Michael Gagliardi
NBA Player Efficiency Rating Predictor
ECE 597ML
University of Massachusetts Amherst

1 Abstract

With sports betting recently becoming legal in Massachusetts, the demand for models that can accurately predict the outcome of sports events is at an all-high in this area. This project used a neural network to predict the player efficiency rating (PER) a player will score in a given game using statistics for previous games such as points, rebounds, assists, minutes played, and age. In doing so, the model will be able to accurately predict the PER a player will have, and in turn give some lucky bettors an upper hand in deciding a winner. Before the model is trained, the data was preprocessed to drop any unnecessary data and handle any missing values, as well as normalizing the numerical data. After selecting the appropriate features, the neural network consisting of three dense layers built using Keras and the Adam optimizer was trained on the training set of data. The MSE (mean squared error) between predicted and actual values for the model was then calculated, and cross-validation was used to tune the hyperparameters.

2 Introduction

The ultimate purpose of this model is to predict the PER by basketball players based on the various pre-selected features. This is a regression model that aims to understand the relationship between various basketball and personal (such as age) statistics and their overall efficiency, making predictions accordingly. This model could potentially be used to improve basketball teams as a method of performance evaluation. The general manager or coach of the team could use this model to play the players who will have the highest effect on the scoreboard. By understanding how efficient their starting lineup is and how efficient the opposition is expected to be based on various lineups, the coach can make changes on both the defensive and offensive end of the ball to hopefully sway the results of the game. This could involve double-teaming the opposing player that is expected to perform the best, or drawing up plays for the player on their team that is expected to have a really good game. In addition to that, they could use this data to help figure out which areas of the players' game needs the most work in order to improve their efficiency, i.e. if the player improves their court-vision in hopes of increasing the number of assists, it may lead to more open looks for that player down the road, increasing their number of points, which can improve their overall efficiency. Outside of the organization, this model can be used for sports bettors, hoping to have a better prediction as to which players will perform the best, placing their bets accordingly.

3 Dataset and Features

The dataset used is the "Seasons_Stats.csv" dataset from Kaggle (https://www.kaggle.com/datasets/drgilermo/nba-players-stats), which contains data for players from NBA seasons from 1950 - 2017. Before a model could be developed and trained, preprocessing was performed on the data. Some statistics were not recorded until later in the NBA's history, and as a result, any rows where data was not present was dropped for the sake of uniformity. In addition to that, unnecessary columns 'blanl' and 'blank2' were dropped as they had no data. The remaining features (besides 'Name', 'Team' 'Year' and 'Position') were all valuable numerical data, and as result they were all normalized due to the disparity of ranges (a player could average 40 minutes a game and 2 steals a game, those 2 steals are a lot more meaningful than the 40 minutes). The data set was then split into training and test sets, where the test set was about 20%, and a random seed value was used in order to reproduce. All features were used as input features, and the target variable of 'PER' was used as the output.

In short, the dataset was preprocessed by removing rows with empty values, dropping empty columns, and normalizing all numerical features. The model was then trained on this data set using the Keras Sequential API to construct it and also provide L2 regularization. In total, the model has 3 dense layers using the ReLU function as the activation function within the hidden layers, and a linear activation function as the output. To measure the efficacy of the model, the mean squared error was calculated. Using cross-validation with 3 folds, the model was able to identify the highest performing combination of epochs and batch size as part of the hyperparameter tuning.

