Predicting NBA Game Outcomes Using Supervised Machine Learning Models

Reginald McLean, James Zhou, Alykhan Sewani, Ali Aboubih

I. Introduction

Sports are a favourite pastime for people around the world. This global popularity has resulted in a huge number of statistics being collected and organized, a goldmine for data scientists. This makes predicting the outcome of sporting events both a fun pastime, and a lucrative one when it comes to sports betting. Using Machine Learning (ML) for predictive sports modeling is a very well documented field, with basketball being the second most studied sport [1].

It has been shown that ML can be used to generate a profit from sports betting [5]. In this case the greater the accuracy of the models the greater the potential profit. However, even with the amount of current documentation that exists, there appears to be a glass ceiling of 75% accuracy in current models [6]. This could stem from the fact that, even with the varying amount of models that have been used, similar features are being selected. Feature selection has been argued to be even more important than model selection in this problem. Another issue is that data sets vary drastically from study to study, inhibiting researchers from constructively comparing their findings with previous work and leading to unclear development [7].

Datasets for basketball statistics are maintained by the NBA on their website, as well as multiple Kaggle datasets which fit the needs of this project. The most popular models for sports prediction tend to be artificial neural networks (ANN) and support-vector machines (SVM), with the latter performing the best in many instances [3] [4].

A dataset from something or another was selected for this project, due to the specific statistics it had. The statistics were averaged for each team over the last 5, 10, and 15 games. These averages were separated into three different datasets for comparison. ANOVA tests were conducted on the datasets to select the top 15 features. Once those features were extracted, the final dataset was normalized. Initially, logistic regression was used as a baseline model. More complex models were selected from the literature to be: ANNs, SVM, Decision Trees (DT), and XGBoost. All models were tuned and iterated before a final test was run for comparison.

II. Problem Statement & Dataset

A. Problem Statement

There is plenty of compiled data made public by professional sports organizations such as the NBA. This data can be used to predict the outcomes of future matchups which can benefit parties such as sports analysts, ticket sales, fans, and those involved in sports betting. The goal of this study is to analyze the performance of different machine learning algorithms and select our optimum algorithm for predicting match outcomes.

B. Dataset

With the vast collection of extensive data in professional basketball found on websites such as the Official NBA website, Basketball Reference, and ESPN, we needed to compile data that was appropriate for the modern game. This data collection dated back to the middle of the 20th century with more detailed player level statistics, advanced performance metrics, etc. starting to appear in the 1980s. To predict team performance, we needed a dataset with these more detailed statistics over many seasons.

We narrowed our research down to the NBA Enhanced Box Scores and Standings 2012-2018 dataset from Kaggle, which includes data for every regular-season game starting from the 2012-2013 season, up until the 2017-2018 season. It includes detailed team-level statistics, which tally the player-level data for each game, and calculate advanced statistics such as Efficiency Differential, and Offensive or Defensive ratings which proved to be valuable. Moreover, the 2012-2018 period is representative of the dynamics of today's NBA as it includes the majority of today's most dominant teams and players. We also had over 7500 games to train our models on, which was sufficient and produced results that are in-line with our literature review.

The team box-score data is laid out where there are two rows per game, one row for the home team, and one row for the away team. Each row had two sets of columns with team statistics and opponent statistics as in the statistics for each game was repeated twice from different perspectives, which is useful for some sports-modeling.

III. Methods & Models

A. Data Engineering

We needed to get the data in a format where every row presents the rolling average of statistics for either team in the match-up over the past n games. This smooths the data and removes noise for implementing our classification models. First, the data was sorted based on game date and the team IDs. A dictionary was created that was indexed by team ID and game date to store the raw data from the dataset. Irrelevant columns, such as team division, and referee names were disregarded in our model.

