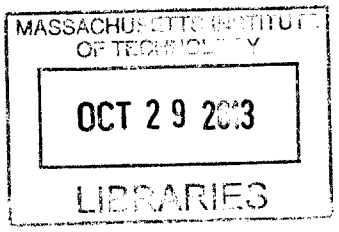
Modelling the NBA to Make Better Predictions

ARCHIVES



by

Keshav Puranmalka

B.S., Massachusetts Institute of Technology(2012)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2013

© Massachusetts Institute of Technology 2013. All rights reserved.

Author .....
Department of Electrical Engineering and Computer Science
August 23, 2013

Certified by .....
Prof. Leslie P. Kaelbling
Panasonic Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by .....
Prof. Albert R. Meyer
Chairman, Masters of Engineering Thesis Committee

Modelling the NBA to Make Better Predictions

by

Keshav Puranmalka

Submitted to the Department of Electrical Engineering and Computer Science
on August 23, 2013, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

Unexpected events often occur in the world of sports. In my thesis, I present work that models the NBA. My goal was to build a model of the NBA Machine Learning and other statistical tools in order to better make predictions and quantify unexpected events. In my thesis, I first review other quantitative models of the NBA. Second, I present novel features extracted from NBA play-by-play data that I use in building my predictive models. Third, I propose predictive models that use team-level statistics. In the team models, I show that team strength relations might not be transitive in these models. Fourth, I propose predictive models that use player-level statistics. In these player-level models, I demonstrate that taking the context of a play into account is important in making useful prediction. Finally, I analyze the effectiveness of the different models I created, and propose suggestions for future lines of inquiry.

Thesis Supervisor: Prof. Leslie P. Kaelbling
Title: Panasonic Professor of Computer Science and Engineering

Acknowledgments

I thank Professor Kaelbling for initially spurring my interest in computer science as a freshman in 6.01, in Machine Learning as a senior, and for providing invaluable guidance throughout my research.

I thank my parents for their unconditional love and unequivocal support throughout the years. Mom and Dad, you've instilled in me a strong appreciation for education, a desire to always strive for better, and to always pursue, above else, what is right. I am eternally grateful for all you have done for me; I would not be who I am today without your guidance.

I thank my brother, Raghav for being my closest friend. Raghav, I've always been able to talk to you when I needed a friend and turn to you in times of need. I've looked up to you my entire life; it is because you set such a high bar for excellence that I have managed to achieve all that I have.

I thank my taijii and tauji for being a second set of parents, for scolding me when parents would not, and supporting me when parents might not.

I thank Nikola Otasevic and Ameya Shroff for being my closest friends and adopted brothers. I thank Ameya for all the late nights of studying, for watching with me every movie and TV show known to man, and for all of the conversations and debates we've had over the years. Ameya, your friendship means more to me than I could possibly describe in words. I thank Nikola for providing valuable insights into NBA strategy during our "research" sessions. Nikola, you have provided much-needed perspective in my life on countless occasions and have often helped me realize what is truly important in life. Your idealism and pursuits to make a better human condition are something I will always admire.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Unexpected" events often occur in the world of sports: we frequently hear of "upsets" and of teams doing much better or worse than previously expected. For example, after acquiring Dwight Howard and Steve Nash during the off-season, the 2012-3 Los Angeles Lakers were expected to be one of the favorites to win the championship. However, they tied for the 10th best record in the league, and were swept in the first round of the playoffs. Even for those with domain expertise (NBA coaches, management, players, and analysts), it can become difficult to tell which events were actually unexpected, and which weren't. In addition to "unexpected" events occurring, many events that occur in sports are inherently random.

Because of this underlying uncertainty, it is useful to make make good predictions and be able to distinguish between events that are random (normal deviations from expected performance) and those which are truly unexpected (significant deviations from expected performance, perhaps due to uncontrollable factors such as injuries). In my thesis, I make many steps towards achieving this goal.

In section 1.1 of the introduction, I first formally define the problem that I will work on. Next, in section 1.2 of the introduction, I describe the types of data available.

In chapter 2 of my thesis, I summarize much of the previous work that has been done. In chapter 3, I describe novel features that I extract from play-by-play data that I will use in building predictive models. In chapter 4, I describe the models that I build to predict outcomes in games. Some of these models only focus on team-

level information in predicting outcomes. In these models, I show that team strength relationships might not be transitive. Other models integrate individual contributions to model the strength of a team. In these models, I show how different players' observed skill-sets suggest both synergistic and dis-synergistic effects. Furthermore, in these player models, I suggest the value of incorporating the context of when a play occurs. In chapter 4, I also analyze and compare the major results from my thesis, as well as suggest future lines of research in this problem.

1.1 Problem Statement and Evaluation Criteria

My goal in this thesis is to predict outcomes in the NBA. There are many different problems within this range that I could attack. For example, I could attempt to predict the number of wins a team will get in a season, or who will win the championship in a given year, or the winner of the MVP award. However, instead of focusing on those problems, I have decided to instead predict the outcome of an individual game.

The biggest reason I chose to focus on predicting the outcome of an individual game is because the number of times the other events happen is rare. For example, there is only one MVP award winner and one championship winner every year. There are only 30 teams whose total number of wins I could predict in a given year. On the other hand, there are 1230 games in the regular season of the NBA alone. With many more games to predict, not only can we build a better predictive model, but we can also gain more useful insights from the models that we build. Furthermore, it becomes easier to test the model's usefulness with many more events.

In predicting the games, for each game in the season and in the playoffs, I will output both the probability of either team winning, as well as who is more likely to win. Then, I will test the predictions using three different error functions:

1. **Number of wins** predicted correctly. I will add 1 for every win predicted correctly. The goal is to maximize this function.

2. **Sum of squared error**. When I predict a probability p of winning, I will add

$(1-p)^2$ to the total if the team won, and add p^2 otherwise. The goal is to minimize this function.

3. **FairScore fuction.** Over the last five years, the home team has won 60% of its game. Hence, it is easier to predict a home victory than an away victory. To compensate for this, the FairScore function will add .4 for every correct home win and .6 for every correct away win. The goal is to maximize this function.

1.2 Available Data

In this section, I describe the type of statistics that are publicly available for the NBA. In general, there are two categories of publicly available statistics:

1. Box score statistics
2. Play by play statistics

There is also a third type of data, visual tracking data, that teams have recently started collecting. However, only 15 of the 30 NBA teams collect visual processing data. Furthermore, the visual processing data is not available publicly, [11].

1.2.1 Box Score Statistics

In basketball, a box score contains information for one particular game between two teams; in particular, it contains the following statistics for each player on each team:

1. The number of minutes played in the game (MP)
2. The number of field goals made in the game (FG)
3. The number of field goals attempted in the game (FGA)
4. The number of 3-point field goals made in the game (3P)
5. The number of 3-point field goals attempted in the game (3PA)

6. The number of free throws made in the game (FT)
7. The number of free throws attempted in the game (FTA)
8. The number of offensive rebounds grabbed in the game (ORB)
9. The number of defensive rebounds grabbed in the game (DRB)
10. The number of assists in the game (AST)
11. The number of steals in the game (STL)
12. The number of blocks in the game (BLK)
13. The number of turnovers committed in the game (TOV)
14. The number of personal fouls commited in the game (PF)
15. The number of points scored in the game (PTS)

Additionally, the box score contains some team-level information in addition to the totals above. In particular, it indicates which 5 players started the game. They also sometimes include "Team Rebounds" – rebounds that were not allocated to a particular player – as well as "Team Steals" and "Team Turnovers" – again, when the individual statistics were not assigned to any individual players.

Box score data is publicly available on Basketball-reference since the 1985-6 NBA season. [1]. This gives us almost 30 years of seasonal data.

