Team 199 Final Project Report

Project Members:

Riley Buchanan - rbuchanan32

Desmond Kin Yeung Hui - dhui8

Jeffrey Karpan - jkarpan3

Hung-Hsi Lin - hlin83

Janaka Jayalalal Jayar Mohotti Bandara Rm -jjrmb3

Khoa Dang Vu Tran - ktran300

Abstract

Interpreting large amounts of data in numerical format from any domain can be challenging. Fantasy baseball users are often overwhelmed with tabular statistics and left without an easy way to interpret the data or customize the output based on user preferences. We introduce a new platform that combines player projection and rankings with an interactive user interface that allows the user to see visual representations of the data to help guide their decision making process. This platform uses gradient boosting regression models to provide accurate projections for the upcoming season. The rankings can be personalized based on the user's preferred statistics and willingness to take risks. We also provide interactive visualizations to help users see a player's career progression, or to compare two selected players and their projections. Visualizations are built using the D3 JavaScript library.

Keywords

Algorithms, Projections, Machine learning, Gradient Boosting Regression, User Interfaces, D3, Major League Baseball, Fantasy Baseball

Introduction

According to IBISWorld, the market size of fantasy sports is estimated to be $8.2 billion in 2020 in the US. A survey taken by the Fantasy Sports and Gaming Association in 2018 showed that 39% of fantasy players played fantasy baseball. With so many players, it creates a demand for sports analytics and projections because everyone wants to gain an edge on their opponents.

Currently, with Major League Baseball (MLB) sports sites, statistics and projections are only given in a tabular format. The data is, usually, only given using standard league settings. This tool is designed to accurately project MLB players' stats and rankings, and give the user the ability to put in their own league settings and get custom player projections and rankings based on those settings. Users have the freedom to select the factors they think are important to customize their own algorithm. This provides transparency to the user and gives them the ability to add or remove certain features that they believe should be part of the algorithm. In addition, one can visualize historic data and compare players using charts. Our platform offers the capability to study multiple players' statistics with friendly, straightforward charts. With visual tools, we believe that it gives users efficient ways to digest data, make draft and in-season decisions, and find trends.

To measure the accuracy of our approach, projections for previous seasons have been compared to the actual statistics from that individual season. Since the 2020 season has been delayed, benchmarking by comparing our projections to those made by other sports websites will not be possible.

Data and models are stored on GitHub. Projection models are written in Python. Visualizations are created using both Python and D3. Web framework is made using Flask, with MySQL being used as a database. Docker is used to package up the application.

Innovations

This platform offers several novel features that are not available on existing fantasy baseball platforms. For one, the models incorporate a player's variance and injury history into a total measure of risk. Users are able to input a risk aversion score that will adjust the rankings based on their preference. Users are also able to customize which statistics are being used to calculate the projected fantasy output, customizing the platform to a setting that matches their specific league scoring settings. Lastly, our interactive visualizations provide a more intuitive way for users to digest the large amounts of data. One can look at a player's career trajectory season by season for a selected statistic in an easy to read bar chart. This chart also shows games played for each season, so that the user can see if a particular season is affected by games played. One can also select two players and compare them visually with a bar chart showing future projections generated by our models, or compare their output in past seasons in which they both played.

Related Work

Numerous models have been previously built to do this, including Nate Silver's PECOTA, which makes predictions based on previous output of similar players, and Tom Tango's Marcel, which uses previous player data combined with the player's age [6,11,16,21]. Other projects have used a naïve Bayes model for predicting player statistics [1]. Huddleston used a Bayes model to conclude that the overall value of pitchers was higher than that for batters, which is useful insight in a fantasy baseball context and for our player ranking algorithm [22]. Support vector machine models and K-nearest neighbor models have been used for predicting output after separating players by position, which will be necessary in our model due to the format of fantasy baseball [20]. Multiple linear regression models are a popular choice given the number of possible predictors in the baseball context and have been shown to produce accurate projections [2,16,21]. These models require some form of variable selection, because a model with too many predictors can be inaccurate, tough to understand, and difficult to visualize. Many different methods for variable selection have been tested, from penalized regression models comparing Lasso and elastic net, to principal component analysis [4,18,19]. More advanced machine learning techniques can also be used, such as artificial neural networks and random forests [6,11,19]. Previous research suggests that principal component analysis and regression generate more accurate projections than naïve Bayes models or artificial neural networks and would also be easier to explain and interpret [12,19]. A potential problem in prediction is the various contexts from which players come to the MLB. Players may come from playing in college, a foreign league, or from Minor League Baseball. To adjust statistics for players from different environments, a combination of regression models and correlation coefficients can be used [17].

