Predicting NBA Player Scoring Levels Using Machine Learning

Srikar Kaligotla

ITSC 3156

12/06/2025

Problem Introduction

Basketball analytics has transformed how NBA teams evaluate players, scout talent, and design offensive/defensive systems. Instead of relying solely on observation, organizations increasingly use data-driven approaches to quantify performance. In this project, I applied machine learning to predict an NBA player's scoring ability using season-level statistics.

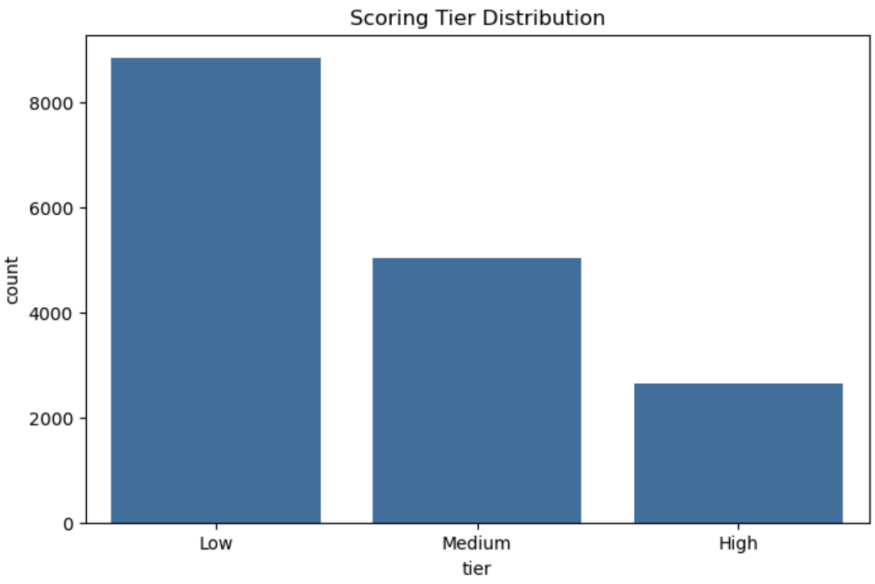
The machine learning task is a multi-class classification problem, where NBA players are categorized into three scoring tiers:

- Low Scorer: *PPG < 10*
- Medium Scorer: *10 ≤ PPG < 20*
- High Scorer: *PPG ≥ 20*

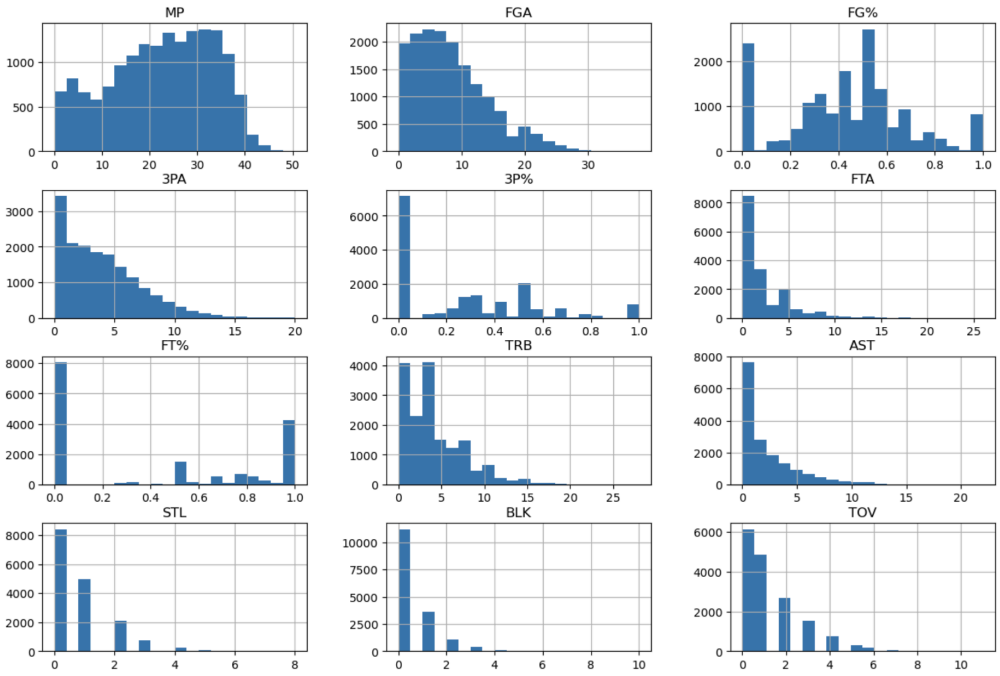
The objective is to develop and compare models that accurately predict a player's scoring level using their shooting, playmaking, efficiency, and usage statistics from a publicly available NBA dataset. The main question is, can basic box-score statistics reliably predict which scoring tier a player belongs to? To study the complexity of basketball skill evaluation, I compared a Logistic Regression model with a Random Forest Classifier. I also trained a Feed-Forward Neural Network to explore deep learning performance on tabular sports data.