1.2.2 Play-by-play Statistics

Play by play statistics are an expansion of box score statistics, but are much more detailed. They provide the following information:

1. Times when player substitutions are made. This allows us to determine exactly which players were on the court at a particular time in the game.

2. Time information about the individual box score statistics. This will allow you to know not only how many (for example) assists a player had, but when he had them.

3. Counter-party in the individual box score statistics. For example, you will know which shot a player assisted, in addition to knowing when he assisted. Same information for fouls and steals.

4. The type of each field goal attempted – for example, a jumper, a dunk, a layup, etc.

5. Fouls *drawn*. Although fouls drawn can be estimated by the number of free throw attempts, the additional free throw on a foul after a made shot, non shooting fouls, and fouls on three point shots make this estimate imprecise.

6. The times when possession changes between teams.

Some play-by-play data also contains shot locations – an (x, y) tuple denoting the the coordinate of the location of the shot, relative to some fixed point on the court (usually the basket).

On the surface, it seems like the information available with play-by-play data is similar to box-score data, since mostly the same events are recorded. However, the play-by-play data also gives us context for when those events occurred. For example, since we know at every time which ten players are on the court, we might be better able to determine the defensive value certain sets of players add, as well as synergies between players of different skillsets which aren't available with box score data. For example, it is perhaps possible that one player attracts a lot of defensive attention, so his team-mates get open shots when he is on the floor. This information wouldn't necessarily be present in box score data, but could be more easily extracted from play-by-play data. In chapter 3 of my thesis, I talk more about some of the more novel features I've extracted from play-by-play data. In my thesis, I only use play-by-play data in my analysis.

1.2.3 Visual Tracking Data

In addition to keeping track of box-score statistics and play-by-play data, 15 of the 30 NBA teams (as of May 5, 2013) have recently started to keep track of visual tracking data. Visual tracking data contains the following information:

1. It contains the (x, y) location of every player on the court, relative to a fixed point (either the basket or the corner of the court). This information is updated 30 times a second.

2. It contains the (x, y, z) location of the ball on the court. This information is also updated 30 times a second.

With visual tracking data, you can do a lot more than you could with play-by-play or box score data. Because you can identify 30 times a second where every player is on the court, you could easily identify when a player misses a defensive assignment or is out of position offensively. You could also much more easily measure team-mate effects: you could measure exactly how much defensive attention an offensive player attracts and how that helps (or hurts) his team-mates, or you could see how well a person defends the paint. This would allow us to much better identify combinations of players which would be successful or unsuccessful and ultimately make better predictions.

However, there are three main reasons why visual tracking data was not included in my thesis:

- Visual Processing data only became available starting in 2010. Furthermore, only half of the teams collect this data. Drawing meaningful results when ignoring half of the league and from a span of three years might be futile.

- The data is still very raw. Visual processing data only has the locations of the players and the ball 30 times a second, which is extremely low-level information. You would have to perhaps first identify a lot of medium level information such as categorizing plays, formations etcetera before being able to make high level predictions. Extracting this middle level information is a challenge in itself.

- Finally, the data is not publicly available, as it is only available to "select partners" of the NBA.

Because of the above issues, I had decided to focus my inquiries into the NBA by using play-by-play-data.

Chapter 2

Literature Review

In this chapter, I describe some of the previous work that has been done in the field of basketball analytics, a field which includes both predictive and descriptive analytics. Traditional analysis has focused mostly on evaluative/descriptive statistics, *i.e.* those that focus on describing what has already happened. The goals of those analyses is usually to evaluate players and/or teams and not necessarily to make predictions. However, they are still useful to review, as they are a good starting point in analyzing basketball. I discuss the descriptive statistics in section 2.1. Next, I review some predictive models of the NBA in section 2.2. In each of the sections, I also talk about reasons why I improve upon the existing work.

2.1 Descriptive Models

There exist a multitude of models for the NBA built by hand by domain experts. They can generally be separated into three categories:

1. Team based analysis.
2. Box Score based analysis.
3. Plus-minus based analysis.

I describe the major works in their individual sections below, and then discuss reasons for why the descriptive models could use improvements in section 2.1.4.

2.1.1 Team based analysis

Team based analysis look at factors that drive winning at a team level. In particular, if basketball is though of as an economic game, every team has an equal number of possessions in a game (give or take a few, depending on who ends the quarter with the ball), so the objective of the offensive team should be to maximize the return (points) of that possession, while the objective of the defensive team should be to minimize it (note that towards the end of the game or end of a quarter, the objective can also depend on how much time is left, but these possessions generally make up a small portion of the entire game). This insight leads to the team level statistic of points scored per possession and points allowed per possession.

Dean Oliver, in Basketball on Paper ([13]), identifies "four factors" that drive points scored and allowed per possession: shooting percentage, turnover percentage, rebounding percentage, and affinity to get to the free throw line. These four factors, at the team level, are the commonly used in team level analysis of basketball.

Finally, another common team level statistic that is used is Pythagorean expectation, which is defined below

$$PE = \frac{PS^x}{PA^x + PS^x}$$

Where the Pythagorean expectation (PE) is an estimate of the winning percentage, PS is the number of points the team has scored, PSA is the number of points the team has allowed, and x is an exponent determined empirically. ([1])

2.1.2 Box-Score based analysis

Box-Score based analysis generally tries to look at each box score for players, and assign values to each of the individual box score statistics. Instead of considering totals for the box score statistics, however, they consider the box score statistics per minute, to judge a player's value on a per-minute basis. They also generally assume that this value scales linearly in the number of minutes the player plays. The most

common box-score based statistics are the following:

1. Wins Produced. Wins produced was developed by David Berri in 1999. The method starts at factors that drive wins at a team level, namely offensive and defensive efficiency. It then divides credit for factors which drive offensive efficiency into the players responsible for the box score statistic. Finally, the defensive credit is split equally among all players on the team, weighted by minutes played. Note that "dividing" the credit is done by linear regression (so, for example, offensive efficiency at a team level is regressed against individual box score statistics at a team level). Ultimately, the goal is to estimate the number of wins a player produces for his team, had he played the entire game. Hence, since there are 10 players on the court at at time, and one win is assigned per game, the average wins produced should be about .100. A detailed way to calculate it can be found on the wages of wins website. [7] [2]

2. Win Shares is an extension of Bill James' work on baseball onto basketball. The methods to calculate the offensive portion of win shares is similar to that of wins produced. Defensive win shares, however, are calculated at an individual level instead of at a team level. This is done by first finding the position a player, and then regressing his defensive efficiency the offensive efficiency of his opponent at the same position. Finally, the sum of the offensive and defensive win shares is the total win shares of the player. Similar to wins produced, it is common to look at win shares per 48 minutes of playing time, as opposed to just the total number of win shares. The work was performed by people at Basketball-Reference. A detailed method of calculation can be found on the basketball-reference website. [1]

3. PER, or Player Efficiency Rating. PER was developed by John Hollinger, and it assigns coefficients to box score statistics per minute played, and then takes the sum of the coefficients multiplied by the box score statistic, and adjusts for the pace of a team. The process that Hollinger used to assign the coefficients are not transparent. This statistic is most commonly used among fans and used

often in the media, but is less commonly referred to in academic work. A more detailed explanation can be found on the basketball-reference website. [1]

4. Wins Above Replacement Player, or WARP. WARP is another system that assigns values to each of the box-score statistics, and the value of a player is the sum of the weights multiplied by the number of times that player performs a box-score statistic. Like PER, the weights seem arbitrary; quoting the creator of WARP, Kevin Pelton, "WARP requires a number of assumptions - the value of assists, the trade-off between usage and efficiency, and replacement level." Like PER, WARP does not attempt to account for defensive performance. [4]

2.1.3 Plus-minus based analysis

Using play by play data, it is possible to calculate the "plus-minus" statistic of a player. The "plus-minus" statistic of a player is simple the number of points his team scored while the player was on the floor minus the number of points the opposing team scored while the player was on the floor. Because the opposing team can score more points than the player's team, this statistic can be negative (hence the name, "plus-minus").

