Review of Predictive Artificial Intelligence Models in Major League Baseball Analytics

Deepak Pandey[1*], Rajeev Gupta[2]

[1*]M.M. Institute of Computer Technology & Business Management, Maharishi Markandeshwar (Deemed to be University), Mullana, Ambala, Haryana, India, Email- englishdepartmentiec@gmail.com

[2] Department of Computer Science and Engineering, M.M. Engineering College, Maharishi Markandeshwar (Deemed to be University), Mullana, Ambala, Haryana, India

Abstract –In the present time, MLB - Major League Baseball is a professional biggest sporting event at international level. And the researchers are very much keen about the prediction of results in this particular game. Many have tried envisaging and forecasting the outcome of these matches on the basis of trend and scenario. But the predicting outcome accuracy levels of these matches are not up to the mark, it ranges between 50 to 60 percent only. Therefore, nowadays to improve this accuracy, various artificial-intelligence based methods are being used. The purpose of this research is to analyze and compare different machine learning models used by various authors over the years. It will assess these models on dimensions like Flexibility, Complexity and Interpretability. With the help of these comparisons the researcher will also suggest how improvements can be made in these models so that higher level of accuracy can be achieved in predicting the results of the game.

Keywords—Major League Baseball, Machine Learning, Prediction Model, Artificial Intelligence

1. Introduction

Teams spend a lot of money on visual analysis to increase team effectiveness and performance for better outcomes. Moneyball, a book written by Michael Lewis in 2003, promoted the concept of data analytics techniques for further data analysis in any sports event. Moneyball demonstrates how statistical data analysis may be utilized to enhance the selection of match players, management strategy to play any game, and ultimately assist a team in winning a game in baseball [1]. Baseball is a tremendously complex sport, involving numerous variables, such as player performance, team spirit, the environment, and strategy, all of which influence how a game turns out. As a result of the effect of numerous elements, the matches of a baseball game are unpredictable, and for viewers, this unpredictable nature is what makes a baseball game so intriguing [2]. In the present time, Major League Baseball (MLB) is the biggest professional sporting event at the international level. A baseball game's outcome is influenced by a wide range of variables, making it very challenging to predict with any degree of accuracy. In a research study [3], the accuracy of the next generation of gamers was below 60 percent. Artificial intelligence encompasses a wide range of methodologies, including linguistics, bias, vision, planning; real time sports match analysis, natural language processing, decision science, etc. Yang and Swartz [2004] gathered statistics from 179 MLB games. The parameters used in the suggested 2-stage Bayesian technique comprised the ratios of the two teams' winning percentages, overall batting averages, ratios of the ERAs (Earned run average) of the two starting pitchers, and home field advantage. They correctly predicted who would win each MLB division [4]. However, the prediction's accuracy will be pretty high [11]. To predict player wins above replacement (WAR), researchers used random forest (RF), gradient-boosted trees (GBT), regression trees (RT), and linear regression (LR) [5]. Baseball lends itself to statistical analysis due to the volume of data and the discrete nature of the game, and baseball clubs across the tournament have notably adopted data-driven and statistical analysis throughout time [6]. Saber metrics refers to any statistical study that goes beyond simple descriptive statistics [7]. The Log5 technique is used to calculate the probability of a strikeout given a specific pitcher-hitter combination [8]. They used Bayesian approaches to estimate a player's batting average during the 2006 MLB season. For implementation and testing purposes, Jiang et. al. used the 2005 matches batting averages as training data. As a result, the analyst considers and prefers Bayesian approaches instead of the least-squares linear regression model [9]. MLB is made up of 30 teams from the United States and one from Canada. The season usually lasts from late March or early April until late September or early October, with postseason play taking place in October and early November. During the regular season, each team plays a total of 162 games, with a mix of home and away games against the other teams in its league. The teams are split into two leagues, the American League (AL) and the National League (NL), with each league having three divisions of five teams. The team with the best record in each division, as well as two additional "wild card" teams from each league, progress to the playoffs after the regular season.

2. Methodology

2.1 Artificial Intelligence Models in MLB

Artificial intelligence is significant in every aspect of contemporary culture. Deep Learning (DL) and Machine learning (ML), both are both subsets of artificial intelligence (AI). To create a complete AI-based system in sports, authors create machine learning or deep learning mechanisms that are written in machine language using complex mathematical expertise. Thus, machine learning allows you to do tasks such as classifying, analyzing, and estimating data from a given dataset. It has provided different sports prediction models in the last few years. It basically converges on applications that improve their decision-making ability or prediction accuracy over time. Because of its capacity to predict outcomes, artificial intelligence is currently being applied in baseball for both practical and academic reasons.
With more observations, machine learning analysis becomes more reliable and accurate. When one thinks about extremes, this is easily demonstrated. Think about guessing whether a pitcher will throw a fastball next. If there are only a few observations of the pitcher's previous pitches, estimating the next pitch will be exceedingly tough. Some important AI-based models used by researchers are discussed below:

a). GLM – Generalized Linear Models

The concepts of logistic regression and linear regression are quite similar. Linear regression is a technique used to describe the relationship between a set of "features" and a "target". It also identifies the coefficients and intercepts that get estimated for "y", the model closest to the actual target value. Equation 2.1 is used by generalized linear models.
Linear regression equation:

$$y = k + B_1x_1 + B_2x_2 + \ldots + B_nx_n \quad \ldots\ldots\ldots\ldots (2.1)$$

Here, x is the features (baseball statistics), and y is the target value (total wins), which is also known as the dependent variable. Logistic regression uses the logit function, which is the inverse of the more well-known sigmoid function, to get a discrete output from a linear combination of input variables.
The logistic regression model was employed as a binary response variable to determine whether a team made the playoffs or not at the end of the season [20].
Generally, equations 2.2 and 2.3 are used by the logistic regression model for computation.
Logistic curve:

$$p(X) = \frac{e^{B0+B1X}}{1+e^{B0+B1X}} \quad \ldots\ldots\ldots\ldots\ldots\ldots (2.2)$$

Log-odds:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = B_0 + B_1X \quad \ldots\ldots\ldots\ldots\ldots\ldots (2.3)$$

Here, p indicates the event's probability. (p(x)/ (1 – p(x))) indicates the odds ratio, B_0 indicates a constant term, and B_1 indicates the regression coefficient

b). K-Nearest Neighbor Classifiers (KNN)

