Use of Machine Learning and Deep Learning to Predict the Outcomes of Major League Baseball Matches

Mei-Ling Huang * and Yun-Zhi Li

Abstract: Major League Baseball (MLB) is the highest level of professional baseball in the world and accounts for some of the most popular international sporting events. Many scholars have conducted research on predicting the outcome of MLB matches. The accuracy in predicting the results of baseball games is low. Therefore, deep learning and machine learning methods were used to build models for predicting the outcomes (win/loss) of MLB matches and investigate the differences between the models in terms of their performance. The match data of 30 teams during the 2019 MLB season with only the starting pitcher or with all pitchers in the pitcher category were collected to compare the prediction accuracy. A one-dimensional convolutional neural network (1DCNN), a traditional machine learning artificial neural network (ANN), and a support vector machine (SVM) were used to predict match outcomes with fivefold cross-validation to evaluate model performance. The highest prediction accuracies were 93.4%, 93.91%, and 93.90% with the 1DCNN, ANN, SVM models, respectively, before feature selection; after feature selection, the highest accuracies obtained were 94.18% and 94.16% with the ANN and SVM models, respectively. The prediction results obtained with the three models were similar, and the prediction accuracies were much higher than those obtained in related studies. Moreover, a 1DCNN was used for the first time for predicting the outcome of MLB matches, and it achieved a prediction accuracy similar to that achieved by machine learning methods.

Keywords: major league baseball; one-dimensional convolutional neural network; artificial neural network; support vector machine; prediction model

1. Introduction

The popularity of streaming media has helped promote interest in large sporting events. The National Basketball Association (NBA), Major League Baseball (MLB), and the National Football League (NFL) are well-known sports organizations whose events attract a large number of loyal spectators and fans. According to Statista, among the five major North American sports leagues in 2019, the NFL had the highest average attendance per game in North America (66,479 spectators) [1]; MLB had the second-highest average attendance per game (28,317 spectators) [1]. The details and outcome of each game and the performance of the players have become a daily topic of interest for fans, giving rise to many business opportunities (e.g., sports lotteries). Thus, game outcomes and player performance constitute sources of highly valuable information.

MLB is the highest level of professional baseball in the world and is the oldest North American professional sports league. Lim and Pedersen [2] found that in 2017, each MLB team generated an average of USD 300 million, which was equivalent to that generated by each NFL team on average, ranking second among all sports leagues. Forbes also reported that on average, MLB teams in 2019 were valued at USD 1.8 billion, an increase of 8% over the previous year. Approximately half of the MLB teams were valued at more than USD 1.5 billion, and the average revenue generated by teams was USD 330 million [3].
MLB teams invest millions of dollars to improve team performance through the conduction of various analyses. In 2003, Michael Lewis published a book called *Moneyball: The Art of Winning an Unfair Game* in 2003, which popularized the use of data analysis tools in the field of sports [4]. Baseball is regarded as quite complex, with many factors affecting the outcome of a game, including player performance, team spirit, weather, and tactics. The events of a baseball game are unpredictable because they are influenced by various factors; for fans, this unpredictability is what makes a baseball game so interesting.

*Moneyball* shows how statistical data analysis can be used in the field of baseball to improve player selection and game management and thus help win a game. Moreover, with the development of the sports gaming industry, analytical tools for predicting the outcome of sports competitions have become crucial [5]. In the past, most people made judgments based on subjective circumstances or experience; however, the prediction accuracy of these judgments was poor. Many scholars now use Baseball-Reference, FanGraphs, Baseball Prospectus and other similar websites to collect statistics on each team and player; they then apply machine learning methods to predict the outcome of MLB matches. Yang and Swartz [6] proposed a two-stage Bayesian model to predict the division winner in MLB. Chen [7] used logistic regression and artificial neural networks (ANNs) to build a prediction model for MLB matches. Soto Valero [8] used machine learning methods based on an ANN, support vector machine (SVM), decision trees, and a k-nearest-neighbors (k-NNs) algorithm to predict the outcome of MLB matches. Elfrink [9] used a random forest algorithm, eXtreme gradient boosting (XGBoost), a linear model, and boosted logistic regression to construct models for predicting the outcomes of baseball games.

The aforementioned studies used machine learning methods to predict the outcomes of MLB matches. Only a few scholars have employed deep learning methods to predict the outcomes of sporting events. Chen et al. [10] proposed a self-encoding two-dimensional (2D) convolutional neural network (2DCNN) in 2020 to predict the performance of NBA players and the outcomes of matches; the results were then compared with those obtained using an ANN. Recently, deep learning has become popular and has been used in various fields. A convolutional neural network (CNN) is one of the most popular deep learning networks and has been used to successfully solve machine learning problems [11]. A CNN can process data in various formats; it is generally used in image processing and has achieved outstanding results. A one-dimensional (1D) CNN (1DCNN) is suitable for processing data such as sequences or signals, and it is relatively easy to train [12]. These CNNs have been used for tasks such as crop yield prediction [13], the prediction of wind speed and direction [14], unipolar depression diagnosis [15], and the prediction of film and television ratings [16], with outstanding results.

Horvat and Job [17] reported on the accuracies of sports outcome prediction. The highest accuracy was noted for the outcome prediction of American football matches (>90%), followed by cricket matches (highest accuracy rate >85%). The accuracy of outcome prediction for baseball matches was relatively low (50–60%). Therefore, this study aimed to establish prediction models (1DCNN, ANN, and SVM) for predicting the outcomes of MLB matches by selecting influencing variables. Next, whether feature selection has an impact on the performance of the models when predicting the outcome of a match is discussed. This article is structured as follows: Section 2 presents the literature review. Section 3 describes the databases and methods used in this study. Section 4 presents the results of the study and compares the results of this study with those of other studies. Section 5 presents the conclusions.

### 2. Literature Review

This section introduces the related literature of selected variables of MLB matches; methodology of feature selection; prediction of the outcomes of sport matches; application of 1DCNN to demonstrate the importance and sufficiency for the object of this study.
2.1. Variables Selected in Baseball Research

In the 1970s, Bill James began to write the book *Baseball Summary* and proposed sabermetrics (also known as baseball statistics) in the late 1970s [18]. There is no clear distinction between common baseball statistics and sabermetrics. Common statistics include the numbers of home runs (HRs), runs batted in (RBIs), and stolen bases. Data such as the on-base percentage (OBP), on-base plus slugging (OPS), and earned run average (ERA) that need to be calculated are called sabermetrics.

Many websites record MLB baseball game information, and each website records slightly different statistics. The most popular websites include Baseball-Reference, FanGraphs, Sean Lahman’s database, and Retrosheet. Related studies [8,9,19] have used competition files from Retrosheet and different feature selection methods to select the most relevant variables for input into the models for predicting the outcome of a match. Tolbert and Trafalis [20] used baseball data from FanGraphs, Sean Lahman’s database, and mlb.com to predict the outcome of an MLB league championship. No studies have used data from Baseball-Reference to predict the outcome of a baseball game. The data recorded on the Baseball-Reference website are sufficient and detailed; these data include the number of pitches in the plate appearance (PA) (Pit) for batters, number of swinging strikes (StS) for pitchers, and game score (GSc) for starting pitchers (SPs). Therefore, in this study, the Baseball-Reference website was selected for the collection of baseball data.

In 2018, Koseler and Stephan [21] reviewed the literature on outcome prediction for MLB matches. Most studies predicted pitchers’ performance, which indicates the importance of pitchers in baseball games. Elfrink [9] used the competition data of 30 teams in the MLB from 1930 to 2016 to construct a predictive game win–loss model. The research variables were hitting, pitching, and defensive data, all of which were team data. Chen et al. [7] constructed a predictive game win–loss model for the Boston Red Sox and the New York Yankees in 2014, and only selected data from the SP were used as the input variables. Selecting different research variables provided different prediction results. Therefore, to compare the prediction accuracies, the present study divided the pitcher data into two data sets: data set 1 included the data on the SP only, and data set 2 included data on all pitchers.

