Predicting NBA Draft Outcomes

Nandha Gopikannan, Dillon New, Richie Fleming
Luddy School of Informatics, Computing, and Engineering Department of Computer Science
Indiana University
Bloomington, Indiana
nagopi@iu.edu, dinew@iu.edu, riflemin@iu.edu

***Abstract*—This project focuses on leveraging machine learning techniques to predict the outcomes of the NBA draft, aiming to provide valuable insights for teams and analysts in making informed decisions during the player selection process. The study utilizes a comprehensive dataset encompassing various player attributes, collegiate performance metrics, and other relevant factors.**

The project's findings contribute to the growing field of sports analytics and provide NBA teams with a data-driven tool to enhance their decision-making process during the draft. Additionally, the Python-based approach and open-source nature of the code encourage further exploration and collaboration within the sports analytics community.

I. Introduction

In the dynamic and highly competitive realm of professional basketball, the NBA draft stands as a pivotal event where teams strategically select players who hold the potential to shape the future of their franchises. The task of predicting the success of prospective NBA players has long been a challenge, given the multifaceted nature of player evaluation and the myriad factors influencing athletic performance. This project addresses this challenge by employing machine learning techniques and Python programming to develop a predictive model for NBA draft outcomes.

The significance of accurate player assessment cannot be overstated, as teams invest substantial resources in identifying talent that aligns with their strategic goals and team dynamics. Traditional scouting methods rely on subjective observations and expert opinions, but advancements in data science present an opportunity to complement and enhance these approaches. By leveraging a comprehensive dataset encompassing player attributes, collegiate performance metrics, and various other relevant features, this project aims to provide a data-driven perspective on player potential.

Python, a versatile and widely adopted programming language, serves as the foundation for the implementation of machine learning algorithms. The exploration of diverse algorithms, coupled with rigorous evaluation techniques, contributes to the creation of a robust and reliable framework for predicting NBA draft outcomes.

This endeavor not only seeks to provide teams with a valuable decision support tool but also contributes to the evolving field of sports analytics. The insights gained from this project illuminate the role of specific attributes and performance metrics in shaping a player's draft position, thereby adding a quantitative dimension to the traditionally qualitative process of player evaluation. The transparency of the Python-based approach fosters collaboration and encourages further exploration within the sports analytics community, advancing our collective understanding of talent assessment in the world of professional basketball.

II. Data Collection

A. How and what data we needed

The foundation of this project rested upon the acquisition and preparation of a comprehensive dataset, vital for training and evaluating the machine learning models. The data gathering process involved utilizing the ESPN archive as a primary source. The raw data was initially retrieved and organized within an Excel spreadsheet for subsequent analysis. This dataset encompassed a diverse range of attributes, including player statistics, collegiate performance metrics, and various other factors deemed relevant for predicting NBA draft outcomes.

Upon importing the data into a Google Sheet, an essential step in the data cleaning process ensued. A notable challenge arose from the inclusion of college initials appended to the end of player names. To address this, a Sheets formula was employed to systematically remove capital letters from the end of each player's name until lowercase letters were encountered. This meticulous cleaning process ensured consistency in the representation of player names, eliminating potential discrepancies that could impact the accuracy of subsequent analyses.

In addition to player performance metrics, obtaining information on drafted players was crucial for the training and evaluation of the machine learning models. To achieve this, a Kaggle database was utilized, providing a reliable source of information on players who had transitioned from collegiate to professional basketball through the NBA draft. This supplementary data facilitated the creation of a labeled dataset, distinguishing between drafted and undrafted players.

The integration of data from ESPN and Kaggle allowed for a comprehensive exploration of player attributes and performance metrics in the context of NBA draft outcomes. This meticulous approach to data collection and cleaning laid the groundwork for subsequent feature engineering and machine learning model development, ensuring the accuracy and reliability of insights derived from the predictive models.

III. Data Cleaning

After organizing the dataset and integrating player statistics with pertinent draft data, we found it necessary to further process the data to make it suitable for analysis in Python. We opted for the Pandas library for data cleaning and management due to its robust and efficient functions for handling data. Initially, we combined the data we collected from multiple seasons to establish a comprehensive training set. Subsequently, we carried out various cleaning operations. This encompassed mapping non-numeric values, such as a player's position and their draft status, into numeric or binary values. Following this, we eliminated any data points that contained NaN values to ensure the integrity of each data point under test and to prevent potential issues during analysis. This rigorous data processing and cleaning methodology enabled us to prepare a clean, reliable dataset for our subsequent analysis.

```
datatokeep = ['NAME','POS','GP','MIN','PTS','FGM','FGA','FG%',
              '3PM','3PA','3P%','FTM','FTA','FT%','REB','AST',
              'Drafted?','Draft Pick','Draft Year']
```

Fig. 1. Player Data Attributes

IV. Model Training and GUI Development

For all of our models, we opted to use the Python scikit-learn library due to its efficient data processing capabilities and built-in functions for a variety of models. Our initial approach to predicting whether a player would be drafted was to employ a K-Nearest Neighbor (KNN) classifier.

```
#K-Nearest Neighbors
def knn(data, new_data):
    feature_cols = ['GP','MIN','PTS','FGM','FGA','FG%','3PM','3PA','3P%','FTM','FTA','FT%','REB','AST']
    label_col = 'Drafted?'
    knn = KNeighborsClassifier()
    X = data[feature_cols]
    y = data[label_col]
    knn.fit(X, y)
    new_data = new_data[feature_cols]
    prediction = knn.predict(new_data)
    return prediction
```