Using the plus-minus statistic as a foundation, Rosenbaum developed a statistic called the "adjusted plus-minus." The adjusted plus-minus tries to adjust for various factors that can disproportionately inflate or deflate plus-minus statistics. For example, if a player always played with the best players on his team, his plus-minus would be artificially higher. A detailed explanation of the method for adjusting these numbers can be found in Fearnhead ([8]).

2.1.4 Shortcomings in Descriptive Models

In this section, I will describe some shortcomings of the descriptive statistics. First, these models were built to **evaluate** teams or players, not necessarily **predict** what's going to happen if a set of players plays a different set of players. For example, Win Scores and Wins Produced were designed to measure how many wins a player

contributed to a team. While this isn't necessarily a shortcoming of the model, the fact that the models were designed for evaluation purposes suggests that the designers might have made trade-offs which reduced their predictive power. Furthermore, it is not necessarily clear in all cases how to turn the model from an evaluative one to a predictive one.

All team level models suffer from the following major shortcoming: they fail to account for changes in minute allocation. Changes in minute allocation can come from many sources. For example, a injury could derail a major star (or two) for extended periods of time. Furthermore, a mid-season trade could change the roster of the team. Finally, the minute allocation is usually significantly different between the playoffs and the regular season – examining data from 2011-2012, on average, around eight players played at least than 8% of the minutes for each team in the regular season, while only six players played at least than 8% of the minutes for each team in the playoffs. This suggests that teams tend to play (whom they perceive to be) their best players more often in the playoffs than they do in the regular season. This could lead to models overvaluing teams with strong benches and undervaluing teams with weak benches.

The individual box-score based models suffer the following shortcomings:

1. **They assume that player performance is independent of another player's performance** by assigning the value from a box score statistic to the person responsible for that statistic. However, this might not necessarily be the case. For example, if a player scored 11 points, he would get the credit for the value of 11 points, according the the model. However, if those 11 points were all scored because someone drove to the basket and drew defensive attention, freeing up the shooter for an open shot, certainly the person driving the ball should receive some credits too. Berri suggests in his book ([6]) that there is not much evidence that players (or external factors like coaching) have influence on how their team-mates play, so this assumption is valid. I will show later that, with high probability, players (or some other external factor) do significantly affect how other players play.

2. They might not divide defensive credit (or fault) fairly. Wins produced divides the credit evenly, so if a good defensive player plays with four bad ones, his defense might also look bad. Win Scores divides credit based on the opposition at the same position. This again might not work well, because basketball is a team game. For example, if a guard lets his opposition drive to the basket because he's not a good driver, and the center fouls him instead of letting him attack, the guard still gets the blame for the free throws because the opposing guard played his position. PER and WARP don't even attempt to divide defensive credit.

3. They fail to take into account the context of the play; not all statistics of the same type are created equal. For example, a foul is generally a bad play, but a non-shooting foul is generally worse than a shooting foul (but a shooting foul with 12 seconds left in the game when your team is losing is a good play). Box-score statistics don't distinguish between the types of fouls, nor do they provide context about the fouls. Understanding the context of plays using play-by-play data might allow us to more fairly assign credit for exactly what happened.

The adjusted plus-minus model also suffers from the first flaw, since it assumes that a team strength is a linear sum of player strengths, ignoring any player-to-player interaction that might be present.

These models assume that a player's performance is independent of his team-mates' performance. However, I will provide evidence that, for at least one player, LeBron James, this is not true (note that I did not run the analysis for all 500 players in the NBA. I just chose one player whom I thought might have an impact on his team-mates' performances). For example, let us look at the three-point shooting of the top three shooters other than LeBron James (by number of shots taken) on Cleveland Cavaliers in the 2009-2010 season – the last year James played there (data from [1]).

Player Name	3PT Shots Made	3PT Shots Taken	3PT Field Goal Percent
Mo Williams	159	371	42.9%
Anthony Parker	108	261	41.4%
Daniel Gibson	71	149	47.7%

Let us also look at the player's 3 point field goal percent from seasons that they did not play with LeBron James (aggregated).

Player Name	3PT Field Goal Percent
Mo Williams	35.8%
Anthony Parker	40.3%
Daniel Gibson	39.4%

In their final season with LeBron James, the players shot the three point ball much better than they have in their careers without James. If we consider their career 3PT percentage without James to be their true percentage, then, according to a computer simulation, they would have collectively shot as well as they did in 2009-2010 with probability < 0.0014.

If we focus on one player in particular, Mo Williams, on his 3 Pt shooting on seasons he played with and without James, we get the following:

Player Name	3PT Shots Made	3PT Shots Taken	3PT Field Goal Percent
Mo Williams with James	342	791	43.2%
Mo Williams without James	482	1345	35.8%

If we consider Williams' 3PT shooting percentage without James to be his true three point percentage, then, he would have shot as well as he did when he played with LeBron James with probability $< 2 \cdot 10^{-5}$.

The evidence above suggests that something was different when Williams (and the other 2 players) played with James. It is entirely possible that maybe Williams was in a better stage of his career during the two seasons he played with James, but it is also very possible that James significantly improved his 3-point shooting by being on

the court at the same time, and hence taking defensive attention away from Williams, which would suggest that both the assumption that player's play is independent might be false, as well as suggesting that just assigning values to box-score statistics might mislead one to overvalue or undervalue a player's importance.

2.2 Predictive Models

In literature, there are also a few **predictive** models for the NBA, and many more for the NCAA. Most of these models attempt to solve the same problem that I am attempting to solve; that is, to predict the outcome of one game.

There are some models that attempt to predict outcomes in the NBA. However, most of these models just seem to simply apply machine learning techniques to box score data without doing much else. These models simply take team-level box score data and apply techniques such as logistic regression, naive Bayes', or SVNs. Some of them also use some method for feature selection. However, they don't seen to present any interesting features or other insights from their work. For example, one model used linear regression on box-score data from 1992 to 1996 to predict 69% of the games correctly [5]. Another model uses a Naive Bayes classifier on box-score data to correctly predict 67% of 700 games from 2009 to 2010 [12].

There are also an abundance of models that attempt to predict outcomes in NCAA basketball. These models seem to be much more interesting. One model by Kvam et al., for example, applies a hidden markov model, where we can observe a team's strength when it plays another team, and these observations get updated as teams play other teams [9]. While this model does take into account injuries (as the observations after the injuries will tend to reflect a new state of team strength), it might take a while for injuries to actually reflect in the model. Furthermore, this model also assumes that team strength relations are transitive, which we show is not necessarily true in some cases.

Another model by Nate Silver uses a Naive Bayes classifier to predict outcomes in the NCAA tournament [14]. However, as inputs to the model, instead of using

team box-score statistics, he uses team strengths from 5 other models, as well as pre-season predictions. He demonstrates that the pre-season predictions are really important in the NCAA; teams that over perform during the season relative to pre-season predictions tend to do poorly in the NCAA tournament (all else equal), and teams that under perform during the season relative to pre-season predictions do well in the NCAA tournament (all else equal). This evidence suggests a regression to the mean sort of phenomenon. Silver also uses other inputs to the model, such as distance travelled by both teams, and discovers that teams that travel more tend to perform worse than they would otherwise. While Silver's model seems to perform well, his methods doesn't quite scale to the the problem in hand. First, his inputs are five other predictive models which are regularly published; there don't seem to be such published predictions in the NBA. Second, he only predicts outcomes in the tournament, which has 63 total games, whereas my goal is to predict outcomes for over 1000 games in a season.

