

---

# CS229 Final Project: Predicting NBA Game Outcomes

---

## Author

Jaak Uudmae juudmae@stanford.edu

## Abstract

1 The goal of this paper was to predict upcoming NBA game scores based on just  
2 previous game scores and home/away advantage. By comparing the scores one  
3 can also predict the game winner. For the baseline game winner accuracy an SVM  
4 Classifier was trained on each team. That achieved an average of 62% accuracy of  
5 picking the correct team to win. In order to predict game scores a Linear Regression  
6 (LR) and Neural Network Regression (NNR) models were used. The LR approach  
7 achieved a 64% game winner prediction accuracy while being about 10 points  
8 off for each team. The NNR approach achieved a 65% game winner prediction  
9 accuracy while being about 12 points off for each team. Given that the game scores  
10 vary at around that much throughout the season, the results were satisfying.

## 11 1 Introduction

12 Basketball is a popular sport around the world due to the amount of uncertainty in each game. Upsets  
13 happen all the time and every team has a high chance to win any game. The goal of this paper was  
14 to see if it was possible to predict game scores based on just previous game scores and remove that  
15 uncertainty. In order to do that three different models were implemented: SVM Classifier, Linear  
16 Regression, and Neural Network Regression. The input to the models were just the teams playing  
17 and which team was home/away. The SVM model received additional features of previous wins and  
18 losses in the season. The output for the SVM model was if the home team won or not and for the  
19 other two models the output was the predicted home team and away team scores.

20 The data collected for this project was also used in a CS238 project with Paul Steenkiste to predict  
21 NBA game outcomes using POMDPs. All of the methods defined below were used only for CS229  
22 and the methods in the CS238 project were used only for the CS238 part. The code and the writeup  
23 for the CS238 project are available at <https://github.com/jaagu/CS238FinalProject>

24 All of the code for this paper is available at <https://github.com/jaagu/CS229FinalProject>

## 25 2 Related Work

26 There has been work done on this problem before. One of the examples comes from Renato Amorim  
27 Torres who wrote a paper titled *Prediction of NBA games based on Machine Learning Methods*  
28 (Torres, 2013). The goal of that paper was to predict the winner of a game. For the linear regression  
29 they used features like win-loss percentage for both teams, point differential per game for both teams,  
30 win-loss percentage for previous 8 games for both teams, and win-loss percentage as a visitor and  
31 home team for both teams. With all of those features they achieved a result of 66.91% accuracy.

32 Another Linear Regression algorithm to predict the winner was used in a paper by M.Beckler, H.Wang,  
33 and M.Papamichael *NBA Oracle* (Beckler, Wang, Papamichael, 2008). Using previous games, they  
34 achieved a result of 73% accuracy, which was the best result found.

35 For the Autumn 2016 CS229 Final Project G.Avalon, B.Balci, and J.Guzman wrote a paper *Various*  
36 *Machine Learning Approaches to Predicting NBA Score Margins* (Avalon, Balci, Guzman, 2016).  
37 Their goal was to predict margins for each game which is slightly different than the goal for this  
38 project. From the margin they also tried to see how accurate they were at predicting the winner. The  
39 Gaussian Discriminant Analysis achieved at 65.54% accuracy, the Linear Regression 64.26% and  
40 Random Forest 61.36%. For the input they used 218 features which is much more than what this  
41 paper used.

### 42 **3 Dataset and Features**

43 The data used in this project is from stats.nba.com, which is a publicly available website for statistics  
44 associated with the NBA. In order to scrape the data, a Python library called nba\_py was used  
45 (Uriegas, E, 2017). The training data were all the games between 2013-2016 (March) which is around  
46 4500 games. The test data were the rest of the games of the 2016 season which is about 400 games.  
47 For the SVM model the training and test data were a subset of the whole data set consisting of only  
48 games for the team that the model was trained for. Resulting in around 150 training games and 14  
49 test games.

50 There were 33 input features for the SVM:

- 51 1. The first 30 features captured the teams that were playing since there are 30 teams in the league.
- 52 Twenty eight "0s" for teams that weren't playing and two "1s" for teams that were playing.
- 53 2. Indicator if the team that the model was trained for was playing home or away.
- 54 3. Count of wins so far in the season for the team that the model was trained for.
- 55 4. Count of losses so far in the season for the team that the model was trained for.

56 The true output was an indicator if the team that the model was trained for won the game or not.

57 There were 60 input features for the Linear Regression and Neural Network Regression:

- 58 1. The first 30 features captured the team that was playing home. 29 "0s" for teams that weren't  
59 playing home and one "1" for the home team.
- 60 2. The latter 30 features captured the team that was playing away. 29 "0s" for teams that weren't  
61 playing away and one "1" for the away team.

62 The true output was a vector consisting of two values: the home team's score and the away team's  
63 score.

64 All of the features were built after scraping the game data. The SVM classifier was built in order to  
65 get a baseline accuracy to aim for with the regression algorithms.

