

Assignment

1

Kaggle Competition

Shi Wu
Student

ID:

24623845

26th August 2023

36120 - Advanced Machine Learning Application
Master of Data Science and Innovation
University of Technology of Sydney

Table of Contents

1. Executive Summary	2
2. Business Understanding	3
a. Business Use Cases	3
b. Key Objectives	4
3. Data Understanding	5
4. Data Preparation	7
5. Modeling	7
a. Approach 1: LightGBM without feature engineering	8
b. Approach 2: LightGBM with feature engineering	8
c. Approach 3: stacking (LightGBM + Random Forest)	9
6. Evaluation	10
a. Evaluation Metrics	10
b. Results and Analysis	10
c. Business Impact and Benefits	11
d. Data Privacy and Ethical Concerns	12
7. Deployment	13
a. Process of deployment	13
b. Challenges	14
c. Recommendations	14
8. Conclusion	15
References	16

1. Executive Summary

Players not previously drafted are new to the NBA draft process. The draft is a big step in a player's basketball career. Fans and analysts watch closely to see which players are chosen by teams and guess how they might do in the NBA. This project uses a machine learning tool to predict if a college player will get drafted based on their game stats. This helps teams figure out what makes a player more likely to be drafted. Players also learn what they need to work on to improve their chances. Overall, this model gives teams better information to make smart draft choices.

Additionally, In this project we aim to determine the correlation between a college player's season performance and their probability of being drafted into the NBA. Understanding the link between college statistics and draft probability can greatly optimize the processes and decisions related to scouting, investing, and planning in the NBA.

2. Business Understanding

Annually, the NBA introduces new players, known as 'rookies'. Many of these rookies come from college basketball teams. Observers and experts often speculate on which college players will make it to the NBA based on their college performance.

a. Business Use Cases

This study aims to use a machine learning model to predict if a college basketball player will be drafted into the NBA, focusing only on their game statistics from the college season. The benefits of this model include:

- Improved Player Scouting: NBA teams can identify which college players might be potential candidates for the NBA draft.
- Draft Planning: The model can help teams prepare for the draft by highlighting players with a high likelihood of being drafted.
- Rating Verification: Teams can compare the model's predictions with their own assessments to ensure consistency.


This research offers a data-driven approach to help NBA teams make informed decisions about potential new players. But there exist some challenges and opportunities:

Challenges:

1. Numerous Players: Manually assessing every aspiring college player for the NBA is daunting.
2. Bias in Evaluation: Traditional scouting can sometimes be subjective and inconsistent.
3. Limited Scouting Resources: Teams need to ensure they focus on the most promising players.
4. Different Playing Styles: College basketball dynamics can differ from the NBA, complicating evaluations.

Opportunities:

1. Data Availability: College basketball offers a wealth of statistics for analysis.

- 
2. Efficient Scouting: A model can guide teams on which players to watch closely.
 3. Objective Insights: Machine learning reduces human bias, offering more consistent predictions.
 4. Draft Edge: Teams can gain an advantage in the draft by using data-driven insights.

The project aims to address scouting challenges by harnessing the opportunities that data analytics provides. Hence machine learning algorithms are particularly relevant in this context because they can analyze large amounts of data quickly, recognize patterns, and make predictions that might not be immediately apparent to human evaluators. By processing historical data of players who successfully transitioned from college to the NBA, the model can identify crucial indicators of success. This data-driven strategy not only improves on traditional scouting techniques but also introduces a previously impossible level of objectivity and effectiveness.

b. Key Objectives

To develop a machine learning model capable of predicting the likelihood of a college basketball player being drafted into the NBA based on their season statistics. This tool offers NBA teams a data-backed way to choose players, adding to what scouts already do. It helps teams focus on top players and confirms their own evaluations.

