Stat 451 All-Star Project

Eddie Morelli
ebmorelli@wisc.edu

Andrew Olson
alolson7@wisc.edu

Abstract

The intent of this project is to create a machine learning model that accurately predicts whether or not a player in the MLB will be an "all-star" player. There are 68 players in the MLB each year that are selected to be an all-star player. For this project, we are only evaluating hitters as the statistics for other players does not correspond well to hitters. To accomplish this we collected data for each players baseball statistics (hits, runs, strikeouts, etc.) for each year since 2014. This data was divided into a 75% training set and a 25% test set. Machine learning models such as KNN, Logistic Regression, Decision Trees, Random Forests, and ensemble methods were used to create an accurate model that predicted whether or not a player will be an all-star based on this players statistics. Additionally, we will determine what factors are most influential in determining who will be an all-star player according to our machine learning models.*

**see section 4.*

1. Introduction

1.1. All-Star Game and Player Selection

Each year, players from Major League Baseball (MLB) teams are chosen to participate in the All-Star game, a showcase of the players who have had the best season. Players earn All-Star status through either fan votes or MLB team managers' votes, depending on their position. Starting fielders, specifically, are voted solely by fans. Thus, it is not guaranteed that the "best" fielders make the all-star game; voting may cause inaccuracy due to some teams having bigger fan bases, or fans voting based on players' reputations. Baseball statistics, such as hits and home runs, may provide an objective measure of the players' performances during the regular season. Thus, the purpose of this project is to create a classifier, using machine learning methods, to predict MLB All-Star fielders based on player statistics.

1.2. History of Statistics in Baseball

Pre-21st century Statistics in baseball are as old as the game itself. Statistics like base hits date back to the 1860's [5]. Throughout the late-19th and early-20th centuries, more stats like batting average and runs batted in were recorded by team statisticians. Thus, we have access to the fundamental baseball statistics of early greats, like Ty Cobb, Honus Wagner, and Babe Ruth.

Sabermetrics In 1971, the Society for American Baseball Research (SABR) was created, giving birth to the term "SABRmetrics", more commonly stylized as "sabermetrics". Bill James, an influential baseball statistician, defined sabermetrics as "the search for objective knowledge about baseball" [3]. Bill James would go on to write tens of books about baseball, including yearly handbooks of baseball statistics.

Including the introduction of sabermetrics, statistical analysis in baseball during the 20th century only increased. Computers were used to analyze data. Statistics were often made available to the public, through the newspaper or other avenues.

21st century Statistical analytics in baseball was at an all-time high. A pivotal time in baseball was when the 2002 Oakland Athletics won the American League West division, led by a statistical approach to adding players to the team. This season was highlighted in the 2011 film *Moneyball*.

In the early 2000's, many MLB stadiums implemented technology to measure things like pitch speed. But, in 2015, this was taken to another level with the implementation of Statcast. As described on the MLB website: "Statcast is a state-of-the-art tracking technology that allows for the collection and analysis of a massive amount of baseball data, in ways that were never possible in the past. Statcast can be considered the next step in the evolution of how we consume and think about the sport of baseball" [1]. High-tech cameras installed in every MLB stadium allowed new, groundbreaking statistics to be recorded, such as velocity of balls off the bats of hitters, or RPM's of balls thrown by pitchers.

By 2021, MLB's Standard Stats glossary held 72 stats, including offense, defense, pitching, and team statistics [8]. On top of that, the MLB recognizes 32 Statcast metrics across offense, defense, and pitching.

2. Related Work

2.1. Predicting Hall of Famers

One similar study that was done by graduate students at Georgia Tech used machine learning techniques to predict what players would be inducted into the baseball Hall of Fame[7]. In this study the students used clustering algorithms such as KMeans with Principal Component analysis (PCA) to create an unsupervised model that used both recent data, and all time data to make predictions. Additionally, the students created a supervised machine learning model with a neural network that performed significantly better than the unsupervised model. They found that their supervised machine learning algorithms were, for the most part, very accurate in predicting Hall of Famers, reaching an accuracy as high as 94% using a neural network.

