MODELING DAILY FANTASY BASKETBALL

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Martin Jiang

March 2023

COMMITTEE MEMBERSHIP

TITLE: Modeling Daily Fantasy Basketball

AUTHOR: Martin Jiang

DATE SUBMITTED: March 2023

COMMITTEE CHAIR: Rodrigo Canaan, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Ayaan Kazerouni, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Franz KurfessMugizi Rwebangira, Ph.D.
Professor of Computer Science

ABSTRACT

Modeling Daily Fantasy Basketball

Martin Jiang

Daily fantasy basketball presents interesting problems to researchers due to the extensive amounts of data that needs to be explored when trying to predict player performance. A large amount of this data can be noisy due to the variance within the sport of basketball. Because of this, a high degree of skill is required to consistently win in daily fantasy basketball contests. On any given day, users are challenged to predict how players will perform and create a lineup of the eight best players under fixed salary and positional requirements. In this thesis, we present a tool to assist daily fantasy basketball players with these tasks. We explore the use of several machine learning techniques to predict player performance and develop multiple approaches to approximate optimal lineups. We then compare each different heuristic and lineup creation combination, and show that our best combinations perform much better than random lineups. Although creating provably optimal lineups is computationally infeasible, by focusing on players in the Pareto front between performance and cost we can reduce the search space and compute near optimal lineups. Additionally, our greedy and evolutionary lineup search methods offer similar performance at a much smaller computational cost. Our analysis indicates that due to how player salaries are structured, it is generally preferred to construct a lineup consisting of a few stars and filling out the rest of the roster with average to mediocre players than to construct a lineup where all players are expected to perform about the same. Through these findings we hope that our research can serve as a future baseline towards developing an automated or semi-automated tool to optimize daily fantasy basketball.

ACKNOWLEDGMENTS

Thanks to:

- My parents, for their support throughout my undergraduate and graduate education
- Dr. Canaan, for his advice and guidance throughout my thesis and masters experience
- Andrew Guenther, for uploading this template

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

With an estimated $1.8 billion dollars spent on sports betting advertising in 2022 [1], it's difficult to watch any sporting event without seeing ads for DraftKings or FanDuel. Through these two companies, sports betting has quickly taken over Europe and America. Millions of users place wagers on a wide variety of sports every day, and while many hobbyists have tried developing their own models to gain an advantage over other participants, daily fantasy sports still remains a relatively new topic in academia. Briefly stated, daily fantasy sports presents users with the challenge of both predicting how players will perform on any given day, and using these predictions to create a lineup within some fixed constraints.

In this paper, we present a tool to assist users with this challenge, specifically in the context of daily fantasy basketball. We first go over the relevant background information; introducing the NBA, daily fantasy basketball, and the scoring rules our paper is focused on. We then present some of the existing research surrounding the topic, along with introducing a few algorithms that our methods were inspired by. The remaining bulk of the paper is spent describing how we developed the methods used to predict player performances and create lineups. These methods were then evaluated by combining each heuristic with each lineup creation method, and comparing them to see which methods stood out above the others. Lastly, we conclude the paper by summarizing our findings and discussing various avenues of future work.

1.1 Ethical Considerations

Critics in the past have argued that daily fantasy sports should be classified as a form of online gambling, which is illegal in the United States due to the Federal Wire Act of 1961 [2]. This law prohibits the use of wire communications to place sports wagers across state lines. However, interpretations of this law, its implications, and its applications have fluctuated over time. Originally proposed and designed to suppress organized crime in the mid 60s, the rise of internet-based gambling, daily fantasy sports, and continual changes in political power changed how this law was applied several times over the last few decades. In 2006, the debate of whether or not daily fantasy sports should classify as online gambling was largely settled because of the Unlawful Internet Gambling Enforcement Act [3], which exempted daily fantasy sports due to the general agreement that they require a degree of skill that separates it strictly from gambling [4]. As of 2022, individual states have jurisdiction over whether or not daily fantasy sports are illegal, and they are currently legal in 43 out of 50 states.

There is also discussion surrounding the ethics of using software tools to assist individuals in daily fantasy sports. Some view the use of outside tools as providing an unfair advantage that others may not have access to, however DraftKings itself does not prohibit individuals from using outside tools or information when participating in daily fantasy sports, and the vast majority of regular participants in daily fantasy sports do use some form of outside information when creating lineups. Many forms of external information designed to assist daily fantasy sports players can be found online through forums, podcasts, projections, or other paid online tools. Although there is inherent luck involved in sports, a high degree of knowledge and expression of skill is needed to consistently perform well in daily fantasy sports.

Chapter 2

BACKGROUND

This chapter begins with a brief explanation of the NBA and fantasy sports. We first highlight the origin, growth, and structure of both the NBA and fantasy sports. Afterwards, we introduce daily fantasy sports and the popular platforms they're hosted on, along with the classic mode tournament format structure and scoring rules.

2.1 The Origin and Growth of the NBA

The National Basketball Association (NBA) was originally founded in 1946 as the Basketball Association of America and rebranded as the NBA after merging with the National Basketball League in 1949. As of 2022, the NBA consists of 30 teams split into a Western Conference and an Eastern Conference. Each team can have a maximum of 15 players and plays 82 games over the course of the regular season, which typically lasts from October through April. After the regular season, the top 8 teams from both conferences proceed to compete in the playoffs, which consist of of three rounds and are followed by the NBA finals. Each of these series is played in a best-of-seven format, where the loser is eliminated and the winner advances and plays against the winner from another series until one team remains.

Basketball is a team sport that is typically played 5-on-5, where the goal is to score more points than the opposing team by shooting a basketball through a net and preventing them from scoring. However, because this paper is centered more on daily fantasy basketball than the sport itself, the rules and intricacies of the game will not

be discussed further. For a more in-depth explanation of the game of basketball and its rules, please refer to the NBA Rule Book [5] or Donald (2023) [6].

The popularity of basketball has grown steadily since the sport's invention in the late 19th century and is currently at an all time high, largely due to television contract deals and notable players such as Michael Jordan, LeBron James, and Stephen Curry. Currently, basketball is America's second most popular sport in terms of viewership, trailing only behind football. In addition to this, according to NBA commissioner Adam Silver, during the 2021-2022 season the NBA surpassed $10 billion in revenue for the first time, bouncing back from a rough year after COVID-19 [7], and nearly tripling over the last decade from $3.7 billion in 2012 [8].

2.2 Traditional Fantasy Basketball

Although the complete origins of fantasy basketball are unknown, it is believed to have originated in the mid-60s from an adaptation of the rules used in fantasy baseball and football. Participants in a fantasy basketball league would meet prior to the beginning of the NBA season, and draft players to form their fantasy team. Every week, these players would be scored based on how they played, and for each team, these scores would be summed together. The participant whose team scored the most points that week would be considered the winner, and the participant who won the most weeks throughout the season won the fantasy league.

Traditional fantasy sports have grown in popularity throughout the 21st century, largely due to their ability to be played over the internet on platforms like ESPN and Yahoo. Most traditional fantasy basketball leagues are played using a head-to-head or rotisserie format, where teams are drafted in a snake or auction style. In a head-to-head league, fantasy teams compete one-on-one, and players are scored based on their

real-life performance in the following box score categories: points, rebounds, assists, steals, blocks, threes, field-goal percentage, free-throw percentage, and turnovers. At the end of the week, the individual whose fantasy team scored more points wins, and at the end of the season the individual who won the most weeks is declared the winner of the league. In a rotisserie league (often abbreviated to roto league), an individual competes against all other players each week. Instead of assigning point values to player performances, the scoring categories are counted separately, and the individual whose fantasy team accumulates the most in one box score category wins that category for the week. The individual who wins the most categories overall throughout the entire course of the season wins the fantasy league.

However, because teams can only be altered through trading with another player or free agency, traditional fantasy basketball teams largely remain unchanged throughout the NBA season. This slow pace and somewhat long-term time investment also makes it difficult to come back when players have significant injuries. These shortcomings in the inflexibility of changing players on a roster and the slow pace eventually led to the development and growth of daily fantasy sports.

2.3 Daily Fantasy Sports

Daily fantasy sports are somewhat similar to traditional fantasy sports, but a few structure adjustments are introduced to address the weaknesses of traditional fantasy sports stated above. The overall goal of selecting players to maximize fantasy points scored remains the same, but in daily fantasy sports, players are assigned salaries and lineups are constructed by "purchasing" players using imaginary currency with the constraint of a hard salary cap along with a strict roster size and position requirement. Tournaments typically last no longer than a week, and individuals are often encour-

aged to enter multiple lineups, allowing them to construct different lineups according to different strategies, matchups, and formats. Having a quick turnaround also allows players to restart regularly with a blank slate on a regular basis, rather than having to recoup losses throughout an NBA season like one would when participating in traditional fantasy sports. What used to be a somewhat long-term commitment became fast and convenient through mobile applications. Through companies like DraftKings and FanDuel, players can play from their phones, win, and cash out all within the same day.

