Predicting NBA Player Statistics Through the Use of Machine Learning

Morgan Gribbins, Doga Pamuklar, William Ritchie, Cameron Simbeck

May 6, 2022

Github Repository

1 Introduction

In recent years, machine learning has seen a rise in popularity due to many factors, including its predictive capabilities for real world scenarios. Projects vary in complexity from simple regression and classification models to complicated deep neural networks.

While simple in nature, regression models provide valuable insight into data that would otherwise be impossible to predict through human means and can give an advantage over competitors in certain markets. From day trading in the stock market to housing market trends, regression models can be surprisingly accurate even on such nuanced and multifaceted problems. We demonstrate this by utilizing both a linear model and a neural net to predict statistics of players in the National Basketball Association (NBA).

The rest of this paper is formatted in the following. First, it discusses the relevance of machine learning in Section 2. In Section 3 it then describes our implementation of both a linear model and a deep neural network. Section 4 discusses the results of our models as well as faults and potential improvements to be made to our project. Section 5 presents a final analysis of our work as well as our conclusions.

2 Background

Machine learning has risen dramatically in popularity in recent years due to its usefulness in solving many real world problems. At its core, machine learning provides a tool for classifying or predicting outcomes based on input data. This is achieved by using deep learning models, classified into either supervised, unsupervised, or reinforced learning. Due to the scope of the paper and our use of supervised models, we will skip over the specifics of the latter two.

Supervised models predict an expected output (such as the behavior of the stock market) by molding the model to mimic training data. Jordan et. al. states this best saying, "...the training data take the form of a collection of (x, y) pairs and the goal is to produce a prediction y* in response to a query x* ... supervised learning systems generally form their predictions via [this] learned mapping f(x)..." [1]. These models are provided a set amount of data to initially train with. Once the model has digested this data, it is then provided data it has never seen before (test data) to assess its accuracy. Loss functions are implemented to achieve this, quantifying the difference between the output values produced by the model and the actual labels from the dataset.

Input by input, the model is optimized by attempting to minimize the loss until a it is negligible and the model produces accurate results. While there are many ways to quantify loss, we chose to use Mean Square Error (MSE), which is defined as

$$MSE = \sum_{i=1}^{n}(y_i - y_{*i})^2,$$

for actual y and prediction y_*. This is the preferred loss function for linear regression, and it fits well for the format of our result vector — additionally, MSE is a fairly intuitive measure of difference, which makes our results easy to interpret.

The field of machine learning is rapidly expanding every day, with researchers developing new models that are increasingly efficient and effective for a variety of difficult to predict scenarios. However for our problem, simple linear regression models and deep neural networks are sufficient. A good source for foundational information can be found in Jordan et. al. [1], which discusses machine learning fundamentals and the applications thereof. Maulud et. al. [2] provides insight into linear regression models, of which our linear model was derived. Finally, the foundations for a regression neural net can be found in the work done by Specht [3].

3 Methodology

3.1 Data Preprocessing

For our subject, we chose to analyze the statistics from NBA players in seasons between the years 1996 and 2019. The player statistics dataset was obtained from Kaggle and contains 11,700 data entries. Each entry contains 22 features varying from draft results, physical qualities and game statistics for each player in the form of a comma-separated values (CSV) file.

Certain values in the draft year, draft round, and draft number were undefined due to some players not being drafted, therefore we set each missing value as the mean of said feature for that season of the draft. This ensures the effects of the fabricated values we are inserting are minimized. We then randomly partitioned our data set into test, training and validation sets of 80%, 10% and 10% respectively with different arrays for the input data and output labels.

We designated 13 features as independent variables (input neurons), and 7 as dependent variables (output neurons). These were chosen based on causality: As an example, height, weight and age should affect the amount of points a player scores, not the other way around. Five inputs are non-numerical and were converted to one-hot vectors, making our input neuron count greater than 13; each of these one-hot inputs have their own dense layer, giving a 13 neuron input layer.

We used the PyTorch library to create and tune our linear regression and neural network models. In order to process our dataset to feed into PyTorch, we used Pandas. Finally, to display our results, we used matplotlib.

Our variables are as follows:

Variable Name	Column Title	I/O	Description
Team Abbreviaton	team_abbrev	Input	Abbreviation of player's team
Age	age	Input	Age of player during the season
Player Height	player_height	Input	Height of player (cm)
Player Weight	player_weight	Input	Weight of player (kg)
College	college	Input	College player attended
Country	country	Input	Country of origin of player
Draft Year	draft_year	Input	Year player was first drafted
Draft Round	draft_round	Input	Round player was drafted
Draft Number	draft_number	Input	Draft number player was picked
Games Played	gp	Input	Games played during season
Season	season	Input	Season other statistics are pulled from
Usage Percentage	usg_pct	Input	Percentage of team plays performed by player while in the game
Points Per Game	pts	Output	Average points per game for season
Rebounds Per Game	reb	Output	Average rebounds per game for season
Assists Per Game	ast	Output	Average assists per game for season
Offensive Rebound Percentage	oreb_pct	Output	Percentage of available offensive rebounds secured by player while in game
Defensive Rebound Percentage	dreb_pct	Output	Percentage of available defensive rebounds secured by player while in game
True Shooting Percentage	ts_pct	Output	Percent of shots made factoring in freethrows and threes
Assist Percentage	ast_pct	Output	Percent of assists made (excluding free throws) while in game

