# 601.475 Final Project Writeup

Raj Gosain, Angela Hu, Alan Zheng, Kuleen Sasse

December 1, 2024

## 1 Problem

We want to use Machine Learning techniques to try to predict the outcomes of NBA basketball matches using previous data. We aim to project the probability each team will win and compare this with the implied probabilities on sportsbooks to measure how our predictions compare to their odds.

## 2 Why is it important?

Sports betting is a multi-billion dollar industry that has exploded in popularity due to recent deregulation ("Why US," 2024). Being able to create accurate forecasting models would allow users to have a lucrative edge.

## 3 Previous Work

Previous studies have been published that apply machine learning to various sports. Shi, Moorthy, and Zimmermann (Zimmermann et al., 2013) found that despite initial expectations, simple models like the Naive Bayes model perform even better than more complex models for NCAAB predictions. However, there appears to be an upper limit for the predictive capabilities of models across various sports, including soccer and basketball. Thabtah, Zhang, and Abdelhamid found defensive rebounds as the strongest predictor for NBA game results (Thabtah et al., 2018). Deep Learning has been applied to predicting NBA players' performance, but was found to perform worse than Machine Learning (Nguyen et al., 2021), potentially due to limited data size with few predictor variables.

## 4 Data

For our project, we required two sources of data: betting data and scoring data from NBA games. We obtained the betting data through a `kaggle.com` dataset called "Basketball Betting Dataset." This dataset scraped betting data various sorts of betting data like spreads, overs, moneyline odds etc. from 2007 to 2022 (Broas, J). We combined this data with data that was scraped from a popular NBA data aggregator `basketball-reference.com`. This data consisted of both scores, basic player stats, and advanced player stats for each game from 2016-2022 (Paruchuri, V). These player stats for both teams were both averaged across all the lineup. In addition, the max for all players on a team was taken for each feature across the lineup. This yielded 136 columns. We inner joined these datasets ending up with around 6500 games to train our models on. In the dataset, the 'team' feature denotes the home team and 'team_opp' denotes the visiting team. All features without the _opp suffix are for the home team, and all features with the _opp suffix are for the opposing team. Due to the temporal aspect of the data, we did not want the model to look at the stats at the end of the game to make a prediction. To alleviate this problem, we did a running average of the data of all games played up to a team up to that point during that season. To give an example, say the LA Lakers and the Golden State Warriors were to play a match where the Lakers had played 8 matches previously and the warriors played 9 games previously that season, then what the row of data for the model to predict the outcome of that game would consist of the averages of the statistics of the Lakers from their 8 previous games that season and averages of the statistics of the Warriors from their 9 previous games that season. Games early on in the season were dropped as strengths of teams could not be evaluated effectively.

## 5 Training

We'll frame this as a supervised binary classification task, using five models: logistic regression, random forest, decision tree, support vector machine, and multi-layer perceptron. We will predict whether the home team wins. We will use 2016-2021 seasons as the training and validation data and the 2022 set as the testing data.

We have a two step process to train the most ideal models. We first place our models into a forward feature selection algorithm to choose the most important features. We upper bound the number to be 30 as there are around 150 features that we want to reduce to the most important set. This feature selector uses cross validation to choose the best features across all folds. Because of the temporal aspect of the data, we use the `TimeSeriesSplit` cross validation object from `sklearn` to split the data while keeping the temporal aspect of the data. After selecting the best features, we then run a `GridSearchCV` with the a small set of hyperparameters to further dial in the model. The details of the parameters selected will be discussed further.

## 6 Evaluation

We evaluate our trained models in two ways. We first measure the accuracy on the held out a held out test dataset which will be the 300 games from 2022. In addition, we will simulate deploying our models during the 2022 season, seeing if our model could theoretically make money from sports betting. The model starts with $1000 in starting betting money. For each game in the season, we feed the data into the model and allow the model to output probabilities of the home team winning and the away team winning. Then, we calculate the bet we are placing on the game using the Kelly criterion. The Kelly criterion is a mathematical formula to calculate the most optimal bet that maximizes long term award. It is calculated in Equation 6.

$$f^* = p - \frac{1-p}{b}$$

$f^*$ is the fraction of the current money to wager. $p$ is the predicted probability of a win. $b$ is the multiplier of the amount of money you'll gain off the bet. We use our models to predict $p$ and use the odds given by the bookmakers for $b$ to calculate $f^*$. We subtract the amount of money from our current money. If the bet is correct, we make $b$ times our bet. If not we lose our money.

