this highly granular dataset we built our dataset. First we grouped on the game ID's to create a list of all the games played in a given season. Then from the original statcast data we prepared two tables. One which tracked all the player level batting data for each team in every game, and one that tracked all the player level pitching data for each team in every game. Table 1 contains a summary of the batter statistics collected and Table 2 the pitcher statistics. It should be noted that for a given game there are 9 batters per team and 1 pitcher per team. Which means that we have 60 inputs per game.

We then identified the last M games for each team given the current game. And then joined the games table with the batting and pitching data on each team's previous M games. This effectively created our data set, where every row contains the outcome (home_team win or loss) and then the stats for the players on each of the 2 teams playing in the game from their respective previous M games.

Table 1: Collected statistics for every batter in each game

Statistic	Name	Formula	Description
BA	Batting Average	$\frac{H}{AB}$	Percent of official at-bats that result in a hit
OBP	On-Base Percentage	$\frac{H+BB+HBP}{AB+BB+HBP+SF}$	Percentage of time a batter successfully reaches a base
SP	Slugging Percentage	$\frac{TB}{AB}$	Number of bases a player earns per at-bat

Table 2: Collected statistics for starting pitchers in each game

Statistic	Name	Formula	Description
ERA	Earned Run Average	$\frac{ER}{IP} * 9$	Average runs allowed over 9 innings
K9	Strikeouts per 9 Innings	$\frac{K}{IP} * 9$	Average strikeouts per 9 innings
WHIP	Walks & Hits	$\frac{W+H}{IP}$	Average number of walks and hits a per inning pitched [2]

This method generated around 2,430 samples per season. We then augmented the dataset by switching the statistics for the players on the home and away team, as well as flipping the outcome of the game. Creating a synthetic copy of a real game in which the home and away team switched places. We feel that this is an acceptable approach since previous research has found that there is no statistically significant home field advantage in the MLB [3].

[2]Detailed explanations of the stats abbreviations can be found in Appendix A: Glossary of Batting and Pitching Stats.

3.2 Proposed Methodology

To address the two hypothesized inaccurate assumptions. We propose the following approach.

Firstly, we leverage the player level statistics rather than team level statistics for every game. Our argument for this decision is that the increased granularity in the data will improve the signal. As an example, if we have two teams with the same average hitting statistics. Except team 1 has one player that hits very well, with the rest of the team hitting poorly, while team 2 has all of its players hitting average. We would expect team 2 to have better results, since typically in baseball to score points requires multiple batters to hit the ball in succession to advance the players around the bases, and score runs. Additionally, as teams rotate between 5 starting pitchers, when collecting pitching data from previous games we only looked at games in which the starting pitcher for the given game also pitched. As starting pitchers are announced well in advance of the start of a game, we believe it is reasonable to assume that the starting pitchers for the game we are trying to predict are known.

Our second proposed change is to create a rolling window of historic games to aggregate when predicting the outcome of a given game.We chose to test this approach using a few different models. For each model we performed 5-fold cross validation on 70% of the dataset to identify the optimal combination of parameters and then tested the selected model configuration on the reserved 30% of the data as a test set.

3.3 Selected Models

To ensure, that our conclusions would be model agnostic, we tested the effect of the proposed approach on 4different models. For each model we tuned the hyperparameters using a grid search approach on the baseline dataset.

3.3.1 Logistic Regression

Logistic Regression is an extremely popular model used in binary classification. It is a fairly quick model to fit and perform predictions, which allows us to efficiently determine the optimal hyperparameters. The selected hyperparameters for this model are listed in Table 3

Table 3: Logistic Regression Hyper Parameters

Parameter	Options	Description	Selected Value
Penalty	L1, L2	Type of regularization prenalty to use	L1
C	1, 50, 100, 1000	Controls the strength of regularization	1
Solvers	Liblinear, newton-cg, lbfgs, sag	Optimization algorithm	liblinear

3.3.2 Support Vector Machine (SVM)

The SVM classifier demonstrated the best performance among the four models evaluated by Li et al.[5]. This classifier is particularly effective due to its ability to identify the optimal decision boundary that maximizes the margin between the two classes (home team win/loss). Particularly, we used the RBF (Radial Basis Function) kernel to capture non-linear relationships between the input features. The selected hyperparameters for this model are listed in Table 5

Table 4: Support Vector Machine Hyper Parameters

Parameter	Options	Description	Selected Value
C	0.1, 1, 10, 100	Controls the strength of regularization	1
gamma	'scale', 'auto', 0.1	Controls the influence range of individual points	'auto'