3.2 Linear Regression Model

We initially tackled this problem with a simple, baseline linear regression model, which we trained with PyTorch's Adam optimization algorithm in order to minimize mean squared error loss on our linear model. To accomplish this, we ran 80% of our data through the model repeatedly until validation set loss plateaued out. Additionally, we trained and tested a deeper linear regression model with multiple linear layers in order to allow for more a more nuanced fit. The loss per epoch of training as well as a histogram of testing loss can be seen in figure **??**.

3.3 Neural Network Model

Our neural network model consists of 5 separate layers for string and string-like parameters, as well as 6 dense layers, 5 of which with a 20% dropout probability in order to combat over-fitting. Additionally, we use both tanh and ReLU for activation functions for our layers. Similarly to the linear model, we trained our neural network with PyTorch's Adam optimization algorithm and mean squared error loss in order to minimize validation set loss. For this data we again split our data into 80%, 10%, and 10% sized sets for training, validation, and testing. The loss per epoch of training as well as a histogram of testing loss can be seen in figure 2.

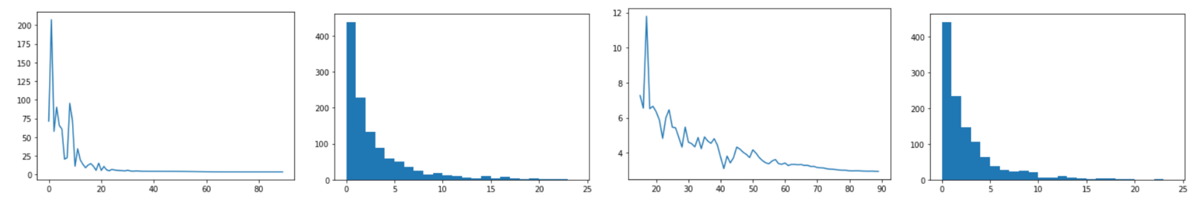

Figure 1: From left to right: Validation loss per epoch for linear regression, testing loss for linear regression, validation loss per epoch for deep linear regression, testing loss for deep linear regression.

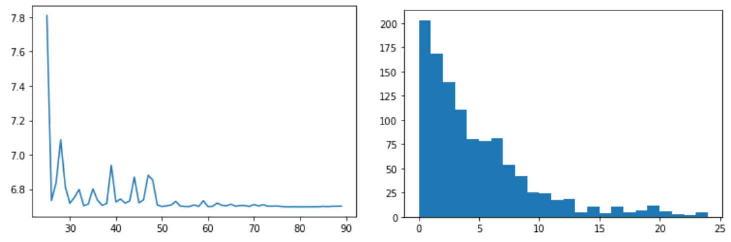

Figure 2: Validation loss per epoch (left) and test loss (right) for neural network.

4 Discussion

Our shallow linear regression model took approximately 2-3 minutes to train to full efficacy, with validation set loss plateauing (with an average 3 mean squared error) around 25 epochs in, while the deeper linear regression model took around 16 minutes or 90 epochs for training, with similar results between the two. Our neural network took significantly longer, and after 90 epochs it plateaued around 6.5 mean squared error. This is a slightly worse mean squared error, which is likely a result of the relatively small dataset for such a deep neural network.

These results show that a more complex model is not necessarily better. The model needs to fit the type and size of data and the complexity of the focused problem. For our data-oriented, relatively simple problem, linear regression suited best and the loss from linear regression was more than adequate. If we were able to hypothetically reduce loss with a more complex model, it would result in diminishing returns. Training a more complicated neural network may have too many negatives such as longer training times or power consumption costs.

5 Conclusion

Our results show that a simple problem required a simple machine learning model to best fit the data. The shallow linear model provided the lowest loss in the fewest amount of epochs, followed by the deep linear model and finally by the deep neural net. Regardless, all three models tested were able to predict statistics of NBA players with a low loss after 90 epochs.

As stated prior, we postulate our dataset was too small to produce accurate results for our neural network model. However, with the addition of significantly more data or a larger set of input neurons, a complex model like our neural network would likely surpass the efficacy of a linear regression model. However, as stated above, it may result in diminishing returns. Overall, our models provided useful insight with real world data.

References

[1] Jordan, Michael I., and Tom M. Mitchell. ”Machine learning: Trends, perspectives, and prospects.” Science 349, no. 6245 (2015): 255-260.

[2] Maulud, Dastan, and Adnan M. Abdulazeez. ”A review on linear regression comprehensive in machine learning.” Journal of Applied Science and Technology Trends 1, no. 4 (2020): 140-147.

[3] Specht, Donald F. ”A general regression neural network.” IEEE transactions on neural networks 2, no. 6 (1991): 568-576.