## 7 Results

### 7.1 Logistic Regression

The sequential feature selector for 30 features selected the following predictors:

'mp' (minutes played),
'blk' (blocks),
'drb%' (defense rebounds),
'blk%' (block percentage),
'usg%' (usage percentage),
'ft_max' (maximum free throws for a given player),
'orb_max' (maximum offensive rebounds for a given player),
'trb_max' (maximum total rebounds for a given player),
'tov_max' (maximum turnovers for a given player),
'+/-_max' (maximum plus-minus for a given player),
'ftr_max' (maximum free throw rate for a given player),
'ast%_max' (maximum assists percentage for a given player),
'blk%_max' (maximum block percentage for a given player),
'drb_opp' (defensive rebounds for opposing team),
'trb_opp' (total rebounds for opposing team),
'pts_opp' (points scored for opposing team),
'ts%_opp' (true shooting percentage for opposing team),
'orb%_opp' (offensive rebound percentage for opposing team),
'drb%_opp' (defensive rebound percentage for opposing team),
'trb%_opp' (total rebound percentage for opposing team),
'tov%_opp' (turnover percentage for opposing team),
'usg%_opp' (usage percentage for opposing team),
'ortg_opp' (offensive rating for opposing team),
'ft%_max_opp' (maximum free throws for a given opposing player),
'drb_max_opp' (maximum defensive rebounds for a given opposing player),
'+/-_max_opp' (maximum plus-minus for a given opposing player),
'orb%_max_opp' (maximum plus-minus for a given opposing player),
'trb%_max_opp' (maximum total rebounds for a given opposing player),
'ortg_max_opp' (maximum offensive rating for a given opposing player),
'total_opp (points scored by opposing team)'

A small note is that since minutes played is being selected as a feature, you can tell that there is not really a need to select more than 30 features. Minutes played barely tells you anything since every team plays 240 minutes per game plus maybe a bit more if they have to go into over time.

For hyperparameter tuning, we compared l1 and l2 regularization, a regularization parameter (C) of 0.01, 0.1, 1, and 10, 100, and a solver of liblinear and saga.

The best combination found through grid search consisted of the following parameters: C=1, max_iter=100, and penalty=l1, solver=liblinear, 'tol'=0.001. This gave a final test accuracy of 58%.

The simulated principle when applying the model over games in the 2022 season is shown below (Figure 1).

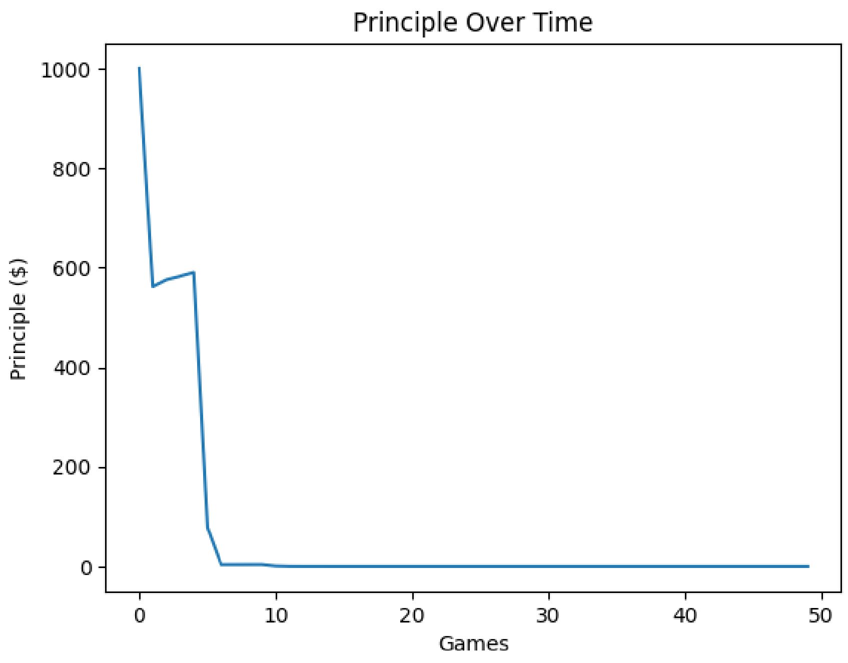

Figure 1: A figure displaying the performance of logistic regression modeling employing the betting strategy described above

