**Using Machine Learning to Predict the NBA’s Most Improved Player Award**

**Tyler Hoyle**

**Department of Data Science, University of Wisconsin – Green Bay**

**DS785: Capstone**

**April 26, 2024**

**Abstract**

Each year the National Basketball Association awards players based on their performance throughout the 82-game regular season. The most improved player award is designed to honor an up-and-coming player who has made a dramatic improvement. What makes this award especially interesting is the subjectivity of what constitutes a dramatic improvement. Other awards, like the most valuable player, are quite predictable, often having the same players as finalists year after year. The most improved player award has never had the same player win twice, and the winner is often a player who was not seriously considered at the beginning of the season.

This project explored the viability of machine learning models to consistently predict the winner of the most improved player award for the 1987-2023 seasons. The viability of using the results from the exploration above as part of a sports betting strategy was also explored.

Using data from basketball\_reference.com and the NBA API, six models were trained and tested on multiple groupings of data and evaluated using traditional regression metrics, and custom metrics, including the number of seasons where the model correctly predicted the winner. The Random Forest and XGBoost models performed the best, correctly predicting the winner for 28 of 36 (78%) seasons, including the most recent 9 seasons in a row. Despite good prediction performance, the use of the outputs in a betting strategy was found to not be viable, offering a modest return of 7.4% for the 2020-23 seasons. This is due to unfavorable betting odds offered by sportsbooks at the end of each season because of the nature of the award.

**Table of Contents**

List of Tables………………………………………………………………………………………5

List of Figures……………………………………………………………………………………..6

Chapter 1: Introduction……………………………………………………………………………7

Background………………………………………………………………………………..7

Research Motivation………………………………………………………………………8

Scope of Project……………………………………………………………………………………9

Project Objectives………………………………………………………………………………...11

Conclusion………………………………………………………………………………………..12

Chapter 2: Literature Review……………………………………………………………………………...12

Machine Learning in Sports………………………………………………………………………13

Model Predictions as a Betting Strategy………………………………………………………….17

Conclusion………………………………………………………………………………………..19

Chapter 3: Methodology…………………………………………………………………………………..19

Methodology Overview…………………………………………………………………………..20

Data Collection Details…………………………………………………………………………...24

Data Preparation Details………………………………………………………………………….25

Model Selection and Evaluation………………………………………………………………….33

Conclusion………………………………………………………………………………………..35

Chapter 4: Results………………………………………………………………………………………....36

Model Results Before Hyperparameter Tuning…………………………………………………..36

Model Results After Hyperparameter Tuning…………………………………………………….39

Ensemble Results…………………………………………………………………………………42

Sports Betting Viability…………………………………………………………………………...42

Conclusion………………………………………………………………………………………..43

Chapter 5: Discussion……………………………………………………………………………………..43

Findings…………………………………………………………………………………………..44

Objectives Conclusions…………………………………………………………………………..45

Implications for Professional Use………………………………………………………………...45

Limitations………………………………………………………………………………………..46

Suggestions for Future Research…………………………………………………………………46

Potential Accuracy Improvements……………………………………………………….46

Potential Betting Improvements…………………………………………………………47

Conclusion………………………………………………………………………………………………...47

References…………………………………………………………………………………………………49

Appendix…………………………………………………………………………………………………..54

Code to Gather Data from NBA API……………………………………………………………..55

Code Used for EDA………………………………………………………………………………56

Code Used for Modeling………………………………………………………………………….60

Modeling Info…………………………………………………………………………………….65

**List of Tables**

Table 1: 3 Sigma Cutoff for Each Statistic Used to Filter the Dataset…………………………...32

Table 2: Comparison of Models Results for 1987-2019 Seasons Before Hyperparameter Tuning……………………………………………………………………………………………38

Table 3: Comparison of Models Results for 2020-2023 Seasons Before Hyperparameter Tuning............................................................................................................................................40

Table 4: Comparison of Models Results for 1987-2019 Seasons After Hyperparameter Tuning……………………………………………………………………………………………41

Table 5: Comparison of Models Results for 2020-2023 Seasons After Hyperparameter Tuning……………………………………………………………………………………………42

Table 6: Ensemble Results for 1987-2019 & 2020-2023 Seasons……………………………….43

**List of Figures**

Figure 1: Games Played Distribution of Players who Received MIP votes vs Players with No Votes (Complete Dataset)………………………………………………………………………..28

Figure 2: Points Per Game Difference of Players who Received MIP Votes vs Players with No Votes...……………………………………………………………………………………………30

Figure 3: Minutes Per Game Differential of Players who Received MIP Votes vs Players with No Votes with Cutoff………...............................................................................................................31

Figure 4: KDE Plot of Award Share of the Filtered Dataset vs the Post SMOTE Dataset …...…33

**Chapter 1: Introduction**

# **Background**

Each year the National Basketball Association (NBA) gives awards to players based on their performance throughout the 82-game regular season. Each award is selected by a global panel of 100 sportswriters and broadcasters from companies like Fox Sports, ESPN, NBA.com, and many more. The most improved player (MIP) award, also known as the George Mikan Trophy, is designed to honor an up-and-coming player who has made a dramatic improvement from the previous season or seasons (NBA.com). At the conclusion of the season, each panelist casts their first, second, and third place votes. First-place votes are worth five points; second-place votes are worth three points, and third-place votes are worth one point. The player with the highest point total wins the award. For example, the winner of the 2022-23 MIP award was Lauri Markkanen who received 69 first-place votes, 27 second-place votes, and 4 third-place votes, resulting in 430 total points.

What makes the MIP award interesting is the subjectivity of what constitutes a dramatic improvement, making it much different from other awards. For example, the NBA’s most valuable player (MVP) award has many of the same candidates year after year and it is very rare for a candidate to win the award without being considered at the beginning of the season. On the other hand, the MIP award has never been won twice by the same player and unlike the MVP, it can go to very different types of players. A good example of this is the 2021-22 award. This award went to Ja Morant, a second overall pick who was living up to his draft expectations and was far and away the best player on his team. This same year Jordan Poole received a similar number of votes, yet only started 51 of the 82 games that season and was considered by some to not be one of the top three players on his team.

Sports betting makes understanding these nuances much more important. Sports betting is the act of placing monetary wagers on the results of sports events. There are many types of bets a person can make when it comes to sports betting. While all types of bets can play an important role in an overall betting strategy, for the purpose of this research, the focus will only be on moneyline betting. A moneyline bet refers to a wager on if a team or player will win or lose a game or contest. In the context of this report, this refers to who will win or lose the most improved player award. Each bet is offered by a sports book company and bettors can wager their money for a certain return based on the odds that the sports book offers. A common example of sports book odds is a game between two even teams, both having a 50% chance of winning. In this case, the odds would likely be -110 for each team, implying the bettor has to risk $110 to earn $100.

# **Research Motivation**

According to Statista, sports betting revenue in the United States has increased from approximately $430 million in 2018 to over $7.5 billion in 2022 (2023). In 2021 alone, sports betting on a weekly basis doubled in the United States (Etuk et all, 2022). With this recent boom in popularity, an opportunity for sports betting companies and bettors has surfaced. Record cash flows, along with additional availability to make bets in real-time, make it increasingly important for sports books to maintain favorable odds. Competition between sports books also opens up opportunities for both sports books and bettors. Increased competition gives companies an incentive to offer promotions to attract new customers, which bettors can take advantage of. In both cases, it is beneficial for the party to have a good understanding of what may happen. For companies, this type of work can be used to reduce risk while offering promotions that are capable of bringing in new customers, while for bettors this type of work could help them better utilize the promotions that are available.

This use of machine learning for prediction has been studied and applied to many types of sporting events in the academics space, like the NBA most valuable player award as seen in the work by Chapman (2023). Sports books have also been calculating odds and accepting bets on these types of events for quite some time. However, how this is done is not publicly available knowledge and differs from company to company. For example, as of February 22nd, 2024 according the Vegasinsider, a bettor could bet on Tyrese Maxey to win the most improved player award through Fanduel at -200 odds or they could make the same bet through MGM at -105 odds. This topic warrants additional investigation due to the lack of general knowledge of how to best calculate odds and the absence of current academic work on this specific award.

The main motivation of this project was to explore the viability of machine learning models in predicting the outcome of the MIP award voting process. Another motivation was to explore if the process could be streamlined in a way where it makes sense to use model outputs as part of a betting or odds creation processes.

# **Scope of Project**

With the motivations above in mind, this project explored the ability of different types of machine learning models to predict the outcome of the MIP voting process with the historical context of the award used as input. The scope of this project included the assessment of:

* The history of MIP award and how it affects who can win.
* The exploration of the available data and engineering of necessary features.
* The model creation and evaluation process.
* Interpretation of model results.
* Viability of use of the models and potential future research.

The history of the award was examined by looking at the winner for each year and that player’s statistics for their career up to that point. One example of a finding that came as a result of this research is that no player has ever won the MIP after they had been selected to the all-star team. This fact, and others like it, helped the author better understand that the award is meant to recognize a certain type of player, above average players making the jump to the top echelon. Players that were already considered to be in the top echelon are recognized with other awards, like all-NBA awards. This research helped to narrow the focus to these players within the data and emphasize their characteristics within the models to increase overall performance.

The exploration of data was done with care and close attention to detail. The data selected was gathered from basketball\_reference.com and the NBA Api, both sources having confirmed to use official NBA data. This was of the utmost importance because the quality and accuracy of the data could drastically impact model performance. Throughout this process many features were engineered to better address the problem. Some examples include features showing the difference in statistics between the current season and the previous season, like the difference in minutes per game or points per game. Since the award is about the player who has made the biggest improvements between the previous season and the current season, it is intuitive to include these kinds of features.

The modeling process began with the selection of appropriate types of models. This was largely driven by previous academic work. The models that were selected had either performed well in previous literature or processed data differently than those models that had performed well in previous literature. Those that process differently were also selected due to previous academic work regarding creating ensembles that perform well in betting environments. The creation process involved training and testing each model, as well as a weighted average ensemble, on two different subsets of data. Each model was then evaluated using typical regression metrics, as well as custom metrics, that better determine effectiveness in the context of this research.

Using the results from both subsets of data, each metric was investigated and used to determine the effectiveness of the models. The results of each model were compared with one another across all metrics to validate the results and determine effectiveness.

Lastly, the viability of the model in a betting environment was explored, using the MIP betting odds for the current season at the time of this writing. Potential future research was also explored to provide ideas of ways to build upon this research.

While a brief description of the scope was offered here, each section of the scope is also discussed in more detail in later sections in the report.

# **Project Objectives**

The primary objective of this project was to construct and evaluate models to predict the winner of the MIP award using multiple methods. The paper discusses the specifics of the methods applied to accomplish this objective in detail in Chapter 3. The goal of this project was to create a model or ensemble of models that could accurately predict the player who received the most votes for each season from 1987-2019, while remaining accurate during the 2020-2023 seasons. The models ranged from relatively simple like the support vector machine to very complex models, including Gradient Boosting models and a Neural Network. Each model was also evaluated using custom metrics to ensure the models were directly addressing the problem at hand. The most important of these metrics was the ‘Correct Predicted Winner’ metric. This metric showed how many seasons the model had correctly predicted the winner of the award, meaning the higher the number, the more seasons the model was correct with its prediction. Comparing each modeling method to the others using custom and traditional regression metrics gave the author confidence that the chosen strategies are fit to accomplish the objectives of this project.