The K in the K-NN nomenclature denotes the number of closely related neighbors that the algorithm will consider while making a classification and regression decision. For high-dimensional datasets, K-NN is computationally costly; it is required to calculate distances for every observation and lacks a simple objective function to optimize, unlike linear and logistic models [21]. (See Figure 2.1)

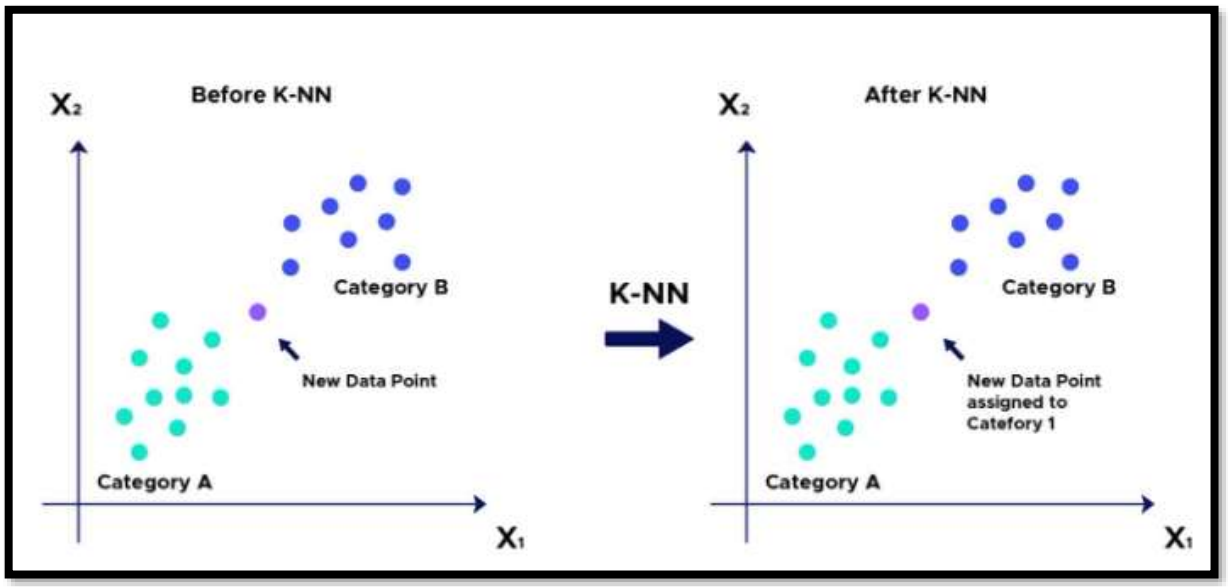

Figure 2.1 K-Nearest Neighbor Classifiers (KNN)

[Source: datascientest.com]
The Euclidean Distance is the most often used metric for assessing "closeness" between instances.
Assume $J_1 = (J_{1_1}, J_{1_2}, \ldots, J_{1_n})$ and $J_2 = (J_{2_1}, J_{2_2}, \ldots, J_{2_n})$ be two cases.

Equation 3.4 is used to calculate the Euclidean Distance (ED).

$$ED(J_1, J_2) = \sqrt{\sum_{j=1}^{n} (J_{1_j} - J_{2_j})^2} \quad \ldots\ldots\ldots\ldots (2.4)$$

This technique is mostly dependent on two factors: the value of n and the user-customized measure. The output is decided by two factors: firstly, whether the expected outcome is a label, and secondly, whether the expected outcome is a numerical value. [22]. For a binary problem (such as victory or loss in baseball) in baseball, the perspective user can see how the probability of a home side winning is generated. Its mathematical probability can be computed as equation 2.5 [23].

$$p(w = 1) = \frac{1}{k} \sum_{j=1}^{n} J(w_j = 1) \ldots\ldots\ldots\ldots\ldots\ldots (2.5)$$

Here, p (w=1) denotes the Home Team Win's probability, and w_j is the response variable for KNN.

c). Support Vector Machines (SVM)

The support vector machine (SVM) is a popular machine learning approach. It quickly gained popularity following its invention in the 1990s. It cannot handle nominal data; hence, preprocessing is necessary to transform it to numerical data. SVM is useful for regression, multi-classification, binary classification, etc. This SVM approach supports linear, polynomial, Gaussian, exponential, Laplacian, hyperbolic or sigmoid, Anova radial basis, radial-basis function, wavelet, spectral, Mahalonibus, dot, neural, multi-quadratic kernels, and so on. The concept is simple, and it is mostly used to choose a hyperplane as the decision boundary that can split variables based on their category (0 or 1). The major disadvantage of the support vector machine technique is that it is unsuited for large data sets and performs badly when the data set has more noise, i.e., target classes overlap. SVM identifies the extreme points/vectors that help create the hyperplane. These extreme instances are referred to as support vectors, and the technique is known as the Support Vector Machine.

d). Artificial Neural Networks (ANNs)

ANNs (Artificial Neural Networks) have been used in a wide range of baseball analysis applications, including player performance prediction, game outcome prediction, and player scouting. One common use of ANNs in baseball is to predict a batter's performance against a certain pitcher. The network is trained using historical data in this situation, such as the batter's prior performance against other pitchers, the pitcher's pitch selection, and the game circumstances. This information is then used by the network to forecast the likelihood of the batter getting a hit or reaching base against a certain pitcher in a given circumstance. Another use of ANNs for baseball is predicting games. In this instance, the network is trained using a variety of game data, including the team's win-loss record, starting pitcher performance, and offensive and defensive statistics. The network then uses this data to predict the likelihood of each side winning the game. ANNs were applied in player scouting to forecast a player's professional success based on physical attributes and performance data. In this scenario, the network is trained using historical data, such as physical characteristics and prior player performance records, to anticipate a prospect's chances of success in the Major Leagues. The sigmoid function was applied with given input, hidden, and output states.

Figure 2.2 illustrates the concept of sigmoid function states.