To acquire a statistics average for the last n games, this ordered dictionary was required. The statistics were averaged for the last 5, 10, and 15 games in separate datasets. For each team ID, an average was taken at each game date consisting of the previous n games; this data replaced the dictionary row containing the raw data – the only parts of the data untouched were the win or loss column as this was the target output. These averaged data points then replaced the original non-averaged row in the initial data frame where the data was first read. This resulting dataset was then in a form that we didn't want to work with. In this form, the data read such that each game had two rows: one for the home team with the statistics of the previous opponents that they had faced, and one for the away team of a game with the previous statistics of the opponents they had played.

We tackled this by going into the original dataset and deleting the opponent columns entirely. This left us with only one column of data, that had a row of home team data and away team data for each game. Running our original averaging algorithm on this version of the dataset meant that for each row, we now have an average of how that specific team performed over the last n games. We then sorted the dataset manually by "Home"/"Away", followed by sorting each category by game date, followed by concatenating both blocks in a way that left us with one row per game – a set of columns with home statistics, and a set of columns with away statistics, recalculating pace and possession.

B. Feature Selection

This task was dependent on the features that are present in the dataset. In this case, we have a set of numerical input features with a binary categorical output. Because of this combination we used ANOVA testing to determine the correlation between each feature and the output. The feature selection algorithms from the package scikit-learn were used to perform this task. Once the correlation between features was visualized, the top 15 features were selected to be used as the inputs to our models. The number 15 was used after a visual inspection of the correlations, as 15 seemed to encompass all of the highly correlated features as well as a few less correlated features.

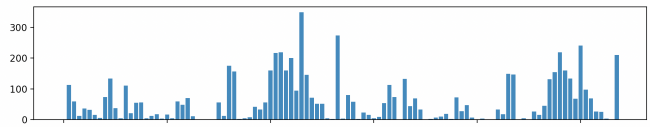

Fig. 1. An example of the generated feature selection graphs

C. Data Normalization

Algorithms that rely on combining/manipulating features through feature mapping can run into issues when there is a large difference in the scale between features. This also presents a problem when updating weights through gradient descent as unscaled data can cause large errors for initial parameter guesses in the update rule resulting in an unstable learning process. Therefore we normalize our data to be within [-1, 1] so we don't run into these errors.

D. Model Selection

Models were selected based on what was found in relevant literature. ANNs and SVM have been extensively used in this domain previously [3] [4]. DT have been applied to other applications in the sports domain, namely predicting the hall of fame status of Major League Baseball players [9]. We applied DT to our problem because we wanted to see if it was efficient in another sports domain. Similarly to DT, gradient boosting was chosen because of its efficiency in predicting English Premier League Team Season Win Percentages [10]. We thought that this application was similar to our problem and wanted to include it for that reason.

E. Model Implementation

For this project, models were implemented from pre-existing libraries such as scikit-learn and Keras. For a baseline model, logistic regression (LR) was chosen due to the problem being binary classification. The more complex models that were implemented are: DT, XGBoost, SVM, and ANNs. For each model the dataset was split into a ratio of 70/10/20 for training/validation/test datasets. Optuna was used to tune the hyperparameters of each model, with the objective of maximizing the validation accuracy. This was done for each 5, 10, and 15 game average datasets. Once a tuned set of hyperparameters was obtained for each situation, the models were trained on the training dataset, and ran predictions on the testing set. This process was run 10 times for each situation to provide statistical weight behind the results. The average training and validation accuracies were compared for bias-variance analysis to determine if regulation needed to be added/reduced or if a more complex model was needed. The model was then re-tuned with the added changes and the bias-variance analysis was done on the new results. Once a model was finalized it was trained on the training dataset and ran predictions on the test data set for the final results. This was also done 10 times and average for statistical significance. For the final comparison between models accuracy was used.

F. Bias-Variance Analysis

The bias-variance analysis performed in this paper was performed by examining the training accuracy and comparing it to the validation accuracy. Using this technique, we believe that we could create a valid way to ensure that we were not creating a model that was biased towards one class or the other, or creating a model that would not perform well on unseen data.