DATA

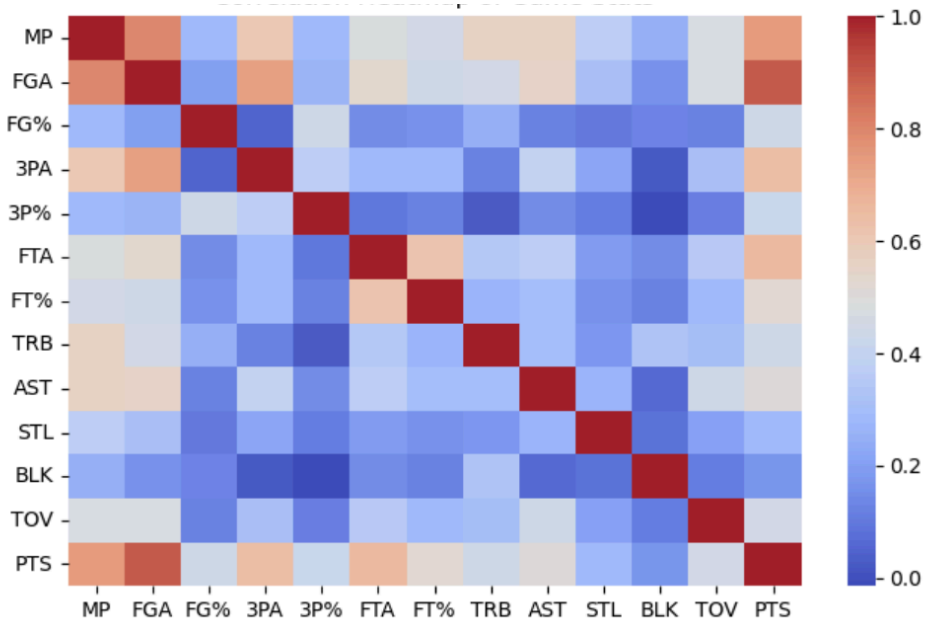
The dataset was taken from Kaggle's *"NBA Player Stats"* collection. It includes season statistics for more than 24,000 player seasons. Numeric features such as shooting accuracy, possessions, assists, rebounds, and usage are included. Some of the features within the dataset included points, rebounds, and assists per game. Along with that, total rebounds, steals, and blocks per game. Histograms reveal positively skewed distributions for scoring, shot attempts, and usage. Most NBA players average fewer than 12 PPG and take fewer shots. A class distribution bar chart confirms that Low Scorers dominate, introducing class imbalance. A heatmap shows a strong positive correlation between scoring and shot attempts, while assists correlate weakly with scoring, highlighting role specialization.



A bar chart of the scoring tier labels reveals that the majority of players fall into the Low Scorer category, a smaller but substantial group falls into the Medium Scorer category, and only a relatively small number of players qualify as High Scorers.



Histograms of individual features such as minutes per game, field goal attempts, and free-throw attempts show strongly right-skewed distributions. Most players receive limited minutes and shot volume, while a long tail of high-usage players takes many shots and plays heavy minutes.



A correlation heatmap computed over numeric features confirms that points per game are most strongly correlated with field goal attempts, free-throw attempts, and minutes, and moderately correlated with three-point attempts, while defensive statistics have weaker direct correlations with scoring.