## 66 **4 Methods**

### 67 **4.1 SVM Classifier**

68 For the SVM Classifier a sklearn library was used (Scikit-learn: Machine Learning in Python). A  
69 "kernel trick" was used in order to deal with the 33 features without ever having to represent the  
70 weights.

71 The SVM is trying to find an hyperplane dividing the data into two categories. It solves this problem:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^n \zeta_i$$

subject to  $y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i,$   
 $\zeta_i \geq 0, i = 1, \dots, n$

72

73 The decision was made using this function:

74

75 Where the kernel function K is the radial basis function:  $\exp(-\gamma \|x - x'\|^2)$

$$\text{sgn}\left(\sum_{i=1}^n y_i \alpha_i K(x_i, x) + \rho\right)$$

76 **4.2 Linear Regression**

77 For the linear regression the sklearn library was used (Scikit-learn: Machine Learning in Python).  
 78 The algorithm fits a linear model with coefficients  $w = (w_1, \dots, w_{60})$  to minimize the residual sum  
 79 of squares between the observed responses in the dataset, and the responses predicted by the linear  
 80 approximation:

$$\min_w \|Xw - y\|_2^2$$

81

82 The input was the teams that were playing and the output was the scores.

83 **4.3 Neural Network Regression**

84 In order to implement the Neural Network Regression, Keras framework was used (Chollet, 2015).  
 85 Keras was used since it uses the GPU for training, which allowed for fast training.

86 The network was defined with three hidden layers. The first layer has 60 neurons. Second one 30  
 87 neurons. Third one 16 neurons. The choice of the wide network was made after testing out different  
 88 models and this one achieving the best results. The same goes for the number of hidden layers.

89 The first two layers had a dropout of 0.1, which means that a tenth of the fraction of input units were  
 90 set to 0 at each update during training time in order to prevent too much overfitting. Since there's  
 91 some randomness in every basketball game, overfitting was not preferred.

92 All of the layers used 'relu' activation which is a max value between 0 and input. It was chosen  
 93 because the derivative of it is simple (just 1) and that made the training quicker. The output layer had  
 94 a linear activation. The initialization for all layers was done using random normal distribution.

95 The batch size for each update was defined as 100 games to make the training faster.

96 The loss function that the neural network used was mean squared error. This made sense since our  
 97 goal was to predict scores as accurately as possible.

98 The input was the teams that were playing and the output was the scores.

99 **5 Results**

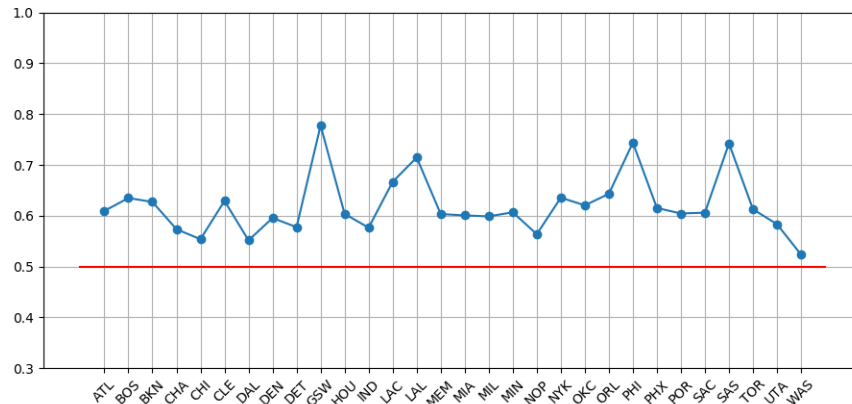


Figure 1: SVM prediction for each team

100 For each of the methods the accuracy for predicting the correct game winner on the test data was  
 101 calculated. The SVM model achieved an average of 62% accuracy (see Figure 1). That was used as a  
 102 baseline to see how good the score predictions are for the other two models. Relative to the previous  
 103 work done, the accuracy was satisfying since the input features were much smaller and less detailed.  
 104 The Linear Regression (LR) surpassed the SVM accuracy with 64% and the best result came from  
 105 the Neural Network Regression (NNR) with 65%.  
 106 For predicting the scores an average distance between prediction and the true value was used. The  
 107 LR model did better than the NNR. It was off by about 10 points for both home and away scores,  
 108 while the NNR was off by around 12p each. The results are very satisfying since the average standard  
 109 deviation for the teams in the NBA based on the 2013-14 season was around 12 points (Zovas, 2015).  
 110 The training for the Neural Network converged pretty quickly with little changes happening after  
 111 2000 epochs (the training for the network was stopped after 64000 epochs). Each tick in the x-axis  
 112 represents a time when progress was made in the learning algorithm and when the model was saved  
 (see Figure 2).