Many groups are keen on this project. NBA teams want to pick top players. Scouts need solid data. Players and their coaches want feedback, and companies look for future NBA stars to sponsor. The project uses the machine learning model to look at a lot of data from college basketball games. This helps NBA teams plan for the draft. Scouts and experts get an extra tool to help them decide how good players are. Players and coaches get feedback on what to work on. And companies can choose which players they might want to sponsor.

To sum up, this project uses a lot of basketball data to help different people make better decisions about NBA players.

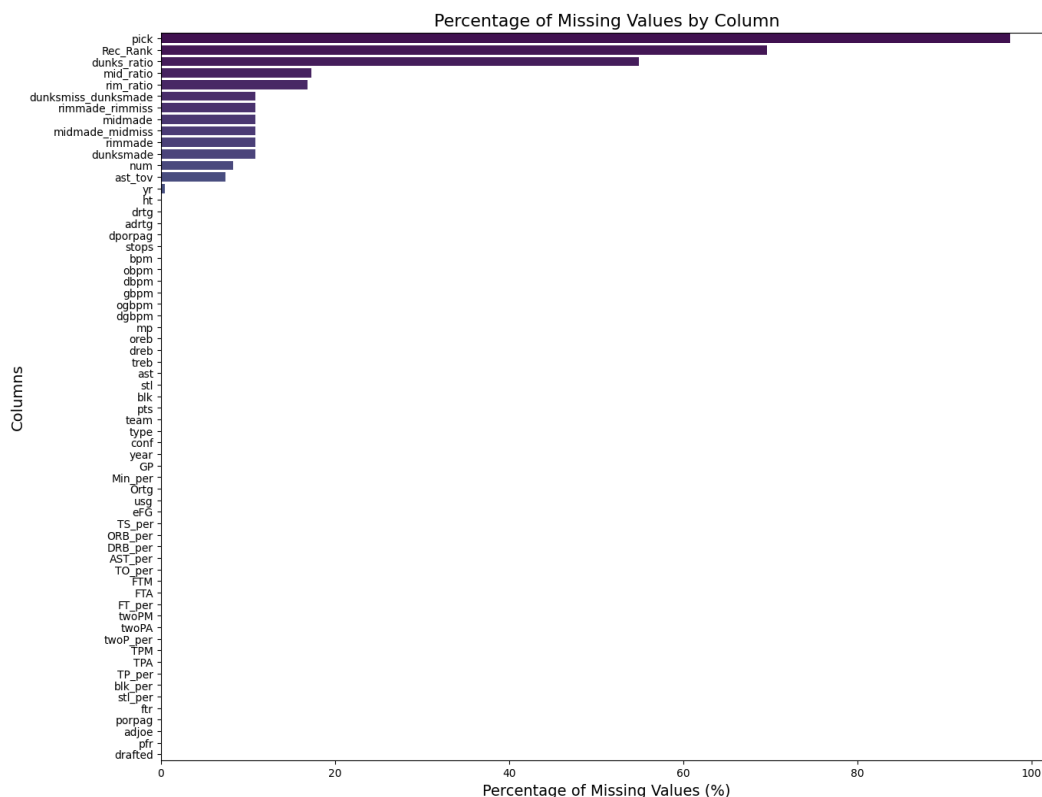
3. Data Understanding

We download the dataset from Kaggle. The dataset appears to be related to basketball players, possibly from college basketball or another league. The dataset used for the project consists of 56,091 entries and 64 columns, detailing various statistics and attributes related to basketball players. The last column, drafted, is a binary variable indicating whether the player was drafted or not. This could be the target variable if the project's goal is to predict the draft status of players. By using function `info()` we can see there are six categorical features in this dataset.