Once we have formidable projection models, we will need to optimize lineups based on the stats and risk preferences that the user specifies. Previous work has shown the limits of comprehensive constraint optimization with runtime on the order of $O(n^k)$, where k is the number of players in a lineup. With hundreds of players to choose from and lineups consisting of 20 or more players, this is unreasonable. Herman and Ntsoso suggest a beam search method to reduce runtime for the optimization calculation [12]. Frontin describes a methodology on how to find probability of a predicted WAR in upcoming seasons based on previous WAR and current age. Parts of this methodology helped guide how we determine the probability of different predicted statistics [13,14]. Allen has written a textbook that includes a derivation of mean and variance from a linear function [3]. This is used to help find the mean and variance of our linear models, which are plugged into a Gaussian distribution so that we can assess the probability of our models' projected outcomes.

For player comparison, Koop has previously compared offensive performance using stochastic frontier models and Bayesian analysis to find players with similar attributes [15]. PECOTA calculates a similarity scores to find a group of maximally similar players.

The ever-increasing amount of sports data highlights the need for efficient ways to digest them. SportViz attempts to tackle the need for better visualizations in sports statistics by helping the consumer be able to quickly derive meaning from aggregated data for a particular baseball game [5]. This is done by using preattentive processing attributes such as win/loss colors and segmented bars, which allows for quick pattern recognition. The idea of visualizing trends in player performance is introduced in Interactive Tool for Fantasy Football Analytics [20]. Here, visualizations are made to compare player fantasy points scored by position as well as week to week trends. These ideas can be expanded to compare different player statistics, not just fantasy points.

Research has been done to quantify different types of sports data in order to understand how to visualize it [10]. Three types were identified, box-score data, tracking data and meta-data. Effective ways to visualize this data is something that is still evolving as new data is introduced [10]. Another aspect for creating better visualizations is the user-interface design. The topic of end-user flexibility has been explored and emphasizes the need for greater flexibility [7]. This can be achieved with filtering, search, and level of detail adjustments within a dashboard. Both StatCast and SnapShot detailed their UI design for viewing specific types of sports data [8,9]. StatCast focuses on viewing spatiotemporal baseball data and SnapShot for viewing hockey shots data. The functions for querying and filtering data are called out in both applications. In addition, they put into consideration the need to export or share data for the purposes of additional analysis using different tools [8,9].

Methods

We have created models for projecting 16 different statistics, combining standard fantasy statistics for batters and pitchers with three extra statistics for each to allow for flexibility when adjusting for custom league settings. We have models for a pitcher's wins, saves, ERA, WHIP, strikeouts, innings pitched, losses, and home runs given up. We have models for a batter's home runs, RBIs, batting average, stolen bases, runs, on-base percentage, total bases, and slugging percentage. In order to create accurate statistical projections, various models have been tested using data from prior seasons to forecast results for the following season. Multiple linear regression models proved to be inaccurate. Exploratory analysis shows that the linearity and constant variance assumptions necessary for a good fit in linear regression do not hold for the majority of predicting variables. Stepwise variable selection and LASSO regression techniques did not significantly improve the models.

Eventually gradient boosting regression was proven to have better explanatory power for baseball performance in future seasons so it was selected as our model of choice. We used scikit-learn's GradientBoostingRegressor module in Python to develop our models. The models use two prior seasons of data for forecasting, and appropriate statistics have been normalized by game. 61 and 59 statistics each for season n-1 and n-2 were used for predicting season n's hitting and pitching statistics respectively.

