Final Project:
Predicting Strikes from MLB Pitching Data

Miguel A. Sanchez Cortes,* Gabriele Volzone,† Sofia Crobeddu,‡
Pasquale Luca Tomassino,§ and Laura Lopez Sanchez¶
Master in Data Science, Sapienza University of Rome
Fundamentals of Data Science
(Dated: December 27, 2023)

Abstract: In this project we studied the features that affect an umpire's decision-making process when predicting a strike using data from pitches of the MLB 2022 season. We then used these features to build Logistic Regression and Neural Network classifiers in order to explore their performances when trying to predict strike calls. We found that geographical features (pitch position, zone, distance from center, etc.) are important features for umpires to decide whether a pitch is a strike or not. In the same way, we found that both of our classifiers perform similarly when using this feature to try and predict strikes with an AUC value near 0.98 indicating good predictive performance.

I. INTRODUCTION

Since the popularization of Data Analytics in baseball, the sport has undergone a transformative evolution, entering in a new era characterized by a data-driven approach to player evaluation, game strategy, and overall performance optimization. This revolution, often referred to as the *Moneyball* era [1], was ignited by the pioneering work of individuals such as Billy Beane and Theo Epstein, who leveraged advanced statistical analysis to uncover undervalued players and exploit inefficiencies in the traditional scouting and recruitment processes.

Despite the proliferation of Data Analytics and advancements in technology, baseball has not evolved in all fronts, the determination of balls and strikes[2] still heavily relies on the subjective judgment of human umpires. This inherent subjectivity has led to ongoing debates about the consistency and accuracy of strike zone calls, since predicting strikes is a complex task influenced by various factors such as pitch movement, batter stance, and the umpire's individual interpretation. Efforts to modernize this aspect of the game have given rise to discussions[3–5] surrounding the implementation of pitch tracking systems, cameras, and artificial intelligence in order to provide a more objective and standardized strike zone.

In this project we intend to explore some of the features that affect an umpire's decision-making process when predicting a strike and using them to build and compare the performances of different Machine Learning classifiers. The objective of doing this is to study how this technology can be leveraged to enhance the accuracy and consistency of strike calls in baseball.

II. METHODS

For analysis in this project and to build our Machine Learning classifiers we used pitching data from the 2022 MLB season [6] containing approximately 300k non-null entries (pitches) along with 25 different features that characterize each one of the entries (see I and Figure 1). This dataset is imbalanced, with rougly 66% of entries being labeled as *not strike* and the other 33% as strike.

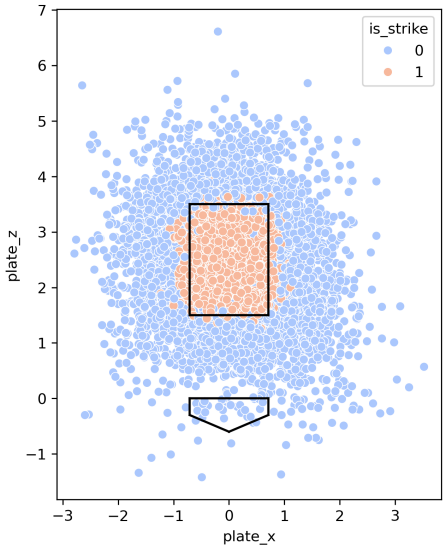

FIG. 1. Pitch Distribution of the Dataset. In this Figure we can observe the labels of a sample of pitches from our dataset given their x and z coordinates above the plate.

A. Data Pipeline

As we can observe in I this dataset includes several non-numeric features that needed to be modified in order to be fed to Machine Learning classifiers. Therefore,

---

* Contributed equally to all parts of this project
† Contribution: Feature Engineering and Selection
‡ Contribution: Logistic Regression Model
§ Contribution: Hyperparameter Tuning
¶ Contribution: Model Performance Analysis

we performed One-Hot Encoding to categorical features to ensure all our features are numeric along with data standardization to ensure all features contribute equally to the training of our models. At the same time, we divided our dataset into train, test and validation sets following a proportion of 60:20:20. We used the training set to train our classification models, the validation set to tune the hyperparameters of these models and the test set to analyze the performance of these models (see Figure 2 for more details).

