Does Tabular Data Augmentation Improve Predictive Performance for Major League Baseball Drafting?

JACKSON GAZIN* and ANH NGUYEN*, Wake Forest University, USA

Although many college baseball players dream of a career in Major League Baseball (MLB), the vast majority do not make it. The job of effectively scouting the right player for a major league team is difficult and requires substantial resources. We aim to implement an effective pre-trained and fine-tuned classification model to predict whether a college baseball player will make it to the MLB or not, given their performance and competition. We address class imbalance in the college baseball data using a conditional tabular generative adversarial network to augment synthetic data. We found that the synthetic data did not improve our model's overall model's predictive performance with a significantly lower specificity and a slightly higher sensitivity: it did a better job at predicting those who make the MLB to actually make the MLB. Our best model is a logistic regression model using only the real data, with an accuracy of 0.7501. In fact, our final model correctly predicted 85.57% of those in our test set who did make the MLB and 78.22% of those who did not make the MLB.

CCS Concepts: • **Computing methodologies** → **Cross-validation**; **Supervised learning by classification**; **Classification and regression trees**; • **Applied computing** → **Decision analysis**.

Additional Key Words and Phrases: major league baseball, drafting

ACM Reference Format:
Jackson Gazin and Anh Nguyen. 2018. Does Tabular Data Augmentation Improve Predictive Performance for Major League Baseball Drafting?. *ACM Trans. Graph.* 37, 4, Article 111 (August 2018), 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The Major League Baseball (MLB) is a professional baseball organization in the United States of America. The MLB draft is the main way to select amateur baseball players from high schools, colleges, and other amateur baseball clubs for MLB teams. Drafting the right player for a team is a complicated matter. This process involves a combination of traditional in-person scouting by professional scouts and empirical analysis of performance using statistics, data mining, machine learning, and operations research. The analytics teams examine baseball statistics like batting averages, runs batted in (RBIs), and earned run averages (ERAs). Teams want to ensure they are getting the most out of their draft picks and prefer to draft players who have a high likelihood of making it to the MLB. It should be noted that although all players drafted by an MLB team become professional baseball players, they must advance through many minor league teams to reach the MLB, which is an extremely challenging task. Teams want to ensure that their draft picks have a high likelihood of making the MLB.

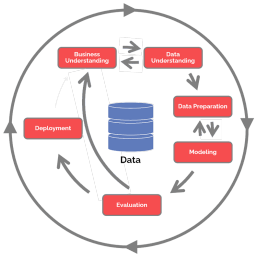

Fig. 1. **CRISP-DM diagram**. Diagram of the six different phases in CRISP-DM and how they relate to each other.

Classification task

We therefore will build a classification model which predicts whether college hitters who played professional baseball will make the MLB or not. We will utilize information on all college hitters who had their last year of college baseball from 2012 to 2019.

Data augmentation

Although there are many college baseball players who played professionally in our training set (3428 players), only 314 players actually made it to the MLB. Therefore, we will consider two models, both trained on datasets with an equal number of players who made it to the MLB and those who did not.

Our first model will use K-means downsampling to train on a dataset composed of 314 players who made it to the MLB and 314 players who did not.

Our second model will use Generative Adversarial Networks (GANs) to upsample the data so that we have 3114 players who made the MLB and 3114 players who did not in our dataset. Note that, although this is a much larger training set, it comprises 2800 synthetic data points or GAN-generated data points of players who made the MLB.

The Cross Industry Standard Process for Data Mining (CRISP-DM) is the process model used for data science projects. The sequential and iterative phases in CRISP-DM are: business understanding, data understanding, data preparation, modeling, evaluation, and deployment (Figure 1). All steps in the CRISP-DM were consistent across these two methods, except for the difference in up-sampling and down-sampling. More specifically, we built the same pipelines across each stage of the CRISP-DM process for both models, training on the original dataset and the dataset with synthetic data. The fitted pipelines, unsurprisingly, however, yielded different results given the differences in the datasets.

We found that the synthetic data did not improve the accuracy score of our predictive model.

*Both authors contributed equally to this research.

Authors' address: Jackson Gazin, gazij22@wfu.edu; Anh Nguyen, nguyp22@wfu.edu, Wake Forest University, 127 Manchester Hall, Winston-Salem, North Carolina, USA, 27109.

ACM 0730-0301/2018/8-ART111
https://doi.org/XXXXXXX.XXXXXXX

2 PRIOR WORKS

2.1 Sabermetrics

MLB scouting is a challenging task, as it involves predicting the long-term performance of a player based on their current play. Sabermetrics, the statistical analysis of baseball, measures in-game activity. Predictive models based on baseball sabermetrics have been extensively used in scouting. For example, analyzing career trajectories is a common statistical analysis performed for baseball players [Baumer 2018]. Various techniques in Statistics, Data Mining, and Machine Learning have been applied to the field of sabermetrics to improve predictive performance. Data Mining is particularly useful for gaining insights, analyzing games, and supporting decision-making. Ofoghi et al. provided a framework for using data mining in sports data [Ofoghi et al. 2013]. Mitchell proposed KATOH, a statistical model to predict MLB scouting using statistics from Minor League performances [Mitchell 2014]. Danovitch built a model to predict whether a player will be scouted into the MLB using text mined from written scouting reports conducted by professional scouts, achieving an accuracy of 0.8249 [Danovitch 2019]. Gerber and Craig proposed a mixed effects multinomial logistic-normal model for baseball scouting that can quantify the uncertainty of model predictions [Gerber and Craig 2021]. Smith et al. built a Bayesian classifier to predict Cy Young Award winners in American baseball with an accuracy of 0.8 [Smith et al. 2007].

Models used for actual player scouting are private proprietary models owned by baseball teams. Thus, we do not know how these models are trained or how they perform. We aim to build an accurate and publicly available pre-trained model that can accurately classify whether a college hitter will make it to the MLB or not.

2.2 Tabular data augmentation and class imbalance

Data augmentation is a powerful technique for creating synthetic data from pre-existing data. This approach addresses the problem of small sample size and class imbalance since it allows us to sample the desired amount of synthetic data to increase our sample size and the number of samples in the minority classes. Many GAN-based approaches have been developed for learning and simulating numerical and categorical datasets [Li et al. 2021], [Aziira et al. 2020].

