electronics

Article

A Hybrid Machine Learning Model for Predicting USA NBA All-Stars

Alberto Arteta Albert [1,*,†], Luis Fernando de Mingo López [2,†], Kristopher Allbright [1,†] and Nuria Gómez Blas [2,†]

[1] College of Arts and Sciences, Troy University, 129-A MSCX, 600 University Avenue, Troy, AL 36082, USA; kallbright@troy.edu
[2] Escuela Técnica Superior de Ingeniería de Sistemas Informáticos, Universidad Politécnica de Madrid, 28031 Madrid, Spain; fernando.demingo@upm.es (L.F.d.M.L.); uria.gomez.blas@upm.es (N.G.B.)
* Correspondence: aarteta@troy.edu
† These authors contributed equally to this work.

Abstract: Throughout the modern age, sports have been a very important part of human existence. As our documentation of sports has become more advanced, so have the prediction capabilities. Presently, analysts keep track of a massive amount of information about each team, player, coach, and matchup. This collection has led to the development of unparalleled prediction systems with high levels of accuracy. The issue with these prediction systems is that they are proprietary and very costly to maintain. In other words, they are unusable by the average person. Sports, being one of the most heavily analyzed activities on the planet, should be accessible to everyone. In this paper, a preliminary system for using publicly available statistics and open-source methods for predicting NBA All-Stars is introduced and modified to improve the accuracy of the predictions, which reaches values close to 0.9 in raw accuracy, and higher than 0.9 in specificity.

Keywords: sports statistics; sports patterns classification; sports awards; MLP; random forests; adaboost

Citation: Albert, A.A.; de Mingo López, L.F.; Allbright, K.; Gomez Blas, N. A Hybrid Machine Learning Model for Predicting USA NBA All-Stars. *Electronics* **2022**, *11*, 97. https://doi.org/10.3390/electronics11010097

Academic Editors: Heung-Il Suk, Imad Rida, Lunke Fei, Dan Istrate and Amir Mosavi

Received: 22 October 2021
Accepted: 22 December 2021
Published: 29 December 2021

1. Introduction

In the third century BC, the Grecian hero Phayllos recorded a long jump of 16.3 m [1]. This number has been considered inaccurate and the methods to measure it have been questioned considering that the current men's long jump world record stands at 8.95 m [1].

To again put the period into perspective, Phayllos' accomplishments occurred over a century before Alexander the Great's conquests. While the number in the record may be inaccurate, the fact that humans have long been interested in recording the athletic achievements of standout performers is obvious.

As time has gone on, a greater interest in athletics and greater time devotion towards it have developed. This is due to a multitude of factors; however, the leading causes can be attributed to the increase in leisure time in the average person's household, increased societal stability allowing for an upward trending move in Maslow's hierarchy of needs, and the increase in the frequency of as well as access to athletic events. To see the ancient Olympics, a citizen may need to walk or ride for days simply arrive at the event. Now, as the ability to record games in more precise ways has developed, an interest in the outcomes of games becomes more of an interesting piece among the average person. Instead of hearing that Phayllos jumped 16.3 m, we can see a player's full stat line as well as a video recording of the event. Some modern recording methods are used to provide insights into player movements using computer vision [2].

Since the records are correctly verified, athletic feats have been recorded in official rankings. These range from thorough, blow-by-blow accounts of duels to simple kill counts. Records of accomplishments in the Olympic games date back to the first Olympics in 776 BC [3]. At this point, many of the events of the biblical old testament were in full swing [4]. While these records do exist, their precision and accuracy are not optimal and

not comparable with modern sports statistics [5] as the first example described in this section. These records were sometimes exaggerated beyond belief as well. This becomes apparent with the statement that in the 5th be America-centered. In Alan Schwarz's book *The Numbers Game* [6], he highlights the history of baseball's come up as a numbers-focused game. He begins the exploration by talking about how the Olympic Ball Club of Philadelphia, in 1837, mandated that a scorebook be kept to record runs scored by all players. This trend continued up until 1858 when box scores including 11 columns were published in major newspapers from across the United States. By 1883, the American Association was awarding the pitching championship to Tim Keefe because of his earned-run-per-at- bat-ratio [6]. This obscure stat is not present in today's books, but it goes to show how early American sports fans allowed statistics to become a normal part of the sports experience. The father of sports analytics, unsurprisingly, found his start in Baseball. Bill James' background in economics, not sports, lead him to pioneer sabermetrics, or sports analytics. James initially began publishing a column in local newspapers offering statistical answers to odd baseball questions. Later on, in 1977 [7], James began publishing the Bill James Baseball Abstract, in which he studied box scores and created new statistical insights involving the stats [8]. While this revolution was confined to baseball, the ideas came to the public's eye with Billy Beane, the Oakland A's, and the popular book and film, Moneyball. Soon thereafter James' ideas began being adopted by other sports. Since James has risen to relevance in the past decades, his ideas have been adopted, cited, and referenced by everyone from academia to various sports [9]. While this work has become widely influential, James' influence remains the heaviest within the sport of baseball [10].

The performance of any recognition system heavily depends on finding a good and suitable feature representation space where observations from different classes are well separated [11].

In 1981, James Naismith invented the game of basketball as a method of keeping his physical education students occupied indoors during the winter. By 1982, the first public game of basketball was played in Springfield, Massachusetts. At this point, the game only consisted of 13 rules and was a very simple version of what it is today. The first professional league was founded in 1898. Basketball spread outward from the United States to Europe, with the first international game played in 1909. The National Basketball Association (NBA) was formed in 1946, becoming the foundation of the league known today. The NBA has kept stats since its inception but began to step up the game in 1979–1980 when they introduced the three-point line to the game. Soon thereafter, the league began keeping almost all statistics used today. The major exception was the Games Started stat, which began in 1981–1982 [12].

With 1984–1985 came the first gambling odds on the NBA championship. This was a major step forward as oddsmakers were beginning to use Bill James' philosophies, just applied to basketball. Some gambling odds were introduced as late as 2008–2009. As these odds were introduced and as technology improved, the oddsmakers developed ever more sophisticated methods of predicting winners. In the modern, information, era, these oddsmakers collect thousands of points of data on each player in attempts to predict the winners of different titles and awards.

