Data Driven Player Efficiency Rating (PER) Prediction For NBA Draft Prospects

Sam Peters, Ethan Ogle, Matthew Mullis, Joseph Murphy

CSCI 4380

Dec 5, 2025

Abstract

This project tries to determine whether college statistics, NBA Draft Combine measurements, and biographical information can predict a player's Rookie Player Efficiency Rating (PER). Using three public datasets to create a single unified table, we trained several machine learning models including linear regression, tree-based ensembles, and a 3 layer neural network. XGBoost achieved the best performance while the neural network performed the worst. Our findings show that Rookie PER could have strong performance from pre-draft features, the results are sensitive to data limitations and the limitations of certain models.

1 Introduction

Predicting NBA players' performance has always been a challenge for teams, scouts, and analysts. Draft decisions can determine how the future of a franchise performs. Statistics and athletic measurements help describe who a player is before the draft, but the data can be very noisy and it's unclear how much that information predicts how well a player performs when they are in the NBA. Understanding pre-draft data would be very meaningful in predicting professional performance and is a great way to evaluate new talents.

Player Efficiency Rating (PER), created by John Hollinger, is one of the more popularly used ways to rate a player's overall per-minute impact. PER uses scoring, rebounding, passing, and defensive events, creating a single metric that reflects a player's on-court presence. By focusing on Rookie PER, we have a good measure of how effectively a player transitions into the NBA environment during their first season.

In this project, we investigated if Rookie PER can be predicted strictly from pre-draft data, specifically college performance, NBA Draft Combine measurements, and basic biographical details. These datasets capture a prospect's production, athletic profile, and position capabilities, which together could make patterns that contribute to early-career success.

To explore this more, we combined several datasets into a singular unified table, and trained several machine learning models. The goal is to compare the models and to try to understand if Rookie PER is easily predicted using pre-draft information.

2 Literature Review

2.1 Individual Studies

1. Several studies have tried to link pre-draft performance to professional success. (1)Jain used *beta regression* with random intercepts to model normalized PER from college and NBA Combine data; the study found that a player's *position*, *weight*, *rebounds*, *field-goal percentage*, *height* and *vertical leap* were the most important predictors of normalized PER. Beta regression accounted for the bounded

nature of PER and allowed longitudinal trends to be captured through random effects. This work provides a modelling framework and identifies variables that influence PER.

2. Chen et al. developed a neural-network model to predict the NBA Most Valuable Player (MVP) using performance data from 1997-2019. Although their goal differs from ours, their method demonstrates that neural networks trained on historical player statistics can forecast award outcomes. They trained and tested their network using season-by-season performances; the model correctly predicted MVPs for specific seasons. This suggests that non-linear models can capture complex relationships in basketball data.
3. Greene examined whether college careers predict NBA success by analysing first- and second-round picks from 1985-2005. He used endpoints such as PER, Win Shares and Win Shares per 48 minutes to assess success. Greene's study provides evidence that collegiate performance and physical attributes can forecast professional productivity, supporting the feasibility of our proposed work
4. Coates and Oguntimein studied whether college production predicts both the length and success of NBA careers for players drafted between 1987 and 1989, using regression models that linked collegiate box-score metrics to professional outcomes such as games played, minutes, and overall career value. They found that stronger college performance is associated with better draft position and longer, more productive NBA careers, although the predictive power varies across different outcome measures.
5. Berri and Schmidt analyzed whether college performance reliably predicts NBA productivity by examining draft outcomes and player efficiency metrics across multiple seasons. Their study found that many commonly emphasized college statistics, such as points per game, are poor predictors of NBA success, while efficiency-based measures (like shooting percentages and rebounding rates) translate more consistently to the professional level. They also showed that NBA teams often overvalue scoring volume and undervalue efficient, well-rounded players during the draft process. This work highlights the importance of emphasizing efficiency metrics over raw output, a principle that aligns closely with our use of variables like true shooting percentage and per 40 statistics in modeling rookie PER.