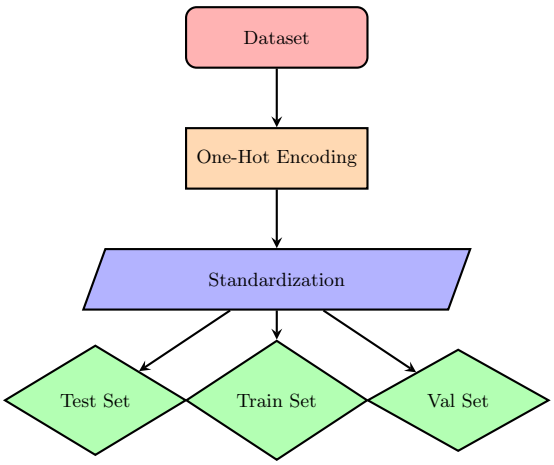

FIG. 2. Data Pipeline. We performed One-Hot Encoding to categorical data, standardization to numerical data and finally divided our dataset into train, test, and validation sets with a proportion of 60-20-20.

B. Feature Engineering & Selection

Since the dimensionality of our dataset is relatively big, we also performed *dimensionality reduction* when training our Machine Learning classifiers. To do this, we first performed Feature Engineering in order to create new, and informative features from our existing data so that later we could potentially select the most important features to reduce our dataset dimensionality without losing information. In Appendix B we can observe the definitions of our 6 newly-constructed engineered features.

Once we performed Feature Engineering along with One-Hot Encoding and Data Standardization we ended up with 54 total features in our dataset. Since this could be potentially catastrophic for training since it could lead to overfitting and high training times for our classification models, we performed Feature Selection by Random Forest Importance [7] in order to obtain the **top-10** most important features that best characterize baseball pitches.

A Random Forest model inherently provides a built-in feature ranking mechanism. During the training process, the algorithm assesses the importance of each feature by measuring the decrease in impurity (e.g., Gini impurity) that it causes when used for splitting nodes in the trees. Features that consistently lead to more significant reductions in impurity are deemed more important. This implicit ranking allows us to easily identify and select the most informative features for our model.

In Figure 5 we can see the Top-10 most important features according to a Random Forest Classifier. As we can see, our Feature Engineering process successfully captured information relevant to the Decision Process an umpire carries on when deciding if a pitch is a strike or not since our newly-defined Attack Zones along with the Distance of a Pitch from the center are within the most important features according to the Random Forest model.

C. Classifiers Construction & Training

With our dataset already separated into train-test-validation sets and with our newly selected most-relevant features, we decided to build three different classification models in order to compare their performances:

1. **Benchmark Model (Random Classifier):** A Random Classifier (also known as a *null model*) is a simple Machine Learning model that provides basic predictions based on the class distributions or simple rules in our dataset. Given that our dataset is **imbalanced** we chose to build a *Most Frequent* Random Classifier. The precision of this classifier will always be equivalent to the proportion of the most represented class in our dataset. If more complex classification models don't perform better than this benchmark model, this could mean that the classifier might be learning to predict the majority class most of the time, achieving high accuracy but failing to capture the minority class.

2. **Logistic Regression Model:** Logistic Regression is a linear model commonly used for binary classification problems. This model assumes a linear relationship between the input features and the log-odds of the predicted class. It utilizes the logistic function to convert the output into the range $[0, 1]$, representing the probability of belonging to the positive class. This model is particularly useful for its interpretability and simplicity. However, its effectiveness depends on the linearity assumption, and it may struggle with capturing complex relationships in more complex data.