Tabular data contain a mixture of continuous and discrete variable columns. Modeling the distribution for each column in a tabular dataset and sampling from these distributions is a challenging task. Xu et al. introduced CTGAN, a GAN-based synthetic tabular data generator that addresses issues with multi-modal distributions for continuous variables and imbalanced groups for discrete variables [Xu et al. 2019]. CTGAN was found to outperform existing Bayesian learning networks in modeling tabular data.

Habibi et al. implemented the CTGAN model to address the issues of unlabeled and imbalanced tabular datasets in botnet detection [Habibi et al. 2023]. Using a multilayer perceptron on data with synthetic observations, their model achieves an accuracy of 0.9893. An et al. proposed a method that combines K-means clustering and CTGAN to address imbalanced tabular datasets for prediction tasks [An et al. 2021]. They demonstrated that this method achieves a higher F1-score and G-means metric compared to SMOTE while also helping with overfitting.

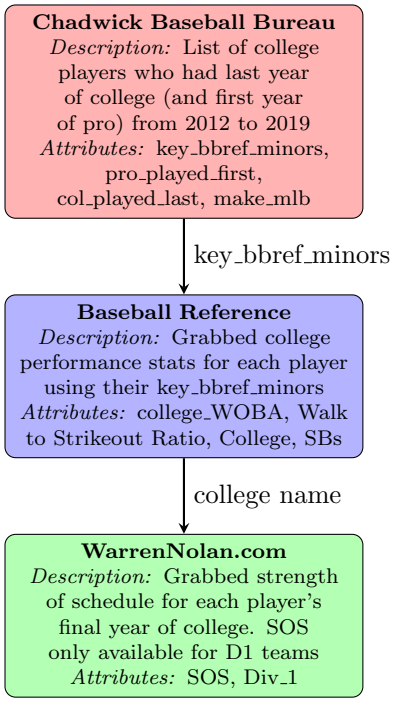

Fig. 2. **Web Scraping Flowchart**.We harnessed data from the Chadwick Baseball Bureau database to obtain the Baseball Reference IDs for college players who began their professional baseball careers between 2012 and 2019. Using these IDs, we extracted information from Baseball Reference, including the College Weighted On-Base Average (wOBA). Lastly, we utilized the players' college names and years to retrieve their corresponding season's Strength of Schedule

3 METHODS

3.1 Data Collection

The Chadwick Baseball Bureau is an open-source database that provides an extensive amount of information. Interestingly, one of the authors (Jackson Gazin) is included in the database for having played three years of college baseball. Our data collection and web scraping process stemmed from this database to retrieve information both from Baseball Reference and Warren Nolan (Figure 1).

The `baseballr` package, developed by Petti in 2021 [Petti and Gilani 2021], is an R package that allows users to access data on all players in the Chadwick Baseball Bureau database. We used this package to select players who had their last year of college baseball between 2012 and 2019 and also played some professional baseball. We chose this timeframe because these years are recent enough to exhibit similar performance trends. College baseball performance has varied over the years, but players from this timeframe have had sufficient time to potentially make it to the MLB. Most players need at least four years before we can determine if they can reach the

MLB or not, as demonstrated in this chart [Gaines 2013]. We decided to use 2019 as our endpoint. While we considered excluding this year due to its proximity to the current year, we found no significant change in performance when we omitted it.

The dataframe returned by `baseballr` only contained information about keys to different baseball databases to access those players, along with information on whether each player made the MLB or not and unimportant demographic information (for our problem). We then filtered this dataset to only contain players who had information on whether they made the MLB or not.

We used this database and each player's associated Baseball Reference key to scrape information on each player's College Weighted On Base Percentage, position in college, school name from Baseball Reference, stolen bases, walk-to-strikeout ratio, and age from WarrenNolan.com [Nolan [n. d.]]. Baseball Reference, which is a popular baseball database, offers valuable insights. A player's on-base percentage indicates the percent of time a batter reaches base compared to the number of times they have a plate appearance. wOBA, which adjusts a player's on-base percentage by the value of each method of reaching base, is likely the most informative metric.

However, not all colleges are created equal, and competition also varies across different levels of College Baseball in the US: Division 1, Division 2, Division 3, NAIA, and JUCO. We therefore utilized the Warren Nolan database which has information on each team's Strength of Schedule going back to the year 2012. Note that this strength of schedule information was only available for Division 1 teams. This web scraping allowed us to create a new indicator variable called Div_1 which indicated whether or not a player played Division 1 in their last year of college. For players who did not play Division 1, we assigned an SOS of 0. We were then ready to begin our CRISP-DM process. Note, all web scraping abides by the rules of each website we used, as seen in our functions.

We ended up with nine variables at the end of our data collection process which were suitable to begin our CRISP data mining process: key_bbref_minors college_WOBA, walk_to_strikeout, age,college, SB, Div1, and Position (Table 1).

We were then ready to begin the CRISP Data Mining Process on this data set

Table 1. **Data set variables.** A data dictionary of all variables in our dataset after importing from `baseballr` package and scraping the internet

Variable	Description
key_bbref_minors	Unique identifier for players in the Baseball Reference Minor Leagues database.
make_mlb	Binary variable indicating whether the player made it to Major League Baseball (1 for yes, 0 for no).
college_WOBA	Weighted On-Base Average (WOBA) of the player during their college baseball career.
walk_to_strikeout	Ratio of walks to strikeouts for the player during their college baseball career.
age	Age of the player.
college	Name of the college where the player played college baseball.
SB	Stolen bases by the player.
SOS	Strength of Schedule (SOS) for the college baseball team.
Div1	Binary variable indicating if the college is in Division 1 (1 for yes, 0 for no).
Position	Position played by the player.

3.2 Phase 2: Train-Test-Split, Row Selection, Feature Engineering and Class-Imbalance

We then split the data set into the train and testing set with a ratio of 8:2. Note, that throughout this process any steps that involved randomization were used with a random seed so our results could be reproducible.