G. Model Regularization

Unfortunately not all of the our models have the same regularization technique. Therefore, for LR, we applied L1, L2, and elastic-net, which is a combination between L1 and L2. In SVM we applied a linear, Gaussian, and a sigmoid kernel.

ANNs and XGBoost applied both L1 and L2 regularization. For DT, Random Forests(RF) were used.

IV. Results

In this section we present our results for each model. In each subsection a small discussion about the preliminary models we trained is included, then a small discussion of the effect of regularizing the model is included as well. Lastly, the performance of all the models are compared by comparing average testing performance to determine which approach was the best.

A. Logistic Regression

The first model tested for this problem was a logistic regression model without regularization using up to 10000 epochs and a tolerance of 0.0001. The results of the accuracies on each training set can be found in I, II, and III. The 15-game dataset using n=10 independent trials was the most accurate with an accuracy of 0.661 +/- 0.0021 on the training set, an accuracy of 0.665 +/- 0.014 on the validation set, and an accuracy of 0.651 +/- 0.0017 on the test set. As, this was the simplest model, this served as a baseline for the more complex models. Based on the similarity between the validation set and the training set, there was very little variance error. And, from the 65% accuracy, this was well below the 75% accuracy ceiling which demonstrated the bias error with our baseline. To find improvements in this model, three types of regularization were applied – an L1 regularizer, an L2 regularizer, and an elastic-net regularizer (a combination of L1 and L2). For the optimum accuracy, the associated hyperparameters were optimized based on the validation set. As can be seen in Tables I, II, and III, the accuracies for each regularizer with the associated datasets were about the same. The highest accuracy was found with the elastic-net regularizer on the 15-game dataset with accuracies of 0.662 +/- 0.0016, 0.665 +/- 0.013, and 0.656 +/- 0.0015 for the training set, the validation set, and the test set, respectively. Due to the same bias error, more complex models were evaluated.

	Base Model	L1 Reg	L2 Reg	Elastic-Net
Training Accuracy	0.643 (0.0027)	0.646 (0.0031)	0.642 (0.0015)	0.643 (0.0032)
Validation Accuracy	0.638 (0.022)	0.643 (0.021)	0.636 (0.020)	0.648 (0.024)
Testing Accuracy	0.618 (0.0019)	0.617 (0.0018)	0.609 (0.0018)	0.622 (0.0011)

TABLE I
LOGISTIC REGRESSION RESULTS 5 GAME AVERAGE [AVG(STD)]

B. Support Vector Machine

SVM is a more complex model compared to logistic regression – it optimizes the distance between datapoints and the decision boundary. The hyperparameters of three different kernels were optimized. The linear kernel was the simplest kernel, and it showed the best accuracy on the 10-game dataset with a training accuracy of 0.649 +/- 0.0029, a validation accuracy of 0.651 +/- 0.026, and a test accuracy of 0.643 +/- 0.0012. The more complex Gaussian and sigmoid kernels were applied with the sigmoid kernel being the most accurate at accuracies of 0.663 +/- 0.0026, 0.658 +/- 0.022, and 0.652 +/- 0.0026, for the training set, validation set, and test set, respectively (Tables IV, V, and VI). The accuracies from this model were very similar to the logistic regression model with the elastic-net regularizer.

	Base Model	L1 Reg	L2 Reg	Elastic-Net
Training Accuracy	0.656 (0.0024)	0.656 (0.0016)	0.654 (0.0023)	0.644 (0.0021)
Validation Accuracy	0.657 (0.015)	0.646 (0.011)	0.656 (0.016)	0.653 (0.020)
Testing Accuracy	0.642 (0.0021)	0.637 (0.0011)	0.638 (0.0018)	0.639 (0.00099)

TABLE II
LOGISTIC REGRESSION RESULTS 10 GAME AVERAGE [AVG(STD)]

