
NBA Outcome Modeling

Krish Katariya

Georgia Institute of Technology
kkatariya3@gatech.edu

Matthew Lim

Georgia Institute of Technology
mlim70@gatech.edu

Elaine Tang

Georgia Institute of Technology
etang49@gatech.edu

Colin Vu

Georgia Institute of Technology
cvu33@gatech.edu

Abstract

We investigate machine learning methods for predicting the combined points total for NBA game sports betting (the "Over-Under" betting category). Leveraging fully-connected neural networks, recurrent sequence models (LSTMs), and the collaborative filtering method, we train on historical box-score and betting-odds data to predict the outcomes of NBA games accurately in respect to time and different team match-ups.

1 Introduction

Sports betting is one of the fastest growing betting industries in the country, with an estimated 30% year-to-year growth [1]. As sports betting grows, so does the importance of being able to make accurate sports predictions and establish reliable betting lines. To that end, we investigate novel data-driven wagering decisions and employ several machine learning models that predict the total points scored in a game, also known as the Over-Under betting line, of NBA games. Our models include random forests, neural networks, collaborative filtering, and LSTMs.

2 Background & Related Work

Past efforts to predict sports betting outcomes have used traditional statistical approaches. Many utilize linear regression-based methods such as those proposed by Massey [2], which assign ratings to teams by solving a system of equations based on past game performance outcomes but is limited in its ability to capture non linear interactions. Recently, machine learning methods, particularly neural networks, have also risen in popularity for predicting sports betting outcomes due to their capacity to model complex relationships in data. Bunker and Susnjak [3] offer an overview of different machine learning applications in sports betting, concluding that deep learning methods or a combination of different machine learning methods frequently outperform simpler models.

Several works have attempted to address NBA-specific outcome modeling. Loeffelholz et al. [4] developed a neural network model to predict NBA game winners, achieving 74% accuracy on historical outcomes, demonstrating the relative success of using feedforward architectures. Meanwhile, Zimmermann et al. [5] examined various machine learning approaches for predicting player-specific performances, showing that metadata such as individual player metrics could be used to support team-level inferences.

This paper is a recreation and extension of The Bank is Open: AI in Sports Gambling [6].

30 2.1 Data

31 We scraped our NBA data from online sports websites. This data includes odds or betting data and
32 game statistics such as points scored, rebounds, assists, steals, and more. In total, 3 websites were
33 used.

- 34 • Sportsbook reviews [7]: This website contains betting and point data for every game in an
35 NBA season. It includes the money line for each game, which is the odds to win the game,
36 in addition to the point score at the end of each quarter.
- 37 • Basketball reference [8]: This website hosts a wide variety of NBA statistics, but we focus
38 on using the box score data for each game. The box score includes information about each
39 team and player after the game, such as the point scores, 3-pointers, rebounds, free throws,
40 field goals, and more.
- 41 • Team rankings [9]: This website also stores information about every NBA matchup, but we
42 specifically scraped each NBA team’s average scoring margin per day. Scoring margin is
43 the difference between how many points a team scores and how many points the other team
44 scored in a single game.

45 This data is scraped in either excel or csv file format. The data processing system works as follows:

- 46 1. Clean odds and game DataFrames and map varying team names to 3-letter abbreviations for
47 name standardization, and merge odds with game stats.
- 48 2. Compute chi-square tests comparing Over/Under vs. Moneyline.
- 49 3. Compute rolling averages of box-score stats over recent games for each team.
- 50 4. For each home game, collect 1) One-hot encodings of home and away teams, 2) Recent
51 game stats for both teams, accounting for travel distances, and 3) Season-to-date average
52 stats of opponents.
- 53 5. Stack these into an (games, # of recent games, features) array and save (X, y) as pickled
54 NumPy arrays for modeling.

55 3 Methods

56 3.1 Baseline Model (Random Forest)

57 **Overview** To first create a baseline model, we create decision trees to perform regression. Each
58 tree is trained on a bootstrap sample of the data and makes a prediction by averaging its leaf outputs.
59 The forest’s final prediction is the mean of all individual tree predictions:

$$\hat{y}(x) = \frac{1}{T} \sum_{t=1}^T h_t(x),$$

60 where T is the number of trees and h_t is the t -th tree’s prediction.

61 3.2 Fully Connected Neural Networks

62 Our neural network implementation leverages PyTorch to create a deep fully-connected architecture
63 for predicting the total points scored in NBA games. The model processes 1524 features representing
64 each team’s statistics from their previous three games, capturing recent performance trends and
65 matchup dynamics.

