

# Using Web Scraping, Sabermetrics and Modern Machine Learning Techniques to Predict MLB Games

Tony Owens

Adviser: Xiaoyan Li

## Abstract

*This paper involves using data mining to acquire relevant statistics and implementing several machine learning classification models to predict the results of Major League Baseball games. I examine the use of FanGraphs for data mining to simplify the extraction code and allow for the use of more advanced statistics. I also examine whether the classification models perform better with regular or “box score” statistics compared to the advanced statistics or “saber-metrics”. The best performing subset of features were the box score statistics, and the best performing models were the Logistic Regression model, Support Vector Machine and the Neural Network. The accuracy for all of the subsets either met or exceeded that of existing related works.*

## 1. Introduction

In this paper, I sought to maximize the prediction accuracy of classification models when predicting the outcomes of Major League Baseball (MLB) games. In the MLB, there are 30 teams who each play 162 games in the regular season totaling 2430 games each year. This paper looks to assess how well a predictive model can perform when trained with the offensive (hitting) and pitching stats for both teams in each given game. To acquire data for the models, I used FanGraphs as opposed to common data sources such as Retrosheet [9] or the Lahman database [5], as I believe they allow for the use of more advanced statistics. As a secondary goal of this paper, I intended to determine whether advanced statistics or “saber-metrics” possess greater predictive power than more traditional or “regular” statistics. Advanced stats in baseball are often more complex than traditional stats, as they often seek to include additional context or grasp more general trends than regular stats. For example, a regular statistic used to measure the ability of a hitter is batting average

(BA) or the amount of hits a batter achieves per at bat. A higher BA is generally considered better, but additional context is needed to properly evaluate a hitter. Simply looking at a player's BA does not provide any context as to their performance within the league as a whole, nor does it account for their team's home stadium, within which a team plays 81 of their games. In baseball the dimensions of stadiums vary from team to team, and the stadium combined with the location (altitude) impacts how far the ball will travel when hit, and what hits are more likely to produce positive outcomes. An advanced stat which takes all of this into consideration is weighted runs created plus (wRC+). The + in wRC+ indicates that the stat is normalized with 100 representing the league average. A player with a wRC+ of 110 is 10% better at generating runs for their team than an average player. These stats can provide benefits when comparing between years since they account for the run environment of the given season. Additionally, the statistic takes into account the "ballpark factor" or how offensive production compares in a given stadium compared to the average offensive production. I hypothesize that the advanced stats will exhibit greater predictive power than the traditional stats, as they are able to extract more information about an individual team's hitting and pitching ability.

## **2. Related Work**

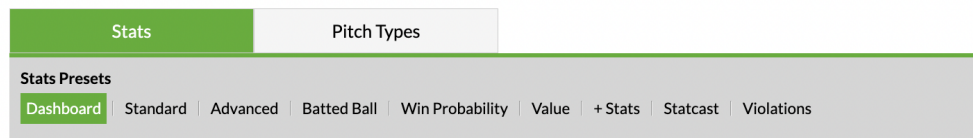
Predicting MLB games with machine learning has been popular since at least the early 2010s. In the past, it was more common to use game logs from websites such as Retrosheet and the Lahman database as shown by [4], [2] and [12]. These websites are popular as they are publicly available and offer a straightforward interface to download the data in the form of text files. From there it is common to write scripts to parse the information in the game logs in order to acquire the offensive and defensive stats for each team. In feature selection [4] manually selected the features and finished with 7, while [12] used several feature scoring metrics to determine the top 60 features for the models, and [2] focused on feature importance for selection.

In [4] the models employed were Logistic Regression (Log Reg), a Support Vector Machine (SVM), an AdaBoost classifier and a LogitBoost classifier. [12] used a K-Nearest Neighbors model

(KNN), Artificial Neural Networks, Decision Trees and SVMs in their prediction. [2] utilized a Logistic Regression elastic net model in all predictions. The best performing model for both [4] and [12] was the SVM. Both [4] and [12] used regression and classification techniques for predicting the results. The regression technique predicted each team's run total with the team with the higher of the scores classified as the winner. In each of the papers, the regression technique performed worse than the classification technique. Both [4] and [4] papers finished with an accuracy of just under 60% for the best performing models, while [2] finished with a top end accuracy of 61.8%

### 3. Approach

In this paper I use the FanGraphs [3] and CBS sports [1] websites to acquire the data for the predictive models. On the FanGraphs website, the user can access hitting and pitching stats for individual players or teams.



