Predicting outcomes of NBA games

Mihika Johorey, Pratyush Ranjan Tiwari, Sai Vineet Reddy

November 27, 2018

mihika.johorey_ug19@ashoka.edu.in,
pratyushranjan.tiwari_ug19@ashoka.edu.in,
saivineetreddy.thatiparthi_ug19@ashoka.edu.in

Abstract

Our goal in this project is to predict the outcome of a National Basketball Association game using past game statistics. We use a MultiLayered perceptron to predict Win - Loss as well as game scores, achieving a maximum accuracy of 89%. We have also iterated over all possibilities of available features to ascertain which statistical variable has the maximum impact on testing accuracy.

Through our project we wanted to determine the possible benefits and limitations of using just game statistics. Feature set importance give us insight on what game strategies are a better predictor of game results in the current NBA seasons.

1 Introduction

Basketball is a faced-paced game that is popular world over, with NBA being arguably the most important tournament, generating over $7.4 billion dollars in revenue in the 2016-2017 season. (Forbes, 1) It's high uncertainty as well as dynamic score lines and upsets make it a challenging sport to forecast accurately. However, the prevalence of statistical and Machine Learning research in prediction of sporting events has increased drastically in the past two decades, with betting and fantasy sports becoming a lucrative business.

Our model uses a Multi-layered Perceptron in an attempt to accurately predict Win-Loss outcomes as well as Score lines. Our inputs are past statistical variables, collected over the 2014-2017 game period - using data from over 4000 games.

The reason we chose to work on Basketball instead of other sports is because this game is a lot more 'skill based' than other sports, so a statistical model is bound to work better. (Fong, 1)

2 Related Work

"Prediction of NBA games based on Machine Learning Methods" (Torres, 2013) was one of the first few papers that surveyed and compared different methods of Machine Learning for sports prediction. We used this paper heavily in our decisions of model creation.

A project by Stanford undergraduates in 2016, titled *"Predicting NBA Game Outcomes Margins"* (Jaak Uudmae), used 33 input features and used three different models for prediction, with its best result being the Neural Network with a 65% accuracy. Both these models predicted the score line, on the basis of which the Win or Loss was decided.

3 Data Set and Features

We retrieved our data from stats.nba.com, as well as the kaggle dataset `https://www.kaggle.com/ionaskel/nba-games-stats-from-2014-to-2018` using seasons 2014-2017 for training, and 2018 as the testing set. One of our models used all the data for training and instead of testing, we predicted games that are yet to happen.

We cleaned the data of incompleteness/non-applicability using the Pandas library (pd.cleanse), and also fixed headers manually. Team names could not be used, as our model could not process strings as inputs. Originally, each team was given a specific index from 1-30, as there were exactly 30 teams in the NBA. However, it was observed that this reduced our accuracy on training, as the indices were affecting the weights in the Neural network, with the teams with higher indices getting a higher win probability, despite being the weaker team. To fix this issue, we explored the concept of *One-Hot encoding*. We used the same concept on the Home-Away column and the Win-Loss column.

This way, the teams later in the dataset did not affect the weights in the network.

In the end, the following representation was used to feed into our Neutral Network:

There were 85 input features for the Neural Network:

1. The first 30 features captured the team that was playing at home, with 1 in its entry and 0's in all others.
2. The next 30 captured the team that was playing away, with 1 in its entry and 0's in all others.
3. The 61st indicator if the team that the model was trained for was playing home or away. We used 1 to indicate home and 0 to indicate away.
4. The next 12 was the input statistics for the home team, and then the same for the away team, totaling to 24 statistics.

The following 12 statistics were used for each team:

Field Goals Attempted, Accuracy of Field Goals,3 Pointers scored, Assists, Steals, Blocks, Turnovers, Total Fouls, Free Throws Attempted, Free Throws Accuracy, Offensive Rebounds and Total Rebounds.

We arrived at this combination of statistics after running our model on different combinations of the parameters and looking at which combination gave us the best validation accuracy.

Since the data was one large set, we split it on the basis of date into two different sets. One was used for testing and another for training. Depending on the model, we either used the Win-Loss column as our target or the scoreline.

