Predicting NBA Single Season Per Game Statistics

Yang Yi, Levi Kaster, Allen Huang, Patrick Norrick
CSE 543T
5/09/2022

Abstract

In this report, we apply several learning paradigms to NBA player data. Our goal is to investigate and compare the predictive power of these models when applied to data on an individual player's points scored, total rebounds, and total assists in a given season. In comparing a k-Nearest-Neighbors model, a Linear Regression model, a Random Forest model, an Artificial Neural Network mode, and a Dense Neural Network Model, we find that the Dense Neural Network is most effective at predicting player statistics based on previous season data. Suggestions for further research are given.

1. Introduction

Over the past twenty years, the sports industry has become dominated by data. Analysts, coaches, scouts, and fans all utilize the seemingly unlimited data from professional and amature athletics to make predictions about the future outcomes of sporting events. The prospect of using data to make more accurate predictions has immense monetary and entertainment value, as the sports betting market alone is expected to be worth $140.26 billion by 2028 (Bloomberg). Algorithms which allow for accurate prediction while minimizing time and space complexity, therefore, will play a large role in determining which players get drafted, how rosters are created, how sports are broadcast, and how betting takes place. Over the next several decades, it is possible that the industry will record and publish enough player-data and in-game data to allow for highly sophisticated prediction given the appropriate learning techniques.

Specifically, we are interested in prediction within the NBA. There is currently a debate within the sport as to whether traditional human-based prediction is more accurate than machine-learning based approaches. Our goal, therefore, is to evaluate and compare several machine-learning models using data on points scored, assists, and rebounds. We collect and analyze NBA player statistics from 1980-2021 to employ and evaluate the efficacy and efficiency of several models including k-nearest neighbor, neural networks, linear regression, and random forest. Our target variables include the number of points scored, rebounds made, and assists made in a given season by a player given data on their previous performance.

2. Related Theory and Practice

A majority of the current research within the application of sports-prediction falls into one of two categories. Some researchers, along with many professionals within the industry, are interested in using classification problems applied to a given team. For example, Cheng, Zhang, Kyebambe, & Kimbugwe (2016) use the maximum entropy principle to predict which team will win the NBA playoffs. In doing so, they find that the NBAME (NBA Maximum Entropy) model that they construct does a fairly good job of predicting the outcome of NBA games. They indicate, however, that the prediction of the outcome of NBA games, particularly playoff games, can be very difficult because “there are many unforeseeable factors such as the relative strength of either team, the presence of injured players, players’ attitudes, and team managers’ operations that determine the winner or loser” (Cheng, Zhang, Kyebambe, & Kimbugwe, 2016). Other team-based classification problems include determining whether a team will make the playoffs in a given year, and whether or not a team will gain a certain player in the off-season.

The other type of problem sufficiently covered in the literature are team-based regression problems. For example, a prominent question within the industry is predicting how many points a given team will score in a game. If one could algorithmically predict this with a high degree of accuracy, he or she would be able to profit greatly through well-placed bets. Because the challenges and potential reward for accurate prediction in this domain is not restricted to the NBA, much of the research comes from other sports. For example, Wiseman compares various regression models in predicting winning PGA scores. In particular, he looks at linear regression, neural network regression, Bayesian linear regression, decision forest regression, and boosted decision tree regression to determine which is most useful and accurate in predicting PGA scores. In doing so Wiseman found that linear regression and Bayesian linear regression were the two top-performing algorithms, with validation scores indicating that these models could predict the winning score could be predicted within three points 67% of the time. Wiseman does, however, indicate that feature selection plays a decisive role in determining whether a specific model will be useful in predicting points scored.

Similarly, Maszczyk, Gołaś, Pietraszewski, Roczniok, Zając, & Stanula (2013) use data from javelin throws to test the efficacy of neural network and nonlinear regression models for predicting the distance thrown in competition. They find that "the created neural models offer much higher quality of prediction than the nonlinear regression model (absolute network error 16.77m versus absolute regression error 29.45 m)" (Maszczyk, Gołaś, Pietraszewski, Roczniok, Zając, & Stanula, 2013). The authors conclude that neural models are superior to regression models when applied to sports-related data, yielding better optimization properties. Our goal, therefore, is to test this claim by regression models, neural models, and nearest-neighbor models to an individual-based regression problem using NBA data. While the literature explores classification and team-based regression, very few have looked at the efficacy of these models in predicting how individual players will perform over the course of an entire season. We use data on player history to predict how many points a player will score in a given season, how many assists they will make, and how many rebounds they will make.