2.2. Feature Selection

Feature selection helps to interpret data, reduce computational requirements, and reduce the impact of the curse of dimensionality, thereby improving the performance of the prediction model [22]. In 2010, Trawiński [23] used eight feature selection methods in the Weka to select the most valuable attributes from 15 variables; the top five variables with the most votes were input into the prediction model. In 2016, Soto Valero [8] used five feature selection methods in Weka to select the 15 most important features from the original 60 variables using the majority vote process; these 15 features were input into the prediction model. In 2013, Jia et al. [18] divided baseball data into three categories—batting, pitching, and team data—and used three methods (best subset selection, forward selection, and backward selection) to select features for each category; seven input variables were finally selected. In 2014, Chen [7] used logistic regression for feature selection; sensitivity analysis was used to calculate the importance of each independent variable, and the independent variables with a relative importance of more than 80% were selected. In 2018, Elfrink [9] used XGBoost for feature selection. XGBoost, decision trees, and random forest models all use tree-based models to rank features according to their importance and determine the best features.

Many feature selection methods exist; the main objective of feature selection is to reduce the number of features and select the most relevant features as input variables to improve the prediction accuracy of machine learning. The feature selection method used in this study is ReliefFAttributeEval, and this study investigated whether feature selection has an impact on model prediction performance.
2.3. Predicting the Outcome of a Match

Sports analysis technology has been developing at a rapid pace, and it is generally regarded as one of the most popular topics in analysis technology [24]. Most studies have used traditional machine learning methods for analysis and prediction. For example, in 2019, Gu et al. [25] proposed ensemble methods to improve an SVM for predicting the outcomes of National Hockey League matches and improve recruitment and salary decisions. The average accuracy obtained was >90%. In 2019, Cai et al. [26] proposed a hybrid ensemble learning framework combining bagging and the random subspace method for predicting the outcome of the 2016–2017 Chinese Basketball Association season. The data set included 20 teams and 380 games. The final prediction results obtained were compared with those obtained using naive Bayes and ANN models; the proposed hybrid ensemble learning framework had a higher accuracy (of 84%) and a higher F1-Score (of 82%). In 2020, Chen et al. [10] proposed the use of a self-coding 2DCNN to predict NBA players’ performance and the outcomes of matches. The prediction accuracies of the model for all NBA team data from 2014 to 2017 were 91.78% and 91.64%, respectively. Li [27] proposed an improved back-propagation neural network (BPNN) in 2020 to predict the outcome of the main tournament of the Union of European Football Associations, which is the Champions League, and compared the results with those obtained using multiple linear regression (MLR) and detrended cross-correlation analysis. The improved BPNN had a prediction error of almost zero with high accuracy and reliability. Therefore, a neural network prediction model can provide a certain theoretical basis for sports competition and analysis.

In 2004, Yang and Swartz [6] collected data from 179 MLB games in the 2001 regular season. The ratio of the winning percentages of the two teams, ratio of the two teams’ overall batting averages, ratio of the ERAs between the two SPs, and home field advantage were used as variables for the proposed two-stage Bayesian model. They were effective in predicting the MLB division winner. In 2013, Jia et al. [18] collected MLB competition data from 2007 to 2012 using five machine learning methods: random forest, SVM, adaptive boosting (AdaBoost), logistic boosting (LogitBoost), and MLR. Competition data from 2007–2011 were used for training, and 2012 data were used for testing. The SVM exhibited the highest prediction accuracy of 59.6%. In 2014, Chen [7] used an ANN to construct a model for predicting the winning team in a match; they used logistic regression analysis to construct a prediction model for the two teams and compared their model predictions. They used data from the games of the New York Yankees and the Boston Red Sox from 2006 to 2013 and used 127 games between 2006 and 2102 to train the model. The results of the study showed that the ANN achieved an accuracy rate of 72.22%; subsequently, they used 19 games in 2013 to verify the prediction performance of the model, and the prediction accuracy was 73.68%.

In 2016, Soto Valero [8] used four machine learning methods—an SVM, an ANN, a decision tree, and a k-NN algorithm—and used them for predicting the outcomes of MLB matches using data between 2005 and 2014 of the 30 teams. The four methods were employed for classification and regression, and the outcome of a match was verified using tenfold cross-verification. The highest accuracy was obtained with the SVM classification method (59%).

In 2016, Tolbert and Trafalis [20] used an SVM to develop an MLB championship prediction model. SVMs with a linear kernel, quadratic kernel, cubic kernel, and Gaussian radial basis function (RBF) kernel were employed to predict the American League champions, National League champions, and the World Series champions for the 2015 season. In addition to predicting the poor performance of losing teams in the National League, the SVM with the RBF could predict the favorable performance of the champions of the American League and World Series. Although the results provided by the SVM were conflicting, the parameters and baseball statistics can be adjusted in the future to improve the prediction performance. In 2018, Elfrink [9] used four machine learning models—a random forest algorithm, XGBoost, a linear model, and boosted logistic regression—to
predict the outcomes of MLB matches. Elfrink used data from 38,870 games between the 1930 and 2015 seasons for training and used data from 2,428 games in the 2016 season for testing; fivefold cross-validation was employed, and the XGBoost model achieved the highest prediction accuracy, at 55.52%.

These studies conducted for predicting the outcomes of MLB matches used machine learning methods to construct the prediction models, and the accuracies of these models ranged from 55% to 73%. Most of these studies used an SVM and ANN to construct the prediction models, which obtained high accuracies. The present study used a 1DCNN, ANN, and SVM to construct MLB prediction models by selecting relevant variables and compared the prediction accuracies of these models.

2.4. DCNN-Related Literature

In recent years, CNNs have been widely used in various fields; they can process data of various formats, such as 1D data (signals and sequences), 2D data (images), and three-dimensional (3D) data (videos), with good results. 1DCNNs have been used in many fields in recent years. Mumtaz and Qayyum [15] proposed two deep learning frameworks, 1DCNN and 1DCNN + long short-term memory (LSTM) for diagnosing unipolar depression, and the highest accuracy rate achieved was 98.32%. Their results showed that the proposed deep learning framework could successfully diagnose depression automatically. Khaki et al. [13] proposed a deep learning framework for crop yield prediction using a CNN–recurrent neural network (RNN); they combined their proposed framework with other popular methods such as the random forest algorithm, deep fully connected neural network method, and least absolute shrinkage and selection to predict the yields of corn and soybeans in the United States from 2016 to 2018. Their results showed that the CNN–RNN model outperformed the other prediction methods. They used a 1DCNN to capture weather characteristics and soil data and used LSTM in the RNN to capture changes in crop yields over time due to genetic modification.

Harbola and Coors [14] proposed two 1DCNN deep learning algorithms—a 1D single CNN (1DS) and a 1D multiple CNN (1DM)—to predict the wind direction and speed from German and Dutch wind data sets. The 1DS predicts the main wind speed and direction using 1D time-series wind data, whereas the 1DM uses multiple views of time-series wind data to improve the accuracy of the 1DS. The total accuracies in predicting wind speed and wind direction were 95.2% and 95.1% for the 1DS and 98.8% and 99.7% for the 1DM, respectively. Ning et al. [16] proposed an effective regression model based on a generative CNN to predict movie ratings. The model also included a solution to overcome the problem of insufficient training samples. The experimental results obtained from using IMDb data indicated that their proposed model outperformed a set of baselines and state-of-the-art approaches; their model had a 9.3% lower mean square error and a 7.6% higher hit ratio. The results can provide a reference for investors, movie studios, and interested users.