**Figure 1: FanGraphs dashboard where the user can specify the type of stats to display.**

As shown in Figure 1, these stats are broken into several categories, including “Standard” and “Advanced”. This paper will only include the “Standard” and “Advanced” stats for both pitchers and hitters in the data set used to train the classification models.

```
'''https://www.fangraphs.com/leaders.aspx?pos=all&stats=bat&lg=all&qual=0
&type=0&season=2023&month=0&season1=2023&ind=0&team=0&ts&rost=0&age=0
&filter=&players=0&startdate=2023-01-01&enddate=2023-12-31'''
```

**Figure 2: FanGraphs URL with the startdate and enddate parameters circled**


On the FanGraphs website, the user is able to specify parameters within the url as shown in Figure 2. The most useful of these parameters for this paper are the “start date” and “end date” parameters. These fields set the time frame for which the data is recorded.

Making use of these parameters, as well as the parameters for the year and type of season (spring training, regular season, or playoffs), I wrote a python script to iterate from the start date of a season

The image shows the FanGraphs dashboard interface. At the top, there are two tabs: 'Leaderboards' (selected) and 'Splits Leaderboards'. Below these are three main categories: 'Player Stats', 'Team Stats' (selected), and 'League Stats'. Under 'Team Stats', there are three sub-tabs: 'Batting' (selected), 'Pitching', and 'Fielding'. Below the tabs, there are several filter options: 'League: All Leagues', 'Team: All Teams', and checkboxes for 'Split Teams', 'Active Roster', and 'HOF'. A row of position abbreviations is shown: All, P, C, 1B, 2B, SS, 3B, RF, CF, LF, OF, DH, NP. Below this, there are dropdowns for 'Single Season: 2023' and 'Split: Full Season', along with a 'Min PA: 0' dropdown. Further down, there are checkboxes for 'Split Seasons', 'Rookies', and 'Multiple Seasons', with 'Multiple Seasons' set to '2023' to '2023'. A 'Submit' button is next to these. Below that is a 'Custom Date Range' section with 'Enter Date' fields and a 'Submit Custom Date' button. At the bottom, there is an 'Ages' section with '14' to '58' and a 'Filter Age' button. The footer text says 'Currently viewing seasons between 2023 and 2023'.

**Figure 3: The FanGraphs dashboard where the user can specify their desired statistics**

to the final regular season game, and utilize the python requests package to scrape the FanGraphs page and record the standard and advanced hitting and pitching stats for each team on each date in a table. Adding the date as a field in the table left me with a table consisting of every team's relevant stats for each day in the season. Since each season starts and ends on a different date, I used March 30th and September 30th as the bounds for a "season". These stats represented the features in the data set. With the features completed, I needed to include the results of the games. For the results I utilized the CBS sports MLB schedule website. This website displays the score of each completed MLB game on a given date, as shown in Figure 4, and the date is included as a parameter in the url in the format YMD. For example, April 20th 2023 is encoded as 20230420. Using the CBS website, I appended the results of each game, along with the date the game took place in a schedule table consisting of all games throughout the season. From there, I could merge this table with the features table based on the date and the team to create a table with each team's statistics from the start of the season, up to and including the day before the game took place. Since some of the stats overlap for hitters and pitchers, e.g. a team's batting average and the batting average against the team's pitchers are both encoded as AVG, the prefixes "Hit\_" and "Pit\_" were used to denote which side of the ball the features referred to. From there, in the schedule table I included the statistics of the home and away teams for each game, with the prefix "Home\_" or "Away\_". This left me with the final data set, which included each game in a season and the statistics for the home team and away team up to


<https://www.cbssports.com/mlb/schedule/20230420/>


### MLB Schedule

[Schedule](#)
[Buy Tickets](#)