Chapter 3

Novel Features

An important contribution of my thesis is to suggest various different novel features that I extract from play-by-play data and use in my predictive models. In particular, I extract features from play-by-play data which add context to the plays. Furthermore, I also extract features which model team-to-team match ups, minute distributions, and other features that are useful in predicting outcomes in games. In this chapter, I outline some of the various features that I extract.

The first set of features that I present are mostly features that aren't novel; they are used in other models or otherwise used in literature, or are slight variations of already known features. However, these features are still useful to mention in case the reader is not familiar with them.

The second set of these features described depend on minute allocation, with the idea being that more consistent minute allocations could perhaps indicate that a team or player is stronger.

The third set of features attempt to measure the impact that a player has on his team-mates performance both offensively and defensively.

In the fourth set of features, I attempt to capture inter-team dependencies such as when an good 3pt shooting team plays a poor 3pt defending team, and other similar situations.

Finally, the most important set of features I present can be found in section 3.5. In this section, I present features which add context to traditional box-score statistics

by taking into account the shot clock and the general efficiencies of the offense and defense of the team. The key insight for these set of features is accounting for the fact that an offensive possession becomes less valuable as more time is used in the possession.

Note that in this chapter, I don't discuss the usefulness of these features in detail. Instead, I discuss how these features are useful in making predictions in chapter 4.

3.1 Borrowed Features

Many features which are necessary to create a good predictive model in the NBA already exist. For example, as previously mentioned in the paper, about 60% of the games in the NBA are won by the home team, so it is natural to use home team as a feature in any predictive model. Below are some other features which are fairly common in the NBA or sports models:

- **Rest.** How much rest the team has gotten recently. There are different ways of measuring this, such as the number of games played in last x days or the number of days since the last game.

- **Distance Travelled.** Nate Silver demonstrated that the further away the team is from their home, the worse they tend to do in the NCAA. We put this to the test in the NBA.

- **Altitude.** Furthermore, teams located in higher altitudes (e.g. the Denver Nuggets and Utah Jazz) tend to have better home-court advantage than other teams

There are other features which others have tested, but have found that the features aren't significant in predicting performance. Some of these features, such as clutch performance, however, are found not significant due to a result of not appropriately being able to measure what the feature is trying to measure. I describe a better way to measure clutch performance in section 3.1.1.

Finally, I describe some other features that are relatively new and haven't been tested in predicting outcomes. For example, one such feature, expected points per shot (XPPS), attempts to capture how effective teams are at taking and allowing "good" shots in an offense or defense. I take this idea and attempt to measure how good teams and players are in their shot selection in section 3.1.2.

3.1.1 Measuring Clutch Performance

One factor that other models attempt to incorporate is some measure of how good a team is in the "clutch," generally defined as game situations which are more important than others, which amounts to situations when the game is close and the game is almost over. The usual way of measuring clutch performance, however, is generally to take a team's record for games which ended "close," for an arbitrary measure of close (usually if the game ended by a score differential of less than 10 points). They generally discover that this measure of "clutch" is not statistically significant in predicting future outcomes [14]. However, the general way of measuring clutch performance might not accurately capture whether the game was actually close because of the following reasons:

- Many games that are actually close in the fourth quarter (and would be considered clutch games with clutch situations) have the final score become lopsided because of high variance and low expected value strategies (foul, and then take a quick shot) at the end. These games would not show up if we look at the score at the end of the game.

- Likewise, many games that where one team had relatively no chance of winning often end up close enough to be counted as a "clutch" game at the end of the game.

Because of the above reasons, I propose a different measure of whether a game was clutch, a measure which I believe better incorporates whether a team or individual performs better in the "clutch." In particular, I propose that if the game was within

reach at any point in the last quarter of gameplay, it should be considered a clutch game. In particular, I propose the following measure:

1. If in the last two minutes of the game, the score of the game is within 3 points, then the game should be considered close.

2. If the first rule is not satisfied, then if the score of the game is within 5 points in the last 6 minutes of the game, then the game should be considered close.

3. If the first two rules are not satisfied, then if the score of the game is within 9 points within the last 12 minutes of the game, then the game should be considered close.

The thresholds above were chosen by looking at play-by-play data and finding (approximate) cut-offs where a team that is losing still has at least a 10% chance of making a comeback.

3.1.2 Measuring Shot Selection and Efficiency

Simplistically, to win in basketball, you must score more points than your opponent. The two main ways of doing this are to capture more possessions than your opponent, and to be more efficient with those possessions. In this section, I create features which attempt to capture (and allow us to make predictions) how efficient a team is in scoring (offensive efficiency), as well as capture how efficient a team is in stopping the other team from scoring (defensive efficiency). Traditional features that measure offensive and defensive efficiency are Points scored per possession, and points allowed per possession. While these features allow us to measure how efficient the offense and defenses are at a high level, they don't allow us any insight into why these offenses or defenses are more efficient, and don't allow us to measure more subtle match-up issues which might occur.

Instead of looking at total points per possession, Ian Levy had an idea to look at expected points per possession: Levy broke down the types of shots into five different types, and the NBA averages for points per shots of those types, data which

is presented in table 3.1 [10]. His dataset categorized each shot into one of five categories (which are in the table).

Shot Type	Points Per Shot (of that type)
Restricted Area	1.183
In The Paint (Non-RA)	0.793
Mid-Range	0.788
Corner 3	1.157
Above The Break 3	1.048

Table 3.1: This table shows the shot type, and the NBA average for points per shot per shot type. Taken from Ian Levy [10]

However, my dataset categorized shots slightly differently, and into 8 different categories, which is presented in table 3.2

Shot Type	Points Per Shot (of that type)	Frequency of Shot Type
Dunk	1.828	5.13%
Layup	1.096	24.72%
Jump	0.772	41.21%
3pt	1.100	22.38%
Tip	0.935	1.84%
FadeAway	1.098	0.48%
Hook	1.025	2.87%
Bank	1.357	1.34%

Table 3.2: This table shows the shot type, the NBA average for points per shot per shot type, and the frequency of that particular shot.

Hence, perhaps to be a good offensive team, a team should take more of the shots that are good (have higher points per shot). Using this insight, Ian created a metric which he called expected points per shot, which is formally defined in equation 3.1:

$$XPPS = \Sigma_i f_i \cdot PPS_i \tag{3.1}$$

Hence, XPPS (for a particular team or player) is just the sum over all shot types i of the frequency of i multiplied by average points per shot of i. Note that offensive $XPPS$ attempts to measure the team's offensive shot selection, where as defensive $XPPS$ attempts to measure the team's defensive shot selection. In Figure 3-1, I show a plot of the team's offensive efficiency vs the teams offensive $XPPS$.