In addition, this data has following limitations:

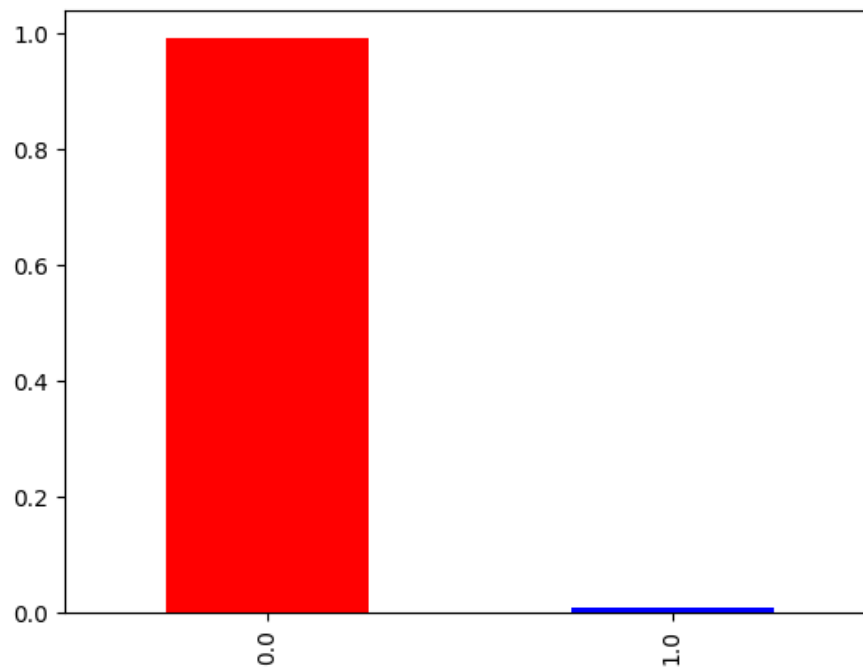
- **Missing Data:** Several columns have missing data. According to figure 1, Some columns, like `Rec_Rank` and `pick`, have more than 60% missing values. Handling these missing values will be crucial.

figure 1. Percentage of missing value of variables



- **Imbalance:** The drafted column is heavily imbalanced, with only about 0.96% of players being drafted. This could pose challenges for modeling, as most machine learning algorithms perform best when the number of samples in each class is roughly equal.

figure 2. distribution of target value



- Potential Bias: Without knowledge of the data collection process, there might be potential biases in the dataset. For instance, if the data is only from a specific region or excludes certain types of players, the findings might not be generalizable.
- Lack of Contextual Information: Some columns might require domain-specific knowledge to fully understand, and additional contextual information or metadata could be beneficial.

4. Data Preparation

During this phase, the data was preprocessed to ensure that it was clean, consistent, and suitable for modeling. The target variable “drafted” was defined as y in our all experiments. Each model build has its own data preparation, but the following points are common to all of them:

- We dropped those variables with more than 50% missing values, replaced all the missing value with ‘unknown’ for categorical features. And we imputed missing values in the numeric features with the mean value of the respective feature.
- And We dropped one variable with single values, since it is trivial for training classification model.
- OrdinalEncoder coding was applied to all discrete variables;
- we setting the ‘class_weight’ parameter of classifier to ‘balanced’ to computes class weights based on the number of samples in each class, helping to give more importance to the minority class during the training process.

5. Modeling

LightGBM is a fast and efficient gradient boosting framework, ideal for large datasets. It handles categorical features, supports parallel processing, and has built-in overfitting prevention. For optimization, we employed grid search with 5-fold cross-validation.

While LightGBM is efficient, it can sometimes overfit on noisy data. In such cases, ensemble methods like stacking can be beneficial. Stacking combines different models, like LightGBM and Random Forest, leveraging their individual strengths for better performance. For instance, while LightGBM may overfit, Random Forest's inherent resistance to overfitting can provide balance.