Apr 19

Apr 20

Apr 21



Thursday, April 20, 2023

| AWAY   | HOME  | RESULT               | PITCHER (W)           | PITCHER (L)          | PITCHER (S)      |
|--|---|----------------------|-----------------------|----------------------|------------------|
|  Minnesota    |  Boston        | BOS 11 - MIN 5       | T. Houck<br>(3-0)     | K. Maeda<br>(0-3)    | —                |
|  L.A. Angels  |  N.Y. Yankees  | NY Yankees 9 - LAA 3 | N. Cortes<br>(3-0)    | P. Sandoval<br>(1-1) | —                |
|  Cincinnati   |  Pittsburgh    | PIT 4 - CIN 3        | R. Contreras<br>(2-1) | L. Weaver<br>(0-1)   | D. Bednar<br>(6) |
|  Colorado     |  Philadelphia  | COL 5 - PHI 0        | R. Feltner<br>(1-2)   | M. Strahm<br>(1-2)   | —                |
|  L.A. Dodgers |  Chi. Cubs     | LAD 6 - CHC 2        | C. Ferguson<br>(1-0)  | M. Fulmer<br>(0-2)   | —                |
|  San Diego    |  Arizona       | SD 7 - ARI 5         | B. Honeywell<br>(2-0) | K. Nelson<br>(3-1)   | J. Hader<br>(6)  |
|  N.Y. Mets    |  San Francisco | NYM 9 - SF 4         | K. Senga<br>(3-0)     | S. Manaea<br>(0-1)   | —                |

**Figure 4: The MLB schedule on the CBS website for April 20th, 2023**

and including the day before the game. I repeated this process for each season from 2013 to 2022, with the exception of the year 2020. The decision to exclude the 2020 season was largely based on abnormalities within the year due to the COVID restrictions. The 2020 season lasted only 60 games, as opposed to the usual 162 games. Additionally, players had the ability to opt out of the season, a policy which did not exist in previous years. However, the most important change, and the one that ultimately swayed my decision to omit the year 2020 is the fact that the games were played without fans and some teams were required to play their home games at a neutral site. Given the small sample size, and all of the changes in the year, I came to the conclusion that 2020 was not representative of a typical MLB season, and as a result excluded all data from 2020 in the final data set.

## Benefits

The use of FanGraphs and parameterized web scraping serves two major purposes. What is most common when performing such research, is that the data is retrieved from a database such as

Retrosheet in the form of game logs (sheets that include each at bat in the game). As in [12], [2] and [4] the researcher is able to retrieve information for a given game and create summary statistics for that game. From there these stats are combined to represent the accumulative stats for a team on a given date. If a game is the tenth game of the season for both teams, the stats are calculated using each team's previous nine games. In contrast, the FanGraphs approach simply involves setting the start date as the date prior to the first game of the season, and setting the end date to the day prior to the date of the game in question. After the home and away team are acquired from the CBS website, the information is merged together and the user has all of the stats for the game in question. Since the researcher is not responsible for calculating the statistics, it lowers the risk of errors which can impact future calculations.

### **3.1. Data set**

The final data set contained 21020 rows, each representing a unique game, and 158 columns which are a combination of identifiers (home team, away team) features and labels. For each game, there are hitting and pitching features for both teams. While both [4] and [12] included the score of each game in their predictions for their regression approach, the data set in this paper uses a Boolean label (whether or not the home team wins).

### **Models**

The models in this paper were all implemented in Python. I built the Neural Network in PyTorch while all other models were created with sci-kit learn [8] (sklearn). The sklearn models are Random Forest (RF) Classifier, Gradient Boosting (GB) Classifier, Support Vector Machine (SVM), K Nearest Neighbors (KNN), and Logistic Regression (LR). For each of the sklearn models, I implemented the model with the default hyper-parameters and tested its accuracy on the testing set. This was considered the "baseline accuracy" for the model. The accuracy after tuning the hyper-parameters was considered the "tuned accuracy".

## 4. Implementation

### 4.1. Baseline

In the data set, the home team wins 53.4% of the games. This means any model which cannot perform at this baseline is not worth using, as one would be better off predicting the home team to win every game.

### 4.2. Split

Since the regression technique performed comparatively worse, this paper only uses classification to predict the game results. I split the data using a 60-20-20 train, validation, testing split.

### 4.3. Feature Selection

The feature selection took place across each subsection (all features, regular features, advanced features) individually. The `f_classif` module in `sklearn` was used for feature selection. For each subsection of features, I calculated the ANOVA-F value (f score) and the p score of all features. From there I limited the features to those which ranked in the top 60% for f score and had a p value less than 0.01.