2.2. Predicting MLB injuries

A different study done by Karnuta et al in the Orthopaedic Journal of Sports Medicine used machine learning algorithms to predict next season injury risk for MLB players. In this study the team used a collection of four online datasets collecting many features such as age, performance metrics, and injury history to use with machine learning models[8]. They tested fourteen different machine learning models and created a ensemble of the three best performing models. In addition, thirteen of the fourteen models chosen performed better than logistic regression.

These machine learning models were built using the scikit-learn Python library (Version 0.20.3). In addition, they split their large dataset into a 90% training set and a 10% test set. To measure how influential each feature was in relation to injury risk was calculated using XGBoost model with the "Gini" performance metric. The top 3 ensemble created was the best performing model achieving an accuracy of 70%± 2.0% for predicting future injury.

In their analysis some of the most important variables were found to be amount of injuries, amount of pitches hit, and weighted cutter runs per 100 pitches. Interestingly, this model could also predict the anatomic region of the injury for all players besides pitchers. A top 3 ensemble model was also created for determining the anatomic region of the injury. This model performed slightly worse with an accuracy of 63.0% ± 3.6%, however, this is not too surpising as predicting the location of the injury is more difficult to accomplish with the given data as opposed to determining if a player will suffer an injury in future years. Although this study was focused on a different subject within the MLB, its methods and data collection are similar to our processes.

3. Proposed Method

3.1. Logistic Regression

Logistic regression is a classifier that predicts the conditional probability of a label being 0 or 1. For this model and each model going forward the classifier we are predicting is whether or not a player is an all star. y = 1 corresponds to the player being an all star. Logistic Regression calculates this probability of being an all star as follows:

$$P(y=0) = \frac{1}{1+\exp(\beta^\top x)},$$

where x is the feature vector of a player containing relevant statistics, and the vector β is estimated by minimizing the negative log likelihood of the training set. The negatigve log likelihood is calculated as follows:

$$\begin{aligned} &-\log L(\beta) \\ &= -\log(\prod_{i=1}^{N} P_i^{1-Y_i}(1-P_i)^{Y_i}) \\ &= -\sum_{i=1}^{N}(Y_i\beta^\top x_i - \log(1+\exp(\beta^\top x_i))), \end{aligned}$$

N corresponds to the training sample size, x_i denotes the feature vector of the i^{th} player in the training set, Y_i is its label, and P_i, the probability above.

In our case, we will also standardize all the features using a StandardScaler for use in our Logistic Regression Model. This is because some stats such as batting average (BA), are probabilities that must be less than 1 and greater than 0. In comparison to a feature such as number of games, the BA and other small features may not be as weighted as heavily as large features. Standardization solves this problem and gives each feature the same range so no one feature is weighted more based on the values of the feature. This will give us a better idea into what performance metrics are most important in determining if a player will be an all-star player or not.

Additionally, we will evaluate the logistic regression coefficients of each feature to see which features have a positive correlation to being an all star, and which features have a negative relationship to see which features have a negative effect to becoming an all-star player.

3.2. k Nearest Neighbors

The k Nearest Neighbors (KNN) classification algorithm whose training step involves remembering the data exactly. In the fitting step, each new sample with features x^*, the algorithm calculates:

$$d_i = ||x^* - x_{\text{train},i}|| \text{ for } i = 1, 2, ..., N,$$

With $x_{\text{train},i}$ denoting the features of the i^{th} training sample, and N is the size of the training sample, and d_i is the distance function. Then we pick the k indices from 1 to N that are associated with the k smallest distances that are arrived from the equation above.

3.3. Decision Tree

Decision Trees are a supervised machine learning method which will be used exclusively for classification for our purposes, however, Decision Trees supports regression as well. Decision Trees predicts the label of a new player (all-star player or not), by searching for the leaf node the player belongs to. The parameters for Decision Tree such as the max depth of the tree of the criterion for splitting a node are important to how the model performs. To determine the best parameters for our data we will use a grid search that manually tests a range of hyper parameters and selects the parameters that performs best.