	Base Model	L1 Reg	L2 Reg	Elastic-Net
Training Accuracy	0.661 (0.0021)	0.662 (0.0022)	0.662 (0.0015)	0.662 (0.0016)
Validation Accuracy	0.665 (0.014)	0.654 (0.019)	0.660 (0.013)	0.665 (0.013)
Testing Accuracy	0.651 (0.0017)	0.650 (0.002)	0.655 (0.0016)	0.656 (0.0015)

TABLE III
LOGISTIC REGRESSION RESULTS 15 GAME AVERAGE [AVG(STD)]

Kernel	Linear	Gaussian	Sigmoid
Training Accuracy	0.639 (0.0044)	0.642 (0.0023)	0.642 (0.0052)
Validation Accuracy	0.632 (0.022)	0.637 (0.019)	0.642 (0.024)
Testing Accuracy	0.613 (0.0013)	0.605 (0.0016)	0.622 (0.0016)

TABLE IV
SVM RESULTS 5 GAME AVERAGE [AVG(STD)]

Kernel	Linear	Gaussian	Sigmoid
Training Accuracy	0.649 (0.0029)	0.657 (0.0019)	0.652 (0.0023)
Validation Accuracy	0.651 (0.026)	0.650 (0.020)	0.664 (0.023)
Testing Accuracy	0.643 (0.0012)	0.639 (0.0034)	0.639 (0.0028)

TABLE V
SVM RESULTS 10 GAME AVERAGE [AVG(STD)]

C. Decision Trees

DT implementation was conducted using the scikit-learn. Our best validation accuracy was 64.5 +- 1.4% on the 15-game-average dataset. In our bias-variance trade-off, we trade variance for bias, having trees have a low bias and a high variance, and overfitting the data to get the best results on the training data, yielding lower accuracies. Regularization, in its traditional definition of applying a penalty to the cost function, is hard to implement, as there is no global cost function. To

Kernel	Linear	Gaussian	Sigmoid
Training Accuracy	0.598 (0.0019)	0.656 (0.0017)	0.663 (0.0026)
Validation Accuracy	0.592 (0.034)	0.660 (0.012)	0.658 (0.022)
Testing Accuracy	0.588 (0.0041)	0.649 (0.0013)	0.652 (0.0026)

TABLE VI
SVM RESULTS 15 GAME AVERAGE [AVG(STD)]

decrease overfitting, we can use a bagging ensemble method such as RF. RF Classifiers output the majority prediction of many DT, which yielded better testing accuracies, but it significantly overfit our data, even more so than DT. This means we ended up with a lower bias, and higher variance, which confirms our concern that that the limiting factor in our study is feature selection, as our models, and our tuning, didn't improve our results as expected when it comes to bias-variance analysis, even if our overall accuracies improved. Our highest validation accuracy for the ensemble classifier was 64.9 +- 2.2 % on the 10-game-average dataset. However, our improvement in accuracy was marginal, which still shows that

	Random Forests	Decision Trees
Training Accuracy	0.787 ± 0.005	0.641 ± 0.004
Validation Accuracy	0.629 ± 0.019	0.631 ± 0.022
Testing Accuracy	0.612 ± 0.005	0.610 ± 0.004

TABLE VII
DECISION TREES RESULTS 5 GAME AVERAGE

	Random Forests	Decision Trees
Training Accuracy	0.809 ± 0.003	0.647 ± 0.004
Validation Accuracy	0.649 ± 0.022	0.639 ± 0.022
Testing Accuracy	0.6334 ± 0.002	0.611 ± 0.010

TABLE VIII
DECISION TREES RESULTS 10 GAME AVERAGE

	Random Forests	Decision Trees
Training Accuracy	0.800 ± 0.003	0.653 ± 0.002
Validation Accuracy	0.645 ± 0.014	0.645 ± 0.014
Testing Accuracy	0.644 ± 0.003	0.631 ± 0.006