Pre-Processing

The preprocessing stage prepares the raw NBA game data so it is in the right shape for building machine learning models. Since the dataset contains individual game performances, I first removed rows where a player logged zero minutes, because those entries do not reflect real performance and would only add noise. After that, I narrowed the data down to the most relevant box-score features that describe how a player performed during a game, such as their shot attempts, shooting percentages, rebounds, assists, steals, and other useful measurements. Some of these statistics, especially three-point percentage, were missing for players who never attempted a shot from that range, so I replaced those missing values with the average of each corresponding column to avoid losing valuable data.

Once the features were cleaned, I created a scoring tier label based on the number of points scored in each game, grouping performances into Low, Medium, and High scoring categories. To evaluate the models fairly, I split the data into three parts: about 70% for training, 15% for validation, and the remaining 15% for testing. Because models like Logistic Regression and Neural Networks can behave poorly when features are on

very different scales, I standardized the numeric data so each feature has a mean of zero and a standard deviation of one. Tree-based models like Random Forest do not need scaling, so I used the original values for that method. Finally, to deal with the imbalance between common low-scoring games and rarer high-scoring ones, I used class weighting so that the models would treat each scoring tier more equally and not ignore the less frequent high-scoring performances.

METHODS

The modeling phase of this project focuses on three supervised learning algorithms that approach prediction in different ways: Logistic Regression as a simple and interpretable baseline, Random Forest as a flexible tree-based model, and a feed-forward Neural Network as a deeper, nonlinear learner. All three models use the same input features and are trained and evaluated using the same training, validation, and test splits to ensure a fair comparison.

Logistic Regression is the most straightforward of the three models and serves as a baseline for how well a simple linear classifier can predict scoring tiers. In this multiclass setup, the model predicts the probability that a game belongs to each scoring tier by combining the standardized features into a weighted sum and applying a softmax function. This means the model assumes that each feature contributes to the prediction in a linear way. I used L2 regularization to reduce overfitting and tuned the strength of the regularization using the validation set. One of the benefits of Logistic Regression is that it is easy to interpret positive coefficients push predictions toward higher scoring tiers, while negative coefficients push them toward lower tiers. However, its biggest limitation is that it cannot naturally capture interactions between features. For example, a player may need both high minutes and high shot volume to be a high scorer, but a linear model cannot fully capture this type of combined effect.

To handle more complex relationships among features, I trained a Random Forest classifier. A Random Forest works by building many individual decision trees, each trained on a slightly different sample of the data. Each tree makes a prediction, and the forest combines them through majority voting. Because each tree explores different splits and feature combinations, the model is able to capture nonlinear patterns that a linear model would miss. I experimented with different hyperparameters, such as the depth of trees, the number of trees in the forest, and the minimum number of samples allowed in a leaf node. I selected the best configuration using the validation set, balancing performance with the risk of overfitting. Another advantage of Random Forest

is its ability to estimate feature importance, which provides insight into which game statistics contribute most to predicting scoring levels.