Of course, this type of problem poses challenges that other researchers do not face. For example, it is possible that a player for whom we would like to predict the number of points scored next year was injured for half of this season, thus biasing the prediction downwards. We follow the advice of Bunker & Thabtah (2017), who provide analysis and suggestions for applying machine-learning models to general sports-related data. For instance, they stress the importance of incorporating domain knowledge when choosing and evaluating a model. They additionally suggest that a held-out training test split provides a better way of performing model testing for sports data than traditional validation, as the order of the data is preserved.

Because our data comes from the NBA, there is likely a large amount of parity across players. It is natural, therefore, to consider applying similarity based models when trying to predict how a player will perform in a given season. As such, we apply a K-Nearest-Neighbors model to our dataset when predicting points scored, total rebounds, and total assists. Given complexity and nonlinear nature of the problem at hand, and following the suggestions of Maszczyk, Gołaś, Pietraszewski, Roczniok, Zając, & Stanula (2013), some form of a neural-network model is a likely candidate for success in individual-based regression prediction within the NBA. We also test a linear regression model so as to have a simple model to be used as a baseline for comparison. Finally, we use a random-forest model with bagging to see whether aggregating diverse weak-learners provides us with accurate predictions.

3. Technical Details

3.1 Dataset Collection and Pre-Processing

In order to accomplish the goal of predicting single season per game points, rebounds and assists, we treated our problem like three different regression problems that share the same input features. As such, each data point in our dataset represents per game statistics about a player, with the 3 target features being the points, rebounds and assists data for that same player the following year. Additionally, although our dataset includes 3 target features, we only use one at a time when performing the regression. This means that the only features used in making predictions are the input features from the season prior to the season we are trying to make predictions about.

The final dataset used in our analysis contained 29 input features, chosen because of accessibility of the data, and common usage of these features in current NBA Analysis. Similar features have been used in making predictions at the single game level (Cheng, Ge, et al., 2016). These features used can be found below in **Table 1.**

Feature	Abbr.	Feature	Abbr.	Feature	Abbr.
Player Name	Player	3 Point Attempts	3PA	Defensive Rebounds	DRB
Position	Pos	3 Point Percentage	3P%	Total Rebounds	TRB
Age	Age	2 Pointers Made	2P	Assists	AST
Team	Tm	2 Point Attempts	2PA	Steals	STL
Games Played	G	2 Point Percentage	2P%	Blocks	BLK
Minutes Played	MP	Effective FG%	eFG%	Turnovers	TOV
Field Goals Made	FG	Free Throws Made	FT	Personal Fouls	PF
Field Goal Attempts	FGA	Free Throw Attempts	FTA	Total Points	PTS
Field Goal Percentage	FG%	Free Throw Percentage	FT%	Year	Season
3 Pointers Made	3P	Offensive Rebounds	ORB		

Table 1: This table shows the input features used in our dataset, and the abbreviations they were given. Note that each of these numerical features are per game averages for the year.

The data used for our analysis was obtained from www.basketball-reference.com, which is a widely used site that has various compiled basketball statistics dating back to 1946. The data was gathered through web scraping into a pandas dataframe, using the BeautifulSoup python package and code inspired by the article "Web Scraping NBA Stats" (Sanchez, O.). This raw data included the per game single season statistical information of all NBA players from the years 1980 to 2021.

After this, data cleaning was performed. All columns not included in the 29 input features from **Table 1** were dropped, and then all data points that did not have a 'Games Played' of at least 41 were dropped. This meant that each player in our data played at least half of the season, which was done to reduce the effect that any injuries might have on our results. Finally, the points, rebounds and assist averages of each player were added to previous season's data for that same player. These 3 new columns were given the names 'Target_PTS', 'Target_TRB', and 'Target_AST'. After cleaning, the final dataset that we used contained 10,041 data points and 32 total features, with 29 input features and 3 target features.