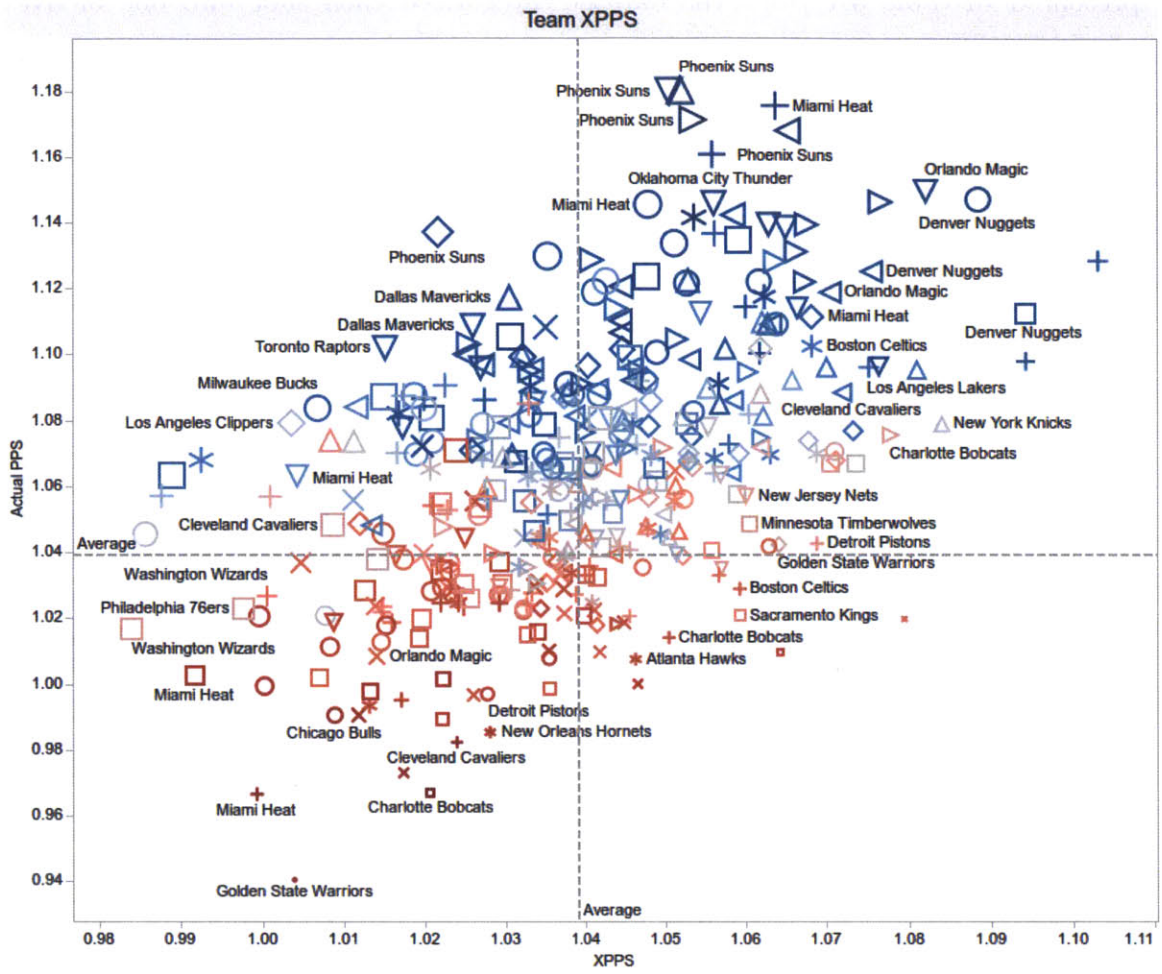


Figure 3-1: A plot of the team's offensive efficiency vs $XPPS$. Data is from 2001 to 2013. The chart is from Levy [10]

We can see that in general, as a teams shot selection ($XPPS$) improves, their offensive efficiency also improves. However, there are two other important things we can notice in the graph: the relationship between $XPPS$ isn't completely one-to-one, and that the teams with the best efficiencies don't have the highest $XPPS$. These observations suggest that there might be a diminishing returns effects during shot selection; if you only take shot types which are "good," then perhaps the other team will realize this, and defend the good shots more carefully and the bad ones less carefully. Furthermore, in addition to taking good shots, a team must also be making good shots to be extremely efficient. Because of these observations, instead of proposing $XPPS$ as a feature, I use the following (for each shot type): the frequency

of that shot type, and the efficiency of it (both at team and individual level). This allows us to incorporate both effects.

In Figure 3-2, I show a plot of the team's defensive efficiency vs the teams offensive $XPPS$.

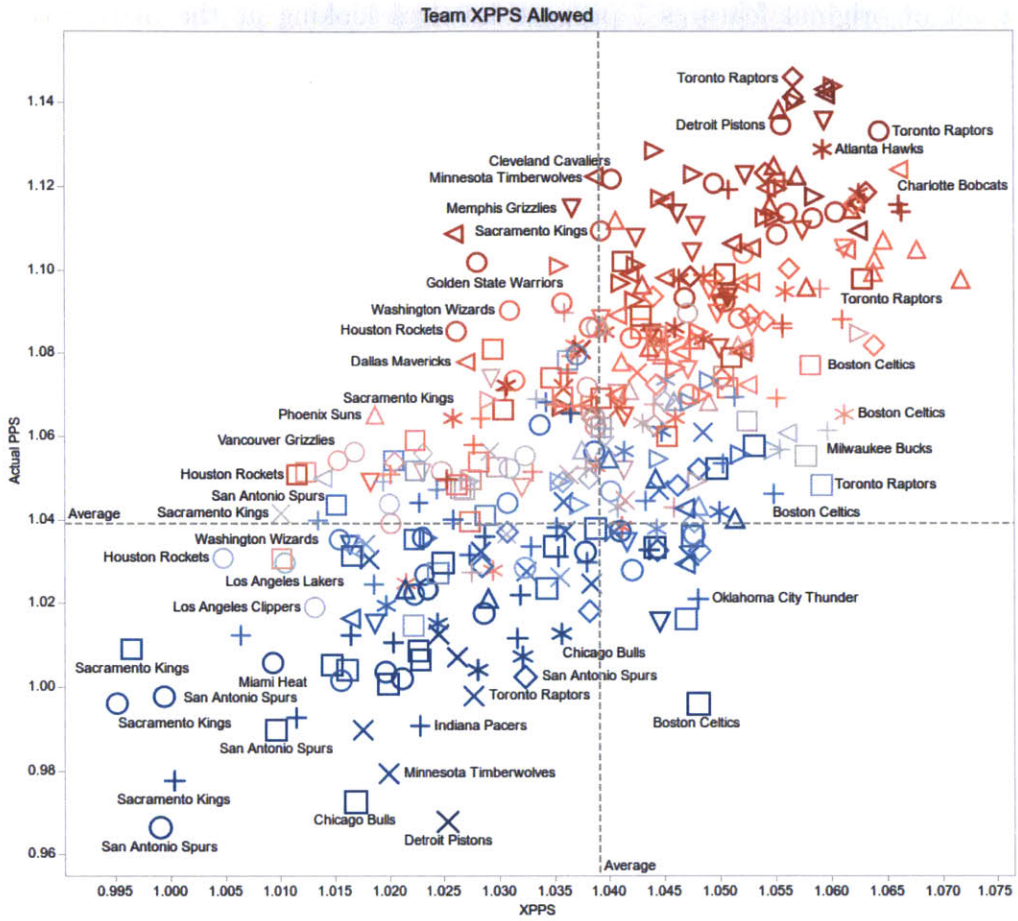


Figure 3-2: A plot of the team's defensive efficiency vs $XPPS$. Data is from 2001 to 2013. The chart is from Levy [10]

We can see a similar pattern: the teams which have a lower defensive $XPPS$ have better defensive efficiencies. However, there seems to be less of a deviation from the best fit line; defensive $XPPS$ seems to be a better indicator of defensive efficiency than offensive $XPPS$ is of offensive efficiency. Nevertheless, we use the same features for defense: the frequency of that shot type, and the efficiency of it (both at team and individual level).

In sections 3.3 and 3.4, we derive more features from these features which better

capture team-to-team effects and player-player effects than traditional features.

3.2 Minute Distribution

The first set of original features I propose involves looking at the distribution of minutes at both the team and individual level.

In the NBA, five players play for 48 minutes each. One hypothesis is that if a team is ideally built, they have five players who play really well together, regardless of who their opponent is, and are capable of playing all 48 minutes on a night to night basis. Hence, perhaps one measure of how good a team is could be how much a team deviates from this ideal distribution, because not deviating from this distribution would suggest that at any point, the coach knows who his five best players are. However, deviating from this ideal distribution could be an indication that the team has good bench (substitute) players, and are thus could be an indication of a stronger team. Regardless, I propose a deviation from this ideal distribution as a useful feature measure of team strength. More formally, I propose the following measure:

$$ID_{dev} = \Sigma_{i=1}^{i=5}(48 - m_i)^2 \tag{3.2}$$

In equation 3.2, ID_{dev} represents a measure of the deviation from the "ideal" distribution in a game, and m_i is the number of minutes that the player with the i'th most minutes plays in a particular game. Again, the hypothesis is that an "ideal" ID_{dev} is 0, and the larger the ID_{dev}, the farther away from ideal (or worse) the distribution gets. I use the average ID_{dev} as an input in some of the various models that I build.