3.2.1 Missing data and Feature Engineering. In the first part of our Phase 2 pipeline, we identified approximately 146 NA rows in our training set. These were primarily players who only participated in independent league baseball, which lacks affiliation with MLB teams. We chose to remove these rows as they were less informative. However, we didn't exclude all players from independent leagues, recognizing that some might advance to minor league teams. We then performed feature engineering on our dataset by one-hot encoding our categorical variables such as college and position. This increased our dimension from the 8 features for learning (Table 1) to 591 features for learning. However, at this point, our training set comprised 3114 players who did not make the MLB and 314 who did. As shown in [Sun et al. 2009], when performing classification, it is ideal to train a model on a dataset with a balanced response variable distribution. We, therefore, split our remaining pipelines into two paths; one where we down-sampled our dataset to 314 players who made the MLB and 314 who did not using K-Means Clustering, and one where we up-sampled our dataset using CTGAN synthetic data generation to have 3114 players who made the MLB and 3114 who did not in our dataset. Note, even though this resulted in two different models and datasets, we fit the same pipeline structure in both processes.

Table 2. **Count value by whether the players make it to the MLB**. Value counts of the the Make MLB before balancing the training set 3114 of our training data points were comprised of players who did not make the MLB while 314 were comprised of players who did.

Make MLB	Count
0 (No)	3114
1 (Yes)	314

3.3 Phase 2: Data Balancing

3.3.1 Down-Sampling with KMeans Clustering. Since we had only 314 rows for the minority class (those who were drafted), we first opted to balance our data by sampling 314 rows from the class of those who did not make it to the MLB using K-means clustering [Duan et al. 2020]. We considered many different values of K and chose the value of K that offered the best combination of high cross-validation mean accuracy and low standard deviation

in accuracy. Using K= 50 provided the best combination of mean cross-validation accuracy and standard deviation (Table 3). For each cluster, we selected an equal number of points closest to that centroid. More formally, we clustered our training data points of players who were not drafted into 50 clusters, and down-sampled by taking approximately an equal number of players from each cluster. Our down-sampled training set then had an equal number of players who made the MLB and did not make the MLB (314 each). Note that for most of this process, we evaluated each choice based on accuracy since our training data is balanced. It has been shown that when a set is balanced, we can optimize our model based on accuracy [Thabtah et al. 2020].

Table 3. **K-Means Cross-validation Accuracy results.** K Cross Validation Accuracy Scores for down sampling. We can see that K=50 has the best performance. Logistic Regression was used for these scores.

Method	Mean Accuracy	Standard Deviation
Random Sampling	0.7518	0.0518
KMeans 314 Equal	0.7548	0.0271
KMeans 5 Equal	0.7675	0.0205
KMeans 10 Equal	0.7644	0.0339
KMeans 20 Equal	0.7564	0.0313
KMeans 25 Equal	0.7754	0.0388
KMeans 50 Equal	**0.7818**	**0.0285**
KMeans 75 Equal	0.7612	0.0279
KMeans 100 Equal	0.7707	0.0204

3.3.2 Data Balancing with Up-Sampling (CTGan). For our synthetic data model, rather than downsampling our majority class (those who did not make it to the MLB) to have as many data points as our minority class (those who did make it to the MLB), we up-sampled our minority class to match the size of our majority class. Formally, we built a CTGAN model using our cleaned training set (feature engineered and NAs removed) to generate a large number of synthetic data points. We then sampled 2800 synthetic data points from this output, which had the value of 'make_mlb' equal to 1. This required creating 44800 synthetic data points in total. Furthermore, we decided not to consider any college variables in this synthetic data creation, as these indicator variables were found to decrease performance (and vastly increase computation time in the generation process). This significantly reduced the number of variables we had at the end of Phase 2 in our CRISP-DM process. The new dataset, a combination of the original real data and the synthetic data, now has 6228 rows and 24 columns.

Although the synthetic box plots of numeric variables generally match the distribution of the variable with the same response value, they are not robust to outliers and seem to model the distribution without deference to outliers (Figure 3).

3.4 Phase 3: Feature Selection

3.4.1 Feature Selection for Down-Sampled Data. In Phase Three for our down-sampled dataset, we selected the best route for feature selection, scaling, and normalizing by performing a grid search. This search considered different scalers, normalizers, and the number of

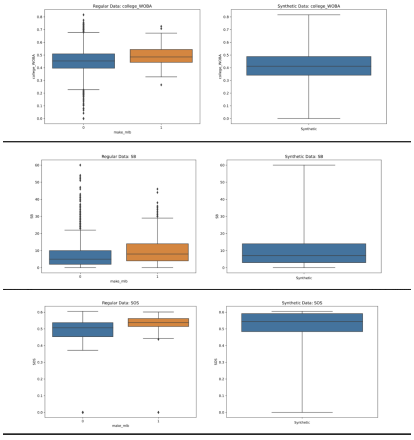

Fig. 3. **Box plot of numerical variable distribution compare to their synthetic counterpart.** Figure 2 presents a side-byside boxplot comparison for three distinct variables: college_WOBA, SB, and SOS, contrasting their distribution in the original dataset against their synthetic counterparts created via GAN. Intuitively, our synthetic data seems to match the general shape of the original data, without any deference to outliers. In general, the GAN appears to treat outliers as regular points and builds a distribution that encapsulates outliers as non-outliers.

columns to reduce our dataset. For each K or set of features considered, we selected the set of features with the function SelectKBest. SelectKBest has been shown to outperform other feature selection algorithms in data mining tasks [Saeed and Hama 2023; Zulfiker et al. 2021]. This method, from the Scikit-Learn library, selects the top K features based on univariate statistical tests. It compares the importance of each feature against the target variable, effectively allowing us to retain only the most influential variables. By using these methods, we aimed to optimize the feature space to enhance the model's performance while avoiding overfitting and maintaining computational efficiency. This step was crucial in refining our model, ensuring that it was fed with the most relevant and impactful features for predictive analysis. We then chose the best combination of scaler, normalizer, and K that had the best combination of high mean cross-validation accuracy, low cross-validation standard error, and a low value of K. Let X be the Mean Cross-Validation Accuracy for a choice in our grid search, Y be the Standard Deviation of the Cross-Validation Accuracy, and Z be the value of K. Our scoring involved these three variables to choose the K, scaler, and normalizer that had preferred higher X values, lower Y values, and low Z values, all else being equal (Equation **??**). 3.4.1).

$$X + Y * -.05 + .001 * Z \tag{1}$$

We found that using 8 features, combined with a robust scaler and no normalizer, was the best way to reduce our training set (Table 4). Note that we used our baseline model, Logistic Regression, to obtain these cross-validation accuracy metrics.