4 Methods

4.1 Training and Validation

Using a Multi-layered Perceptron, we created two models for prediction of match outcomes, based on their true outputs.

a. **Win-Loss classifier**

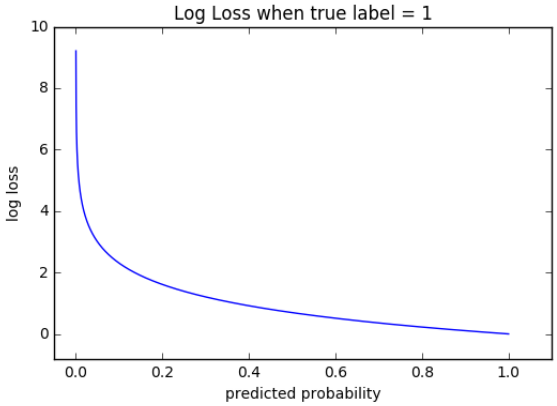
b. **Score-line predictor**

Both of these models had a similar set up. The inputs to both being an 85 feature vector as described above, having 3 hidden layers with 30, 25 and 10 neurons respectively. A dropout of 0.3(30 percent) was also used for both models during training - which proved to have a higher accuracy than the 78% achieved with a dropout of 0.25. We arrived at the number of hidden layers and the dropout percentage through simple trial and error.

However, the two models differed in the following ways:

- For the Score-predictor, ReLu was used as the activation function and loss was calculated using Mean Sqaured Error (MSE). We tried doing this with a sigmoid activation function, but as expected, it could not match the scorelines as they were usally between 80-120.
- For the Win-Loss classifier, The sigmoid function was used as the activation. In this case, the ReLu activation did not work because ReLu has a range $[0, \infty)$ of outputs, which makes it difficult to classify as 0 or 1. Additionally, 'Binary Cross Entropy' was used to record Error instead of Mean Squared Error (MSE). Again, in a binary outcome, using MSE as the loss function might lead to a non-convex error plot, wherein a local minima can be attained. Cross Entropy leads to a convex problem that makes it easier to find the optimum in a binary classifier.

Cross entropy loss for binary classification $= -(\mathbf{y}\mathbf{log}(\mathbf{p}) + (\mathbf{1} - \mathbf{y})\mathbf{log}(\mathbf{1} - \mathbf{p}))$



To train the data, we used an 80-20 split and *k-cross validation*, wherein we would train on 80% of the data and then validate on the remaining 20.

We noticed was that our model gave the same accuracy when the batch size was 1 as opposed to when the batch size was 300, with batch training being much faster.

After about 40 epochs, our training and testing accuracy remained the same in both the models, with an increase in number not changing accuracy significantly.