A second measure of minute distribution that I propose aims to measure players' variations in minutes played. Here, the hypothesis is that in addition to playing the (perceived) best players on the team the most, playing the same players consistently might imply another degree of strength for the team. Furthermore, for a particular player, the fact that that player plays the same minutes consistently could imply that his role in the team is well defined, implying value at the player level. More formally,

I propose the following as a feature:

$$CONS_i = \frac{\sigma[MPG_i]}{\mu[MPG_i]} \tag{3.3}$$

In the equation 3.3, $CONS_i$ is a measure of a player i's consistency of his minutes per game played (MPG). Instead of taking the standard deviation of MPG_i, however, I decided to take the standard deviation of MPG divided by the mean of MPG as a more useful measure of consistency. For example, we would consider a player who plays 40 minutes half the time and 35 the other half to have more consistent minutes than a player who plays 5 minutes half of the time and 0 the other half. A lower $CONS_i$ for a player indicates more consistent minutes. I use $CONS_i$ as a feature in the player models. At a team level, I use the average $CONS_i$ for the 7 players with the most minutes as a feature.

3.3 Player-to-Player Interactions

In this section, I describe features that attempt to better incorporate player-to-player dependencies. As described earlier, most predictive models tend to assume that a players value to a team is not dependant on who that player plays with; however, I propose various features which model attempt to model some player-to-player dependencies, and, in later chapters, actually show that these features are useful in making predictions, implying that a player's value is indeed dependant on who that player plays with (Note that in 2.1.4, I discussed some reasons why the assumption that player's play independently of there team-mates could be flawed).

In order to measure a player's impact, I propose features that look at player-player pairs offensively, and measure how a player's shot selection and shot effectiveness changes when another player is playing. To measure a player's defensive effectiveness, I measure how a player changes the opposing player's and team's shot selection and shot effectiveness.

In particular, on offense, I look at the difference between a player's team-mates' shot frequencies and efficiency when that player is playing and when he isn't; this is

done on a player-to-player level and not a team to team level. For example, if for a player, I compare the difference of the shot frequencies and efficiency (by shot type) between when the another one of his team-mate when that team-mate is playing with him and without him. This is done for all player-player pairs on a particular team. On defense, I look at team performance change on defense (measured by change in shot efficiency and allowed shot selection), as well as impact on the change in efficiency and shot selection of the player he is guarding.

I talk more about the predictive capabilities of these features in chapter 4. However, there are one interesting result which are worth presenting now: LeBron James was indeed one of the highest in terms of increasing his teammates' 3pt efficiency **and** shot selection when he was on the floor in 2008-9.

3.4 Team-to-Team Interactions

In this section, I describe features that attempt to better incorporate team-to-team dependencies. My goal in this section is to measure things like what happens when a good rebounding team plays a bad one, or when a good 3pt shooting team plays a bad one, etc. The features I describe are pretty simple, but are still really useful. In particular, I first rank the teams in the following:

- Free Throw Make Percentage. How good they are at shooting free throws.
- Free throws per possession. How often they take free throws.
- Fouls committed per game.
- Fouls drawn per game.
- Frequency in taking different types of shots.
- Frequency in allowing different types of shots.
- Efficiency in taking different types of shots.
- Efficiency in allowing different types of shots.

- Offensive rebound collected per opportunity.
- Defensive rebound collected per opportunity.
- Turnovers committed per offensive possession.
- Turnovers forced per defensive possession.

Each team has a rank between 1-30 in each of the above, with a rank of 1 indicating that they are the best in the league in that category. Then, for each pair of relevant rankings (for example, if a team is good at taking 3pt shots, and is playing a team that is bad at defending them), I create a new feature indicating so. I try different cut-offs for "good" and learn which are the most useful at making predictions.

While the features I present in this section are simple, there are a few reasons why they are important. Often times, many of the successful predictive models (SVNs, Logistic Regression) are linear, and don't incorporate these non-linear dependencies. While one could easily use SVNs with higher order kernels, those higher order kernels might over fit otherwise linear dependencies. Hence, it can be important to explicitly state these second order features, instead of allowing the model to learn them.

The most interesting result of these features is that team strength relationships are not transitive as most models assumed. This result is detailed in chapter 4.

3.5 Context Added Metrics

In the NBA, a team has 24 seconds on offense to take a shot that hits the rim; else, they forfeit the offensive possession. Hence, a possession becomes less valuable as the possession ensues because the offense has less time to create a good shot for the team. Often times, shots late in the shot clock are forced; taking a bad shot is still better than taking no shot at all.

We can see this phenomena reflected in play-by-play data. In figure 3-3, I plot the average points scored per shot over (averaged over all teams over the course of a season) vs the number of seconds remaining in the possession.

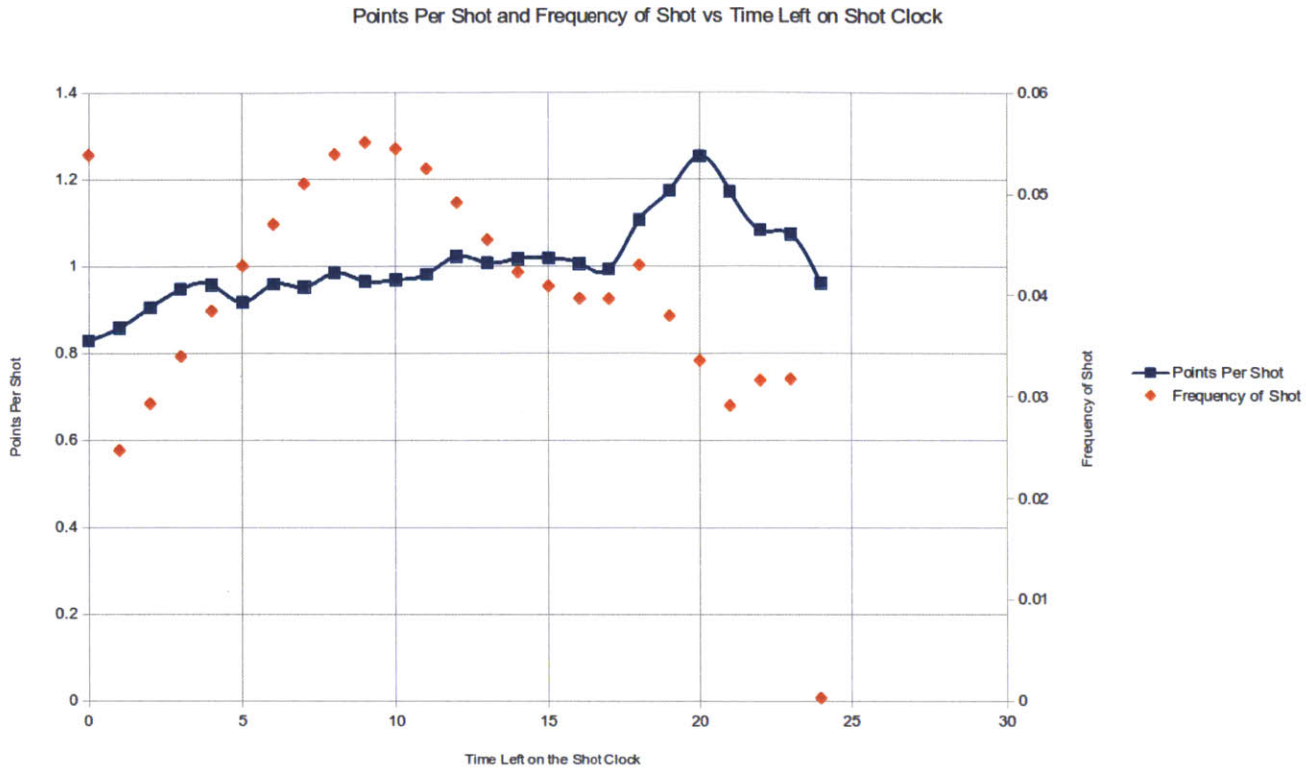


Figure 3-3: A plot of the points per shot (blue points and line) and the frequency of shot (orange points) vs the time left on the shot clock (x-axis). As there is more time left on the shot, the shots tend to get better.