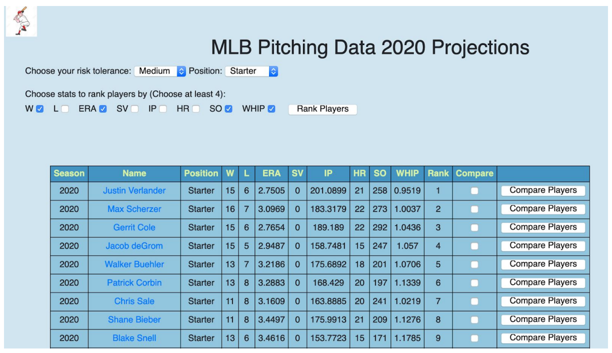
The majority of the dataset being used was retrieved using the pybaseball python package. There was a need to extend the dataset with a unique id for each player. The MLBAM id was used for this purpose. To map each player to their MLBAM id, a set of rules were used, such as, matching first/last name, matching team name and matching individual statistics for a given season. Another extension to the dataset was how many days each player spent injured in a given season. This is an important factor in determining the risk for players. Whether the player was placed on the 10-day or 60-day injured list is not a good measure for this because each player can stay on the list for an indefinite amount of days regardless of which list they are on. To determine when a player is no longer injured, the MLB data api was used, specifically the GET transactions endpoint. Each MLB transaction includes the date of the transaction as well as a description of the event. By using the data and parsing the event description, we were able to get reasonably accurate values for when a player was no longer injured.

One methodology to determine risk for a given player is to find the probability of a given value occuring in a statistical category given a player's age and previous performance. To do this, we used a linear model for each statistic we were predicting where the response was the percent change of the statistic from previous season to current season and the predictors were age and previous season's stat value. We were then able to use the linear model to derive the mean and variance by inputting age and previous performance. The mean and variance is then used as inputs to find a Gaussian distribution to be used to find the probability. These probabilities will be used as input, among other features, to find a singular risk value for each player season. To obtain a singular risk value, probabilities for each individual category were divided into three groupings. Group value of 1 represented the upper third of the probabilities, 2 being the middle third and 3 being the final third. These groupings along with how many days a player spent injured in the previous season were used as inputs for a K-Means algorithm with k=3. The resulting cluster labels represented the risk factor associated for each player with 0=Low, 1=Medium and 2=High.

The predicted stats, risk factor and user input were then used to compute the overall rank of each player for the 2020 season. User input impacts the rankings in two ways. The user is prompted to select from a list of statistics from which they wish to be used in the rankings, these should correspond to the stats that are used in their fantasy leagues. Any category not selected by the user is not used in computing rankings. The user is also prompted to enter their risk tolerance. Depending on the user selection, different weights are applied to player statistics depending on the risk factor assigned to them, which will ultimately affect the overall rank value.

The platform is a web application. We have usedg Flask for the web framework. This framework is written in Python and it allows us to use HTML and JavaScript for our frontend. Our Flask application uses MySQL as a database. To package up the application, we use Docker. There are two containers: one for the application and one for the MySQL database. A script was also made to insert the batting data and pitching data into the database.The pitching data is separated by starters and relievers.

The home page gives the user the option to select an icon for pitchers or for batters. The user is then redirected to a table of 2020 projections for the selected position, in a table that is ranked based on default values for statistics selected and risk level. Figure 1 shows the top of the webpage showing the table of pitchers. The user has the ability to change the rankings by changing the selected statistics or changing the risk tolerance level. One can also use the drop-down button to toggle between starters and relievers. The columns can be sorted by statistic if the user clicks on the column header.



MLB Pitching Data 2020 Projections

Choose your risk tolerance: Medium Position: Starter

Choose stats to rank players by (Choose at least 4):
W ✓ L ☐ ERA ✓ SV ☐ IP ☐ HR ☐ SO ✓ WHIP ✓