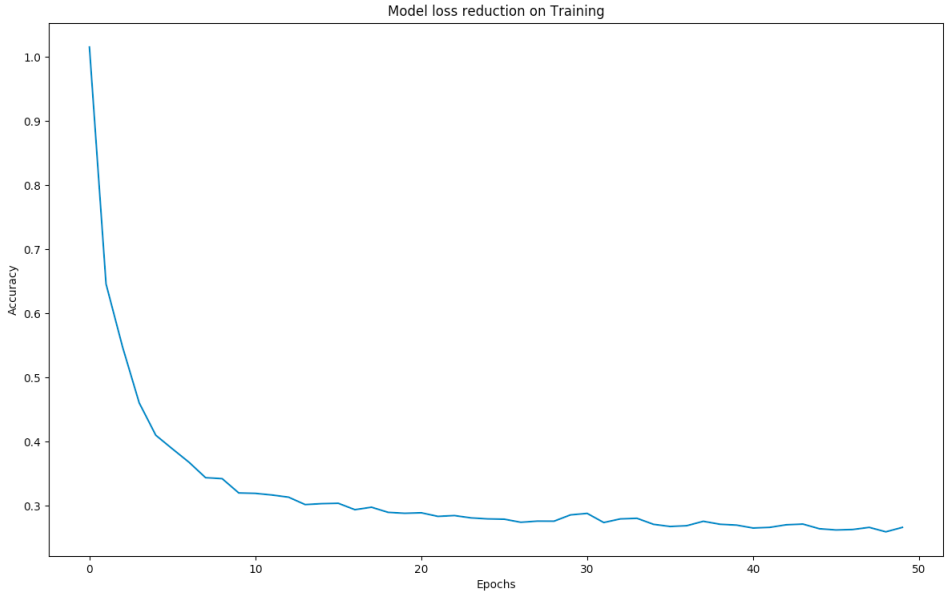


Figure 1: 'Win-Loss model' training loss

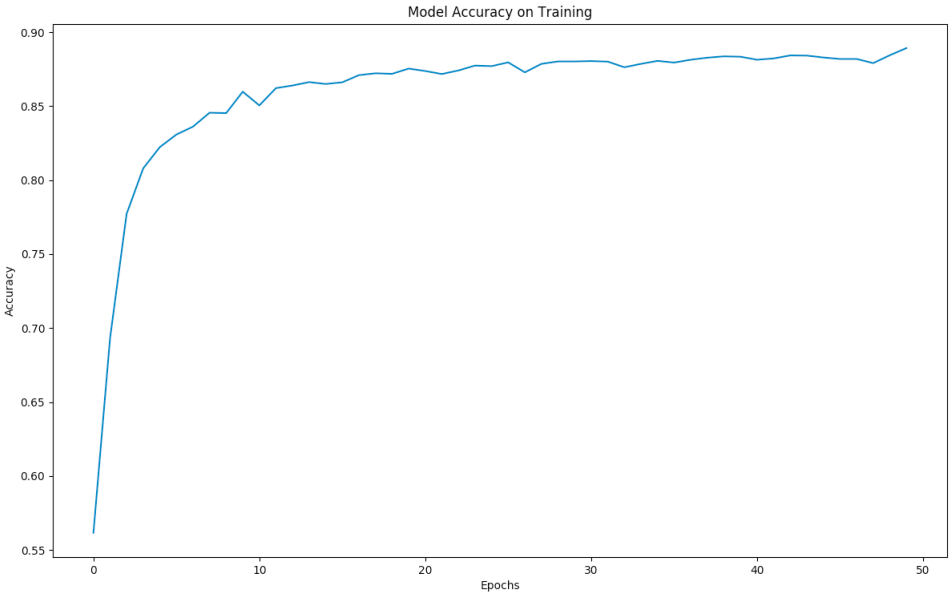


Figure 2: Win-Loss model training accuracy

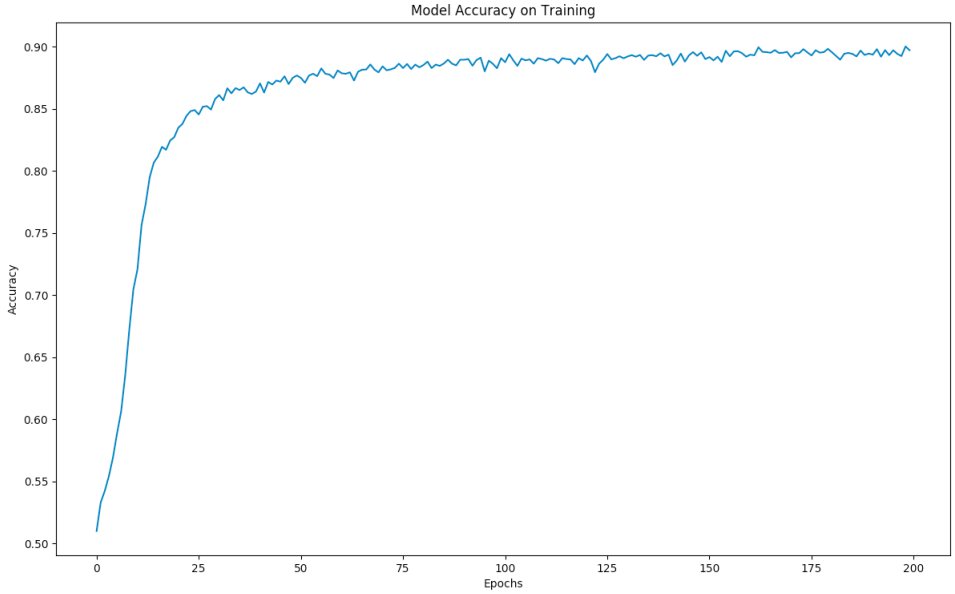


Figure 3: Win-Loss - 200 epochs

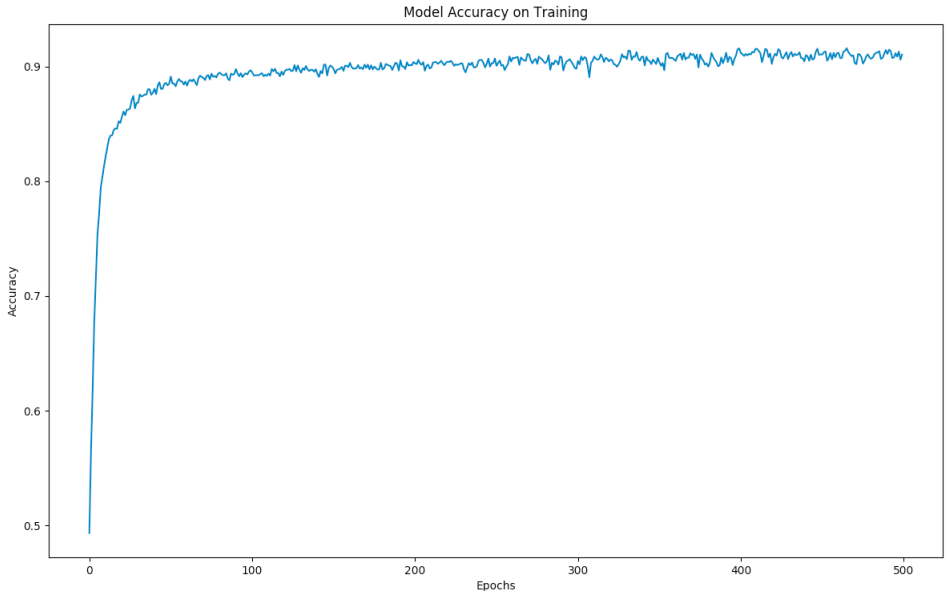


Figure 4: Win-Loss - 500 epochs

4.2 Testing

After the training, we used the remaining 20 percent to validate our model. On average, we achieved a testing accuracy of **89 percent** on the *Win-Loss* model. Whereas when it came to predicting the score lines, our accuracy was a bit lower at **75 percent** [as reported by Keras.]

After training the model, we wanted it to predict games that were yet to happen. To do this, we tested it by using an 85 feature vector, that could input the two teams along with information about whether it was a home or away game. The remaining 15 statistical features were initially set to a constant vector of 1's, on which we got a testing accuracy of 53%. However, we realised that this defeated the purpose of using a neural network in the first place.
Therefore, to improve this accuracy, we initialized the 15 features to the average of all the stats in every game that the teams we needed to predict had played against each other. To check if this worked, we tried predicting games that were yet *Sun, 18 - November - 2018* to happen and the results were as shown in **Table 1**, which shows that we predicted 8/10 games accurately.