TABLE IX
DECISION TREE RESULTS 15 GAME AVERAGE

D. XGBoost

XGBoost (Extreme Gradient Boosting) is a popular algorithm that uses gradient boosted DT. There are many hyperparameters that can be adjusted in the XGBoost library. For this application we set the objective to be binary classification. The other hyperparameters of interest were: max depth, learning rate, gamma, minimum child weight and epochs. These hyperparameters were tuned using Optuna to maximize the validation accuracy. The tuned model was then run 10 times to train and predict for computation of training and validation accuracies. DT algorithms do not require normalized data, so this was done with all 6 datasets, with the 15-game normalized dataset having the highest validation accuracy of 0.659 ± 0.017. Compared to a higher average training accuracy of 0.761 ± 0.008, this is indicative of high variance possibly due to overfitting. To counter this, the model was retuned with the inclusion of the L1 and L2 regularization weights, alpha and lambda. This resulted in a model with average training accuracy of 0.729 ± 0.003 and average validation accuracy of 0.648 ± 0.022 which shows slightly less variance than the nonregularized model. The lower variance indicates that this model will be able to make better predictions on unseen data so it was selected as the final XGBoost model.

	Unregularized	Regularized
Training Accuracy	0.696 ± 0.003	0.678± 0.003
Validation Accuracy	0.632 ± 0.013	0.638 ± 0.021
Testing Accuracy	0.620 ± 0.003	0.620 ± 0.004

TABLE X
XGBOOST RESULTS 5 GAME AVERAGE

	Unregularized	Regularized
Training Accuracy	0.670 ± 0.006	0.666 ± 0.005
Validation Accuracy	0.640 ± 0.022	0.628 ± 0.016
Testing Accuracy	0.629 ± 0.007	0.627 ± 0.008

TABLE XI
XGBOOST RESULTS 10 GAME AVERAGE

	Unregularized	Regularized
Training Accuracy	0.761 ± 0.008	0.729 ± 0.003
Validation Accuracy	0.659 ± 0.017	0.648 ± 0.022
Testing Accuracy	0.646 ± 0.003	0.650 ± 0.003

TABLE XII
XGBOOST RESULTS 15 GAME AVERAGE

E. Artificial Neural Networks

In this paper ANNs only contained a single hidden layer, as initial hyperparameter testing consistently chose smaller networks. There was one output node with the sigmoid activation function so that the binary classification can be done. The number of hidden nodes was tuned as a hyperparameter, as well as the learning rate, and optimizer. As can be seen in tables XIII, XIV, and XV, the highest prediction accuracy was the validation set of the 15 game dataset, on the regularized networks. This model was then chosen as the best model we could find. This regularization yielded a full percentage point more accuracy than the unregularized networks which indicates that the regularization did improve performance. The testing accuracy of 65% is in line with the results that we have seen in previous literature. Next steps to improve the accuracy will be outlined in the Future Work section.

	Unregularized	L1+ L2 Regularization
Training Accuracy	0.641 ± 0.003	0.637 ± 0.014
Validation Accuracy	0.646 ± 0.027	0.631 ± 0.035
Testing Accuracy	0.619 ± 0.003	0.614 ± 0.012

TABLE XIII
ANN RESULTS 5 GAME AVERAGE

	Unregularized	L1+ L2 Regularization
Training Accuracy	0.651 ± 0.002	0.653 ± 0.002
Validation Accuracy	0.647 +- 0.010	0.651 ± 0.017
Testing Accuracy	0.641 ± 0.002	0.639 ± 0.002

TABLE XIV
ANN RESULTS 10 GAME AVERAGE

	Unregularized	L1+ L2 Regularization
Training Accuracy	0.652 ± 0.002	0.661 ± 0.002
Validation Accuracy	0.650 ± 0.014	0.661 ± 0.011
Testing Accuracy	0.642 ± 0.002	0.652 ± 0.003

TABLE XV
ANN RESULTS 15 GAME AVERAGE

F. Overall Comparison

This section deals with the overall goal of predicting the outcome of NBA games. Thus there will be two parts to this decision: which number of games of averaged data performed the best for each model, and then compare the best results of each model to find the best model.