## 7.2 Random Forest

The sequential feature selector was run with 15 features, which took 3.5 hours to run, selecting the following predictors:

‘3p%’ (three-point percentage),
‘ts%’ (true shooting percentage),
‘drb%’ (defensive rebound percentage),
‘usg%’ (usage percentage),
‘pts_max’ (maximum points for a given player),
‘+/-_max’ (maximum plus-minus for a given player),
‘3par_max’ (maximum three-point attempt rate),
‘stl%_max’ (maximum steal percentage for a given player,
‘ft_opp’ (opponent free throws),
‘drb_opp’ (opponent defensive rebounds),
‘stl_opp’ (opponent steals),
‘usg%_opp’ (opponent usage percentage),
‘fga_max_opp’ (maximum opponent field goal attempts for a given player),
‘3p_max_opp’ (maximum opponent three-pointers for a given player),
‘won_opp’ (opponent win rate).

For hyperparemeter tuning, we compared number of estimators of 100 and 200, minimum number of samples to split a node of 2 and 5, minimum number of samples for a leaf of 1 and 2, and the number of features for best split of Square Root and Log Base 2.

The best combination found through grid search consistent of the following parameters: number of estimators = 200, minimum samples to split node = 5, minimum samples for leaf = 2, number of features for split = Square Root. This gave a final test accuracy of 56%.

The simulated principle when applying the model over games in the 2022 season is shown below (Figure 2).

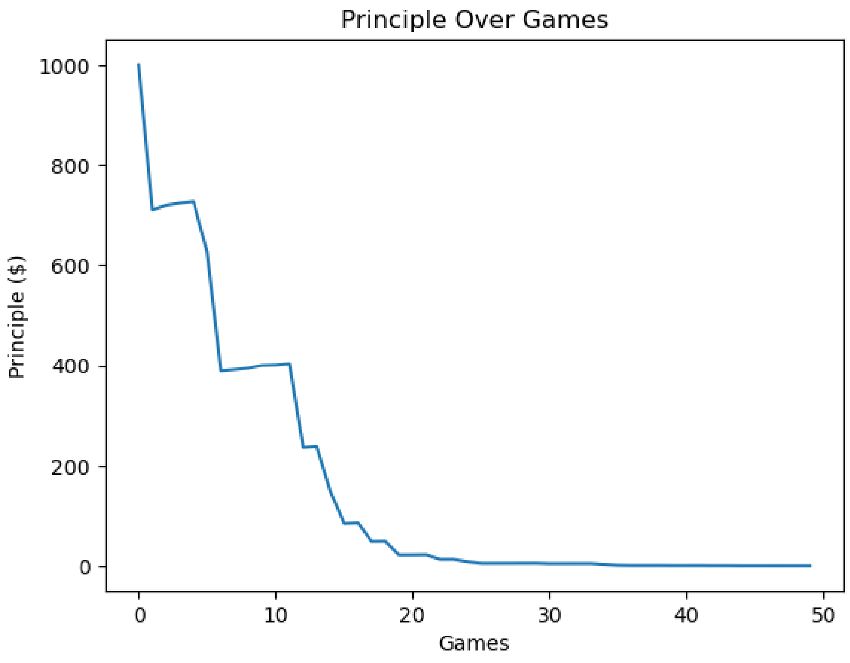

Figure 2: A figure displaying the performance of random forest modeling employing the betting strategy described above

### 7.3 Decision Tree

The sequential feature selector was run with 30 features, which took 1 hour to run, selecting the following predictors:

‘3p’ (three-pointers),
‘orb’ (offensive rebounds),
‘ast’ (assists),
‘efg%’ (effective field goal percentage),
‘trb%’ (total rebounds),
‘ast%’ (assists),
‘stl%’ (steal percentage),
‘tov%’ (turnover percentage),
‘usg%’ (usage percentage),
‘3p_max’ (maximum three-pointers for a given player),
‘fta_max’ (maximum free throw attempts for a given player),
‘ft%_max’ (maximum free throw percentage for a given player),
‘stl_max’ (maximum steals for a given player),
‘tov_max’ (maximum turnovers for a given player),

‘ast%_max’ (maximum assist percentage for a given player),
‘ortg_max’ (maximum offensive rating for a given player),
‘3p_opp’ (opponent three-pointers),
‘fta_opp’ (opponent free throw attempts),
‘ast_opp’ (opponent assists),
‘pts_opp’ (opponent points),
‘3par_opp’ (opponent three point attempt rate),
‘ftr_opp’ (opponent free throw rate),
‘usg%_opp’ (opponent usage percentage),
‘fga_max_opp’ (maximum opponent field goal attempts for a given player),
‘fg%_max_opp’ (maximum opponent field goal percentage for a given player),
‘pf_max_opp’ (maximum opponent personal fouls for a given player),
‘pts_max_opp’ (maximum opponent points for a given player),
‘ortg_max_opp’ (maximum opponent offensive rating for a given player),
‘total_opp’ (opponent total),
‘won_opp’ (opponent win rate).

For hyperparameter tuning, we compared maximum tree depths of 0 to 50, in increments of 10, a minimum number of samples to split a node of 2, 5, and 10, a minimum samples for a leaf node of 1, 2, and 4, split criteria of gini versus entropy, the number of features for best split of None, Square Root, and Log Base 2, and a minimum impurity decrease of 0, 0.01, and 0.1.

The best combination found through grid search consistent of the following parameters: max tree depth = 0, minimum samples to split a node = 2, minimum samples for a leaf node = 1, split criteria = entropy, number of features for split = None, and minimum impurity decrease = 0.01. This gave a final test accuracy of 60%.

The simulated principle when applying the model over games in the 2022 season is shown below (Figure 3).

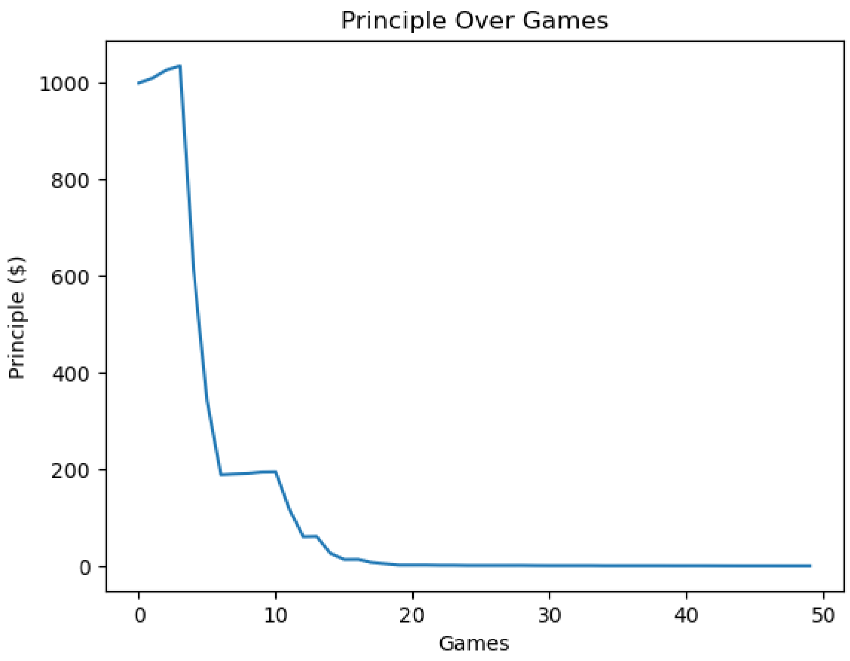

Figure 3: A figure displaying the performance of decision tree modeling employing the betting strategy described above