Daily fantasy sports have also grown significantly over the last decade. Exact numbers are hard to pinpoint, but current estimates of the of daily fantasy sports market value sit at around $20 billion [9], and this number is expected to grow significantly over the coming years. The two companies that hold the largest shares of this market are DraftKings and FanDuel. Both of these companies have risen above their competitors through their aggressive advertising campaigns, accessibility through mobile applications, and sponsorships from various major sports leagues. As of 2022, the NBA recognizes both DraftKings and FanDuel as their official sports betting partners [10].

2.4 Daily Fantasy Tournaments Formats and Scoring Rules

The remainder of this paper will be focused on the context of DraftKings, and trying to develop a tool to assist individuals in creating lineups for daily fantasy basketball. DraftKings hosts a wide variety of competitions that slightly vary in rules and scoring. For this paper, we focused specifically on the classic mode game type. In classic mode, individuals are tasked with selecting a lineup of eight players to fill out the following positions: Point Guard (PG), Shooting Guard (SG), Small Forward (SF), Power Forward (PF), Center (C), Guard (G), Forward (F), and Utility (UTIL). In

addition to the position requirement, individuals must also stay below the $50,000 salary cap. Once lineups are finalized, they are submitted and scored according to their real-life performance. At the end of the day, lineups are scored, and the players whose lineup(s) performed the best receive a cash prize. The scoring breakdown for DraftKings classic mode can be seen in Table 2.1.

Box Score Statistic	**Fantasy Point Value**
Points	+1 Pt
Made 3pt Shot	+0.5 Pts
Rebound	+1.25 Pts
Assist	+1.5 Pts
Steal	+2 Pts
Block	+2 Pts
Turnover	-0.5 Pts
Double-double [1,2]	+1.5 Pts
Triple-double	+3 Pts

Table 2.1: DraftKings Classic Mode Scoring Rules

Winning techniques and strategies can change depending on the structure of the tournament. On DraftKings, payout amounts and number of entries vary greatly, ranging from a few dollars to a few hundred thousand dollars to top performers, and from one entry to an unlimited number of entries. There are also different payout structures, such as double-or-nothing, where the participant either doubles their investment or loses it all, or tiered-payout structures, where the majority of the winnings are handed out to a certain top percentage of the players. All of these factors would alter how a player would approach the problem of creating a lineup. For example, in a high stakes low entry lobby, one would want to select players who have proven to be very consistent or are facing a weaker team. But in a lobby that allows

[1]A "double" refers to a player recording double digits (i.e. at least 10 units) in a box score statistic. In classic mode, the box score statistics that count towards a double-double and triple double are points, rebounds, assists, blocks, and steals.

[2]A player can only earn a maximum of one double-double and one-triple double. For example, a player that recorded 10 points, 10 rebounds, 10 steals, and 10 blocks would receive 1.5 Pts for a double-double and 3 Pts for a triple double, in total earning himself 4.5 Pts. He would not receive multiple double-double or triple-rewards.

a higher number of entries, one may want to create several lineups where four of eight players remain the same, but introduce variance for the rest of the roster by selecting riskier players that might outperform players that are expected to be consistent. For the purpose of the paper, we will be focusing on predicting player performances and generating a single lineup for the purpose of maximizing expected fantasy points per game, and not creating multiple lineups with varying strategies or consideration of risk.

Chapter 3

RELATED WORKS

Although the topic of daily fantasy basketball in academia is still relatively new, researchers and hobbyists have applied a wide array of techniques in an attempt to optimize daily fantasy basketball. In this chapter, we highlight the previous work others have done in attempting to optimize daily fantasy basketball, briefly highlighting their techniques used and summarizing their results. Furthermore, we also introduce a few algorithms from which some of our methods were inspired by.

3.1 Attempts at Optimizing Daily Fantasy Basketball

In 2016, Christopher Barry, Nicholas Canova, and Kevin Capiz [11] used ridge regression to estimate the daily fantasy performance for individual players. Using season rolling averages as a baseline predictor, they developed a ridge regression model from these averages and incrementally added features to see how the root mean-squared error and mean absolute error were affected. Their model was fit using player season averages and fantasy point production up to the current game. Features added thereafter included the opposing team's defensive statistics, the opposing team's defensive statistics by position, a players' number of rest days, increasing the weight of recent performances, and home court advantage. Of these features, it was found that considering rest, home court advantage, and weighting recent performance decreased error by a reasonable margin, but including opponent defensive statistics did not, highlighting the challenges in trying to reduce prediction errors after a certain value.

Our work extends this by further evaluating similar models by creating lineups from their predictions.

Eric Hermann and Adebia Ntoso [12] also attempted to optimize daily fantasy basketball by separating the problem into predicting player performance and constructing lineups using these predictions. They theorized that if they were able to predict player performance better than DraftKings projections, they would be able to identify undervalued players (players who scored more fantasy points compared to their peers with similar salaries) and select those players when constructing lineups. Using the 2014-2015 NBA Season as their training and test set, they compared linear regression using players last five games and a naive Bayes model to predict player performance, and found that while the naive Bayes model was able to perform about as well as DraftKings projections, the linear regression model had significant advantages when making predictions for players that scored more fantasy points. For players that scored at least 20 fantasy points, the model outperformed DraftKing projections by about 7.5%. Constraint satisfaction was used to create lineups, but due to the overwhelming number of players to consider, beam search was used to limit the search space only up to the top 32 candidate lineups. Their work also highlighted the challenges in trying to accurately predict player performance, along with trying to create lineups from these predictions, as they were only able to keep track of the top 32 lineups at any given point during creation.

To account for the inherent randomness in basketball, Charles South, Ryan Elmore, Andrew Clarage, Rob Sickorez, and Jing Cao [13] used a Bayesian model to predict player performance. Their data was sourced from the 2013-2014 season, but they chose to only include the second half to ensure that player performances were relatively stable. Features for their Bayesian model were selected by using the lasso [14] to fit models for each day and retaining the features that recurred the most often. These

features included average fantasy points scored over the last 10 games, whether the player started, average turnovers over the last 10 games, average defensive rebounds over the last 10 games, average field goals over the last 10 games, and opponent strength. To construct lineups after predicting player performance, two approaches were used. The first was a permutation-based approach that generated the vast majority of competitive lineups while removing invalid lineups based on salary, and retaining only the most promising ones. The second was a classification-based approach that incorporated sports betting information such as the line and over/under to help identify successful lineups (a lineup that scored more than 260 fantasy points was considered successful). This system was tested by conducting a hypothetical experiment, simulating 100 daily competitions over the course of the 2015-2016 NBA season. This experiment was repeated 500 times, and they found that with an initial investment of $500, the average total profit was $9,008 with the permutation-based lineup construction and $6,453 with the classification based lineup construction. However, though these results seem promising, they should be taken with a grain of salt, as it is impossible to gauge how informed daily fantasy players are on a regular basis. It is also worth noting that since their model relies on the second half of the NBA season for stability purposes, it's unknown how reliable their methods would be at the beginning of the season.

3.2 Multi-Objective Optimization

As will be seen in section 4.3.1, it is computationally expensive or infeasible to generate all possible line-ups for a given day in order to find the optimal one. For this reason, we turned to the fields of multi-objective optimization and evolutionary computation. Multi-objective optimization refers to techniques (often evolutionary in nature) that attempt to optimize two or more objectives at once. For our use case,

we will be attempting to maximize expected fantasy points while minimizing cost. Evolutionary computation takes inspiration from survival of the fittest, and consists of algorithms and techniques used to gradually drive a population towards optimization by repeatedly removing less desirable individuals and introducing small random changes.

In 2002, Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan [15] developed NSGA-II, a genetic algorithm to solve multiobjective optimization problems. We wanted to highlight NSGA-II because some of the methods we developed were inspired by how NSGA-II handles multi-objective optimization. Specifically, the multi-level Pareto front and the genetic algorithms discussed in Chapter 4 were inspired in part by the non-dominated sorting used in NSGA-II.

NSGA-II improves upon its predecessor, NSGA, by improving computational complexity, introducing elitism, and removing the need for including a sharing parameter, which sets the extent of how different any two solutions must be. NSGA-II differs from other genetic algorithms as it begins the selection process with non-dominated sorting, or finding the set of solutions that are non-dominated to each other but strictly superior to the rest of the solutions. A solution S is considered non-dominated if no other solution outperforms S in all objectives, and the set of non-dominated solutions is also called a Pareto front. To find the solutions on the first non-dominated front, each solution is compared with every other solution. To find the solutions on the second non-dominated front, these solutions are temporarily removed from consideration and the previous step is repeated. This can be repeated until solutions on all Pareto fronts are found.

