# The Bank is Open: AI in Sports Gambling

**Alexandre Bucquet**
bucqueta@stanford.edu

**Vishnu Sarukkai**
sarukkai@stanford.edu

## Abstract

In this paper, we explore the applications of Machine Learning to sports betting by focusing on predicting the number of total points scored by both teams in an NBA game. We use Neural Networks as well as recurrent models for this task, and manage to achieve results that are similar to those of the sports books. On average, our best models can beat the house $51.5\%$ of the time.

## 1 Introduction

Last May, the United States Supreme Court legalized sports betting. While it is still up to every state to pass legislation on the issues, many have already been working on bills. As tracked by ESPN, eight states have already legalized sports gambling, with two more states having signed bills that will legalize gambling in the near future. With its ruling, the Supreme Court paved way to the opening of whole new legal (and thus taxable) market of size \$150 billion according to the American Gaming Association.

In this project, we will explore the applications of Machine Learning in the field of sports betting, using a case study of the National Basketball Association. More specifically, we wish to design sets of models that predict betting indicators for NBA matches. Out of the three main indicators (Over-Under, Money Line and Point Spread), we focus on the Over-Under. In other words, we predict for every NBA game how many combined points the two teams will score. Ideally, we would be able to come up with an estimate for this indicator that is more accurate than that of some betting platforms, and use this to our advantage to place bets.

## 2 Related Work

The paper "Predicting Margin of Victory in NFL Games" [1] uses a Gaussian process model in aiming to model NFL point spreads and beat the Las Vegas line. They aim to beat the line through the incorporation of novel features such as the temperature of the field at kickoff and indicator variables representing team "strengths." However, the model achieves a success rate in predicting games 2% worse than the Vegas line. This may be because high-dimensional Gaussians are not the most appropriate way to model NFL point totals and point spreads. Moreover, NFL data in very scarce as each team only takes the field less than 20 times a year.

The paper "Football Match Prediction using Deep Learning" [2] uses recurrent models to predict the outcome of soccer matches. The architecture developed includes a Long Short Term Memory Network (LSTM) that processes game data at every time interval. This allows the authors to obtain a live prediction of who will win the soccer match every minute as the game progresses. The final model had one LSTM layer with $256$ hidden dimensions, and achieved a test accuracy of $88\%$. This paper illustrates the relevance of recurrent models to sports data and its competitive results suggest that more is to be done for other sports using similar model architectures.

## 3 Dataset and Features

### 3.1 Data Collection

The data collected so far can be classified into two groups: betting odds data and game data, which consists of both data describing team performance and player performance.

We found odds data on Sports Book Review Online, a website that compiles betting data for every NBA games since the 2007-2008 season. The website offers a downloadable excel file for each season. Using a short script, we were able to retrieve all the necessary target variables and the corresponding betting odds offered. The betting indicators scraped were the following:

i Over-Under: the total number of points scored in a game.

ii Spread: the number of points by which the home team wins (or loses).

iii Money Line: a number encoding the amount of money won from placing a bet on the winning team of a game.

In the rest of this project, we focus on the total number of points scored, and we use the Over-Under data collected to solely evaluate our models, and do not include the above data in our features.

For game data, we retrieved data from Basketball Reference using Fran Goitia's NBA crawler [3]. We made minor edits to the scraper in order to account for formatting inconsistencies in the data. This crawler provided a strong base to start collecting data on every team for every game since the 2007-2008 NBA season. The data collected for every game included statistics such as Points Scored, Rebounds, Assists and Steals for both teams.

### 3.2 Feature Building

Using the retrieved data from Basketball Reference, we build a featurized dataset. For every matchup between two teams, we decide to look at both team's past three games. We included simple features such as Points Scored, Points Scored Against, or Total Rebounds, and also more complicated metrics such as Offensive Rating and Plus/Minus. In order to account for opponent strength, we also added each opponent's season averages in all above metrics. Finally, to account for player fatigue, we added the number of days since the last game as well as the distance traveled (see Figure 1).