We ended up selecting eight features: college_WOBA, Age, SOS, college_Arizona, college_Clemson, college_Florida, college_Louisiana State, and Div1_1 (Table 5).

Table 4. **Cross Validation Accuracy Scores of SelectKBest.**For our down-sampled dataset, we observed very similar performance across different values of K, with K=8 performing the best. Logistic Regression was used to calculate these scores.

K	Mean Accuracy	Std Dev	Scaler and Normalizer
1	0.7183	0.0431	StandardScaler, MinMaxScaler
3	0.7358	0.0367	RobustScaler, None
5	0.7501	0.0305	StandardScaler, None
6	0.7485	0.0203	RobustScaler, None
7	0.7501	0.0385	None, None
8	**0.7437**	**0.020**	**RobustScaler, None**
9	0.7533	0.0360	None, None
10	0.7612	0.0236	StandardScaler, None
15	0.7676	0.0341	RobustScaler, None

Table 5. **Features selected using SelectKBest with $K = 8$.** We selected college_WOBA, age, SOS, college_Arizona, college_Clemson, college_Florida, college_Louisana State, and Div1_1 as features in Phase Three of Feature selection for our down-sampled data set.

Column Name	Description
college_WOBA	College Weighted On-Base Average
age	Age of the Player
SOS	Strength of Schedule
college_Arizona	Played at University of Arizona (1 = Yes, 0 = No)
college_Clemson	Played at Clemson University (1 = Yes, 0 = No)
college_Florida	Played at University of Florida (1 = Yes, 0 = No)
college_Louisiana State	Played at Louisiana State Univ. (1 = Yes, 0 = No)
Div1_1	Played in Division 1 College (1 = Yes, 0 = No)

When we examine the boxplots for two of our numeric variables, it becomes more intuitive why the Robust Scaler was an adequate choice. Both variables have visible outliers, and the Robust Scaler was able to handle them appropriately (Figure 4). Notably, the outlier structure was inherent in the way we defined Strength of Schedule, as schools that were not Division 1 automatically received a Strength of Schedule score of 0.

We can now proceed to model selection for this data set

3.4.2 Feature Selection for Up-sampled Data Set. In Phase 3, we trained the same pipeline on our enlarged GAN dataset and found that it was best to reduce our dataset to 7 features with a Standard Scaler and a MinMax Normalizer: college_WOBA, walk_to_strikeout, age, SB, SOS, Position_Rightfielder, and Position_Second (Table 6). Again, our different choices of K had similar accuracies compared to each other, albeit much lower than our down-sampled dataset. However, in this case, our model much preferred using a Standard Scaler and MinMax Normalizer as opposed to a Robust Scaler, and ultimately decided on using K=7 with this scaler and normalizer (Table 7).

We are now ready to proceed for model selection for both our up-sampled and down-sampled data sets.

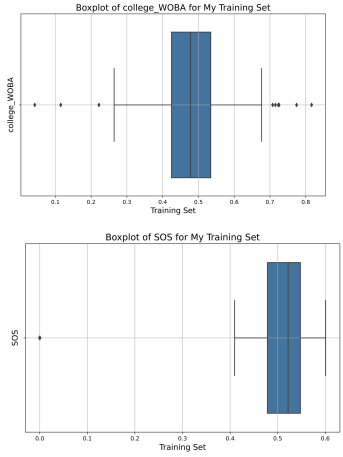

Fig. 4. **Boxplots of numeric variables: College_WOBA and SOS**. The plot shows why a Robust Scaler was adequate. Both College_WOBA and SOS have visible outliers.

Table 6. **Features selection for synthetic dataset.** For our synthetic dataset, we selected the following features: college_WOBA, walk_to_strikeout, age, SB, SOS, Position_Rightfielder, and Position_Second.

Column Name	Description
college_WOBA	College Weighted On-Base Percentage
walk_to_strikeout	College Walk to Strike Out Ratio
age	Age of the Player (At First Year of Pro Baseball)
SB	Number of Stolen Bases
SOS	Team's Strength of Schedule in Last Year of College
Position_Rightfielder	Player's Position is Right Field (1= Yes, 0 = No)
Position_Second	Player's Position is Second Base (1= Yes, 0 = No)

Table 7. **Best CV Accuracy Scores across K for down-sampled data set.** We observed very similar performance across different values of K, with K=8 performing the best. Logistic Regression was used to calculate these scores.

K	Mean Accuracy	Std Dev	Scaler and Normalizer
1	0.5764	0.0058	StandardScaler, MinMaxScaler
3	0.6652	0.0152	StandardScaler, MinMaxScaler
5	0.6626	0.0179	StandardScaler, MinMaxScaler
6	0.6659	0.0130	StandardScaler, MinMaxScaler
7	**0.6700**	**0.0137**	**StandardScaler, MinMaxScaler**
8	0.6662	0.0118	StandardScaler, MinMaxScaler
9	0.6646	0.0155	StandardScaler, MinMaxScaler
10	0.6625	0.0147	StandardScaler, MinMaxScaler
15	0.6777	0.0154	StandardScaler, MinMaxScaler

3.5 Model selection

3.5.1 Model Selection for our Down-Sampled Data Set. Utilizing the processed dataset, we conducted a comprehensive grid search

that spanned several classification algorithms and their associated hyper-parameters. The selection included a Decision Tree Classifier, a Random Forest Classifier, an XGBoost Classifier, a Support Vector Machine, and a k-Nearest Neighbors algorithm. These models were chosen for their widespread application and proven performance across a variety of classification tasks. Decision Trees provide a solid foundation for more complex models and offer interpretability with straightforward decision rules [Umadevi and Marseline 2017]. Random Forests, as ensembles of Decision Trees, generally perform better by reducing overfitting while maintaining accuracy. XGBoost is renowned for its efficiency and effectiveness in capturing non-linear relationships [Chen and Guestrin 2016]. Support Vector Machines are known for their robustness in high-dimensional spaces, particularly effective when the number of dimensions exceeds the number of samples [Umadevi and Marseline 2017]. Lastly, the k-Nearest Neighbors algorithm was included for its simplicity and efficacy in classification, leveraging the spatial distribution of data [Dogan and Tanrikulu 2013].