When the size of the final front in addition to the currently selected individuals exceeds the population size, tie-breaking occurs by using a crowd-comparison approach. Instead of using a sharing parameter, diversity is preserved by first estimating the

density of solutions surrounding a particular solution and then prioritizing solutions that occupy low-density regions.

After selecting the elites, variations of these elites are introduced as new candidates in the population, and the above steps are repeated. NSGA-II was evaluated via simulation on several test problems, and it was found that NSGA-II found a much wider spread of solutions while being able to converge on the Pareto-optimal solutions faster than other multi-objective evolutionary algorithms.

3.3 Knapsack Problem

In this section, we introduce the knapsack problem, provide its formal definition, and describe the challenge of trying to solve it. In addition, we also highlight a few existing algorithms to solve it. We chose to highlight the knapsack problem because the overall problem structure shares similarities with creating fantasy basketball lineups. These similarities are further discussed in section 4.3.5.

The knapsack problem is a combinatorial optimization problem where, given a set of items, each with an assigned weight and value, the goal is to select a collection from these items with the maximum value while ensuring that the total weight of the collection is less than or equal to a certain threshold. The most common form of the knapsack problem is the 0-1 knapsack problem, where each item can be chosen only once or not at all.

Defined formally, given a set of n items numbered 1 through n, each with weight w_i and value v_i, and a maximum weight W, the goal for the 0-1 knapsack problem is to

maximize

$$\sum_{i=1}^{n} v_i x_i$$

subject to

$$\sum_{i=1}^{n} w_i x_i \leq W \tag{3.1}$$

where $x_i \in \{0, 1\}$ denotes how many times item i is selected.

There are many variations of the knapsack problem, but the two most common variations are the bounded knapsack problem and the unbounded knapsack problem. The bounded knapsack problem introduces an upper limit to the number of times an item can be selected for the knapsack, and the unbounded version removes this limit, allowing for any item to be selected any number of times. The knapsack problem and its variations generally appear in resource management when trying to optimize value and minimize waste. A few real world examples of the knapsack problem include trying to select companies for an investment portfolio, fitting the maximum number of packages onto a cargo container, or finding the least wasteful way to cut raw materials.

It should be noted that although the decision version of the knapsack problem (determining if a value V can be achieved without exceeding a weight W) is NP-complete, the optimization version of this problem is NP-hard, and thus there is no known algorithm that can find an optimal solution in polynomial time. The knapsack problem has been studied for over a century, and many researchers have attempted to develop algorithms to solve it efficiently. The most well-known algorithm to solve the knapsack problem uses dynamic programming [16], which has pseudo-polynomial

complexity. Other popular avenues of research involving the knapsack problem include developing approximation algorithms to efficiently find solutions that are close to optimal. A greedy approach can be taken by calculating the value-to-weight ratio for all items, and sorting them in descending order. After sorting the items, add the item with the highest ratio to the knapsack, and continually add the items with the highest ratios until no more items can be added. The pseudocode for this greedy approach can be found in Algorithm 1. While this approach may not always be optimal, it allows for an approximate solution to be found with much less computational complexity.

Algorithm 1 KnapsackGreedyAlg($I = \{i_1, i_2, \ldots, i_n\}$)

Input: A list of items I, each with weight w_i and value v_i, and knapsack capacity W
Output: A list K containing the items in the knapsack

```
K ← ∅
for item i in I do
    i.ratio ← i.value/i.weight          ▷ Calculate the value to weight ratio
end
Sort items in I by value to weight ratio
totalWeight ← 0
for item i in I do
    if totalWeight + i.weight < W then
        totalWeight ← totalWeight + i.weight
        Add i to K
    end
end
Return K
```

Chapter 4

METHODS

Now that we've introduced daily fantasy basketball and some of the relevant work surrounding it, in this chapter we describe the methods we developed to optimize daily fantasy basketball. We first explain how we created the dataset used in our experiments. Afterwards, we describe the different techniques employed to model daily fantasy basketball, which were broken down into two parts: predicting player performances and generating lineups. For any given day, a player's performance would be predicted using a heuristic, and using these predicted scores, lineups were assembled and compared to both a naive baseline and a near optimal solution. By combining the different heuristics with different lineup creation techniques, a wide range of combinations were tested and evaluated.

4.1 Creating the Dataset

The dataset used in our experiments was sourced from public data and collected using BeautifulSoup[1], a Python web scraping library. It consists of two parts: basketball statistics and fantasy sports information. The basketball statistics were collected from Basketball Reference[2], a database containing historical information on NBA teams, players, and games dating all the way back to the inception of the NBA. We used box scores from the 2019-2020 and 2020-2021 NBA seasons, though the web scraping script could easily be adjusted to gather information from other seasons as well. The box scores contain information on a single game about each player's performance. A

breakdown of the box score features collected can be found in the appendix in Table A.1.

The fantasy sports information was sourced from RotoGuru[3], a blog that aggregates daily fantasy sports information across different sports and platforms to assist individuals in creating daily fantasy lineups. The information consists mainly of player salaries and fantasy point production. RotoGuru contains daily fantasy sport information across many sports and platforms, but for our purposes we chose to use the salary and fantasy point information from DraftKings. Players' salaries are assigned by DraftKings, and are usually proportional to the number of fantasy points they are projected to score. Within our dataset, costs range from a minimum of $3000 to a maximum of $13100. A breakdown of the fantasy sports information collected from RotoGuru can be seen in Table 4.1.

Table 4.1: Daily Fantasy Basketball Features

Abbreviation	**Definition**
Date	The date that this game occured
Team	The team this player plays for
Name	The name of this player
Starter	Whether or not this player was a part of the starting lineup for this game
Position	The position(s) this player is eligible for
Salary	The cost assigned by DraftKings to roster this player
FPTS	Fantasy Points - The total number of fantasy points scored by this player during this game

After collecting the data, it was cleaned by fixing naming discrepancies (matching punctuation within names, inconsistent spelling, special characters), and then combined. Overall, the dataset contains information on about 2,100 games, 600 players, and 43,000 total observations with 27 features.

[1]https://beautiful-soup-4.readthedocs.io/en/latest

[2]https://www.basketball-reference.com

[3]http://rotoguru.net

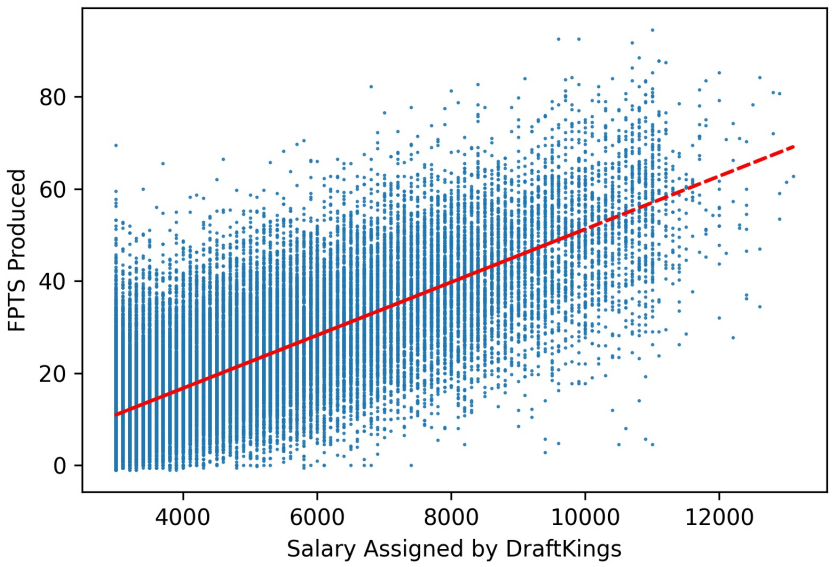
4.2 Predicting Player Performance

In this section, we conduct some exploratory data analysis and explain how we developed the heuristics used to predict player performance. These heuristics were then evaluated by using them to make predictions over the 2020-21 NBA season, and comparing these predictions to players actual fantasy point production.

4.2.1 Predictions based off Salary

Because one of the main constraints when constructing a fantasy basketball lineup is salary, we wanted to establish how accurately and fairly players were priced. The exact criteria by which DraftKings determines player salaries are unknown, but when looking at player salaries versus fantasy point production across the entire dataset in Figure 4.1, a general trend is observed. A player's salary and the amount of fantasy points they produce on any given day tend to follow a linear relationship.