3.2 K-Nearest Neighbor Regression Algorithm

The first algorithm that we tested on our dataframe was the K-Nearest Neighbors (kNN) Algorithm. We initially split the dataset, with 85% of the data going to the training set and 15% to the test set, before performing hyperparameter tuning on the training data to improve optimization quality. Of greatest importance to the kNN algorithm is choosing a correct nearest neighbor (NN) value, so we set out to find the value that would lead to the highest R^2. We chose R^2 as the metric used to determine the success of the model because of ease of interpretation, and because it is a common metric used in analyzing the success of regression predictions (Palmer PB, O'Connell DG, 2009). For each NN value 1 through 25, a KFolds cross regression was performed to find the best number of NNs for predicting each of the 3 target features (PTS, TRB and AST).

Once the number of nearest neighbors had been chosen, we performed dimensionality reduction on the data to improve the efficacy and runtime of the algorithm. As such Principal Component Analysis (PCA) was performed to reduce the dimensionality of our data, since this dimensionality reduction technique has successfully been used for kNN regression in the past, although to our knowledge we are the first to apply this method to sports prediction (Li Tang, Pan & Yao, 2018). Finally, once PCA had been applied to the training and test data, the model was trained and predictions for the test set were generated. Again R^2 was used for determining the success of the model in prediction PTS, TRB, and AST.

All analysis and functions used in this step were carried out using functions from the sklearn library in python, specifically the KNeighborsRegressor() function was used to run the kNN model.

3.3 Linear Regression Algorithm

Next, a linear regression algorithm was used to predict the PTS, TRB, and AST. We chose to include linear regression so that we could compare it with other studies, and because previous research has found linear regression to be successful in sports predictions, such as in Wiseman's study on predicting PGA winning scores (Wiseman, 2016). Like with the kNN algorithm, 85% of the data was split into the training set and 15% was put into the test set.

After this, recursive feature elimination with cross-validation was performed to select the features that should be included in the input data, which was carried out using sklearn's RFECV() function. Once this feature elimination had been performed the model was trained with the selected features and the predictions were generated. Like with the kNN algorithm, the R^2 value of the predictions and ground truth was calculated for determining model success. Additionally, predictions and R^2 values were also generated using linear regression models without the recursive feature selection having been performed, so that the influence of the feature selection could be seen.

All analysis and functions used in this step were carried out using functions from the sklearn library in python.

3.4 Random Forest Model

The next algorithm we applied was the Random Forest Model, which is robust to overfitting and works well with categorical data from its nature of decision trees. We decided to explore the Random Forest Model since it has been a traditional competitor with linear/logistic regression models (Farhadian, 2018).

We split 80% of the data into the training set and the remaining 20% into the test set. Then we specified three targets dataset and included the rest data as the input. This procedure was done for both the training set and test set. Our next step is to fit and predict the data with different criterias and. Once the model has been established, we generated R^2 for both the training set and test set to emphasize the quality of the prediction. Finally, we generated OOB error and testing error for a broader evaluation of the prediction.

All analysis and functions used in this step were carried out using functions from the sklearn library in python.

3.5 Neural Network Models

The Neural Network Models were the final algorithm we explored. First we used an ANN(Artificial Neural Network) model to predict the three criterias. Previous research clearly indicated that the neural network model is much more advantageous than non-linear models (Bunker, 2017). Since we performed a linear regression model, a comparison with the neural network model would be meaningful.

Similarly with the random forest model, 80% of the data was split into the training set and 20% was split into the test set. We have three specified target dataset and rest data as the input for both the training set and test set. Next, the data was scaled so that each feature is scaled to the given range of the training data. After that, the data was trained and predicted for each criteria. Just like all previous algorithms, R^2 was selected as the sign of success of the prediction for each feature(PTS, AST, and TRB).