All of the aforementioned studies employed a 1DCNN for various applications, and all achieved favorable prediction results. However, the 1DCNN has not been used for predicting outcomes in sports. Therefore, after data on baseball variables were obtained, a 1DCNN was used in the present study to establish a model for predicting the outcomes of MLB matches.

3. Methods

This section is structured as follows: (1) describe the structure of this study; (2) give details of the selected variables of MLB matches in this study; (3) present the methodology for feature selection; (4) construct prediction models; and (5) introduce the evaluation index of prediction models.

3.1. Procedure and Structure

The aim of this study was to predict the outcomes of MLB matches during the 2019 season, and this was achieved using deep learning and machine learning methods. The
research framework of this study is displayed in Figure 1. First, we separated the game data of the 30 teams into two data sets, the first of which contained data on batting, the SP, and home and away teams, and the second of which contained data on batting, all pitchers, and home and away teams. Next, we normalized the data and then performed feature selection. We then constructed the deep learning and machine learning models, and fivefold cross-validation was used to determine the prediction accuracy of the models and compare the prediction accuracies of the models before and after feature selection.

Figure 1. The research framework.

3.2. Data Collection and Preprocessing

In the 2019 season, 162 games took place. Because the Chicago White Sox and Detroit Tigers only played 161 games, we collected data from a total of 4858 games. All data were obtained from Baseball-Reference, a website that records data on batting, pitching, and games. For batting, the total group batting data in each game were selected (Figure 2); for the pitching category, because a game includes more than one pitcher, we refer to the first pitcher as the SP and the pitcher who plays later as the relief pitcher (RP). In this study, the starting pitcher data and the overall pitcher data were selected separately (Figure 3). Finally, we selected the variables of the home/away team from the relevant data of the game (Figure 4).

The first data set contained batting variables (13 in total, denoted B1–B13), SP variables (16 in total, denoted SP1–SP16), and one variable for the home/away team (denoted X1); thus, a total of 30 variables were selected for data set 1. The second data set included batting variables (13 in total, denoted B1–B13), variables of all pitchers (18 in total, denoted P1–P18), and one variable of the home/away team (denoted X1); thus, a total of 32 variables were selected for data set 2. Y was the output value (win/loss). The selected variables are listed in Tables 1 and 2.
Figure 2. Atlanta Braves batting statistics.

Figure 3. Atlanta Braves pitching statistics.

Figure 4. Related data.
Table 1. MLB input variable.

| Number | Variable            | Abbr | Number | Variable            | Abbr |
|--------|---------------------|------|--------|---------------------|------|
| B1     | At Bats             | AB   | B1     | At Bats             | AB   |
| B2     | Hits                | H    | B2     | Hits                | H    |
| B3     | Bases on Balls      | BB   | B3     | Bases on Balls      | BB   |
| B4     | Strike out          | SO   | B4     | Strike out          | SO   |
| B5     | Plate Appearances   | PA   | B5     | Plate Appearances   | PA   |
| B6     | Batting average     | BA   | B6     | Batting average     | BA   |
| B7     | On base percentage  | OBP  | B7     | On base percentage  | OBP  |
| B8     | Slugging Percentage | SLG  | B8     | Slugging Percentage | SLG  |
| B9     | On-base plus slugging | OPS | B9     | On-base plus slugging | OPS  |
| B10    | Number of pitches in the PA | Pit | B10 | Number of pitches in the PA | Pit |
| B11    | Strikes             | Str  | B11    | Strikes             | Str  |
| B12    | Putouts             | PO   | B12    | Putouts             | PO   |
| B13    | Assists             | A    | B13    | Assists             | A    |
| SP1    | Innings Pitched     | IP   | P1     | Innings Pitched     | IP   |
| SP2    | Hits                | H    | P2     | Hits                | H    |
| SP3    | Bases on Balls      | BB   | P3     | Bases on Balls      | BB   |
| SP4    | Strikeouts          | SO   | P4     | Strikeouts          | SO   |
| SP5    | Home Runs Hit       | HR   | P5     | Home Runs Hit       | HR   |
| SP6    | Earned Run Average  | ERA  | P6     | Earned Run Average  | ERA  |
| SP7    | Batters Faced       | BF   | P7     | Batters Faced       | BF   |
| SP8    | Number of pitches in the PA | Pit | P8 | Number of pitches in the PA | Pit |
| SP9    | Strikes             | Str  | P9     | Strikes             | Str  |
| SP10   | Strikes by Contact  | Cct  | P10    | Strikes by Contact  | Cct  |
| SP11   | Strikes Swinging    | StS  | P11    | Strikes Swinging    | StS  |
| SP12   | Strikes Looking     | StL  | P12    | Strikes Looking     | StL  |
| SP13   | Ground Balls        | GB   | P13    | Ground Balls        | GB   |
| SP14   | Fly Balls           | FB   | P14    | Fly Balls           | FB   |
| SP15   | Line Drives         | LD   | P15    | Line Drives         | LD   |
| SP16   | Game Score          | GSc  | P16    | Game Score          | GSc  |
|        |                     |      | P17    | Inherited Runners   | IR   |
|        |                     |      | P18    | Inherited Score     | IS   |
| else   | X1                   | H/A  | X1     | Home/Away Team      | H/A  |
Table 2. MLB variable table (dataset 1).

| Game | Variable | B1 | B2 | B3 | B4 | SP1 | SP2 | SP3 | SP4 | X1 | Y |
|------|----------|----|----|----|----|-----|-----|-----|-----|----|---|
| 1    |          | 31 | 5  | 1  | 14 | 7.2 | 2   | 3   | 12  | 1  | 0 |
| 2    |          | 36 | 9  | 3  | 11 | 6   | 7   | 2   | 8   | 1  | 0 |
| 4857 |          | 26 | 4  | 2  | 11 | 6   | 2   | 5   | 7   | 1  | 1 |
| 4858 |          | 44 | 15 | 2  | 13 | 7   | 5   | 2   | 9   | 1  | 1 |

Before inputting the selected variables into the prediction model, min–max normalization was employed to eliminate the relative variance of each variable and to establish a benchmark for subsequent analysis. All selected variables were assigned a value between 0 and 1 according to Equation (1).

\[ X_{\text{nom}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \in [0, 1] \]  

where \(X_{\text{max}}\) and \(X_{\text{min}}\) are the maximum and minimum values in the data, respectively.

3.3. Feature Selection

The main objective of feature selection is to reduce the number of features to increase the prediction accuracy of machine learning. Feature selection methods are classified as wrapper, filter, and embedding methods. ReliefF, which is a filter method, was used as the feature selection method in this study. In the Weka interface, ReliefF is represented by the term ReliefFAttributeEval.

The Relief algorithm was first developed by Kira and Rendell [28] for use with discrete and continuous attributes; however, it could only be used for solving two-group classification problems. In 1994, Kononenko [29] then proposed an extension of the Relief algorithm called ReliefF for use with noisy or incomplete data; this algorithm could be applied to multi-classification problems and regression problems. ReliefF repeatedly obtains samples from the training set and calculates the difference between instances to find the nearest neighbors. The Algorithm 1 ReliefF is as follows.