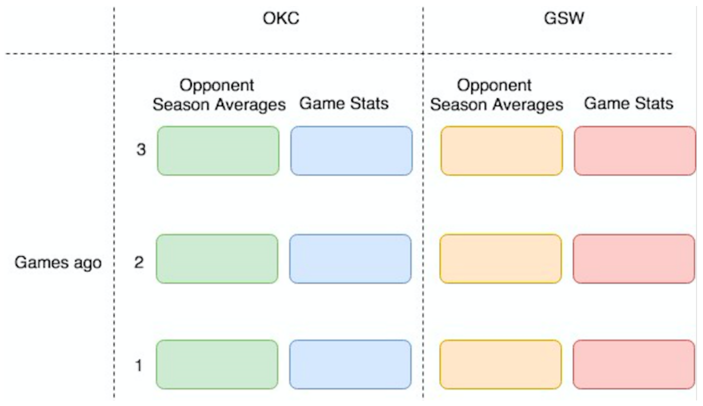

Figure 1: Overview of the features used for a game between the Golden State Warriors (GSW) and the Oklahoma City Thunder (OKC)

## 4 Methods

All the code for this project can be found at https://github.com/abucquet/bank_is_open. We used Python libraries Scikit-learn [4] (Random Forest), Surprise [5] (Singlar Value Decomposition), and PyTorch [6] (Neural Net and LSTM).

For the task of predicting the number of points scored in a game, we minimize Mean Squared Error (MSE) when training all our models. We trained our models on four seasons ($2012-2013$ to $2015-2016$), used the $2016-2017$ season as our validation set, and tested our models on the $2017-2018$ season. This gave us a train set of size 5,069, a validation set of size 1,260 and a test set of size 1,264.

We trained a Random Forest as a baseline before moving on to more complex models.

### 4.1 Collaborative Filtering

This problem is similar to the user/item rating prediction, information about the outcome of previous games between other teams can inform our predicted output for the current matchup. To leverage this, we used Singular Value Decomposition [7] to build a model similar to collaborative filtering. In this model, we build a sparse matrix $A$ that contains the number of points scored by two teams if the teams have faced off in the past few days. In this setting, the rows represent the home team and the columns the away team. Then, we factorize $A$ into $U\Sigma V^T$ by solving

$$min_{U,V,\Sigma} \sum_{ij, A_{ij}\neq 0} (A_{ij} - (U\Sigma V^T)_{ij})^2$$

We hope that the rows of matrices $U$ and $V$ capture information about teams at home and away, respectively, and we then predict $u_i \Sigma v_j^T$ total points scored in a game between teams $i$ (home) and $j$ (away), where $b_k$ is the $k$-th row of matrix $B$.

### 4.2 Neural Network

Like the collaborative filtering model, the Neural Network captures information from the outcomes of previous games between other teams, as during training the network is provided the results of previous games as input along with the identity of the two teams involved in the game. However, the Neural Network has an advantage over collaborative filtering in that it is also able to take features of both teams involved as inputs. Therefore, it can draw on not only the outcomes of previous training examples but also the offensive rating of each of the teams involved over their past three games, etc.

Passing in the feature set created in 3.2, we train a Neural Network with fully-connected layers and ReLU activations. The input data is flattened into a size of $1524$ features. The ReLU activations are used for ease of training and to reduce the likelihood of gradient vanishing (see Figure 2).

### 4.3 Long Short Term Memory Network (LSTM)

Given that games are played sequentially, we decided to use an LSTM to process the past three games one by one. As described in ”Long Short-Term Memory” [8], an LSTM is a recurrent model where we repeatedly computed a hidden state by processing a new timestep (in our case one game). The equations to compute the future hidden state $h_t$ from the current state $h_{t-1}$ and the input $x_t$ are given below:

$$\begin{aligned} z_t &= \sigma(W_z[h_{t-1}, x_t] \\ r_t &= \sigma(W_z[h_{t-1}, x_t] \\ \tilde{h}_t &= tanh(W[r_t * h_{t-1}, x_t] \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \end{aligned}$$

For our task, we implemented an LSTM model with a fully connected layer at the end to output the number of points scored by the two teams for the desired game (see Figure 3).