## 7.4 Support Vector Machine

The sequential feature selector was run with 15 features, which took just under 6 hours to run, selecting the following predictors:

‘stl%’ (steal percentage),
‘tov%’ (turnover percentage),
‘orb_max’ (maximum offensive rebounds for a given player),
‘+/-_max’ (maximum plus-minus for a given player),
‘stl%_max’ (maximum steal percentage for a given player),
‘3p_opp’ (opponent three-pointers),
‘trb_opp’ (opponent total rebounds),
‘pts_opp’ (opponent points),
‘ortg_opp’ (opponent offensive rating),
‘drb_max_opp’ (maximum opponent defensive rebounds for a given player),
‘trb_max_opp’ (maximum opponent total rebounds for a given player),
‘blk_max_opp’ (maximum opponent blocks for a given player),
‘orb%_max_opp’ (maximum opponent offensive rebound percentage for a given player),
‘blk%_max_opp’ (maximum opponent block percentage for a given player),
‘ortg_max_opp’ (maximum opponent offensive rating for a given player).

For hyperparameter tuning, we compared rbf and linear kernels, a regularization parameter (C) of 0.1, 1, and 10, and a kernel coefficient (gamma) of scale, auto, 0.1, and 1.

The best combination found through grid search consisted of the following parameters: C=1, gamma=scale, and kernel=rbf. This gave a final test accuracy of 59%.

The simulated principle when applying the model over games in the 2022 season is shown below (Figure 4).

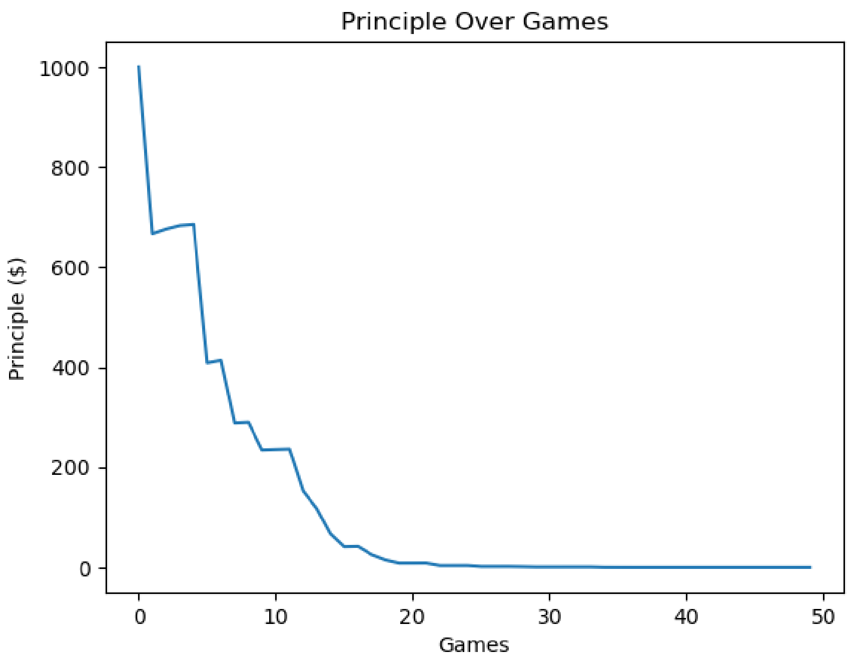

Figure 4: A figure displaying the performance of support vector machine modeling employing the betting strategy described above

## 7.5 Multi-Layer Perceptron

The sequential feature selector for 30 features selected the following predictors:

‘fg%’ (field goal percentage),
‘3p%’ (three-point percentage),
‘stl’ (steals),
‘pts’ (points),
‘trb%’ (total rebound percentage),
‘blk%’ (block percentage),
‘tov%’ (turnover percentage),
‘usg%’ (usage percentage),
‘drtg’ (defensive rating),
‘3p_max’ (maximum three-point field goals for a given player),
‘fta_max’ (maximum free throw attempts for a given player),
‘ft%_max’ (maximum free throw percentage for a given player),
‘stl_max’ (maximum steals for a given player),
‘blk_max’ (maximum blocks for a given player),
‘pts_max’ (maximum points for a given player),
‘+/-_max’ (maximum plus-minus for a given player),
‘drtg_max’ (maximum defensive rating for a given player),