The historical facts described above have shaped the design of the proposed model. The known models used in sport predictions are studied in our proposal and determined the creation the introduced hybrid proposal.

Even though these systems are good starting points to build robust systems to predict winners/all-stars, none of them has been widely recognized to predict successful sports players. That is the main research gap this work aims to address. Our contribution consists of a hybrid system based on known techniques that shows more stable metrics than the ones displayed in the introduction.

2. Current Landscape of Sports Predictions

2.1. Monte Carlo Simulation

As it stands currently, the prediction of sports using machine learning is largely centralized in the gambling field [7], and betting lines are often created by looking at thousands of different relevant data points and by calculating a normal distribution of potential outcomes. This method of using normal distribution as a guideline allows models to simplify down to two variables, mean, and standard deviation. This is a common trick.

A widely used method of developing accurate normal distributions for potential outcomes is called the Monte Carlo Simulation [13]. This is a simulation that is used to predict the likelihood of outcomes where the equations contain random variables. For example, a player being injured is defined as a random variable. There is no method of predicting whether this will occur, so it is best to consider it random for the simulation. The Monte Carlo Simulation works by making guesses about the outcome of these random variables and then by simulating the outcome iteratively. This produces a bell curve that models the likelihood of different outcomes. These Monte Carlo simulations are widely used in the field of sports betting due to their ability to handle unpredictable variables very well and to still produce a model that predicts likely outcomes.

The betting odds are available from many places across the internet, some more transparent than others. One of the most transparent providers of statistical predictions is that of Nate Silver's FiveThirtyEight.com (accessed on 10 October 2021) [14], which provides political odds as well as sports predictions. FiveThirtyEight also uses the Monte Carlo Simulation method mentioned above.

2.2. Tennis

Sascha Wilkens published a paper "Sports Prediction and Betting Models in the Machine Learning Age: The Case of Tennis" [15] discussing the potential for using machine learning models to predict tennis outcomes. She also discusses the possibility of using such a model for sports betting, stating potential strategies. Her results, after studying nearly 39,000 professional tennis matches, were models that predicted an outcome correctly with an overall accuracy equal to 0.7. Her paper has been supremely influential in shaping the methods and results of the research presented in this paper.

2.3. Baseball

Another study published by researchers at Missouri State University discussed predicting the Cy Young Pitching Award winner. This study used a Bayesian classifier to predict this outcome and compared it with other models. When the researchers restricted the dataset to starting pitchers, the model proved near 0.8 of overall accuracy in its predictions.

In 2018, Ryan Pollack of the Hardball Times, a baseball column, undertook the task of attempting to predict the Most Valuable Player (MVP) of each of Major League Baseball's (MLB) two divisions. His implementation was successful, with the confusion matrix shown above. In Ryan's own words:

> "I evaluated the models using precision, recall, and the F1 score. I chose these metrics instead of sensitivity, specificity, and accuracy because the data set is wildly imbalanced. Out of 1517 player-seasons, only 20 (1.3 percent) are classified as having won the MVP, see Table 1. With so little data about what makes an MVP, precision, and recall represent the model's performance better than sensitivity and specificity do." [16].

Table 1. Confusion matrix for MVP predictor model.

	Is ActualMVP	Is Not ActualMVP
Predicted MVP:	13	7
Not predcited MVP:	7	1490

While this approach was moderately successful for Ryan, most researchers have approached the problem by defining target metrics differently [12,17].

2.4. Daniel Bratulić

Daniel Bratulić, a data scientist and writer, tackled the NBA's Most Valuable Player (MVP) prediction problem in 2018. Bratulić's work is reminiscent of Bill James' work in baseball as he worked to answer specific questions using basketball statistics, see Figure 1. The system he developed is based on the NBA MVP voting system [18].