Table 1: Predictions of NBA games on 18th Nov 2018

Team Name			**Model**
Home	**Away**	**Home Win/Loss**	**Prediction**
Nets	Clippers	L	L
Pelicans	Nuggets	W	W
Magic	Lakers	W	W
Hornets	76ers	L	L
Mavericks	Warriors	L	W
Bulls	Raptors	L	L
Pacers	Hawks	W	W
Suns	Thunder	L	W
Rockets	Kings	W	W
Celtics	Jazz	W	L

4.3 Feature Importance

To determine which statistical feature is the most important to predicting outcomes of matches, the attempt was to iterate over each combination of the features in order to ascertain which gives the highest prediction accuracy. There were about $\sum_{i=1}^{30} C_i^{30}$ combination of features to check for. This took us about 4 and a half days to run as we had to check the accuracy of each and every model. We ran a Python script on FloydHub to do this.

5 Results

Table 2: Accuracy

Win - Loss		Score - Line	
Training	Testing	Training	Testing
90%	89%	84%	82%

An interesting result is that during the time we ran our feature importance algorithm to find the best combinations of features, we noticed that whenever **Steals** and **Blocks** were considered as features, our Training and Testing accuracy was about 5 percent higher than the rest of the combinations. This could imply that a Team that does better on Defence, wins the game. It could be also mean that Defence is more important than Offense when it comes to a NBA game.