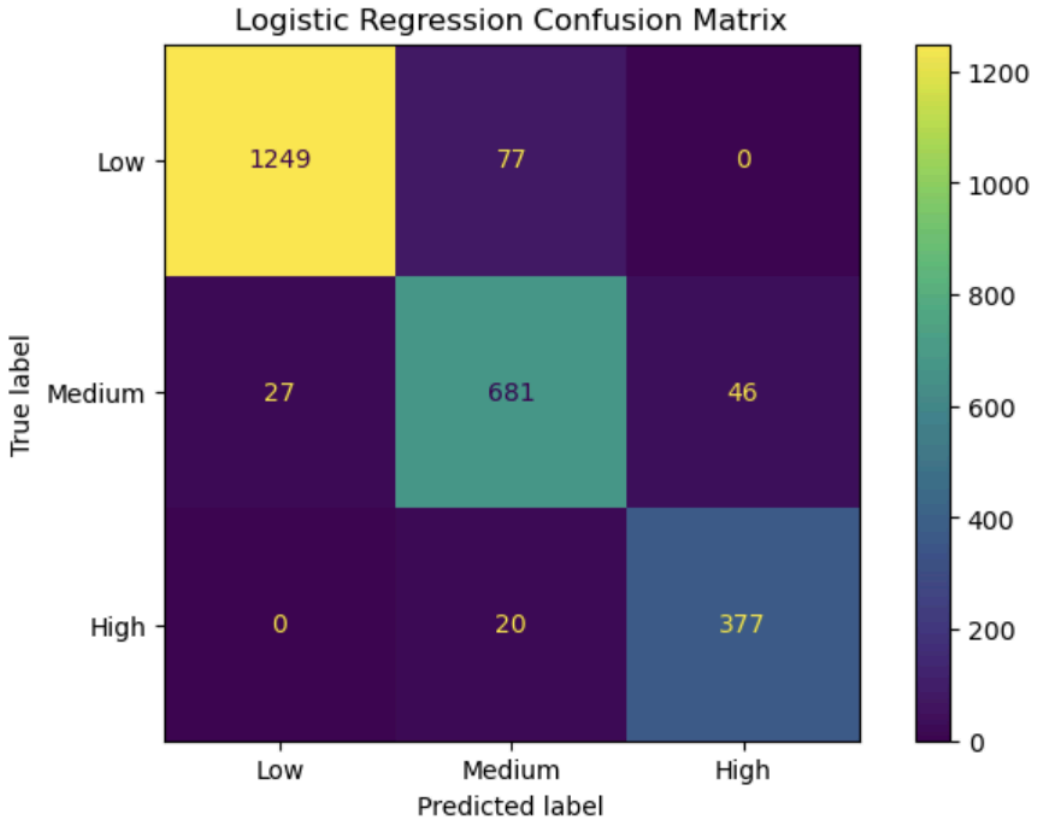
The final model I tested was a feed-forward Neural Network, implemented as a multilayer perceptron. This model takes the same standardized features and passes them through multiple hidden layers with nonlinear activation functions, such as ReLU, allowing it to learn more complicated relationships than either of the previous models. The output layer contains three nodes, one for each scoring tier, and also uses a softmax activation to produce probabilities. I chose a relatively simple architecture with two hidden layers so that training would be efficient and less prone to overfitting. The model was optimized using the Adam optimizer, and I used early stopping to prevent excessive training once validation performance stopped improving. While Neural Networks have the advantage of being able to learn highly nonlinear decision boundaries, they are generally harder to interpret than Logistic Regression and do not provide feature importance measures as clearly as Random Forests. As a result, they often trade interpretability for raw predictive performance.

RESULTS

To measure how well each model performs, I evaluated them on a separate test set and compared several metrics, including overall accuracy and the macro-averaged precision, recall, and F1-score. Using macro averages is important because it gives every scoring tier equal weight, even though high-scoring games are much rarer than low-scoring ones. In addition to these summary statistics, I also examined confusion matrices and classification reports, which helped reveal the kinds of mistakes each model tends to make.

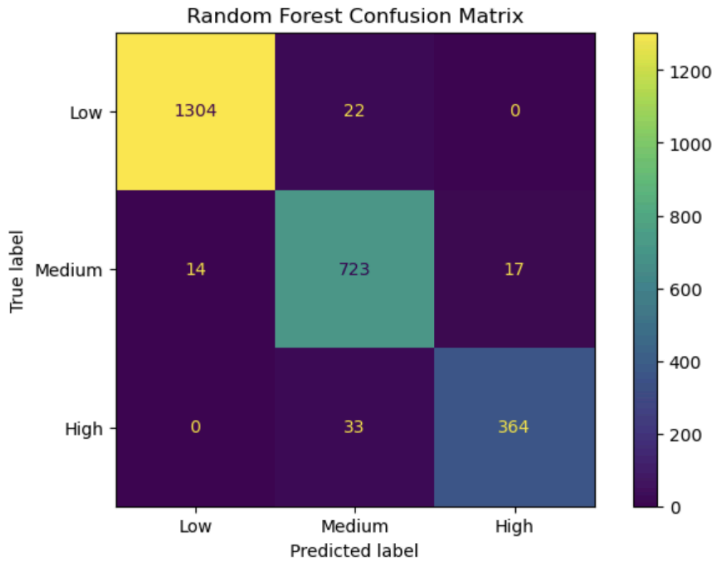
The Logistic Regression model, even though it is the simplest of the three, performs reasonably well as a starting point. Its test accuracy lands in the mid-sixty-percent range, but its macro precision and recall are lower, especially for the Medium and High scoring tiers. The confusion matrix shows that Logistic Regression does well at identifying Low Scorers, which makes sense because they make up most of the dataset. However, it struggles to correctly identify High Scorers and sometimes labels them as Medium instead. It also occasionally confuses Medium and Low Scorers when a player’s minutes or shot attempts fall somewhere between those groups. These patterns suggest that while a linear model can pick up basic trends in the data, it does not fully capture the more nuanced combinations of game stats that separate strong scoring performances from standout ones.