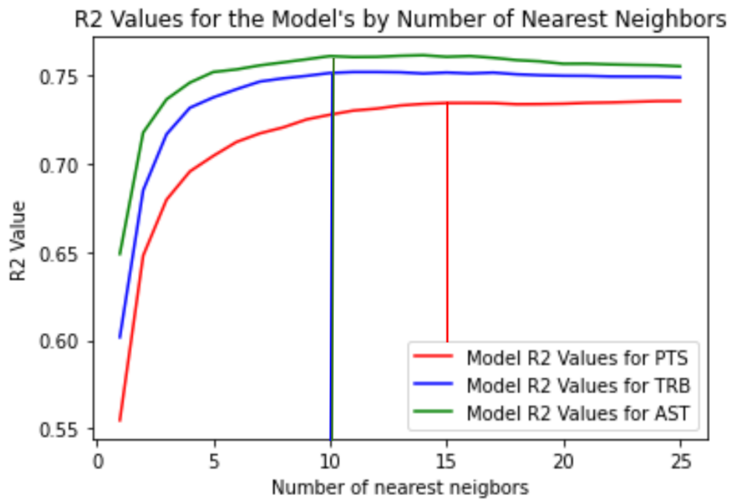
We conducted this part of the analysis in two steps. First, we used the Multi-layer Perceptron regressor from the sklearn library in python to investigate artificial neural networks. Then, we used the Keras API in TensorFlow to investigate dense neural networks (DNNs). Google Colab was used for this purpose.

In order to run the DNN part of this analysis, the Player, Tm, and Pos columns were transformed into numerical data with one-hot encoding. The sequential model was constructed with a normalization layer, 2-3 dense layers, and an output layer with all three labels. Optional dropout layers were placed between dense layers. The hyperparameters, including the number of units in each dense layer, their activation function, the presence of dropout layers, and the learning rate, are tuned with KerasTuner. The model was optimized with an adam optimizer. Initially, the model was trained on 100 epochs, but with the EarlyStopping class the more optimal number of epochs were determined to be around 8 epochs.

4. Experimental Results

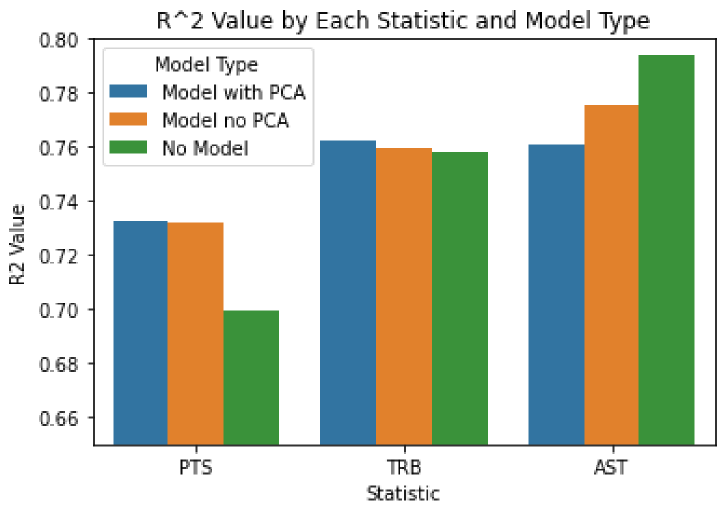
4.1 K-Nearest Neighbor Results

Graph 1 below shows the results of our hyperparameter tuning to find the best number of nearest neighbors for predicting PTS, TRB and AST using the kNN regression model. To choose this best number of NN, we chose the location where the graph appears to peak and flatten out. As seen below, this corresponds to a value of 10 NNs for predicting TRB and AST, and 15 NNs for predicting PTS.



Graph 1: Graph displaying R^2 values for predictions across multiple numbers of NNs. The chosen number for each model is marked with vertical lines.