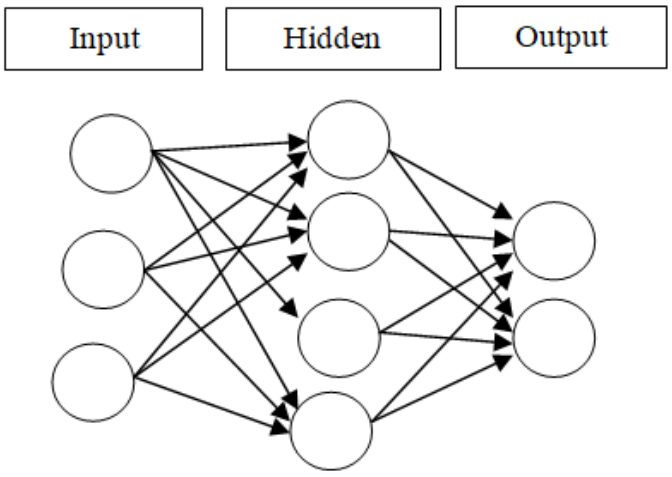

Figure 2.2 Sigmoid function states

Input States: (hitting average difference, pitching ERA difference)
Hidden States: Neuroun1, Neuroun2, Neuroun3

Output States: Win /Loss, qualify for playoff / not qualify for playoff

e). Decision Trees and Random Forests

Typically, decision trees and random forests are used to solve classification and regression issues. The fundamental advantage of a decision tree is that it is adaptable, can be quickly fitted to a dataset, and the resulting model can be viewed and analyzed. Decision trees are less accurate and stable when applied to new data because they produce binary linear splits within it [23]. The random forest, which is a collection of decision trees, is an extension of the decision tree. Random forest demands more processing resources, but it is more reliable than a decision tree. Figure 2.3 illustrates the concept of decision trees and random forests. Tree-based models repeatedly partition the space of observations depending on data conditions such as (home team =1 versus 0; hitter's data; team average). Finally, each final division of observations is assigned a classification or regression value based on the Gini index or entropy. The Gini index is used to calculate the variance across classes, and for a two-class (win/loss) model, it shows the proportion of observations in each partition with a simplified outcome value. Entropy, on the other hand, is a measure of information gain that minimizes the amount of additional depth and splits required to divide tree components. The formula mentioned in equation 2.6 computes the Gini index for a single class.

The Gini index is defined as:

For a single class

$G = \sum_{K=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$ …… (2.6)

Gini index = denoted by G; K = K denotes classes.

$\hat{p}_{mk}$ = Small value close to zero or one.

Gini index for baseball match prediction by using two classes (win (1)/loss (0)) is computed by equation 2.7.

$G = \hat{p}_{m,0}(1 - \hat{p}_{m,0}) + \hat{p}_{m,1}(1 - \hat{p}_{m,1})$ …… (2.7)

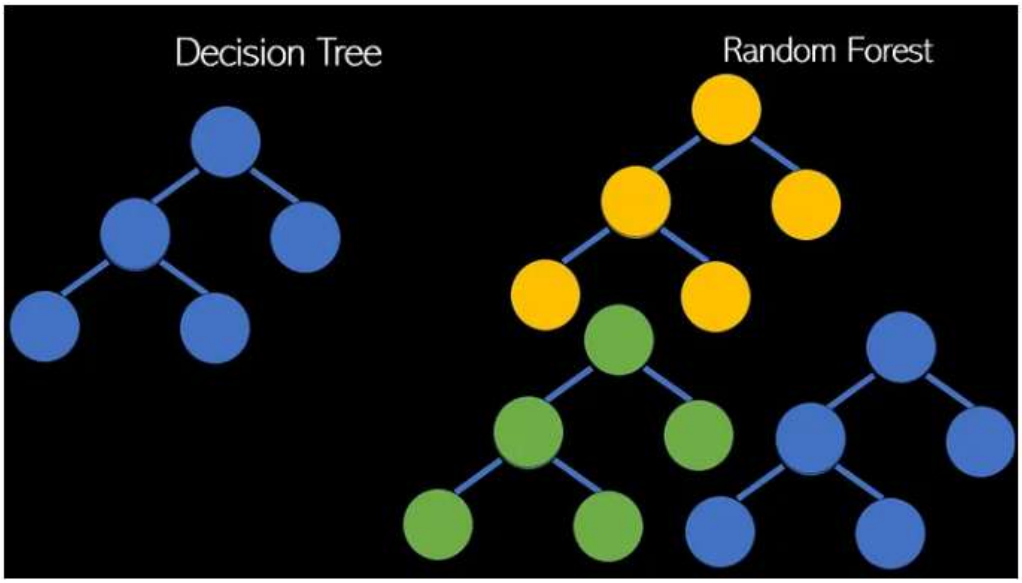

Figure 2.3 Decision Trees and Random Forests

[Source: www.medium.com]

An alternative to the Gini index is entropy (D). The entropy is defined as for a single class is computed by using equation 2.8.

$D = -\sum_{K=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk.}$ …… ..(2.8)

If the $\hat{p}_{mk}$'s are all near zero or near one, the entropy will be near zero, and it will also be tiny if the mth node is pure.

Entropy for baseball matches prediction by using two classes (win (1)/loss (0)) is computed by equation 2.9.

$D = -\left[\hat{P}_{M,0}LOG(\hat{P}_{M,0}) + \hat{P}_{M,1}LOG(\hat{P}_{M,1})\right]$ …….. (2.9)

Random forests, also known as bootstrap random forests, are created by combining numerous decision trees to improve accuracy while minimizing overfitting, which explains their popularity. The random forest algorithm is detailed further below.

- Create n_{tree} bootstrap sample from novel data
- For each bootstrap sample, create an unpruned classification tree with each node randomly sampling m_{try} of the predictors and selecting the best split among other variables.
- Predict new data by aggregating the n_{tree} trees' predictions.

The margin function may be expressed as a formula employing an ensemble of classifiers h1(X), h2(X),...,hk(X) and a random forest training dataset drawn from the distribution of vector X, Y as illustrated in equation 2.10.

$mg(X, Y) = av_k I(h_k(X) = Y) - max_{j \neq Y} av_k I(h_k(X) = j)$ …… (2.10)

f). ELO Ratings System

Elo ratings are a popular method for grading individuals/teams that participate in pairs. Arpad Elo created them as a chess rating system, but they have subsequently extended across sports. In baseball games, one team wins, and the other loses. Elo ratings are comparable to Log5 and have been used to quantify teams' relative strengths over time in order to anticipate baseball game results in the past. The Elo method is interpretable and changeable, but the Elo update process and expectations are predetermined formulas. The Elo model enables generalizing statements about individual games into a season-based prediction summary, giving comparable insights but more precise information. Figure 2.4 illustrates how the outcome of a match and the two competitors' ratings combine to yield ratings gains and losses for the players. Elo allows for the insertion of elements that influence the outcomes of specific matches, as well as the integration of machine learning analysis via covariates that impact team rankings. Finally, since Elo is a continually updated model, it can be tracked over time. The ELO rating system is computed using the various Elo formulas mentioned in equations 2.11 (sigmoid scale of 400), 2.12 (probability of winning team A), 2.13 (probability of winning team B), 2.14 (amount of K), and 2.15 (Elo logistic regression of the final result of a game versus B).