66 The network architecture consists of five fully-connected layers with decreasing width to progressively
67 extract higher-level patterns from the input data: Input layer: 1524 features \rightarrow 512 neurons; Hidden
68 layer 1: 512 \rightarrow 256 neurons; Hidden layer 2: 256 \rightarrow 128 neurons; Hidden layer 3: 128 \rightarrow 64 neurons;
69 Output layer: 64 \rightarrow 1 neuron (point total prediction).

ReLU activation functions follow each hidden layer to introduce non-linearity and mitigate the vanishing gradient problem:

$$f(x) = \max(0, x) \quad (1)$$

The model is trained using the Adam Optimizer with a learning rate of 10^{-6} and weight decay regularization (coefficient = 10^{-5}) to prevent overfitting. We employ Mean Squared Error (MSE) as our loss function:

$$\mathcal{LMSE} = \frac{1}{n} \sum_i (y_i - \hat{y}_i)^2 \quad (2)$$

where y_i represents the actual point total and \hat{y}_i is the predicted value. During training, we monitor both training and validation loss to ensure proper generalization.

3.3 Collaborative Filtering

Setup. Past NBA team matchups form a sparse matrix $R \in \mathbb{R}^{m \times n}$ whose entry r_{ij} is the total points when home team i played against visitor j . Collaborative filtering is commonly used by recommendation systems (i.e. Amazon and Netflix), where rows are "users," columns are "items," and entries r_{ij} predict user-item interaction (i.e. ratings) based on past user-item interactions. Likewise, our implementation treats home teams as "users," visitor teams as "items," and predicted cumulative points as the "interactions".

We approximate R with a rank- k model

$$\hat{r}_{ij} = \mu + b_i + b_j + \mathbf{p}_i^\top \mathbf{q}_j, \quad (3)$$

where μ is the global mean total, b_i and b_j are team-specific bias terms, and $\mathbf{p}_i, \mathbf{q}_j \in \mathbb{R}^k$ ($k = 10$) are latent "offence" and "defence" factors. Equation (3) thus corresponds to the low-rank factorisation $R \approx U\Sigma V^T$ with additive bias terms.

Model parameters minimise the ℓ_2 -regularised loss

$$\mathcal{L} = \sum_{(i,j) \in \Omega} (r_{ij} - \hat{r}_{ij})^2 + \lambda (\|\mathbf{p}_i\|^2 + \|\mathbf{q}_j\|^2 + b_i^2 + b_j^2),$$

with Ω the set of played games. Stochastic gradient descent updates, e.g. for \mathbf{p}_i , are $\mathbf{p}_i \leftarrow \mathbf{p}_i + \eta(e_{ij}\mathbf{q}_j - \lambda\mathbf{p}_i)$ where $e_{ij} = r_{ij} - \hat{r}_{ij}$. We use step size $\eta = 10^{-4}$ and $\lambda = 5 \times 10^{-3}$.

Because rosters and form shift rapidly, the model is retrained using various timeframes; the most recent w days of data. A grid search over $w \in \{3, \dots, 21\}$ found $w = 3$ to 5 (≈ 36 -60 games) minimizes validation error. Finally, we use the Surprise SVD class (SGD optimiser, $k = 10, 1000$ epochs, η and λ as above).

3.4 Long Short Term Memory Networks (LSTMs)

Overview Each LSTM cell processes an input vector x_t , the previous hidden state h_{t-1} , and the cell state c_{t-1} , producing updated states h_t and c_t . The underlying equations are:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (\text{input gate}) \quad (4)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (\text{forget gate}) \quad (5)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (\text{output gate}) \quad (6)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (\text{cell input}) \quad (7)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (\text{new cell state}) \quad (8)$$

$$h_t = o_t \odot \tanh(c_t) \quad (\text{new hidden state}) \quad (9)$$

$\sigma(\cdot)$ is the logistic sigmoid and $\tanh(\cdot)$ is the hyperbolic tangent. W_i, U_i , and b_i are learned parameters.

- One LSTM layer, which reads the input sequence and returns only the final hidden state.
- A dropout layer applied to the output to reduce overfitting.
- Fully-connected dense layer with ReLU activation.
- Dropout layer with the same rate as the previous.
- Output layer with a single unit and linear activation for regression.

The model is compiled with an Adam optimizer and uses MSE loss and mean absolute error (MAE) as metrics.

4 Experiments

4.1 Baseline Model (Random Forest)

Our Baseline Random Forest model is trained on games from the 2007-2008 season, validated against 2008-2009, and tested against 2009-2010. Each game has 1525 features, the same as each other model, except also adding a time variable.