Figure 4.1: Salary vs. FPTS Across the Entire Dataset



Knowing this we then wanted to see how consistently players performed compared to their salaries, and how reliably salary alone could be used as a predictor for the future. To do so, we developed a heuristic as follows: To predict how many fantasy points a player would produce on any given day, we would consider all the players who played the same position in the days prior, and perform linear regression using all prior players' salary and fantasy point production. After calculating the coefficients representing the linear equation relating salary and fantasy points, we then projected the current player's salary onto this equation to predict how they'd perform.

This heuristic is somewhat naive, as it assumes that two players with the same salary who play the same position will perform exactly the same, but we found this step to be a necessary starting point in determining how accurately players were priced. Through repeating this process on all players across the entire dataset, we found that players tended to outperform their predicted performance according to salary 53% of the time, meaning that on average, players performed according to their salary. About half the time they would perform better than history would suggest based on salary, and the other half they performed worse than expected. This indicated that although players on DraftKings were priced somewhat fairly, salary alone was not enough information to determine how a player would perform.

4.2.2 Predictions Using Rolling Averages

Another question we had while developing a heuristic that could accurately predict player performance was how much past information should be considered when trying to predict the future, i.e., how many of a player's previous games should be considered when trying to predict their future performance. We explored this idea by creating a simple heuristic that used a player's rolling averages to predict their performance. For each of the features considered when calculating a player's fantasy point score,

we assume that the player will produce the average of that feature over the last n games. For example, if $n = 3$ and player A recorded 5, 10, and 8 rebounds over his last three games, this heuristic would predict that player A will record 7.67 rebounds in his next game.

Referring back to Table 2.1, if $S_{i,j}$ refers to the ith statistic of the jth game (e.g., $S_{1,3}$ refers to points scored in the third game of the season, $S_{2,4}$ refers to the number of three point field goals made by a player in the 4th game of the season etc.) and W_i denotes the ith weight (e.g. $W_1 = 1$, $W_2 = 0.5$ etc), then a player's predicted fantasy points scored for their Kth game of the season based off their last n games would be:

$$\widehat{FPTS}_{K,n} = \frac{1}{n} \sum_{j=K-n}^{K-1} \sum_{i=1}^{9} W_i S_{i,j}$$

To evaluate how many games should be considered when trying to predict a player's future fantasy point performance, this heuristic was used over the 2019-20 and 2020-21 NBA seasons to predict player performances using their average over the last $n = 3, 5, 7, 10$, and all previous games. These predictions were then compared to their actual performances, and the root mean square error and mean absolute errors can be seen in Table 4.2.

Number of Games	**MAE**	**RMSE**
3	8.10	10.48
5	7.85	10.14
7	7.81	10.05
10	7.76	9.97
n	7.74	9.94

Table 4.2: Error Rates for Rolling Averages

From these values, we see that when trying to predict future performances, considering a greater amount of previous games allows for more accurate predictions to be made, but the return in accuracy decreases as more games are included. Using a player's

rolling averages to predict their performance is also extra unreliable towards the beginning of the season, as information is sparse and variance in performance is high as teams and players are still experimenting with game plans and strategies.

4.2.3 Predictions Using Ridge Regression

Building off of this, we wanted to develop a heuristic that took a less naive approach and incorporated machine learning into predicting how a player would perform, which ideally would perform better than strictly using rolling averages. Similar to Barry (2016)[11], we decided to use ridge regression to predict player performance. Ridge regression is an extension of linear regression that reduces overfitting by introducing a penalizing term to the standard least squares cost function used in linear regression. It is especially useful in situations where multicollinearity exists in a dataset. Multicollinearity exists in a dataset when one or several independent variables can be linearly predicted from other variables with a reasonable degree of accuracy. In our use case, several features can be loosely predicted from the other features within our dataset. For example, players that typically shoot many field goals and free throws and play a significant number of minutes will also typically log a few turnovers, as a good portion of a team's offense is run through them and mistakes are bound to happen over the course of a game.

Specifically, the goal of ridge regression is to select β to minimize

$$\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{M}\beta_j x_{ij})^2 + \alpha \sum_{j=1}^{M}{\beta_j}^2$$

where N is the number of observations, M is the number of features, β is the vector representing the weights, and α is the penalty factor. Our model was fitted using the 2019-20 NBA season rolling averages and fantasy points scored, and 10-fold cross-

validation was used to find the best alpha value. For a more in-depth explanation of ridge regression and its applications, please refer to McDonald (2009) [17].

After fitting the model, we then used it to make predictions on the 2020-21 NBA season. Instead of using DraftKings' scoring criteria as the coefficients for our predictor, we used the weights from the fitted ridge regression model. The heuristic developed using ridge regressions to predict a player's fantasy point production for their nth game is as follows:

$$\widehat{FPTS}_n = \beta_0 + \frac{1}{(n-1)} \sum_{i=1}^{9} \sum_{j=1}^{n-1} \beta_i S_{ij}$$

where $S_{i,j}$ again refers to the ith statistic of the jth game, and β represents the intercept and coefficients of the nine features used.

4.3 Creating Lineups

In this section, we discuss the different attempts made and techniques used to create competitive daily fantasy basketball lineups. Although the true player performances and fantasy points are known within the dataset, creating the best lineup for any given day is a non-trivial task. Brute force approaches can be applied to find the optimal lineups, but doing so can take several hours, if not days, in the worst case scenarios. Recall that to create a lineup, one must select 8 players that fit the positional and salary constraints. For any position, there can be dozens of available players to choose from, leading to billions of possible combinations that need to be checked. For our worst case, the day within our dataset containing the most players, there were a total of 289 available players, and 69 quadrillion total lineup combinations between those players. With our computing resources, we were able to create, validate, and score about 500,000-600,000 lineups per second, but even then it was impossible to conduct

a complete search on days with many players. As a compromise, in this section, we discuss the different approaches taken to approximate optimal solutions.

4.3.1 Depth First Search

The first brute force approach attempted involved using a depth-first search to construct a lineup, and checking the constraints once all the players had been selected. We found this approach to be too unreliable, as it would spend significant amounts of time exploring branches in the tree that failed to satisfy the position or salary constraint. For example, if players 1-4 used up enough of the salary cap such that there wasn't enough remaining salary to complete the roster, then a valid lineup could never be created in this branch of the tree. Attempts to improve this search were made by first sorting the players in each position from most to least expensive, and checking the salary and duplicate player constraint mid search, ending the search and returning to the root node if any of these constraints were not satisfied. Through these improvements, the depth-first search was able to terminate on certain days, but still struggled for the majority of the days in our dataset, finishing on only 66 out of the 350 total days. Despite the early search termination, not enough lineups were excluded to make brute force attempts feasible.

4.3.2 Pareto Front

Through the challenges faced in our initial depth-first search brute force attempt, we learned that the search space was too large to brute force the optimal lineup within a reasonable amount of time. We also noticed that on any given day in the dataset, there were often many redundant players that would never be considered in an optimal lineup. A player who was expected to score 15 fantasy points and

cost \$4,000 would be dominated by another player in the same position who was expected to score 30 fantasy points and cost \$3,000, assuming that the heuristic used to estimate its performance is accurate.

Generally, a player B dominates player A with respect to a heuristic H if player B plays in the same position as player A, is ranked higher by H than player A, and has a lower salary than player A. In the field of multi-objective optimization, the set of non-dominated solutions is also known as the Pareto front. By reducing the set of players to only include those who had the highest heuristic value relative to their salary, we were able to reduce the search to the point where a brute force approach would be feasible, while still creating lineups that would be near optimal given the heuristic. Within our dataset, there were also a few exceptionally busy days where the Pareto front still contained too many players for a solution to be found in a reasonable amount of time. On these days, the player pool for that position is further reduced by randomly selecting a maximum 14 players from the Pareto front for each position. We found 14 to be the sweet spot in terms of the number of players to consider, as it allowed for wide coverage of the Pareto front while still allowing for a close to optimal solution to be found in a reasonable time frame.

The pseudocode for our Pareto front algorithm can be seen in Algorithm 2. To find the Pareto front within a set of players who play the same position, they first must be sorted in ascending order by salary and decreasing order by heuristic value. Afterwards, iterate through the sorted players, keeping track of the highest score thus far. If a player scores more than the current best score, they are added to the Pareto front, and the best score is updated. After iterating through the players, return the players that were on the Pareto front.

Algorithm 2 ParetoFilter($L = \{p_1, p_2, \ldots, p_n\}$)

Input: A list L containing players $\{p_1, p_2, \ldots, p_n\}$, sorted in ascending order by salary and descending order by heuristic value H
Output: A list F containing the players on the Pareto front of L

```
F ← ∅
currentBestScore ← 0
for player p_i in L do
    if p_i[H] > currentBestScore then
        currentBestScore ← p_i[H]
        F ← F ⋃ p_i
    end
end
Return F
```

4.3.3 Multi-level Pareto Front

Through this process we realized that there were edge cases on certain days where, although the total number of players for a given position included dozens of players, due to how players performed, the Pareto front would only contain a few players. Since players can play multiple positions (e.g., a PG can also play as a G or as a Util), if a player X in a Pareto front was selected as, say a PG, players in G that were dominated by X should not be excluded from consideration. This makes the previous algorithm disconsider some potentially optimal lineups while also making it impossible to generate lineups for certain days if too few players were in a Pareto front.