3. **Feed-Forward Neural Network Classifier:** A Neural Network Classifier is a more sophisticated model that consists of interconnected layers of neurons, enabling them to capture complex non-linear

dependencies. The architecture typically includes an input layer, one or more hidden layers, and an output layer. Training involves optimizing the network's parameters using techniques like backpropagation and gradient descent. These models often outperform simpler models when dealing with complex data, making them a valuable candidate for comparison against benchmark and logistic regression models.

In order to ensure optimal results with our classifiers we also performed **Hyperparameter Tuning**. Ideally, we would want a model that minimizes both False Negatives and False Positives. We therefore considered to maximize the F1 Score (the harmonic mean between precision and recall):

$$\text{F1} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{1}$$

The F1 Score reflects a **trade-off** between precision and recall. It is higher when both precision and recall are high and balanced. We then chose to perform Hyperparamter Tuning via **Grid Search** for our classification models by maximizing the F1 Score in order to obtain the optimal combination of hyperparameters that yields the best balance between precision and recall (see Appendix D for the best hyperparameters and 6 for the Neural Network Architecture).

III. RESULTS

A. Most Important Features

As a first result from the feature selection process we can observe from 5 the following things:

- Intuitively we could expect that the position at which the ball crosses the plate after a pitch is relevant in order to classify a pitch into strike, since it always has to cross at the defined Strike-Zone. This is reflected in the *distance from center*, *plate z*, *plate x* and *Attack Zones* features since they are within the top-10 most relevant features according to the Random Forest model. Thus making us conclude that geographical features are the most important features in the decision making process of an umpire since its main goal during the game is assessing that a ball effectively crosses the Strike-Zone.

- We can get another interesting insight: the previous number of strikes in the count of a batter **is** influential on the decision-making process of an umpire, which suggests that an umpire can be biased depending on how high are the stakes on a given pitch. This can be thought in an equivalent to football as referees that are more likely to give a penalty in the last minutes of a very tense game.

B. Machine Learning Classifiers Performance

Since we have an imbalanced dataset, we knew a-priori that *accuracy* is not a good metric to evaluate the performance obtained by our constructed classifiers since a simple Random Classifier could obtain 66% accuracy when classifying. However if we observe the obtained Receiver Operating Characteristic (ROC) curve:

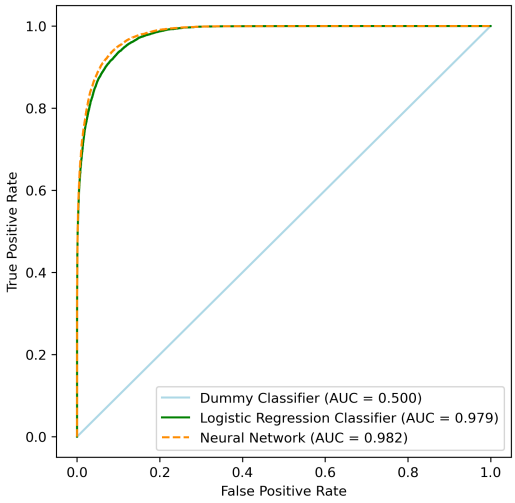

FIG. 3. ROC Curve. Receiver Operating Characteristic curves depicting the discrimination performance of three classifiers: Random ($AUC = 0.5$), Logistic Regression ($AUC = 0.97$), and Neural Network ($AUC = 0.98$). Higher AUC values indicate superior classification ability, with the Neural Network and Logistic Regression models demonstrating impressive predictive accuracy.

We can see the AUC values for both our classifiers are a significant improvement from our previously defined Random Classifier. Nevertheless these values are very similar, diferring only by 0.003, meaning that the ability of both classifiers to discriminate between positive and negative instances is nearly identical.

With AUC values near 0.98 we can say that our classifiers demonstrate a strong ability to distinguish between positive and negative instances in the dataset. This substantial performance improvement over the Random Classifier emphasizes the value of our feature selection and model training processes in creating models that can effectively differentiate between positive and negative instances.