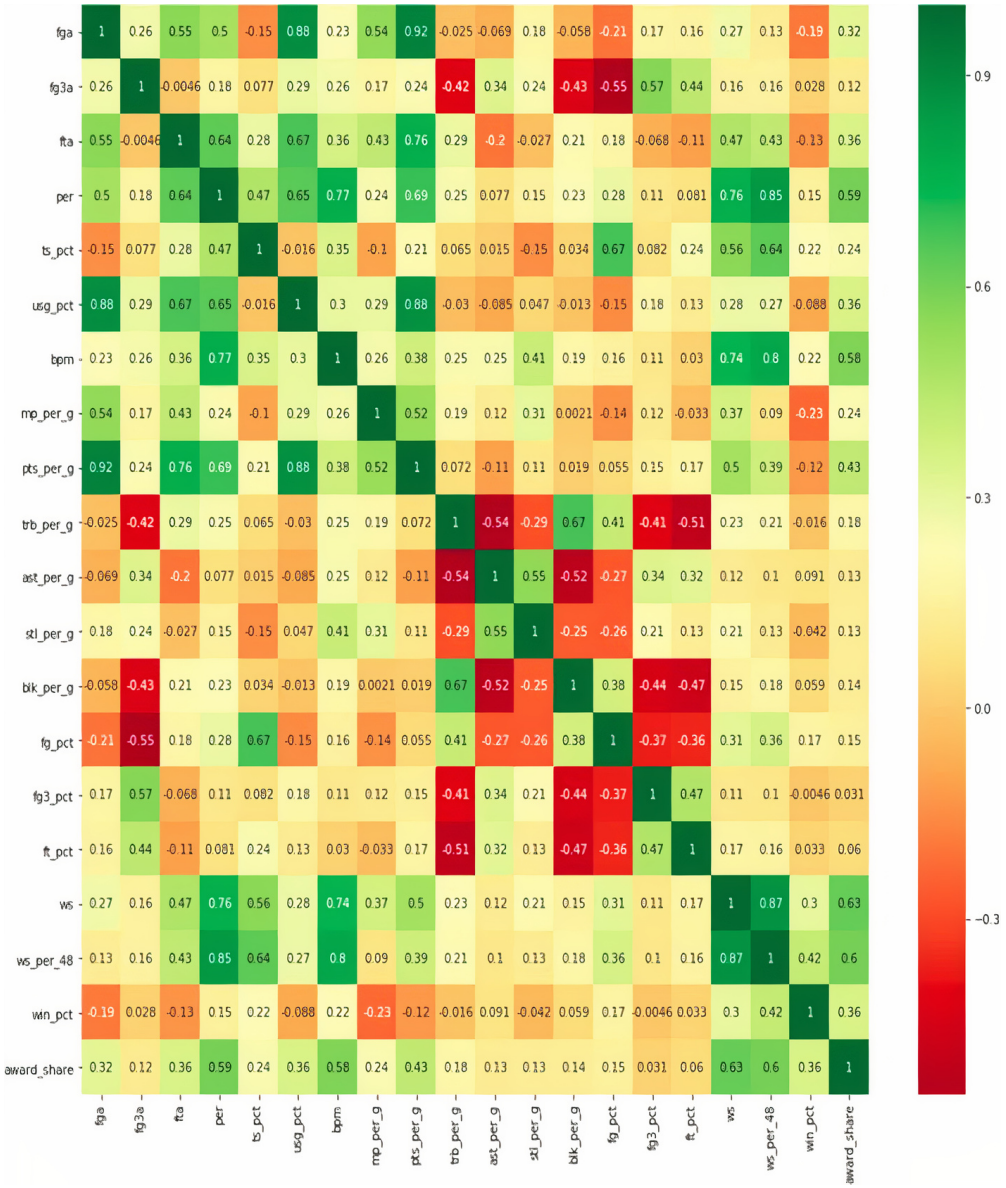

Figure 1. Confusion matrix of a multifeatured model based on the MVP system.

The method used to decide the season MVP has changed throughout the years, but a consistent primary source of votes comes from 100 media members. Each voter is given a ballot with which they can vote for five players. They make a vote for first place, which awards the player 10 points; for second place, which awards the player 7 points; and all the way down to fifth place (1 point). Therefore, it is fairly simple to turn this type of voting system into a regression problem. Bratulić attempted to predict the winner of each season's MVP by predicting a votes distribution, Figure 2. This led him to obtain fairly accurate results [18].

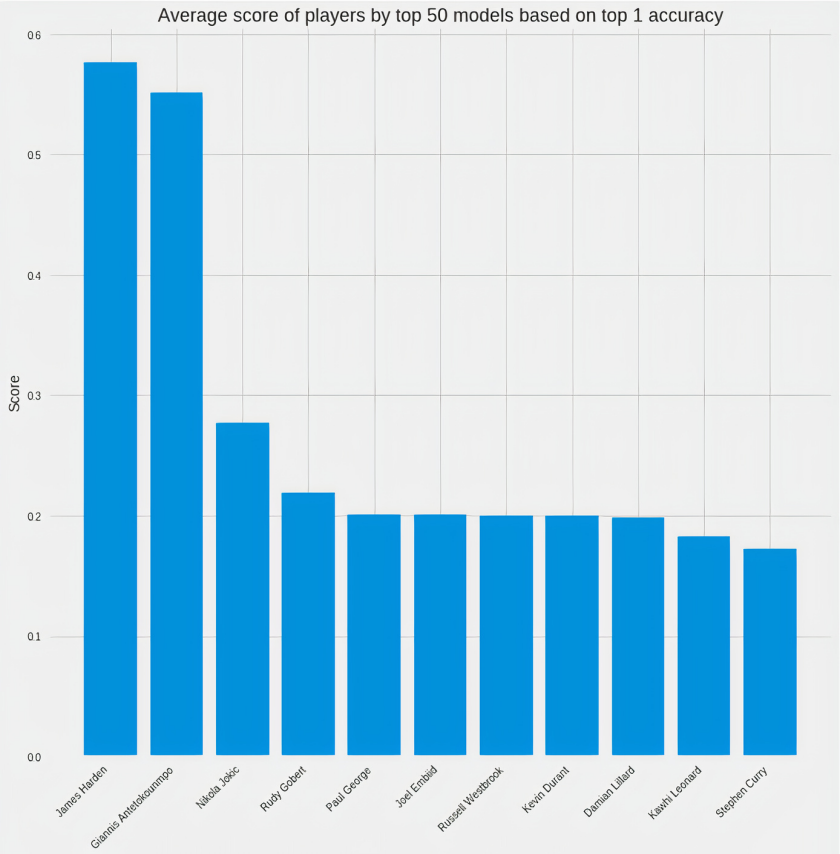

Figure 2. Fifty models' average scores of candidate players.

His method used 19 features, a mix of basic and advanced statistics, to predict award share. Bratulić then calculated and visualized feature correlation, closely correlated pruning features. Using the pruned features, he was able to estimate individual award shares, predicting the top award share player as the winner of the MVP award. This approach also allowed him to predict all vote-getters award shares. This resulted in graphs such as the one displayed in Figure 3.

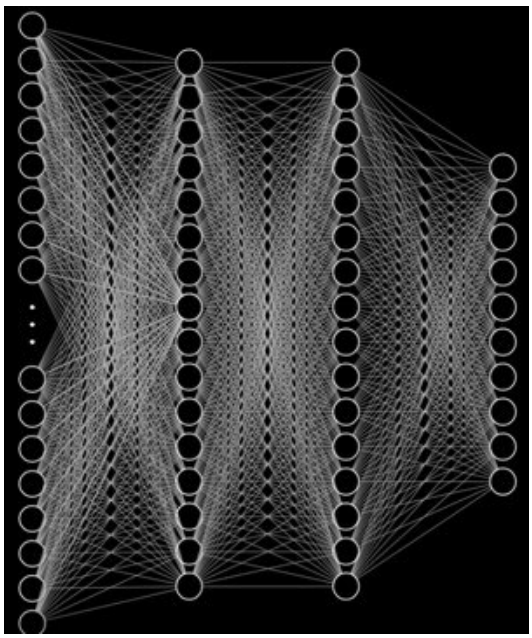

Figure 3. Generic ANN.

3. Methods

3.1. Defining Metrics

The goal of the proposed model is to predict as many NBA All-Stars as possible. While this may seem simple, deciding on a metric to measure the accuracy is a challenge. It is common in Machine Learning modeling to set up in advance the metrics needed to define the most feasible accuracy for the target model. Differing the metrics used to measure the success of a model could result in the same model being "accurate" and "inaccurate" simultaneously. Thus, for this project, the goal is to predict as many NBA All-Stars as possible (correctly or incorrectly), up to 40 per year. As the All-Star teams never consist of more than 30 people total, a goal of 40 may seem reasonably high.