4 Methods

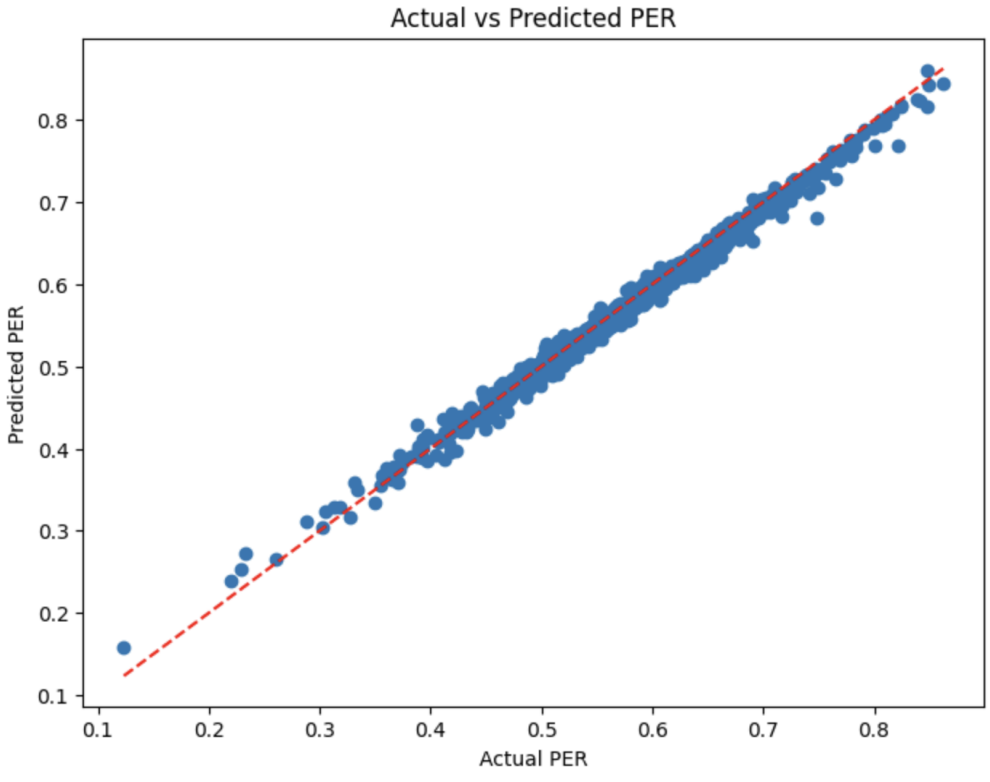
This model mainly uses backpropagation, a form of supervised learning that utilizes weight updates based off of the error between the hypothesis and predefined output. In this particular instance, backpropagation is implemented through the use of the Keras API as a part of the training process. In addition to that, the Adam optimizer is used to train the model. The Adam optimizer creates unique and adaptive learning rates for each parameter, which is meant to optimize the training further than what typical gradient descent is capable of. This optimizer uses the gradient history as well current gradients to adjust the learning rate of each parameter, placing a greater weight on the more recent gradients. This utilizes gradient momentum to accelerate the model's convergence as well as avoid vanishing gradients. The Keras API is used to create the model and train it, as well as implement L2 regularization (which uses the square magnitude of the lambda value) which avoids completely shrinking features since there are only 46 features that are being used.

This model's method can be described in three major steps: Forward Pass, Loss Calculation, and Backpropagation. During forward pass, the inputs are inputted into the neural network layer by