2.2 New Contribution

Together, these studies show that predicting NBA performance from pre-draft information is possible but can be limited by the type and quality of the data available. Several papers (Greene; Coates & Oguntimein; Berri & Schmidt) emphasize that statistics from college alone can have useful data, but the most predictive features tend to be efficiency based metrics rather than raw scoring. Jain's results extend this idea by showing that physical measurements and athletic testing also contribute meaningfully, especially when combined with position information. Meanwhile, Chen et al. highlight the benefits and limitations of neural networks on basketball analytics. They can model more complex relationships, but that's if they're trained on large, rich, well structured datasets. Our project builds on these ideas by merging different college, combine, and rookie datasets into one unified table then systematically comparing models trained on it. By evaluating the different models we can try to learn more about machine learning models' abilities to extract reliable information specifically from pre-draft features.

3 Data Summary

In order to look at how college and NBA draft combine performances could predict rookie year efficiency, we used 3 datasets from Kaggle that had both useful metrics and a wide range of years of collection so that there would be sufficient overlap in order to train models.

3.1 College Basketball Statistics (by Aditya Kumar)

The first dataset has season-level college statistics for NCAA players from 2009 through 2021 and includes both basic and advanced box-score metrics:

- Points, rebounds, assists, steals, blocks
- Shooting percentages (FG%, 3P%, FT%)
- Usage rate, minutes per game
- Personal fouls, turnovers
- Total games played

- Basic biographical information (height, weight, class year, position)

3.2 NBA Draft Combine Data (by Marcus Fern)

The second dataset contains measurements and athletic testing results from the annual NBA Draft Combine from 2000 to 2025, and these features help quantify a player's physical profile and athleticism:

- Height (with and without shoes)
- Wingspan
- Standing reach
- Weight and body fat percentage
- Lane agility time
- Shuttle run time
- Three-quarter sprint
- Max vertical leap and standing vertical leap
- Hand length/width
- Position and shooting drill results (where available)

3.3 Rookie-Year NBA Statistics (by Gabe Salzer)

The third and last dataset contains performance statistics for NBA rookies from 1980 to 2016. This dataset includes players' first year production and PER, which serves as the target variable for the models:

- Rookie minutes per game
- Points, rebounds, assists, steals, blocks
- Shooting percentages, turnovers, games played
- Rookie PER (target variable)

3.4 Combined Dataset

After merging the three datasets by player and draft year, we made one unified dataset. Once cleaned we still have draft prospects from 2009-20016 who had college statistics, NBA Draft Combine measurements and rookie PER performance scores.

We had 217 rows, representing the players and 97 total features including standardized college box-score statistics, shooting efficiency, physical and athletic metrics, and other engineered ratios. The target variable Rookie PER is a per-minute efficiency metric reflecting a player's on court contribution during their rookie season.

For the merged dataset, the rookie PER had a high standard deviation of 3.98 for a mean of 7.16. The range is between 2.10 and 23.80 which is quite wide, indicating the variability in how players transition to the NBA.

4 Preprocessing and Feature Engineering

The first step was to merge the datasets across the overlap in years, 2009 to 2016 such that each row represented a player that had stats recorded in college, the NBA draft combine, and as a NBA rookie. To do this, we also needed to standardize the column naming convention across all datasets, which would also make for easier analysis in the future. Next, in order to ensure our data was suitable for modeling, we prepared our dataset by performing basic data cleaning: removing redundancy, fixing missing values, etc.. We removed irrelevant, redundant, or incomplete variables from our data, leaving a set of features with clear relevance to prediction. Numerical and categorical variables were looked at for quality, including through checks of missing values, collinearity, and variance close to zero. Missing numerical values were given the median of the value, and categorical values were given the most frequent category, while variables with no observed values were removed. We then converted our categorical variables into numerical form to prep them for our use. Finally, we standardized our numerical features so that scaling was consistent across models (especially for algorithms sensitive to magnitude). Outliers were capped rather than deleted as to not fully remove extreme cases, and when necessary, transformations were used on skewed distributions to keep variance stable. Once all of these steps were completed, our data had relevant, nonmissing, nonredundant values for our models.