One of the standard metrics we consider is the sensitivity, formally defined as follows:

$$\text{Sensitivity} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}} \quad (1)$$

For our model, true positives are players that are correctly classified as All-Stars. False positives are players that have been incorrectly classified as All-Stars. In other words, these are the players that the model incorrectly predicts as All-Stars. True negatives are non-All-Star players that were correctly predicted to be non-All-Stars. False Negatives are the All-Stars that were incorrectly predicted to be non-All-Stars.

Pivoting back to sensitivity, it can be easily understood by noticing it is the measurement of All-Stars classified correctly over the total number of All-Stars. As in Ryan Pollack's model from above, the number of true negatives is very large when compared with the true positives. If the metric that was chosen was raw accuracy (total correct predictions/total predictions), most models would be extremely accurate. For example, that would show Ryan's model as 99.06% accurate. This is not the metric targeted to improve as it is already defined as an optimal metric by the author. Since it is known that most players will be correctly classified as non-All-Stars, it seems the best focus on the All-Star group as it is considerably smaller. This is where sensitivity comes into play. It allows measurement of the percentage of correctly identified All-Stars.

3.2. Dataset Development

This project developed a dataset of over 17,000 players scraped from basketball-reference.com (accessed on 10 October 2021) [4]. For this project, a player can be defined as a single person's single-season statistics. For example, Kobe Bryant in 2008 was a different "player" from Kobe Bryant in 2009. Each player is a single season's worth of statistics. The dataset contains all players from the year 1980 until the most recent season. Each player's data contains 30 features, some of which were redundant. Examples of this are Field Goals Made and Field Goals Attempted, in combination with Field Goal Percentage. After the addition of the features, it was still necessary to add in the target values. In this case, these values came in the form of a Boolean value stating whether the player was an All-Star.

Using previous knowledge of basketball and the NBA, the dataset was filtered down to 4833 players using three criteria. The first requirement was that the player must have played in 50 games. The second is that the player must have started in 25 games. This is to weed out the players that were bench players and non-contributors. Rarely do bench players receive the All-Star honor. The last requirement is that a player must have played at least 15 min per game on average. This again is meant to eliminate players that would not have won the award regardless. The models were tested on both the cleaned dataset and the full 17,000 player list with similar results.

3.3. Choosing Models and Cross-Validation

After the development of the dataset, the next step was to test different models to view their accuracy. After choosing sensitivity as a metric, a method of splitting the data needed to be implemented. While the common approach to this is to use sklearn's train_test_split method, this is not the approach taken here. Since sports occur in seasons, it seemed

prudent to use a single season as the test set and every other season as training data. Using this method of splitting the data allowed for a realistic picture of how the models perform on a full season of information, which is the entire goal.

K-fold cross-validation was used to validate the results of each model. Cross-validation is a method of validating results by testing each model on different sets of training and test data. This is particularly helpful as it relies on multiple tests to check model performance. K-fold cross-validation is a method of performing this validation by dividing the data up into folds, training the model on all the data but a single fold and using the left-out fold as test data. This process is repeated for all folds, allowing for an accurate picture of the model's accuracy.

In this case, K-fold cross-validation was used by dividing the data into 40 folds (years) and by performing the validation this way. With each iteration, sensitivity was measured and at the end and then averaged. A simple average was used to obtain an estimation of each model's accuracy to compare them. The graphic below shows the models tested and their average sensitivity. The random forest classifier, the AdaBoost classifier, and the MLP classifier performed the best out of the models tested, see Table 2. These were the models that cut to the next stage.

Table 2. Sample of six Machine Learning models predicting SVC score.

SVC:	0.34
K-Nearest Neighbor:	0.47
Decision Tree classifier:	0.54
Gaussian Process Classifier:	0.56
Random Forest Classifier:	0.56
AdaBoost Classifier	0.56
Multilayer Perceptron Classifier	0.57

3.4. Models

3.4.1. Background: Decision Trees

Before discussing the theory behind the three models chose—AdaBoost, random forest, and Multilayer perceptron (MLP)—it is prudent to review the foundations on which these models stand.

Adaboost and random forest classification methods rely heavily on decision trees. MLP, is a class of feedforward artificial neural networks and is a different sort of model, requiring less background information to understand on a cursory level.

A decision tree is a supervised machine learning algorithm that uses an inverted tree-like structure to make decisions. Decision trees "ask" specific yes/no questions to make decisions and classify data. For example, a decision tree could ask "Is the player's points scored per game (ppg) greater than 20?" in its attempt to predict All-Stars. This question would be the first of many in the tree's attempt to predict whether the player was an All-Star. The goal of each of these questions is to subdivide the data into adequately many subcategories so that each subcategory is classified as either true or false.

The process of deciding how a tree is structured is important as improper placement of different questions can make a tree much more complicated. As an example, consider if a decision tree attempted to predict All-Stars using a question such as personal fouls. The goal of any question is to divide data points up more correctly than they were previously divided. As All-Stars do not commit more personal fouls when compared with non-All-Stars, dividing players based on personal fouls makes very little sense as it would perform terribly when categorizing players. Points per game, on the other hand, is an excellent metric for this division. While there are high-scoring non-All-Stars and low-scoring All-Stars, these players are outliers. On average, All-Stars score more points than their non-All-Star counterparts. Therefore, it follows that dividing players based on points per game would more correctly divide the players when compared with other metrics such as personal fouls.

Decision trees use two different metrics to determine which questions to ask. These metrics are Gini impurity and entropy. Both of these metrics are used to measure how well a specific question divides data. A question is generated, such as "Is the player's points per game greater than 15.6?"; the data are divided using that question; and then, the Gini impurity or entropy are calculated. Once these metrics are calculated, a new question is created, for example, "Is the player's points per game greater than 20?". The process is repeated with the new question until another Gini impurity or entropy value has been obtained. The new values are then compared with the old values. The question with the lowest value out of all possible questions becomes the question used to divide the data. If all new questions categorize the data in a manner that is so poor, it is no better than categorizing the entire group as an All-Star or not an All-Star; then, the group is categorized as a whole instead of being subdivided more.