The secondary object of this project was to explore if the model output produced in the process described above could be implemented into a sports betting strategy, like that developed by Matej, et all (2021). This was done by evaluating the potential payouts received if bets had been made based on the model outputs at the end of each season from 2020-2023, using the available betting odds for the current MIP award at the time of this writing.

## **Conclusion**

Chapter 1 offered an overview of the project, including the background, motivation for the research, and scope. Additionally, the objectives of the study were described. The remainder of this report dives into the details of the available literature, methods, and results of this case study. Chapter 2 reviews the literature regarding predictive modeling in sports, the use of model outputs in betting strategies, and how these relate to the MIP award. Chapter 3 provides details on the methods used to model this problem from data gathering to model evaluation. Chapter 4 discusses the results of the modeling process using custom and traditional metrics. Lastly, Chapter 5 talks through the implications of the research and offers suggestions for potential future research to build upon this project.

## **Chapter 2: Literature Review**

Recently, the availability and popularity of sports betting has seen tremendous growth. This is, in part, due to changes in state laws, which legalized sports betting. At the time of this report, 38 states and Washington DC legally offer sports betting through sportsbooks, according to the American Gaming Association (Interactive Map: Sports Betting in the U.S., n.d.). Sportsbook companies are also offering an increasing number of opportunities to bet through in-play betting. This allows viewers to also make wagers after a game has started, with the odds changing in real-time based on what is happening during the game (Mollenkamp, 2024). The increase in cash flow through sportsbook companies and the rise in available bets has presented an opportunity for sportsbooks, as well as sports bettors, to capitalize on, by improving their ability to predict outcomes. In both cases, strengthening the ability to predict outcomes could give the respective party a competitive edge and improve profitability.

The purpose of this research is to explore the capability of machine learning models to predict sporting event outcomes, in this case, the NBA most improved player award, determine what data is most important in prediction, and to see if these results can lead to positive returns in a sports betting strategy. To inform how this research would be performed, previous works on machine learning in sports, its application to sports betting, and general machine learning best practices were referenced. Google scholar, along with the University’s Library system, were used to gain insight into these topics from academic journal articles, academic research papers, and presentations at relevant conferences. These sources were employed to get a better understanding of the current state of research and best practices to follow throughout the data collection and modeling process.

## **Machine Learning in Sports**

With the availability of structured and unstructured data related to sports continuously increasing, the use of machine learning to predict the outcome of sporting events has been relatively underexplored compared to industries where great strides have been made, like medicine and financial services (Wilkens, 2021). A good example of this is shown in Horvat and Job’s (2020) review of machine learning in sports, where they mentioned having difficulties making comparisons between the available literature due to “the fact that there are practically no papers using the same datasets”. In fields where machine learning practices are more established, heavily explored datasets are often cleaned and used as educational tools for those newly entering the field. The lack of this kind of resource suggests that the topic of sports prediction has seen less interest in comparison. Additional evidence that machine learning in the sports prediction and betting space lags behind many other industries can be seen in Wilkens’ 2021 paper, where they stated that “more traditional statistical approaches still dominate this field” and referred to their research relating to tennis as addressing a “critical research gap”.

After continuing research to see if this subject has since been further explored, it has been concluded that substantial opportunity for investigation was present. Even a quick search in the UWGB library system or Google Scholar showed the difference in the body of research between sports and healthcare. For example, searching for “machine learning in medicine” using the University Library produced over 130,000 results for related peer reviewed research. However, searching for “machine learning in sports” using the same sources resulted in just over 5,000 results for peer reviewed research. The available work spans many different sports, but the majority of studies revolve around predicting outcomes of contests using classification techniques. One example of such a study is the 2023 study by Choi et al. In this study, the authors used Random Forest, Logistic Regression, Linear Support Vector Classifier and Extreme Gradient Boosting models to predict results of soccer matches in the English Premier League. The authors then simulated what betting profits would look like for their models. They found that the Random Forest model had the highest accuracy, but the Logistic Regression model led to the highest returns. A key insight from this study was the impact the sampling technique can have on performance. The authors found that models trained on data selected using balanced sampling performed better than the models trained on data selected using stratified sampling (Choi et al., 2023). The methods behind balanced sampling are further discussed in the Breidt and Chauvet article (2012). While studies like this may not directly relate to the project in this report, they often provided insights that were valuable in the context of this research as shown above. Another example of this can be seen in the Wilkens (2021) study. In this study, Wilkens applies established machine learning techniques to predict future men’s and women’s tennis match results, using historical player and match data, as well as betting odds information. Wilkens found prediction accuracy to not typically surpass 70% and that current rankings along with bookmaker odds provided the majority of the prediction power. A takeaway from this study was that the author found ensembles that combine the outputs from various individual approaches to offer the most promising results (Wilkens, 2021).

While the research in the sports area is relatively limited, works directly related to basketball do exist, and those works influenced how this research was conducted. In their paper reviewing state of the art techniques in predicting the outcome of basketball games, Horvat and Job (2020) summarized the results of nineteen different papers, using many kinds of prediction methods including Neural Networks, SVM, Decision Trees and Random Forests, Logistic Regression, Naïve Bayes, and Gradient Boosting algorithms. They found that Neural Networks, Naïve Bayes, Decision Trees, and Random Forests performed the best throughout the included studies. The key takeaway from this analysis was how critical it is to have a solid understanding of which metrics are relevant for the problem at hand and using that knowledge to carefully select features. The large majority of studies in the review mentioned a difference in performance based on the feature set used. It showed that Neural Networks typically did better with larger feature sets and that traditional machine learning models could improve generalization to new data using smaller feature sets (Horvat & Job, 2020). This was important to keep in mind as research progressed to provide opportunities for improved performance based on model type and features used. In another study that applied machine learning techniques to basketball, Nguyen et al. (2021) used regular season player statistics to predict players’ future performance and popularity. As a proxy for performance, the authors decided to predict each player’s Win Share statistic. To do this they used historic statistics and five different kinds of regression models. For popularity, the authors predicted if a player would be selected to the all-star team. In this case, the authors tested seven different classification algorithms. The authors also tested out deep learning modelling approaches for both future performance and popularity predictions. This study showed promising results with RMSE and MAE scores below 3 when predicting Win Shares, as well as promising classification results with Recall and ROC AUC scores greater than .9. Further details on RMSE and MAE can be found in the Chicco et al article (2021). Two key learnings from this study included0 traditional machine learning techniques outperforming deep learning techniques, and potential techniques to handle imbalanced data, with increased performance when using under-sampling as opposed to over-sampling (Nguyen et al., 2021). Since the dataset and conduct of this research are more similar than the previous studies, these results were given more weight regarding the design of the project represented in this paper.

Research directly related to predicting NBA award winners was also available and had a large impact on how this research was conducted. One such study that was referenced was Chapman’s (2023) work on predicting the NBA MVP. This study used regular season player statistics from 1982 to 2012 to predict the proportion of MVP votes awarded to players. The author trained an independent model for each season and used the combined results to predict the winner. Regression techniques, including Random Forests, XGBoost, and Lightgbm regressors were tested and it was found that the Lightgbm model produced the best result, correctly identifying the award winner 80.65% of the time during training and 100% of the time in testing. A point emphasized in this study was the importance of the Synthetic Minority Over-sampling Technique (SMOTE). This technique introduces new data by taking each minority class sample and creating synthetic examples. This is done by taking the difference between the sample under consideration and its nearest neighbor. The difference is then multiplied by a random number between 0 and 1 and added to the sample under consideration (Chawla et al., 2002). Due to the nature of the award, only very few players receive votes, making the data very unbalanced, which typically makes it harder for models to reach peak performance. The author used SMOTE to address this problem, resulting in a dataset with 50% of players who did not receive a vote and 50% of players who did receive votes, and improving model performance.

## **Model Predictions as a Betting Strategy**

Pursuing the ability to accurately predict uncertain outcomes is an endeavor many would consider worth researching. However, using these predictions for monetary gains can make the topic even more interesting for some. Sports betting opens up this opportunity for those who are interested. To understand what is required to apply model output to real-world betting strategies, Wilkens (2021) study on sports prediction was revisited. In this study, Wilkens explained how model-implied odds are used to decide if a bet would be placed or not, using the following example: “if the model implies a probability of the favorite to win of 80%, bookmaker odds of more than 1/0.8 = 1.25 would be considered worth betting on.” In the context of this research, if the model used for predictions implied a probability of 75% for a certain player to win but bookmaker odds imply a lower probability, the bet would have been considered favorable. The Wilkens study also covered different money management strategies based on bet sizes and how that can affect returns. The strategies covered include a fixed bet size, fixing bet size based on the proportion of current bankroll, fixing bet size based on expected return, using the Kelly criterion (a formulation used to make informed decisions about capital allocation when betting by maximizing the expected value of the log-growth of capital (Jacot et al., 2023)), and using variance optimization. Alongside the context regarding how to implement model output into betting strategies, the Wilkens study also provided insights on prediction and investment performance. Ensembles of models were shown to provide the best prediction as well as return. The author argued that followed logic as the collection of models combined information that differed from one another, which worked especially well because the models processed input data in different manners (Wilkens, 2021). Wilkens used this logic to test placing bets only where multiple models agreed that a bet was favorable and found that when 4 models produced the same result, the returns were the highest.

Articles related to betting results on NBA outcomes were also available and provided meaningful insights. In one study attempting to use machine learning to produce profits betting on NBA outcomes, Hubáček et al. (2019) used logistic regression alongside two types of Neural Networks to gain as much insight as possible from team-level, player-level, and bookmaker statistics. The authors found that incorporating betting odds as an input feature increased the model’s prediction power but actually reduced the ability for use in betting due to correlation with the bookmaker’s predictions. Using additional strategies, the authors found their methods to yield positive returns in experiments with the data from the 2007-2014 NBA seasons. A key takeaway from this study was the importance of decorrelating with bookmaker predictions. If a bettor does not consider this aspect, they would need to significantly outperform bookmakers as bookmakers do not offer fair odds in practice (Hubáček et al., 2019).

## **Conclusion**

The existing literature showed the effectiveness of machine learning in predicting sports outcomes, as well as in betting strategies. Some key learnings from the prediction research were the importance of selecting relevant features, making sure that the dataset is balanced by using a sampling technique, and many insights on how such research should be conducted and what can be expected throughout the process. Betting research also provided valuable insights. From these articles, betting odds and money management were better understood. These papers also brought learnings regarding best practices when implementing model outputs into a betting strategy. Some learnings included the use of ensembles to gain extra information from your dataset and that the model with the most predictive power may not be the best model for betting purposes. The information regarding the ability of machine learning to perform in the sports and betting spaces supported this research and the insights in best practices the author learned helped in guiding its conduct.

## **Chapter 3: Methodology**

The goal of this research was to explore the viability of machine learning models in predicting the outcome of the MIP award voting process for use in the sports betting industry. The ability to accurately predict the probability of outcomes of sporting events has proven valuable for companies in the sports betting industry, as well as for sports bettors. Sportsbook companies use these probabilities to effectively set the odds at which they offer wagers. Having accurate odds helps companies to maximize their profits and to minimize the risk of heavily one-sided bets, by providing payouts that attract equal amounts of bettors on both sides. One example of this is for a game where team A has a 75% chance of beating team B. Using these probabilities, the payout for a $100 bet on team A would be $133 and $400 for team B. For this event, a sportsbook would want to have 3 times as many bettors for team A as they do for team B, to minimize risk. This way, the money paid out to bettors is equal to money wagered, regardless of the outcome. For bettors, being able to predict probabilities even a few percent more accurately than odds reflect can lead to positive gains. Referring back to the previous example, if a sportsbook provides 10 bets with implied odds of 75/25, but the bettor’s model provides probabilities of 70/30. If the bettor’s model is right, and the bettor placed ten $100 bets on the underdog, they would be paid out $1200 on their $1000 of wagers. These examples show that effective predictions provide value to both parties. The following parts of this section include the methodology overview and a deep dive into each section of the methodology. The overview section discusses rationale behind the methods used throughout the project and each deep dive provides details on how the task was accomplished.