| Feature       | Definition   |
|---------------|--|
| Away_Pit_ERA- | Earned Run Average Minus, The amount of runs the visiting team gives up each game adjusted based on the stadiums they play in.                             |
| Home_Pit_ERA- | Earned Run Average Minus, The amount of runs the home team gives up each game adjusted based on the stadiums they play in.                                 |
| Away_Pit_ERA  | Earned Run Average, The amount of runs the visiting team gives up each game.   |
| Away_Pit_WHIP | The amount of walks and hits the visiting team gives up per inning.  |
| Home_Pit_ERA  | The amount of runs the home team gives up each game.   |
| Home_Pit_WHIP | Weighted Runs Created Plus, the amount of runs the home team creates, weighted for the league average and the stadium they play in.                        |
| Home_Hit_wRC+ | Batting Average, The amount of hits the visiting team's pitchers give up per at bat.   |
| Away_Pit_AVG  | Fielding Independent Pitching Minus, A the visiting team's pitchers' performance when only focusing on home runs, hit by pitches, walks and strikeouts.    |
| Away_Pit_FIP- | Weighted Runs Created Plus, the amount of runs the visiting team creates, weighted for the league average and the stadium they play in.                    |
| Away_Hit_wRC+ | Weighted On-base Average, on base percentage weighted with a factor ewline assigned to each possible method for getting on base for the home team.         |
| Home_Hit_wOBA | On Base Percentage, the amount of times the home team reaches base per plate appearance.   |
| Home_Hit_OBP  | Left on Base Percentage, the percentage of baserunners the visiting team allows on base that do not score.   |
| Away_Pit_LOB% | Left on Base Percentage, the percentage of baserunners the home team allows on base that do not score.   |
| Home_Pit_LOB% | On Base Percentage Plus Sluggin Percentage, the sum of the amount of runners that reach base per at bat and the amount of bases per hit for the home team. |

**Table 1: The 15 best features from the feature selection of all possible features**

As shown above in Table 1, the best features are relatively split among the home and away team, and a majority of the features focus on pitching rather than hitting. It also appears as though a majority of the features (ERA-, WHIP, FIP, wRC+, wOBA, LOB%, and OPS belong to the advanced feature subset).

## 4.4. Models

**4.4.1. K Nearest Neighbors** The KNN model is a lazy learner, which means it does not learn any type of specific function based on the distribution of the training data. The main drawback of this model is that it can be overly sensitive to the training data. To make predictions, the KNN stores the training data, and for each point in the testing data it finds the  $k$  nearest points in the training data. From there, it classifies the point based on the labels of the neighboring points in the training data. In related works, it is often the worst performing model, as it often fails to truly grasp the complexity of the data set. I included this model mainly to determine if the performance would match that of what I observed in the related works. When tuning the hyper parameters, I altered the number of neighbors, and whether the neighboring points are weighted equally, or assigned weights based on their distance to the point in question. The KNN performed best across all subsets with a  $k$  value of 50. The choices between weighing the distances appeared less significant.

**4.4.2. Logistic Regression** Logistic Regression models the probability of an event using the log-odds function. In the case of binary classification, a combination of the features is put through the log-odds function and this returns the probability, which is converted to a label of either 0 or 1. Similar to the KNN, Logistic Regression performed poorly in [12] across both the classification and regression methods. However, in [2] the Logistic Regression reached an accuracy of over 61% indicating there was clearly potential for great performance. I included this model as it was utilized within the related works, and typically requires much less time for training and tuning than models such as the SVM, Random Forest or Gradient Boosting. Logistic Regression is also commonly featured in binary classification tasks. When tuning, I altered the regularization parameter, whether or not an intercept was added to the decision function, the method for calculating the penalty (11



distance, l2 distance or elastic net which combines and weighs the two distances), whether the classes are assigned weights, and the solver used. When comparing across subsets of features, the only parameters which arrived at different values after tuning were the regularization parameter, and the class weight. The subset of both regular and advanced statistics (full subset) and the advanced subset ended optimization with the lowest regularization parameter at .010, while the regular subset finished at 0.505. In terms of the class weights, the full subset performed best with balanced class weights, while the regular and advanced subsets performed best with no class weights.

**4.4.3. Random Forest** Random forest models are ensemble models, which work by combining other models to hopefully improve overall performance. A random forest model consists of a group of decision trees, each of which are trained on features from the data set sampled randomly with replacement, meaning some features will appear multiple times in a tree while others may not appear at all. The overall prediction is the result of a majority vote between each of the decision trees. This model was among the best performers in [4], and is utilized frequently in classification and regression problems. For tuning the model, I altered the total number of decision trees within the "forest", the max depth of a decision tree, the minimum number of samples on each side to make a split on the decision tree, the minimum number of samples required in each end or "leaf" node and the amount of features observed when the tree looks for the best possible split. During tuning, in each subset the random forest model performed best with trees of depth 10. The minimum number samples required to split an internal node appeared less important, but the best values for the full, regular, and advanced subsets are 2, 5, and 2 respectively. The regular and advanced subsets performed best when considering the best  $\sqrt{n}$  features while the full subset performed best with the best  $\log_2 n$  features where n is the total number of features in the subset. Each subset performed best with a different number of trees, 50, 150, and 100 for the full, regular and advanced subsets respectively.