To increase the coverage of the player pool while ensuring that new players added were also somewhat efficient themselves, we also developed an algorithm that extended our original Pareto front algorithm. The pseudocode for this algorithm can be seen in Algorithm 3. Inspired by NSGA-II [15], it works similarly to Algorithm 2, but after each pass through the players, will calculate a new level of the Pareto front without regard for players who were already included in higher Pareto front levels, repeating this process a given number of times or until a satisfactory number of players are

found. If the final level contains too many players, players on the last Pareto front are randomly selected. Considering players on multiple Pareto fronts provided increased flexibility when creating lineups, especially on less busy days or in cases where the number of players on the initial Pareto front was significantly smaller than the total number of available players.

Algorithm 3 MultiLevelParetoFilter($L = \{p_1, p_2, \ldots, p_n\}$, n_levels, $n\_players$)

Input: A list L containing players $\{p_1, p_2, \ldots, p_n\}$, sorted in ascending order by salary and descending order by fantasy points, an integer $n\_players$, and an integer n_levels
Output: A list F containing up to $n\_players$ on the first n_levels Pareto fronts of L

```
F ← ∅
level ← 0
while |F| < n_players and level < n_levels do
    ParetoFront ← ParetoFilter(L)
    if |ParetoFront| + |F| > n_players then
        tie breaking, randomly select players from ParetoFront to complete F
    end
    F ← F ⋃ ParetoFront
    level ← level + 1
end
Return F
```

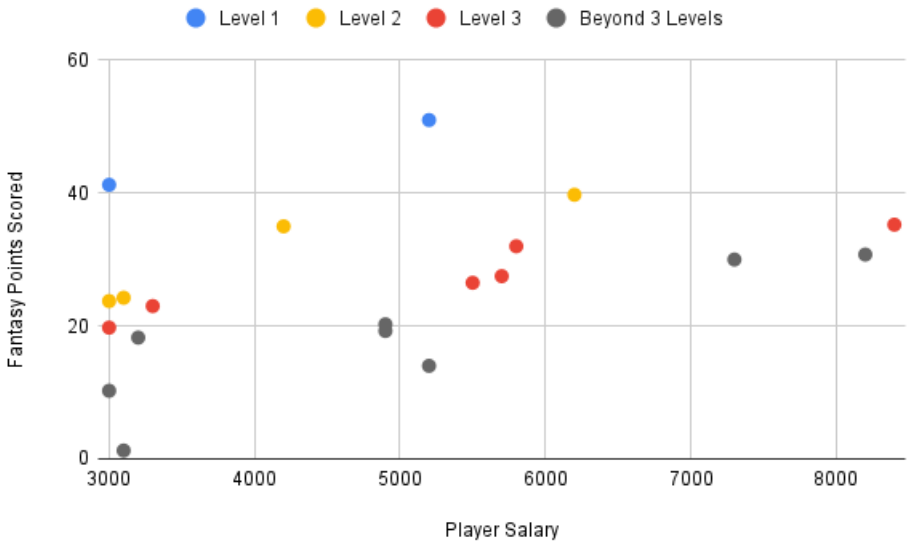
An example of this can be seen in Table 4.3, where the first level is highlighted in blue, the second level in yellow, and the third in red. This can also be observed visually in Figure 4.2, where to be considered on the first Pareto front level, there must be no other players who were both cheaper in salary and scored higher with respect to a heuristic (i.e., to the left of and above a point on the graph). On days where extreme performance outliers occurred, a single Pareto front would typically reduce the player pool far too drastically, limiting the number of players for a given position and reducing flexibility when trying to form a complete lineup. As seen by this example, the vast majority of efficient players can be found by considering only a few levels of the Pareto front.

Name	Salary	FPTS
Sterling Brown	3000	41.25
Devin Vassell	3000	23.75
Malik Monk	3000	19.75
Mason Jones	3000	10.25
Max Strus	3100	24.25
Cody Martin	3100	1.25
David Nwaba	3200	18.25
Caleb Martin	3300	23
Gabe Vincent	4200	35
Norman Powell	4900	20.25
Justin Holiday	4900	19.25
Shake Milton	5200	51
Patty Mills	5200	14
Duncan Robinson	5500	26.5
Danny Green	5700	27.5
Will Barton	5800	32
Terry Rozier	6200	39.75
Tyler Herro	7300	30
Fred VanVleet	8200	30.75
CJ McCollum	8400	35.25

Table 4.3: Multi Level Pareto Front Example

After experimenting with parameters, we settled on using a maximum of 3 levels and 14 players per position. This amounts to a total of about 1.5 billion lineup combinations, which on average takes around 33 minutes to brute force. In Figures 4.3 and 4.4 are histograms for the utility position over the 2020-21 NBA season, which show the average number of players per level on the Pareto front, and the total number of days where it took n Pareto front levels to reach 14 players. For the vast majority of days, it only requires going 2 to 3 levels deep to find the fourteen most efficient players.

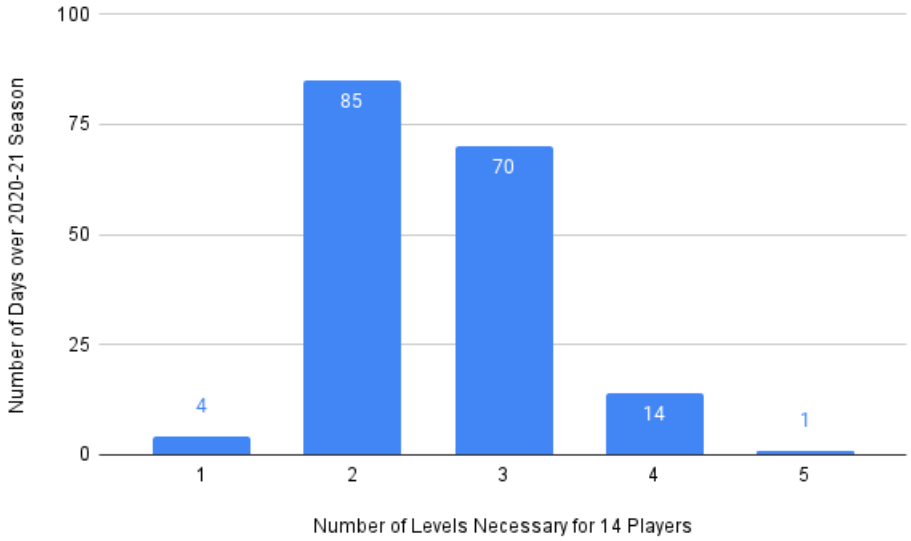
Figure 4.2: Multi Level Pareto Front, Visualized



4.3.4 Genetic Algorithms

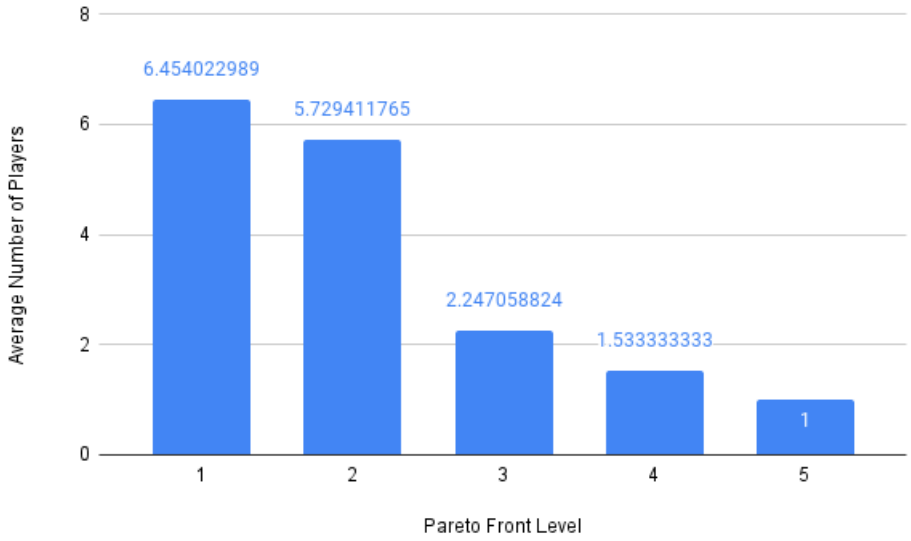
To navigate around the computational complexity required to brute force a solution, we also used genetic algorithms to create fantasy basketball lineups. Genetic algorithms are inspired by survival of the fittest, and generally include four main steps: initialization, selection, mutation, and crossover. Genetic algorithms begin by first initializing a random population. The size of this population can change depending on the nature of the problem, but it should ideally cover a wide area of the search space. After initialization, the population is sorted by fitness, and the best individuals are selected to breed the next generation. The remainder of the population is comprised of new individuals who have gone through genetic operators from the selected elites. There are many genetic operators, but the two main ones are mutation and crossover. Mutation involves randomly altering small portions of an individual, whereas crossover involves creating a new individual from a combination of two parent individuals. By repeating these steps over many iterations, the populations overall fitness improves as weaker members are replaced with variations of stronger ones.