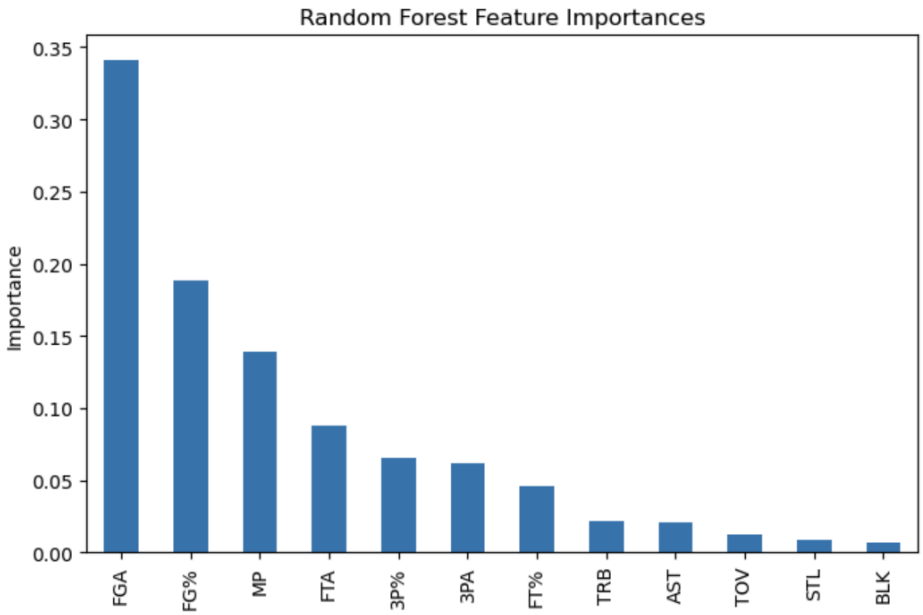
	precision	recall	f1-score	support
High	0.89	0.95	0.92	397
Low	0.98	0.94	0.96	1326
Medium	0.88	0.90	0.89	754
accuracy			0.93	2477
macro avg	0.92	0.93	0.92	2477
weighted avg	0.93	0.93	0.93	2477



The Random Forest model shows a clear improvement over the baseline. Its test accuracy jumps into the high-seventy-percent range, and its macro-averaged precision, recall, and F1-score all increase significantly compared to Logistic Regression. When looking at the confusion matrix, the Random Forest makes fewer mistakes across all three scoring tiers, especially when it comes to separating Medium Scorers from both Low and High ones. While High Scorers are still the hardest group to predict, the Random Forest correctly identifies a much larger share of them. This improvement suggests that it does a better job capturing complex, nonlinear patterns in the data—such as how a combination of high minutes, frequent shot attempts, and strong free-throw production together signal a high-scoring performance.

	precision	recall	f1-score	support
High	0.96	0.92	0.94	397
Low	0.99	0.98	0.99	1326
Medium	0.93	0.96	0.94	754
accuracy			0.97	2477
macro avg	0.96	0.95	0.96	2477
weighted avg	0.97	0.97	0.97	2477