Algorithm 1. ReliefF

\[ \text{Input: for each training instance a vector of attribute values and the class value} \]
\[ \text{Output: the vector W of estimations of qualities of attributes} \]
1. Set all weighs \(W[A] := 0.0;\)
2. \(\text{for } i := 1 \text{ to } m \text{ do begin} \)
3. randomly select an instance \(R_i;\)
4. find \(k\) nearest hits \(H_j;\)
5. \(\text{for each class } C \neq \text{ class}(R_i) \text{ do} \)
6. from class \(C\) find \(k\) nearest misses \(M_j(C);\)
7. \(\text{for } A := 1 \text{ to } a \text{ do} \)
8. \(W[A] := W[A] - \sum_{j=1}^{k} \text{diff}(A, R_i, H_j) / (m \cdot k) + \)
9. \(\sum_{C \in \text{class}(R_i)} \left[ \frac{p(C)}{1 - p(\text{Class}(R_i))} \sum_{j=1}^{k} \text{diff}(A, R_i, M_j(C)) \right] / (m \cdot k) \)
10. \(\text{end;} \)

where \(W[A]\) represents the weight of feature \(A, m\) represents the number of cycles of random training examples, \(R_i\) represents a randomly selected example, \(H_j\) (referred to as the nearest hits) represents the search for \(k\)-nearest neighbors from the same class, \(M_j(C)\) (referred to as the nearest misses) represents the search for \(k\)-nearest neighbors from each category, \(p(C)\) represents the proportion of the category, and \(p(\text{Class}(R_i))\) represents the proportion of randomly selected instances belonging to each class. \(W[A]\) is updated according to the values of \(R_i, H_j,\) and \(M_j(C).\) Finally, the appropriate features are selected
according to the weights. The basic difference between ReliefF and Relief is the selection of k hits and misses; moreover, compared with Relief, ReliefF is more robust against noise.

3.4. Prediction Model Construction

In this study, three prediction models—a 1DCNN, an ANN, and an SVM—were constructed in Python. First, the original variables, that is, the variables prior to feature selection, were input into the 1DCNN, ANN, and SVM. The variables after feature selection were then input into the ANN and SVM. Fivefold cross-validation was employed to determine the prediction accuracy of the models.

3.4.1. 1DCNN

A 1DCNN is suitable for analyzing signal and time-series data. A 1DCNN, 2DCNN, and 3DCNN all have the same characteristics and follow the same processing method. The main difference lies in the dimensions of the input data and the manner in which the convolution kernel (also known as the filter) moves within the data. For example, the convolution kernel of the 2DCNN moves along two directions, and the input and output data are 3D. However, the convolution kernel of the 1DCNN moves in one direction, and the input and output data are 2D. A CNN comprises convolutional layers, pooling layers, and fully connected layers. These layers are constructed differently according to the CNN architecture.

Commonly used CNN architectures such as AlexNet and GoogleNet have well-established network architectures that can be easily applied to or modified for 2D or 3D data; however, a 1DCNN has no established network structure. The network structure of a 1DCNN is built and modified according to the type and size of the data.

We used Keras in Python to construct a 1DCNN model (see Table 3 and Figure 5, taking data set 2 as an example). There were eight layers in total: a 1D convolutional layer, a maximum pooling layer, a 1D convolutional layer, another maximum pooling layer, a dropout layer, a fully connected layer, another dropout layer, and an output layer. The sigmoid activation function was used for solving binary classification problems. The equation is as follows:

$$f(x) = \frac{1}{1 + e^{-x}}$$  \hspace{1cm} (2)

The parameter settings are listed in Table 3. The number of convolution kernels (filters) were set to 16 and 32 in the two 1D convolutional layers; the convolution kernel size was set to 3; the maximum pooling layer window size was 2; the strides were all set to 1; the padding was set to the same value, which means that the input and output data are of the same size; and the dropout was set to 0.2.

Optimization in neural network learning implies determining the ideal combination of parameters with minimal loss. Various optimizers, such as stochastic gradient descent (SGD), Adam, and RMSprop, have been developed. The learning speed of the SGD optimizer is low. Therefore, we used Adam and RMSprop with the grid search method (GridSearchCV) in Python to determine the most suitable optimizer for this model, and the loss function was set to binary_crossentropy to deal with the problem of two-group classification. The batch size was set to 10, 20, and 30, and the epochs were 50, 100, and 150 with fivefold cross-validation.
### Table 3. 1DCNN model architecture (take dataset 2 as an example).

| Layer (Type) | Output Shape | Filters | Kernel_Size | Strides |
|--------------|--------------|---------|-------------|---------|
| Input size   | 32 × 4858    | -       | -           | -       |
| conv1d_1     | 32 × 16      | 16      | 3           | 1       |
| maxpooling1d_1 | 31 × 16    | -       | 2           | 1       |
| conv1d_2     | 31 × 32      | 32      | 3           | 1       |
| maxpooling1d_2 | 15 × 32    | -       | 2           | 1       |
| Dropout      | 15 × 32      | -       | -           | -       |
| Flatten      | 480          | -       | -           | -       |
| Dense(ReLu)  | 50           | -       | -           | -       |
| Dropout      | 50           | -       | -           | -       |
| Sigmoid      | 1            | -       | -           | -       |

### Figure 5. 1DCNN model architecture diagram (take dataset 2 as an example).

3.4.2. ANN

ANNs are some of the most commonly used models for motion prediction in machine learning [30]. A basic ANN comprises an input layer, a hidden layer, and an output layer. The number of hidden layers depend on the complexity of the problem. The higher the number of hidden layers, the slower the learning rate. In [7,8], one hidden layer was used to construct a model for predicting the outcomes of MLB matches. In this study, because a match result is classified as nonlinear and binary (win/loss), the sigmoid function was used as the excitation function of the neuron in the input layer.

In this study, an ANN prediction model was constructed in Python. For data set 1 and data set 2, the input variables in the input layer were 30 and 32, respectively, and the number of neurons in the hidden layer was set to 16 and 17, respectively; the output layer was the game win or game loss (1/0). A dropout layer was added after the hidden layer to prevent overfitting; the dropout value was set to 0.1; the kernel_initializer, which is the initialization method, was placed in the input layer and output layer. The initialization methods include Zeros, Ones, RandomNormal, glorot_normal, and he_normal. We used cross-validation in Python along with GridSearchCV to determine the most suitable initialization method for this model; the sigmoid function was set as the activation function for neurons in the output layer.

The optimizer setting was the same as that in the 1DCNN. Adam and RMSprop were employed with GridSearchCV to determine the most suitable optimizer for this model. The loss function was set to binary_crossentropy, the batch_size was set to 10, 20, and 30, and the epochs were set to 50, 100, and 150 with fivefold cross-validation.
3.4.3. SVM

An SVM, which is based on statistical learning theory, is a supervised machine learning model that was developed in the 1990s [31]. It is used to solve problems such as binary classification, multi-classification, and regression problems. An SVM constructs a hyperplane as a decision boundary to between two categories and establish a clear distinction.

Selecting kernel functions and setting hyperparameters are important steps. Common kernels include the linear kernel, Gaussian RBF kernel, RBF kernel, and polynomial kernel. We employed the most basic linear kernel and the popular nonlinear RBF kernel [32] to construct the prediction models. The regularization parameter (C) affects the performance of the linear-SVM; the regularization parameter (C) and gamma value affect the performance of an RBF-SVM. The C value evaluates the trade-off between the classification accuracy of the training sample and the maximization of the decision function boundary. The higher the C value, the better the decision function can classify all training samples with a smaller margin. If the C value is low, then a larger margin is acceptable, which simplifies the decision-making function; however, the training accuracy is decreased. If the gamma value is too high, then overfitting may occur; however, if it is too low, then the model cannot capture the complexity of the data [33].

We used GridSearchCV in Python to set parameters for the collected data set. Table 4 lists the accuracies obtained with the linear kernel (for various C values of 1, 10, 100, and 1000) and the RBF kernel (for gamma values set to 0.0001, 0.001, 0.1, 1, 10, 100, and 1000 and C values set to 1, 10, 100, and 1000). Fivefold cross-validation was performed. The table indicates the ideal parameter setting for data set 2. The RBF-SVM with a C value of 1000 and gamma value of 0.1 attained the highest accuracy (93.9%).