This first step can be done by examining the tables with the results. It was found in this research that the data that was averaged over 15 games had the highest accuracy, with models that were regularized. This makes sense, 15 games would be enough for any anomaly-like games, when the team plays extremely well or extremely bad, to be smoothed out by the other games. The anomaly games would not affect the overall prediction capabilities of the algorithm. It also stands to reason that the regularized models performed the best as this is exactly why regularization was applied, to improve performance.

In Figure 2 the top accuracies are plotted against each other, with all models using the 15 game average data. In this plot, we can see that our baseline model with regularization applied was the one that performed the best. Further statistical testing could be performed to determine whether the means or medians of the other approaches is different than the baseline, but for now we are only comparing average and standard-deviation. From this comparison we find that the average and standard-deviation of the LR approach is higher, and smaller than the rest of the approaches. SVM and ANN do come extremely close to LR, with XGBoost coming next. Performing the worst in our testing was Random Forests.

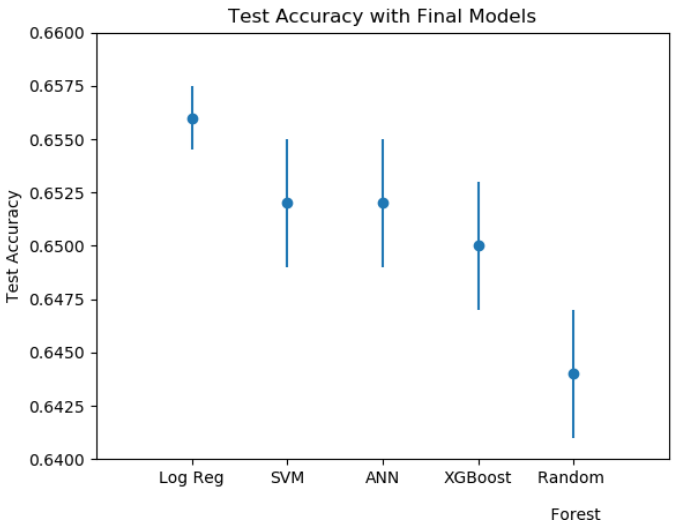

Fig. 2. Summary of best result from each model tested

V. DISCUSSION & FUTURE WORK

With the models finalized, they were run on the test dataset 10 times to get the average accuracies. LR had the highest accuracy on the test data, with SVM and ANN next. XGBoost and RF were fourth and fifth, respectively. Looking at the training and validation accuracies we can see that the SVM and NN have low variance, meaning that the models are suffering from bias. However the decision tree based models, XGBoost and Random Forest, suffered from high variance and overfitting. Unfortunately using the more complex models did not prove to decrease the bias that the logistic regression model was suffering from. This confirms what was speculated earlier, that the feature selection is what is limiting the prediction accuracies for this problem. We were able to achieve accuracies close to what was seen in the literature so this remains a consistent problem. We think that because the LR model performed similarly before and after feature selection, that we have an acceptable amount of variance in the model. This is similar across the other models as the training and validation accuracies are close for most models. However, we believe that there is a high bias that is prevalent in our models because of the drop in testing accuracy across SVM, ANN, XGBoost, and RF.

The limits of this research could be based in the nature of NBA games. This can be explained from different perspectives; we could be limited by the constantly evolving strategies to play, there may be more games that are considered outliers than we expect, or we may not have chosen the right data. In the first scenario, we could examine the strategies taken by NBA teams to win games. Since the 2014-2015 season there has been a large emphasis on shot selection, particularly three-point attempts and layups. Changes such as these may not be fully reflected in the statistics because of the time period of which the data falls in. In our research we used the full dataset from 2012 to 2018 which may be too large of a time series to extract a signal from. Instead, we could have partitioned the data in X year chunks for the algorithm to train and test on. Alternatively we could have not been as aggressive in the feature selection process. Selecting a different number of features may have included the features that included the signal we were searching for. We also could have combined the team data with the player level statistics to try and increase the capabilities of the model(s).