Sigmoid scale of 400:

$$\sigma(x) = \frac{1}{1+100^{\frac{-x}{400}}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.11)$$

Probability of winning (Team-A):

$$\text{Prob}\,(A > B) = \sigma(R_A - R_B) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (2.12)$$

Here, the author will assume Team-A wins, then the error of winning Team-A is the probability of winning Team-B.

$$\text{Err}_{(\text{Win}=A)} = \text{Prob}\,(B > A) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (2.13)$$

Amount of K:

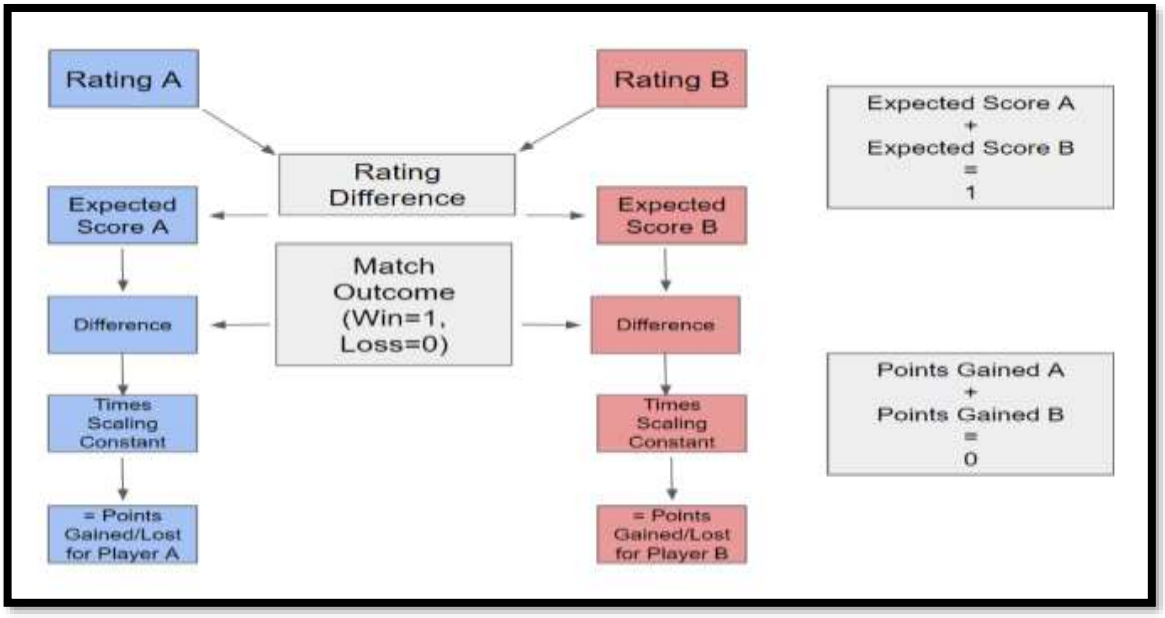

Figure 2.4 ELO Ratings System

[Source: www. thedatajocks.com]

$$R_A^{t+1} = R_A^t + K(w(A\ \text{wins}) - E_A) \quad \ldots\ldots\ldots \quad (2.14)$$

Here, R_A^t = Team-A prior rating

R_A^{t+1} = Team-A updated rating (after the match)

K is known as the K-factor or constant, and specifies how significantly to weight match outcomes.

$w(A\ \text{wins})$ = If A team wins, then 1, if A loses, then 0 (no tied games in MLB).

E_A = Elo logistic regression of the final result of a game versus B (denoted by R_B).

$$E_A = \frac{1}{1+100^{-(R_A - R_B)/400}} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (2.15)$$

The E_A denotes a probability of Team-A (match winning) because ties are impossible in baseball

F) One-Dimensional CNN

A one-dimensional CNN (1DCNN) may be used to evaluate signal and time-series data. The network structure of 1DCNN has not been determined. The 1DCNN network architecture is built and updated based on the data size. For the first time, this technique was used to forecast baseball games [3]. There are eight layers in all, organized as follows:

maximum pooling layer, dropout layer, 1D convolutional layer, connected layer, and output layer, all of which employ the sigmoid activation function.
Following equation 2.16 illustrates the sigmoid activation function.

$$f(x) = \frac{1}{1+e^{-x}} \quad \text{..... (2.16)}$$

G) Boosting models and XGBoost

Boosting modeling series differs from random forest, and it combines a series of weak trees in aggregate. It improves prediction accuracy during training and minimizes data characteristics, allowing for effective handling of huge datasets [23]. For tree-based classifiers, XGBoost analysis is a faster and more accurate gradient boosting approach [24]. Because of its accuracy and speed, this data science technique is highly popular. To develop an effective learning technique through additive training. Figure 2.5 illustrates the concept of XGBoost. The XGBoost method combines all of the weak learners' predictions [25]. Equation 2.17 is the formula to make predictions:

$$f_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = f_i^{(t-1)} + f_t(x_i) \quad \text{..... (2.17)}$$

Where $f_t(x_i)$ is a learner at step t, $f_i^{(t)}$ and $f_i^{(t-1)}$ is the prediction in step t and t-1, and x_i as the input variable.

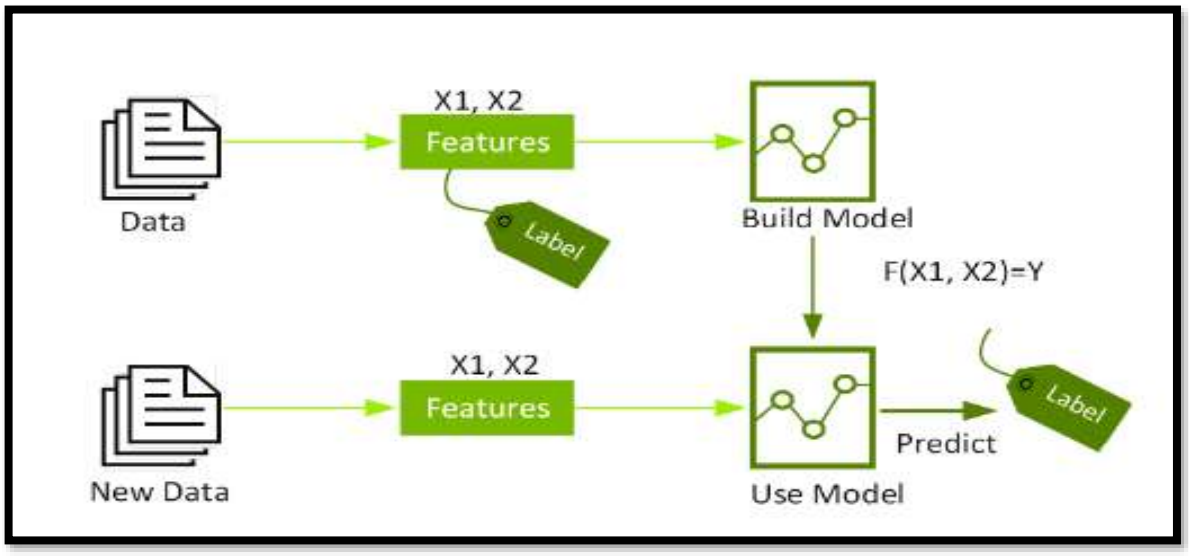

Figure 2.5 XGBoost

[Source: www.nvidia.com]