## 5 Results

After tuning our models on the validation set to minimize validation MSE, we found the following architectures to perform best:

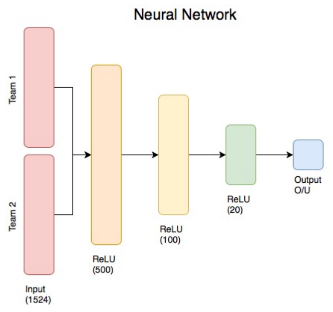

Figure 2: Overview of Neural Network Architecture

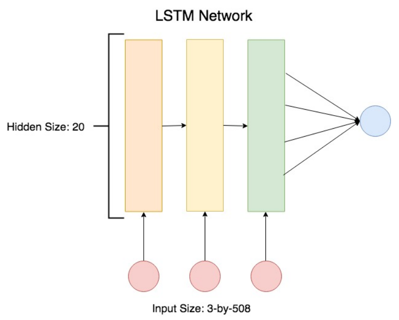

Figure 3: Overview of LSTM Architecture

i Collaborative Filtering: We train a model looking at the games around the league for the past five days, as that threshold minimized the validation MSE (see Figure 4).

ii Neural Network: We use four fully-connected layers, reducing the input of dimension 1524 to size 500, then 100, then 20, then finally a 1-dimensional output that predicts the Over-Under of the desired game (see Figure 2). We used a weight decay parameter of 1 to regularize the weights of the network, and a learning rate of $10^{-6}$ over 5000 epochs, using Gradient Descent. This allowed the network to roughly train to convergence (see Figure 5).

iii LSTM: We use one layer with hidden dimension 20, and a fully connected layer at the end (see Figure 3). We trained our model using Stochastic Gradient Descent with batches of size 10, learning rate of 0.05 and dropout of 0.2 for 100 epochs (see Figure 6).

We evaluate the performance of our models primarily through the Mean Squared Error (MSE) between the Over-Under value predicted by the model and the true point total observed in the game. The Over-Under values predicted by the sports books can serve as a benchmark as calculating the MSE between the sports books' predictions and the observed point totals gives us a sense of how well our models are performing relative to the books. In addition, we can also calculate the percentage of the time that our model would have correctly predicted that the true point total was either over or under the Over-Under number provided by the sports books – we found that the Neural Network correctly chooses Over or Under around 51.5% of the time, while the Collaborative Filtering beats the line 51% of the time on average.

| **Model** | **Train MSE** | **Test MSE** |
|---|---|---|
| Random Forest (baseline) | 955.10 | 910.33 |
| Collaborative Filtering | 2.95 | 459.85 |
| Neural Network | 349.17 | **369.84** |
| LSTM Network | 398.95 | 426.56 |
| Sports Book Over-Under | - | 320.696 |

Table 1: Model Results

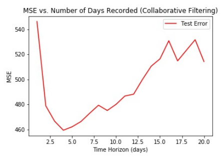

Figure 4: Collaborative Filtering Hyperparameter Search

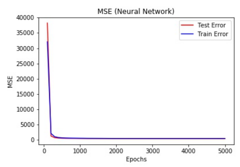

Figure 5: Neural Network Training Curve

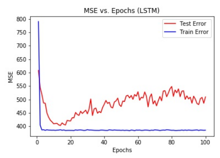

Figure 6: LSTM Training Curve

## 6 Discussion

Due to the rapidly changing nature of the NBA it is difficult to acquire sufficient training data that reflects the way the game is currently played. This can be seen most prominently in the increasing frequency with which NBA teams are attempting (and making) three-point shots. According to the NBA, teams around the league combined for around 15,000 three-pointers made in the $2009 - 2010$ season, and broke the 25,000 mark last year (2017-2018).