‘mp_opp’ (opponent minutes played),
‘fg%_opp’ (opponent field goal percentage),
‘3pa_opp’ (opponent three-point attempts),
‘blk_opp’ (opponent blocks),
‘blk%_opp’ (opponent block percentage),
‘tov%_opp’ (opponent turnover percentage),
‘ortg_opp’ (opponent offensive rating),
‘3p_max_opp’ (maximum opponent three-point field goals for a given player),
‘3pa_max_opp’ (maximum opponent three-point attempts for a given player),
‘trb_max_opp’ (maximum opponent total rebounds for a given player),
‘blk_max_opp’ (maximum opponent blocks for a given player),
‘+/-_max_opp’ (maximum opponent plus-minus for a given player),
‘ortg_max_opp’ (maximum opponent offensive rating for a given player).

For hyperparameter tuning, we compared activation functions ReLu, hyperbolic tangent, and logistic, hidden layer sizes of (50,), (100,), (50, 50), (100, 100), a solver of stochastic gradient descent, and stochastic gradient descent with adam optimizer, and alpha of 0.0001, 0.001, 0.01 (l2 reg term), and a constant or adaptive learning rate.

The best combination found through grid search consisted of the following parameters: activation= tanh, alpha=0.01, hidden_layer_sizes=(50, 50), learning_rate=constant, solver=adam. This gave a final test accuracy of 61%.

The simulated principle when applying the model over games in the 2022 season is shown below (Figure 5).

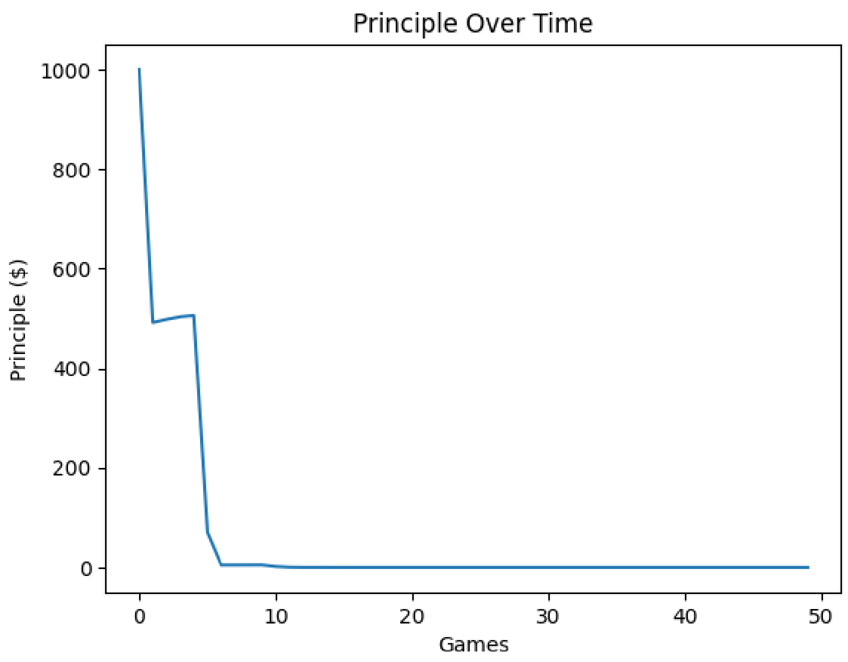

Figure 5: A figure displaying the performance of multilayer preceptron model employing the betting strategy described above

### 7.6 Model Comparison

No features were selected by all five models; however, maximum plus-minus, opponent maximum offensive rating, and usage percentage were used in four of the five, being left out of Decision Tree, Random Forest, and Support Vector Machine respectively.

Random Forest and Support Vector Machine were the slowest to run out of money, with both taking slightly over 20 games to do so. In contrast Logistic Regression took less than 10.