The optimal hyperparameters for each model were determined through this search (Table 8).

Table 8. **Summary of Algorithms and Hyper parameters in Grid Search.** We give the models and hyperparameters that we consider using for our grid search.

Algorithm	Hyperparameters
Decision Trees	- `criterion`: ['gini', 'entropy'] - `max_depth`: [None, 10, 20, 30] - `min_samples_split`: [2, 5, 10] - `min_samples_leaf`: [1, 2, 4]
Random Forest	- `n_estimators`: [50, 100, 200] - `max_depth`: [None, 10, 20, 30] - `min_samples_split`: [2, 5, 10] - `min_samples_leaf`: [1, 2, 4]
XGBoost	- `n_estimators`: [50, 100, 200] - `max_depth`: [3, 4, 5] - `learning_rate`: [0.01, 0.1, 0.2]
Logistic Regression	- `penalty`: ['l1'] - `C`: [0.01, 0.1, 1.0, 10.0] - `solver`: ['liblinear']
Support Vector Machines	- `C`: [0.01, 0.1, 1.0, 10.0] - `kernel`: ['linear', 'rbf']
k-Nearest Neighbors	- `n_neighbors`: [3, 5, 7, 9] - `weights`: ['uniform', 'distance']

When we performed our Grid Search, we optimized each model with the hyperparameters that yielded the highest cross-validation accuracy. We then compared each model to our baseline model, the optimized Logistic Regression model. For each chosen model, we calculated the difference in mean cross-validation accuracy between that model and our baseline. If the difference was positive, indicating a higher mean cross-validation accuracy than our baseline, we constructed a 95 percent confidence interval for this difference to assess statistical significance. If the confidence interval did not contain 0, we concluded that the model significantly outperformed our baseline. If more than one model significantly outperformed the baseline, we would choose the one with the highest mean cross-validation accuracy. However, none of our models significantly outperformed Logistic Regression, so we opted to use this as our classifier (Table 10).

Table 9. **Accuracy Score for Classification Model in Grid Search.** None of our candidate models significantly outperformed Logistic Regression for our down-sampled dataset. Therefore, we proceeded by using a Logistic Regression model with an L1 penalty term.

Model	Best Mean Accuracy, 95% CI for Difference, Best Parameters, and Significance
DecisionTreeClassifier	0.6896, [-0.0839, -0.0243], {criterion: 'entropy', max_depth: 10, min_split: 10, min_leaf: 1}, No
RandomForestClassifier	0.7437, [-0.0381, 0.0382], {n_estimators: 200, max_depth: 10, min_split: 10, min_leaf: 2}, No
XGBClassifier	0.7358, [-0.0580, 0.0422], {n_estimators: 100, max_depth: 3, learning_rate: 0.1}, No
SVC	0.7485, [-0.0378, 0.0346], {C: 10, kernel: 'linear'}, No
KNeighborsClassifier	0.7087, [-0.0690, -0.0138], {n_neighbors: 9, weights: 'distance'}, No
Logistic Regression	**0.7501 {"penalty": "l1", "C": 10.0, "solver": "liblinear"}**

3.5.2 Model Selection for Up-Sampled Data Set. For our up-sampled dataset, we applied the same pipeline to determine the optimal model. We found that most of our candidate models did indeed significantly outperform Logistic Regression for this dataset, although our performance was comparatively worse across the board than with the previous dataset. That being said, we identified an Extreme Gradient Boosting (XGBoost) Classifier model as the most optimal for this dataset (Figure 10).

Table 10. **Accuracy Score for Classification Model in Grid Search with Synthetic Data.**Most of our candidate models significantly outperformed Logistic Regression for our up-sampled dataset. Consequently, we proceeded by using an Extreme Gradient Boosting (XGBoost) Classifier model.

Model	Best Mean Accuracy, 95% CI for Difference, Best Parameters, and Significance
DecisionTreeClassifier	0.7447, [0.0627, 0.0943], {criterion: 'entropy', max_depth: 10, min_samples_split: 10, min_samples_leaf: 4}, Yes
RandomForestClassifier	0.7704, [0.0896, 0.1188], {n_estimators: 100, max_depth: 10, min_samples_split: 2, min_samples_leaf: 4}, Yes
XGBClassifier	**0.7762, [0.0971, 0.1229]**, {n_estimators: 200, max_depth: 4, learning_rate: 0.1}, **Yes**
SVC	0.7442, [0.0615, 0.0875], {C: 10.0, kernel: 'rbf'}, Yes
KNeighborsClassifier	0.7180, [0.0358, 0.0608], {n_neighbors: 9, weights: 'uniform'}, Yes
Logistic Regression	0.6697 {"penalty": "l1", "C": 10.0, "solver": "liblinear"}

3.6 Phase 5: Model evaluation

We evaluated both our Logistic Regression down-sampled trained model and XG Boost Classifier up-sampled model on the same test set. However, for each evaluation, we prepared the test set with the columns selected in both the up-sampled and down-sampled processes. Since the test set was not balanced across the response variable, for each model, we chose the probability threshold that resulted in the best geometric mean of sensitivity and specificity.

Sensitivity measures a model's ability to correctly identify positive cases out of all actual positives, while specificity gauges its ability to correctly identify negatives out of all actual negatives. In imbalanced datasets, where one class is more prevalent, optimizing

for sensitivity alone can lead to a high false positive rate. Optimizing for the best geometric mean of sensitivity and specificity helps strike a balance between correctly identifying both positive and negative instances, making the model more robust in real-world applications, particularly when consequences for false positives and false negatives vary.

Choosing the optimal probability threshold in this manner ensures a well-rounded model performance that considers both classes.