Figure 4.3: Number of Levels Needed for 14 Players Per Position



The genetic algorithm we developed works as follows: For each day of the dataset, the genetic algorithm used a population of fifty random valid lineups. This initial population was then sorted by fitness (fantasy points scored) in descending order and separated into two groups. The bottom twenty five lineups were replaced with mutations or crossovers of a randomly selected top twenty five lineup, making sure to check that these lineups fit the constraints. If during the mutation or crossover process a newly created lineup breaks the constraints, the changes made to that individual are reverted and the mutations or crossover process is attempted again, up to ten times. If after ten attempts a valid lineup is still unable to be created, that individual is discarded, and new parents are chosen to mutate or crossover from. This process was repeated a given number of times until competitive lineups were formed. After conducting this experiment over varying parameters, the parameters that seemed to have the most success were a population size of 50 lineups, a mutation rate of 0.25, a crossover rate of 0.5, run for 50 generations. Using this genetic algorithm to create lineups for all the days led to an average score of 360.22 fantasy points.

Figure 4.4: Average Number of Players Per Pareto Front Level



Repeating this experiment did not always lead to the same lineups due to the inherent randomness in genetic algorithms, but the final lineups were comparable in score to the Pareto-front brute force approach for nearly all the days. When comparing the Pareto front solution to the genetic algorithm solution, we noticed that the genetic algorithm would sometimes outperform the Pareto front approach. This raised some concern, as theoretically the brute force solution should have been our near optimal solution. To double check our approach, the players in the genetic algorithm solution that were not a part of the Pareto front brute force approach were artificially inserted into the Pareto front. We reconducted the experiment, and were able to confirm that the brute force approach is able to outperform or match the genetic algorithm solution when the exclusive players were artificially inserted. It's possible due to unforeseen edge cases or randomness when selecting players from the Pareto front on busier days that the genetic algorithm was able to outperform our Pareto front approach. Nevertheless, using genetic algorithms provided similar performance to the Pareto front brute force approaches while also being less computationally expensive, performing only 16 points worse on average.

Algorithm 4 GENETIC ALGORITHM(μ, λ, G, m, c)

Input: The number of elites μ, the number of offspring λ, the number of generations G, the mutation rate m, and the crossover rate c
Output: The population P after G generations

```
P ←new population with μ + λ random valid lineups
for G generations do
    G ← Evaluate(P)          ▷ Calculates and sorts lineups by fantasy points scored
    G' ← ∅
    newLineups ← ∅
    for i = 1 to μ do
        Add G_i to G'
    end
    for i = 1 to λ do
        newLineup ← A randomly selected lineup from G'
        newLineup ← Mutate(newLineup, m)
        if random < c then
            parent2 ← a randomly selected lineup from G'
            newLineup ← Crossover(newLineup, parent2)
        end
        newLineups ← newLineups ⋃ newLineup
    end
    P ← G' ⋃ newLineups
end
Return P
```

4.3.5 Greedy Algorithm

After establishing our naive random baseline and high-end baselines, we also wanted to develop a greedy algorithm to see how a simpler approach would compare to the genetic algorithm and brute force approaches. Recall that in Chapter 2, we introduced the knapsack problem and presented a greedy algorithm to solve it. By sorting the items by their ratio of value to weight, a solution can be found by greedily selecting items until the knapsack is full. Similarly, a greedy approach for lineup creation can be taken after observing that the problem of creating fantasy basketball lineups can be viewed as an extension of the knapsack problem, with a few extra conditions.

Firstly, the knapsack is separated into eight bins, where each bin represents a position in a fantasy basketball lineup. Each individual item can only be placed in specific bins, and each bin must be filled with exactly one item. All the bins must be filled without repeat items, and their total weight (cost) must be less than or equal to a certain threshold. By reframing our problem through the lens of the knapsack problem, we also developed a greedy algorithm to create lineups.

Our algorithm works by creating two separate lists. The first list contains the players sorted by their heuristic value in descending order, and the second list contains the same players sorted by salary in ascending order. Lineups are formed by iterating through the best performing players and assigning them to the most specific position available, but only if adding this player still allows for a valid lineup to be created. Position specificity is tiered. The top tier includes the starting five positions: PG, SG, SF, PF, C. In the next tier are the combined positions, G which consists of PGs and SGs, and F which consists of SFs and PFs. Lastly, the final tier is the UTIL position which includes all players. Potential lineups are then validated by iterating through the cheapest players, and assigning them to the most specific remaining position when possible, filling out the rest of the lineup. If no lineup can be created this way, then the current candidate player does not fit, and these steps are repeated with the next candidate player. This greedy approach with slight look-ahead ensures that at least one valid lineup can be formed with the currently proposed players. The pseudocode for this algorithm can be seen in Algorithm 5.

Algorithm 5 LineupGreedyAlg(P, C, b)

Input: P, a list of candidate players, C, a list of the cheapest players, and budget b
Output: G, A valid greedy lineup

```
G ← ∅
vacancies ← {PG, SG, SF, PF, C, G, F, UTIL}
for player p in C do
    if length(vacancies) == 0 then
        return G
    end
    potentialLineup ← G
    assign p to the most specific position in vacancies
    add p to potentialLineup
    if checkViability(potentialLineup, vacancies, C, b) then
        add p to G
        remove p.position from vacancies
        budget ← budget − player.salary
    end
end
return G
```

Chapter 5

RESULTS & DISCUSSION

To test and evaluate the methods discussed in Chapter 3, each heuristic (Season Averages, Salary Linear Regression, Ridge Regression) was used in combination with each different lineup search method (Greedy, Pareto Front, Multi-level Pareto Front, Genetic Algorithm), leading to a total of 12 combinations. In this section we discuss and analyze the results from these experiments, starting with heuristic evaluation, and then examining which combinations of heuristics and search methods produced the best performing lineups. Lastly, we discuss why we believe the salary linear regression heuristic along with the greedy lineup search method outperformed the other combinations.

5.1 Heuristics

Each heuristic was evaluated by making predictions over the 2020-21 NBA season. These predictions were then compared to players' actual fantasy point production using mean absolute error and root mean-squared error. These two metrics were chosen because we felt that their combination provided a good way of measuring prediction accuracy. Whereas mean absolute error measures the average magnitude of the prediction errors and gives the same weight to each prediction, root mean squared error measures the average prediction error squared and penalizes outliers more harshly. The equations for the mean absolute error and root mean-square error can be seen below.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\widehat{FPTS}_i - FPTS_i|$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\widehat{FPTS}_i - FPTS_i)^2}$$

The error rates for each heuristic can be seen in Table 5.1. Based off these values, we can see that no single heuristic consistently outperformed the others in both metrics, with the maximum and minimum values for each metric differing by only about 0.1 fantasy points. We believe that one of the reasons why the performance of the heuristics did not improve significantly despite including more features is that generally it is difficult to develop a heuristic that both generalizes well and is also capable of handling outliers.

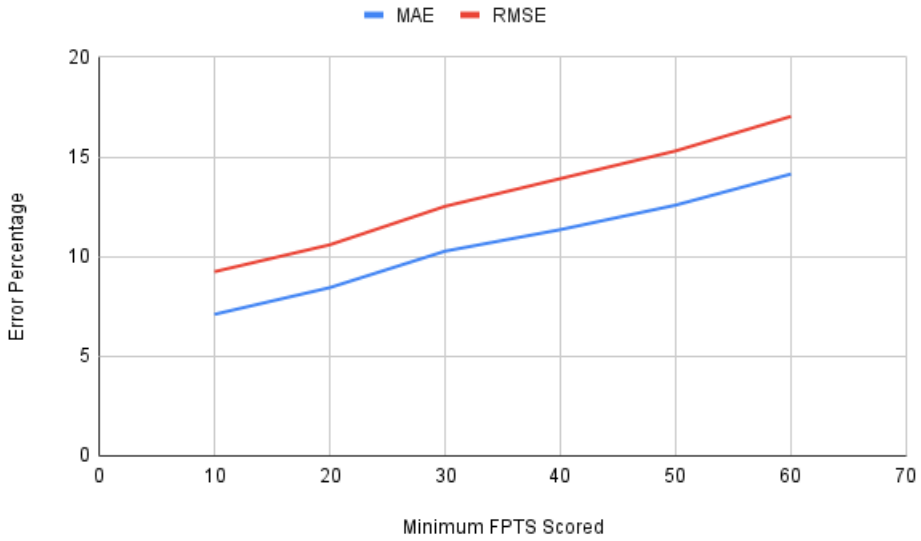
This can also be seen in Figure 5.1, which shows the ridge regression error rates measured against the minimum fantasy point performance to be included in the error calculations. As the minimum fantasy points required increased, the error rate with this heuristic also increased, further showing the challenges in trying to develop a heuristic that both generalizes well for the average player while also being able to predict outlier performances.