To fully understand this method, it is important to define several terms used in the model. These are root, node, and leaf. A root is the base of the tree or the first question that divides the data. A node, in the same vein, is a decision point. The root node is the base of the tree, whereas further down, the nodes are simply called decision nodes. A leaf is a node that does not subdivide the data. These nodes are the parts of the tree that is classified.

Each leaf node is where a final answer has been reached: All-Star or non-All-Star.

The Gini impurity metric is calculated by measuring the true positive, true negative, false positive, and false negative metrics of each leaf and then by adding them up as weighted averages. The weighted values are used instead of true values because one node may only categorize 3 players, whereas another may categorize 50. It is understandable to allow the 50-player node to have more say in the final value than the 3-player decision node. The way that a decision tree decides what the root node should be is by comparing the Gini impurity using each feature as its root. In the end, it chooses the feature with the lowest impurity value for its root node. It continues this process recursively down the tree, choosing each node's decision feature by minimizing impurity. If using another decision node has the worst Gini impurity value, a leaf is used instead.

$$Gini = 1 - \sum_{j=1}^{c} p_j^2 \tag{2}$$

The above description is for Boolean values. To categorize data using numeric features, as in this case, the algorithm would calculate the averages of a feature between adjacent players before checking each of the calculation's impurity values to attempt to separate the data. Therefore, if attempting to separate data by points per game, we would calculate the average between adjacent players in the sorted list and then check to see which of these values would be the best separator of the data using the Gini impurity metric. The lowest of the averages is then chosen to be the question.

Entropy, on the other hand, is much simpler than Gini impurity. It is often considered to be a measure of disorder within data. The goal of each node of a decision tree should be to categorize more and more players correctly the further it goes down the tree. For example, dividing players by points per game is a good first question to initially make a divide among the players, but the next question should improve the true positive and negative rate even further. Below, the mathematical formula for entropy is displayed [19].

$$E(S) = \sum_{i=1}^{c} -p_i log_2 p_i \tag{3}$$

A dataset is considered *tidy* [15] if it contains only items with the same label (All-Star or not an All-Star), and *messy* [15] if it contains a mix of items with differing labels. for example, when there is no item with label 1 in the data ($p_i = 0$) or if the data contains only items of label 1 ($p_i = 1$), then the entropy equals 0. If the data are split evenly, entropy will be maximum as p = 1/2. The summation of these measurements defines the entropy of the entire tree. The lower the entropy at the leaf nodes of the tree, the more players the tree

properly classified [1]. In particular, decision trees as part of data mining have been used in specific sports such as baseball [10,20] and as a fundamental part of sports in general [21,22]

3.4.2. Random Forest

According to The Elements of Statistical Learning [23], "trees have one aspect that prevents them from being the ideal tool for predictive learning, namely inaccuracy".

Random forest, on the other hand, makes up for this while still holding on to the benefits given by decision trees. Random Forest is a machine learning model that can be used in regression tasks as well as classification tasks. The first step in creating a random forest is to create a bootstrapped data set. This means, in essence, that the data are randomly sampled to create a dataset that maintains the same size as the original. The difference is that the same sample is allowed to be used multiple times. The next step is to create a decision tree using the bootstrapped dataset but only using a random subset of features at each step. This process, from creating a bootstrapped dataset to creating a full decision tree is iterated over multiple times until many decision trees have been created. Each of the trees should look different, as different data were used to create them. This is the feature of random forests that allow them to be more effective than individual decision trees. After the model is created, it passes the new information through each tree and tallies up the "votes" from each tree. The category with the most "votes" at the end is the decision tree's classification.

Adaboost is a machine learning algorithm that is most commonly used with decision trees. Consider it a modification on the random forest model. Due to this similarities, the differences are covered in detail here, but the basic idea is the same.

To begin, in random forests, there is no predetermined size on decision trees. To contrast this, in AdaBoost, trees are often just one node and two leaves. This arrangement of a decision tree is often called a stump. This is the first main difference between AdaBoost and random forests. Next, while in random forests each tree obtain an equal "vote", in AdaBoost, some stumps obtain more say in the final classification than others. The last main difference between AdaBoost and random forests is in how the trees are made. In random forests, each tree is made independently of one another. However, in AdaBoost, the order is important. The errors one stump makes influence how the next stump is made.

Making an AdaBoost classifier is fairly straightforward. The process starts by assigning each sample in a dataset a weight. In the beginning, each sample has equal weight, meaning that they are all equally important. After assigning weight, a stump is made for each feature in the dataset. The stumps are created using the processes described in the decision tree described above. After each stump is created, the Gini impurity is calculated for each stump. Instead of allowing each stump to turn into an entire tree, it is simply cut off here by making the stump with the lowest Gini impurity value the first stump in the forest.

The next step is determining how much "say" the newly generated stump gets. This is performed by calculating the total error of the stump by adding the weights of the incorrectly classified samples. Therefore, if there were eight samples and one was classified incorrectly, then the error of that stump is 1/8. The amount of "say" each stump has towards the final tally is calculated using the following equation:

$$\text{Amount of say} = \frac{1}{2} log\left(\frac{1 - \text{total error}}{\text{total error}}\right) \tag{4}$$

Next, the weights of each sample must be modified to allow the algorithm to grow as the next stump responds to the previous one's errors. First, consider the incorrectly classified sample. This sample's "say" increases so that it is more important in the next classification. The formula to increase weight is shown below.

$$\text{New sample weight} = \text{sample weight} \times e^{\text{amount of say}} \tag{5}$$

Another change that is needed is to modify the sample weights for the correctly classified samples. The new weights are lower than the previous so they decrease in importance to the next stump. The formula to decrease weights is shown below.

$$\text{New sample weight} = \text{sample weight} \times e^{-\text{ amount of say}} \quad (6)$$

The final modification needed is the normalization of sample weights from 0 to 1. This is performed by dividing each new sample weight by the new total weight. This number is the normalized sample weight. Once this step is accomplished, the process is repeated on another stump. Once a forest of stumps is created, new data are fed through each stump. In the end, the amount of say for each classification is totaled. The class with the highest amount of say wins the AdaBoost classifier's classification. These slight modifications on random forests make AdaBoost effective.