C. Attack Zones

It is interesting to notice that we can also **recreate** our previously-defined Attack Zones by obtaining the **strike probabilities** given by each of our Classification models. Since by definition each one of these zones marks a different probability of a pitch to be called a strike, we

would expect that our classifiers classifies pitches in each zone accordingly.

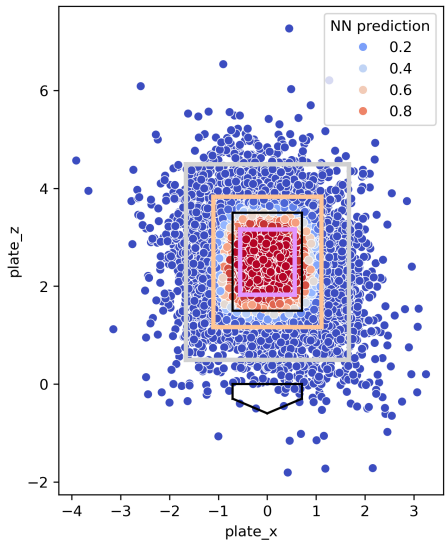

FIG. 4. Strike Probability by Attack Zone. As we can observe, pitches within the Heart Zone (pink) have a very high probability of being called strikes whereas the Shadow Zone (orange), Chase Zone (gray) and Waste Zone (outside the gray area) have decreasing probabilities from 0.6 to 0.2.

From Figure 4 can observe that effectively our classifier associates a higher probability to pitches within the **Heart** zone whereas the probability for each one of the other zones decays from 0.6 to 0.2 depending on the geographic area. Moreover, these probabilities are similar to the ones defined by the Statcast MLB tool [8]:

- **Heart Zone:** Area over the plate where the probability of a pitch resulting in strike is roughly 0.73. In oue case it is near 0.8.
- **Shadow Zone:** Area over the plate where the probability of a pitch resulting in strike is roughly 0.53. In our case it decays from 0.6 to 0.4 as pitches are close to the border.
- **Chase and Waste Zone:** Area over the plate where the probability of a pitch resulting in strike is roughly 0.23 and 0.06 respectively. In our case it is near 0.2 for both zones.

From this result we can conclude that our classifiers successfully capture and align with the predefined strike probabilities in each Attack Zone, demonstrating its ability to effectively predict pitch outcomes across different areas around the strike zone.

IV. CONCLUSIONS & FUTURE WORK

In general, from this project we can conclude the following:

- By performing Feature Engineering we were able to capture and get insights regarding the Decision Process an umpire carries on when deciding if a pitch is a strike or not. As we can observe in our results obtained by Feature Selection, geographical features are the most important features when trying to classify pitches. This was expected since primarily an umpire classifies strikes by observing the spatial coordinates of a pitch when it crosses the plate.

  However, we also observed that other unexpected features play an important role, like the number of pitches before the strike (or not strike) suggesting that an umpire may be influenced by external pressure when making a decision in real-time.

- Our Machine Learning classifiers effectively obtained a good performance when predicting pitch outcomes by taking into account geographical features achieving an AUC value of 0.98 very near to becoming **perfect classifiers**. This could mean that the Machine Learning paradigm can be effectively introduced to predict pitching outcomes in order to achieve better and more robust strike calls.

As future work we could perform a more general exploration of other Machine Learning models in order to compare their performances as well as exploring which features are most important for each one of these classifiers to see if they all share the same importance throughout all models.

Appendix A: Dataset Features

TABLE I. Features included in the 2022 MLB Pitch dataset. Among these features there are 16 numerical features, 6 categorical features and 3 boolean features. The last not included feature is the label: 0 for not strike and 1 for strike.