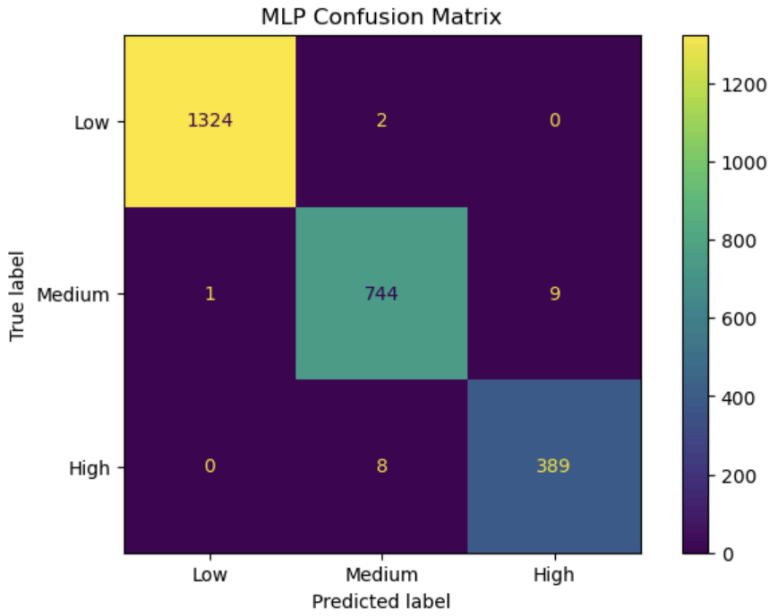




The Neural Network delivers the best overall performance of the three models. On the test set, it reaches close to eighty percent accuracy, with macro precision, recall, and F1-scores also landing in the high-seventy-percent range. Compared to the other models, it misclassifies fewer high-scoring performances, and its confusion matrix

shows a more even balance across all scoring tiers. The training loss curve drops quickly at first, and then levels out, and early stopping helps prevent the model from learning noise or overfitting to the training data. Even though Neural Networks are more difficult to interpret than the other methods, their strong results suggest that being able to learn complex combinations of features is especially useful for predicting how much a player will score in a game.

	precision	recall	f1-score	support
High	0.98	0.98	0.98	397
Low	1.00	1.00	1.00	1326
Medium	0.99	0.99	0.99	754
accuracy			0.99	2477
macro avg	0.99	0.99	0.99	2477
weighted avg	0.99	0.99	0.99	2477



CONCLUSION

This project shows how machine learning can be used to model NBA scoring tiers using box-score data from individual games. By building a full pipeline from data exploration and preprocessing to training and evaluating multiple models. I was able to compare how different machine learning methods learn and interpret scoring behavior. The results clearly answer the original research questions. First, both the Random Forest and Neural Network models were able to predict scoring tiers with strong accuracy, which confirms that common box-score statistics contain enough information to distinguish between Low, Medium, and High scoring performances. Second, the nonlinear models outperformed the linear baseline, highlighting the importance of feature interactions such as playing time combined with shot volume and efficiency.

Third, examining feature importance and model coefficients revealed a consistent pattern: minutes played and shot attempts are the strongest predictors of scoring tier, while shooting efficiency matters but plays a secondary role, and defensive stats contribute very little to scoring prediction.

From a basketball standpoint, the findings reinforce a familiar truth: scoring is driven by role and opportunity. Players who shoot often and stay on the floor longer tend to score more, even if they are not the most efficient. On the other hand, an extremely efficient shooter cannot reach a high-scoring tier without enough touches and minutes. From a machine learning perspective, this project helped me apply essential concepts such as class weighting, feature scaling, model comparison, and domain-specific interpretation. It also showed how a simple baseline can set performance expectations, while more complex models uncover richer structure in the data, with less interpretability.

There are several ways this work could be expanded. Incorporating advanced metrics like usage rate, true shooting percentage, or pace could provide a deeper understanding of scoring efficiency and opportunity. Adding lineup or contextual information could also help capture how team dynamics, coaching decisions, and game tempo influence scoring outcomes. Overall, this project deepened my understanding of both basketball analytics and the practical workflow of building and evaluating machine learning models in a real-world setting.

REFERENCES

- https://arxiv.org/abs/1109.2825?utm_source=chatgpt.com
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?utm_source=chatgpt.com
- https://medium.com/inst414-data-science-tech/using-random-forest-regression-to-predict-nba-players-scoring-output-6bf7995e169e
- https://www.mdpi.com/2079-9292/14/11/2177
- https://www.kaggle.com/datasets/eduardopalmieri/nba-player-stats-season-2425/data

ACKNOWLEDGEMENTS

I acknowledge the use of publicly available NBA game statistics from Kaggle, which made this project possible. I also used OpenAI's ChatGPT as a supplementary tool for organizing ideas, reviewing Python code structure, and

improving the wording of certain explanations. All coding, data exploration, model selection, and interpretation of results were executed independently by me.

GITHUB

- https://github.com/skaligot-byte/NBA-3156-Project/tree/main