Once the number of nearest neighbors was chosen and PCA had been applied, the model was used to generate PTS, TRB and AST predictions for the test set and the R^2 value was calculated for each of these to demonstrate the accuracy of the predictions. Additionally, R^2 values were calculated for running the model without PCA, and for just using the previous season's points as our predictions instead of running a model, which we have termed as the 'No Model' method. These values can be seen summarized below in **Graph 2.** As seen below, running the model with PCA led to slightly better results than the other methods when predicting both PTS and TRB, leading to R^2 values of .732 and .762 respectively. Additionally we see that the model with PCA performed better than the 'No Model' method, with a value of .732 compared to .700; however, in predicting TRB the difference was negligible. The kNN model with PCA performed much worse than the other methods for predicting AST, leading to a R^2 value of .761 compared to values of .775 and .794 when the model was run without PCA and when no model was used. From this we can say that while the kNN model was relatively successful in predicting points when compared to not using a model, it was not successful at predicting TRB or AST. Given the relatively low success of this kNN model, other machine learning methods are likely better for predicting PTS, TRB, and AST than kNN Regression.

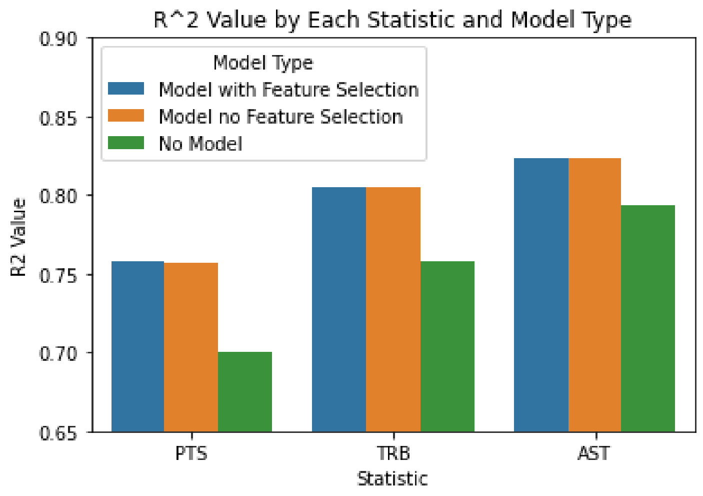


Graph 2: Bar graph displaying the R^2 values of the kNN model with PCA, the kNN model with no PCA, and the No Model method.

4.2 Linear Regression Results

Following the steps outlined in 3.3, recursive feature elimination with cross validation was performed after the train-test split. This was done separately for predicting PTS, TRB and AST, resulting in 3 different sets of features. This feature selection resulted in 5 dropped features for predicting points, 1 dropped feature from predicting TRB, and 2 dropped features for predicting AST. Using these newly selected features the predictions were generated and the R^2 scores were calculated. Predictions and R^2 values were also generated for the linear regression model without feature selection having been done, and for the 'No Model' method which just uses the previous season PTS, TRB, or AST for predictions. These results are displayed in **Graph 3**.

As seen in the graph below, there were essentially no differences between the R^2 values of the linear model with the feature selection and the linear model without feature selection. In fact, both models had R^2 values that were within .001 of each other for predicting all 3 target features. Despite there being no differences between the model with and without feature selection, there were noticeable differences between the linear regression models and the 'no model' method for each of the target features. We see that the linear regression models performed much better than the 'no model' method when predicting points, with a R^2 value of .757 compared to .700. This same trend is true for predicting TRB and AST, as we see that the linear regression model results in R^2 increases from .758 to .805 for TRB, and from .794 to .823 for AST when compared to the 'no model' result. From these results we see that across the board the linear regression model outperforms the use of no model, and as a result we can say that linear regression is a reasonable method for predicting each of PTS, TRB, and AST from previous season stats.

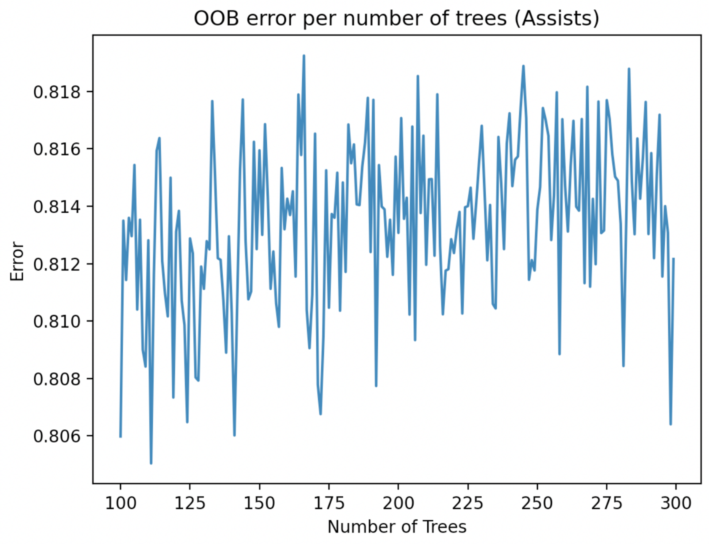


Graph 3: Bar graph displaying the R^2 values of the linear model with feature selection, the linear model with no feature selection, and the No Model method.