3.4. Random Forest

Random Forest is also a supervised machine learning method which is similar to decision trees with the major difference being in Random Forest there are multiple decision trees created. Random Forests fits multiple decision tree classifiers on multiple sub-samples of the training data. Then, Random Forest accumulates this multiple decision trees and uses averaging that can improve the predictive accuracy compared to Decision Trees, as well as control over fitting. Many of the parameters used in Decision Tree, such as max depth and criterion, are also carried over to Random Forest and we will perform the same parameter tuning using Grid Search.

3.5. Voting Ensemble Classifier

An ensemble classifier is one "meta-classifier" that is made up of multiple individual classifier models. Thus, Random Forests can be considered an ensemble classifier, for example, because it is made up of numerous Decision Trees. The Voting Ensemble meta-classifier, on the other hand, surveys the prediction of the lower-level classifiers, and the class that receives the most predictions, or "votes", is the class that is ultimately predicted by the meta-classifier. This implementation used the hard voting method, without weighing the classifiers, which means all lower-level classifiers held equal say in the end result. The Voting Ensemble that will be used in this experiment will include classifiers described above, e.g. Logistic Regression classifiers, k-Nearest Neighbors classifiers, Decision Trees, and Random Forests.

3.6. Stacking Ensemble Classifier

Stacking ensembles are another form of ensemble classifier. Instead of a plurality vote scheme, the meta-classifier learns from the lower-level classifiers and creates a model based on their predictions. Any kind of classifier model can be used as the meta-classifier, so this experiment will attempt to create a Stacking Classifier using meta-classifiers and lower-level classifiers including Logistic Regression, k-Nearest Neighbors, Decision Trees, and Random Forests.

Name	Age	Tm	Lg	G	PA	AB
Fernando Abad	28	OAK	AL	7	0	0
Bobby Abreu	40	NYM	NL	78	155	133
Jose Abreu	27	CHW	AL	145	622	556
Tony Abreu	29	SFG	NL	3	4	4

Table 1. Standard Statistics

Name	Age	Tm	PA	rOBA	Rbat+
Bobby Abreu	40	NYM	155	0.318	101
Jose Abreu	27	CHW	622	0.425	182
Tony Abreu	29	SFG	4	0	-137
Dustin Ackley	26	SEA	542	0.306	94

Table 2. Advanced Statistics

4. Experiments

4.1. Dataset

The data used is a subset of MLB statistics, standard and advanced, from the 2014-2019 and 2021 regular seasons. The 2020 MLB season was shortened due to the COVID-19 pandemic, and there was no all-star game or player selections. Therefore, it was excluded from the analysis.

Retrieval The majority of the data was pulled from baseball-reference.com. The website describes itself as the "complete source for current and historical baseball players, teams, scores and leaders."

Table 4.1 shows an example of the standard statistics pulled from the website.

Table 4.1 shows an example of the advanced statistics pulled from the website.

Merges and Field Engineering The data from these two tables were combined, mapped on the player name and year. Some statistics are a calculation based on other, fundamental statistics. Thus, some features were removed, specifically those that had common fundamental stats. Then, two more fields were added. The first was the target field: whether or not the player was an all-star selection that year, a binary variable (values of 0 for non-all-stars, or 1 for all-stars).