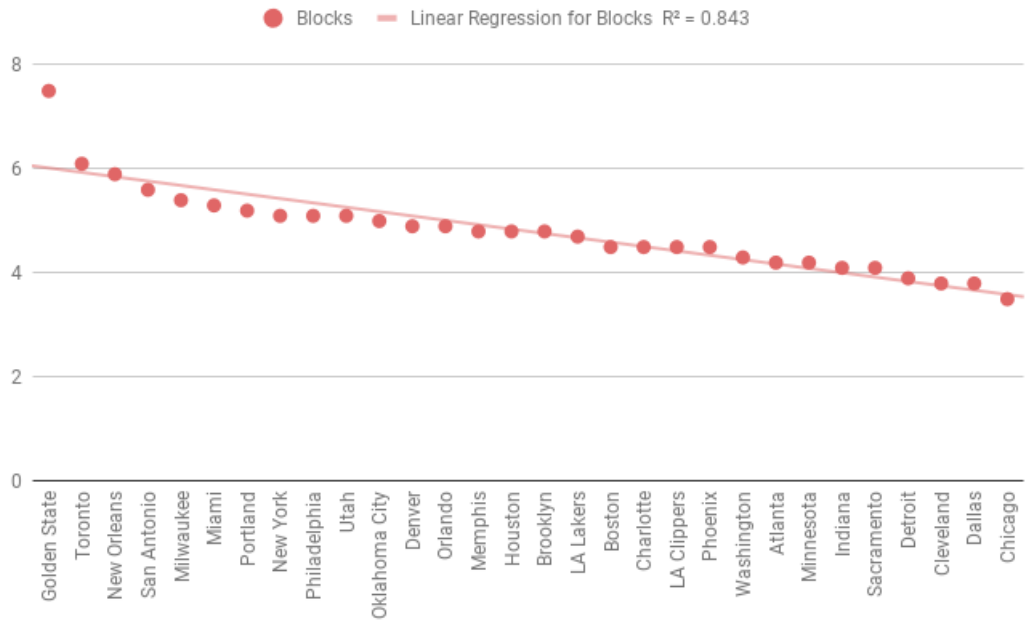


Figure 5: Rank vs. Blocks for 2016-2017 season

6 Conclusion

Our aim was to ascertain how accurately a model trained on team match statistics could perform on games in the 2018 season. While our Win-Loss predictor surpassed the Score predictor's accuracy, both models faired well on the testing set. We built an extra feature on top of the Win-Loss model where we could predict future games by averaging out the stats of the teams.

Our results were impacted by the following:

1. Player retirement, Coach changes as well as MVP transfers between teams could not be factored in. This could be especially noticed as our model wrongly predicts Cavaliers' and Lakers' games against each other.

2. Data size as well as computing ability. Using a larger number of seasons than just 2013-2017 for training (4000 games), may have produced better results.

3. Changes in epoch sizes, number of hidden layers and neurons in each layer could have had an impact on our accuracy rate.

Finally, below is a summary of our Models:

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 30)	2580
dropout_1 (Dropout)	(None, 30)	0
dense_2 (Dense)	(None, 25)	775
dense_3 (Dense)	(None, 10)	260
dense_4 (Dense)	(None, 1)	11

Total params: 3,626
Trainable params: 3,626
Non-trainable params: 0

7 References

1. Communications, Forbes Corporate. *Forbes Corporate. "Forbes Releases 20th Annual NBA Team Valuations."* Forbes, Forbes Magazine, 7 Feb. 2018, www.forbes.com/sites/forbespr/2018/02/07/forbes-releases-20th-annual-nba-team-valuations/5695221734e6.

2. Lieder, Nachi. *Can Machine-Learning Methods Predict the Outcome of an NBA Game?* March 1, 2018). Available at SSRN: https://ssrn.com/abstract=3208101

3. Torres, R. *Prediction of NBA games based on Machine Learning Methods.* Accessed from https: //homepages.cae.wisc.edu/ ece539/fall13/project/AmorimTorresrpt.pdf

4. Uudmae, Jaak. *Predicting NBA Game Outcomes.* Accessed from: https://github.com/jaagu/CS229FinalP

5. Fong, Joss. "Why It's so Much Harder to Predict Winners in Hockey than Basketball." Vox.com, Vox Media, 5 June 2017, www.vox.com/videos/2017/6/5/15740632/luck-skill-sports.

To reference code used in our project:

6. `https://www.floydhub.com/vineetred/projects/ml/10`

7. `https://github.com/vineetred/sportPredictor`

8. Class notes and discussions - Professor Ravi Kothari, CS303, Ashoka University.