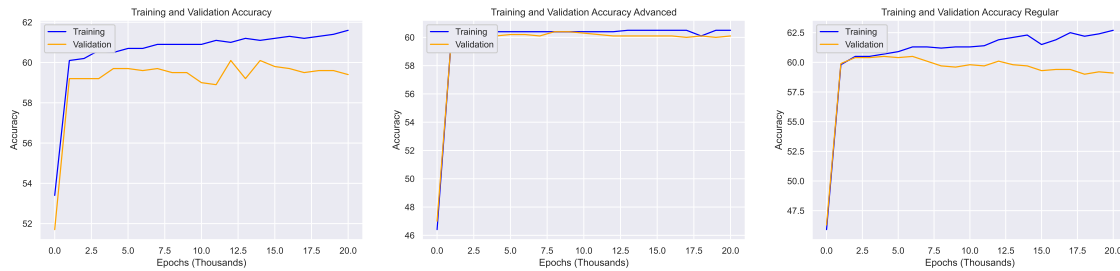
**4.4.4. Gradient Boosting Machine** Gradient Boosting Machines are another example of an ensemble model. A Gradient Boosting Machine (GBM) starts with a decision tree, and adds additional decision trees to the model, which are trained on the error of the previous iteration.

GBMs are capable of achieving exceptional performance, but are also prone to overfitting to the training data. To avoid this, it is common to "prune" the tree, or reduce the size of the tree to limit the impacts of overfitting. This model was not present in the related works, but I have experienced success using the model in the past and believed there was potential for solid performance. When tuning the GBM, I adjusted the number of steps where each step represents adding a tree (number of estimators), how much each individual tree contributes to the overall estimate (learning rate), and the maximum depth of each tree. Each subset performed best with a learning rate of 0.01. For the full subset, the best accuracy was at a learning rate of 0.01, with a depth of two and 701 estimators. For the regular and advanced subsets those numbers were 0.1, 1, 400 and 0.05, 2, 100 respectively.

**4.4.5. Support Vector Machine** The Support Vector Machine appeared often in the related works. This model is extremely popular for classification tasks, since it is often one of the best performing models. SVM works well for classification as it maps each label into a hyperplane in high-dimensional space. From there it can predict the testing data based on where it stands in relation to the labels in this high dimensional space. I chose this model since it performed well in the related works, and exhibits strong performance in general with non-linear data. For tuning, I adjusted the  $\lambda$  regularization parameter, the kernel (linear, polynomial, sigmoid, or radial basis), the degree of the polynomial for polynomial kernels, the zero degree coefficient for the sigmoid and polynomial kernels, and the gamma value, which decides how much the model will adapt to the data when making the hyperplane. A higher gamma value indicates increased curvature, and that the model is more sensitive to each individual data point. The regular subset performed best with a linear kernel whereas the full and advanced subsets performed best with the sigmoid kernel. For the full subset, the best coefficient was -1 and the best gamma value was 0.1.

**4.4.6. Neural Network** The neural network is a staple of Machine Learning for its ability to make connections within data that other models are simply unable to. In related works, the neural network performed generally well in terms of accuracy. I implemented the neural network in PyTorch [7]. For each of the feature subsets (advanced, regular, both) I used the same Neural Network implementation. The Neural Network uses four hidden layers, the ReLU activation function, the

sigmoid function to map the outputs between 0 and 1, the binary cross entropy loss function, and the average stochastic gradient descent optimizer. The only difference between the models is the size of the input, since the subset of both regular and advanced statistics is of length 43 while the other two subsets are of length 15. For the neural networks, each model runs through 100,000 epochs where it makes both training and validation predictions and undergoes a process called back-propagation where the errors are fed back through the model to update its state and improve its performance. To select the best state to use for the testing predictions, I retain the testing loss if it represents an improvement on the previous best, and after all the iterations, I use the recorded state for the testing predictions. The state of the model represents the weights between each neuron at a given stage during training.



**Figure 5: The Neural Network Training and Validation accuracy for each epoch for all the subsets of the data.**

As shown in Figure 5 above, the validation accuracy will stop improving as the neural network over-fits to the training set, but afterwards I use the saved state that minimizes the validation loss.