**Methodology Overview**

The methodology of this project followed the traditional machine learning project life cycle, as shown in Awan’s blog (2022). The steps to be discussed in this section have been listed below in order:

1. Planning
2. Data Preparation
3. Model Engineering
4. Model Evaluation

The planning step is all about understanding the problem and what should be done to address the problem most effectively. As was covered in Chapter 2, the literature review, academic journal articles, academic research papers, and presentations at relevant conferences were researched in order to better understand the history of the MIP award, best practices for machine learning in the sports industry, and using model predictions as part of a betting strategy. This research led to insights regarding all other steps of the project and heavily influenced how it was conducted.

The data preparation step encompassed four main tasks (Awan, 2022):

* Data collection and labeling
* Data cleaning
* Data processing
* Data management

The data collection and labeling process included gathering historic and current player statistics from basketball\_reference.com and the NBA API. These sources were chosen due to their reputation for high data quality. Both sources have data provided by the official NBA stats partner SportsRadar. Since the value being estimated is the share of votes received for each player, data gathered on a player-by-player basis is required. Statistics for each player were gathered for each season due to the annual nature of the award. For example, when predicting the winner of the MIP award for the 2020-21 season, player’s stats from the 1990 season should not be considered, meaning that the voting for the award for 2020-21 season is based only on that season’s stats. Splitting by season allowed for only relevant data to be considered in the modelling process. The objective of these methods was to gather appropriate, reliable data to be used for modeling. The combination of these two sources accomplished just that. Only collecting historical data from basketball\_reference.com was also considered. However, considering the ability to make bets on the MIP award throughout the season, it was determined that real-time data, along with the associated predictions could provide value in both business cases.

The data cleaning and processing steps largely followed standard practices, including handling missing values, handling outliers, and creating additional features. However, since only a few players each year receive votes, the data was very unbalanced. To address this issue, the data was filtered by removing players who were very unlikely to win and was also over-sampled using the Synthetic Minority Over-Sampling Technique (SMOTE). Players were determined to be unlikely to win if their stats fell below three standard deviations less than the mean of the statistics for players who had received votes. For example, the mean points per game for those who received votes throughout the dataset was 16.14. One standard deviation in points per game for that same group was 5.02. The cutoff was then determined by subtracting three times the standard deviation from the mean resulting in a cutoff of: 16.14 – 3\*(5.02) = 1.07. More detail on filtering can be found in the data preparation section below. This combination of under-sampling and over-sampling was selected for multiple reasons. The first is due to the nature of the award. Very few players receive votes, leaving few observations where the target variable (award\_share) has values of interest. While under-sampling alone was mentioned in the Nguyen et al. study (2021), using both techniques better suited this research’s data. Under-sampling helped reduce the noise introduced to the model by players who did not have a chance of winning, like previous all-stars, and over-sampling gave the models more information about what drives award\_share. The Branco et al. study (2017), as well as the Chapman study (2023) also found SMOTE to increase performance in cases very similar to this research.

In terms of data management, the simplest approach was taken, storing historical data in csv files locally and retrieving current data from the NBA API on request. The historical data was also versioned to provide access to the data that was available at each step. Database solutions were considered at the time but were ultimately not chosen to give the author maximum flexibility in terms of experimentation with the data. For the final product, a database solution for historical data should be reconsidered.

The models that were trained during the model engineering process have been listed below:

* Decision Tree Regressor
* Random Forest Regressor
* Gradient Boosting Tree Models
  + XGBoost
  + LightGBM
  + Catboost
* Tensor Flow Neural Network
* Support Vector Regressor

The decision tree was trained as a baseline model to be used for comparison with other models. Since this model is relatively simple, the expectation was that a decision tree would not perform as well as other models. The Random Forest, Gradient Boosting models, and Neural Network were all chosen based on successful results from previous studies, some of which were shown in the literature review section. The support vector regressor (SVR) was chosen with the Wilkens (2021) study in mind. Wilkens found ensembles to perform best, especially when data was handled differently by each model. With the tree models and Neural Networks already present, the SVR made sense to include as it could include additional signal in an ensemble with the other models. Lastly, an ensemble with the models above was trained using the input from each in the calculation of award share. The objective of these models was to provide the best possible prediction performance. The linear regression model family was also considered, but these models did not suit this problem well due to the nature of the data. Hyperparameters were tuned for each model using the Optuna package in python. This was because the package allows for definition of custom objective functions, which was used to tailor each hyperparameter to what performed best.

The model evaluation step differed from typical methods. While typical metrics like mean absolute error (MAE) and mean squared error (MSE) were used to evaluate performance, the key evaluation metric in this study was how frequently the model correctly predicted the winner of the award. The objective of this metric was to ensure that the model gave the bettor or sportsbook the most information about how effective the models were at predicting the winner of the award. The use of root mean squared error (RMSE) was also considered but was not chosen as it would not add information with MSE already being used.

**Data Collection Details**

As stated in the overview, the data used for this project was sourced from basketball\_reference.com and the NBA API. The data gathered consisted of historical and current player-level regular season statistics gathered on a year-to-year basis. The historical data included all players from the 1985-86 season through the 2021-2022 season, with over 50 metrics describing each player’s season. This resulted in a dataset with 16,435 rows and 51 variables, including offensive and defensive stats such as games played, points per game, shooting percentages, rebounds per game, and many more. The data for the 2022-23 season as well as current data for the 2023-24 season was gathered using the NBA API. The 2022-23 data consisted of 362 rows and 28 variables. Since this season has now concluded at the time of this writing, it was appended to the historical dataset to be kept in static form. All of the metrics gathered for the 22-23 season were already included in the historical file and the variables that were not present in the API data were removed from the historical set to keep the number of variables consistent. The current data was also gathered from the NBA API at the time of execution of the scripts to perform the analysis, gathering the same variables as the rest of the data at that point in time. As this data did not have any labels, since the voting had not yet occurred, this was only be used for predictions. Some additional data considering if each player had been previously selected to be an all-star, on an all-NBA team, as MVP, or had already been selected as the MIP was gathered manually via basketball\_reference.com. Finally, the data was labelled with the award share for each year. Award share data was also gathered manually from basketball\_reference.com and appended to the historical dataset. This dataset was a population given it included the entire pool of players. However, as shown in the following section, a sample of this dataset was used for model training.

**Data Preparation Details**

The preparation of the data was driven by exploratory data analysis (EDA). The first step of the EDA was to determine if the data had any missing values. First, the data from the csv file was loaded using pandas python package, it was found that the only rows with null values were fg\_pct, fg3\_pct, fg2\_pct, and ft\_pct. To help inform what these values could be replaced with, the data was filtered to the rows where these values were null. Upon investigation, it was found that these null values were caused by 0 division. For example, fg\_pct was calculated by dividing the number of shots made by the number of shots taken. Using the columns’ labels the formula is as follows: fg\_pct = fg\_per\_g/fga\_per\_g. The other three columns followed the same pattern and when one of the attempts per game column was 0, this resulted in a null value. To handle this, the null values were replaced with zeros. The next step taken was to check the type of each column to ensure no issues were encountered during EDA or modelling. The only issue found here was the fg2\_pct column was listed as object type, rather than as float values. This was due to some values being converted to string where they should not have been. This issue was handled by converting the data back to numbers and setting the type to float64.

Some additional variables that were added to the dataset include the difference in minutes played per game, points per game, assists per game, rebounds per game, steals per game, blocks per game, and turnovers per game. These values were calculated by first sorting the data frame by player and by season. Then, the data was grouped by player and the pandas “diff” function was used on the columns to calculate the difference. This function calculated the difference between a row and the previous row. Grouping by player allowed for calculations of differences to only occur between different seasons for the same player. The difference variables intuitively fit into the dataset as they offered a way for the models to understand how much each player had improved compared to their previous season. One issue that was discovered in this process is that many players had the same name as players who had already played or were currently playing. This would lead to difference stats being calculated between two different players. To prevent this from happening, season difference and age difference columns were calculated. Any rows where season difference and age difference were equal for the same name, indicated the same person. Differences in these values indicated different players. Each unique player with the same name as another player had their name updated and the difference stats were recalculated. This also offered an easy way to identify each player’s first season, as each difference column would result in an N/A value. This was helpful as players cannot win in their first season and was used in the under-sampling process. Two additional columns were also added for comparison purposes. The first was the received\_vote column. This column checked to see if award\_share was greater than zero for each player, assigning a 1 to those with award\_share above zero and a 0 to the rest. The second column was the won\_mip column. This column grouped the data by year and found the max award\_share, which identified the winner for each year. These columns were very useful when looking at differences between winners, vote receivers, and non-vote receivers. Finally, the fga\_per\_g, fg2a\_per\_g, fg3a\_per\_g, and fta\_per\_g columns were removed as they do not add any information due to how they are calculated, as shown above. The total rebound per game (trb\_per\_g) column was also removed as it was calculated by summing the orb\_per\_g and drb\_per\_g columns.

One major issue that was found early in the EDA process was the extreme imbalance of the target variable. Of the players in the complete dataset, 16080 (95.7%) did not receive votes for the MIP award and only 717 (4.3%) did receive votes. As stated in the methodology overview, under-sampling through filtering and over-sampling using the SMOTE technique were implemented to address this issue. The data was filtered using these metrics to remove players highly unlikely to win the award. First, any player that played less than 29 games was removed. This was chosen as the cutoff by examining Figure 1 shown below.

**Figure 1**

*Games Played Distribution of Players who Received MIP votes vs Players with No Votes (Complete Dataset)*

A graph of a game

Description automatically generated with medium confidence

Looking at the graph, very few players had ever received votes while playing below 40 games. Further exploration into the data showed that no player had ever won the award playing less than 50 games and only one player had ever received votes playing 29 games or less. Filtering the data with the cutoff of 29 games, only one player who received votes was removed. Removing this player was negligible as the award share they received for that season was extremely small compared to the award share winners typically receive. This removal allowed for the model to focus less on the noise created by this player receiving votes and removed players who were not considered for the award because they did not play in enough games. Starting in the 2023-24 season, a minimum of 65 games played is a requirement to be eligible for the award, which should be implemented going forward. The dataset was also filtered using minutes, points, minutes difference, points difference, assists difference, rebounds difference, and turnover difference. For each of these metrics a cutoff was established using the 3-sigma rule for normally distributed data. This rule states that data falling outside three standard deviations from the mean signify rare occurrences (Chen, 2024). Before establishing the cutoff, each variable was examined to see if the distribution was approximately normal. An example is shown below in Figure 2.

**Figure 2**

*Points Per Game Difference of Players who Received MIP Votes vs Players with No Votes*

A graph of a graph showing the results of a game

Description automatically generated with medium confidence

Once an approximately normal distribution was confirmed, the cutoff was calculated by subtracting three times the standard deviation from the mean of the group that received votes. This removed many of the players who had no chance of winning while keeping the large majority of those who did receive votes intact. An example can be seen in Figure 3 below.