We found that the optimal probability threshold for our up-sampled data set was 0.25, meaning that we predicted all points with a predicted probability of 0.25 or higher of making the MLB as making the MLB. Conversely, we found that the optimal probability threshold for our down-sampled data set was 0.50. This resulted in much better performance. Our down-sampled model had a geometric mean of 0.81 and correctly predicted 83.51% of those (in our test set) who made it to the MLB and 79% who did not make it. On the other hand, our up-sampled model had a geometric mean of 0.74 and correctly predicted 92 percent of those who made it to the MLB and 60 percent of those who did not make it to the MLB (see Table 11).

Table 11. **Model Performance Metrics across our two models.** Our final model was chosen as the one trained on the down-sampled data as since it had a higher geometric mean of specificity and sensitivity it did a better job of balancing its predictive accuracy on those who made the MLB and those who did not make the MLB.

Model	Classifier	Threshold	Geometric Mean	Specificity	Sensitivity
Model 1	Threshold 0.25	0.25	0.74	0.60	0.92
Model 2	Threshold 0.50	0.50	0.81	0.79	0.84

Thus, our final model was a Logistic Regression Model that utilized a Lasso Penalty, meaning it could set some coefficients to zero, thereby performing variable selection with a lambda value of $\frac{1}{10}$. This small regularizer value implies that the regularization is not very strong but still present, allowing for some feature selection without overly penalizing the size of the coefficients.

With the exception of our Div1_1 coefficient, all of the coefficients were fairly intuitive: college_WOBA, SOS, college_Arizona, college_Clemson, college_Florida, college_Louisiana State were associated with an increase in the predicted probability of making the MLB, while Div1_1 and age were associated with a decrease (see Table 12).

It makes sense that players with higher values of Weighted On-Base Percentage, Strength of Schedule Metrics, or players at highly ranked schools such as Clemson, Florida, or Louisiana State were positively associated with higher chances of making the MLB. It also makes sense that our model prefers a player to be younger when they first started professional baseball, as this further exhibits the youth and potential of a player.

What is confusing is that playing Division 1 baseball is a negative component of our model. Division 1 is the highest and generally most competitive level of college baseball. Many more players make the MLB from Division 1 than any other division. In our training dataset, more than half of the Division 1 players made the MLB, while only around 12 percent of those who did not play Division 1 made the MLB (see Table 13). This discrepancy makes it even more

Table 12. **Coefficients for Each Variable in Chosen Logistic Regression Model.** College_WOBA, SOS, college_Arizona, college_Clemson, college_Florida, college_Louisiana State, are associated with an increase in the predicted probability of making the MLB while Div1_1 and age were associated with a decrease

Variable	Coefficient
college_WOBA	0.8315
age	-0.8603
SOS	1.1888
college_Arizona	0.8337
college_Clemson	3.6839
college_Florida	2.7408
college_Louisiana State	4.2265
Div1_1	-6.5816

intuitive why our model derived a negative coefficient for Division 1. Nevertheless, after performing an ablation test for our model, we found that all coefficients except for Div1_1 were associated with a decrease in predictive performance with the exception of SOS (see Table 14).

Table 13. **Value counts by NCAA Division and whether the players make it the MLB.** While around 56 percent of Division 1 Players made the MLB in our training set, only around 12 percent of non-division 1 players made the MLB.

Division	Made MLB	Did Not Make MLB
Division 1	303	239
Not Division 1	11	75

Table 14. **Ablation Test Results.** All Variables in Our Best Model except for Div1_1 result in a decrease in predictive performance when removing them from our model.

Variable	Original Geo Mean	New Geomean	Geomean Change
college_WOBA	0.811	0.786	0.025
age	0.811	0.753	0.058
SOS	0.811	0.804	0.007
college_Arizona	0.811	0.805	0.007
college_Clemson	0.811	0.810	0.002
college_Florida	0.811	0.810	0.001
college_Louisiana S	0.811	0.802	0.009
Div1_1	0.811	0.818	-0.007

Even though the Division 1 variable should be a useful feature, we opted to remove it from our model based on the results of the ablation test, along with its intuitive coefficient. Furthermore, the way we defined Strength of Schedule (SOS) partly encapsulates SOS into its definition: we assigned all players whose college was not Division 1 an SOS of 0. This SOS definition may explain why the Division 1 coefficient may not have been behaving as expected. Nevertheless, our final model after the ablation test had a geometric mean of 0.82, correctly predicting 85.57% of those who made it to the MLB and 78.22% of those who did not make it to the MLB (see Table 15). Note, our final model had approximately the same coefficient as the model before removing Div1_1 (Table 16) with the exception of SOS which had a reduced positive effect likely due to the fact that non division 1 schools were encoded as 0s.

Table 15. **Performance Metrics of the Final Logistic Regression Model after Ablation Test.** Our final model after the Ablation test had a geometric mean of .82, correctly predicted 85.57% of those who made it to the MLB and 78.22% of those who did not make it to the MLB.

Final Model	Geometric Mean	Specificity	Sensitivity
Logistic Regression	0.82	78.22%	85.57%

Table 16. **Coefficients for Each Variable in Final Chosen Logistic Regression Model after Ablation test.** College_WOBA, SOS, college_Arizona, college_Clemson, college_Florida, college_Louisiana State, are associated with an increase in the predicted probability of making the MLB while age was associated with increase. SOS was smaller than previous model.

Variable	Coefficient
college_WOBA	0.8261
age	-0.8901
SOS	0.3711
college_Arizona	1.0902
college_Clemson	4.1094
college_Florida	3.4923
college_Louisiana State	4.1453

3.7 Resources

All analysis is done using Python v. 3.12.0 [Van Rossum 2009]. The Scikit-learn v.1.3.2 library was use for all phases of CRISP-DM [Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. et al. 2011]. Data augmentation was done using the Synthetic Data Vault v. 1.0 library [Patki et al. 2016]. All code was run on the cloud computing services Google Colab.

4 RESULTS

4.1 Final Model performance

Our final model after the ablation test had a geometric mean of 0.82, correctly predicting 85.57% of those who made it to the MLB and 78.22% of those who did not make it to the MLB (see Table 15). Considering the size of the training set used for this model (n=628), this is extremely impressive. Nevertheless, we can now analyze the distribution of points we predicted incorrectly.