As we can see, shots taken early in the shot clock often are more valuable offensively on average. There are two additional things to note in the above chart. First, there seems to be a noticeable dip at around 12 seconds into the shot clock. Perhaps at that time, . Second the value of shots taken with 24 seconds left on the clock is actually lower than most other times; this is not necessarily because those shots are less valuable, but is likely just an artifact of how I processed the data. If the shot clock inferred from play-by-play data was nonsensical (the time difference between the shot and the previous recorded play was more than 24 seconds away), I said that there were 24 seconds still on the shot clock. However, these possessions made up only .02% of the total shots taken. If we ignore these possessions, there is a fairly strong correlation between the points per possession and the time left on the shot clock ($r^2 = .75$ without those possessions, and $r^2 = .61$ with).

Another interesting trend to notice is the frequencies of the shots for a particular

time left. When there is a lot of time left, the number of attempts remains low (likely because players are selective with attempts at this point). However, as the time left decreases, the number of attempts increases rapidly. The trend implies that as there is less time left, players become less selective which their shots and take more difficult shots because they can expect to take even more difficult shots later.

Two other critical factors that affect how valuable a possession will be is the team that has the offensive possession, and the team that they are playing against. An offensive possession is more valuable to a team who is good at scoring, and it is more valuable against a team which is bad at defending. Taking these factors into account, we can create a new measure of shooting efficiency, which I call context added efficiency.

First, we create a simple model which uses the time left on the shot clock, the offensive team's efficiency, and the defensive teams efficiency as factors to predict the value of that possession. Then, we subtract the estimated value of the possession from the actual value of the possession to get a metric called context added efficiency (CAE, for short). Note that this metrics is the value added for that possession. From this, we derive three new metrics which I use as factors in my model:

- CAE per possession used. This is the context added efficiency per shot that a player took.
- CAE per possession on the floor. This is the context added efficiency per offensive possession that a player was on the floor.
- Total CAE. This is the total CAE that a player has had, summed over all of his possessions.

One of the biggest problems that CAE addresses is the inconsistency of a players shooting efficiencies over seasons. In particular, a player's true shooting percentage in one season only has a correlation (r^2) of .31 from one season to the next, and is actually lower for players who change teams. CAE, on the other hand, is much more consistent year to year ($r^2 = .63$), demonstrating that player's shooting skills don't

change as much as previously expected, but just need to be adjusted situationally. In chapter 4, we see how CAE is useful in predictive models.

Likewise to possessions, turnovers later in the possession are less costly to the team because the possession itself was less valuable. Likewise, steals later in the possession are less valuable to the defensive team. I also make adjustments in valuing turnovers and steals.

Chapter 4

Predictive Models

In this chapter, I describe the models I use to predict outcomes in the NBA, as well as the techniques I used to select features for those models. First, I describe the feature selection algorithms I used in section 4.1. Next, I describe the different models I built. The first set of models I built use the team-level features described in chapter 3. These team-level models can be found in section 4.2. The second set of models I use are models that mainly use the player-level features described in chapter 3. These player-level models can be found in section 4.3. Note that in both the team and player level models, I assign features such that team 1 is always the home team and team 2 is always the away team. Hence, home court advantage is implicitly built into the model.

In general, these models are important for two reasons: first, these models demonstrate that certain features described in chapter 3 are actually important in predicting outcomes, which suggests that the feature is important in some way and might in turn reveal something about the NBA. Furthermore, these models push the (known/published) state-of-the-art in terms of predicting outcomes. Before I delve into these models, let us first discuss what the state-of-the-art is.

One baseline, as mentioned earlier, is predicting 60% of the games correctly by predicting the home team every time. Second, some of the models discussed in other published research claim predicting 68% of the games correctly. Finally, betting lines in Vegas predict approximately 69.5% of the games correctly; these betting lines seem

to be the best publicly available predictors of outcomes in a game [3]. My goal then, is to attempt to correctly predict a higher fraction of games than 69.5%, which both the individual and player models I present achieve.

In this entire chapter, the models were built and cross-validated on the almost the entire season for each season of data, minus a few games that had inconsistent or incomplete data available. No season had more than 20 missing games. Furthermore, while I attempted to rank features using all three criteria mentioned in section 1.1, the criteria presented a very similar set of features. Because other papers (or Vegas odds) only use the number of wins criterion, I decided to focus my analysis on optimizing the number of correct predictions.

4.1 Feature Selection

The goal of a feature selection algorithm is, given a learning algorithm and a dataset, find the best subset of relevant features in the data which work the best (according to some criteria) with the learning algorithm. In my work, I used two primary methods of selecting features, forward selection and a genetic algorithm. I describe the forward selection in detail in section 4.1.1 and the genetic algorithm in detail in 4.1.2. Finally, I compare the two methods, as well as discuss why I did not use other feature selection methods in section 4.1.3.

4.1.1 Forward Selection

The forward feature selection algorithm is a greedy feature selection algorithm that keeps a set of best features, and greedily chooses the next best feature to add to that set. Below, I provide the pseudo-code for this algorithm:

```
forward-selection(LA, D):
    selectedFeatures = the set of selected features.// Initially empty.
    The set is ordered
```

$errorMap$ = empty map. // this map maps the feature (key) to the error at the time the feature was added (value)

$featuresToAdd$ = $D.F$ // this is a list of all the features that we have not yet added to S. It contains all the features in D

while $featuresToAdd$ is not empty:

$bestError = \infty$

$bestFeature = null$

for $feature$ in $featuresToAdd$:

$tempError$ =`train-and-test`$(LA, D, selectedFeatures + feature)$

if $tempError < bestError$:

$bestError = tempError$

$bestFeature = feature$

S.`add`$(bestFeature)$

$errorMap[bestFeature] = bestError$

$featuresToAdd$.`remove`$(bestFeature)$

return $selectedFeatures, errorMap$

The forward selection algorithm above requires as an input a learning algorithm, LA, and data, D. Note that $D.F$ is the set of features in the data. The algorithm calls `train-and-test`, a function that returns the error after applying the learning algorithm to the dataset with the selected features, cross-validating the results, and in certain learning algorithms, performing parameter search. While `train-and-test` differs for each learning algorithm, a basic sample implementation is shown below:

`train-and-test(LA,D, selectedFeatures+feature)`:

$totalError = 0$

for $partition$ in $D.partitions$:

`LA.train(`D-$partition$`)`

$totalError+$ = `LA.test(`$partition$`)`

`return` $totalError$

The data was separated into 10 seasons (partitions) – the seasons served as a natural point to split the data and perform 10-fold cross validation. In the above implementation, `train-and-test` trains the learning algorithm on the data without a season, tests the resulting model on the left out season, and then repeats the process for all seasons, summing up the errors along the way. Finally, it returns the total error summed over all the seasons.