**Figure 3**

*Minutes Per Game Differential of Players who Received MIP Votes vs Players with No Votes with Cutoff*

A graph of different colored squares

Description automatically generated

All observations with a minutes per game differential to the left of the black cutoff line were removed from the dataset. The cutoff for each statistic can be seen in Table 1 below.

**Table 1**

*3 Sigma Cutoff for Each Statistic Used to Filter the Dataset*

A white text with black text

Description automatically generated

Along with the use of these cutoffs to filter, players who had previously been named an all-star, an all-NBA player, MVP. or had already won the MIP award were removed. This resulted from researching the history of the award, and no players with these characteristics had ever won the award. Lastly, players in their first season were also removed from the dataset, as a rookie has never won. This is due to the players not having previous seasons to compare against. These rows were identified by null values in the difference columns. After all of the filtering was complete, the dataset was more balanced, with 5403 (89.3%) players not receiving a vote and 649 (10.7%) players having received votes.

The final step in addressing the class imbalance was the implementation of the SMOTE technique, using the imblearn package. To accomplish this, categorical and irrelevant columns were removed. The portion of data to synthically create was defined as players who received an award share of at least 0.1, allowing the model to focus on players who received a significant share of votes. SMOTE was applied and the resulting data was merged with the filtered dataframe to create a balanced dataset with exactly 5403 players in each group.

A kernel density estimate plot was created to show the difference SMOTE makes in the distribution of the target variable. As shown below in Figure 4, using SMOTE created more data with significant award shares.

**Figure 4**

*KDE Plot of Award Share of the Filtered Dataset vs the Post SMOTE Dataset*

A graph of a graph

Description automatically generated with medium confidence

This balanced dataset allowed for models to have increased emphasis on those who received a significant amount of votes to better learn what drives the award share increase. However, the inclusion of players who received little or no votes helped the models determine what the statistics of those players look like as well. Without a balanced dataset, models would be able to predict 0 award share for all players and still have great MAE and MSE scores.

With imbalance addressed, the next issue to handle was feature scaling. While tree-based models do not require feature scaling (Thenraj, 2021), scaled data is highly recommended for SVM regressors as the scale of the features can influence the optimization of the decision function (Sotelo, 2018). This issue was handled with the MinMaxScaler from the python package called sklearn. This function scales each feature, so the given range is between 0 and 1, allowing for models with sensitivity to feature scaling to better perform.

Lastly, the data was split into training and test splits. For the seasons from 1987-2019, this was done by iterating through the list of seasons and training a model on the data that did not include the current season. The test split was identified by using only the data for the current season. For example, if the first season to be tested was 1987, the data from 1988-2019 was used to train the models and only the 1987 data would be used as a test set. Then, 1988 would be used as the test set and excluded from the training data. This same process was followed for each season from 1987-2019. For the 2020-2023 seasons, the training and testing was done as if it were done in production, using all previous seasons as training data and excluding any seasons after the season being tested. This testing method more resembled reality by excluding any data that would not yet exist when a user is predicting for the current year. This meant that for the 2020 season, all data from 1987-2019 was used for training but no data after the season being tested was included.

**Model Selection and Evaluation**

As stated in the methodology overview section above, the models selected for this project were largely driven by previous academic work.Tree-based models such as the Random Forest and the Gradient Boosting models had seen success in many studies such as the Chapman (2023) and Horvat and Job (2020) studies. Neural Networks were also shown to predict sporting events well in numerous studies including the Horvat and Job study. A Neural Network and SVR model were also included in this research with the Wilkens (2021) study in mind, where Wilkens found ensembles of models with unique approaches to perform the best. With the tree-based models already included, the Neural Network and SVR made sense as they offered unique methods for processing input data that could complement the tree-based models. The linear-regression family of models was also considered. However, they were not chosen as the relationship between the explanatory variables and target was not linear, making these models unsuitable for this problem.

For the model evaluation process, it was very important to make sure the models were being accurately evaluated in the context of the problem at hand. To do this, traditional metrics like MAE and MSE were used but only played a small role. This was because the difference between the prediction and real values were less important than if the model was able to predict the highest value for the player who had won the award. However, MAE and MSE were still included because they gave indications that the models were able to predict well across seasons and gave some insight into how they might generalize to new data. The metric that played the largest role in the evaluation process was the number of seasons the model was able to predict the winner correctly. This was due to sportsbooks typically only offering wagers on the winner of the MIP award. This context made the distinction between the winner and the rest of the players the major focus of this study. For example, who finishes in second place for the award is irrelevant to if a bettor has a successful wager, making only the winner is important. The number of correctly predicted winners also doubled as the number of successful wagers, and in the context of this research was the main indicator of success. The hyperparameters of the Gradient Boosting models and SVR were optimized using this metric as well. This was done using the Optuna package within python. Optuna allowed for custom objective function definition, making it possible to directly maximize the number of seasons where the winner was predicted correctly. Additional metrics like if the model’s top scoring prediction was included in the top 2 and top 3 of the actual voting results were also included to gauge accuracy, but also played a limited role compared to correctly predicted winners.

**Conclusion**

In summary, this research followed the typical life cycle of a machine learning project, completing the following four steps in order: planning, data preparation, model engineering, and model evaluation. The methodology overview section covered the rationale behind the methods executed throughout each step. For the planning step, academic journal articles, academic research papers, and presentations at relevant conferences were researched to better understand the topics at hand. Data preparation included collection and labeling using data from basketball\_refernce.com and the NBA API, typical cleaning steps including filling null values and removing outliers, preprocessing steps including addressing imbalance and splitting for training and testing. The models trained on this data included a Decision tree, a Random Forest, a Support Vector Regressor, Gradient Boosting tree models (XGBoost, Catboost, LightGBM), and a Neural Network. These models were evaluated by looking at how many winners were correctly predicted and hyperparameters were tuned using this same metric. The models were then combined into a weighted average ensemble with weights chosen by a voting classifier. In the next section, the results of these models will be discussed. This will include the results of the models individually, as well as the performance of the ensemble and its potential for use in a betting strategy.

**Chapter 4: Results**

The first objective of this study was to construct and evaluate models to predict the winner of the MIP award using multiple methods. To do this, six models were trained and tested on each player’s season-level statistics. Each model was tested in two different ways. The first was using data from 1987-2019. For each season, the data was split into the season being predicted and the remaining seasons. For example, when predicting the MIP winner in the 2000 season, each model was trained on data from 1987-1999 and 2001-2019. This method was repeated for each season from 1987-2019, totaling 33 seasons. The second method of testing included each model being tested on each season from 2020-2023, totaling four seasons, using data only from previous seasons. If 2021 were the season being tested, all data from 1987-2020 would be used for training, better resembling what the models would be used for in reality.

## **Model Results Before Hyperparameter Tuning**

Each model was evaluated using five metrics including the number of correctly predicted winners, the number of predicted winners who finished in the top 2, the number of predicted winners who finished in the top 3, mean MAE, and mean MSE. The results of the evaluation of the models can be seen in Table 2 below.

**Table 2**

Comparison of Models Results for 1987 – 2019 Seasons Before Hyperparameter Tuning

A screenshot of a graph

Description automatically generated

*The three left-most metrics are shown as a percentage of seasons where the top prediction of the model fell into that category. The total number of seasons for this table was 33.*

Looking at the mean MAE and MSE scores, all of the models had scores very close to zero. If the performance of the models was based on these metrics alone, it would have seemed that the Neural Network (nn) and Catboost models had performed the best. However, since the objective of the models was to correctly predict the winner, the addition of the “Correct Predicted Winner”, “Predicted Winner in Top 2”, and “Predicted Winner in top 3” metrics was required to fully assess the performance of the models. Based on these metrics, it was shown that correctly predicting the winner of the award did not have a direct relationship with MAE or MSE. When investigating correct predicted winners alone, the Random Forest regressor (rfr) performed the best with correct predictions in 24 of the 33 seasons, while having the second highest average MSE. This was because MAE and MSE measure the difference between predictions and actual values for each observation. In the context of this research, having accurate predictions across the entire pool of players was not as important as accurately identifying the top candidate for the award. For example, an ideal model for this problem would always correctly identify the winner with an award share prediction of 1 and predict 0 for the rest of the players. How accurate the predictions for the players who did not win would not matter, but the MAE and MSE scores would likely be higher since those who did receive votes but did not win the award would have larger differences between the prediction of 0 and their actual award share.

After further investigation of each models’ predictions, it was found that there were a few years that all models had trouble with. This included the seasons of 1988, 1990, 1991, 1997, 2006, and 2007. Although all the models struggled with these years, they also performed well as they moved toward the most recent years. In the seasons of 2009-2019, the model that performed the worst was the support vector regressor (SVR) model, correctly predicting 7 of the 11 winners. All other models correctly predicted 9 of the 11 winners in this time frame. If the years where models had issues correctly predicting the winners were corrected and used in future predictions, it would be possible for them to perform worse when generalizing to new data. Due to this reason, no action was taken to predict those years more accurately.

To further address the performance of the models, each model was also tested on the 2020-2023 seasons. This was done the same way that it would be done had the models been used in production. For example, when predicting for the 2020 season, the models would be trained on all the available data before the 2020 season, in this case, 1987-2019. When predicting for 2021, the 2020 season would also be incorporated. The results for each model can be seen in Table 3 below.

**Table 3**

Comparison of Models Results for 2020-2023 Seasons Before Hyperparameter Tuning

A screenshot of a graph

Description automatically generated

*The three left-most metrics are shown as a percentage of seasons where the top prediction of the model fell into that category. The total number of seasons for this table was 4.*

The Neural Network performed notably worse than the others, with only one of the predicted winners resulting in a player that had finished in the top three. The rest of the models performed similarly with 2-4 correct predictions and the predicted winner finishing at least in the top 2 for each season.

## **Model Results After Hyperparameter Tuning**

To see if these results could be improved, the XGBoost, lightgbm, SVR, and Catboost models had hyperparameters tuned using the Optuna package in python. Optuna allows for custom metrics to be minimized or maximized. For this research, the number of correctly predicted winners during the 1987-2019 seasons was maximized. For the Gradient Boosting models, parameters like max depth, number of estimators, and learning rate were tuned. For the SVR model, the kernel function was tuned. Optuna randomly selected values for each hyperparameter based on the provided range, then a model would be trained and tested for each season and the number of correct predictions would be counted. Once the number of correct predictions was finalized, new parameters were selected, and the process began again. For the XGBoost, Lightgbm, and Catboost models, 50 trails were completed. The SVR model only required 4 trails due to the limited number of potential kernel functions. The results of this test are shown below in Table 4.

**Table 4**

Comparison of Models Results for 1987-2019 Seasons After Hyperparameter Tuning

A screenshot of a graph

Description automatically generated

*The three left-most metrics are shown as a percentage of seasons where the top prediction of the model fell into that category. The total number of seasons for this table was 33.*

Comparing the results of these models against those in Table 2, it can be seen that three of the four models increased the number seasons where the winner was predicted correctly. For these same models, we can also see that the MAE and MSE scores increased compared to the models with default hyperparameters. Models like XGBoost and lightgbm minimized these metrics by default and the ‘poly’ kernel function for the SVR model was more effective at identifying the player with the most votes but less effective at predicting the rest of the field. The Catboost model performed worse after this test than it did with its default hyperparameters. This was likely due to the low number of trails being used in the optimization process. Since the values of the hyperparameters were selected at random, a combination that performed better than the default values was not found. If the number of trails were increased from 50 to 500, it is likely that the performance would improve.