4.2 False prediction

We then analyzed the distribution of our features across the data points that we misclassified (Figure 5). Our graphs depict distributions across each feature against their predicted probability. Players who made the MLB (and were incorrectly predicted not to) are labeled in orange, while players who did not make the MLB (and were incorrectly predicted to make the MLB) are labeled in blue. We can observe that, generally, we incorrectly predicted some players with higher than average college WOBA to make the MLB, while conversely, we incorrectly predicted some players with lower than average college WOBA not to make the MLB. This observation aligns with intuition and may reflect players who overperformed or underperformed in professional baseball compared to college.

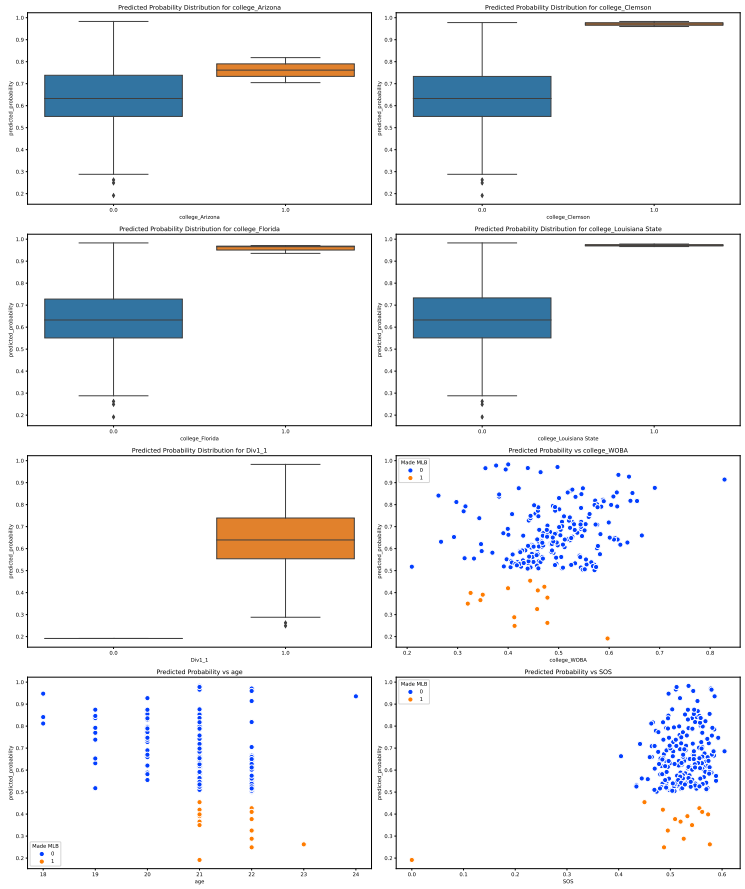

Fig. 5. **Distribution of predicted probability across predictor variables of misclassified observations (blue: not make it to MLB, orange: make it to MLB).** From up to down, and left to right, we consider college_Arizona, college_Clemson, college_Florida, college_Louisiana State, Div1_1, college_WOBA, age, SOS. The model incorrectly predicted some players with higher than average college W_OBA to make the MLB, while conversely incorrectly predicted some players with lower than average college W_OBA to not make the MLB.

4.3 Evaluation of synthetic college baseball data

We use the CTGANSynthesizer to train on our original dataset. After sampling, we evaluate and visualize the synthetic data against the real data. Using the diagnostics feature in the Synthetic Data Vault library, we found that the synthetic data covers over 90% of the categories present in the real data, and it adheres to over 90% of the min/max boundaries set by the real data. The synthetic numerical values distribution closely follows the distribution of the real numerical data (see Figure 4). However, there are more data points in the synthetic dataset that would be outliers compared to the interquartile range of the real data.

5 DISCUSSION

5.1 Data augmentation

From our experiment, the synthetic data generated from CTGAN did not improve the accuracy score in the predictive performance of our model. However, when we consider other metrics, we observe a significant difference. This disparity may be attributed to the fact that, despite upsampling the minority class (those who made it to MLB), we are still training on a relatively small dataset with 6,228 observations.

5.2 Limitations

Consideration of variables beyond sabermetrics, such as notes from baseball scouts or physicality, is crucial. Integrating qualitative insights from scouts and assessing physical attributes can provide a more comprehensive and nuanced understanding of a player's potential. This holistic approach, combining quantitative metrics with qualitative observations, enhances the robustness of our model and contributes to a more well-rounded assessment of baseball talent. This has been done using text mining techniques on scouting reports. However, scouting reports are not widely available for all college players and incorporating this information would likely be infeasible on our model. However, as more batted ball information becomes available at the college level that display the physical attributes of a players performance such as exit velocity and launch angle, utilizing these statistics in future may be helpful.

5.3 Future works

Given that we will have new data on the MLB draft every year, we want our model to be updated as the information comes in. Our model could be updated by adding one more year of data as a year of MLB progresses. For example, at the end of this season we update our model by including college players drafted in 2020. Furthermore, we aim to explore additional methods of augmenting synthetic data for baseball-related prediction tasks, considering that there may be other techniques that could prove useful in enhancing the predictive performance of our classification model. Additionally, we seek to utilize synthetic data to increase the overall number of observations in our training dataset. It may be useful to create synthetic data for those who did not make the MLB as well and this may negate the decrease in performace we had in predicting this data set on our synthetic model.

6 CONCLUSION

Using the college baseball dataset we scraped, we set out to build a classification method for predicting MLB scouting. We attempted to remedy the class imbalance in the dataset using two methods: by down-sampling the majority to the minority using K-Means sampling and by up-sampling the minority to the majority using CTGAN synthetic data. Overall, the best-performing model based on accuracy is the Logistic Regression with an L1 penalty term on the down-sampled dataset. This best model was a logistic regression model using the down-sampled data, with an accuracy of 0.7501, a sensitivity of .8557, and a specificity of .7822.

DATA AND CODE AVAILABILITY

All data used in this paper and the code for the analysis can be accessed through Github.

ACKNOWLEDGMENTS

We thank Dr. Natalia Khuri for her invaluable feedback and comments. We used Chat GPT to check the spelling of our paragraphs.