Fig. 2. Implementation of the K-Nearest Neighbor Classifier

However, our initial testing revealed that the KNN classifier was not as effective as we had anticipated. To address these issues, we decided to implement both a Naive Bayes and a Logistic Regression classifier to ensure we had a robust set of models. The Naive Bayes classifier, despite its simplicity and the assumption of feature independence, can be particularly effective for high-dimensional datasets. It's also computationally efficient, making it a good choice for large datasets. On the other hand, the Logistic Regression model is a powerful and flexible classifier that can model complex, non-linear relationships between features.This decision to implement multiple models was also influenced by the nature of the data we had chosen to use. By using a diverse set of models, we aimed to capture different aspects of the data and improve the overall performance of our predictive system.

In addition to model development, we also created a basic Graphical User Interface (GUI) that accepts a link from Sports Reference for College Basketball. The GUI then outputs a prediction on whether the player in the link would be drafted based on their statistics and our model. The backend of the GUI utilized the Pandas 'readhtml' function to scrape data from the site, and performed similar processing to our earlier data cleaning to make the data suitable for training on our model. The frontend of the GUI, built using the Python Tk library, accepts the link as a text input and outputs a statement on whether or not the player would be drafted if they declared for the NBA draft.

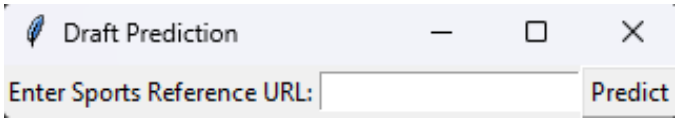

Fig. 3. GUI Link Input Page

V. Testing and Conclusions

To evaluate the effectiveness of our models, we tested them against data from the 2022-2023 NCAA college season and the 2023 NBA draft. Contrary to our expectations, the models were largely ineffective. The K-Nearest Neighbors model, despite being the most effective among our models, yielded quite random results during individual player testing, with no clear reasoning as to why a player would be drafted or not. When conducting individual tests on the Naive Bayes and Logistic Regression classifiers, we found that although their overall predictions were slightly worse, they were more accurate in predicting that players with conventionally good statistics would be drafted, while players with conventionally worse statistics would not be drafted.

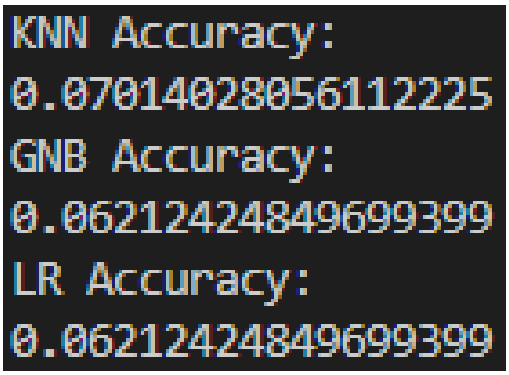

Fig. 4. Testing Results for Each Model

We identified several reasons for the ineffectiveness of these models:

A. Players are often Drafted based on Potential

The NBA draft is often a gamble for teams, where they draft players that they believe will develop into NBA stars. Therefore, players who may not have the best stats often get drafted because of their potential to make an impact on an NBA roster. This impacts our model by having players who aren't the best on their college teams being drafted, affecting the effectiveness of our model because "potential" is not a trackable statistic.

B. Players with Good Stats do not declare for the draft

The best players on college teams are often upperclassmen, who do not intend to declare for the NBA draft. These players skew model prediction results because despite having good stats, they are classified as undrafted. Intent to declare is not a recorded statistic, and cannot be determined simply by looking at a player's statistics. As a result, this fact often hampers the model's effectiveness by causing good players to be classified as drafted when they were actually undrafted because they did not declare.

C. A Large Portion of College Players end up going undrafted

In our training dataset, we had over 16,000 recorded players and entries, and only about 570 of those players were drafted. Because of this overwhelming number of undrafted players, the data and predictions would often skew in their favor regardless of the statistics of drafted players. As a result, classifiers like the K-Nearest Neighbors were ineffective due to this fact specifically, where the nearest neighbor was mostly an undrafted player because of the sheer number of undrafted player data points recorded.

Based on our testing and exploration of our data, we conclude that qualitative factors often play a more significant role in the NBA draft than quantitative factors. Elements such as a player's potential, their ability to interact effectively with teammates and coaches, and other intangible factors can all influence the draft decision and often have a more substantial impact than the raw statistics.

The models we developed are proficient at detecting whether a college player's performance metrics align with those typically seen in drafted players. However, they fall short in predicting the "intangibles" - aspects like a player's potential for improvement, leadership qualities, work ethic, and fit with a team's style of play or organizational culture. These factors, which are often assessed through scouting and personal interviews, are not captured in our dataset and hence, our model cannot predict them.

CONTRIBUTIONS

All group members contributed evenly to the success of the project. Dillon and Richie both worked on collecting the data, as well as sorting and cleaning it for use in the models. Nandha worked on the development of the models, as well as the GUI and the tests. Everyone contributed evenly to the presentation as well as writing the report.