While it was good to see that most models performed better after tuning on historical data, it was also important to test the performance on the 2020-2023 seasons as well to see how they would have performed in production. The results of this test can be seen in Table 5 below.

**Table 5**

Comparison of Models Results for 2020-2023 Seasons After Hyperparameter Tuning

A screenshot of a computer screen

Description automatically generated

*The three left-most metrics are shown as a percentage of seasons where the top prediction of the model fell into that category. The total number of seasons for this table was 4.*

Comparing these results to the results shown in Table 4, it can be seen that the XGBoost model correctly predicted an additional winner, the lightgbm model correctly predicted the same number of winners, and the Catboost and SVR models correctly predicted one less winner than their pre-tuning counterparts. The lightgbm and SVR models were good examples of how performing better on previous data, does not necessarily mean that the models will perform better on unseen data. This increased conviction that some of the seasons that models were having trouble predicting do not provide value when generalizing to new data.

**Ensemble Results**

Lastly, an ensemble of all models was trained. This was done using the predictions from the tuned XGBoost and lightgbm models, and default Neural Network, Random Forest, SVR, and Catboost models. The predictions of each model were weighted as follows: XGBoost predictions \* 0.225 + lightgbm predictions \* 0.225 + Catboost predictions \* 0.225 + rfr predictions \* 0.225 + SVR predictions \* 0.05 + nn predictions \* 0.05. Models that performed well on all tests were given higher weights and poorly performing models received lower weights. The ensemble was subjected to the same tests as in the other sections. The results can be seen in Tables 6 below.

**Table 6**

Ensemble Results for 1987-2019 & 2020-2023 Seasons

A screenshot of a report

Description automatically generated

The ensemble had average performance compared to the standalone models when predicting for the 1987-2019 seasons. However, for the 2020-2023 seasons, it stood out as one of the top performers, with the correct winner being predicted for all four seasons.

**Sports Betting Viability**

Based on the results of the 2020-2023 seasons, the viability of using model predictions in a sports betting strategy was also evaluated. Starting in 2020 and assuming the betting odds at that time were the same for previous years as they were at time of this writing (-1350), a starting amount of $100 in 2020, would have turned into $133 in 2023 after the four correct predictions. This equates to a yearly growth rate of 7.4%. Due to the nature of the award, the odds at the end of the season are very unfavorable for the bettor and using the models in this way would not be recommended.

**Conclusion**

Six models were trained and tested using data from the 1987-2019 and 2020-2023 seasons. The Random Forest performed the best without hyperparameter tuning. The XGBoost and Lightgbm models performed the best with hyperparameter tuning. A weighted average ensemble of all models was also trained and tested on both sets of data. The performance was similar to the other models for the 1987-2019 seasons but stood out in the 2020-2023 seasons, correctly predicting the winner for all four years. The XGBoost, Random Forest, and ensemble performed well enough to justify its use in future predictions, however, due to the unfavorable betting odds at the end of the season, they are not recommended to be used for betting on the winner just before it is announced.

**Chapter 5: Discussion**

This chapter wraps up the project by summarizing key points and discussing the implications of the results presented in Chapter 4. In the findings section, a summary of the key findings of the study is presented. The conclusions section takes a look at the two main project objectives presented in chapter 1 and discusses whether the objectives were fulfilled. The implications section discusses what the results presented in this report mean for those who are looking to accomplish similar objectives in a professional setting. Lastly, the limitations and suggestions sections walk through the limitations of the results and offer suggestions to build upon the research presented here.

**Findings**

When testing for seasons from 1987-2019, the models were found to correctly predict the winner between 58%-73% of the 33 seasons, with the SVR model with the least number of correct predictions and the Random Forest model with the highest number of correct predictions. After tuning hyperparameters with the goal of maximizing the number of correct predictions, the XGBoost and Lightgbm models tied the Random Forest model with a correct prediction for 73% of seasons. Further investigation into the predictions revealed a large difference in performance between the 1987-2000 seasons and the 2001-2019 seasons. For example, the XGBoost model with tuned hyperparameters correctly predicted 8 of the 14 (57%) seasons between 1987-2000 but had correct predictions for 16 of the 19 (84%) seasons from 2001-2019. All models showed an increase in performance across the 2001-2019 timeframe, compared to the 1987-2000 timeframe.

When testing for the 2020-2023 seasons, the models correctly predicted the winner between 25% and 100% of time. The Neural Network had the worst performance, only predicting the correct player to win the award for 1 of the 4 seasons. The Catboost and Random Forest models both predicted the winner for each of the four seasons. After hyperparameter tuning, the XGBoost model also predicted each winner for the 2020-2023 seasons correctly. Based on performance for all seasons from 1987-2023, the Random Forest and the tuned XGBoost models had the best performance, correctly predicting the winner for 28 of 36 (78%) seasons, including correct predictions for the 9 most recent seasons.

A weighted average ensemble was also tested in the same format as above, giving models who performed better on their own more weight. For the 1987-2019 seasons, 67% of the winners were predicted correctly, and each of the winners for 2020-2023 seasons were predicted correctly. Overall, the models showed strong predictive ability, especially when looking at recent years.

**Objectives Conclusions**

The primary objective of this project was to construct and evaluate models to predict the winner of the MIP award using multiple methods. This objective was accomplished by training and testing six different models, as well as a weighted average ensemble of the six models, on two sets of data. The models were then evaluated using traditional and custom metrics to assess performance. The performance of each model was then compared against its peers. The use of these methods gave the author confidence the performance of the models was legitimate and provided evidence of what generalization to new data could look like for each model.

The secondary objective of this project was to explore if the model output produced in the process described above could be implemented into a sports betting strategy. This was accomplished by using the model outputs of the best performing models for the 2020-2023 seasons. With correct predictions for each season, it was assumed that a wager was placed and won at the end of the NBA season. Betting odds used to calculate the return were assumed to be the same as they were at the time of this writing, as the NBA season concluded at the same time. Using these assumptions, the yearly return calculated was 7.4%. Due to the relatively low return, along with the risk of the winner-takes-all format of the bets, the use of the model outputs as part of a betting strategy is not recommend in their current state.

**Implications for Professional Use**

The primary learning that can be considered for professional use based on the results of this is the avoidance of betting on the MIP award at the end of the NBA season. Due to the nature of the award, typically the pool of players has been narrowed to two or three heavy favorites, resulting in very unfavorable odds for bettors. While betting at the end of the season is not advised, the models did perform well, especially after the 2000 season. The results presented here indicate that machine learning techniques are a viable method for predicting the outcome of the MIP award. However, changes to the timing of the data and betting strategy would have to be changed to produce better returns. Suggestions for changes are discussed in the final section of this chapter.

**Limitations**

The primary limitation of this study is the use of end of season data. As previously stated, the nature of the award typically leads to unfavorable odds for bettors, resulting in poor returns regardless of accurate model predictions. Another limitation was the exclusion of certain statistics that could potentially lead to better performance. These stats include team-based stats like how many games the team won and how they placed overall. Advanced player statistics like player efficiency rating and win shares could also be included to give additional information on their performance. From a betting perspective, potential strategies and the timeframe analyzed were limited, offering only a small window into potential returns.

**Suggestions for Future Research**

For those who are interested in building upon this research, some suggestions to explore have been listed below.

**Potential Accuracy Improvements:**

* Update the threshold for observations to be created by SMOTE to a higher number. This would change the data synthetically created to only players with a higher award share, giving additional emphasis on players who receive high vote shares. This could help models better differentiate between two players who would receive high award shares in the same season.
* Include team-based and advanced NBA statistics in the modeling process. Stats like a team’s win percentage along with advanced player stats like player efficiency rating, win shares, and value over replacement player, could add additional information about each players performance.
* Sentiment data from articles or social media could be incorporated to see who fans and experts are favoring for the award.
* Additional modeling techniques could be explored. For example, it is possible to further explore Neural Network architecture to see if it improves performance or use more advanced ensemble techniques to get the most out of each model.
* Allow for more trials for hyperparameter tuning**.** 50 trails were used for the Gradient Boosting models, but ideally a huge number of trails and possible combinations of parameters could be tested to maximize accuracy.

**Potential Betting Improvements:**

* Exploration of additional strategies and an increased timeframe could provide more information on how the models would have performed in reality.
* Use mid-season data to make bets when odds are much more favorable for bettors.

**Conclusion**

In conclusion, six models and a weighted average ensemble of the six models were trained and tested on two segments of data. The Random Forest and XGBoost models performed the best, correctly predicting the winner of the MIP award for 28 of the 36 (78%) seasons, including correct predictions for the last nine seasons in a row. The viability of use in betting was evaluated using the results for the 2020-2023 seasons, assuming betting odds for the previous seasons were the same as they were for the current season. It was found that the four correct predictions resulted in a gain of 33%, equating to 7.4% on a yearly basis. Due to the risk involved in a winner-takes-all betting scenario, it is not recommended to use the model outputs for betting. However, the suggestions listed above could provide improved accuracy and increased returns. Overall, the models provide a good baseline for prediction and with some improvements made, the viability of model outputs in a betting strategy can be reevaluated.

## 

## **References**

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna. *Association for Computing Machinery*. https://doi.org/10.1145/3292500.3330701

Awan, A. A. (2022, October 3). The machine learning life cycle explained. <https://www.datacamp.com/blog/machine-learning-lifecycle-explained>

Basic regression: Predict fuel efficiency. (n.d.). *TensorFlow*. <https://www.tensorflow.org/tutorials/keras/regression#normalization>

Biau, G., Cadre, B., & Rouvière, L. (2019). Accelerated gradient boosting. *Machine Learning*, 108(6), 971–992. https://doi.org/10.1007/s10994-019-05787-1

Branco, P., Torgo, L., Ribeiro, R. P., & LIAAD-INESC TEC DCC-FCUP, University of Porto. (2017). SMOGN: a Pre-processing Approach for Imbalanced Regression. In L. Torgo, B. Krawczyk, P. Branco, & N. Moniz (Eds.), *Proceedings of Machine Learning Research* (Vol. 74, pp. 36–50). <https://proceedings.mlr.press/v74/branco17a/branco17a.pdf>

Breidt, F. J., & Chauvet, G. (2012). Penalized balanced sampling. *Biometrika*, 99(4), 945–958. https://doi.org/10.1093/biomet/ass033

Chapman, A. L. (2023). The application of machine learning to predict the NBA regular season MVP (Order No. 30487274). *Available from ProQuest Dissertations & Theses Global*. (2811391944).<https://ezproxy.uwgb.edu/login?url=https://www.proquest.com/dissertations-theses/application-machine-learning-predict-nba-regular/docview/2811391944/se-2>

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *The Journal of Artificial Intelligence Research*, 16, 321–357. <https://doi.org/10.1613/jair.953>

Chen, J. (2024, March 13). Normal Distribution: what it is, uses, and formula. *Investopedia*. <https://www.investopedia.com/terms/n/normaldistribution.asp>

Chicco, D., Warrens, M. J., & Jurman, G. (2021). The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. *PeerJ. Computer Science*, 7, e623–e623. https://doi.org/10.7717/peerj-cs.623

Choi, B., Lee Kien Foo, & Chua, S.-L. (2023). Predicting Football Match Outcomes with Machine Learning Approaches. *Mendel* ..., 29(2), 229–236. https://doi.org/10.13164/mendel.2023.2.229

Etuk, R., Xu, T., Abarbanel, B., Potenza, M. N., & Kraus, S. W. (2022). Sports betting around the world: A systematic review*. Journal of Behavioral Addictions*, 11(3), 689-715. <https://doi.org/10.1556/2006.2022.00064>

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 63(1), 3–42. https://doi.org/10.1007/s10994-006-6226-1

Horvat, T., & Job, J. (2020). The use of machine learning in sport outcome prediction: A review. *WIREs Data Mining and Knowledge Discovery*, 10(5). <https://doi.org/10.1002/widm.1380>

Hubáček, O., Šourek, G., Železný, F. (2019) Exploiting sports-betting market using machine learning. *International Journal of Forecasting,* Volume 35, Issue 2, 2019, Pages 783-796, https://doi.org/10.1016/j.ijforecast.2019.01.001.