The next limitation could be that NBA games contain more outliers than we were expecting. There could be games included in the dataset where the winning teams statistics do not conform to what our model learns to be a winning team, and yet the team does win. This could hurt the capabilities of

our model.

These limitations lead us to what we could do in the future to further this research. We could follow the aforementioned method of partitioning data into X number of years, or even performing a rolling window set of predictions. We could also examine training on one season, and testing on the next. This would allow for season by season changes in playing styles to be a more pronounced signal in the data. Otherwise we could train on the regular season and test on the post season. We can also increase the rolling average number to 20 or even 25 games.

Lastly, we could apply different kinds of models to this problem. In our models, there is no memory of previous games between the two teams. A model that could be used to have some sort of memory is a recurrent neural network or a long short term memory model. These models could allow for a favourable matchup to become even more favourable for the model.

VI. Implementation & Code

The code used to do each of the steps in this work, data engineering, feature selection, normalization, hyperparameter tuning, and final tests was implemented in Python. Data engineering was completed using a combination of Pandas and Numpy, where data was loaded using Pandas, and a Numpy list was created that stored the merged data. Feature selection and data normalization was completed using scikit-learn, namely the SelectKBest, Normalizer, and f_classif functions. These steps were performed once the data engineering steps were completed. Finally the data was saved using Pandas. Algorithm implementations were done using scikit-learn, the XGBoost library, and Keras for the ANNs. This allowed us to focus more on the problem at hand rather than ensuring that our implementation was correct. Hyperparameter tuning was performed using the Optuna package. This package allowed us to create numerical or categorical options for the package to select from using Python syntax. Therefore individual parameters of each algorithm can have their own range of values to select from. Once this was completed for each algorithm, non-regularized and regularized, the final tests were completed. Once again these tests were completed using scikit-learn or Keras, depending on the algorithm.

References

[1] R. Bunker, T. Susnjak, "The Application of Machine Learning Techniques for Predicting Results in Team Sport: A Review," Pre-print submitted to arXiv, https://arxiv.org/pdf/1912.11762.pdf.

[2] P. Vračar, E. Štrumbelj, and I. Kononenko, "Modeling basketball play-by-play data," Expert Systems with Applications, vol. 44, pp. 58–66, 2016.

[3] A. Fayad, "Building My First Machine Learning Model: NBA Prediction Algorithm," Medium, 12-Jul-2020. [Online]. Available: https://towardsdatascience.com/building-my-first-machine-learning-model-nba-prediction-algorithm-dee5c5bc4cc1. [Accessed: 30-Sep-2020].

[4] K. Puranmalka, "Modelling the NBA to Make Better Predictions," Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2013, https://books.google.ca/books?id=hSOmoAEACAAJ.

[5] O. Hubáček, G. Šourek, and F. Železný, "Exploiting sports-betting market using machine learning," International Journal of Forecasting, vol. 35, no. 2, pp. 783–796, 2019.

[6] Z. Shi, S. Moorthy, A. Zimmerman, "Predicting NCAAB match outcomes using ML techniques - some results and lessons learned," Pre-print submitted to arXiv, https://arxiv.org/pdf/1310.3607.pdf.

[7] HAGHIGHAT, Maral; RASTEGARI, Hamid; NOURAFZA, Nasim. A Review of Data Mining Techniques for Result Prediction in Sports. Advances in Computer Science : an International Journal, [S.l.], p. 7-12, nov. 2013. ISSN 2322-5157.

[8] DOMINGOS, Pedro. "A unified bias-variance decomposition." Proceedings of 17th International Conference on Machine Learning. 2000.

[9] Mills, Brian M. and Salaga, Steven (2011) "Using Tree Ensembles to Analyze National Baseball Hall of Fame Voting Patterns: An Application to Discrimination in BBWAA Voting," Journal of Quantitative Analysis in Sports: Vol. 7: Iss. 4, Article 12.

[10] https://www.kaggle.com/abharg16/predicting-epl-team-season-win-percentages