MLPs (Multi-layer Perceptron) is a form of neural network based on the concept of a perceptron. A perceptron is a machine learning algorithm that inputs a vector of n + 1 dimensions and outputs another vector. A perceptron has a learning rate that governs how quickly it adjusts its decision boundary as well as a bias that acts the role of an axis intercept. After a perceptron calculates its result, the value is passed through an activation function. This function performs the role of smoothing the result into an accepted output. For example, if the perceptron outputs a 0.323 but the classes are 0 or 1, then the activation function takes 0.323 as its input and output a 0 or 1.

Multi-layer perceptrons are slightly more complex but can be simple enough if you consider the perceptrons as black boxes that input values with different weights attached to each, adjust them, and then output a new value. The illustration below shows how an MLP is organized.

Each node in the illustration is a perceptron. When an input is given, it starts at the left-hand side of the illustration and works its way towards the right. The leftmost layer is called the input layer as it is the layer that takes the initial inputs. Each perceptron adjusts the initial inputs and outputs the result to every perceptron in the second layer. This would mean that each perceptron in the second layer has some inputs equal to the input layer's size. The perceptrons continue to adjust themselves based on new information and adjust weights and biases. This allows each neuron (perceptron) to predict more correctly each time. After passing through the input layer into the second and third layers (all layers except the input and output are considered hidden layers), the network feeds into the output layer. The output layer's length depends on the number of outputs needed by a specific use case. For example, an output layer could output the probability percentage that a handwritten digit is each digit 1–10. The output layer can be as long as it needs to be to output the necessary number of outputs.

4. Results

In predicting whether an NBA player was an All-Star, each of these three algorithms was incredibly effective. Running a test case from the 1998 NBA season, it is possible to see real outputs as well as statistics on performance in a real-world test case.

4.1. Random Forest

The random forest classifier was the worst performer among the models tested with the 1998 test data. The random forest results can be seen below.

Out of the 115 predictions made from that season, it received a raw accuracy score of 87.3%, actually scoring between the other models tested. As discussed before, sensitivity is the primary metric that was used to determine accuracy whereas raw accuracy is secondary. This 11/21 or 0.524 means that the model correctly predicted 52% of true All-Stars. It is important to note that this number, while seemingly low, is very high for what the model attempted to predict. The sensitivity number of 0.524 coupled with the raw accuracy measurement of 87.3% means that overall accuracy was very high and over half of the

All-Stars were predicted correctly. These results provide much encouragement for the future of sports predictions, see Table 3.

Table 3. Random forest metrics for the target dataset.

Year:	1998
Predictions:	115
Raw accuracy:	0.878
True Positives:	11
False positives:	4
True Negatives	90
False Negatives	10
Sensitivity:	0.524
Specificity:	0.957

4.2. Adaboost

Adaboost performed much better than random forest in its sensitivity measurement. The score of 0.619 is the standout measurement with AdaBoost. The raw accuracy, on the other hand, dropped slightly, see Table 4. The accuracy drops by 2 percent; however, this model proves to be more sensitive in real-world numbers. That jump would mean two extra All-Stars were predicted correctly. Statistically, the 2% drop in raw accuracy has little to no effect on the model.

Table 4. Adaboost metrics for the target dataset.

Year:	1998
Predictions:	115
Raw accuracy:	0.852
True Positives:	13
False positives:	9
True Negatives	85
False Negatives	8
Sensitivity:	0.619
Specificity:	0.94

4.2.1. MLP

The multi-layer perceptrons, on the other hand, were the clear winner among the three models. This model performed as well as AdaBoost in terms of sensitivity and outperformed both the other models in raw accuracy. This was the most effective model in performing All-Star classifications, see Table 5.

Table 5. Multilayer perceptron metrics for the target dataset.

Year:	1998
Predictions:	115
Raw accuracy:	0.887
True Positives:	13
False positives:	5
True Negatives	89
False Negatives	8
Sensitivity:	0.619
Specificity:	0.94

4.2.2. Hybrid Proposal

Considering that each of these models performed well in overall testing, our proposal considers a formula of both methods assigning a variable weight to the prediction. The results suggest that the known techniques tested in this paper have proven to be decent models in terms of sensitivity and specificity. Our proposal adds a new layer of pattern

detection by defining a basic ANN in which the input neurons are the output of the previous models. This ANN has one hidden layer. Adaboost, MLP, and random forest are logical models that could be used to accurately predict All-Stars for any upcoming NBA season. Initially, weights were assigned with random weights. The topology proposed follows the following architecture:

The input has this format $(x_0, \omega_0), (x_1, \omega_1), (x_2, \omega_2)$, where x_i represents the input; ω_i is the initially assigned weights; and ω_{ij} represents the weight assigned to neuron i, which connects to neuron j, as any ANN standard architecture. The ANN is a standard 3-6-1 topology. The input layer has three neurons, following the shape of the observations in the datasets and the requirements. The input layer's weighting is defined randomly, and the activation function is the sigmoid function. The hidden layer has six neurons fully connected to the input and output layers.

The output layer contains one neuron and represents a value between 0 and 1: 1 is the highest probability of being the star player, and 0 the lowest. The architecture and topology was chosen by focusing on the MLP stucture and the observations shown below. As MLP showed a decent overall accuracy, it was the main inspiration in the hybrid proposal, where we tried to use a similar basis to the one described above.

The observations in the dataset follow the pattern illustrated below:

```
[Pos Year Age Tm G GS MP FG FGA FG% 3P 3PA 3P% 2P 2PA 2P% eFG% FT FTA FT% ORB DRB TRB AST STL
BLK TOV PF PTS] [STAR]
```

where [STAR] represents the output label 1 if the player is a star that year or 0 if not. Pos refers to the position that the player played, Year means the year that the player played, Age is the age of the player when they played in that year, and Tm is the team that the player played on that year. These are the main variables that are part of the observation for every potential player. [STAR] is the prediction, and it is the variable used to construct the accuracy metric in the next sections.

The models composed of ANN (Adaboost, Random Forest, and MLP) feed off the same data set. This weighted combination of the conventional three models has not been a subject of research research, which makes this approach novel in this field.