Interactive Map: Sports Betting in the U.S. (n.d.). *American Gaming Association*. <https://www.americangaming.org/research/state-gaming-map/>

Jacot, B. P., & Mochkovitch, P. V. (2023). Kelly criterion and fractional Kelly strategy for non-mutually exclusive bets. *Journal of Quantitative Analysis in Sports*, 19(1), 37–42. <https://doi.org/10.1515/jqas-2020-0122>

Jiang, J. (2022, May 26). Simple Weighted Average ensemble | Machine Learning. *Medium*. <https://medium.com/analytics-vidhya/simple-weighted-average-ensemble-machine-learning-777824852426>

Lim, Y. (2022, April 8). State-of-the-Art Machine Learning Hyperparameter Optimization with Optuna. *Medium*. https://towardsdatascience.com/state-of-the-art-machine-learning-hyperparameter-optimization-with-optuna-a315d8564de1

Matej, U., Gustav, Š., Ondřej, H., & Filip, Ž. (2021). Optimal sports betting strategies in practice: an experimental review. *IMA Journal of Management Mathematics*, *32*(4), 465–489. <https://doi-org.ezproxy.uwgb.edu/10.1093/imaman/dpaa029>

Mollenkamp, D. T. (2024, February 5). Live betting: What it is and how it works. *Investopedia*. https://www.investopedia.com/live-betting-definition-5217206

NBA & ABA Most Improved Player (George Mikan Trophy) award winners | Basketball-Reference.com. (n.d.). Basketball-Reference.com. https://www.basketball-reference.com/awards/mip.html

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*, 1–19. <https://doi.org/10.1080/24751839.2021.1977066>

Parameter tuning |. (n.d.).

https://catboost.ai/en/docs/concepts/parameter-tuning

Parameters Tuning — LightGBM 4.3.0.99 documentation. (n.d.). https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html#tune-parameters-for-the-leaf-wise-best-first-tree

Rithp. (2023, January 16). Optimizing XGBOOSt: A Guide to Hyperparameter tuning. *Medium*. https://medium.com/@rithpansanga/optimizing-xgboost-a-guide-to-hyperparameter-tuning-77b6e48e289d

Robertsunderhaft. (2022, July 13). Predicting the NBA MVP. *Kaggle*. <https://www.kaggle.com/code/robertsunderhaft/predicting-the-nba-mvp#Data-Preprocessing>

Smets, K., Verdonk, B., Jordaan, E. M., Department of Mathematics & Computer Science, University of Antwerp, Antwerp, Belgium, & Core Research & Development, Dow Benelux B.V., Terneuzen, The Netherlands. (2021). Evaluation of performance measures for SVR hyperparameter selection. http://adrem.uantwerpen.be/bibrem/pubs/IJCNN2007.pdf

Sotelo, D. (2018, May 6). Effect of feature standardization on linear support vector Machines. *Medium*. <https://towardsdatascience.com/effect-of-feature-standardization-on-linear-support-vector-machines-13213765b812>

Swar. (n.d.). GitHub - swar/nba\_api: An API Client package to access the APIs for NBA.com. GitHub. <https://github.com/swar/nba_api>

Thenraj, P. (2021, December 14). Do Decision Trees need Feature Scaling? - Towards Data Science. *Medium*. https://towardsdatascience.com/do-decision-trees-need-feature-scaling-97809eaa60c6

Wilkens, S. (2021). Sports prediction and betting models in the machine learning age: The case of tennis. *Journal of Sports Analytics*, 1–19. <https://doi.org/10.3233/jsa-200463>

**Appendix**

**Code to Gather Data from NBA API:**

from nba\_api.stats.endpoints import playercareerstats  
from nba\_api.stats.static import players  
from nba\_api.stats.endpoints import boxscoreadvancedv2  
import pandas as pd  
  
active = players.get\_active\_players()  
ids = []  
for i in range(len(active)):  
 id = active[i]['id']  
 ids.append(id)  
  
names = []  
for id in ids:  
 if id in exclusions:  
 print('excluded')  
 else:  
 player = players.find\_player\_by\_id(id)  
 name = player['full\_name']  
 names.append(name)  
  
df = pd.DataFrame()  
exclusions = []  
for id in ids:  
 career = playercareerstats.PlayerCareerStats(per\_mode36='PerGame', player\_id=[id])  
 try:  
 last\_row = career.get\_data\_frames()[0].iloc[-1]  
 df = pd.concat([df, last\_row.to\_frame().T])  
 df = df.reset\_index(drop=True)  
 except IndexError:  
 print('stop')  
 exclusions.append(id)  
  
df\_2022 = pd.DataFrame()  
exclusions = []  
for id in ids:  
 career = playercareerstats.PlayerCareerStats(per\_mode36='PerGame', player\_id=[id])  
 try:  
 last\_row = career.get\_data\_frames()[0].iloc[-2]  
 df\_2022 = pd.concat([df\_2022, last\_row.to\_frame().T])  
 df\_2022 = df\_2022.reset\_index(drop=True)  
 except IndexError:  
 print('stop')  
 exclusions.append(id)  
  
df\_2022['player'] = names  
  
df\_2022\_only = df\_2022[df\_2022['SEASON\_ID']=='2022-23']  
df\_2022\_only.to\_csv('C:/Users/sierr/Desktop/MSDS/Capstone/NBA\_Dataset\_2022.csv')

**Code Used for EDA:**

import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
import numpy as np  
from imblearn.over\_sampling import SMOTE  
from sklearn.preprocessing import MinMaxScaler  
# reading in data from excel file  
df = pd.read\_csv('C:/Users/sierr/Desktop/MSDS/Capstone/NBA\_Dataset\_currentV3.csv')  
# looking at first few rows of data  
df.head()  
# taking a look at non-null values and column types for comparison purposes  
df.info()  
  
# updating counts so all are numeric, then updating type to float  
df.loc[df['fg2\_pct'] == '#DIV/0!', 'fg2\_pct'] = 0  
df['fg2\_pct'] = df['fg2\_pct'].astype('float64')  
  
# creating columns to check if a player received votes and if they won the award for comparison  
df['received\_vote'] = df['award\_share'].apply(lambda x: 1 if x > 0 else 0)  
winners = df.groupby(by="season").max('award\_share')  
winners['won\_mip'] = 1  
df = df.merge(winners[["award\_share", "won\_mip"]], on=["season", "award\_share"], how="left")  
df["won\_mip"] = df["won\_mip"].fillna(value=0)  
  
# checking to make sure all correct values are returned  
winners\_df = df[(df['won\_mip'] == 1) & (df['season'] > 1986)].sort\_values(by='season')  
  
# removing fga columns as they are calculated by fg/fg\_pct  
plt.scatter(df["fga\_per\_g"], df["fg\_per\_g"] / df["fg\_pct"])  
plt.xlabel("fg\_per\_g")  
plt.ylabel("fga\_per\_g / fg\_pct")  
plt.title("FG Column Relationship")  
plt.show(block=True)  
  
# removing unnecessary columns, trb\_per\_g included  
df = df.drop(columns=["Unnamed: 0", "fga\_per\_g", "fg3a\_per\_g", "fg2a\_per\_g", "fta\_per\_g", "trb\_per\_g"], axis=1)  
  
  
# function form Sunderhaft kaggle notebook 2022  
def func(pct, allvals):  
 absolute = int(np.round(pct / 100. \* np.sum(allvals)))  
 return "{:.1f}%/n({:d})".format(pct, absolute)  
  
  
# pie chart showing who received votes vs who did not  
plt.pie(df['received\_vote'].value\_counts(),  
 autopct=lambda pct: func(pct, df['received\_vote'].value\_counts()),  
 pctdistance=1.25)  
plt.title('Players that Received MIP votes (Complete Dataset)')  
plt.legend(['No Votes', 'Received Votes'])  
plt.show(block=True)  
  
# Under-Sampling by removing players who have no chance of winning  
  
# checking correlations to see what we could filter off of - diff columns have highest correlations  
corr\_matrix = df.corr()  
sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm')  
plt.show(block=True)  
  
  
# function to make plot showing difference between those who received votes and did not  
def vote\_diff\_plot(col):  
 plt.hist(df[df['received\_vote'] == 1][col], bins=10, alpha=0.7, density=True, label='Received Votes')  
 plt.hist(df[df['received\_vote'] == 0][col], bins=10, alpha=0.7, density=True, label='No Votes')  
 plt.ylabel('Proportion')  
 plt.xlabel('Minutes Per Game Difference')  
 plt.legend(['Received Votes', 'No Votes'])  
 plt.title('Minutes Per Game Difference with Cutoff')  
 plt.axvline(x=mpgdiff\_cutoff, color='black')  
 plt.show(block=True)  
  
  
# Going to check to see who has received votes by games played - looks like very few under 30 games  
vote\_diff\_plot('g')  
# checking vote receivers under 30 games - going to use 29 games to only remove Sean Elliot in 2000  
under\_30 = df[(df['received\_vote'] == 1) & (df['g'] < 29)]  
games\_cutoff = 29  
  
# checking gs  
vote\_diff\_plot('gs')  
# checking vote receivers under 20 games started - not helpful as bench player can win  
under\_20 = df[(df['received\_vote'] == 1) & (df['gs'] < 20)]  
  
# checking mp\_per\_g  
vote\_diff\_plot('mp\_per\_g')  
# going to use 3 std from mean as it is approximately normal  
minutes\_cutoff = df[df['received\_vote'] == 1]['mp\_per\_g'].mean() - 3 \* df[df['received\_vote'] == 1]['mp\_per\_g'].std()  
# checking who is removed - removes 2 all with ~0 award share  
minutes\_cutoff\_df = df[(df['received\_vote'] == 1) & (df['mp\_per\_g'] < minutes\_cutoff)]  
  
# checking pts\_per\_g  
vote\_diff\_plot('pts\_per\_g')  
# going to use 3 std from mean as it is approximately normal  
points\_cutoff = df[df['received\_vote'] == 1]['pts\_per\_g'].mean() - 3 \* df[df['received\_vote'] == 1]['pts\_per\_g'].std()  
# checking who is removed - removes 0  
points\_cutoff\_df = df[(df['received\_vote'] == 1) & (df['pts\_per\_g'] < points\_cutoff)]  
  
# checking mpg\_diff  
vote\_diff\_plot('mpg\_diff')  
# going to use 3 std from mean as it is approximately normal  
mpgdiff\_cutoff = df[df['received\_vote'] == 1]['mpg\_diff'].mean() - 3 \* df[df['received\_vote'] == 1]['mpg\_diff'].std()  
# checking who is removed - removes 0  
mpgdiff\_cutoff\_df = df[(df['received\_vote'] == 1) & (df['mpg\_diff'] < mpgdiff\_cutoff)]  
  