H) Pythagorean Win Expectation

The most basic and widely used statistical model available currently was developed by Bill James and known as Pythagorean Win Expectation, so named due to the form of the equation bearing a resemblance to Pythagoras' Theorem [26]. This technique predicts the number of games a team should expect to win in each season based on the number of runs they score and allow. This is generally used in a reverse engineering approach, which examines a team's performance at the conclusion of the season to assess whether or not they won as many games as they should have given the number of runs scored vs the number of runs allowed. It may also be applied in a predictive capacity by calculating how many runs a team should score and allow in an upcoming season in order to determine how many wins a team should expect in that upcoming season. To apply Naive Bayes in MLB predictions, features such as team statistics (e.g., runs, batting averages, ERA) and game context (e.g., home/away, weather conditions) are used. Each feature contributes to the overall probability of a team winning or losing a game. For example, the likelihood of a win is higher if a team has a strong batting average and a low ERA. The classifier calculates the conditional probabilities for each feature and combines them to determine the overall probability of each outcome. In research and practical applications, Naive Bayes has shown to be effective for classifying outcomes with reasonable accuracy. Although it assumes feature independence and a simplification often not true in sports, it still performs well due to its robustness and low computational cost, making it a valuable tool in MLB predictive analytics. The Pythagorean Win Expectation Model, in its original form, is expressed as a simple equation illustrated in equation 2.18:

$$\text{win \%} = \frac{(\text{Runs Scored})^2}{(\text{Runs Scored})^2 + (\text{Runs Allowed})^2} \quad \text{..... (2.18)}$$

- Runs Scored is the total number of runs scored by the team.
- Runs Allowed is the total number of runs allowed by the team.

- 2 is an exponent, often around 1.83 or 2, which adjusts the formula to better fit actual results

3.2 Comparative study of various AI models for MLB match outcome prediction

The primary focus is to determine which team wins a baseball game, disregarding specific outcome components like the final score. This frames the problem as a binary classification task, where the goal is to identify features that account for variance in game outcomes. The literature review is focused on the binary classification of individual baseball games and series; however, there is a wide range of machine learning applications addressing this problem, both in academic research and in analyses by sports writers. Below, the key examples from these sources are highlighted.

Figure 2.6 showcases a comparative analysis of AI-based models, focusing on four major categories: Highly Accurate Models, Highly Interpretable Models, Highly Flexible Models, and High Complexity Models. To explain these categories and their properties in detail, we analyze each of them.

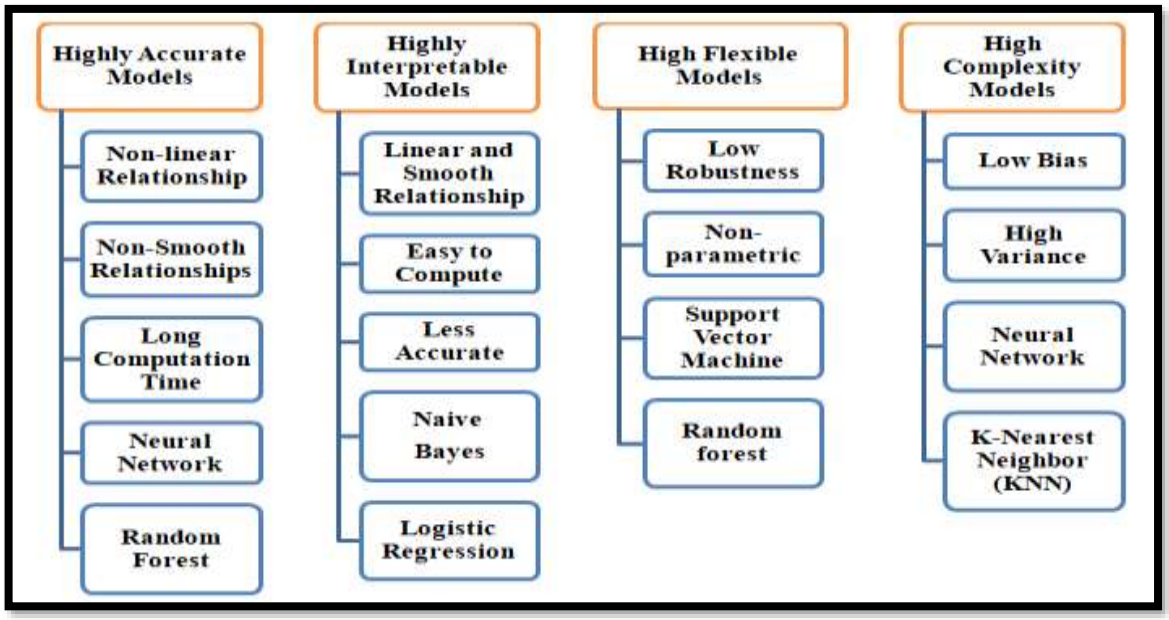

Figure 2.6 Comparative Analysis of Existing AI-Based Models

Highly Accurate Models prioritize precision over interpretability, meaning they can capture complex relationships in data. The accuracy of such models is usually achieved at the cost of increased computational requirements and difficulty in interpretation. These models can handle non-linear relationships between variables, where changes in input don't lead to proportional changes in output. A non-linear model is advantageous when the true relationship between features is complex. In finance and economics, non-linear models are frequently applied to capture non-linear dependencies in stock prices .

Non-smooth relationships refer to sudden changes in the relationship between variables, which cannot be captured by simpler models. Ensemble methods (like Random Forest) are often employed here, given their ability to adapt to these abrupt shifts in data. The highly accurate model is computationally intensive due to the complexity of their architecture. For instance, deep neural networks require extensive computational resources, often demanding GPUs for parallel computation .