Table 4. SVM GridSearchCV-dataset 2.

| Kernel | C   | Gamma | Accuracy |
|--------|-----|-------|----------|
| Linear | 1   | -     | 91.1%    |
|        | 10  | -     | 92.7%    |
|        | 100 | -     | 93.1%    |
|        | 1000| -     | 93.6%    |
| RBF    | 1   | 1000  | 50.1%    |
|        | 1   | 100   | 50.6%    |
|        | 1000| 0.1   | 93.9%    |
|        | 1000| 0.01  | 93.3%    |
|        | 1000| 0.001 | 91.7%    |

Best combination of parameters = {'kernel': 'RBF', 'C': 1000, 'gamma': 0.1}

3.5. Performance Evaluation Index

A binary confusion matrix was generated in the win–loss prediction, and the evaluation index was derived on the basis of the actual results and predicted results. True positive (TP) indicates that a win has been correctly predicted as a win, false negative (FN) indicates that a win has been incorrectly predicted as a loss, false positive (FP) indicates that a loss has been incorrectly predicted as a win, and true negative (TN) indicates that a loss has been correctly predicted as a loss (Table 5). The accuracy was calculated using Equation (3).

\[
\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \quad (3)
\]
Table 5. Binary Confusion Matrix

| Actual results | Win | Lose | True Positive (TP) | False Negative (FN) | True Positive (TP) | False Negative (FN) |
|----------------|-----|------|---------------------|---------------------|---------------------|---------------------|
|                | Win | Lose |                     |                     |                     |                     |
|                | True Positive (TP) | False Negative (FN) |                     |                     |                     |                     |
|                | False Positive (FP) | True Negative (TN) |                     |                     |                     |                     |

4. Results

We used Weka to perform feature selection of the team data variables and used Python to construct the deep learning models and machine learning classifiers. The following sections present the prediction results obtained using the 1DCNN, ANN, and SVM models.

4.1. Results Obtained without Feature Selection: Data Set 1

Data set 1 contains 30 variables. We first normalized the data and then divided the data set into a training data set (80% of the data; 3886 datapoints) and a testing data set (20% of the data; 972 datapoints) with fivefold cross-validation.

4.1.1. 1DCNN

The confusion matrices for the fivefold cross-validation of the 1DCNN are presented in Table 6. The average accuracy rate was 91.44%. Take the confusion matrix with CV = 1 as an example; among the outcomes of the 972 matches, 482 matches were correctly predicted wins, 30 matches were incorrectly predicted losses, 396 matches were correctly predicted losses, and 64 matches were incorrectly predicted wins. A total of 879 matches were predicted correctly and 94 matches were predicted incorrectly, yielding an accuracy rate of 90.33%.

Table 6. Confusion matrix of 1DCNN (dataset 1).

| CV | Win | Lose | Accuracy |
|----|-----|------|----------|
| 1  | 482 | 30   | 90.33%   |
| 2  | 475 | 37   | 92.28%   |
| 3  | 462 | 50   | 91.87%   |
| 4  | 466 | 46   | 90.95%   |
| 5  | 469 | 43   | 91.77%   |
| Average | 469 | 43 | 91.44% |

4.1.2. ANN

The hidden layer is set to 16 with GridSearchCV; kernel_initializer is set to glorot_uniform; optimizer is RMSprop; the epochs are 150; the batch_size is 10, and the confusion matrices for fivefold cross-validations of ANN are shown in Table 7. The average accuracy rate is 92.02%. Take the confusion matrix with CV = 1 as an example; among the outcomes of the 972 matches, 446 matches were correctly predicted wins, 66 matches were incorrectly predicted losses, 440 matches were correctly predicted losses, and 20 matches were incorrectly predicted wins. A total of 886 matches were predicted correctly and 86 matches were predicted incorrectly, yielding an accuracy rate of 91.15%.
4.1.3. SVM

The confusion matrices for fivefold cross-validations of SVM are shown in Table 8. The classification accuracies of the test set (972 matches) are 91.26%, 90.65%, 91.56%, 92.08%, and 91.26%, respectively. The average accuracy rate is 91.36% using the RBF-SVM with C setting of 100 and gamma of 0.1 for dataset 1. Take the confusion matrix with CV = 1 as an example: among the outcomes of the 972 matches, 432 matches were correctly predicted wins, 51 matches were incorrectly predicted losses, 455 matches were correctly predicted losses, and 34 matches were incorrectly predicted wins. A total of 887 matches were predicted correctly and 85 matches were predicted incorrectly, yielding an accuracy rate of 91.15%.

Table 7. Confusion matrix of ANN without feature selection (dataset 1).

| Prediction Results | CV | Win | Lose | Accuracy |
|--------------------|----|-----|------|----------|
| Actual results     | 1  | Win | 446  | 66       | 91.15%   |
|                    |    | Lose| 20   | 440      |          |
|                    | 2  | Win | 440  | 72       | 90.64%   |
|                    |    | Lose| 19   | 441      |          |
|                    | 3  | Win | 481  | 31       | 92.49%   |
|                    |    | Lose| 42   | 418      |          |
|                    | 4  | Win | 485  | 27       | 92.59%   |
|                    |    | Lose| 45   | 415      |          |
|                    | 5  | Win | 473  | 39       | 93.21%   |
|                |    | Lose| 27   | 433      |          |
| Average           |    |     |      |          | 92.02%   |

Table 8. Confusion matrix of SVM without feature selection (dataset 1).

| Prediction Results | CV | Win | Lose | Accuracy |
|--------------------|----|-----|------|----------|
| Actual results     | 1  | Win | 432  | 51       | 91.26%   |
|                    |    | Lose| 34   | 455      |          |
|                    | 2  | Win | 447  | 52       | 90.65%   |
|                    |    | Lose| 36   | 437      |          |
|                    | 3  | Win | 439  | 52       | 91.56%   |
|                    |    | Lose| 36   | 437      |          |
|                    | 4  | Win | 439  | 47       | 92.08%   |
|                    |    | Lose| 35   | 451      |          |
|                    | 5  | Win | 459  | 43       | 91.26%   |
|                |    | Lose| 34   | 436      |          |
| Average           |    |     |      |          | 91.36%   |

4.2. Results Obtained with Feature Selection: Data Set 1

With average merit > 0, 10 variables B1, B5, B6, B7, B8, B9, B10, B12, SP16, and X1 are selected from the original 30 variables for ANN and SVM prediction models for dataset 1.

4.2.1. ANN

The hidden layer is set to 16 with GridSearchCV; kernel_initializer is set to glorot_uniform; optimizer is Adam; the epochs are 150; the batch_size is 10, and the confusion matrices for fivefold cross-validations of ANN are shown in Table 9. The average accuracy rate is 92.06%. Take the confusion matrix with CV = 1 as an example; among the outcomes
of the 972 matches, 475 matches were correctly predicted wins, 37 matches were incorrectly predicted losses, 420 matches were correctly predicted losses, and 40 matches were incorrectly predicted wins. A total of 895 matches were predicted correctly and 77 matches were predicted incorrectly, yielding an accuracy rate of 92.08%.

4.2.2. SVM

The average accuracy rate is 91.94% using the RBF-SVM with C setting of 1000 and gamma of 0.1 for dataset 1. The confusion matrices for fivefold cross-validations are shown in Table 10. The classification accuracies of the test set (972 matches) are 90.74%, 91.16%, 92.28%, 91.36% and 94.14%, respectively. Take the confusion matrix with CV = 1 as an example; among the outcomes of the 972 matches, 444 matches were correctly predicted wins, 43 matches were incorrectly predicted losses, 438 matches were correctly predicted losses, and 47 matches were incorrectly predicted wins. A total of 882 matches were predicted correctly and 90 matches were predicted incorrectly, yielding an accuracy rate of 90.74%.