4.3 Random Forest Results

Following the procedures described in 3.4. We generated R^2 values for both the training set and the testing set. For the training set, the resulting R^2 values are .963 for PTS, .970 for TRB, and .973 for AST. Noticing that these R^2 values all approach 1, which indicates that the majority of the predictions are accounted for and the fitting was extremely successful. Moreover, our generated R^2 values for PTS, TRB, and AST for the test set were .748, .764, and .822 respectively. Finally, the OOB error and testing error were performed. The OOB errors for these three features are all pretty close to each other(.741, .791, and .808) and are slightly below 1. The testing errors for TRB(1.15) and AST(.808) are moderate and only PTS has relatively high testing error(9.18).

In addition, it is worthwhile to mention that the OOB error does not change much with the number of decision trees we choose, which can be demonstrated by **Graph 4** which is an example of the AST OOB error relative to the number of trees.



Graph 4: OOB error relative to number of decision trees

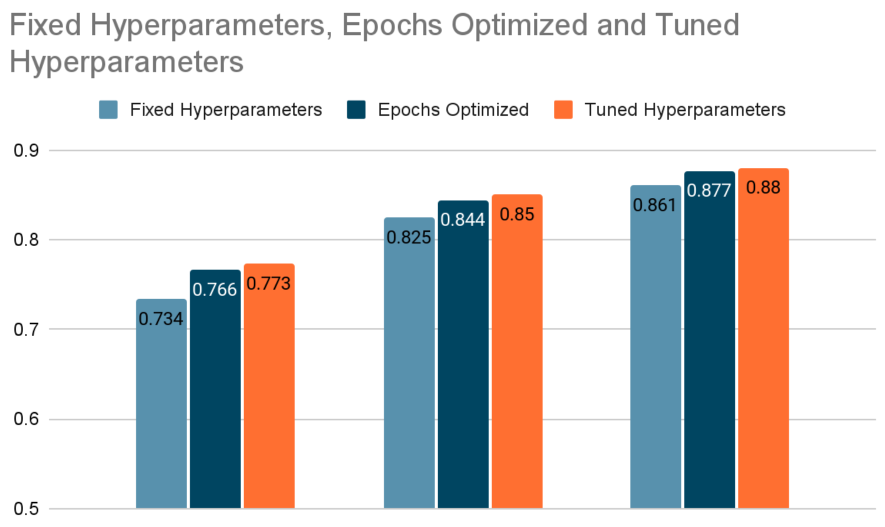
4.4 Neural Network Results

In the first part of our analysis using sklearn to investigate the ANN, we split and scaled our data for modeling and prediction. The modeling used 10000 epochs, meaning that each data point will be used 10000 times in the learning process. In addition to that, we applied the adam solver which is a stochastic gradient-based optimizer for weight optimization so that the coefficient of determination of the modeling would be optimal. Afterwards, the R^2 values are generated after fitting and predicting our data, leading to a decent result of .735 for PTS, .795 for TRB, and .814 for AST.

The second part using DNN in TensorFlow was conducted in a few stages. Initially, the hyperparameters were fixed to obtain a proof-of-concept baseline performance metric. Two dense layers of 100 and 32 units respectively with ReLu activation functions were employed, and the model trained for 100 epochs. The generated predictions had an R^2 value of 0.734 for PTS, 0.825 for TRB, and 0.861 for AST.

Because of the large number of data entries, we were aware that the high epoch count may lead to overfitting, so we optimized the number of epochs with the EarlyStopping class. Subsequent runs also employed the class. The optimal epoch count in this otherwise unchanged configuration was found to be around 9, yielding an R^2 value of 0.766 for PTS, 0.844 for TRB, and 0.877 for AST.