The second field engineered was a "Value Ranking" of the player's team's value. Because the selections are made based on a fan vote, it would make sense that a player from

Name	allstars
Age	Runs (R)
Year	Hits (H)
ID	Strikeouts (SO)
Team (Tm)	Value Ranking
Stolen Bases (SB)	Isolated Power (ISO)
Games Played (G)	Runs Batted In (RBI)
Home Runs (HR)	Batting Average (BA)
Slugging Percentage (SLG)	
Win Probability Added (WPA)	
Run Scoring Percentage (RS%)	
Stolen Base Percentage (SB%)	
On-base plus Slugging (OPS)	

Table 3. Fields in processed dataset

a more popular team would receive more votes. Team value was used to measure team popularity. The team value rankings were retrieved from statista.com [4]. These were converted into a "power ranking" system. For example, a player for the New York Yankees would have a Value Ranking of 1, because the Yankees are the most valuable team. A player from the Chicago Cubs, would have a Value Ranking of 4, because the Cubs are the fourth-most valuable team. Players who played on multiple teams (due to trades, waivers, etc.) had a Value Ranking that averaged each of the teams it played on. For example, a player who was traded from the Yankees to the Cubs, and only played for those 2 teams during the season, would have a Value Ranking of 2.5.

Finally, pitchers and players with less than 50 plate appearances were excluded from the final dataset. Heading into our implementation of the models, our data included 21 columns, listed in table 4.1.

4.2. Software

All of the model-building and analysis was done in the programming language Python. Additional Python libraries were critical to the project, including pandas, sklearn, mlxtend, and matplotlib. Some data manipulation was performed in Microsoft Excel. The Jupyter Notebook environment was used as an interface for Python programming in the form of IPython Notebook (.ipynb) files.

4.3. Hardware

The code was run on the personal computers of group members without the need for external computing power.

4.4. Miscellaneous Functions

The majority of data pre-processing used the numpy and pandas libraries. The data was split into training and test sets using scikit-learn's `train_test_split()` function.

To implement Grid Search for tuning hyperparameters, we used scikit-learn's `GridSearchCV`.

4.5. Logistic Regression

We used the LogisticRegression model from the scikit-learn library. Additionally we used the Pipeline feature from the scikit-learn library to implement standardization to the feature vector before it is applied to logistic regression. Thus, we used the StandardScaler feature from scikit-learn which standardizes the features by removing the mean and scaling to unit variance. The standard score of a sample x is calculated by:

$$z = \frac{(x - \mu)}{s},$$

where μ is the mean of the training sample and s is the standard deviation of the training samples.

While performing Logistic Regression as well as all the other machine learning models we will test multiple feature vectors containing different performance metrics to determine how the different feature vectors will affect the accuracy of the model.

4.6. k Nearest Neighbors

For the k Nearest Neighbors (KNN) classification algorithm we used the KNeighborsClassifier from scikit-learn. To find a value of k that leads to a high accuracy we will tune the parameter for k by grid search. The values of k tested was k starting at 5 to 100 with a step size of 5. The best value of k from these valued tested proved to be $k = 35$. This could be further optimized with smaller increments, however, we did not find a significant increase in accuracy for additional computing required to find an optimal k.

4.7. Decision Tree & Random Forest

We will use the DecisionTreeClassifier and the Random Forest models from scikit-learn. Similar to the other models we will use hyperparameter tuning to find more optimal values for the parameters. For the case of Random Forest the parameters we will be tuning are max_depth and criterion. max_depth is the is the depth of the decision tree and criterion is how each node is split. For our Decision Tree model we will tune more hyperparameters as hyperparameter tuning for Decision Trees is significantly less computationally expensive in comparison to Random Forest. Thus, we included the parameters "min_samples_split" and "max_features" for tuning in Decision Trees. min_samples_split is the minimum number of samples require to split a node, and max_features is the number of features that are considered when looking for the best split.

4.8. Voting Ensemble Classifier

Two voting ensemble classifiers were implemented. The first voting ensemble classifier contained an ensemble of various classifier methods. Grid Search was used to optimize the hyperparameters of many of the lower-level classifiers. The lower-level classifier included one of each of these models:

1. Models with optimized parameters
    (a) Logistic Regression (penalty = None, solver = 'newton-cg')
    (b) k-Nearest Neighbors (n_neighbors = 8)
    (c) Random Forest (criterion = 'entropy', max_depth = 10, max_features = 'auto')
    (d) Decision Tree (criterion = 'gini', max_depth = 2, max_features = None, min_samples_split = 2)
2. Models with default parameters
    (a) Multilayer Perceptron (a type of Neural Network, via scikit-learn's `MLPClassifier` object)
    (b) Adaptive Boosting (a form of Gradient Boosting, via scikit-learn's `AdaBoostClassifier` object)

The second voting ensemble classifier included only the three best-performing lower-level models. This included Logistic Regression, Random Forest, and Adaptive Boosting. More detail about the accuracy of these models is described in section 5.