Name	Type	Definition
Pitch Type	Categorical	Pitch type codes.
Release Speed	Numerical	Pitch velocity as ball comes out of hand.
Release Pos. X	Numerical	Horizontal Release Position of the ball measured in feet from the catcher's perspective.
Release Pos. Z	Numerical	Vertical Release Position of the ball measured in feet from the catcher's perspective.
Release Pos. Y	Numerical	Release Position of pitch measured in feet from the catcher's perspective.
Spin Rate	Numerical	Spin rate of the pitch.
Spin Axis	Numerical	Spin Axis in the two-dimensional x-z plane in degrees from 0 to 360.
Pfx	Numerical	Horizontal movement in feet from the catcher's perspective.
Pfz	Numerical	Vertical movement in feet from the catcher's perpsective.
Plate X	Numerical	Horizontal position of the ball when it crosses home plate from the catcher's perspective.
Plate Z	Numerical	Vertical position of the ball when it crosses home plate from the catcher's perspective.
SZ Top	Numerical	Top of the batter's strike zone set by the operator when the ball is halfway to the plate.
SZ Bottom	Numerical	Bottom of the batter's strike zone set by the operator when the ball is halfway to the plate.
Stand	Categorical	Side of the plate where the batter is standing.
Pitch Throws	Categorical	Hand the pitcher throws with.
Balls	Numerical	Pre-pitch number of balls in count.
Strikes	Numerical	Pre-pitch number of strikes in count.
Outs When Up	Numerical	Pre-pitch number of outs.
Inning	Numerical	Pre-pitch inning number.
Inning TopBot	Categorical	Pre-pitch top or bottom of inning.
IF Fielding Alignment	Categorical	Infield fielding alignment at the time of the pitch.
OF Fielding Alignment	Categorical	Outfield fielding alignment at the time of the pitch.
On 1B	Boolean	Runner on first base represented as true/false.
On 2B	Boolean	Runner on second base represented as true/false.
On 3B	Boolean	Runner on third base represented as true/false.

Appendix B: Engineered Features

The features we decided to extract from our dataset were:

1. **Attack Zones:** Attack Zones [9] are a concept first introduced by Tom Tango at MLB Advanced Media [10]. The purpose of Attack Zones is to provide more context to pitches thrown in and around the Strike-Zone. There exist 4 Attack Zones:

   - **Heart Zone:** The area where all pitches should be called strikes.
   - **Shadow Zone:** It is characterized as the area where pitches are called strikes 50% of the time.
   - **Chase Zone:** The Chase Zone covers the area that is definitively outside of the true Strike-Zone and the Shadow Zone.
   - **Waste Zone:** The Waste Zone covers the area where pitches should never be called strikes (unless a batter is swinging at a pitch in the Waste Zone).

   This feature adds context to pitch location and the Strike-Zone by subdividing pitch location into four groups. This also characterizes pitches by their inherent probability of being called strikes depending on their position.

2. **Distance from Center:** We define a pitches *distance from center* as the Euclidean distance between the center of the Strike-Zone and the coordinates for each pitch. Let's remember that the euclidean distance between two points A_1 and A_2 in this case is defined as:

$$d(A_1, A_2) = \sqrt{(x_2 - x_1)^2 + (z_2 - z_1)^2}, \tag{B1}$$

   where (x_1, z_1) and (x_2, z_2) are the coordinates of points A_1 and A_2 respectively. From this feature we can expect that pitches with a high distance from the center are unlikely to be called strikes and viceversa.

3. **Strike Zone Length:** This feature captures the full length of the Strike-Zone for a given pitch (and batter). This length is simply obtained as:

$$l = |sz_{top} - sz_{bottom}|, \tag{B2}$$

where sz_{top} is the distance above the home plate of the top part of the Strike-Zone and sz_{bottom} is the distance above the home plate of the bottom part of the Strike-Zone.

4. **Total Pitch Movement:** We define this feature as the Euclidean distance between the movement coordinates of a pitch from the catchers perspective and the origin point $(0, 0)$. This feature should capture how far away is a pitch with respect from the center from where the pitcher threw the ball.

5. **Movement Above/Below Average:** This feature takes advantage of the last calculated feature and obtains the average total movement for each pitch type obtains the total movement above/below that average for each pitch. In this way we can quantify **how much** a given pitch moved above/below average depending on its pitch type.