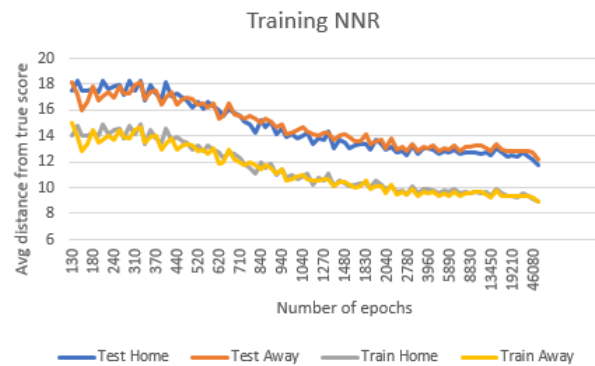


Figure 2: Training of NNR

113  
 114 There are several explanations for why the NNR was worse for predicting the scores, but better at  
 115 picking the correct winner. Since the game scores change a lot from game to game even if the same  
 116 teams are playing, the Neural Network did not perfectly fit the training game scores while the LR  
 117 method did. The NNR seems to have captured something more inherit than scores since it was able  
 118 to be more accurate at picking winners even though the score prediction weren't as good.

The results are summarized in the table below (see Figure 3).

	Model 1:SVM	Model 2: LR	Model 3: NNR
TEST: Average Accuracy	62.07%	63.75%	<b>64.95%</b>
TEST: Home Score Distance	N/A	<b>9.8203</b>	11.7816
TEST: Away Score Distance	N/A	<b>9.9472</b>	12.1915
TRAIN: Average Accuracy	<b>70.99%</b>	66.85%	70.34%
TRAIN: Home Score Distance	N/A	8.7656	8.8918
TRAIN: Away Score Distance	N/A	8.8426	8.8631

Figure 3: Results

119

## 120 6 Conclusion and Future Work

121 The goal was to see if it was possible to predict game scores based on just previous game scores.  
 122 Both models predicting the scores achieved good results given that the team scores vary at around  
 123 12p on average (based on 2013-14 season data). The Linear Regression model achieved a slightly  
 124 better result than the Neural Network Regression model, but the Neural Network Regression was a  
 125 more accurate on the test set for predicting the game winner.

126 The results were affected by various factors:  
127 1. Data size was too small. There were a total of around 4500 games from 2013-2016.  
128 2. Team rosters/coaches change between seasons so the team's scoring is not consistent from season  
129 to season.

130 Overall, I think the neural network has the potential to overtake LR predictions for the game scores.  
131 All three of these methods implemented were a great way to go over the material learned for this  
132 class and it was great to see those algorithms in action.

133 For the future work, given more data to train the neural network, the results could be improved. In  
134 addition, trying out different neural networks (width, number of hidden layers, number of neurons  
135 each layer) could yield better results.

## 136 7 Contributions

137 All of the work described above was done by Jaak Uudmae. I'd like to thank Paul Steenkiste for the  
138 help with the data formatting.

## 139 References

- 140 [1] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel,  
141 Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David  
142 Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay; *Scikit-learn: Machine Learning in Python*.  
143 12(Oct):28252830, 2011.
- 144 [2] Chollet, Francois and others (2015). *Keras*. Retrieved from <https://github.com/fchollet/keras>.
- 145 [3] Uriegas, E (2017). *nba\_py*. Retrieved from [https://github.com/seemethere/nba\\$\\_py](https://github.com/seemethere/nba$_py)
- 146 [4] Zovas, J (2015). *Detailed Look AT NBA Team-By-Team Consistency - 2013-*  
147 *14 Season*. Accessed from [https://www.sportingcharts.com/articles/nba/](https://www.sportingcharts.com/articles/nba/detailed-look-at-nba-team-by-team-consistency-during-the-2013-14-season.aspx)  
148 [detailed-look-at-nba-team-by-team-consistency-during-the-2013-14-season.aspx](https://www.sportingcharts.com/articles/nba/detailed-look-at-nba-team-by-team-consistency-during-the-2013-14-season.aspx)
- 149 [5] Beckler, M and others (2008). *NBA Oracle*. Accessed from [https://www.mbeckler.org/coursework/](https://www.mbeckler.org/coursework/2008-2009/10701_report)  
150 [2008-2009/10701\\_report](https://www.mbeckler.org/coursework/2008-2009/10701_report).
- 151 [6] Avalon, G and others (2016). *Various Machine Learning Approaches to Predicting NBA Score*  
152 *Margins*. Accessed from [http://cs229.stanford.edu/proj2016/report/Avalon\\_balci\\_guzman\\_](http://cs229.stanford.edu/proj2016/report/Avalon_balci_guzman_various_ml_approaches_NBA_Scores_report.pdf)  
153 [various\\_ml\\_approaches\\_NBA\\_Scores\\_report.pdf](http://cs229.stanford.edu/proj2016/report/Avalon_balci_guzman_various_ml_approaches_NBA_Scores_report.pdf)
- 154 [7] Torres, R (2013). *Prediction of NBA games based on Machine Learning Methods*. Accessed from [https:](https://homepages.cae.wisc.edu/~ece539/fall13/project/AmorimTorres_rpt.pdf)  
155 [//homepages.cae.wisc.edu/~ece539/fall13/project/AmorimTorres\\_rpt.pdf](https://homepages.cae.wisc.edu/~ece539/fall13/project/AmorimTorres_rpt.pdf)