Each of the lower-level models were implemented using scikit-learn methods, as described in previous sections. The Voting Ensemble Classifier was implemented using mlxtend's `EnsembleVoteClassifier` object.

4.9. Stacking Ensemble Classifier

Twelve total stacking classifiers were implemented. The first six classifiers were implemented using the same lower-level classifiers as the first voting ensemble. Furthermore, 6 different meta-classifiers were tested, each with default parameters:

1. Logistic Regression
2. k-Nearest Neighbors
3. Random Forest
4. Decision Tree
5. Multilayer Perceptron
6. AdaBoost

Each of these models were implemented using scikit-learn methods, as described in previous sections. The Stacking Classifier was implemented using mlxtend's `StackingClassifier` object.

The second set of six classifiers were implemented using the three best-performing lower-level classifiers, similar to the voting ensemble. The same 6 meta-classifiers were used in this set.

5. Results and Discussion

5.1. Logistic Regression

The Logistic Regression model used in conjunction with StandardScaler performed very accurately even before hyper-parameter tuning. The Logistic Regression achieved a test accuracy of 91.7% and a train accuracy of 93.1%. After hyper-parameter tuning the test accuracy remained roughly the same, but had a slightly lower train accuracy of accuracy 92.9%. This is because the default parameters chosen by sci kit learn were revealed to be the best performing parameters after hyper-parameter tuning. The slightly lower train accuracy shows that the hyper parameter tuning may have reduced overfitting, however, this difference is so small that it is essentially neglible. Additionally, figure 1 displays the coefficient weights for each feature.

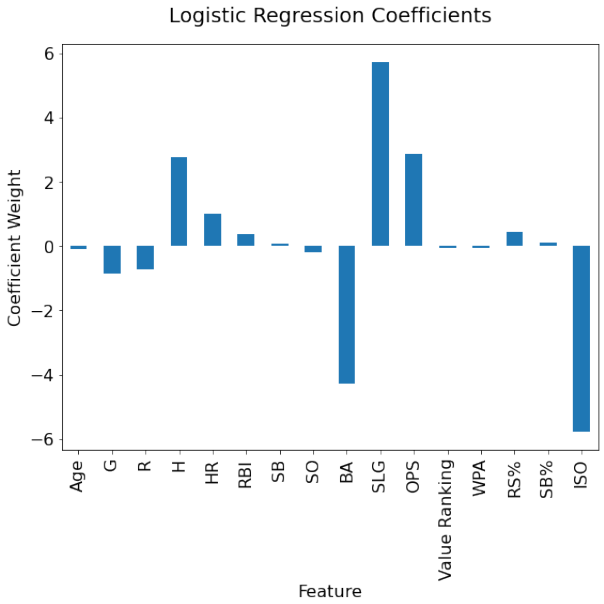

Figure 1. Logistic Regression Coefficients

The figure above shows that the most important feature that has a positive correlation to becoming an all-star player is Slugging Percentage (SLG). The most important feature with a negative correlation to becoming an all-star player is Isolated Power (ISO).

5.2. k Nearest Neighbors

For our k Nearest Neighbors model we achieved a test accuracy of 91.4%. After changing from the default value of k from 5 to 25 which was found to perform better via a grid search the model still performed nearly identically with an accuracy of 91.4%. Thus changing the value of k did not affect the test accuracy much. However, the hyper-parameter tuning may still have reduced the amount of over fitting.