3.3.3 Extreme Gradient Boosting (XGBoost)

XGBoost excels at capturing non-linear relationships and feature interactions, making it ideal for the complex interdependencies in batter and pitcher statistics. Additionally, XGBoost's optimized implementation, including tree pruning and parallel processing, ensures efficient training and evaluation, even with large datasets. The selected hyperparameters for this model are listed in Table 5

Table 5: XGBoost Hyper Parameters

Parameter	Options	Description	Selected Value
n_estimator	50,100,200	Number of trees in the model	100
max_depth	3,5,10	Maximum depth of each tree, controls model complexity	3
learning_rate	.01,.1,.2	Step size for weight updates, balances speed and performance	0.1
subsample	.6,.8,1	Fraction of training data used for each tree	1
colsample_bytree	.6,.8,1	Fraction of features used for each tree	1
gamma	0,1,5	Minimum loss reduction required to split a node	5
min_child_weight	1,3,5	Minimum sum of instance weights for a split	1

3.3.4 Neural Network

Neural Networks have become one of the most commonly used predictions models due to their ability to capture complex and non-linear relationships between inputs and the target variable. Due to the computational intensity of training the model, hyperparameters were selected in series rather than through a full grid search, the selected hyper parameters are shown below in Table 6

Table 6: Neural Network Hyper Parameters

Parameter	Options	Selected Value	Comments
# of Hidden Layers	1, 3, 5	5	Determined empirically
Nodes per Hidden Layer	10, 50, 100, 200, 300	100	Determined empirically
Dropout	0, 0,05, 01, 02, 05	0.2	Determined empirically
Epochs	10, 50, 75, 150, 250	75	Balanced performance and time tradeoff
Batch size	16, 32, 64, 128	64	Balanced performance and time tradeoff
Learning Rates	1e-6, 1e-5, 1e-4, 1e-3	1e-4	Balanced performance and time tradeoff

4 Results

To evaluate the models performance we have chosen accuracy as the primary metric. The accuracy of the predictions is calculated simply as:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

This metric was chosen based on the inherent characteristics of the model. Firstly, in the context of predicting the outcome of a game, the misclassification cost of a False Positive (FP) or a False Negative (FN) are the same. This is not necessarily always true, indeed look at application areas in healthcare where a False Negative can be, in some cases, life threatening. For this reason metrics like precision or recall were not prioritized. We also considered the area-under-the-curve of the ROC curve. While the dataset was perfectly equal between home team wins and losses (recall we augmented our data by duplicating all rows and switching home and away team stats). The AUC metric can help confirm that our model is performing the classification task well.

4.1 Baseline

To create a baseline performance metric each of the proposed models were evaluated on a baseline dataset. This dataset was built as the rolling average of all previous games within the season for a

given player statistic and team.

$$z_i^{(N)} = \frac{1}{N-1} \sum_{j=1}^{N-1} x_i^{(j)}$$

where $z_i^{(N)}$ is the rolling average for the i$^{\text{th}}$ statistic in the N^{th} game and $x_i^{(j)}$ is the i$^{\text{th}}$ statistc for the j$^{\text{th}}$ game. This approach yielded the following accuracies on the test set:

Table 7: Baseline Model Test Predictions

Model	Test Accuracy	Test AUC
Logistic Regression	54.78	0.5479
Support Vector Machine	55.17	0.5604
XGBoost	55.66	0.5793
Neural Network	54.11	0.5655

4.2 Simple Average of Rolling Window

To determine the optimal window of previous games to consider when predicting the outcome of the current game we employed an empirical sweep and 5-fold cross validation of possible window sizes for each of our selected models. For a window of size M we calculate our features as:

$$z_i^{(N)} = \frac{1}{M} \sum_{j=N-M}^{N-1} x_i^{(j)}$$

To ensure we maintained the maximum number of samples in our dataset, for games with insufficient previous games (i.e., the 2nd game of a season for a team). We considered all of their previous games from that season. We tested values for M from 1 to 20, due to computational constraints.