In the stacking process, initial predictions from base models serve as input to a meta-model, often logistic regression, which effectively combines these predictions, offering insights into each base model's importance (Zhou, 2012).

a. Approach 1: LightGBM without feature engineering

In this experiment we just started with a simple LightGBM model with Grid Search to establish a baseline. We did not do some specific feature engineering except dropping the variables with more than 50% missing ratio. We tuned the following hyperparameters for LightGBM model:

- **learning_rate**: It represents the learning rate or step size used during the gradient boosting process. Smaller values result in slower convergence but potentially better performance, while larger values can lead to faster convergence but possibly worse performance. The grid includes three values: 0.01, 0.1, and 1.
- **n_estimators**: This parameter specifies the number of boosting rounds or trees in the ensemble. More trees can result in better performance but also increase the risk of overfitting. The grid includes five values: 20, 50, 100, 200, and 500.
- **num_leaves**: It defines the maximum number of leaves in each tree. A higher number of leaves can model more complex relationships but may also lead to overfitting. The grid includes three values: 31, 62, and 93.
- **reg_alpha**: This parameter adds L1 regularization to the model, which can help prevent overfitting by encouraging sparsity in the learned weights. The grid includes four values: 0.0, 0.1, 0.5, and 1.0.
- **reg_lambda**: similar to **reg_alpha**, this parameter adds L2 regularization to the model. L2 regularization discourages large weights in the model, making it less likely to overfit. The grid includes four values: 0.0, 0.1, 0.5, and 1.0.

For future experiments, it may be worth exploring some other feature engineering (e.g., Polynomial features, which creating polynomial combinations of the original features) and feature selection to achieve higher performance.

b. Approach 2: LightGBM with feature engineering

In the second experiment we used same preprocessor as previous experiment 1, but we explored feature engineering works and feature selection. Firstly, we added the following features:

- `avg_points_per_game` (pts/GP): This is the average number of points scored by a player per game. It gives an idea about the scoring capability and consistency of a player. A higher value indicates that the player scores more points on average per game.
- `Ortg_diff_from_mean` (`ortg-ortg_mean`): This represents the difference between a player's offensive rating (Ortg) and the mean offensive rating of all players. It helps in understanding how a player's offensive rating compares with the average offensive rating. A positive value indicates that the player's offensive rating is above average, while a negative value indicates below-average performance.
- `usg_Ortg_interaction` (`usg*Ortg`): This captures the combined effect of a player's usage percentage and offensive rating. Players with both high usage and high offensive rating will have a higher value, indicating they are key offensive players for their team.
- `Ortg_square` (`ortg^2`): Squaring the offensive rating can help in capturing non-linear relationships in the data. For example, the impact of very high or very low offensive ratings might be better captured with this squared term.
- `pts_rolling_avg` (with `window=3`): This is the rolling average of points scored by a player over the last 3 games. It provides a smoothed measure of a player's scoring performance over the most recent 3 games. This can help in identifying short-term trends in a player's scoring capability, such as if they're currently in a hot streak or experiencing a slump.

And then we applied feature selector to filter three features with missing ratio higher than 60%, one feature with a single unique value, 12 features with a correlation magnitude greater than 0.95, and keep the remaining 50 features that contribute to cumulative importance of 0.99 by running LightGBM model for 10 times. The rest of the hyperparameters tuning and other steps are the same as in the previous test.

c. Approach 3: stacking (LightGBM + Random Forest)

In the third experiment we kept the same preprocessing or feature engineering as previous experiment, but we employed a Stacking Ensemble technique that combines predictions from two primary models: LightGBM (LGBM) and Random Forest (RF), used a logistic regression model as the meta-learner.

Figure 3. parameters of random search

```

params = {
    'stacking_lgbm_n_estimators': [100, 200, 500, 1000],
    'stacking_lgbm_learning_rate': [0.01, 0.05, 0.1],
    'stacking_lgbm_num_leaves': [31, 62, 93],
    'stacking_lgbm_max_depth': [-1, 10, 20],
    'stacking_lgbm_min_child_samples': [20, 30, 40],
    'stacking_lgbm_reg_alpha': [0, 0.1, 0.5, 1],
    'stacking_lgbm_reg_lambda': [0, 0.1, 0.5, 1],

    'stacking_rf_n_estimators': [100, 200, 500, 1000],
    'stacking_rf_max_depth': [None, 10, 20, 30],
    'stacking_rf_min_samples_split': [2, 5, 10],
    'stacking_rf_min_samples_leaf': [1, 2, 4],

    'meta_C': [0.001, 0.01, 0.1, 1, 10],
    'meta_penalty': ['l1', 'l2'],
    'meta_solver': ['liblinear', 'lbfgs']
}