6. **Pitcher-Hitter Alignment:** This feature combines the pitcher's throwing hand and the batter standing position to create new categories depending on the alignment of both pairs(left handed-left stand, left handed-right stand, right handed-left stand, right handed-right stand). This feature can capture how relevant the standing and the throwing hand of a pitcher is for determining a pitch as a strike.

Appendix C: Most Important Features

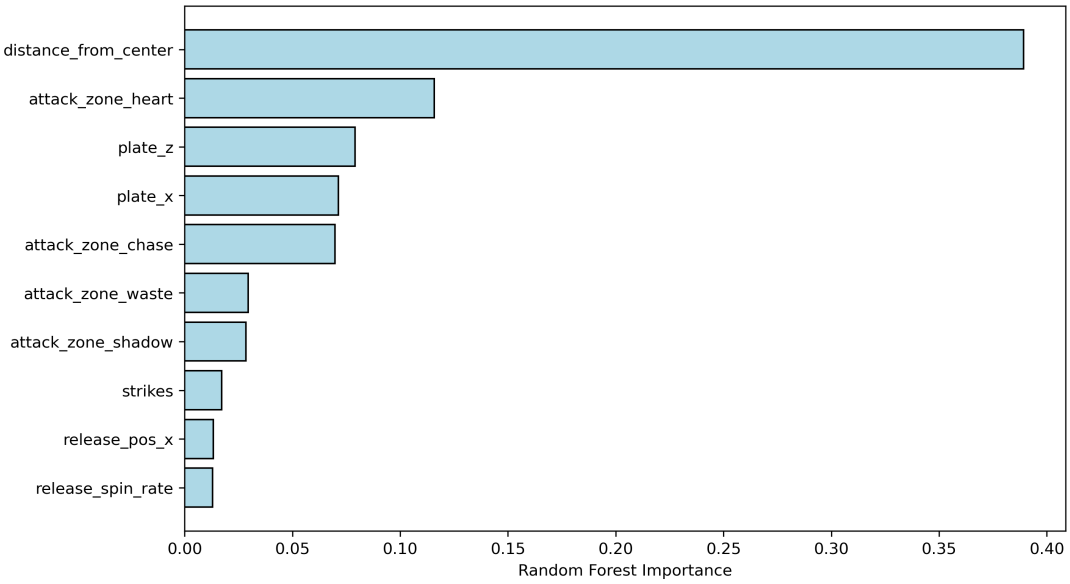

FIG. 5. Top-10 Most Important Features according to a Random Forest Classifier. We can notice that almost all of these features are geographical features that always contain the coordinates of a given pitch regarding to the plate.

Appendix D: Optimal Hyperparameters

- **Logistic Regression Model:** The main hyperparameters we can explore and tune in Logistic Regression for the *sklearn package*[11]are:
  - **solver:** Algorithm to use in the optimization problem. This are, the algorithms that minimize the log-likelihood function. Using grid search its optimal value found was: Saga [12].

- **penalty:** Specifies the norm of the penalty. This parameter is used to control the regularization of the model. Regularization is a technique used to prevent overfitting by adding a penalty term to the cost function. The purpose of regularization is to reduce the complexity of the model, which can lead to better generalization performance on unseen data. The most common types of regularization are L1 and L2. Using grid search its optimal value found was: L1 [13].
- **C (Inverse Regularization Strength):** This parameter must be a positive float. Since we're performing model regularization, the regularization term normallu penalizes large coefficients. Since the size of each coefficient in the loss function depends on the scale of its corresponding variable, scaling the data is required so that the regularization penalizes each variable equally. The regularization strength is determined by C and as C increases, the regularization term becomes smaller (and for extremely large C values, it's as if there is no regularization at all). Using grid search its optimal value found was: $C = 0.01$.