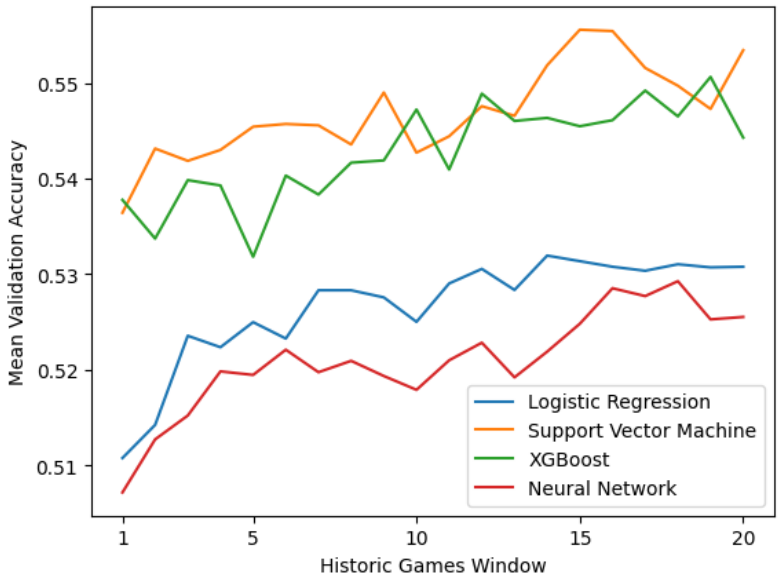

Figure 1: Accuracy of Selected Models based on historic window size

At each of the identified optimal window sizes the test accuracy and AUC for each model was then evaluated with the optimal window being between 14-19 games. Suggesting that a relatively larger window is beneficial.

Table 8: Rolling Simple Average Model Test Predictions

Model	Optimal Window Size	Accuracy	AUC
Logistic Regression	14	53.77	0.5370
Support Vector Machine	15	54.70	0.5530
XGBoost	19	55.91	0.5566
Neural Network	18	50.27	0.5094

4.3 Temporal Weighting

To create a temporally weighted-average we first fixed the historic window for each model based on the optimal result from the simple average approach described in the section above. Then we recalculated the average for each statistic as:

$$z_i^{(N)} = \sum_{j=N-M}^{N-1} \sigma(j) \cdot x_i^{(j)} \qquad \text{,where } x_i^{(j)} \text{ is the i}^{\text{th}} \text{ statistic in the j}^{\text{th}} \text{ most recent game}$$

and the $\sigma(\cdot)$ function defined as an extension of the softmax function:

$$\sigma(j) = \frac{e^{(N-j)\cdot t}}{\sum_{k=N-M}^{N-1} e^{k \cdot t}}$$

Where t is the temperature coefficient and j represents the j$^{\text{th}}$ game of the season. This creates normalized exponentially decaying weights with lower weights on games that are less recent.