```

For hyperparameter tuning, instead of an exhaustive grid search, we utilized Random Search tuned the parameters of LightGBM, Random Forest and meta-learner. This method samples a fixed number of parameter settings from the specified distributions. We're aiming for 10 iterations, assessing model performance using the AUC score, and employing a 5-fold cross-validation strategy.

6. Evaluation

a. Evaluation Metrics

The project uses the AUC metric to evaluate the models. AUC measures a model's ability to distinguish between positive and negative outcomes. A higher AUC indicates better model performance. For scouting, where accurate predictions on player success are vital, using AUC ensures the model's predictions are reliable and effective.

b. Results and Analysis

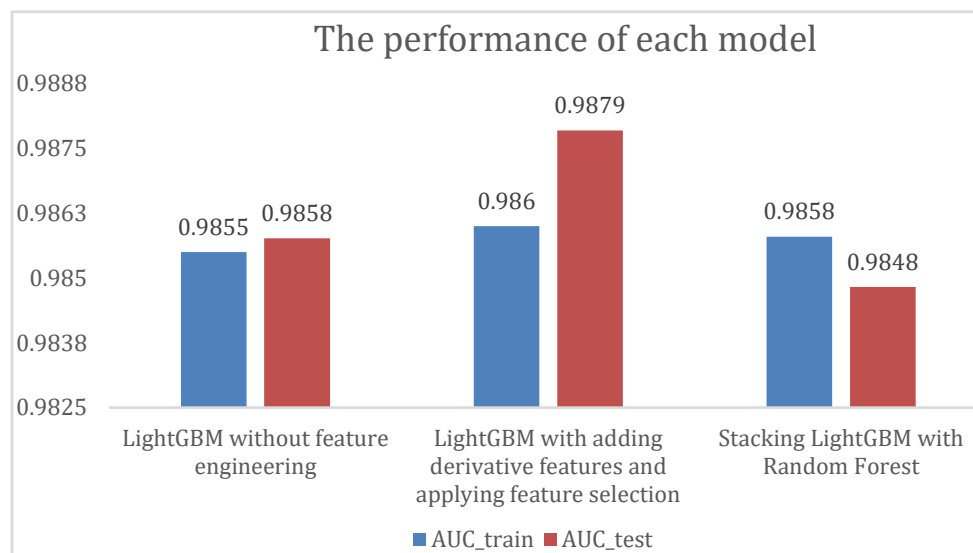
The first model performs almost equally well on both the training and testing datasets, suggesting that it's neither overfitting nor underfitting.

The second model, after feature engineering, shows a slight improvement in both training and testing AUC compared to the model without feature engineering. The testing AUC is also slightly

higher than the training AUC, which is a good sign. This suggests that the feature engineering step was beneficial.

The stacked model has a slightly lower testing AUC compared to the individual LightGBM models. This suggests that stacking did not provide a benefit in this scenario, at least in terms of AUC.