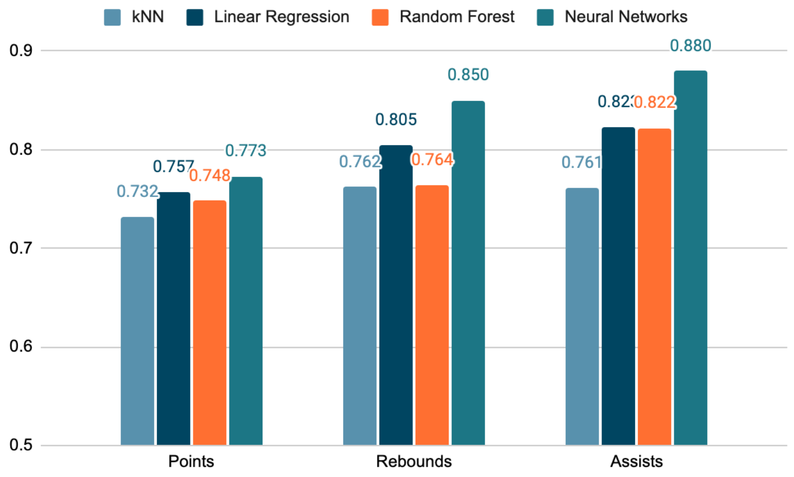
The KerasTuner was used to configure the remaining hyperparameters. The Hyperband algorithm with mean squared logarithmic error loss function was used to initialize the tuner, which used 20% of the training set as validation for each run. The tuner was executed multiple times with 2 and 3 dense layers with similar results in terms of model performance. A typical model with tuned hyperparameters had a dense layer with 480 units using ReLu, a dropout layer with 0.25 dropout rate, a dense layer with 320 units using tanh, and an output layer. The resulting R^2 values are 0.773 for PTS, 0.850 for TRB, and 0.880 for AST. Comparing these results from the DNN to ANN from earlier, we see that the DNN was more successful than the ANN due to its higher R^2 values.



Graph 5: Bar graph of R^2 values of DNN with fixed hypermarametes, only epoch optimized, and tuned hyperparameters

5. Discussions, Observations, and Comparisons

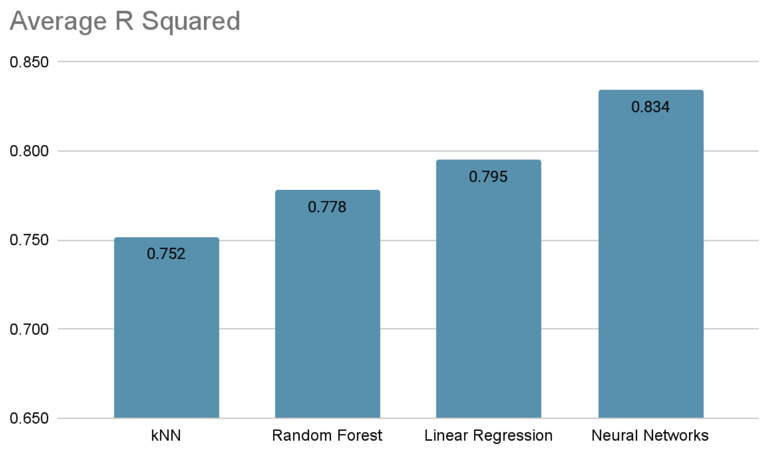
Compiling the results from our kNN, Linear Regression, Random Forest, and Neural Networks tests, **Graph 6** shows the general trend from our experiments. While maintaining only a small lead on the Points variable, Neural Networks enjoy bigger leads in terms of predicting power in Rebounds and Assists. Linear Regression holds the second place in all three variables.



Graph 6: Bar graph of R^2 values of four method's predictions on Points, Rebounds, and Assists. We graph the result of just the DNN here because of its success over the ANN.