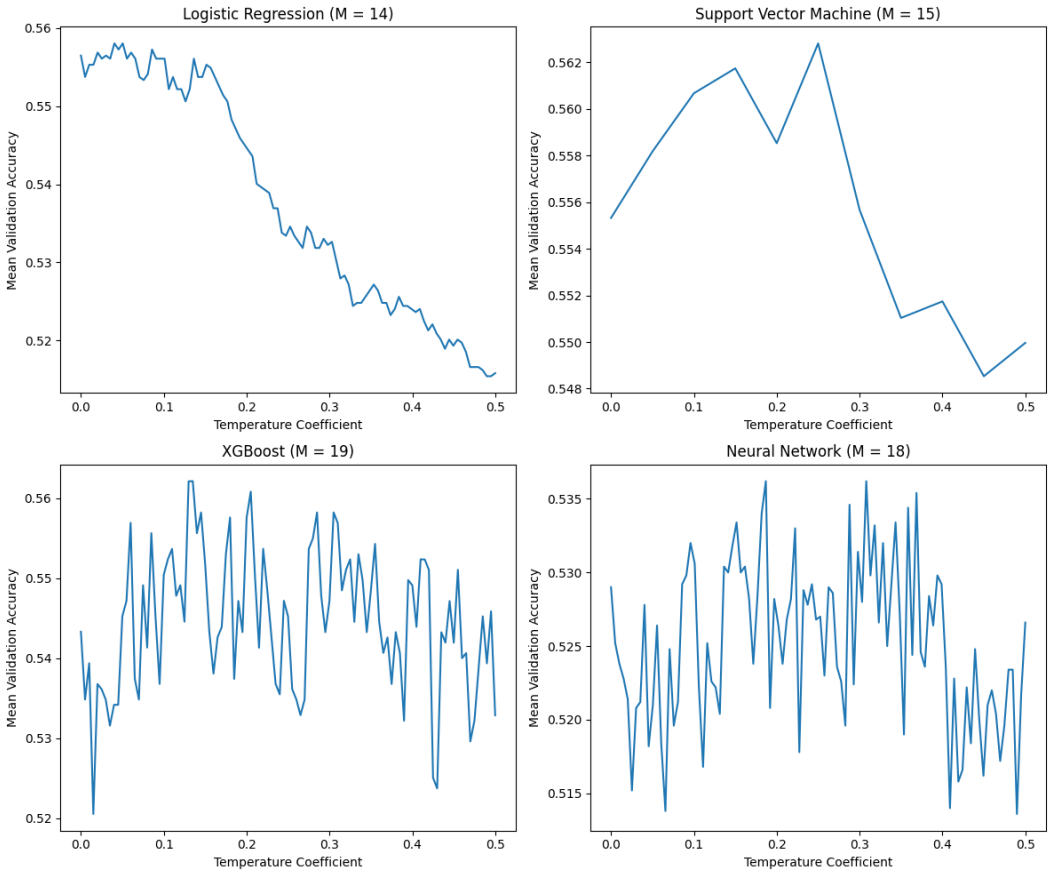

Figure 2: Mean Validation Accuracy for different Temperature Coefficients

Based on Figure 2 we can see that the temperature coefficient had a very noticeable trend on the logistic regression and SVM models and a less clear trend on the other two models. However, the test results at the optimal temperature coefficient show improvement over the simple average for all models as seen in Table 9.

Table 9: Rolling Weighted Average Model Test Predictions

Model	Games Window	Temperature	Accuracy	AUC
Logistic Regression	14	0.04	55.24	0.5526
Support Vector Machine	15	0.25	56.41	0.5584
XGBoost	19	0.135	55.91	0.5465
Neural Network	18	0.15	52.48	0.5148

5 Conclusion

Based on the results of the experiments outlined in this paper we can reflect on the validity of the 2 hypotheses we tested. Firstly, based on comparing the achieved accuracy rates of our models with existing research. We see that collecting statistics at the player level does not seem to improve the accuracy of the model. It should be noted that the models were trained on datasets from different time periods which will affect results and that it is unclear whether the reported accuracies by Li et al are validation or test accuracies [5]. Secondly, we showed that while taking the simple average of statistics over a reduced window did not consistently improve accuracy, applying a time weighted average on this window improves accuracy over the baseline on average by 1% and over the simple weighted average by 2%. All models had a similar optimal window between 14-19 games while the temperature coefficient varied from 0.04-0.15. While these are small percentage increases, in a domain where the state-of-the-art models achieve only around 65% an improvement of a few percentages is meaningful.

Future extensions of this work could include evaluating performance with larger time windows and decoupling the size of rolling windows for pitchers and batters to better capture their distinct contributions. Additional improvements could involve incorporating more granular statistics, such as exploring the covariance between different batters, as well as interactions between batters and opposing pitchers. Moreover, integrating non-player-related statistics, such as weather data, injury reports, and sentiment analysis of social media posts related to the game, could enhance prediction accuracy. Finally, moving from hard predictions (win/lose) to soft predictions, like score differential, could offer more nuanced insights into the factors influencing game results.

Table 10: Model Performance across all datasets [3]

Model	Metric	Baseline	Simple Average	Weighted Average	Previous Work[5] [4]
Logistic Regression	Accuracy	54.78	53.77	**55.24**	*56.17*
	ROC-AUC	0.5479	0.5370	0.5526	*0.5465*
SVM	Accuracy	55.17	54.70	**56.41**	*65.92*
	ROC-AUC	0.5604	0.5530	0.5584	*0.6510*
XGBoost	Accuracy	55.66	**55.91**	**55.91**	*55.52* [1]
	ROC-AUC	0.5793	0.5566	0.5465	
Neural Network	Accuracy	**54.11**	50.27	52.48	*53.70*
	ROC-AUC	0.5655	0.5094	0.5148	*0.5709*

[3] Best accuracy across datasets produced in this paper for each model have been bolded

[4] Note the quoted paper was unclear whether these were validation of test scores

References

[1] Tim Elfrink. Predicting the outcomes of mlb games with a machine learning approach. *Vrije Universiteit Amsterdam*, 552, 2018.

[2] Becky Harris, John T. Holden, and Gil B. Fried. *The Business of Sports Betting*. Human Kinetics Publisher, 2024.

[3] Tomislav Horvat and Josip Job. The use of machine learning in sport outcome prediction: A review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(5):e1380, 2020.

[4] Ondřej Hubáček, Gustav Šourek, and Filip Železný. Exploiting sports-betting market using machine learning. *International Journal of Forecasting*, 35(2):783–796, 2019.

[5] Shu-Fen Li, Mei-Ling Huang, and Yun-Zhi Li. Exploring and selecting features to predict the next outcomes of mlb games. *Entropy*, 24(2), 2022.

[6] Johannes Stübinger, Benedikt Mangold, and Julian Knoll. Machine learning in football betting: Prediction of match results based on player characteristics. *Applied Sciences*, 10(1), 2020.