Hyperparameter	Value
Number of trees, T	150
Maximum tree depth	1
Split criterion	Mean Squared Error (MSE)

Table 1: Hyperparameters for the LSTM model.

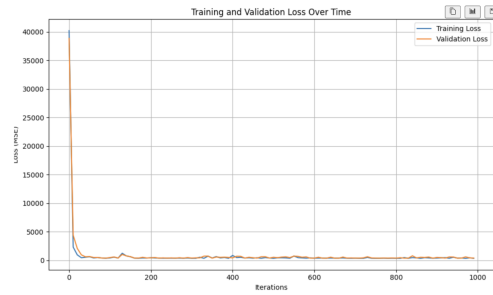
Dataset	Loss
Train	1280.88
Validation	811.99
Test	5178.90

Table 2: Final model losses.

4.2 Fully Connected Neural Networks

We evaluate our fully connected neural network on NBA seasons 2012–2018, using 2012–2016 seasons for training (5,069 games), 2016–2017 for validation (1,260 games), and 2017–2018 for testing (1,264 games). The model processes 1,524 features per game, representing both teams’ performance statistics from their previous three games. We train for 1,000 epochs using SGD with a learning rate of 10^{-5} and weight decay of 10^{-6} for regularization. The model shows convergence within 100 epochs, demonstrating its ability to capture the underlying patterns in the data despite the inherently high-variance nature of NBA scoring totals.

Loss Accuracy	Train MSE	Test MSE
Neural Network	332.99	369.38



4.3 Collaborative Filtering

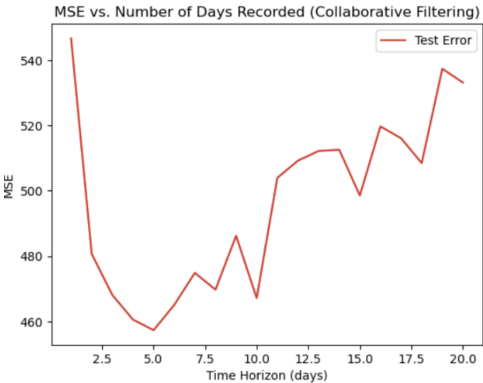
While our model has similar results to the original neural network, our model converges in 30 epochs while the original paper converges in 100-150 epochs.

Experimental setup. We evaluate the CF pipeline on NBA games from the 2007–08 through 2017–18 seasons. The model uses a sliding window approach where, at the start of each game-day, only matches from the preceding w days are visible (with w varied from 1 to 20 days to determine the optimal window size). Metrics are computed *chronologically* to avoid look-ahead bias, with the model being retrained for each new window of games.

Prediction accuracy. Table 3 reports the Mean Squared Error (MSE) of the CF model against three baselines:

Table 3: Mean Squared Error (MSE) of CF model vs. baselines

Method	Train MSE ↓	Test MSE
Collaborative Filtering	2.85	369.84



127

128 This plot shows the MSE when considering different sized windows of data (last X number of days).
129 Based on this plot, we can infer that a window of 3-5 days is optimal because it balances prioritizing
130 relevant information of sports athletes with having a sufficient amount of data to inform our model.

131 **4.4 Long Short Term Memory Networks (LSTMs)**

132 Our Long Short Term Memory (LSTM) model is trained on games from the 2012-2013, 2013-2014,
133 2014-2015, and 2015-2016 seasons, validated against 2016-2017, and tested against 2017-2018. Each
game has 1524 features.

Hyperparameter	Value
Number of LSTM units	128
Dense layer size	64
Dropout rate	0.10
Batch size	5
Number of epochs	30
Optimizer	Adam
Loss function	MSE
Patience	5 epochs

Dataset	Loss
Train	546.63
Validation	399.40
Test	398.29

Table 5: Final model losses.

Table 4: Hyperparameters for the LSTM model.

134

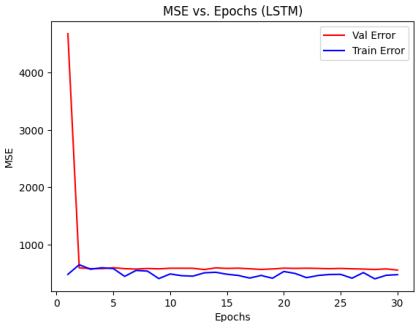


Figure 1: LSTM Loss (MSE) across 30 epochs

135 5 Conclusion

136 5.1 Discussion

137 Overall, Neural Networks had the best test MSE, at 369.38. LSTM posted similar results (test MSE
138 = 398.29) by modeling temporal dependencies, and the Random Forest Baseline model performed
139 much worse than the other models. Collaborative filtering remains competitive with the two better
140 models, but its reliance on very recent data, with an optimal 3–5 day window, may require frequent
141 retraining, which could be inefficient when compared to other models. The deeper models demanded
142 more training time and hyperparameter searching, whereas collaborative filtering could be retrained
143 quickly but missed broader factors, leading to poorer results.