5.3. Decision Trees & Random Forest

The Decision Tree model achieved an accuracy of 91.7% with hyper-parameter tuning to find the optimal parameters. Random Forest performed slightly better with a test accuracy of 92.2%. Since Random Forest is an ensemble method that uses multiple Decision Trees it is logical that it would perform better. Overall these are two of our best performing models, and if we were to choose one we would select Random Forest due to its ensemble nature with selecting multiple Decision Trees for its classification.

5.4. Voting Ensemble

The first Voting Ensemble, consisting of all previous models, had a lower accuracy than expected. The accuracy of this ensemble on the test dataset was 91.5%, and 94.1% on the training dataset. Our Logistic Regression model had an accuracy of about 92.0% on the test set, so it was likely the other classifier models, which had lower accuracy, that pulled down the accuracy of the ensemble. Another possible contribution to the lower ensemble accuracy was the fact that similar features were used for each of the classifiers in the ensemble. If we were to use different features in the lower-level classifiers, we may have been potentially able to raise the test set prediction accuracy.

The second Voting Ensemble, constructed from the three most accurate lower-level classifiers (Logistic Regression, Random Forest, and Adaptive Boosting), achieved a test accuracy of 92.2%, one of the best performances of any classifier. The accuracy on the training set was 94.1%. This confirms that the less-accurate lower-level classifiers were pulling down the accuracy of the first voting ensemble, but still begs the question of using different features. For discussion on misclassifications, this classifier will be used to make the predictions.

5.5. Stacking Ensemble

The first set of stacking classifiers contained 6 models, as described in section 4. Five models had the same training and test dataset accuracy. For the Logistic Regression, Random Forest, Decision Tree, Multilayer Perceptron, and Adaptive Boosting meta-classifiers, the training accuracy was a very high 97.8%, yet the test accuracy was only 92.0%. In contrast, the k-Nearest Neighbors stacking classifier only achieved train and test accuracy of 97.8% and 91.8%, respectively. This may be due to the fact that the hyperparameters for each of the meta-classifiers were not tuned, and the k-Nearest Neighbor algorithm was affected the most by this. Altogether, this first set of stacking classifiers had an excellent accuracy on the training dataset, yet did not perform nearly as well on the test dataset.

The second set of stacking classifiers also contained 6 models, but fewer lower-level classifiers (Logistic Regression, Random Forest, and Adaptive Boosting). Each of these meta-classifiers achieved a familiar training and test dataset prediction accuracy: 97.8% and 92.0%, respectively. It is odd that almost every stacking classifier achieved the same accuracy, but it may tell about the similarity of the predictions among lower-level classifiers.

5.6. Misclassifications

Altogether, while our models predicted all-stars fairly well, there was some misclassification error. Thus, statistics may be a good predictor for all-star selections, but some bias may still remain.

False Positives One goal of this experiment was to see who had not been selected as an all-star, but was predicted by the models to have been an all-star. These players may have deserved an all-star selection based on their performance, but had been "snubbed." Some players were considered a snub in multiple seasons. Those players were Anthony Rizzo, Brian Dozier, Joey Votto, and Nelson Cruz. A few things may be taken away from these snubs:

- Anthony Rizzo and Joey Votto are great players, but they both play first base, a position that is typically very talented, especially in terms of hitting. Thus, the slots for this position were likely filled. Perhaps a better classifier model would be achieved if it was done for each position separately.
- Nelson Cruz is a great hitter, but is not a good fielder. We used primarily hitting stats when building our models. Thus, his lack of fielding skill may have been a factor in his non-selection to the all-star team, which was not able to be reflected in our model's predictions.

In addition, there is one snub that stood out off the bat: Bryce Harper did not receive an all-star selection in 2021. Bryce Harper did not do well in the first half of the 2021 season, yet he had an outstanding second half, leading for him to win the National League Most Valuable Player award. His full-season statistics were great, so it makes sense why our classifier would predict that he was an all-star. But, because the all-star game is considered the halfway point of the season, it is clear why he did not make the all-star

game. Our models predicted all-stars based on the full season of statistics. One thing that would be useful to consider in further machine learning analysis of all-star selections is that it may be better to only use data from prior to the all-star selection date.