A Glossary of Batting and Pitching Stats

The key abbreviations and metrics used in the batting and pitching statistics, as introduced in Section 3.1, are outlined and explained below:

- **Batting stats:**
  - **H**: Hit = single + double + triple + homerun
  - **AB**: At bat
  - **BB**: Base on balls
  - **HBP**: Hit by pitch
  - **SF**: Sacrifice flys
  - **TB**: Total bases $=$ single $+ (2 \times$ double$) + (3 \times$ triple$) + (4 \times$ homerun$)$
- **Pitching stats:**
  - **ER**: Earned runs
  - **IP**: Innings pitched
  - **K**: Strike outs
  - **W**: Walks
  - **H**: Hits

B Model Results

This appendix contains the results of the 5 fold cross validation accuracies and AUC scores for each of the models on the three steps covered in the paper.

B.1 Logistic Regression

Table 11: 5-fold cross-validation results for the optimal Logistic Regression model

	Baseline		**Simple Average**		**Weighted Average**	
Fold	**Accuracy (%)**	**AUC**	**Accuracy (%)**	**AUC**	**Accuracy (%)**	**AUC**
1	54.52	0.5565	53.81	0.5339	53.13	0.5360
2	53.61	0.5574	54.12	0.5418	57.42	0.5738
3	53.78	0.5445	54.38	0.5479	59.18	0.5905
4	54.45	0.5633	53.83	0.5393	54.88	0.5489
5	55.16	0.5692	54.84	0.5450	54.40	0.5441
Average	54.30	0.5582	54.20	0.5416	55.80	0.5587

B.2 SVM

Table 12: 5-fold cross-validation results for the optimal SVM model

	Baseline		Simple Average		Weighted Average	
Fold	**Accuracy (%)**	**AUC**	**Accuracy (%)**	**AUC**	**Accuracy (%)**	**AUC**
1	54.29	0.5496	54.57	0.5357	56.51	0.5362
2	56.00	0.5703	56.00	0.5636	55.26	0.5349
3	54.29	0.5588	56.71	0.5611	55.18	0.5537
4	53.00	0.5409	55.36	0.5656	57.68	0.5487
5	53.21	0.5447	55.14	0.5440	56.79	0.5424
Average	54.16	55.29	55.56	55.40	56.28	54.32

B.3 XGBoost

Table 13: 5-fold cross-validation results for the optimal XGBoost model

	Baseline		Simple Average		Weighted Average	
Fold	**Accuracy (%)**	**AUC**	**Accuracy (%)**	**AUC**	**Accuracy (%)**	**AUC**
1	54.69	0.5654	54.40	0.5513	56.49	0.5433
2	54.98	0.5689	54.48	0.5550	54.87	0.4930
3	53.39	0.5513	54.13	0.5580	54.07	0.5297
4	55.60	0.5751	53.85	0.5594	58.31	0.5629
5	54.81	0.5728	55.28	0.5631	57.33	0.5604
Average	54.69	56.67	54.43	55.73	56.21	53.78

B.4 Neural Network

Table 14: 5-fold cross-validation results for the optimal Neural Network model

	Baseline		Simple Average		Weighted Average	
Fold	**Accuracy (%)**	**AUC**	**Accuracy (%)**	**AUC**	**Accuracy (%)**	**AUC**
1	53.84	0.5556	52.97	0.5282	53.26	0.5489
2	54.33	0.5589	52.88	0.5348	54.79	0.5591
3	54.89	0.5672	52.60	0.5328	54.11	0.5446
4	53.93	0.5551	53.69	0.5413	52.23	0.5197
5	54.80	0.5613	52.50	0.5245	55.40	0.5392
Average	54.36	0.5596	52.98	0.5323	53.96	0.5423

C Division of Work

Below is a table listing how the team divided work to complete the project. While the table lists which member owned and drove each part of the work, no part of this project was done in complete isolation and both members contributed thoughts, ideas, and solutions to every section beyond just the ones they are listed as owning.

Table 15: Division of Work

Section	Marcus Uhthoff	Steven Tsai
Introduction and Background	Owned Section	Contributed
Research of Prior Work	Owned Section	Contributed
Approach	Shared Responsibility	Shared Responsibility
Raw Data Collection	Contributed	Owned Section
Dataset Processing	Shared Responsibility	Shared Responsibility
Model Design and Evaluation	Owned development of Logistic Regression and Neural Network	Owned development of XGBoost and Support Vector Machine
Conclusions	Shared Responsibility	Shared Responsibility
Report Preparation	Owned Section	Contributed

D Source Code

All of the source code for this project can be found at:
https://github.com/Stevey8/MLB-WL-prediction