144 The superior performance of the fully connected network (MSE = 369.38) suggests that capturing
145 non-linear interactions among hundreds of box-score features is critical. Furthermore, the LSTM’s
146 similar performance (MSE = 398.29) indicates that short-term temporal patterns carry predictive
147 signal, and working on better utilizing longer-range dependencies may help. Furthermore, the competi-
148 tive performance of collaborative filtering highlights the value of latent team-vs-team interactions,
149 yet its dependence on a narrow 3–5 day retraining window raises concerns about model drift. The
150 random forest, while easier to interpret and faster to train, fails to model the complex interactions that
151 drive high-variance NBA scoring, resulting in suboptimal accuracy.

152 Below are a table and chart that represent our final losses:

Dataset	Train Loss (MSE)	Val Loss (MSE)	Test Loss (MSE)
Random Forest	1280.88	811.99	5178.90
Collaborative Filtering	2.85	N/A	457.32
Neural Networks	332.99	369.41	369.38
LSTM	546.63	399.40	398.29

Table 6: Our Model Losses Comparison



Figure 2: Loss Comparison Chart (Logarithmic)

153 5.1.1 Future Work

154 With more time and resources, we could improve upon the work done in this paper in the following
155 ways:

156 Firstly, we could use more diverse data and richer contextual features. For example, we could include
157 features such as player-level statistics, injury reports, rest days, back-to-back indicators, and more.

158 We could also consider using different model architectures such as transformer-based sequence
159 models. We could feed the past T games for each team into a Transformer encoder with positional
160 embeddings and attention headers, allowing the model to weight non-contiguous games while not
161 being limited by a fixed window. We could also consider using graph neural networks (GNNs) as
162 they offer a natural way to model all of our model matchups.

163 Lastly, we could consider building a real-time deployment pipeline that retrains based on live odds
164 and in-game stats. This extension would create the most impact on real sports bettors, but may be
165 difficult to use in practice.

166 References

- 167 [1] Gustafson, B. (2025, March 28). *2024: A year of growth for sports betting revenue*. CBS Sports. <https://www.cbssports.com/betting/news/2024-a-year-of-growth-for-sports-betting-revenue/#:~:text=Total%20tax%20revenue%20from%20sports,dramatic%20growth%20in%20sports%20betting>.
- 170 [2] Massey, K. (n.d.). *Statistical Models Applied to the Rating of Sports Teams*. Massey Ratings. <https://masseyratings.com/theory/massey97.pdf>
- 172 [3] Bunker, R., & Susnjak, T. (2022). The Application of Machine Learning Techniques for Predicting
173 Match Results in Team Sport: A Review. *Journal of Artificial Intelligence Research*, 73, 1285–1322.
174 <https://doi.org/10.1613/jair.1.13509>
- 175 [4] Loeffelholz, B., Bednar, E., & Bauer, K. W. (2009). Predicting NBA
176 games using Neural Networks. *Journal of Quantitative Analysis in Sports*, 5(1).
177 https://www.researchgate.net/publication/24015640_Predicting_NBA_Games_Using_Neural_Networks#:~:text=Loeffelholz%20et%20al.,expert%20scores%20of%2069%25.
- 179 [5] Zimmermann, A., Moorthy, S., & Shi, Z. (2013, October 14). *Predicting college basketball match outcomes using machine learning techniques: Some results and lessons learned*. arXiv.org. <https://arxiv.org/abs/1310.3607>
- 181 [6] Bucquet, A., & Sarukkai, V. (2018). *The Bank is Open: AI in Sports Gambling*. CS229: Machine Learning.
182 <https://cs229.stanford.edu/proj2018/report/3.pdf>
- 183 [7] *Historical NBA Scores and Odds Archives @ Guide to NBA odds & betting lines*. Sportsbook Reviews. (n.d.).
184 <https://www.sportsbookreviewsonline.com/scoresoddsarchives/nba/nbaoddsarchives.htm>
- 185 [8] *Basketball Statistics & History of every Team & NBA and WNBA players*. Basketball Reference. (n.d.).
186 <https://www.basketball-reference.com/>
- 187 [9] *NBA Basketball*. Team Rankings. (n.d.). <https://www.teamrankings.com/nba/>