## 5. Evaluation

For each of the models, I used accuracy as an evaluation metric.

| <b>Model</b>              | <b>Baseline Accuracy</b> | <b>Tuned Accuracy</b> |
|---------------------------|--------------------------|-----------------------|
| Logistic Regression       | 60.2%                    | 58.9%                 |
| K Nearest Neighbors       | 55.6%                    | 56.9%                 |
| Gradient Boosting Machine | 59.5%                    | 59.3%                 |
| Random Forest             | 56.1%                    | 57.7%                 |
| Support Vector Machine    | 59.5%                    | 57.6%                 |

**Table 2: The accuracy for all models using a combination of advanced and regular statistics**

| <b>Model</b>              | <b>Baseline Accuracy</b> | <b>Testing Accuracy</b> |
|---------------------------|--------------------------|-------------------------|
| Logistic Regression       | 60.5%                    | 60.6%                   |
| K Nearest Neighbors       | 55.2%                    | 57.3%                   |
| Gradient Boosting Machine | 60.5%                    | 59.1%                   |
| Random Forest             | 56.9%                    | 58.6%                   |
| Support Vector Machine    | 61.0%                    | 60.7%                   |

**Table 3: The accuracy for all models using the top 15 regular statistics**

| <b>Model</b>              | <b>Baseline Accuracy</b> | <b>Testing Accuracy</b> |
|---------------------------|--------------------------|-------------------------|
| Logistic Regression       | 59.9%                    | 58.8%                   |
| K Nearest Neighbors       | 55.6%                    | 57.9%                   |
| Gradient Boosting Machine | 59.7%                    | 59.5%                   |
| Random Forest             | 57.3%                    | 58.2%                   |
| Support Vector Machine    | 59.2%                    | 58.3%                   |

**Table 4: The accuracy for all models using the top 15 advanced statistics**

| Stats       | Testing Accuracy |
|-------------|------------------|
| Full Subset | 57.4%            |
| Regular     | 60.5%            |
| Advanced    | 59.9%            |

**Table 5: The accuracy for each neural network with each subset of features**

| Subset             | Best Model                           | Best Accuracy |
|--------------------|--------------------------------------|---------------|
| Regular & Advanced | Logistic Regression                  | 60.6%         |
| Regular            | Support Vector Machine               | 61.0%         |
| Advanced           | Logistic Regression & Neural Network | 59.9%         |

**Table 6: The best model and its accuracy for each of the three subsets**

## 5.1. Results

Out of all the models, the Log Reg model deviated the most from my expectations. While the model ranked as one of the least accurate models in [4], within this paper it was consistently among the best performers. This is especially impressive when we consider that the Log Reg model takes the least time to train, which makes it a more attractive option for anyone looking to scale this work to include more data. As shown in Table 5, the Neural Network performs comparably well with the regular and advanced features but when they are combined the performance drops off considerably.

## 6. Conclusions and Future Work

### 6.1. Conclusions

In my view, this work accomplished its goals of using ML models to classify major league baseball games, and to measure the differences in accuracy between saber-metrics and traditional statistics. The accuracy of my models meets or exceeds that of the existing literature, and I believe the FanGraphs approach was useful in practice.

## 6.2. Surprising Performance

Out of all the models, the Log Reg model deviated the most from my expectations. The model performed extremely well for [2], but ranked as one of the least accurate models in [4]. Within this paper it was consistently among the best performers, as shown by 6. This is especially impressive when we consider that the Log Reg model takes the least time to train, which makes it a more attractive option for anyone looking to scale this work to include more data.

**6.2.1. Subset comparisons** The performance of models matched or exceeded that of the current literature. While each feature subset is close in accuracy, Tables 2, 3, and 4 clearly show that the worst performing subset is the advanced features when used by themselves. The regular subset and the full subset were close in performance but the regular subset finished as the better of the two. This demonstrates that the advanced statistics are valuable for making predictions, but struggle in the absence of the regular statistics. While the regular subset of features was the best performing of the bunch, its advantage was by no means definitive. It would require future research to truly determine which features are better and by what margin, however this work provides an interesting finding which I plan to explore in greater detail.

## 6.3. Future Work and Limitations

**6.3.1. Pitcher Stats** There are many ways in which one can improve upon this work. Regrettably, this work does not include the statistics for the starting pitcher of each game. The overall pitching stats represent an average of all of the pitchers, but this fails to capture each individual pitcher's performance. This presents an issue when there is great variance between the quality of a team's starting pitchers. I believe that even the inclusion of something as simple as the ERA or WHIP of each team's starting pitcher would reduce the error of the model by a decent margin. This data is available via FanGraphs in the "probable pitcher" section, which displays the statistics for the pitcher who is deemed most likely to start in a given game. There are issues with this data, since the pitchers need to be matched with a specific team and some pitchers show up without an associated team. This phenomenon does not appear often, but it would reduce the amount of data available.