Table 9. Confusion matrix of ANN after feature selection (dataset 1).

| CV | Win | Lose | Accuracy  |
|----|-----|------|-----------|
| 1  | 475 | 37   | 92.08%    |
| 2  | 450 | 59   | 90.43%    |
| 3  | 461 | 51   | 92.80%    |
| 4  | 467 | 45   | 92.90%    |
| 5  | 461 | 51   | 92.08%    |
|    |     |      | Average   |
|    |     |      | 92.06%    |

Table 10. Confusion matrix of SVM after feature selection (dataset 1).

| CV | Win | Lose | Accuracy  |
|----|-----|------|-----------|
| 1  | 444 | 43   | 90.74%    |
| 2  | 450 | 56   | 91.16%    |
| 3  | 448 | 43   | 92.28%    |
| 4  | 448 | 55   | 91.36%    |
| 5  | 462 | 33   | 94.14%    |
|    |     |      | Average   |
|    |     |      | 91.94%    |

4.3. Results Obtained without Feature Selection: Data Set 1

Data set 2 contains 32 variables. We first normalized the data and then divided the data set into a training data set (80% of the data; 3886 datapoints) and a testing data set (20% of the data; 972 datapoints) with fivefold cross-validation.
4.3.1. 1DCNN

The confusion matrixes for fivefold cross-validations of 1DCNN are shown in Table 11. The average accuracy rate is 93.40% using optimizer Rmsprop; epoch is set at 100; batch_size is 30. Take the confusion matrix with CV = 1 as an example; among the outcomes of the 972 matches, 483 matches were correctly predicted wins, 29 matches were incorrectly predicted losses, 427 matches were correctly predicted losses, and 33 matches were incorrectly predicted wins. A total of 910 matches were predicted correctly and 62 matches were predicted incorrectly, yielding an accuracy rate of 93.62%.

Table 11. Confusion matrix of 1DCNN (dataset 2).

| Prediction Results | \( \text{CV} \) | \( \text{Win} \) | \( \text{Lose} \) | Accuracy |
|--------------------|---------------|---------------|---------------|----------|
| \( \text{Actual results} \) | 1 | Win | 483 | 33 | \( \text{Win} \) | 29 | 427 | 93.62% |
| | 2 | Win | 469 | 23 | \( \text{Lose} \) | 43 | 437 | 93.21% |
| | 3 | Win | 472 | 21 | \( \text{Lose} \) | 40 | 439 | 93.72% |
| | 4 | Win | 476 | 32 | \( \text{Lose} \) | 36 | 428 | 93.00% |
| | 5 | Win | 468 | 20 | \( \text{Lose} \) | 44 | 440 | 93.42% |
| \( \text{Average} \) & | | | | 93.40% |

4.3.2. ANN

The hidden layer is set to 17 with GridSearchCV; kernel_initializer is set to he_uniform; optimizer is adam; the epochs are 150; the batch_size is 10, and the confusion matrixes for fivefold cross-validations of ANN are shown in Table 12. The average accuracy rate is 93.91%. Take the confusion matrix with CV = 1 as an example; among the outcomes of the 972 matches, 481 matches were correctly predicted wins, 31 matches were incorrectly predicted losses, 435 matches were correctly predicted losses, and 25 matches were incorrectly predicted wins. A total of 916 matches were predicted correctly and 56 matches were predicted incorrectly, yielding an accuracy rate of 94.24%.

Table 12. Confusion matrix of ANN without feature selection (dataset 2).

| Prediction Results | \( \text{CV} \) | \( \text{Win} \) | \( \text{Lose} \) | Accuracy |
|--------------------|---------------|---------------|---------------|----------|
| \( \text{Actual results} \) | 1 | Win | 481 | 25 | \( \text{Win} \) | 31 | 435 | 94.24% |
| | 2 | Win | 480 | 30 | \( \text{Lose} \) | 32 | 430 | 93.62% |
| | 3 | Win | 465 | 20 | \( \text{Lose} \) | 47 | 440 | 93.11% |
| | 4 | Win | 478 | 19 | \( \text{Lose} \) | 34 | 441 | 94.55% |
| | 5 | Win | 481 | 27 | \( \text{Lose} \) | 31 | 433 | 94.03% |
| \( \text{Average} \) & | | | | 93.91% |
4.3.3. SVM

The average accuracy rate is 93.81% using the RBF-SVM with C setting of 1000 and gamma of 0.1 for dataset 2, and the confusion matrices for fivefold cross-validations of SVM are shown in Table 13. The classification accuracies of the test set (972 matches) are 93.42%, 95.16%, 93.11%, 94.03% and 93.33%, respectively. Taking the confusion matrix with CV = 1 as an example, it can be known that among the 972 matches, 440 matches were successfully predicted to win, 27 matches were misjudged; 468 matches were successfully predicted to lose, 37 matches were misjudged. A total of 908 matches were predicted correctly, and 64 matches were predicted incorrectly, giving an accuracy rate of 93.42%. Take the confusion matrix with CV = 1 as an example; among the outcomes of the 972 matches, 451 matches were correctly predicted wins, 26 matches were incorrectly predicted losses, 474 matches were correctly predicted losses, and 37 matches were incorrectly predicted wins. A total of 908 matches were predicted correctly and 64 matches were predicted incorrectly, yielding an accuracy rate of 93.42%.

Table 13. Confusion matrix of SVM without feature selection (dataset 2).

| CV | Win | Lose | Accuracy |
|----|-----|------|----------|
| 1  | 440 | 27   | 93.42%   |
| 2  | 451 | 21   | 95.16%   |
| 3  | 423 | 25   | 93.11%   |
| 4  | 457 | 35   | 94.03%   |
| 5  | 438 | 27   | 93.33%   |

4.4. Results Obtained with Feature Selection: Data Set 2

With average merit > 0, 25 variables (except B4, B13, P4, P11, P12, P13, and P15) are selected from the original 32 variables for ANN and SVM prediction models for dataset 2.

4.4.1. ANN

The hidden layer is set to 16 with GridSearchCV; kernel_initializer is set to glorot_uniform; optimizer is Adam; the epochs are 150; the batch_size is 10, and the confusion matrices for fivefold cross-validations of ANN are shown in Table 14. The average accuracy rate is 94.18%. Take the confusion matrix with CV = 1 as an example; among the outcomes of the 972 matches, 474 matches were correctly predicted wins, 38 matches were incorrectly predicted losses, 441 matches were correctly predicted losses, and 19 matches were incorrectly predicted wins. A total of 915 matches were predicted correctly and 57 matches were predicted incorrectly, yielding an accuracy rate of 94.14%.

4.4.2. SVM

The average accuracy rate is 94.16% using the RBF-SVM with C setting of 1000 and gamma of 0.1 for dataset 2. The confusion matrices for fivefold cross-validations of SVM are shown in Table 15. The classification accuracies of the test set (972 matches) are 94.65%, 94.03%, 93.93%, 94.65% and 93.52%, respectively. Take the confusion matrix with CV = 1 as an example; among the outcomes of the 972 matches, 458 matches were correctly predicted wins, 24 matches were incorrectly predicted losses, 462 matches were correctly predicted losses, and 28 matches were incorrectly predicted wins. A total of 920 matches were
predicted correctly and 52 matches were predicted incorrectly, yielding an accuracy rate of 94.65%.