# checking ppg\_diff  
vote\_diff\_plot('ppg\_diff')  
# going to use 3 std from mean as it is approximately normal  
ppgdiff\_cutoff = df[df['received\_vote'] == 1]['ppg\_diff'].mean() - 3 \* df[df['received\_vote'] == 1]['ppg\_diff'].std()  
# checking who is removed - removes 4 with very small shares  
ppgdiff\_cutoff\_df = df[(df['received\_vote'] == 1) & (df['ppg\_diff'] < ppgdiff\_cutoff)]  
  
# checking ast\_diff  
vote\_diff\_plot('astpg\_diff')  
# going to use 3 std from mean as it is approximately normal  
astdiff\_cutoff = df[df['received\_vote'] == 1]['astpg\_diff'].mean() - 3 \* df[df['received\_vote'] == 1]['astpg\_diff'].std()  
# checking who is removed - removes 1 with very small shares  
astdiff\_cutoff\_df = df[(df['received\_vote'] == 1) & (df['astpg\_diff'] < astdiff\_cutoff)]  
  
# checking rebpg\_diff  
vote\_diff\_plot('rebpg\_diff')  
# going to use 3 std from mean as it is approximately normal  
rebpg\_diff\_cutoff = df[df['received\_vote'] == 1]['rebpg\_diff'].mean() - 3 \* df[df['received\_vote'] == 1]['rebpg\_diff'].std()  
# checking who is removed - removes 0  
rebpg\_diff\_cutoff\_df = df[(df['received\_vote'] == 1) & (df['rebpg\_diff'] < rebpg\_diff\_cutoff)]  
  
# checking tov\_diff  
vote\_diff\_plot('tov\_diff')  
# going to use 3 std from mean as it is approximately normal  
tov\_diff\_cutoff = df[df['received\_vote'] == 1]['tov\_diff'].mean() - 3 \* df[df['received\_vote'] == 1]['tov\_diff'].std()  
# checking who is removed - removes 0  
tov\_diff\_cutoff\_df = df[(df['received\_vote'] == 1) & (df['tov\_diff'] < tov\_diff\_cutoff)]  
  
# removing values based on cutoffs  
filtered\_df = df[(df['g'] >= games\_cutoff) & (df['mp\_per\_g'] >= minutes\_cutoff) &  
 (df['pts\_per\_g'] >= points\_cutoff) & (df['mpg\_diff'] >= mpgdiff\_cutoff) &  
 (df['ppg\_diff'] >= ppgdiff\_cutoff) & (df['astpg\_diff'] >= astdiff\_cutoff) &  
 (df['rebpg\_diff'] >= rebpg\_diff\_cutoff) & (df['tov\_diff'] >= tov\_diff\_cutoff)]  
  
# removing all players with previous total > 0 and rows with null diff columns  
filtered\_df = filtered\_df[(filtered\_df['previous\_total'] == 0) & (filtered\_df['mpg\_diff'].notnull())]  
  
# separating numerical columns from categorical columns for graphs  
num\_cols = [col for col in filtered\_df.columns if  
 filtered\_df[col].dtype == np.float64 or filtered\_df[col].dtype == np.int64]  
cat\_cols = [col for col in filtered\_df.columns if  
 filtered\_df[col].dtype != np.float64 and filtered\_df[col].dtype != np.int64]  
  
pct\_cols = [col for col in filtered\_df.columns if 'pct' in col or 'award' in col]  
diff\_cols = [col for col in num\_cols if 'diff' in col]  
reg\_cols = [col for col in num\_cols if col not in pct\_cols and col not in diff\_cols and 'season' not in col]  
# Looking for clear outliers from descriptive stats  
stats = pd.DataFrame(filtered\_df.describe())  
  
# checking numerical columns for outliers  
sns.boxplot(data=filtered\_df[pct\_cols])  
plt.xticks(rotation=30)  
plt.show(block=True)  
  
sns.boxplot(data=filtered\_df[diff\_cols])  
plt.xticks(rotation=30)  
plt.show(block=True)  
  
sns.boxplot(data=filtered\_df[reg\_cols])  
plt.xticks(rotation=90)  
plt.show(block=True)  
  
filtered\_df['previous\_mvp'].value\_counts()  
filtered\_df['previous\_mip'].value\_counts()  
filtered\_df['previous\_all\_star'].value\_counts()  
filtered\_df['previous\_all\_nba'].value\_counts()  
  
sns.pairplot(data=filtered\_df[pct\_cols])  
plt.show(block=True)  
  
sns.pairplot(data=filtered\_df[diff\_cols])  
plt.show(block=True)  
  
sns.pairplot(data=filtered\_df[reg\_cols[0:5]])  
plt.show(block=True)  
  
stats = []  
p\_vals = []  
cols = []  
for col in pct\_cols:  
 sns.distplot(filtered\_df[col], kde=False, bins=10)  
 plt.title(f'Distribution of {col}')  
 plt.xlabel('Value')  
 plt.ylabel('Frequency')  
 plt.show(block=True)  
  
# filtering results in more balanced data, however it is still quite imbalanced  
plt.pie(filtered\_df['received\_vote'].value\_counts(),  
 autopct=lambda pct: func(pct, filtered\_df['received\_vote'].value\_counts()),  
 pctdistance=1.25)  
plt.title('Players that Received MIP Votes Post Filtering')  
plt.legend(['Received Votes', 'No Votes'])  
plt.show(block=True)  
  
# Using SMOTE to balance  
cols\_to\_drop = ['player', 'team\_id', 'previous\_mvp', 'previous\_mip', 'previous\_all\_star', 'previous\_all\_nba',  
 'previous\_total', 'season\_diff', 'age\_diff', 'received\_vote', 'won\_mip']  
smote\_df = filtered\_df.drop(cols\_to\_drop, axis=1)  
classes = filtered\_df["award\_share"] > .1  
sm = SMOTE(random\_state=42)  
smote\_df, classes = sm.fit\_resample(smote\_df, classes)  
# indexing synthetic data  
smote\_df['is\_synthetic'] = smote\_df.index >= len(filtered\_df)  
# creating dfs for modelling  
final\_forest\_df = smote\_df.merge(filtered\_df, how='left')  
scaler = MinMaxScaler()  
final\_SVM\_df = pd.DataFrame(scaler.fit\_transform(smote\_df))  
final\_SVM\_df.columns = scaler.get\_feature\_names\_out()  
final\_SVM\_df = final\_SVM\_df.merge(filtered\_df, how='left')  
final\_forest\_df.to\_csv('C:/Users/sierr/Desktop/MSDS/Capstone/forest\_df.csv')  
# kde plot before and after smote  
sns.kdeplot(filtered\_df['award\_share'], label="Original")  
sns.kdeplot(smote\_df['award\_share'], label="Post SMOTE")  
plt.legend()  
plt.title("KDE Plot of Award Share of Original Data vs After SMOTE")  
plt.show(block=True)  
# post smote balance  
plt.pie(classes.value\_counts(),  
 autopct=lambda pct: func(pct, classes.value\_counts()),  
 pctdistance=.25)  
plt.title('Players that Received MIP Votes Post SMOTE')  
plt.legend(['Received Votes', 'No Votes'])  
plt.show(block=True)

**Code Used for Modeling:**

import catboost as cb  
import lightgbm as lgb  
import numpy as np  
import optuna  
import pandas as pd  
import tensorflow as tf  
import warnings  
from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.pipeline import make\_pipeline  
from sklearn.preprocessing import StandardScaler  
from sklearn.svm import SVR  
from tensorflow import keras  
from tensorflow.keras import layers  
from xgboost import XGBRegressor  
  
# reading in dataframe  
df = pd.read\_csv('C:/Users/sierr/Desktop/MSDS/Capstone/forest\_df.csv')  
# shuffling data  
df = df.sample(frac=1).reset\_index(drop=True)  
# selecting columns to be used in training  
train\_cols = ['age', 'g', 'gs', 'mp\_per\_g', 'fg\_per\_g',  
 'fg\_pct', 'fg3\_per\_g', 'fg3\_pct', 'fg2\_per\_g', 'fg2\_pct', 'ft\_per\_g',  
 'ft\_pct', 'orb\_per\_g', 'drb\_per\_g', 'ast\_per\_g', 'stl\_per\_g',  
 'blk\_per\_g', 'tov\_per\_g', 'pf\_per\_g', 'pts\_per\_g', 'mpg\_diff',  
 'ppg\_diff', 'astpg\_diff', 'rebpg\_diff', 'stl\_diff', 'blk\_diff', 'tov\_diff', ]  
# selecting identifier columns  
id\_cols = ["season", "player", "team\_id", "award\_share", "won\_mip", "is\_synthetic"]  
# identifying target column  
target = 'award\_share'  
  
# creating df used for training  
model\_df = df[train\_cols]  
# creating id df for post training  
train\_id = df[id\_cols]  
# creating df of target vals  
target\_df = df[[target]]  
  
# filtering the train df to allow for testing on data the model has not seen  
train\_df = model\_df[train\_id['season'] <= 2019]  
train\_targets = target\_df[train\_id['season'] <= 2019]  
seasons = train\_id.season.unique()  
train\_seasons = np.sort(seasons[seasons <= 2019])  
  
test\_df = model\_df[train\_id['season'] > 2019]  
test\_df2 = test\_df[~train\_id["is\_synthetic"]]  
test\_seasons = seasons[seasons > 2019]  
test\_seasons = np.sort(test\_seasons)  
test\_targets = target\_df[(train\_id['season'] > 2019) & (~train\_id["is\_synthetic"])]  
test\_id = train\_id[(train\_id["season"] > 2019) & (~train\_id["is\_synthetic"])]  
  
warnings.filterwarnings("ignore", message="Boolean Series key will be reindexed")  
np.random.seed(1)  
  