Season	Name	Position	W	L	ERA	SV	IP	HR	SO	WHIP	Rank	Compare	
2020	Justin Verlander	Starter	15	6	2.7505	0	201.0899	21	258	0.9519	1	☐	Compare Players
2020	Max Scherzer	Starter	16	7	3.0969	0	183.3179	22	273	1.0037	2	☐	Compare Players
2020	Gerrit Cole	Starter	15	6	2.7654	0	189.189	22	292	1.0436	3	☐	Compare Players
2020	Jacob deGrom	Starter	15	5	2.9487	0	158.7481	15	247	1.057	4	☐	Compare Players
2020	Walker Buehler	Starter	13	7	3.2186	0	175.6892	18	201	1.0706	5	☐	Compare Players
2020	Patrick Corbin	Starter	13	8	3.2883	0	168.429	20	197	1.1339	6	☐	Compare Players
2020	Chris Sale	Starter	11	8	3.1609	0	163.8885	20	241	1.0219	7	☐	Compare Players
2020	Shane Bieber	Starter	11	8	3.4497	0	175.9913	21	209	1.1276	8	☐	Compare Players
2020	Blake Snell	Starter	13	6	3.4616	0	153.7723	15	171	1.1785	9	☐	Compare Players

Fig 1. Projected 2020 statistics for pitchers, with options to change risk tolerance level, stats to include in the rankings, and to compare players.

This page also gives the user the ability to choose two players of interest and be shown a graph comparing the two player's statistical output side-by-side.

If the user clicks on a player's name, they are directed to a new page where that player's career statistics are shown in tabular format, ordered by season. On this page, the user can choose a statistic of interest and see a graph showing that player's career performance of that statistic by season. Figure 2 shows an example, displaying Mike Trout's career RBI output by season. While his total RBIs in 2011 were low, the blue dot showing games played gives the necessary context that he only played 16 games that season.

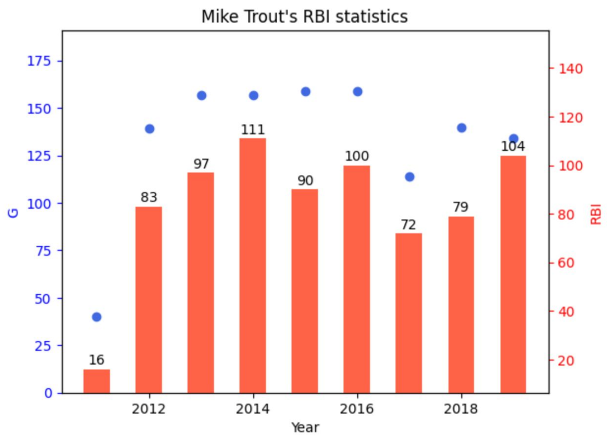


Fig 2. Mike Trout's career RBIs by season

Evaluation

While building our predictive models our goal was to predict an individual players performance based on their previous statistics. A single model was built for each of the sixteen stats we were predicting. To quantify the quality of our models we used two quality metrics: mean absolute error (MAE) and R^2. Our testing methodology was set up in Python as follows:

1. Load up all player stats from 1996-2019 and separate out a randomly selected 70% for model training and 30% for testing. Fix the seed so subsequent iterations would be done with the same splits.
2. Build the model with the training data.
3. Use the model to predict the test data and compare it to the actual historical data. We used the mean_absolute_error and r2_score functions from the sklearn module.

We initially tried multiple linear regression but the R^2 values were mostly < 0.2. Variable selection increased R^2 values, but still not higher than .35. Residual analysis suggested that many predictors did not have linear relationships with the response variable, so the models would not be good without transformations of the data. We moved on to gradient boosted trees using sklearn's GradientBoostingRegressor. We varied the n_estimators, learning_rate, and max_depth hyperparameters until the R^2 values went up to ~0.5.

In the end we used n_estimators = 1000, learning_rate = 0.01, and max_depth = 3. The n_estimators and learning_rate parameters are interrelated: the smaller the learning_rate the higher n_estimators must be for a sufficient fit. Too high of a learning_rate resulted in overfitting. The learning_rate of 0.01 was sufficiently small and combined with 1000 estimators empirically provided the best quality metrics.

Overfitting also occurs as max_depth increases, on the other hand with max_depth < 3 the fit suffered because the trees were too weak to fit the data when only 1000 of them were used.

At this point we were modeling stats only based on the previous year (n-1). We next added year n-2 to the features as well. The change to MAE and R^2 were negligible. Lastly, we normalized the stats by game where appropriate. This led to a 5% improvement in R^2 for batting stats, and a big reduction in R^2 for pitching stats.