False Negatives On the other hand, there were some "false negatives" predicted by our model, i.e. players who may have been undeserving of their all-star selections. There were 19 players who our classifier considered "undeserving" of an all-star selection in multiple seasons. However, 7 of these players played the catcher position, a position that is not expected to be a good hitter. This further begs some considerations that were discussed in the False Positives section: position, and the need for more features besides hitting statistics.

6. Conclusions

Altogether, the statistics that we used seem to be a useful predictor for all-star selections, as our classifiers achieved test accuracy of greater than 92%. Some of the most important statistics for all-star prediction include Hits and Slugging Percentage.

However, there are a few considerations for more direction in future studies:

1. Consideration of what factors affect selection for different fielding positions
2. A wider range of features, including defensive or team statistics
3. Only using statistics compiled prior to all-star selection (i.e. excluding the second half of the season)

7. Acknowledgements

We would like to acknowledge Ian Lawson (iglawson@wisc.edu) for his contributions to the early stages of project development and his contributions to our Project Proposal. Additionally, all data was retrieved from baseball-reference.com [2] and statista.com [6].

8. Contributions

Both Eddie Morelli and Andrew Olson contributed to many steps of the project. Each downloaded data, coded a solution for pre-processing the data, built classifier models, tuned model hyperparameters, evaluated results, and wrote various sections of this report.

8.1. Andrew Olson

Andrew Olson downloaded the standard statistics from baseball-reference, and took the first steps in data exploration using Python along with the pandas library. His role in pre-processing included shaping the dataset for input into sklearn and mlxtend's model objects, as well as feature selection. Andrew built Logistic Regression models, k-Nearest Neighbor classifiers, Decision Tree classifiers, and Random Forest Classifiers on multiple subsets of features. He then conducted Principal Component Analyses over the features. Andrew implemented Grid Search with tenfold cross-validation to tune hyperparameters for various models. He contributed to the report in sections including, but not limited to, "abstract", "Related Work", "Proposed Method", and "Results and Discussion".

8.2. Eddie Morelli

Eddie Morelli downloaded the advanced statistics, all-star data, and team value data from baseball-reference and statista. His part in pre-processing was combining the data into one "master" dataset that would ultimately be used to build models. Eddie built various classifier models with different subsets of features, based upon Andrew's code. He also built ensemble classifiers using the Voting and Stacking methods. He implemented Grid Search to tune hyperparameters of various models. Finally, Eddie contributed to the report in sections including, but not limited to, "Introduction", "Experiments", and "Results and Discussion".

References

[1] A guide to sabermetric research. *Society for American Baseball Research*.

[2] Mlb stats, scores, history, & records. *Baseball-Reference*.

[3] Standard stats: Glossary. *MLB.com*.

[4] Statcast: Glossary. *MLB.com*.

[5] H. Chadwick. *The Base Ball Player's Book of Reference*. Published by J.C. Haney amp; Co., no. 100 Nassau Street., 1867.

[6] C. Gough. Mlb franchise values us 2021. *Statista*, Apr 2021.

[7] J. W. K. J. S. H. Jaewon Lee, Jonghoon Won. Predicting mlb hall of famer by machine learning approach.

[8] J. M. Karnuta, B. C. Luu, H. S. Haeberle, P. M. Saluan, S. J. Frangiamore, K. L. Stearns, L. D. Farrow, B. U. Nwachukwu, N. N. Verma, E. C. Makhni, M. S. Schickendantz, and P. N. Ramkumar. Machine learning outperforms regression analysis to predict next-season major league baseball player injuries: Epidemiology and validation of 13,982 player-years from performance and injury profile trends, 2000-2017. *Orthopaedic Journal of Sports Medicine*, 8(11):2325967120963046, 2020. PMID: 33241060.