Figure 4. Model evaluation

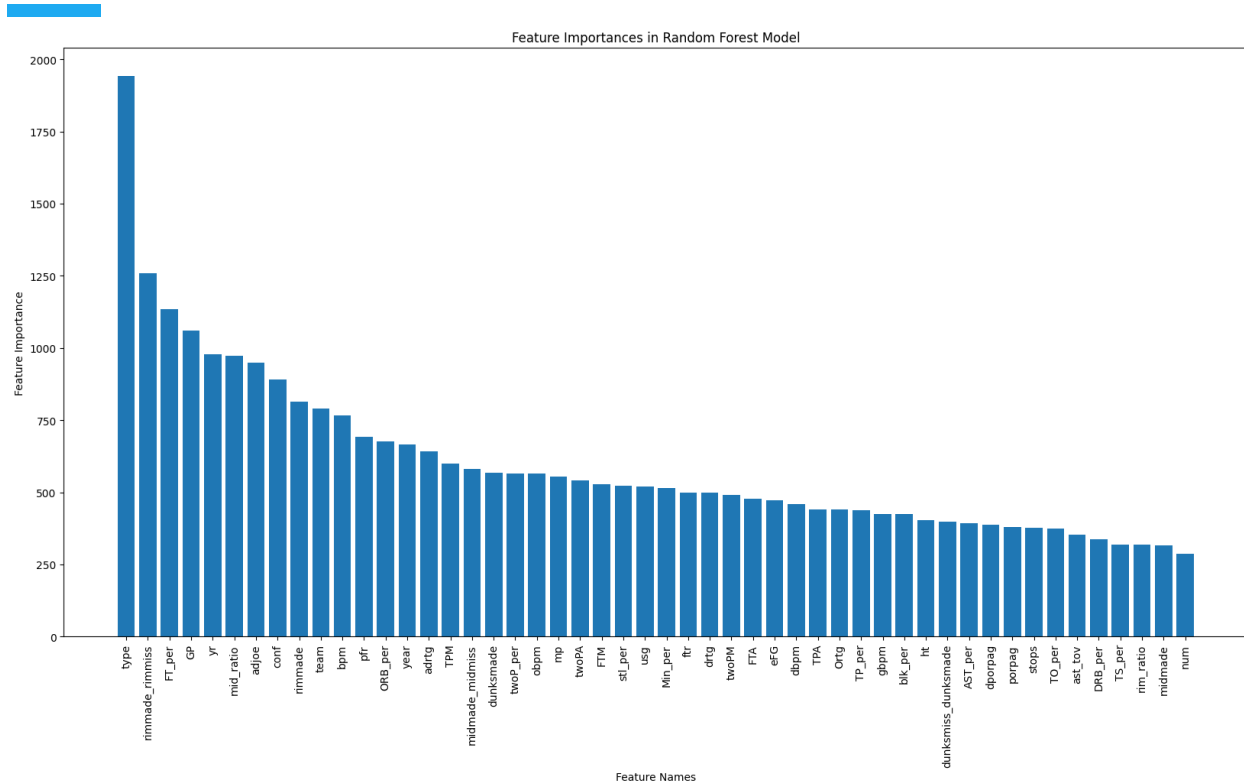


Based on these results, the LightGBM with added derivative features and applied feature selection appears to be the best performing model as it has the highest testing AUC of 0.98785. If the goal is to maximize the AUC score on the testing dataset, this model should be preferred. However, other considerations such as model interpretability, computational costs, and other evaluation metrics might influence the final decision.

c. Business Impact and Benefits

Figure 3 illustrates that the best results suggest that type (1942), rimmd_rimmiss (1260), FT_per (1134), GP (1060) are significant factors influencing NBA draft decisions, The top features in this dataset focus on game efficiency (like shooting close to the basket and overall offensive efficiency), experience (games played and possibly years of experience), and game situations or categories (type). These features provide a comprehensive understanding of a player's or team's performance and their ability to succeed in various game situations.

Figure 5. feature importance



Our model uses stats like rimmade_rimmiss, FT_per, and GP to help NBA teams pick college players. Teams can now choose players more wisely for drafts. Coaches can adjust training based on what the model shows, helping players get better in specific areas. Also, scouts can save time by looking at players who match the model's key points. Overall, our model makes NBA decisions clearer and can save time and money.

d. Data Privacy and Ethical Concerns

Utilizing extensive player performance data, the project aims to modernize scouting practices, offering a more data-driven approach. While this promises objective and efficient assessments, it brings forth data privacy and ethical concerns.