Highly Interpretable models are designed for simplicity and ease of interpretation, sacrificing some accuracy for transparency. Interpretable models allow researchers and analysts to make clear inferences about the relationships between input variables and outputs. A linear relationship between variables, which makes the model outputs easier to explain. Linear models, such as linear regression, assume that the effect of one unit change in an input variable leads to a constant change in the output variable. Despite their simplicity, these models frequently fail in settings involving nonlinear and complicated interactions. For example, in clinical risk prediction, linear models are used due to their simplicity, although they are less accurate than machine learning techniques . These models are based on simple mathematical assumptions; they need far less computer resources than more precise or flexible models. This makes them especially valuable in scenarios when computing resources are limited. Logistic regression and Naïve Bayes are highly interpretable models for binary classification applications. It is commonly used in the social sciences and health studies because of its interpretability and ability to shed light on variable relationships . For example, it can quickly determine how particular risk variables influence health outcomes. Logistic regression is especially useful when the connection between variables is linear and categorical variables are present. However, its performance degrades in non-linear settings .

Highly Flexible Models refers to the ability to model complex, non-linear relationships in data. Highly flexible models can adapt to a variety of data structures, making them suitable for tasks involving noisy or unstructured data. Highly flexible models are often less robust because they can easily overfit the data, especially when there is noise. Overfitting

occurs when the model captures the noise in the data as if it were part of the underlying pattern. Flexible models like decision trees can often be prone to overfitting unless regularized .

Many highly flexible models are non-parametric, meaning they don't make assumptions about the form of the underlying data distribution. Instead, these models derive their structure from the data itself. Examples include Support Vector Machines (SVM) and k-nearest neighbors (KNN). Non-parametric models are essential in cases where the data distribution is unknown or difficult to define [87].

SVMs and random forests are popular examples of highly flexible models. They both are capable of handling linear and non-linear classification tasks by using kernel functions to map data into higher dimensions. SVMs are particularly effective when there is a clear margin of separation between classes. Despite their flexibility, SVMs can become computationally expensive and less interpretable compared to linear models .

High complexity models are characterized by their ability to fit highly complex data structures at the cost of being difficult to interpret. They are most useful when dealing with large datasets and scenarios where predictive power is the priority over explanation. Furthermore, highly complex models tend to have low bias because they can capture intricate relationships between variables. Bias refers to the error introduced by approximating a real-world problem with a simplified model. In high complexity models like deep neural networks, the approximation is much closer to reality, resulting in lower bias but potentially higher variance. But the complexity of these models often leads to high variance, meaning that they are sensitive to fluctuations in the training data .

Models like decision trees and KNN can be highly accurate on training data but may perform poorly on unseen test data due to overfitting. Methods like random forests or ensemble learning techniques are often used to mitigate this problem by averaging predictions from multiple models [90].

KNN is a simple, non-parametric method used for classification and regression. Although it is easy to understand, KNN can become computationally expensive when dealing with large datasets because it stores all the training data and searches for the nearest neighbors during prediction. It's often used in recommendation systems or pattern recognition tasks. However, due to its sensitivity to noise and its performance limitations with high-dimensional data, KNN may not be the best choice for all tasks .

Table 2.1 illustrates the comparative study of various AI models for the prediction of major league baseball match outcome in terms of prediction accuracy, feature extraction capability, interpretability, strengths, weakness and MLB sustainability.

Table 2.1 Comparative study of various AI models for the prediction of major league baseball match outcome

Model	Prediction Accuracy	Feature Extraction Capability	Interpretability	Strengths	Weaknesses	MLB Suitability
GLM (Generalized Linear Models)	Medium	Low (requires manual feature selection)	High	Interpretable, statistically grounded	Linear assumptions, underfit complex patterns	For basic analysis
K-Nearest Neighbors (KNN)	Low-Medium	None (lazy learner)	Medium	Simple, intuitive	Poor with high dimensions and large datasets	Weak for MLB
SVM (Support Vector Machines)	High	Medium (through kernel trick)	Low-Medium	Effective in high-dimensional spaces	Hard to tune, less interpretable	Situational use (small datasets)
Artificial Neural Networks (ANNs)	High (with enough data)	High (learns latent features)	Low	Captures complex patterns	Needs large data, less explainable	Good with large datasets
Decision Trees	Medium	Medium	High	Easy to interpret, fast	Overfits easily, unstable	Good baseline
Random Forests	High	High (feature importance)	Medium	Robust, handles non-linearity	Less interpretable than trees	Strong choice for tabular MLB data
ELO Ratings System	Medium	Low	High	Simple, interpretable, adaptive	Ignores other match factors (e.g., injuries, weather)	Good for team strength

						estimation
1D CNN (Convolutional Neural Network)	Medium-High	High (learns spatial/temporal patterns)	Low	Learns sequence patterns (e.g., batting order, pitch sequence)	Not ideal for structured/tabular stats	Useful for sequential MLB data
Boosting Models (e.g., AdaBoost, LightGBM)	High	High (robust feature learning)	Medium	Handles non-linearity, accurate	Prone to overfitting if not tuned	Widely used in sports analytics
XGBoost	High	High	Medium	Best-in-class for tabular data, fast, scalable, and interpretable via SHAP	Less effective with raw, unstructured data	Basic for most MLB tasks
Pythagorean Win Expectation	Low-Medium	None (heuristic formula)	High	Simple and historically insightful	Ignores granular features	Basic baseline for team strength

2.3 Why Naive Bayes, SVM, and Random Forest work well for MLB Match Outcome Prediction?

MLB outcome prediction is a binary or multiclass classification task (e.g., Win vs Loss, Home Win vs Away Win), where models are trained on historical player, team, and game data. Now, a question arises: why Naive Bayes, SVM (Support Vector Machine), and Random Forest are often considered effective models for predicting Major League Baseball (MLB) match outcomes?

Table 3.2 explains the reasons why the mentioned models work well and their limitations. Table 3.3 predicts the criterion behind the selection of Naive Bayes, SVM (Support Vector Machine), and Random Forest for MLB match prediction outcomes.

Table 2.2: Why Naive Bayes, SVM, and Random Forest work well for MLB Match Outcome Prediction?