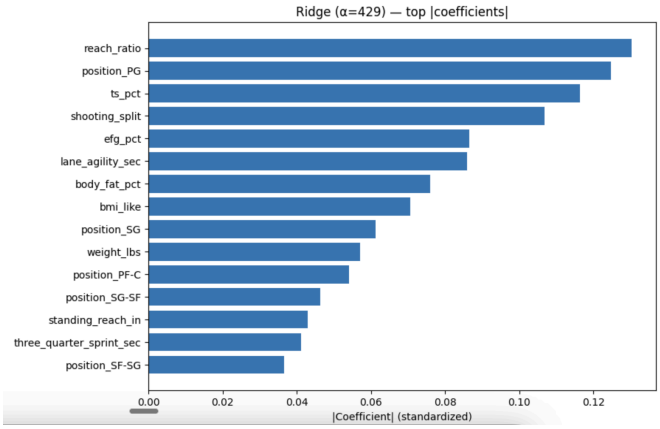
5 Modeling

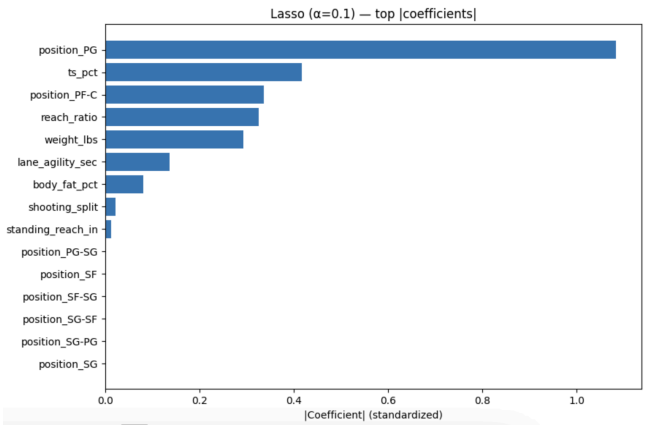
We treated rookie PER as a continuous regression target and compared several model families with increasing flexibility. All models shared a common preprocessing pipeline. Numeric features (college efficiency and per 40 stats, combine measurements, and engineered ratios) were computed using the median and standardized using StandardScaler. Categorical features, primarily position, were imputed with the most frequent category and encoded using one-hot encoding with unknown categories ignored at prediction time. This preprocessing was implemented via a column transformer and sklearn pipelines so that imputation, scaling, and encoding steps were consistently applied inside cross-validation and hyperparameter tuning. For the non-neural models (linear regression, ridge, lasso, decision tree, random forest, and XGBoost) were evaluated using 5 fold cross validation. With the 3 layer neural network evaluated using a single 80/20 train-test split.

As a baseline, we first fit an ordinary linear regression model on a restricted feature set: college shooting efficiency (eFG%, TS%), basic combine measurements (height with shoes, wingspan, standing reach, weight, body-fat percentage, vertical jumps, agility times, sprint times, bench reps), and position. This gave us a simple linear benchmark to compare against more complex models.