As part of our test bench we had a model building dashboard which provided visualizations to aid in model and parameter selection. Figures 3 and 4 show some of those visualizations. Fig. 3 compares the values predicted by the model versus the actual values for our test set and displays the quality metrics of MAE and R^2. Fig. 4 plots n_estimators on the x-axis versus deviance (the metric of error in the predictions used to build the model) on the y-axis for both the training and test data. This is useful to identify a range of acceptable estimators to use for a given learning_rate. If n_estimators get too large the test data curve's deviance will begin to increase which is a sign of overfitting.

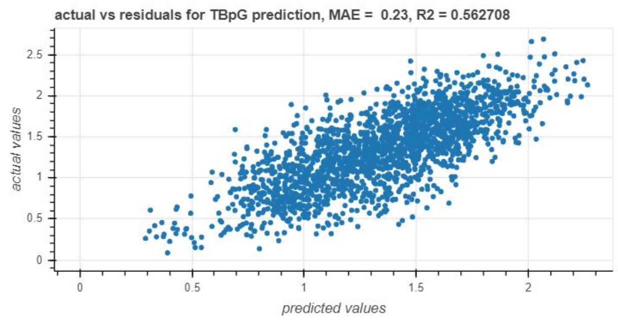


Fig 3. Actual versus predicted total bases per game (TBpG)

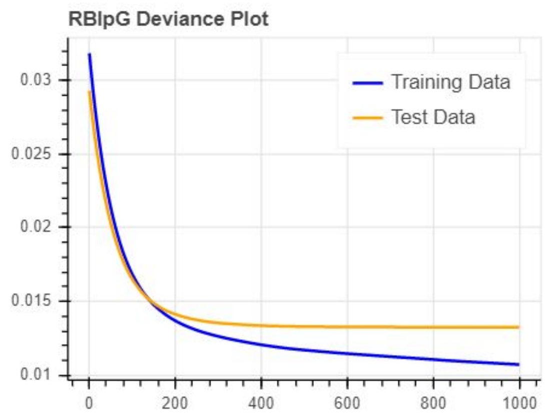


Fig 4. N_estimators versus deviance for RBIs per game (RBIpG)

Discussion

Baseball has been collecting stats longer than most sports have, and was the first sport to widely use advanced analytics to strategize. Because of this, pre-cleaned datasets are readily available for the baseball analytics community. Data was made available via open source packages and APIs. Retrieving 20 years of player season by season data was made much simpler than originally anticipated because of the PyBaseball Python package. The resulting dataset had all of the features we needed and so much more, which enables future work to be done using this dataset. While the dataset was comprehensive in that it has a large number of statistics, it was lacking in some categories, namely providing a unique identifier and injury history for players. Work to include this as part of the dataset proved to be time consuming (e.g., unique ID lookup API calls, manual resolution) and difficult to achieve accurately for many reasons. For example, to determine injury history, evaluation of event text was required. Future work to include this as part of an open source package like PyBaseball would be worthwhile.

For modeling statistics, we found that multiple linear regression was not very accurate. Even variable selection did not result in a significant increase to the R^2 value of the models. Residual analysis showed that many predictors do not have linear relationships with the stat being predicted, yet transformation of some predictors failed to significantly improve the models.

We found that using gradient boosted trees provided a better fit than linear regression. A given season's statistics were predicted based on the two previous seasons and statistics were normalized by game when appropriate. R^2 values of > 0.5 were common with MAE ~0.1-0.2 of the average of the given statistic, which we found to be powerful enough for ranking purposes. To further improve the fit additional models could be explored such as random forests or neural networks. Expanding the independent variables beyond those statistics available via the PyBaseball package could also prove worthwhile. It is noteworthy that the three batting statistics with the lowest R^2 were all percentages that had narrow ranges amongst the population. Perhaps some normalization techniques could improve the fit.

There are two pitching statistics that the GradientBoostingRegression algorithm did not perform as well on: ERA and WHIP. Exploratory analysis suggests that these two stats have high variance from season to season, which could explain why it was difficult to create an accurate model for these statistics.

For access to the datasets and models used for this project, check out: https://github.com/cse6242teamsport/fantasy_baseball

Distribution of work

For the proposal, progress report, and final deliverables the work has been evenly distributed.