Furthermore, it would require an analysis into the specific pitchers who are omitted, to ensure that unexpected bias is not introduced when games with those pitchers are removed from the data set. Even with these issues, I am confident that the starting pitcher statistics would provide valuable information to the model, and could offer considerable performance gains. I was also unable to include fielding statistics with the current FanGraphs implementation, however I do not believe this was as significant of an omission as the starting pitcher statistics. In the MLB the difference between an elite fielder and a league average or even below average fielder does not come into play in the majority of situations. In a given game there may only be two or three plays where the fielder is required to make a difficult play, but the pitcher on the other hand is involved in every pitch and a bad performance can easily swing a game in favor of the other team.

**6.3.2. Regression Approach** Additionally, I believe the regression approach implemented in some related works could be worth attempting. Even if the regression approach did not match the accuracy of the classification approach, it would be an interesting point of analysis to determine how the models performed in relation to each other. All of this data is available on the CBS website in the same way in which I acquired the winner and loser, but I did not include it in the final data set.

**6.3.3. Betting Odds** Furthermore, in [4] the betting odds were included as a baseline for model performance. I believe that this baseline would provide further context to the results and better gauge the models' performance. The current baseline merely serves to demonstrate that the model is better than predicting the home team in every game. The betting odds on the other hand are a much more accurate predictor and it would be beneficial to know which if any models outperform the betting odds, and if so by what margin. If a model is able to consistently outperform the betting odds, it could offer financial advantages to anyone who uses it to bet on MLB games.

**6.3.4. Time Frame Analysis** Given the opportunity, I would also go back and include an analysis of the data based on the month in which the game took place, as did [4] and [2]. They concluded that the accuracy improves as the season goes on, which makes sense since the statistics involved represent a larger portion of the season. I believe it could have strengthened my analysis, if I would have examined whether this trend persists in my data. Additionally, I could have explored whether

the trend is stronger with certain models, or with certain subsets of features. I hypothesize that the regular statistics would outperform the advanced statistics earlier in the season. In a smaller sample size, the advanced statistics could capture more of the current trends between a team, as there would be more variance in the regular stats earlier in the season. For example, if a team starts off the year well and hits a significant amount of home runs, but they are considerably outperforming their expected amount of home runs, advanced statistics are less likely to overrate the team based on this temporary performance. This would be an excellent hypothesis to test with further research on the subject.

## **7. Acknowledgements**

I would like to thank Dr. Xiaoyan Li for her guidance throughout the semester. During her seminar I learned important best practices and techniques which I used to write this paper. I would also like to thank the students in COS IW03, who offered insightful feedback throughout the process of writing this paper. Finally, I would like to thank my friends and family, for providing me support which was instrumental in getting through the semester.

## **8. Honor Code**

This paper represents my own work in accordance with University regulations. - Tony Owens

## **9. Appendix**

### **9.1. GitHub**

The code used for the models in this paper as well as the data set itself is available in the public GitHub repository linked below:

[https://github.com/tonyowensjr/COS\\_IW03\\_Independent\\_Work](https://github.com/tonyowensjr/COS_IW03_Independent_Work)



## 9.2. FanGraphs Stat Glossary

The full glossary with all of the stats explained can be found at:

<https://library.FanGraphs.com/FanGraphs-library-glossary/>

## 9.3. MLB Stat Glossary

The MLB glossary provides a brief explanation of common baseball terms and I believe it is better for readers who are not familiar with baseball. It can be found at:

<https://www.mlb.com/glossary>

## 9.4. Extraction Code

```
import sys
# Iterate through all dates from March 30th to September 30th inclusive
for date1 in np.arange(datetime(year,start_mon,start_day), datetime(year,end_mon,end_day), timedelta(days=
    tt += 1
    time.sleep(.2)
    date = date1.strftime("%Y-%m-%d")
    try:
        # acquire the schedule for the given day from cbs
        cbs_url = f"https://www.cbssports.com/mlb/schedule/{date.replace('-', '')}"
        matches = pd.read_html(requests.get(cbs_url).content)[0]
        matches['Away'] = matches['Away'].map(acronyms)
        matches['Home'] = matches['Home'].map(acronyms)
        matches['Winner'] = np.empty(len(matches.index))
        matches['Loser'] = np.empty(len(matches.index))
        matches.insert(1, 'Date', [date] * len(matches.index))
        for j in matches.index:
            matches.loc[j, 'Winner'] = str(matches[matches.index == j]['Result']).split(' ')[1].split('
            matches.loc[j, 'Loser'] = str(matches[matches.index == j]['Result']).split(' ')[1].split('
        all_matches.append(matches)
        time.sleep(.2)
        # get the regular hitting stats from fangraphs
        fg_url = f"https://www.fangraphs.com/leaders.aspx?pos=all&stats=bat&lg=all&qual=0&type=0&season={y
        c = pd.read_html(requests.get(fg_url).content)[-2].iloc[:30,:1]
        c.columns = c.columns.droplevel(1)
        c.insert(1, 'Date', [date] * 30)
        bat_reg.append(c)
        time.sleep(.2)
        # get the advanced hitting stats from fangraphs
        fg_url = f"https://www.fangraphs.com/leaders.aspx?pos=all&stats=bat&lg=all&qual=0&type=1&season={y
        c = pd.read_html(requests.get(fg_url).content)[-2].iloc[:30,:1]
        c.columns = c.columns.droplevel(1)
        c.insert(1, 'Date', [date] * 30)
        bat_adv.append(c)
        time.sleep(.2)
        # get the regular pitching stats from fangraphs
        fg_url = f"https://www.fangraphs.com/leaders.aspx?pos=all&stats=pit&lg=all&qual=0&type=0&season={y
        c = pd.read_html(requests.get(fg_url).content)[-2].iloc[:30,:1]
        c.columns = c.columns.droplevel(1)
        c.insert(1, 'Date', [date] * 30)
        pit_reg.append(c)
        time.sleep(.2)
        # get the advanced pitching stats from fangraphs
        fg_url = f"https://www.fangraphs.com/leaders.aspx?pos=all&stats=pit&lg=all&qual=0&type=1&season={y
        c = pd.read_html(requests.get(fg_url).content)[-2].iloc[:30,:1]
        c.columns = c.columns.droplevel(1)
        c.insert(1, 'Date', [date] * 30)
        pit_adv.append(c)
        if not tt % 10:
            print(date)
    except ValueError or KeyError:
        continue
```

Figure 6: The simplified extraction code as a result of the FanGraphs implementation

## 9.5. Review

The information about each model is based on a combination of material from the COS IW seminar and the following sources [10], [11], and [6].

## 9.6. Inspiration

The format of the appendix was partially inspired by [2].

## References

- [1] CBS Sports, “Mlb schedule,” <https://www.cbssports.com/mlb/schedule/>, Accessed 2023-04-20.
- [2] A. Cui, “Forecasting outcomes of major league baseball games using machine learning,” 2020. Available: [https://fisher.wharton.upenn.edu/wp-content/uploads/2020/09/Thesis\\_Andrew-Cui.pdf](https://fisher.wharton.upenn.edu/wp-content/uploads/2020/09/Thesis_Andrew-Cui.pdf)
- [3] FanGraphs, “Fangraphs,” <https://www.fangraphs.com/>, Accessed 2023-04-20.
- [4] R. Jia, C. Wong, and D. Zeng, “Predicting the major league baseball season,” *CS229 Machine Learning Final Project*, pp. 1–5, 2013.
- [5] S. Lahman, “Lahman Baseball Database,” <http://www.seanlahman.com/baseball-archive/statistics/>, 2016, version 2016.
- [6] MathWorks, “Machine learning models,” <https://www.mathworks.com/discovery/machine-learning-models.html>, Accessed April 25thm 2023.
- [7] A. Paszke *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” *Advances in Neural Information Processing Systems*, vol. 32, pp. 8024–8035, 2019.
- [8] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [9] RetroSheet, “RetroSheet game logs,” <https://www.retrosheet.org/gamelogs/>, Accessed 2023-03-20.
- [10] T. Shin, “All machine learning models explained in 6 minutes,” <https://towardsdatascience.com/all-machine-learning-models-explained-in-6-minutes-9fe30ff6776a>, 2020, accessed on April 15th, 2023.
- [11] H. Singh, “Understanding gradient boosting machines,” <https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab>, 2018, accessed on April 19th, 2023.
- [12] C. S. Valero, “Predicting win-loss outcomes in mlb regular season games—a comparative study using data mining methods,” *International Journal of Computer Science in Sport*, vol. 15, no. 2, pp. 91–112, 2016.