REFERENCES

Chunsheng An, Jingtong Sun, Yifeng Wang, and Qingjie Wei. 2021. A K-means Improved CTGAN Oversampling Method for Data Imbalance Problem. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*. 883–887. https://doi.org/10.1109/QRS54544.2021.00097 ISSN: 2693-9177.

A. H. Aziira, N. A. Setiawan, and I. Soesanti. 2020. Generation of Synthetic Continuous Numerical Data Using Generative Adversarial Networks. *Journal of Physics: Conference Series* 1577, 1 (July 2020), 012027. https://doi.org/10.1088/1742-6596/1577/1/012027 Publisher: IOP Publishing.

Jim Albert Baumer, Benjamin S. 2018. *Analyzing Baseball Data with R, Second Edition* (2 ed.). Chapman and Hall/CRC, New York. https://doi.org/10.1201/9781351107099

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

Jacob Danovitch. 2019. Trouble with the Curve: Predicting Future MLB Players Using Scouting Reports. Pittsburgh, PA, USA. https://www.stat.cmu.edu/cmsac/conference/2019/assets/pdf/TWTC.pdf

Neslihan Dogan and Zuhal Tanrikulu. 2013. A comparative analysis of classification algorithms in data mining for accuracy, speed and robustness. *Information Technology and Management* 14, 2 (June 2013), 105–124. https://doi.org/10.1007/s10799-012-0135-8

Huajuan Duan, Yongqing Wei, Peiyu Liu, and Hongxia Yin. 2020. A Novel Ensemble Framework Based on K-Means and Resampling for Imbalanced Data. *Applied Sciences* 10, 5 (Jan. 2020), 1684. https://doi.org/10.3390/app10051684 Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

Cork Gaines. 2013. Most Baseball Draft Picks Will Still Be In The Minors Four Years From Now. https://www.businessinsider.com/chart-how-long-it-takes-a-draft-pick-to-reach-major-league-baseball-2013-6

Eric A. E. Gerber and Bruce A. Craig. 2021. A mixed effects multinomial logistic-normal model for forecasting baseball performance. *Journal of Quantitative Analysis in Sports* 17, 3 (Sept. 2021), 221–239. https://doi.org/10.1515/jqas-2020-0007 Publisher: De Gruyter.

Omar Habibi, Mohammed Chemmakha, and Mohamed Lazaar. 2023. Imbalanced tabular data modelization using CTGAN and machine learning to improve IoT Botnet attacks detection. *Engineering Applications of Artificial Intelligence* 118 (Feb. 2023), 105669. https://doi.org/10.1016/j.engappai.2022.105669

Der-Chiang Li, Szu-Chou Chen, Yao-San Lin, and Kuan-Cheng Huang. 2021. A Generative Adversarial Network Structure for Learning with Small Numerical Data Sets. *Applied Sciences* 11, 22 (Jan. 2021), 10823. https://doi.org/10.3390/app112210823 Number: 22 Publisher: Multidisciplinary Digital Publishing Institute.

Chris Mitchell. 2014. KATOH: Forecasting Major League Hitting with Minor League Stats. https://tht.fangraphs.com/katoh-forecasting-a-hitters-major-league-performance-with-minor-league-stats/

Warren Nolan. [n. d.]. 2023 College Baseball Home Page | WarrenNolan.com. https://www.warrennolan.com/baseball/2023/index

Bahadorreza Ofoghi, John Zeleznikow, Clare MacMahon, and Markus Raab. 2013. Data Mining in Elite Sports: A Review and a Framework. *Measurement in Physical Education and Exercise Science* 17, 3 (July 2013), 171–186. https://doi.org/10.1080/1091367X.2013.805137 Publisher: Routledge _eprint: https://doi.org/10.1080/1091367X.2013.805137.

Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. 2016. The Synthetic Data Vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 399–410. https://doi.org/10.1109/DSAA.2016.49

Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V., and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P., and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and, and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. 2011. Scikit-learn: Machine Learning in {P}ython. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

Bill Petti and Saiem Gilani. 2021. baseballr: The SportsDataverse's R Package for Baseball Data. https://billpetti.github.io/baseballr/

Mariwan Hama Saeed and Jihad Ibrahim Hama. 2023. Cardiac disease prediction using AI algorithms with SelectKBest. *Medical & Biological Engineering & Computing* (Sept. 2023). https://doi.org/10.1007/s11517-023-02918-8

Lloyd Smith, Bret Lipscomb, and Adam Simkins. 2007. Data mining in sports: predicting Cy Young award winners: Journal of Computing Sciences in Colleges: Vol 22, No 4. *Journal of Computing Sciences in Colleges* 22, 4 (April 2007), 115–121. https://dl.acm.org/doi/abs/10.5555/1229637.1229658

Yanmin Sun, Andrew K. C. Wong, and Mohamed S. Kamel. 2009. Classification of imbalanced data: a review. *International Journal of Pattern Recognition and Artificial Intelligence* 23, 04 (June 2009), 687–719. https://doi.org/10.1142/S0218001409007326 Publisher: World Scientific Publishing Co..

Fadi Thabtah, Suhel Hammoud, Firuz Kamalov, and Amanda Gonsalves. 2020. Data imbalance in classification: Experimental evaluation. *Information Sciences* 513 (March 2020), 429–441. https://doi.org/10.1016/j.ins.2019.11.004

S. Umadevi and K. S. Jeen Marseline. 2017. A survey on data mining classification algorithms. In *2017 International Conference on Signal Processing and Communication (ICSPC)*. 264–268. https://doi.org/10.1109/CSPC.2017.8305851

Fred L. Van Rossum, Guido and Drake. 2009. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling Tabular data using Conditional GAN. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc. https://papers.nips.cc/paper_files/paper/2019/hash/254ed7d2de3b23ab10936522dd547b78-Abstract.html

Md. Sabab Zulfiker, Nasrin Kabir, Al Amin Biswas, Tahmina Nazneen, and Mohammad Shorif Uddin. 2021. An in-depth analysis of machine learning approaches to predict depression. *Current Research in Behavioral Sciences* 2 (Nov. 2021), 100044. https://doi.org/10.1016/j.crbeha.2021.100044

Received 8 December 2023