References:

[1] Albert Jim. Improved component predictions of batting and pitching measures. *Bowling Green State University* (2015)

[2] Alexander Stroud, Brian Bierig, Jonathan Hollenbeck. Understanding Career Progression in Baseball Through Machine Learning

[3] Allen, Michael. "Derivation of the Mean and Variance of a Linear Function." *Understanding Regression Analysis*, Springer, 1997, pp. 191–216.

[4] Andre Elisseeff. Isabelle Guyon. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research 3* (2003) 1157-1182

[5] Andy Cox. John Stasko. SportVis: Discovering Meaning in Sports Statistics Through Information Visualization. *College of Computing / GVU Center Georgia Institute of Technology*

[6] Arlo Lyle.Baseball Prediction Using Ensemble Learning. *University of Georgia*. (2007)

[7] Bartram, L. Correll, M. Fisher, D. Sarikaya, A. Tory M. What Do We Talk About When We Talk About Dashboards? *IEEE Transactions on Visualization and Computer Graphics,* vol. 25, no. 1, pp. 682-692, Jan. 2019

[8] Boyle, J. M. Pileggi, H., Stolper ,C. D. Stasko J. T., SnapShot: Visualization to Propel Ice Hockey Analytics. *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2819-2828, Dec. 2012.

[9] Cervone, D. Chiang, J. Dietrich C. Lage,M. Ono, J. P. and Silva, C. T. StatCast Dashboard: Exploration of Spatiotemporal Baseball Data. *IEEE Computer Graphics and Applications*, vol. 36, no. 5, pp. 28-37, Sept.-Oct. 2016.

[10] Carpendale, S [1]. Perin, C [1,2]. Stasko, J. T [3]. Stopler, C. D [4]. Vuillemot [5,] R. Wood [2], J. State of the Art of Sports Data Visualization. (2018)

1 Department of Computer Science, University of Calgary, Canada.

2 Department of Computer Science, City, University of London, UK.

3 School of Interactive Computing, Georgia Institute of Technology, USA.

4 Department of Mathematics and Computer Science, Southwestern University, USA.

5 École Centrale de Lyon, Université de Lyon, France.

[11] Daniel Calzada. DeepBall: Modeling Expectation and Uncertainty with Recurrent Neural Networks. *SABR Analytics Conference.* (2019)

[12] Eric Hermann and Adebia Ntoso. Machine Learning Applications in Fantasy Basketball. *Stanford University*.

[13] Frontin, Cory. “Baseball ProGUESTus: Casting Some Uncertainty on How Players Age.” *Baseball Prospectus*, 17 Oct. 2019, www.baseballprospectus.com/news/article/54551/baseball-proguestus-casting-some-uncertainty-on-how-players-age/

[14] Frontin, Cory. “Baseball ProGUESTus: Casting Uncertainty on How Players Age (Part 2).” *Baseball Prospectus*, 24 Oct. 2019, www.baseballprospectus.com/news/article/54726/baseball-proguestus-casting-uncertainty-on-how-players-age-part-2/

[15] Gary Koop (2002) Comparing the Performance of Baseball Players, *Journal of the American Statistical Association*, 97:459, 710-720, DOI: 10.1198/016214502388618456

[16] Johnathon Tyler Clark. Regression Analysis of Success in Major League Baseball. University of South Carolina - Columbia (2016)

[17] Kevin Salador. Forecasting Performance of International Players in the NBA. *MIT Sloan Sports Analytics Conference* (2011)

[18] Mushimie Lona Panda. Penalized Regression Model For Major League Baseball Metrics. *University of Georgia*. (2014)

[19] Nicholas Aaron King. Predicting A Quarterback's Fantasy Football Point Output For Daily Fantasy Sports Using Statistical Models. *The university of Texas Arlington* (2017)

[20] Parikh, Neena Interactive Tools for Fantasy Football Analytics and Predictions Using Machine

Learning. *Massachusetts Institute of Technology* (2015).

[21] Sarah Reid Bailey. Forecasting Batting Averages in MLB. *Simon Fraser Universit*y (2017)

[22] Scott D. Huddleston. Hitters vs. Pitchers: A Comparison of Fantasy Baseball Player Performances Using Hierarchical Bayesian Models. *Brigham Young University* (2012)