Table 14. Confusion matrix of ANN after feature selection (dataset 2).

| Actual results | Prediction Results | CV | Win | Lose | Accuracy |
|----------------|--------------------|----|-----|------|----------|
| Win            | Lose               | 1  | 474 | 19   | 38       | 94.14%   |
| Lose           |                    | 2  | 483 | 29   | 29       | 94.03%   |
| Win            | Lose               | 3  | 478 | 18   | 34       | 94.65%   |
| Lose           |                    | 4  | 481 | 26   | 31       | 94.14%   |
| Win            | Lose               | 5  | 478 | 25   | 34       | 93.93%   |

Average 94.18%

Table 15. Confusion matrix of SVM after feature selection (dataset 2).

| Actual results | Prediction Results | CV | Win | Lose | Accuracy |
|----------------|--------------------|----|-----|------|----------|
| Win            | Lose               | 1  | 458 | 28   | 24       | 94.65%   |
| Lose           |                    | 2  | 444 | 34   | 24       | 94.03%   |
| Win            | Lose               | 3  | 473 | 34   | 37       | 93.93%   |
| Lose           |                    | 4  | 475 | 32   | 20       | 94.65%   |
| Win            | Lose               | 5  | 456 | 34   | 29       | 93.52%   |

Average 94.16%

4.5. Performance Comparison of the Prediction Models

Table 16 lists the prediction results, before and after feature extraction, obtained from inputting the two data sets into the three prediction models.

For data set 1, the prediction accuracies of the ANN were 92.02% and 92.06% before and after feature extraction, respectively. The prediction accuracies of the SVM were 91.36% and 91.94% before and after feature extraction, respectively. The prediction accuracy of the 1DCNN was 91.44% for data set 1. Although the accuracy of the 1DCNN was slightly lower than the accuracies of the ANN and SVM after feature selection, the prediction performance of the 1DCNN was slightly higher than that of the SVM before feature selection.

For data set 2, the prediction accuracies of the ANN were 93.91% and 93.99% before and after feature extraction, respectively. The prediction accuracies of the SVM were 93.90% and 94.16% before and after feature extraction, respectively. The prediction accuracy of the 1DCNN was 93.4% for data set 2. The three models exhibited similar prediction performance.

The prediction accuracies obtained with data set 2 were higher for all models, which means that using the data of all pitchers is better than using the data of only the SP. Although the performance of the SP during a game is very important, the performance of the RP also must be considered to obtain more complete game information and thus
more accurate prediction of the outcomes of matches. This trend was also observed for the variables after feature selection. Ten variables (AB, PA, BA, OBP, SLG, OPS, Pit, PO, GSc, H/A) were selected for data set 1, in which only one variable was related to the first pitcher (GSc). Of the 25 variables selected for data set 2 (AB, H, BB, PA, BA, OBP, SLG, OPS, Pit, Str, PO, IP, H, BB, HR, ERA, BF, Pit, Str, Ctxt, FB, GSc, IR, IS, H/A), 13 were related to all pitchers. The use of all pitcher game data leads to more accurate predictions of game outcomes than the use of only SP data.

Table 16. The prediction results of the three models.

| Dataset     | Model | No Feature Selection | Feature Selection |
|-------------|-------|----------------------|-------------------|
| Dataset 1   | 1DCNN | 91.44%               |                   |
|             | ANN   | 92.02%               | 92.06%            |
|             | SVM   | 91.36%               | 91.94%            |
| Dataset 2   | 1DCNN | 93.4%                |                   |
|             | ANN   | 93.91%               | 94.18%            |
|             | SVM   | 93.90%               | 94.16%            |

4.6. Discussion

Because studies on the outcome prediction of MLB matches have employed data from different years and over different periods to make predictions, comparing the prediction accuracies reported in different studies may be inappropriate. Moreover, the selection of match variables for inclusion in prediction models affects the prediction results. Suitable variables should be selected into a prediction model to achieve satisfactory model performance. Jia et al. [18] collected game data on variables related to scoring in three categories: hitter, pitcher, and team. After feature selection, they obtained 7 variables from the original 14 variables and achieved a prediction accuracy of 59.6%. Soto Valero [8] collected competition data from two websites. Because the items recorded by each website were slightly different, more research variables needed to be collected from more data sets. In [8], 15 variables were obtained after feature selection, and the prediction accuracy was 59%. Chen [7] collected 60 variables for hitters, SPs, match time, and temperature (°F) and then selected four variables for inclusion in their prediction model. Elfrink [9] obtained data pertaining to five categories: game time (day/night), home/away team, baseball field, competing team, and day of the week; Elfrink expanded the number of variables collected for match outcome prediction and achieved relatively low accuracy (55.52%). In our study, we collected data on hitters, pitchers, and the variable of whether the team was the home or away team and achieved high prediction accuracies.

Studies have used machine learning methods to predict the outcomes of MLB matches. The prediction accuracies obtained by Jia et al. [18], Chen [7], and Soto Valero [8] were 59.6% (SVM), 73.68% (ANN), and 59% (SVM), respectively (Table 17). The ANNs and SVMs developed in the aforementioned studies outperformed other methods. For data set 2, our ANN model exhibited the highest prediction accuracy (94.18%), followed by the SVM (94.16%) and the 1DCNN (93.4%).
### Table 17. Comparison of prediction models for MLB matches.

| Literature               | Input Variables (after Feature Selection)            | Method *                  | Accuracy |
|--------------------------|------------------------------------------------------|---------------------------|----------|
| Jia et al. [18]          | BA, RBI, OBP, ERA, H, E, and Win% for each team      | Random forest             | 59.60%   |
|                          |                                                      | SVM                       |          |
|                          |                                                      | AdaBoost                  |          |
|                          |                                                      | LogitBoost                |          |
|                          |                                                      | MLR                       |          |
| Chen [7]                 | SO(A), GSc(H), BB(A), ER(A)                          | Logistic regression       | 73.68%   |
|                          | SO(H), BB(A), SO(A), WHIP(H) **                      | ANN                       |          |
| Soto Valero [8]          | isHomeClub, Log5, PE, WP, RC, HomeWonPrev,          | SVM                       | 59.00%   |
|                          | VisitorWonPrev, BABIP, FP, PitchERA, OBP, SLG,       | ANN                       |          |
|                          | VisitorLeague, HomeVersusVisitor, Stolen             | Decision Tree             |          |
|                          |                                                      | Lazy Learners(KNN)        |          |
| Elfrink [9]              | AB, AVG, OBP, SLG, OPS, BA/RISP, WHIP, RA           | Random forest             | 55.52%   |
|                          |                                                      | Linear model              |          |
|                          |                                                      | XGBoost                   |          |
| This study               | AB, H, BB, PA, BA, OBP, SLG, OPS, Pit, Str, PO, IP, H, | 1D-CNN                    | 94.18%   |
|                          | BB, HR, ERA, BF, Pit, Str, Ctct, FB, Gsc, IR, IS, H/A | ANN                       |          |
|                          |                                                      | SVM                       |          |

*: Boldface represents the best prediction performance in this research; **: H represents the home team, and A represents the away team.

A major difference between our study and previous studies is the data source used for constructing the prediction models. We collected data from Baseball-Reference to establish two data sets and investigated whether selecting data with only the SP or with all pitchers in the pitcher category had an effect on the prediction accuracy. In addition, the prediction performance of the models before and after feature selection was compared. The results of this study indicate that including the data of all pitchers (data set 2) yields more accurate predictions of the outcomes of MLB matches. Although the performance of the SP is very important and is often used as an indicator of whether a game may be lost or won, the performance of the RP can sometimes be a decisive factor influencing the outcome of the game. During feature selection for data set 1, only one game score (GSc) variable was selected from the SP data. It is clear that more complete game information on all pitchers would increase the accuracy of match outcome prediction. The prediction accuracies of the three prediction models for MLB matches proposed in this study were higher than 90%, which is considerably higher than the prediction accuracies obtained in related studies (accuracies of 55–74%).

To the best of our knowledge, this is the first study to employ a 1DCNN model to predict the outcomes of MLB matches, and the results indicate that the model achieved favorable prediction performance. Although the ANN and SVM models outperformed the 1DCNN model, the 1DCNN model can automatically extract features, thereby avoiding the time-consuming process of feature extraction.