This does not completely follow the test accuracies when applying each model over games in the 2022 season (Table 1): Random Forest has the lowest test accuracy at 56% and is just as successfull as Multi-Layer Perceptron with the highest test accuracy of 61%, while Decision Tree has a higher test accuracy of 60% but fails to perform as well during betting. For reference, a good baseline would be the winrate of the home team. The home team wins 57% of time in the data we use so our models do not outperform this baseline by much.

| **Model Type** | **Accuracy** |
|---|---|
| Baseline (Home Team Win%) | 57% |
| Logistic Regression | 58% |
| Random Forest | 56% |
| Decision Tree | 60% |
| Support Vector Machine | 59% |
| Multi-Layer Perceptron | 61% |

Table 1: Model Accuracies on Test Data

## 8 Difficulties Encountered

We encountered several challenges during this project that hindered our progress.

The first challenge was obtaining data from `basketball-reference.com`. To gather the necessary data, we built a custom web scraper. However, our initial scraping attempts were blocked because we made too many requests too quickly. This forced us to implement delays between requests, but we soon realized that scraping enough data to train our models would take an impractical amount of time on our hardware. As a result, we decided to use a pre-scraped version of the website, which unfortunately provided less data than we had initially planned for.

The second challenge stemmed from the limitations of our computer hardware. Training our models was extremely slow, so we opted for smaller thresholds in forward selection to speed up the process. While this saved time, it likely led to suboptimal models because we were unable to explore the full range of potential features.

Lastly, despite our extensive efforts in feature selection, model selection, and hyperparameter optimization, our models achieved only middling results. We suspect this outcome was largely due to the limited amount of data available for training, which constrained the performance of our models.

## 9 Implications of Work

Overall, our work contributes two key advancements. First, we developed a series of models that demonstrate a slight edge over the house, suggesting that with further refinement, it could be possible to create models with a more significant advantage. Second, we created a robust pipeline for training and testing sports prediction models, offering a structured approach to building and evaluating such systems. Together, these tools provide sports bettors with valuable resources for crafting stronger, data-driven betting strategies.

## 10 Extensions

Our struggles with achieving strong performance in our models highlight several promising directions for future work. One key extension would involve scraping additional data to enhance our dataset. Another idea is to incorporate ELO or ranking data as a feature, capturing the temporal dynamics of team rankings throughout the season. A third avenue for improvement could be modeling the impact of individual players, accounting for trades, injuries, or other factors that significantly affect game outcomes. We could also integrate more detailed shooting statistics, such as floor location data, to refine player-specific performance metrics. Additionally, leveraging sports media—such as news articles, blogs, social media, and cable analysis through NLP and sentiment analysis could provide valuable insights, as these sources often include pre-game predictions and expert commentary. Lastly, exploring more powerful models, such as deeper neural networks, could help achieve higher performance levels.

## 11 Citations

Broas, J. (2021, December 31). Basketball betting dataset. Kaggle. https://www.kaggle.com/datasets/visualize25/basketball-betting-dataset Nguyen, N. H., Nguyen, D. T. A., Ma, B., & Hu, J. (2021). The application of machine learning and deep learning in sport: predicting NBA players' performance and popularity. Journal of Information and Telecommunication, 6(2), 217–235. https://doi.org/10.1080/24751839.2021.1977066

Paruchuri, V. (n.d.). Project-walkthroughs/nba_games at master · dataquestio/project-walkthroughs. GitHub. https://github.com/dataquestio/project-walkthroughs/tree/master/nba_games

Thabtah, F., Zhang, L. & Abdelhamid, N. NBA Game Result Prediction Using Feature Analysis and Machine Learning. Ann. Data. Sci. 6, 103–116 (2019). https://doi.org/10.1007/s40745-018-00189-x

Why US sports betting could become a $45 billion business. (2024). Goldmansachs.com. https://www.goldmansachs.com/insights/articles/why-us-sports-betting-could-become-a-45-billion

Zimmermann, A., Moorthy, S., & Shi, Z. (2013). Predicting college basketball match outcomes using machine learning techniques: some results and lessons learned. arXiv preprint arXiv:1310.3607.