If players weren't aware their data would be analyzed in this manner, consent becomes a pressing issue. While individual data might not be sensitive, its collective use must be ethically sound to protect players' futures.

In addition, the historical data might contain biases. Even with objective evaluations from the model, biased data can lead to biased outcomes, risking unfair evaluations. Moreover, for the

sake of players and other involved parties, understanding the model's workings is essential due to its profound impact on players' careers.

To mitigate these concerns:

- Use data in a way that maintains player anonymity.
- Continually check the model for biases to ensure fairness.
- Clearly explain to players how their data is used and how the model functions.
- Allow feedback from players to refine the model further.

By addressing these aspects, the project balances innovative scouting with ethical considerations.

7. Deployment

Deploying a machine learning model involves several steps and tools, including version control (Git), containerization (Docker), web frameworks (Flask), and API design. We plan to do following work if we desired to deploy it into production:

a. Process of deployment

- Version Control: It's important to use Git for version control of our code and models. We should create a Git repository and commit and push every significant change to our code or models. This helps keep track of the project's history and facilitates collaboration among multiple people.
- Model Serialization: The trained model needs to be serialized (for instance, using pickle or joblib) and saved to disk so it can be used during deployment.
- Building the API: Build an API using Flask or another web framework that can receive requests from clients, process them, and return results. For instance, we could design a

POST request that takes a JSON with the model's input data and returns the model's predictions.

- **Dockerize the Application:** Create a Dockerfile that contains all the dependencies and commands needed to run our application. For instance, we might need to install Python, Flask, and any Python libraries needed to load and run our model. We can then build and run a Docker container that contains our entire application. Using Docker helps ensure consistency and portability across different environments.
- **Deployment:** Finally, we need to deploy Docker container to a server. This can be done in various ways, such as deploying directly to a cloud server, or using Kubernetes for container orchestration.
- **Documentation:** Prepare thorough documentation detailing the model's development process, data preprocessing steps, hyperparameters, performance metrics, and any limitations or assumptions. This will help ensure a smooth handover to other teams and enable future improvements or troubleshooting.

b. Challenges

As data evolves, models can lose their accuracy, and ensuring they make predictions quickly becomes a challenge, especially when rapid responses are essential. The financial burden of deploying and maintaining models can grow, especially when scaled up, and simultaneously, it's crucial to manage updates to these models and ensure they operate without biases.

c. Recommendations

By integrating tools like Jenkins, the process of deploying becomes more streamlined, and with platforms such as AWS Sagemaker, overseeing and scaling models becomes more manageable. It's beneficial to remain abreast of modern techniques, incorporate feedback from users for better outcomes, and consistently document procedures to ensure clarity in operations.

8. Conclusion

The project successfully developed a machine learning model that predicts the likelihood of college basketball players being drafted into the NBA. We found some significant factors influencing NBA draft decisions, such as `rimmde_rimmiss`, `FT_per`, and `GP`, were identified. These metrics provide insights into a player's game efficiency, experience, and adaptability in various game situations. Consequently, the model's data-driven approach promises to make scouting more objective, reduce biases, and streamline the draft process.

While our current model makes adequate predictions, there is evident potential for improvement. Firstly, by expanding our dataset to include player injuries and psychological assessments, we might dramatically improve its predictive skills. Moreover, it's vital to periodically update the model with fresh basketball data to ensure its relevance. Given the rising concerns about data privacy, it's imperative that we ensure everyone is clear about how their data is being utilized. Consequently, holding regular discussions with players, coaches, and other stakeholders is essential. These interactions can not only foster trust but also gather invaluable feedback for model improvement. Furthermore, by merging our model with other algorithms, there's potential to achieve even more precise predictions.

References

OpenAI. (2021). ChatGPT based on GPT-4 architecture. <https://www.openai.com/>

Zhou, Z. H. (2012). *Ensemble methods: foundations and algorithms*. CRC press.