Our result shows that in the case of predicting individual NBA player stats, Maszczyk, Adam, et al.'s preferred Neural Networks model performs the best compared to kNN, Random Forest, and Linear Regression. While the models tested cover a wide range and are reasonably optimized, there are more models available and more hyperparameter tuning that could be done. Notwithstanding, Neural Networks, especially Dense Neural Networks used in our testing, did show a significant advantage over other models tested.

6. Recommendations



Graph 7: Bar graph of average R^2 values over all three variables in ascending order

As shown in **Graph 7**, when it comes to sports-related or similarly formatted data, Neural Networks is the regression algorithm to be tried first and compared against. More algorithms to consider may include Support Vector Machines, Gaussian Regression, and Lasso Regression. On the Neural Networks front, apart from further hyperparameter tuning, models other than Dense Neural Network can be explored. One such model of note is Long Short-Term Memory (LSTM): the NBA data entries contain players' performance over time, from which LSTM may be more able to extract information.

7. Contribution of Team Members

Patrick Norrick performed the literature review of relevant studies, and used his findings to propose comparing the k-NN, Random Forest, Linear Regression, and Neural Network algorithms for predicting the single season per game points, rebounds and assists. Additionally, Patrick wrote the 'abstract', 'Introduction', and 'Related Theories and Practice' sections of this paper. Levi Kaster performed the data collection and preprocessing, and wrote the code for the k-Nearest Neighbor and Linear Regression models. He also wrote the corresponding sections in the 'Technical Details' and 'Experimental Results' portions of the paper and created the software package in github. Allen Huang designed and ran the Random Forest and Artificial Neural Network models, and wrote the corresponding sections in the 'Technical Details' and 'Experimental Results' portions of the paper. Yang explored using Dense Neural Networks on TensorFlow on our dataset, reported his results, and compared the results of all our models. Finally, he also wrote the 'Discussions, Observations, and Comparisons' and 'Recommendations' portions of this paper.

8. References

Bunker, Rory P., and Fadi Thabtah. “A Machine Learning Framework for Sport Result Prediction.” *Applied Computing and Informatics*, vol. 15, no. 1, 2019, pp. 27–33., https://doi.org/10.1016/j.aci.2017.09.005.

Cheng, Ge, et al. “Predicting the Outcome of NBA Playoffs Based on the Maximum Entropy Principle.” *Entropy*, vol. 18, no. 12, 2016, p. 450., https://doi.org/10.3390/e18120450.

Li Tang, Heping Pan & Yiyong Yao (2018) PANK-A financial time series prediction model integrating principal component analysis, affinity propagation clustering and nested k-nearest neighbor regression, Journal of Interdisciplinary Mathematics, 21:3, 717-728, DOI: 10.1080/09720502.2018.1456825

Maszczyk, Adam, et al. “Application of Neural and Regression Models in Sports Results Prediction.” *Procedia - Social and Behavioral Sciences*, vol. 117, 2014, pp. 482–487., https://doi.org/10.1016/j.sbspro.2014.02.249.

Palmer PB, O'Connell DG. Regression analysis for prediction: understanding the process. Cardiopulm Phys Ther J. 2009;20(3):23-26.

Sanchez, O. . Web scraping NBA stats. Medium. 2019, March 18, from https://medium.com/@osanchez2323/web-scraping-nba-stats-4b4f8c525994

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

“Sports Betting Market Size Worth $140.26 Billion By 2028: Grand View Research, Inc.” *Bloomberg.com*, Bloomberg, 19 Oct. 2021, https://www.bloomberg.com/press-releases/2021-10-19/sports-betting-market-size-worth-140-26-billion-by-2028-grand-view-research-inc.

Sports Reference LLC "NBA Player Stats: Per Game." Basketball-Reference.com - Basketball Statistics and History. https://www.basketball-reference.com/.

Wiseman, Oisin. “Using Machine Learning to Predict the Winning Score of Professional Golf Events on the PGA Tour.” 2016, *National College Of Ireland*.

Farhadian, M., Torkaman, S. & Mojarad, F. Random forest algorithm to identify factors associated with sports-related dental injuries in 6 to 13-year-old athlete children in Hamadan, Iran-2018 -a cross-sectional study. *BMC Sports Sci Med Rehabil* 12, 69 (2020). https://doi.org/10.1186/s13102-020-00217-5