5.1 Ridge and Lasso

After the linear baseline, we moved to regularized linear models to address multicollinearity and perform soft feature selection. We used Ridge regression (L2 penalty) and Lasso regression (L1 penalty) on an expanded feature set that included the engineered variables (wingspan/height ratio, reach/height ratio, power-to-weight, BMI-like metric, shooting split, and usage proxy where available). For Ridge, we used RidgeCV with a log-spaced grid of α values ranging from 10^{-3} to 10^{3} and 5-fold cross-validation based on mean squared error. For Lasso, we used LassoCV with an initial log-spaced grid between 10^{-3} and 10^{1} and expanded to smaller α values when the model collapsed to all-zero coefficients. Both models were wrapped in the shared preprocessing pipeline so that scaling and imputation were fit only on training folds. The resulting Ridge model retained all standardized features with shrunk coefficients, whereas the final Lasso model selected a small subset of predictors with non-zero weights, enabling easier interpretation of which pre-draft variables were most associated with rookie PER.





5.2 Decision Tree

We also used a preprocessing pipeline for our decision tree. After preprocessing, we used k-fold cross-validation so we could get an estimate of the accuracy of prediction without biases. We tuned the max_depth, min_samples_split, and criterion hyperparameters in order to improve the accuracy with interpretability. Information gain and gain ratio were also computed for key features as to compare the split choices on the decision tree made by scikit-learn. The resulting tree was decently interpretable, at least showing a clear hierarchy and displaying which variables were found to be most predictive. Because decision trees tend to overfit, the deeper trees showed high training accuracy but worsened generalization through cross-validation. Minimizing depth and constraining split size produced a more stable model with decent predictive power. The importance of each feature that we found shows the attributes strongly associated with our response variable, and comparison with information gain further shows why features were chosen.

5.3 Random Forest

We used the same preprocessing pipeline for the random forest. During development of our model, we tuned hyperparameters such as n_estimators, tree depth, and maximum feature sampling. Each split had random feature selection so the individual trees contained different structural patterns, reducing risk of overfitting to any set of predictors. Our resulting random forest model displayed better generalization compared to the individual decision tree, with smoother and more stable performance across folds. Scores of feature importance found using impurity based metrics shined light on a more even contribution of multiple

variables instead of heavily relying on a single split. The ensemble behavior also kept the model from being heavily affected by noise and outliers. Though the random forest is less predictable than the tree, the improved accuracy and better variance makes it a stronger model comparatively.

5.4 XGBoost

We again used the preprocessed dataset but applied gradient boosted decision trees this time, iteratively improving performance by fitting trees to the errors of earlier trees. The learning rate, maximum depth, and number of boosting round hyperparameters were tuned with cross-validation. Since XGBoost is sensitive to scaling and sparsity, preprocessing steps such as feature encoding and controlled handling of missing values were important to ensure stability. The resulting model showed the strongest predictive performance of the three previous ones. Its structure allowed it to capture nonlinear relationships and unique interactions that decision trees and random forests can not easily model. Analysis of feature importance showed which predictors contributed most, giving insight into performance improvement across the boosting rounds. Even though XGBoost is the least interpretable of the three models (due to ensemble depth and its structure), it provided the highest accuracy and most consistency across cross-validation.

5.5 3 Layer Neural Network

We also implemented a feed forward neural network to test whether a nonlinear model could capture higher order interactions among pre-draft features. The network consisted of fully connected layers with ReLU activation. We experimented with hidden layer sizes of 64 and 128 units, optional dropout (0.0-0.2), and Adam learning rates between 0.0005 and 0.001. All categorical variables were one-hot encoded, and all numeric features were standardized using the shared preprocessing pipeline so the neural network received only numeric inputs. We trained each configuration for 40 epochs using mean-squared error as the loss function and mini-batch gradient descent (batch size = 32). Hyperparameters were tuned manually because the small dataset and long training times made grid search impractical. The goal was to evaluate whether the network could uncover nonlinear structure that linear models could not.