When we were training the models, we found while while we had data available starting from the $2007 - 2008$ season, we achieved the highest levels of validation accuracy when the training dataset started from the $2012 - 2013$ season rather than from the $2007 - 2008$ season. This suggests that the earlier seasons are not very representative of the validation season (which was the $2016 - 2017$ season) possibly because of the changing nature of the NBA.

It is worth noting that the Collaborative Filtering model, which doesn't use any team features, significantly outperformed the Random Forest. This shows the high variance in our data as well as the strong seasonal trends that a model needs to encompass in order to be accurate on this task.

On Figure 6, it seems like the LSTM model doesn't learn much after the first few epochs of training and is overfitting to the training data. This is likely due to the fact that our data set is rather small (around 5,000 example), and LSTMs needs lots of data to achieve successful training, both in terms of learning and overfitting. Using the validation set, we weren't able to find appropriate ways to reduce overfitting without compromising the validation accuracy.

We achieved a test MSE of 369.84 for our best model, the Neural Network. This value is higher than the MSE of the sports books' predictions, which is 320.70, but is relatively close to the level of accuracy of the books. When considering that NBA teams are scoring roughly a combined 200 points per game, MSE values of over 300 from even sports books may seem high. However, this is simply a reflection of the high-variance nature of NBA games, where anything from injuries to players to a strong shooting night or the decision of a coach to rest his star players after building a lead can all lead to huge swings in the overall point totals of a game. The variance of the data was also reflected in the need for a relatively large weight decay parameter while training the Neural Network to prevent overfitting.

Encouragingly, when placing bets against the sports books on games in the test dataset using the Neural Network, the predictions made were correct with respect to the actual outcomes 51.5% of the time, as mentioned earlier. However, note that in the real world, the fees associated with betting mean that successful long-term betting patterns need to be correct at least 52-53% of the time. In the future, it would be interesting to explore if directly predicting whether the true result of the game is over of under a sports book's Over-Under prediction would lead to higher levels of accuracy in this secondary metric.

## 7 Conclusion and Future Work

Overall, we were able to achieve encouraging results using our models, as our best predictions rivaled the accuracy of sports books. Indeed, our Neural Network had a MSE of 369.84, compared to the 320.696 MSE of sports book. On average, our model was able to pick the correct side of the Over-Under 51.5% of the time.

In the future, there is definitely potential to improve the model through augmenting our data set. First of all, the odds lines offered by sports books themselves offer a large amount of information on the potential outcomes of games, including data that indicates the direction the odds lines are moving in the hours before the game. It is entirely possible that the odds lines would indicate that the sports books are very good at predicting the outcomes of games involving certain teams, but tend to skew in some direction when trying to predict the outcomes of games involving other teams. In addition, while our current models only use team-level data, incorporating player-level data could add additional layers of nuance while accounting for the impact of injuries, or player fatigue.

Further exploration of model architectures could potentially improve our results as well. Due to the similarity of our current problem (predicting betting indicators from the home team and away team) to classical recommendation systems (predicting ratings for a given user and item), we could definitely explore adapting algorithms used by Netflix, Amazon, and others for recommendation.

## References

[1] Jim Warner. Predicting margin of victory in nfl games: Machine learning vs. the las vegas line. Technical report, Technical Report, 2010.

[2] Daniel Pettersson and Robert Nyquist. Football match prediction using deep learning. 2017.

[3] Fran Goitia. Basketball reference scraper. `https://github.com/FranGoitia/basketball_reference`, 2017.

[4] Scikit-learn. `https://scikit-learn.org/stable/`.

[5] Surprise. `http://surpriselib.com/`.

[6] Pytorch. `https://pytorch.org/`.

[7] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numerische mathematik*, 14(5):403–420, 1970.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

## Contributions

In this project, both Vishnu and Alexandre contributed equally to the writing of this report. Alexandre completed around two thirds of the data retrieval/cleaning process, built the Collaborative Filtering model and the LSTM network. Vishnu completed the remaining third of the data retrieval/cleaning process, built the baseline models as well as the Neural Network. The rest was achieved jointly by both members.