Model	Why it works well	Limitations
Naive Bayes	• Fast and simple: Great for high-dimensional data like player stats, weather, team history, etc. • Works well with text/numerical data: Useful if you incorporate textual game summaries or social media sentiment. • Handles small sample sizes: Ideal when data per matchup is limited.	Assumes feature independence, which may not always be valid (e.g., team batting average and runs scored are related).
Support Vector Machine (SVM)	• Excellent at binary classification (e.g., Win vs Loss). • Effective with complex, non-linear relationships using the kernel trick. • High accuracy on clean, balanced datasets.	Requires careful feature scaling, and training is slow for large datasets.
Random Forest	• Handles both categorical and continuous variables (e.g., weather, player positions, batting average). • Captures feature interactions and non-linear patterns. • Robust to overfitting due to an ensemble of decision trees. • Provides feature importance, helping identify key predictors (e.g., pitcher ERA, home field advantage).	Less interpretable than simple models.

Table 2.3 Criterion behind the selection of Naive Bayes, SVM, and Random Forest for MLB Match Outcome Prediction

Criterion	Naive Bayes	SVM	Random Forest
Speed	Very fast	Moderate	Slower than NB, faster than SVM
Accuracy	Good (simple patterns)	High (complex patterns)	Very High (ensemble learning)

Overfitting Risk	Low	Medium	Low
Feature Engineering Needed	Minimal	Requires scaling	Minimal
Works well with	Text, simple data	Medium-sized structured data	Large, complex datasets
Output Interpretability	High	Medium	Moderate

Table 2.4 Comparison of Prediction Models used in MLB matches based on the Data Set used and Accuracy

Literature	Method	Year	No. of datasets used	Data Set	Accuracy
Chen (2014)	ANN, Logistic regression	Six years (2006-2012)	02	Yankees and Red Sox Teams Data	73.68%
Valero (2016)	SVM, ANN, Decision Tree, Lazy Learners (KNN)	Nine years (2005-2014)	02	Retro sheet and Lahman database	59.00%
Jia et. al (2013)	Random forest, SVM, Adaptive Boost, Logistic Boost, MLR	Seven years (2005-2012)	01	Retro sheet	59.60%
Huang and Li (2021)	1D-CNN, ANN, SVM	One year (2019)	01	Baseball-Reference	94.10%

From the above Table 2.4 shows that Jia et al., Chen, and Valero obtained accuracy levels of 59.6%, 73.68%, and 59%, respectively, while Huang and Li reached 94.10%. Huang and Li's (2021) study only collected data for one year, 2019. Li has only utilized one feature selection approach (Relief), which poses a challenge in maintaining accuracy. Instead, using a different feature selection strategy and a bigger data set may have yielded better results.

Jia et al. analyzed data from 2005 to 2012 and observed that it is extremely difficult to predict the outcome of a game like baseball. Secondly, there is a mention about how feature selection can improve the prediction levels, but the feature selection of his study was also very limited, and Jia realized it when the predictability level was not on the expected lines. Soto Valero analyzed data from 2005 to 2014. In his study, Valero used four prediction algorithms and determined that more statistical data is needed to enhance the results. Chen worked on two approaches to predict game outcomes and gathered data from 2006 to 2012. Because only chosen SP data were utilized as input variables, the predictability level in this study is rather high. The various datasets of MLB matches for outcome predictions were used by various eminent researchers. Changing the factors produced different prediction results, but in order to make an accurate and reliable prediction, it is important that every dimension that has an effect on the game should be incorporated in the model used for making a prediction. So, to enhance the accuracy it is important that these methods are studied and analyzed in a proper manner. The data sample must be large enough to allow for some generalization. At the same time, keep in mind that the variables included in the study should encompass all components of the game that might influence the game's outcome.

From Table 2.5, it can be comprehended that most of the authors used regression techniques, which gives the probability of winning or losing. The study mentioned the relationship between different methods and the translation of these into the probable outcome of the game result. In most of the studies, authors have used past data to make predictions; however, the usage of linear regression is comparatively easy, but it results in decreased accuracy when it comes to prediction. Under this comparison, flexibility, complexity, and Interpretability are measured, and it was found that the flexibility of SVM, ANN, 1DCNN, Gradient boosting, and Bayesian models is high. The complexity of SVM, ANN, 1DCNN, KNN, and Bayesian models is high. Interpretability is highest in Logistic regression, Pythagorean expectation, Elo rating, and random forest. But if we look at the future perspective, deep learning techniques like Artificial Neural Networks could be the way forward for predicting the game results.

Table 2.5 Comparison of different models of Artificial Intelligence used in MLB dimensions like flexibility, complexity, and Interpretability

Method	Literature	Applications in baseball	Flexibility	Complexity	Interpretability
K-Nearest Neighbors (KNN)	Valero & Koestler et al.	Bays' Rule is used to compute the conditional probability of a victory or loss with respect to the k-	Medium	High	Low

		nearest other finds using a distance metric.			
Logistic regression (GLM)	Yaseen Elfrink et al. Al Tait et. al & Jia et. al	It examines a linear connection between the variables to produce a result that may be classified as a win or a loss.	Medium	Low	High
Decision trees and random forests	Valero (2016) Elfrink et. al & Jia et. al	Classifies each segment after recursively dividing the data into pieces based on knowledge gained; To avoid over-fitting, random forests take into account bootstrap subsets of characteristics at each split.	Medium	Medium	Medium/high; lower for randomforests
Support Vector Machine	Huang et. al Yaseen & Valero Tolbert et. al	Builds polynomial kernel decision boundaries that categorize games as wins or losses in the hyperspace of the game and establish clear distinction.	High	High	Low
Artificial Neural Nets	Huang (2021) & Valero (2016) Chen (2014)	The decision-making process in the brain uses successive layers of "neurons" with different edge Weights.	Very high	High	Negligible
Bayesian Model	Soicher	Predicting the match result by using the Bayesian rule.	High	High	Medium
Elo Ratings	Heumann & Hvattum et al.	Ratings are kept track of as team skill levels that describe past probabilities of winning a game; ratings are updated based on the outcome and the other team's rating.	Medium	Low	High
Gradient Boosting	Elfrink et. al & Jia et. al	Merges some weaker predictive models, such as trees, by successively altering weights, boosting overall predictability.	High	Medium	Medium
Linear Regression	Bailey	Predicting the batting averages by simple Linear regression.	Low	Low	Medium
Pythagorean Expectation	Heumann	Calculates an expected outcome (win/loss) based on the team's run-scoring and run–prevention abilities	Low	Medium	High
1-DCNN	Huang	For the first time, 1-DCNN was used to make	High	High	Low

		predictions regarding the outcomes of MLB games.			