# see models and params.txt for models and params  
  
# HP Optimization  
def objective(trial):  
 counter = 0  
  
 param = {  
 'max\_depth': trial.suggest\_categorical('max\_depth', [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None]),  
 'min\_samples\_leaf': trial.suggest\_categorical('min\_samples\_leaf', [1, 2, 4]),  
 'min\_samples\_split': trial.suggest\_categorical('min\_samples\_split', [2, 5, 10]),  
 'n\_estimators': trial.suggest\_categorical('n\_estimators',  
 [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]),  
 }  
  
 for season in train\_seasons:  
 print(f"Season: {season}")  
  
 train\_fold = train\_df[train\_id["season"] != season]  
 train\_target = train\_targets[train\_id["season"] != season]  
 val\_fold = train\_df[(train\_id["season"] == season) & (~train\_id["is\_synthetic"])]  
 val\_targets = train\_targets[(train\_id["season"] == season) & (~train\_id["is\_synthetic"])]  
 val\_id = train\_id[(train\_id["season"] == season) & (~train\_id["is\_synthetic"])]  
  
 rfr = RandomForestRegressor(\*\*param)  
 rfr.fit(train\_fold, train\_target.to\_numpy()[:, 0])  
  
 preds = rfr.predict(val\_fold)  
 most\_votes = np.argmax(preds)  
 won\_mip = val\_id.iloc[[most\_votes]]["won\_mip"].values[0]  
 if won\_mip == 1:  
 counter += 1  
 else:  
 counter = counter  
 return counter  
  
  
study = optuna.create\_study(direction='maximize')  
study.optimize(objective, n\_trials=50)  
  
xgb\_params = {'n\_estimators': 121,  
 'max\_depth': 7,  
 'learning\_rate': 0.31481936178743364,  
 'colsample\_bytree': 0.5067762388773397,  
 'min\_child\_weight': 18}  
  
gbm\_params = {'num\_iterations': 110,  
 'max\_bin': 457,  
 'max\_depth': 8,  
 'learning\_rate': 0.4119792828422604,  
 'num\_leaves': 90}  
  
normalizer = tf.keras.layers.Normalization(axis=-1)  
normalizer.adapt(np.array(model\_df))  
  
counter = 0  
validation\_scores = {"season": [], "mae": [], "mse": [], "won\_mip": [], "was\_top\_two": [], "was\_top\_three": [],  
 "id\_info": []}  
# 1987 - 2019  
for season in train\_seasons:  
 print(f"Season: {season}")  
  
 train\_fold = train\_df[train\_id["season"] != season]  
 train\_target = train\_targets[train\_id["season"] != season]  
 val\_fold = train\_df[(train\_id["season"] == season) & (~train\_id["is\_synthetic"])]  
 val\_targets = train\_targets[(train\_id["season"] == season) & (~train\_id["is\_synthetic"])]  
 val\_id = train\_id[(train\_id["season"] == season) & (~train\_id["is\_synthetic"])]  
  
 xgb = XGBRegressor(\*\*xgb\_params)  
 xgb.fit(train\_fold, train\_target.to\_numpy()[:, 0])  
  
 preds = xgb.predict(val\_fold).flatten()  
  
 mae = mean\_absolute\_error(preds, val\_targets.to\_numpy()[:, 0])  
 mse = mean\_squared\_error(preds, val\_targets.to\_numpy()[:, 0])  
 top\_two = val\_id.iloc[np.argsort(preds)[-2:]]  
 was\_top\_two = sum(top\_two["won\_mip"]) > 0  
 top\_three = val\_id.iloc[np.argsort(preds)[-3:]]  
 was\_top\_three = sum(top\_three["won\_mip"]) > 0  
 print("Predicted top three players in MVP voting with their actual award\_share:")  
 print(top\_three.iloc[::-1])  
  
 most\_votes = np.argmax(preds)  
 score = np.amax(preds)  
 won\_mip = val\_id.iloc[[most\_votes]]["won\_mip"].values[0]  
 if won\_mip == 1:  
 counter += 1  
 else:  
 counter = counter  
 player = val\_id.iloc[[most\_votes]]["player"].values[0]  
  
 validation\_scores["season"].append(season)  
 validation\_scores["mae"].append(mae)  
 validation\_scores["mse"].append(mse)  
 validation\_scores["won\_mip"].append(won\_mip)  
 validation\_scores["was\_top\_two"].append(was\_top\_three)  
 validation\_scores["was\_top\_three"].append(was\_top\_three)  
 validation\_scores["id\_info"].append(val\_id.iloc[[most\_votes]])  
  
cbr\_df = pd.DataFrame(validation\_scores)  
cbr\_df['won\_mip'].sum()  
cbr\_df['was\_top\_two'].sum()  
cbr\_df['was\_top\_three'].sum()  
cbr\_df['mae'].mean()  
cbr\_df['mse'].mean()  
  
  
# xgb\_df = pd.DataFrame(validation\_scores)  
# xgb\_df['won\_mip'].sum()  
# xgb\_df.groupby('won\_mip')['mse'].mean()  
  
# gbm\_df = pd.DataFrame(validation\_scores)  
# gbm\_df['won\_mip'].sum()  
# gbm\_df.groupby('won\_mip')['mse'].mean()  
  
  
  
# gbm\_params = study.best\_params  
  
counter = 0  
validation\_scores = {"season": [], "mae": [], "mse": [], "won\_mip": [], "was\_top\_two": [], "was\_top\_three": [],  
 "id\_info": []}  
# 2020 - 2023  
for season in test\_seasons:  
 print(f"Season: {season}")  
  
 train\_fold = model\_df[train\_id["season"] < season]  
 train\_target = target\_df[train\_id["season"] < season]  
 val\_fold = model\_df[(train\_id["season"] == season) & (~train\_id["is\_synthetic"])]  
 val\_targets = target\_df[(train\_id["season"] == season) & (~train\_id["is\_synthetic"])]  
 val\_id = train\_id[(train\_id["season"] == season) & (~train\_id["is\_synthetic"])]  
  
 xgb = XGBRegressor(\*\*xgb\_params)  
 gmb = lgb.LGBMRegressor(\*\*gbm\_params)  
 cbr = cb.CatBoostRegressor()  
 rfr = RandomForestRegressor()  
 svr = make\_pipeline(StandardScaler(), SVR())  
 model = keras.Sequential([  
 normalizer,  
 layers.Dense(64, activation='relu'),  
 layers.Dense(64, activation='relu'),  
 layers.Dense(1)  
 ])  
 model.compile(loss='mean\_absolute\_error',  
 optimizer=tf.keras.optimizers.Adam(0.001))  
  
 xgb.fit(train\_fold, train\_target.to\_numpy()[:, 0])  
 gmb.fit(train\_fold, train\_target.to\_numpy()[:, 0])  
 cbr.fit(train\_fold, train\_target.to\_numpy()[:, 0])  
 rfr.fit(train\_fold, train\_target.to\_numpy()[:, 0])  
 svr.fit(train\_fold, train\_target.to\_numpy()[:, 0])  
 model.fit(train\_fold, train\_target.to\_numpy()[:, 0])  
  
 xgb\_preds = xgb.predict(val\_fold).flatten()  
 gmb\_preds = gmb.predict(val\_fold).flatten()  
 cbr\_preds = cbr.predict(val\_fold).flatten()  
 rfr\_preds = rfr.predict(val\_fold).flatten()  
 svr\_preds = svr.predict(val\_fold).flatten()  
 nn\_preds = model.predict(val\_fold).flatten()  
  
 preds = (  
 xgb\_preds \* 0.225 + gmb\_preds \* 0.225 + cbr\_preds \* 0.225 + rfr\_preds \* .225 + svr\_preds \* .05 + nn\_preds \* .05)  
  
 mae = mean\_absolute\_error(preds, val\_targets.to\_numpy()[:, 0])  
 mse = mean\_squared\_error(preds, val\_targets.to\_numpy()[:, 0])  
 top\_two = val\_id.iloc[np.argsort(preds)[-2:]]  
 was\_top\_two = sum(top\_two["won\_mip"]) > 0  
 top\_three = val\_id.iloc[np.argsort(preds)[-3:]]  
 was\_top\_three = sum(top\_three["won\_mip"]) > 0  
 print("Predicted top three players in MVP voting with their actual award\_share:")  
 print(top\_three.iloc[::-1])  
  
 most\_votes = np.argmax(preds)  
 score = np.amax(preds)  
 won\_mip = val\_id.iloc[[most\_votes]]["won\_mip"].values[0]  
 if won\_mip == 1:  
 counter += 1  
 else:  
 counter = counter  
 player = val\_id.iloc[[most\_votes]]["player"].values[0]  
  
 validation\_scores["season"].append(season)  
 validation\_scores["mae"].append(mae)  
 validation\_scores["mse"].append(mse)  
 validation\_scores["won\_mip"].append(won\_mip)  
 validation\_scores["was\_top\_two"].append(was\_top\_three)  
 validation\_scores["was\_top\_three"].append(was\_top\_three)  
 validation\_scores["id\_info"].append(val\_id.iloc[[most\_votes]])  
  
cbr\_df = pd.DataFrame(validation\_scores)  
cbr\_df['won\_mip'].sum()  
cbr\_df['was\_top\_two'].sum()  
cbr\_df['was\_top\_three'].sum()  
cbr\_df['mae'].mean()  
cbr\_df['mse'].mean()

**Modeling Info (Plugged in and out of Modeling Script depending on what was being trained and tested):**

**### xgboost**

**# xgb = XGBRegressor()**

**# xgb.fit(train\_fold, train\_target.to\_numpy()[:, 0])**

**# xgboost params**

**# params = {**

**# 'n\_estimators': trial.suggest\_int('n\_estimators', 50, 500),**

**# 'max\_depth': trial.suggest\_int('max\_depth', 3, 8),**

**# 'learning\_rate': trial.suggest\_float('learning\_rate', 0.005, .5, log=True),**

**# 'colsample\_bytree': trial.suggest\_float('colsample\_bytree', 0.5, 1, log=True),**

**# 'min\_child\_weight': trial.suggest\_int('min\_child\_weight', 1, 300)**

**# }**

**# best xgb params**

**xgb\_params = {'n\_estimators': 121,**

**'max\_depth': 7,**

**'learning\_rate': 0.31481936178743364,**

**'colsample\_bytree': 0.5067762388773397,**

**'min\_child\_weight': 18}**

**### lightgbm**

**# gmb = lgb.LGBMRegressor()**

**# gmb.fit(train\_fold, train\_target.to\_numpy()[:, 0])**

**# lightgbm params**

**# params = {**

**# 'num\_iterations': trial.suggest\_int('n\_estimators', 50, 500)**

**# 'max\_bin': trial.suggest\_int('n\_estimators', 250, 500),**

**# 'max\_depth': trial.suggest\_int('max\_depth', 3, 8),**

**# 'learning\_rate': trial.suggest\_float('learning\_rate', 0.005, .5, log=True),**

**# 'num\_leaves': trial.suggest\_int('num\_leaves', 50, 125)**

**# }**

**# best gbm params**

**gbm\_params = {'num\_iterations': 110,**

**'max\_bin': 457,**

**'max\_depth': 8,**

**'learning\_rate': 0.4119792828422604,**

**'num\_leaves': 90}**

**### catboost**

**# cbr = cb.CatBoostRegressor()**

**# cbr.fit(train\_fold, train\_target.to\_numpy()[:, 0])**

**# catboost params**

**# params = {**

**# 'iterations': trial.suggest\_int('iterations', 50, 500),**

**# 'depth': trial.suggest\_int('depth', 4, 10),**

**# 'learning\_rate': trial.suggest\_float('learning\_rate', 0.005, .5, log=True),**

**# }**

**#best cbr params**

**cbr\_params = {'iterations': 441,**

**'depth': 9,**

**'learning\_rate': 0.4119792828422604}**

**### random forest**

**# rfr = RandomForestRegressor()**

**# rfr.fit(train\_fold, train\_target.to\_numpy()[:, 0])**

**### SVR**

**# svr = make\_pipeline(StandardScaler(), SVR())**

**# svr.fit(train\_fold, train\_target.to\_numpy()[:, 0])**

**# svm params**

**# params = {**

**# 'kernel': trial.suggest\_categorical('kernel', [‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’]),**

**# }**

**#best svm params**

**kernel='poly'**

**### Neural Net**

**normalizer = tf.keras.layers.Normalization(axis=-1)**

**normalizer.adapt(np.array(train\_features))**

**model = keras.Sequential([**

**normalizer,**

**layers.Dense(64, activation='relu'),**

**layers.Dense(64, activation='relu'),**

**layers.Dense(1)**

**])**

**model.compile(loss='mean\_absolute\_error',**

**optimizer=tf.keras.optimizers.Adam(0.001))**

**model.fit(train\_fold, train\_target.to\_numpy()[:, 0])**