- **Neural Network Classifier:**

  To build our Neural Network Classifier we selected as an activation function the ReLU function along with the Cross Entropy Loss as a loss function and the ADAM optimizer [14]. The reason behind this was mainly because these are proven parameters that work well for classification problems. The other tuned hyperparameters were:

  - **Learning Rate:** The step size at which the model adapts during training. A higher learning rate may speed up convergence but risks overshooting the optimal solution, while a lower learning rate may converge more slowly but with more stability. Using grid search its optimal value found was: $l = 0.1$.
  - **Hidden Size:** he size of the hidden layers in the neural network. This affects the capacity of the model to capture complex patterns. Too few neurons may result in underfitting, while too many may lead to overfitting. Using grid search its optimal value found was: $h = 8$.
  - **Epochs:** The number of times the entire training dataset is passed through the model during training. Too few epochs may result in underfitting, while too many may lead to overfitting. Using grid search its optimal value found was: $h = 1000$.
  - **Regularization Technique (Dropout):** The Dropout Technique is employed to prevent overfitting by adding penalties for overly complex models. Specifically, this technique involves randomly "dropping out" (i.e., setting to zero) a proportion of neurons in the hidden layers during each training iteration. In this case, the hyperparameter is the **probability** of dropping out neurons. Using grid search its optimal value found was: $h = 0.01$.

Appendix E: Neural Network Architecture

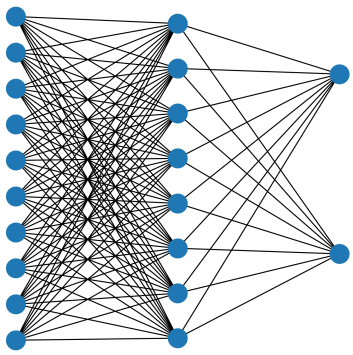

FIG. 6. Feed-Forward Neural Network Architecture. Neural Network Classifier utilized to distinguish strikes on pitching data. It is formed by one hidden layer with 8 nodes and an output layer of 2 nodes where the output represents the probability of a pitch being classified as strike (1) or not (0).

[1] R. Elitzur, Data analytics effects in major league baseball, Omega **90**, 102001 (2020).

[2] In baseball, a *ball* is a pitch that is thrown by the pitcher but is not swung at by the batter and does not pass through the strike zone. The strike zone is an imaginary area over home plate, and its boundaries are defined by the batter's knees and the midpoint between the top of the shoulders and the top of the pants. If a pitched ball crosses through this zone and the batter doesn't swing, it is called a *strike*.

[3] J. Huang and H.-J. Hsu, Approximating strike zone size and shape for baseball umpires under different conditions, International Journal of Performance Analysis in Sport **20**, 133 (2020), https://doi.org/10.1080/24748668.2020.1726156.

[4] G. Sidle and H. Tran, Using multi-class classification methods to predict baseball pitch types, Journal Name , 85 (2018).

[5] Y. Sugizaki, J. Nakazato, M. Tsukada, and H. Esaki, Umpire assistance system in baseball game (2023).

[6] `https://www.kaggle.com/competitions/nwds-xstrikes/data?select=train.csv`.

[7] M. A. M. Hasan, M. Nasser, S. Ahmad, and K. I. Molla, Feature selection for intrusion detection using random forest, Journal of information security **7**, 129 (2016).

[8] `https://baseballsavant.mlb.com/visuals/swing-take?playerId=545361`.

[9] `https://www.si.com/mlb/2019/05/11/chris-paddack-san-diego-padres-first-pitch-strikes`.

[10] We learned this notion thanks to the following Kaggle notebook: `https://www.kaggle.com/code/nickwan/attack-zones-visualizing-the-strike-zone`.

[11] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`.

[12] A. Defazio, F. Bach, and S. Lacoste-Julien, Saga: A fast incremental gradient method with support for non-strongly convex composite objectives, in *Advances in Neural Information Processing Systems*, Vol. 27, edited by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger (Curran Associates, Inc., 2014).

[13] R. Tibshirani, Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society. Series B (Methodological) **58**, 267 (1996).

[14] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization (2017), arXiv:1412.6980 [cs.LG].