6 Results and Analysis

6.1 Ridge and Lasso

The plain linear regression baseline performed poorly: it achieved a cross-validated MAE of about 3.26, RMSE of 4.25, and an R^2 of approximately −0.24, indicating that it did worse than predicting a constant mean PER. Ridge regression modestly improved on this baseline. With a best α of roughly 4.3×10^2, the in-sample scores were MAE ≈ 3.02, RMSE ≈ 3.90, and R^2 ≈ 0.04. This indicates that even after regularization and expanded features, the linear model could only explain around 4% of the variance in rookie PER and still exhibited fairly large errors. Lasso regression, after zooming in on smaller α values to avoid an all-zero solution, achieved similar accuracy with MAE ≈ 3.10, RMSE ≈ 3.97, and R^2 close to 0. The main value of Lasso was interpretability: by shrinking many coefficients exactly to zero, it highlighted a small subset of features as the most relevant. Coefficient inspection showed that position (especially being classified as a point guard or power-forward/center), true shooting percentage, reach-to-height ratio, and weight carried relatively larger standardized coefficients. Lane agility time and body-fat percentage also appeared with non-negligible weights. This aligns with basketball intuition that efficiency, positional context, length, and basic athleticism matter, but the weak R^2 suggests these factors alone do not let a linear model accurately forecast rookie PER.

6.2 Decision Tree

The decision tree prior to tuning performed fairly poorly with an MAE of 3.63, an RMSE of 4.79, and a negative R2 of -0.61. This suggests overfitting and low generalization for our model. After tuning our hyperparameters by restricting max_depth to 2 and number of minimum samples per split to 5, results of our tree improved greatly. The tuned version had an MAE of 2.51, RMSE of 3.20, and an R2 of 0.35 (much better). Though improved, this is still a pretty modest R2. The results of feature importance showed that only a small number of features were consistently selected, which shows the decision tree's greedy splitting process. The strongest split had the most information gain, and gain-ratio checks displayed the tree's internal choices. Despite improved metrics after it was tuned, the results for the decision tree are still weak due to the high variance and sensitivity to single predictors.

6.3 Random Forest

The random forest without tuning gave much better stability and predictive accuracy compared to the individual decision tree with an MAE of 2.82, RMSE of 3.58, and R2 of 0.13. Tuning parameters like number of trees to 200, max_depth to 6, and min_samples_split to 5 improved performance further. The tuned random forest had a MAE of 1.46, RMSE of 1.83, and R2 of 0.79 which indicates strong explanatory power. The distribution of feature importance was more even than the decision tree, with a range of features contributing moderate amounts rather than a select few dominating. This makes sense due to the random forest's bootstrapping of samples since this would reduce bias toward any single predictor. The model showed high stability across folds and showed no sign of excessive overfitting. In total, the model gave excellent prediction accuracy and reliability.

6.4 XGBoost

XGBoost had the best performance in baseline and after tuning compared to decision trees and random forest. At its base, it had an MAE of 2.87, RMSE of 3.75, and R2 of 0.01 which is low. After tuning max_depth to 3, learning_rate to 0.05, and n_estimators to 100, the model now had an MAE of 1.15, RMSE of 1.42, and R2 of 0.87 which is the best among all models, easily making it the best between the three models. Results of feature importance showed that XGBoost assigned larger total gain scores to a set of good predictors that continued to contribute effective splits. Unlike random forest, this model found unique interactions between features, and improvements in our metrics display its adequacy. In total, XGBoost was our best-performing model due to its low error and high R2.

6.5 3 Layer Neural Network

Among all the models, the neural network produced the worst results. After testing, a few configurations with differing hidden layer size, dropout rate, and learning rate. The network failed to converge to a meaningful result. The best achieved MAE was between 2.26-3.16, RMSE values between 3.13-4.11, and R2 values between -0.32 and -1.25. One key observation we made was that as the number of epochs increased, the model tended to overfit the data. During training, the loss on the training set decreased, and the MAE and RMSE remained relatively high. All the R2 values were negative across all the different configurations, which

implies it fitted the model on noise rather than learning any patterns. The dataset is also small, heavily one-hot encoded for the size, and relatively sparse, making it difficult for the model to perform well. Neural networks require larger, denser datasets with higher signal, so it's understandable that it didn't work well on this dataset.