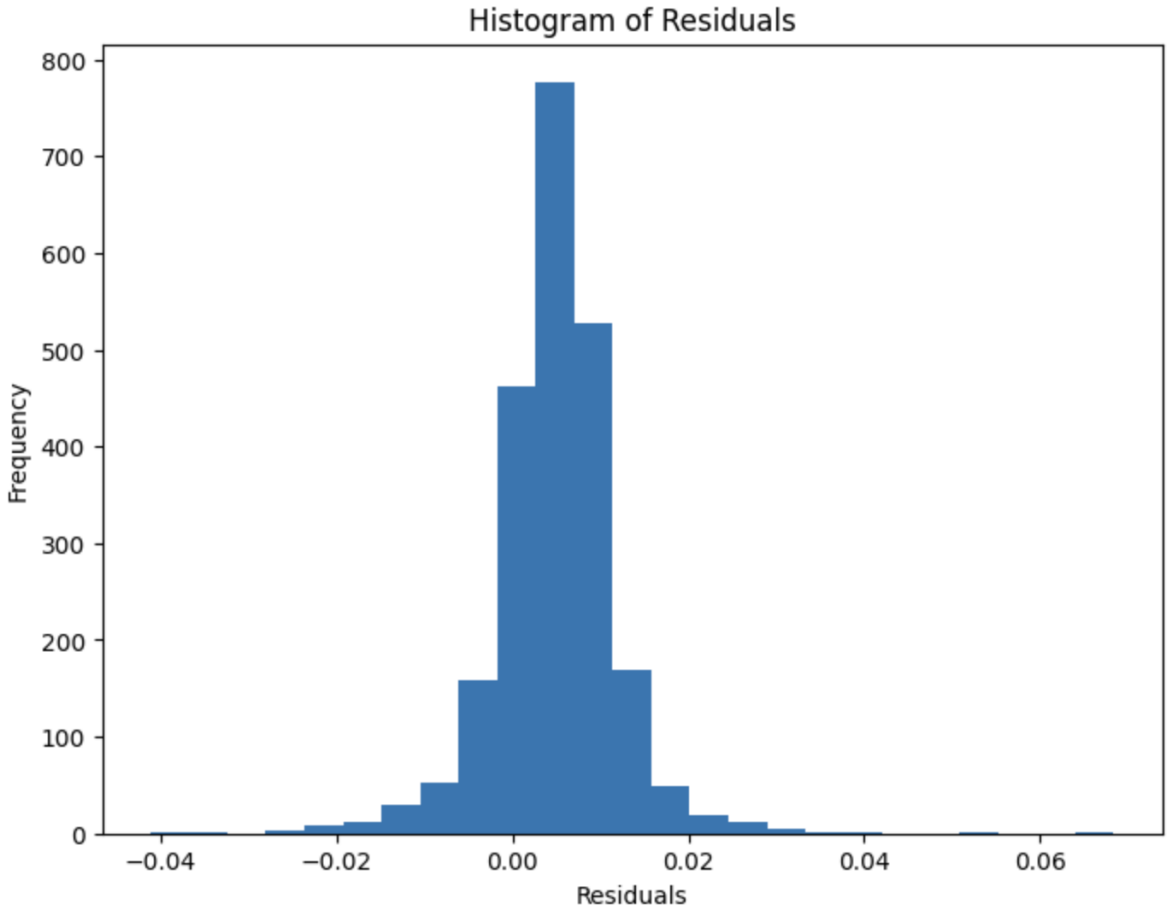
layer, each of which conducts a linear transformation (w*x +b) followed by the non-linear activation function, in this case, the ReLU function. Aside from the first layer, the input of each layer is the output of the previous, and after all the calculations are done, the output layer produces the hypothesis, or predicted PER. After the predicted PER is produced, the prediction is compared with the actual output from the training set to calculate loss; this model uses MSE for its loss function, which is calculated by: MSE = (1/m) * ∑(ŷ - y)^2, where m is the number of training examples. Finally, the gradients of the weights and bias for each parameter are calculated using the chain rule, and updates to the weights and bias are made through the Adam optimizer. The optimizer is ultimately meant to minimize the cost function which results in a more accurate prediction. This process is then done multiple times through many epochs, in an attempt to again produce the most accurate prediction. Hyperparameters were manually initialized, with a learning rate of 0.001, a batch size of 32, and an epoch size of 100 initially selected. These initial values were chosen through experimentation to produce the best results. To examine these hyperparameters, grid search was used by exhaustively experimenting with each combination of hyperparameters (in this case batch size and epochs). 3-fold cross validation was then used to identify the most effective hyperparameters, i.e., the set of hyperparameters that minimized the cost function the greatest. 3-fold cross validation involves splitting the data into 3 subsets, training and evaluating the model trained on those subsets, and using a different data subset as the validation set each time.

5 Experiments/Results/Discussion

The main measure of the results of the model is the MSE, which came out to be relatively low, with the best model mean squared error being 7.17525373581752e-05. After conducting the hyperparameter tuning, the best model consisted of 150 epochs and a batch size of 32. The predicted versus actual normalized PER was then plotted, as shown below:



The redline indicates the predicted PER (remember that the final layer of the neural network used a linear activation, and as result produced a linear result) and the blue dots indicate the actual PER. Overall the model is fit quite well, as the line runs through the points extremely well. To further evaluate the efficacy of this model, a residuals histogram was produced (shown below). Residuals are the difference between the predicted PER and the actual PER, so high frequency of low residuals indicate a well-fit model.



Based on the plots shown above, and the low MSE, it can be reasonably concluded that this model is an accurate predictor of PER based on a player's historical statistics. Note that this model takes into account the players' age, minutes played, and games played, so it can also accurately predict one player's PER throughout various stages of their career, which increases the usability in player and team management.

6 Conclusion/Future Work

In this project, a model was developed, trained and evaluated that was constructed using a neural network to predict the player efficiency rating (PER) based on their individual historical statistics. The dataset was found on Kaggle, and then preprocessed to drop any unnecessary data,

and normalized to ensure appropriate scaling. Then using Keras IPA which utilized the Adam optimization algorithm, a 3 layer neural network was built, using backpropagation as its primary learning algorithm. The model then underwent hyperparameter tuning to find the most effective batch size and number of epochs, using grid search with 3-fold cross validation to do so. To evaluate the effectiveness of the model, mean squared error (MSE) was calculated, and the model was able to achieve an MSE of 7.17525373581752e-05 using a batch size of 32 and 150 epochs. With more time and resources, additional improvements could be made such as feature engineering, where new features were developed that more accurately predicted a player's PER. Different architectures of neural networks could also be implemented, and their effectiveness could then be appropriately evaluated, and potentially lead to better results. Overall, this model showed the ability to accurately predict a player's PER based on their historical statistics.

7 Resources