Elfrink and Bhulai (2018) employed less flexible approaches, such as generalized linear regression models and logistic regression, alongside tree-based methods like random forests and gradient boosting. Despite addressing the same classification problem, their models achieved slightly lower classification accuracies, ranging from 53.75% to 55.52%, compared to Valero (2016), whose models reached 55.98% to 58.92%. Similarly, Pharr (2019) utilized gradient boosting techniques, specifically LightGBM and XGBoost, achieving improved accuracies of 59.3% and 60.7%, respectively.

Anecdotally, consider managerial decisions, such as those made by Oakland manager Billy Beane, as portrayed in Moneyball [50], where the researcher had to gain the trust of his scouts and team ownership to facilitate trades for new players. In such situations, the absence of interpretable models could undermine his confidence and persuasiveness, ultimately rendering even theoretically sound and accurate models ineffective in practice. While predictive accuracy is crucial, it is important to recognize that baseball prediction models may be applied in contexts where more than numerical precision is required.

There are numerous potential applications for binary classification in baseball. Simple examples include classifying a matchup between two teams as a win or a loss, predicting whether a player will attempt a bunt, and determining if a team will intentionally walk a batter. In the first instance, the observations may consist of a vector of players along with their individual performances, utilizing statistics such as on-base percentage or more advanced metrics. For the scenarios involving bunting and intentional walks, situational data—such as the number of outs, the game score, and the tendencies of players or managers—can be factored into the analysis. Based on the findings reported in the literature, we recommend starting with either a support vector machine (SVM) approach or linear discriminant analysis (LDA). Both algorithms have demonstrated strong predictive accuracy. While Hoang et al. advocate for the use of LDA, they note that the improvements observed are marginal compared to the SVM approach. SVMs are relatively straightforward to construct and manipulate. Ultimately, each analyst should engage in experimentation to identify the algorithm that best suits their specific problem domain. In our literature review, the authors identified several algorithms that were frequently employed, notably support vector machines (SVMs), k-nearest neighbors (KNN), and Bayesian inference. Among the articles, it has been examined that Bayesian inference emerged as the predominant method for regression tasks, which were the most commonly addressed in the literature. Some studies utilized established machine learning software such as WEKA or R for their analyses, while many researchers opted to implement their algorithms manually. This indicates both the significant potential and the necessity for the practical application of these methods. It underscores the point that future researchers and analysts should not confine themselves to existing software solutions.

3. Conclusion and future scope

The present study aims to review the artificial intelligence-based existing models for predicting the outcome of MLB matches. The eminent researchers had put their efforts into adopting a different technique to forecast the accuracy of the match outcome. The accuracy level ranges from 50 to 60 percent and has more to do with the number of factors chosen for the study. One more observation which comes out of the study is the accuracy levels; that are good when the data set is taken for a lesser duration, i.e., one or two years. On the other hand, when data is taken for a longer duration, like ten to fifteen years, the accuracy levels go down. There is a need to work on a model that can improve the accuracy in cases where data is taken for longer durations, more than ten years. This can be improved by selecting variables that have a higher impact on the game outcome.

4. References

1. Moneyball Lewis, M. (2003). Moneyball: The Art of Winning an Unfair Game. New York: W.W. Norton & Company.
2. Chen, T. H. (2014). Predicting Baseball Game Outcomes Using Machine Learning Techniques. International Journal of Computer Science in Sport, 13(2), 1–12.
3. Elfrink, R., & Bhulai, S. (2018). Predicting outcomes of Major League Baseball games. International Journal of Sports Science and Engineering, 12(1), 25–34.
4. Yang, T. Y., & Swartz, T. B. (2004). A Two-Stage Bayesian Model for Predicting Winners in Major League Baseball. Journal of Data Science, 2(1), 61–73.
5. Pharr, J. (2019). Predicting MLB player WAR using machine learning techniques. Journal of Sports Analytics, 5(3), 145–156.
6. Baumer, B., Jensen, S., & Matthews, G. (2015). OpenWAR: An open source system for evaluating overall player performance in Major League Baseball. Journal of Quantitative Analysis in Sports, 11(2), 69–84.
7. James, B. (1982). The Bill James Baseball Abstract. Ballantine Books.
8. Doo, W., Kim, Y., & Kim, S. (2018). Modeling the probability of a batter/pitcher matchup event. Journal of Quantitative Analysis in Sports, 14(4), 215–229.
9. Jiang, Y., Paul, A., & Reddy, C. (2007). Bayesian estimation of batting averages in Major League Baseball. Statistical Methodology, 4(3), 320–332.

10. Huang, C. Y., & Li, Y. H. (2021). Prediction of Major League Baseball games using One-Dimensional CNN. Applied Artificial Intelligence, 35(9), 712–728.
11. Yaseen, Z. (2019). Machine learning approaches for baseball match prediction. International Journal of Computer Applications, 178(7), 20–27.
12. Bailey, M. (2011). Linear regression analysis of batting averages in Major League Baseball. Journal of Sports Statistics, 6(2), 55–66.
13. Soto Valero, C. (2016). Predicting Win-Loss outcomes in MLB regular season games using machine learning. International Journal of Computer Science in Sport, 15(2), 48–63.
14. Koestler, T., Johnson, R., & Brown, P. (2017). K-nearest neighbor models for baseball outcome prediction. Journal of Sports Analytics, 3(1), 11–20.
15. Tolbert, A., Labelle, J., & Worthley, R. (2018). Support Vector Machines for baseball game prediction. Procedia Computer Science, 140, 354–361.
16. Jia, H., Li, X., & Wang, Z. (2013). Major League Baseball match outcome prediction using Random Forest and SVM. Expert Systems with Applications, 40(16), 6228–6237.
17. Heumann, C. (2014). Elo ratings and sports prediction systems. Journal of Quantitative Analysis in Sports, 10(3), 231–245.
18. Hvattum, L. M., & Arntzen, H. (2010). Using Elo ratings for match result prediction in association football. International Journal of Forecasting, 26(3), 460–470.
19. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32.
20. Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics, 29(5), 1189–1232.
21. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794).
22. Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1), 21–27.
23. Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297.
24. Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386–408.
25. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323, 533–536.
26. James, B. (1981). The Pythagorean theorem of baseball. Baseball Abstract, 14–16.
27. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
28. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning. Springer.
29. Soicher, L. (2010). Bayesian inference models for sports analytics. Journal of Statistical Sports Analysis, 8(2), 99–112.
30. Quinlan, J. R. (1986). Induction of decision trees. Machine Learning, 1(1), 81–106