7 Discussion and Future Work

7.1 Discussion

The results of this project show the challenges of predicting Rookie PER from pre-draft data alone. For some models they were able to perform well on the 5 fold cross-validation, such as the tuned Random Forest and the XGBoost. Those R2 show that around 79-87% of the variance. Since the models aren't perfect there's still unexplained variance, suggesting that some important factors affecting rookie PER are not in the dataset.

One notable observation was the difference between the linear models and the tree-based models. Ridge and Lasso identified plausible predictors from the features like efficiency metrics, position, and physical measurements, giving some interpretability about the relationships. But they struggled with nonlinear relationships that are a part of the complexity of basketball performance.

The ensemble tree models handled the interactions between physical and skill-based features better and produced far more stable predictions, and agrees with existing work that boosted trees perform well with structure tabular data especially compared to neural networks and linear models.

The poor performance of neural networks on this dataset also agrees with existing work. Since the dataset is pretty small and is sparse due to one-hot encoding, the network overfit quickly and fails to generalize. Deep learning has had success in basketball analytics (MVP prediction or player tracking models), the applications typically had richer, denser, and just over better datasets. The lack of high signal features limits much of what models can learn from purely pre-draft information.

Overall, our results from experimenting suggest that while pre-draft college and combine metrics hold little predictive value, they are not enough to accurately estimate a rookie's PER. Rookie PER has to be influenced

by more factors that are in the dataset we used, and the models that handle these weaknesses still struggle to be useful for predicting a rookie's transition into the NBA.

7.2 Future Work

In the future, several things could be done to help make Rookie PER more predictable and reliable.

- We can include richer pre-draft features
    - Play-by-play efficiency
    - Shot-chart efficiency
    - BPM, RAPM (impact metrics)
- Expand the sample size
    - Collecting a more uniformed dataset rather than unifying several datasets into one merged dataset, allowing for more samples with more uniformity.
- More advanced hyperparameter optimization (for neural networks)
    - Allowing for optimized tuners to find better/best model configurations.

8 Bibliography

This project will follow MLA citation style. The most significant references include:

- Hollinger, John. *Player Efficiency Rating*. 2002. A foundational text defining PER and explaining the formula. PER measures per-minute productivity and adjusts for pace.
- Jain, Areen. "Forecasting NBA Players' Efficiency: Integrating College and Combine Predictors with Beta Regression." *Intelligence Planet Journal of Mathematics and Its Applications*, vol. 2, no. 3, 2025, pp. 1-10. The study models normalized PER using beta regression and identifies key predictors such as position, weight, rebounds, field-goal percentage, height and vertical leap.
- Chen, Yuefei, et al. "A Neural Network Model of the NBA Most Valued Player Selection Prediction." *International Conference on Pattern Recognition and Artificial Intelligence*, 2019, pp. 1-5. This conference paper trains a neural network on NBA performance data to predict MVPs and shows that non-linear models can correctly identify award winners.

- Greene, Alexander C. “The Success of NBA Draft Picks: Can College Careers Predict NBA Winners?” 2015. This study analyzes 841 draft picks from 1985-2005 and uses metrics such as PER, Win Shares and Win Shares per 48 minutes to assess whether college performance predicts NBA success.
- Coates, Dennis, and Babatunde Oguntimein. “The Length and Success of NBA Careers: Does College Production Predict Professional Outcomes?” *International Journal of Sport Finance*, vol. 5, no. 1, 2008, pp. 15-28. Although not directly cited above, this paper uses 1987-1989 draft data to evaluate the relationship between college productivity and professional success; it will inform our modelling choices.
- Berri, David J., and Martin B. Schmidt. “College Performance and Professional Basketball Success: The Case of the NBA Draft.” *Journal of Productivity Analysis*, vol. 35, no. 1, 2011, pp. 25–35.