4.3. Training Phase ANN

The training phase aims to solve an optimized combination of the three models. The objective function is defined as the mean of the residual sum of squared errors. (See formula below).

$$f(\omega_0, \omega_1, \omega_2) = \sum_{0}^{2} [\omega_i f_i - y_i]^2 \tag{7}$$

where f_i is the known component function of the hybrid proposal. f_0 = random forest prediction, f_1 = adaboost prediction, f_2= MLP prediction, and y_i is the Real observation (the real value assigned to the player).

For the training phase of the ANN, setting the gradient to 0 or working with the *Gradient Descent* algorithm are well-known methods [17]. Setting the gradient equal to 0 and solving the systems of equations is usually ruled out due to the high complexity. n^p, where p is the number of weights to adjust. However, its advantage is that it can calculate the optimal value without waiting for convergence as other methods such as *Gradient Descent* or *Particle Swarm*. This method was chosen to be the one to train the ANN as it is usually the most accurate. The number of weights is small, and therefore, the problems of high complexity in the number of calculations associated with this method are not present.

Table 6 displays the weights obtained from the training phase and the training error (normalized value of the residual sum of squares) that occurred when setting the gradient = 0).

Table 6. Obtained weights from the training phase and the training error setting gradient to 0.

Training Error	**0.0824432**
ω_0	0.28873472
ω_1	0.38407212
ω_2	0.32719316

The coding environment has four different related technologies: Keras Version 2.3.0, TensorFlow 2.0, Anaconda 3.51, and Python 3.6.

4.4. Testing and Avoiding Overfitting

Overfitting can occur when minimizing the training error. The weights of the ANN might be adjusted to minimize that training error. However, there might be another combination of weights with a higher training error but a lower testing error. That usually occurs when the number of neurons in the intermediate layers is more extensive than needed. One of the most effective methods to measure an overfitted model is using cross-validation.

In our case, cross-validation divides the observations into ten datasets: nine for training and one for testing.

In every iteration, the testing accuracy is calculated. The total accuracy is defined as the average of the ten rounds. Cross-validation compares ten possible optimal combinations of weights based on the observations in the training datasets. Choosing the model with a good test accuracy helps decrease overfitting as it compares the test accuracy of different optimized training models, see Figure 4.

Year	Player
1998	Ray Allen
1998	Clyde Drexler
1998	Joe Dumars
1998	Tim Duncan
1998	Kevin Garnett
1998	Grant Hill
1998	Allen Iverson
1998	Michael Jordan
1998	Jason Kidd
1998	Karl Malone
1998	Reggie Miller
1998	Alonzo Mourning
1998	Chris Mullin
1998	Dikembe Mutombo
1998	Shaquille O'Neal
1998	Gary Payton
1998	Mitch Richmond
1998	David Robinson
1998	Dennis Rodman
1998	Arvydas Sabonis
1998	John Stockton

Year	Player	STAR
1998	Shareef Abdur-Rahim	FALSE
1998	Clyde Drexler	TRUE
1998	Tim Duncan	TRUE
1998	Kevin Garnett	TRUE
1998	Grant Hill	TRUE
1998	Jeff Hornacek	FALSE
1998	Allen Iverson	TRUE
1998	Michael Jordan	TRUE
1998	Jason Kidd	TRUE
1998	Karl Malone	TRUE
1998	Alonzo Mourning	TRUE
1998	Dikembe Mutombo	TRUE
1998	Reggie Miller	TRUE
1998	Shaquille O'Neal	TRUE
1998	Mitch Richmond	TRUE
1998	David Robinson	TRUE
1998	Arvydas Sabonis	TRUE
1998	Dennis Rodman	TRUE
1998	Detlef Schrempf	FALSE
1998	John Stockton	TRUE
1998	Rod Strickland	FALSE
1998	Antoine Walker	FALSE
1998	Chris Webber	FALSE
1998	Jayson Williams	FALSE
1998	Kevin Willis	FALSE
1998	Otis Thorpe	FALSE

Figure 4. Sample of ANN performance. Green: hits. Red: faults.

4.5. Testing—Results in ANN

As mentioned in the previous section, our ANN is tested in different epochs using K-fold cross-validation. This is described as follows:

1. Split the training data set into six equal subsets or folds. $f_1, f_2, \cdots, f_6$
2. For $i = 1 \rightarrow 6$
   (a) The data set data in f_i works as the validation set, and the rest of the $k-1$ folds are the cross-validation training set.
   (b) Estimating the accuracy of your machine learning model by averaging the accuracies derived in all of the k cases of cross-validation.

Accuracy: Tables 7 and 8 shows the accuracy found in every k-fold rounded to five digits of precision for proper display.

Table 7. Accuracy in every k-fold.

0.89726	0.90351	0.90045	0.89756	0.90812	0.903221

Table 8. ANN metrics for testing the same dataset.

Year:	1998
Predictions:	115
Raw accuracy:	0.887
True Positives:	17
False positives:	9
True Negatives	85
Sensitivity:	0.81
Specificity:	0.904

Example test. To illustrate this point, a union of the predictions made by each model is shown below. The incorrectly classified players are highlighted in red. Moreover, it is a correct list of All-Stars for that year. The green highlights show the accurately predicted players.

When comparing the results of this proposal with the models in [9,15], the improvement is also shown. In [9], Tolbert used Support Vector Machines with different kernels for baseball players with accuracy values reaching 0.6. In the Pollack confusion matrix, specificity and sensitivity are considerably lower.

As is visible, this combined, hybrid model performed significantly better than each model in terms of sensitivity, while the raw accuracy was not diminished. Although the prediction accuracies do not reach values such as 0.97, unlike other ML methods for patterns recognition, these numbers are around the highest in prediction successful stars in different sports. This is due to the different prediction methods of each model. Each model predicts a specific subset of players as an All-Star, but that subset differs between models. The results suggest that this proposal is an improvement in previous models, and this study concludes that it is a reliable approach as it is a combination of known and reliable methodologies.

Therefore, combining the models into a single prediction set allows for correct guesses to stand while minimizing the effects of the, mostly unanimous, incorrect predictions. This can be seen by the over 20% increase in terms of sensitivity.

5. Conclusions

While each model performed admirably on its own, the hybrid model outstripped them by a significant margin.