It's worth noting that although ideally more time would have been spent further developing and refining these heuristics, due to the inherent luck involved in basketball, it would be impossible to perfectly predict player fantasy performance, and increasingly challenging to improve these heuristics beyond a certain level of error. For reference, Barry (2016)[11] was unable to push beyond a mean absolute error of 6.6 and a root mean-squared error of 9, even after adding additional features to their ridge regression model.

Heuristic	MAE	RMSE
Season Rolling Averages	7.70	9.94
Linear Regression using Salary	7.76	9.83
Ridge Regression	7.73	9.87

Table 5.1: Heuristic MAE and RMSE

Figure 5.1: Ridge Regression Error Rate



5.2 Lineup Creation

To evaluate the performance of the heuristic and lineup creation combinations, for each day a heuristic was used to predict player performance, and using these predictions, lineups were created using the greedy algorithm, the genetic algorithm, single level Pareto front, and multi-level Pareto front brute force approaches. These lineups were then scored and compared to both a naive baseline and an oracle solution. The naive baseline was created for each day by taking the average score of 1,000 randomly generated valid lineups, and serves as a low-level baseline when comparing models. For the high-end baseline, oracle solutions were created by using the single level Pareto front, multi-level Pareto front, and genetic algorithm with players' true fantasy points scored. Because complete brute force approaches were infeasible, the

oracle solution is not provably optimal, but we felt that it was close enough to optimal when trying to compare models. For reference, South(2019)[13] considered 260 points competitive enough to be above average compared to other players, and the oracle solutions average scores are well above that threshold.

Lineup Search Approach[1,2]	**Avg. Prediction**	**Avg. FP**
Naive	n/a	167.6
Oracle Greedy	n/a	344.27
Oracle Single Level PF	n/a	372.59
Oracle Multi Level PF	n/a	377.43
Oracle Genetic Algorithm	n/a	359.79
Salary Linear Regression Greedy	235.23	255.76
Salary Linear Regression Single Level PF	237.05	242.59
Salary Linear Regression Multi Level PF	237.11	249.17
Salary Linear Regression GA	234.04	229.32
Season Rolling Averages Greedy	252.39	237.12
Season Rolling Averages Single Level PF	274.08	243.38
Season Rolling Averages Multi Level PF	274.26	243.43
Season Rolling Averages GA	262.98	241.22
Ridge Regression Greedy	251.95	236.59
Ridge Regression Single Level PF	270.63	245.6
Ridge Regression Multi Level PF	270.74	246.0
Ridge Regression GA	260.27	246.85

Table 5.2: Lineup Creation Results

Looking at the average fantasy points scored across the 2020-21 NBA season in Table 5.2, a few promising combinations can be seen. For all heuristics besides salary linear regression, the Pareto front and genetic algorithm approaches outperformed the greedy algorithm. The most accurate combination in terms of predicted score versus actual score was the salary linear regression and genetic algorithm model, which overestimated by 4.72 points on average. In addition, all heuristics besides salary linear regression tended to overestimate performance.

[1]PF - Pareto Front

[2]GA - Genetic Algorithm

Interestingly, for most cases the added benefit of including multiple Pareto front levels in the brute force search was marginal, and only increased the average predictions by fractions of a point. Though allowing for subsequent Pareto front levels did not yield significant improvements to the average prediction, it's possible that including more Pareto front levels and players per position (which requires more computational power) could improve this.

The combination that yielded the highest average score was the salary linear regression combined with the greedy algorithm. Upon first glance, this seemed somewhat unexpected, but with further analysis we can understand why this happened. Recall that DraftKings assigns salaries to players based on how they are expected to perform. Looking at Table 5.3, which displays the average expected fantasy point production [3] of a player for a given position, the most expensive players provide much more value per \$1,000 of salary than the both the cheapest and average player. Similarly, recall that the linear regression heuristic is based solely on salary, and that given player A and player B play the same position, it will always predict that the more expensive player will perform better. This idea is further emphasized by the greedy algorithm, which will select the player with the highest heuristic value, given that a valid lineup can be created. Because the greedy lineup approach selects the players with the highest heuristic value, which in this case was tied to player salary, it selected a few stars and filled out the rest of the lineup with average to cheaper players. This suggests that when selecting a lineup, it's better to rely on a few star players while balancing out the rest of the roster, than selecting a lineup filled with players that are expected to have average performances.

Because none of these models were able to consistently surpass 260 fantasy points, they probably wouldn't fare well as a standalone solution for daily fantasy basketball,

[4]Expected Fantasy Points - Fantasy Points / Salary $\times$ 1,000

Position	All Players	Bottom 20%	Top 20%
PG	4.31	2.98	4.92
SG	4.13	2.68	4.79
SF	4.12	2.61	4.70
PF	4.07	2.60	4.74
C	4.32	3.01	4.84
G	4.20	2.69	4.91
F	4.09	2.59	4.75
UTIL	4.17	2.64	4.86

Table 5.3: Expected Fantasy Points per $1k Salary, Cheapest vs. Most Expensive Players

although they could be used to assist players when trying to predict player performance and create lineups. It's also worth noting that when evaluating lineups, while drafting the best players generally results in a good team, an outstanding team is needed to consistently perform well in a tiered payout system. In these competitions, about half of the total prize money is given to the top-10 players, and while these lineups generally will include some consistent star players, they also more likely than not will include some high-risk players that happened to perform well on that night. For our purposes, we did not incorporate risk when creating lineups, but rather tried to minimize the error between our predicted and actual fantasy point production. In addition, the more other players in the same lobby select the same players as our model does for their lineup, then the general advantage one would have over others using these methods would be diminished.

Chapter 6

FUTURE WORK & CONCLUSION

Optimizing daily fantasy basketball remains largely unexplored in academia, and is a challenging problem due to the inherent variance and luck involved in professional sports. In this thesis, we designed and developed a system to assist individuals participating in daily fantasy basketball.

Our contributions include:

- Creating a pipeline that generates datasets by extracting basketball box score and daily fantasy sports information
- Developing several heuristics (Season Average, Salary Linear Regession, Ridge Regression) to predict player performance
- Developing several lineup creation methods (Greedy, Pareto Front, Multi-level Pareto Front, Genetic Algorithm)

Through the optimizations made to our lineup creation methods, we were able to create near-optimal lineups by reducing the search space to only contain players on the Pareto front between salary and heuristic value. We were also able to create lineups with comparable performance in a much shorter amount of time through evolutionary algorithms. Of the twelve heuristic and lineup creation combinations, the one that showed the most promise was the salary linear regression heuristic paired with the greedy algorithm lineup creation, which scored 255.76 fantasy points on average. Although this didn't surpass the 260 point threshold to be considered competitive,

we believe that our findings serve as a good baseline as a tool that can be used to assist users in making these decisions.

There are still several open areas one could explore if they wished to expand upon the research done in this paper. In regards to predicting player performance, more time could be spent developing more robust and accurate heuristics. This could be done by incorporating more features such as the number of days a player has rested before the current game, if the current game is being played at home versus away, the defensive metrics of the opposing team, or other advanced statistical metrics. Also, additional exploratory data analysis could be conducted to determine which of these features are more important than others when predicting player fantasy point production. Another possibility could include incorporating some other machine learning techniques to predict player performances, or developing different heuristics for each position, as certain methods may be more effective for certain positions rather others.

For lineup creation, next steps of future research could include creating multiple lineups with varying risk levels, and evaluating them either through simulations or real world competitions. Unfortunately, due to the timing during which our research took place, we did not have the chance to test our methods in these contexts. Another way to increase the variance of the lineups generated would be to consider the percentage of other users who also own a candidate player, as the general advantage one user has over others is diminished if this candidate player is rostered unanimously. Another area of interest could be seeing how effective these methods of creating lineups would be in other sports, such as football, baseball, or hockey, just to name a few.