### 5. Conclusions and Suggestions

We predicted the outcomes of MLB matches by collecting the match data of 30 teams in the 2019 season and using 1DCNN, ANN, and SVM prediction models. The prediction accuracies of these models were then compared. We investigated whether the selection of input variables and the characteristics before and after feature selection had an impact on the prediction accuracy. The prediction results indicated that 1DCNN, ANN, and SVM models achieved higher prediction accuracies when the data of all pitchers were considered than when only the data of the SP were considered.

The three prediction models proposed in this study all achieved high prediction performance and can thus be used to provide some reference information for fans, team managers, and baseball enthusiasts. The proposed prediction models can also be used to...
predict game outcomes for other sports; however, their prediction performance cannot be verified without experimental simulation.

The contributions of this study are as follows:

1. This is the first study to use a 1DCNN model to predict the outcomes of MLB matches.
2. Unlike machine learning models, the 1DCNN model automatically extracts features.
3. This study established two data sets to compare the accuracy of match outcome prediction when only data on the SP were considered with that when data on all the pitchers were considered. The prediction results demonstrate that using the game data of all pitchers yields higher accuracy.
4. This study used data of hitters and pitchers from Baseball-Reference.com, and the three proposed models outperformed models developed in related studies in terms of prediction accuracy for the outcomes of MLB matches.

The limitations of this study are as follows:

1. Most related studies have collected more than 5 years of match data to construct prediction models, whereas this study only used data from 2019. Thus, the amount of data was relatively small.
2. Only one feature selection method (ReliefF) was used in this study; thus, the performance of the prediction models when other feature selection methods are used remains unknown. Different feature selection methods should be used to verify the performance of the proposed models.

The adjustments or modifications of the CNN model are encouraged to improve its prediction accuracy. In addition, we selected only technical variables to develop the prediction models. Future research should consider including data on tactical strategies or measures of individual player performance (such as ball speed) to improve the prediction accuracy of MLB match outcomes.

Author Contributions: Conceptualization, M.-L.H. and Y.-Z.L.; Methodology, M.-L.H. and Y.-Z.L.; Validation, M.-L.H.; Formal Analysis, M.-L.H. and Y.-Z.L.; Software, Y.-Z.L.; Investigation, M.-L.H. and Y.-Z.L.; Writing—Original Draft Preparation, M.-L.H. and Y.-Z.L.; Writing—Review and Editing, M.-L.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data were obtained from Baseball-Reference https://www.baseball-reference.com/ (accessed on 15 October 2020).

Acknowledgments: This research did not receive any specific grants from funding agencies in the public, commercial, or not-for-profit sectors.

Conflicts of Interest: The authors declare no conflict of interest.

References
1. U.S. Major Sports: Average Attendance 2019 | Statista. Available online: https://www.statista.com/statistics/207458/per-game-attendance-of-major-us-sports-leagues/ (accessed on 2 March 2021).
2. Lim, N.; Pedersen, P.M. Examining Determinants of Sport Event Attendance: A Multilevel Analysis of a Major League Baseball Season. J. Glob. Sport Manag. 2019, 1–18. [CrossRef]
3. Baseball Team Values 2019: Yankees Lead League At $4.6 Billion. Available online: https://www.forbes.com/sites/mikeozanian/2019/04/10/baseball-team-values-2019-yankees-lead-league-at-46-billion/?sh=21b472fe69b2 (accessed on 21 January 2021).
4. Elitzur, R. Data analytics effects in major league baseball. Omega 2020, 90, 102001. [CrossRef]
5. Fialho, G.; Manhães, A.; Teixeira, J.P. Predicting Sports Results with Artificial Intelligence—A Proposal Framework for Soccer Games. Procedia Comput. Sci. 2019, 164, 131–136. [CrossRef]
6. Yang, T.Y.; Swartz, T. A Two-Stage Bayesian Model for Predicting Winners in Major League Baseball. J. Data Sci. 2004, 2, 61–73.
7. Chen, C.-W. Construction of the Winner Predictive Model in Major League Baseball Games: Use of the Artificial Neural Networks. Sport. Exerc. Res. 2014, 16, 167–181. [CrossRef]
8. Soto Valero, C. Predicting win-loss outcomes in MLB regular season games—a comparative study using data mining methods. *Int. J. Comput. Sci. Sport* **2016**, *15*, 91–112. [CrossRef]
9. Elfrink, T. Predicting the Outcomes of MLB Games with a Machine Learning Approach, Vrije Universiteit Amsterdam. Business Analytics Research Paper 2018. Available online: https://beta.vu.nl/nl/Images/werkstuk-elfrink_tcm235-888205.pdf (accessed on 2 May 2021).
10. Chen, M.Y.; Chen, T.H.; Lin, S.H. Using Convolutional Neural Networks to Forecast Sporting Event Results. In *Studies in Computational Intelligence*; Springer: Berlin/Heidelberg, Germany, 2020; Volume 866, pp. 269–285.
11. Albawi, S.; Mohammed, T.A.; Al-Zawi, S. Understanding of a convolutional neural network. In Proceedings of the 2017 International Conference on Engineering and Technology, ICET 2017, Antalya, Turkey, 21–23 August 2017; pp. 1–6.
12. Kiranyaz, S.; Avci, O.; Abdeljaber, O.; Ince, T.; Gabbouj, M.; Inman, D.J. 1D convolutional neural networks and applications: A survey. *Mech. Syst. Signal. Process.* **2021**, *151*, 107398. [CrossRef]
13. Khaki, S.; Wang, L.; Archontoulis, S.V. A CNN-RNN Framework for Crop Yield Prediction. *Front. Plant. Sci.* **2020**, *10*, 1750. [CrossRef]
14. Harbola, S.; Coors, V. One dimensional convolutional neural network architectures for wind prediction. *Energy Convers. Manag.* **2019**, *189*, 70–75. [CrossRef]
15. Mumtaz, W.; Qayyum, A. A deep learning framework for automatic diagnosis of unipolar depression. *Int. J. Med. Inform.* **2019**, *132*, 103983. [CrossRef]
16. Jia, R.; Wong, C.; Zeng, D. Predicting the Major League Baseball Season. CS229 Machine Learning Final Project 2013. pp. 1–5. Available online: http://cs229.stanford.edu/projects2013.html (accessed on 2 May 2021).
17. Harbola, S.; Coors, V. One dimensional convolutional neural network architectures for wind prediction. *Energy Convers. Manag.* **2019**, *189*, 70–75. [CrossRef]
18. Beneventano, P.; Berger, P.D.; Weinberg, B.D. Predicting Run Production and Run Prevention in Baseball: The Impact of Sabermetrics. *Int. J. Bus. Humantit. Technol.* **2012**, *2*, 67–75.
19. Jia, R.; Wong, C.; Zeng, D. Predicting the Major League Baseball Season. CS229 Machine Learning Final Project 2013. pp. 1–5. Available online: http://cs229.stanford.edu/projects2013.html (accessed on 2 May 2021).
20. Tolbert, B.; Trafalis, T. Predicting Major League Baseball Championship Winners through Data Mining. *Athens J. Sport.* **2016**, *3*, 239–252. [CrossRef]
21. Kira, K.; Rendell, L.A. *A Practical Approach to Feature Selection*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1992.
22. Kononenko, I. Estimating attributes: Analysis and extensions of RELIEF. *Lect. Notes Comput. Sci.* **1994**, *784* LNCS, 171–182.
23. Bunker, R.P.; Thabtah, F. A machine learning framework for sport result prediction. *Appl. Comput. Inform.* **2019**, *15*, 51–60. [CrossRef]
24. Cortes, C.; Vapnik, V. Support-Vector Networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
25. Duan, K.; Keerthi, S.; Poo, A. Evaluation of simple performance measures for tuning SVM hyper parameters. *Neurocomputing* **2003**, *51*, 41–59. [CrossRef]
26. RBF SVM Parameters—Scikit-Learn 0.24.1 Documentation. Available online: https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html (accessed on 3 March 2021).