This hybrid model requires the user to use a multimodel to better improve accuracy in sports predictions as well as in other fields. The significant jump in accuracy is a conclusive sign that total sensitivity can be increased even higher.

The results show that these models have received the best scores that can be expected with the studied data sets. However, improvements can still be made. The first one involves the data shape. While the data is comprehensive to the player, more than a player's statistics go into earning an All-Star roster spot, such as a player's popularity. That affects voting. Thus, considering that each player's followers on various social media platforms could play a part in future iterations of these predictions, another possible improvement that could be made to the data is to include the player's statistics with regards to their team. For example, a team's win/loss record as well as a player's win shares (an estimation of how many wins each player contributes to a team throughout the season) could be included.

Other improvements could involve using deep learning models. Beyond using deep learning models, it is feasible to use combinations of other models such as the one attempted above. This would allow for another layer of abstraction beyond the basic models used

here. Perhaps layering these models in this manner could provide a system for weighting each entry to weed out false positives as well.

Overall, this attempt to predict NBA All-Stars was successful. Each model performed admirably independently before coming together to predict over 80% of true positives correctly. The next steps in making this technology widely accessible will be to make these suggested improvements to the dataset as well as to publicize the methods used in this work as well as those developed in the future.

This work also shows that combining different models might be an excellent choice for predicting sport features in the future. Thus, further work needs to be conducted in exploring combining different techniques as this can open new ways to improve the known models rather than finding new ones.

Author Contributions: Conceptualization, A.A.A. and K.A.; methodology, N.G.B.; software, L.F.d.M.L.; validation, K.A., A.A.A. and N.G.B.; formal analysis, L.F.d.M.L.; investigation, L.F.d.M.L.; writing—original draft preparation, A.A.A. and K.A.; writing—review and editing, N.G.B.; supervision, L.F.d.M.L. All authors have read and agreed to the published version of the manuscript.

Funding: The authors declare no conflicts of interest. This research has no funding sponsors so they had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Acknowledgments: This work was partially supported by the Comunidad de Madrid under Convenio Plurianual with the Universidad Politécnica de Madrid in the actuation line of Programa de Excelencia para el Profesorado Universitario. This work has been partially supported by projects Seguridad de Vehículos AUTOmóviles para un TRansporte Inteligente, Eficiente y Seguro. SEGVAUTO4.0 P2018/EMT-4362, and CICYT PID2019-104793RB-C33.

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

References

1. Ricaud, B. A Simple Explanation of Entropy in Decision Trees. 2021. Avaliable online: https://bricaud.github.io/personal-blog/entropy-in-decision-trees/ (accessed on 10 November 2021).
2. Maheswaran, R. The Math Behind Basketball's Wildest Moves, 2020. Avaliable online: https://www.ted.com/talks/rajiv_maheswaran_the_math_behind_basketball_s_wildest_moves (accessed on 10 November 2021).
3. Committee, I.O. The History And Tradition Of The Olympic Games. 2018. Avaliable online: https://olympics.com/en/video/the-history-and-tradition-of-the-olympic-games (accessed on 10 November 2021).
4. Bible Timeline. 2021. Available online: biblehub.com (accessed on 10 November 2021).
5. Basketball Statistics and History. 2000. Available online: Basketball-Reference.com (accessed on 10 November 2021).
6. Schwartz, A. A Numbers Revolution. 2004. Avaliable online: https://www.espn.com/mlb/columns/story?columnist=schwarz_alan&id=1835745 (accessed on 10 November 2021).
7. James, B. *Beyond Baseball*; PBS: Dallas, TX, USA, 2005. Avaliable online: https://www.beyond-baseball.com/(accessed on 10 November 2021).
8. James, B. *The Bill James Baseball Abstract*; Ballantine Books: New York, NY, USA, 1982.
9. Tolbert, B.; Trafalis, T. Predicting Major League Baseball ChampionshipWinners through Data Mining *Athens J. Sport.***2016**, *3*, 239–252. *Int. J. Comput. Sci. Sport* **2016**, *15*, 91–112. [CrossRef]
10. 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]
11. Rida, I.; Al-Maadeed, N.; Al-Maadeed, S.; Bakshi, S. A comprehensive overview of feature representation for biometric recognition. *Multimed. Tools Appl.* **2018**, *79*, 4867–4890. [CrossRef]
12. Covers, T. How Are Betting Lines Created. 2016. Avaliable online: https://www.youtube.com/watch?v=FbKBNZjPWM0 (accessed on 10 November 2021).
13. Eckhardt, R. Stan ulam, john von neumann, and the monte carlo method. *Los Alamos Sci.* **1987**, *15*, 131–136.
14. Boice, J.; Silver, N; Paine, N. 2020 NFL Predictions, 2020. Avaliable online: https://projects.fivethirtyeight.com/2020-nfl-predictions/quarterbacks/ (accessed on 10 November 2021).
15. Wilkens, S. Sports Prediction and Betting Models in the Machine Learning Age: The Case of Tennis. *J. Sport. Anal.* **2021**, *7*, 99–117. [CrossRef]
16. Pollack, R. Predicting the 2018 MVPWinners with Machine Learning. 2018. Avaliable online: https://tht.fangraphs.com/mvp-2018-mookie-betts-christian-yelich-machine-learning/ (accessed on 10 November 2021).
17. Jarvis, J. A Survey of Baseball Player Performance Evaluation Measures. 2015. Avaliable online: http://knology.net/webmigrate/ (accessed on 10 November 2021).

18. Bratulic, D. Basketball Analytic. 2019. Avaliable online: https://towardsdatascience.com/analysis-of-after-timeout-plays-in-nba-f69952f74779 (accessed on 10 November 2021).
19. Vajda, S. The Mathematical Theory of Communication. *Math. Gaz.* **1950**, *34*, 312–313. [CrossRef]
20. Neil, G. Analysis, The statistical revelation that has MLB hitters bombing more home runs than the steroid era. *Washington Post*, 1 June 2017.
21. Sugato, R. The Evolution and Future of Analytics in Sport. *Proem Sports | Sports Analytics | Singapore and India*, 22 June 2017.
22. Leigh, S. CHANGING THE GAME: The Rise of Sports Analytics. *Forbes*, Publish date August 18th, 2015. Retrieved 22 April 2018.
23. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2009.