BIBLIOGRAPHY

[1] B. Adgate, "Sports Betting Is Revving Up Ad Spending For Fourth Quarter," *Forbes*, Sep 2021. [Online]. Available: https://www.forbes.com/sites/bradadgate/2022/09/15/sports-betting-is-revving-up-ad-spending-for-fourth-quarter

[2] Wire Act. 18 U.S.C. § 1084. 1961.

[3] Prohibition on Funding of Unlawful Internet Gambling. 31 U.S.C. § 5361. 2006.

[4] B. A. Evans, A. Hornby, J. D. Pitts, and J. Roush, "Evidence of Skill and Strategy in Daily Fantasy Basketball," *Journal of Gambling Studies*, vol. 34, no. 3, pp. 757–771, Sep 2018. [Online]. Available: https://doi.org/10.1007/s10899-018-9766-y

[5] "NBA Official Rules 2020-21," *NBA*, Oct 2021. [Online]. Available: https://ak-static.cms.nba.com/wp-content/uploads/sites/4/2021/11/2021-22-NBA-Rule-Book.pdf

[6] L. Donald, "Basketball," *Encyclopedia Britannica*, Jan 2023. [Online]. Available: https://www.britannica.com/sports/basketball

[7] T. Reynolds, "NBA revenue hits a record $10 billion," *Chicago Sun-Times*, Jul 2022. [Online]. Available: https://chicago.suntimes.com/2022/7/13/23206931/nba-nets-more-than-10-billion-in-revenue

[8] J. Pompliano, "The NBA's $10 Billion Business," Huddle Up, blog, Oct. 22, 2022. [Online]. Available: https://huddleup.substack.com/p/the-nbas-10-billion-business

[9] E. Dixon, "Study: Fantasy sports market to grow 9.5% to US$22.3bn in 2021," *SportsPro Media*, Sep 2021. [Online]. Available: https://www.sportspromedia.com/news/fantasy-sports-global-market-value-2021-nfl-mlb-nba/

[10] "DraftKings, FanDuel become NBA's co-official sports betting partners," *NBA*, Nov 2021. [Online]. Available: https://www.nba.com/news/draftkings-fanduel-become-nbas-co-official-sports-betting-partners

[11] C. Barry, N. Canova, and K. Capiz, "Beating DraftKings at Daily Fantasy Sports," 2016. [Online]. Available: https://web.stanford.edu/class/stats50/projects16/BarryCanovaCapiz-paper.pdf

[12] E. Hermann and A. Ntoso, "Machine Learning Applications in Fantasy Basketball," 2015. [Online]. Available: https://cs229.stanford.edu/proj2015/104_report.pdf

[13] C. South, R. Elmore, A. Clarage, R. Sickorez, and J. Cao, "A Starting Point for Navigating the World of Daily Fantasy Basketball," *The American Statistician*, vol. 73, no. 2, pp. 179–185, Apr. 2019. [Online]. Available: https://doi.org/10.1080/00031305.2017.1401559

[14] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[15] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[16] R. Andonov, V. Poirriez, and S. Rajopadhye, "Unbounded knapsack problem: Dynamic programming revisited," *European Journal of Operational*

Research, vol. 123, no. 2, pp. 394–407, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221799002659

[17] G. C. McDonald, "Ridge regression," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 1, pp. 93–100, 2009.

[18] A. Winschel, "Daily Fantasy Basketball Lineup Optimizer," *Chair's Message*.

APPENDICES

Appendix A

DATASET

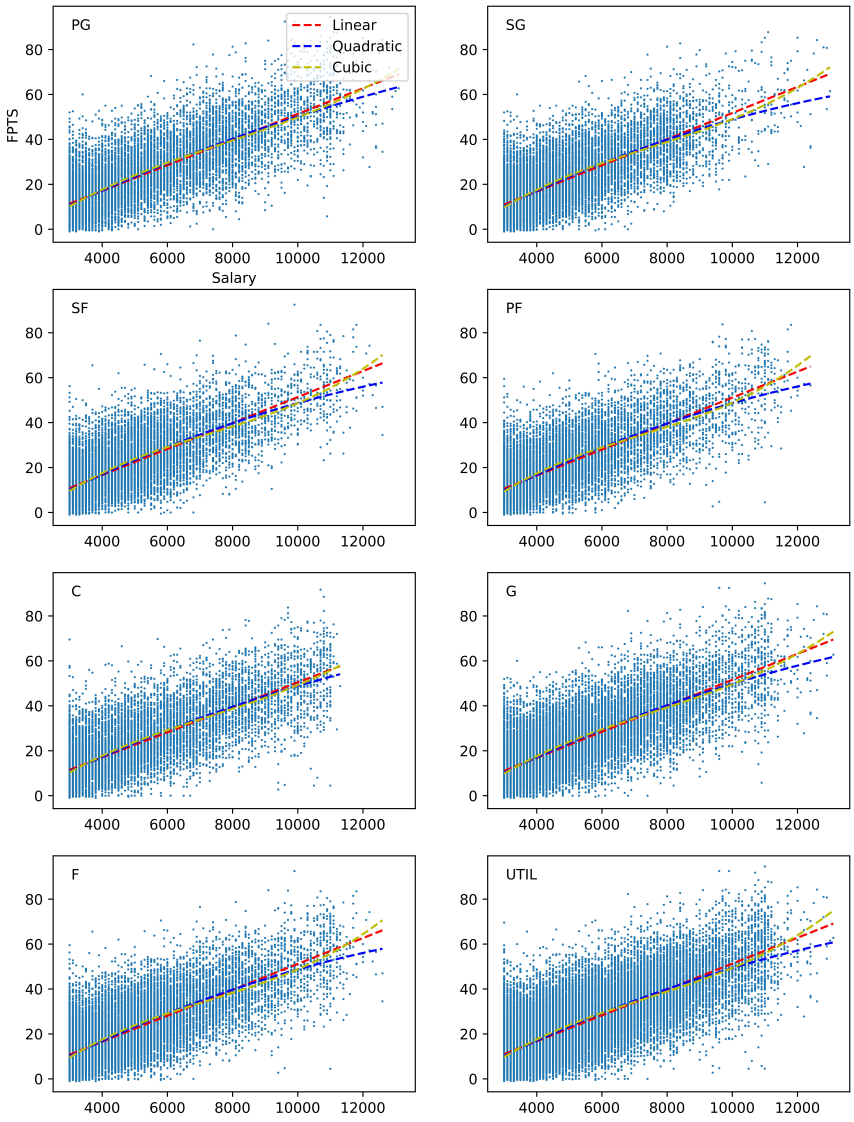
Table A.1: Basketball Reference Box Score Features

Abbreviation	Definition
Date	The date that this game occured
Team	The team this player plays for
Name	The name of this player
MP	Minutes Played - The number of minutes played by a player
FG	Field Goals - The Number of field goals a player has made, includes both 2 pointers and 3 pointers
FGA	Field Goal Attempts - The number of field goals a player has attempted, includes both 2 pointers and 3 pointers
FG%	Field Goal Percentage - The percentage of field goal attempts that a player makes (FG/FGA)
3P	3 Point Field Goals Made - The number of 3 point field goals a player has made
3PA	3 Point Field Goals Attempted - The number of 3 point field goals that a player attempted
3P%	3 Point Field Goal Percentage - The percentage of 3 point field goal attempts that a player makes (3P/3PA)
FT	Free Throws - The number of free throws that a player has made
FTA	Free Throws Attempted - The number of free throws that a player has attempted
FT%	Free Throw Percentage - The percentage of free throw attempts that a player makes (FT/FTA)
ORB	Offensive Rebounds - The number of rebounds a player has collected while they were on offense
DRB	Defensive Rebounds - The number of rebounds a player has collected while they were on defense
TRB	Rebounds - Occurs when a player recovers the ball after a missed shot. The number of total rebounds a player has collected (ORB + TRB)
AST	Assists - Occurs when a pass leads directly to a made basket. The number of assists by a player
STL	Steals - Occurs when a defensive player takes the ball from a player on offense, causing a turnover. The number of steals by a player
BLK	Blocks - Occurs when a defensive player tips the ball when an offensive player attempts a shot. The number of blocks by a player
TOV	Turnovers - Occurs when an offensive player loses the ball. The number of turnovers a player commits
PF	Personal Fouls - The number of personal fouls a player has committed
PTS	Points - The number of points a player has scored
+/-	Plus Minus - The point differential when a player is on the floor

Appendix B

SALARY VS FANTASY POINT TRENDLINES

Figure B.1: Salary vs. FPTS, by Position



Appendix C

HEURISTIC AND LINEUP CREATION BOX PLOTS

Figure C.1: Heuristic + Lineup Creation Box Plots