The implementation of `train-and-test` above assumes that the learning algorithm requires no additional modifications, such as selecting parameters or assigning dependencies between features. Some of the learning algorithms I used, however, did require either parameter selection or additional modifications. For example, when using Bayes' Nets, I needed to attempt different combination of dependency relationships. Likewise, when I used SVMs with a RBF kernel, I needed to select the parameters γ and C. An example modified `train-and-test` implementation for SVMs with a RBF kernel using grid search on γ and C is shown below:

```
train-and-test(LA,D, selectedFeatures+feature):
    bestError = ∞
    bestGamma = null
    bestC = null
    for gamma in 2^-10, 2^-9, 2^-8, 2^-7, 2^-6, 2^-5, ..., 2^10:
        for C in 2^-10, 2^-9, 2^-8, 2^-7, 2^-6, 2^-5, ..., 2^10:
            totalError = 0
            for partition in D.partitions:
                LA.train(D-partition, C, gamma)
                totalError+ = LA.test(partition)
            if totalError < bestError:
                bestError = totalError
                bestGamma = gamma
                bestC = C
    return bestError, bestGamma, bestC
```

4.1.2 Genetic Feature Selection

In my work, I also used a genetic feature selection algorithm. In this algorithm, I kept a population of candidate solutions, or subsets of features. Initially, I initiated the population with the 100 best candidate solutions out of 1000 completely random solutions. To create new solutions, I randomly picked two solutions, crossed them over, and mutated random bits.

I represented each candidate solution as a string of bits, where the i'th bit was 1 if the i'th feature was part of the candidate solution, and 0 otherwise. To initialize the population, I first created 1000 candidate solutions. To create these 1000 solutions, I created 50 solutions that used each feature with probability .025, 50 solutions that used each feature with probability .075, 50 solutions that used each feature with probability .125, ..., and 50 solutions that used each feature with probability .975. I then ranked each of these 1000 solutions using the `train-and-test` algorithm described earlier, and added the best 100 solutions to the initial population.

To add new solutions to the population, I first picked two random solutions from the existing population. I then crossed the two existing solutions over, and added random mutations to the new solution. Finally, I ranked the new solution using the `train-and-test` function, and added the solution to the population if the solution met certain criteria. In particular, I added the new solution to the population if it was better than 10%, 25%, 50%, 75%, or 90% of the existing population (I used different thresholds in different runs of the genetic algorithm). Below, I have provided pseudo-code for the genetic algorithm that I used below:

```
genetic-feature-selection(LA, D):
    population = set of candidate solutions. each solution also has an
    associated fitness (error)// Initially empty.
    for i in 0, 1, 2, 3, ..., 19:
        repeat 50 times:
            candidate =""
            repeat length(D.F) times:
```

if `random()` $\leq .025 + .05 \cdot i$

candidate+ = "1"

else:

candidate+ = "0"

population.`add`(*candidate*, `train-and-test` $(LA, D, candidate)$)

sort *population* in decreasing order of performance

delete last 900 candidates in *population*

repeat:

parentOne = random entry from *population*

parentTwo = a different random entry from *population*

candidate = `produce-child`(*parentOne*, *parentTwo*)

candidateError = `train-and-test` $(LA, D, candidate)$)

if *candidate* better than *threshold* percent of current *population*:

population.`addInOrder`(*candidate*, *candidateError*) // adds the candidate so the population is still sorted decreasing order of performance

The `genetic-feature-selection` algorithm depends on the `random` function, a function that returns a random double between 0 and 1, and on the `produce-child` function, a function that takes two candidate solutions and produces a new candidate solution based on the two parents by crossing over the parents and adding in random mutations. In my implementation of `produce-child`, I used various methods to cross the parents, and I also varied the probability of a mutation. Below, I have provided pseudo-code for my implementation of `produce-child`.

`produce-child(parentOne, parentTwo)`:

child ="";

randomNumber =`randomInt`$(1, 3)$ //used in determining how to cross over

if *randomNumber* = 1 : //one-point crossover

splitPointOne =`randomInt`$(1, length(parentOne))$

$child = parentOne[1 : splitPointOne] + parentTwo[splitPointOne : length(parentOne)]$

if $randomNumber = 2$: //two-point crossover

$splitPointOne =$`randomInt`$(1, length(parentOne))$

$splitPointTwo =$`random`$(splitPointOne, length(parentOne))$

$child = parentOne[1 : splitPointOne] + parentTwo[splitPointOne : splitPointTwo] + parentOne[splitPointOne : length(parentOne)]$

if $randomNumber = 3$: //uniform crossover

for i in $1, 2, 3, \ldots, length(parentOne)$:

if `randomInt`$(1, 2) = 1$:

$child+ = parentOne[i]$

else:

$child+ = parentTwo[i]$

$probabilityMutation =$`random`$()$

for i in $1, 2, 3, \ldots, length(child)$:

if `random`$() \leq probabilityMutation$:

if $child[i] =$"1":

$child[i] =$"0"

else:

$child[i] =$"1"

return *child*

The `produce-child` function depends on the function `randomInt`(a, b), a function that returns a random integer between a and b (inclusive). In the `produce-child` function, I first randomly chose one of three crossover methods: one-point crossover, two-point crossover and uniform crossover. The different cross-over methods are visually represented in figure 4-1. After crossing over and producing a child, I randomly introduced mutations into the child, with variable probability. Then, I finally return the child to the `genetic-feature-selection` algorithm.

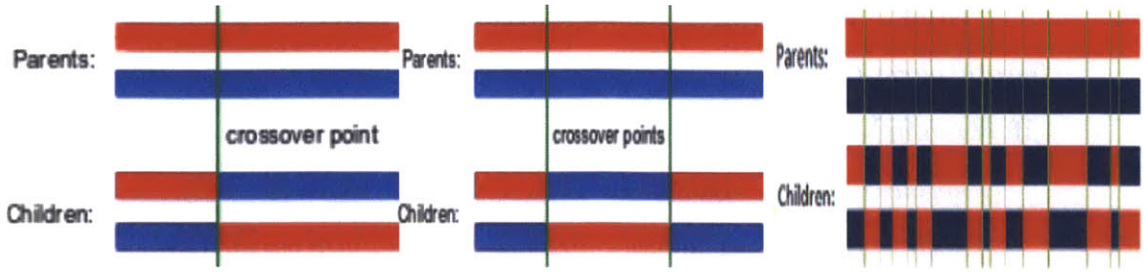


Figure 4-1: Visual illustrations of the three different crossover techniques that I used. On the left is one-point crossover. In the middle is two-point crossover. Finally, in the right, is uniform crossover. The images are compiled from wikipedia. The rightmost image is licensed under the GNU Free Documentation License, and the others are licensed under the creative commons license [15].

4.1.3 Comparison of Feature Selection Algorithms

In this section, I discuss the performance of the feature selection algorithms I used, as well as reasons for why I did not use other feature selection algorithms. To normalize results across the feature selection algorithms, the results in this section only reflect using one learning algorithm, namely, using support vector machines with a RBF kernel function. I ran the algorithms on a computer with a Intel Core i5-3320M Processor and 12GB of DDR3 memory.

In figure 4-2, I plot the results of running both the greedy forward feature selection algorithm as well as the genetic feature selection algorithm versus the time it took for the algorithm (measured in seconds). I used 10-fold cross validation, but did not tune the SVM parameters, when I generated the results for figure 4-2. Both algorithms were run on team-level data.

As shown in figure 4-2, the forward selection algorithm took about an hour to reach a performance level that was close to its best, while the genetic selection algorithm quickly reached a level close to its best, but did not significantly improve after that. Furthermore, the forward selection algorithm always performed better than the genetic algorithm.

Given that the forward selection algorithm required one hour without any parameter search, a parameter search for γ and C between $\{2^{-10}, 2^{-9}, 2^{-8}, \ldots, 2^{8}, 2^{9}, 2^{10}\}$ during feature selection seemed unreasonable because testing one parameter would re-

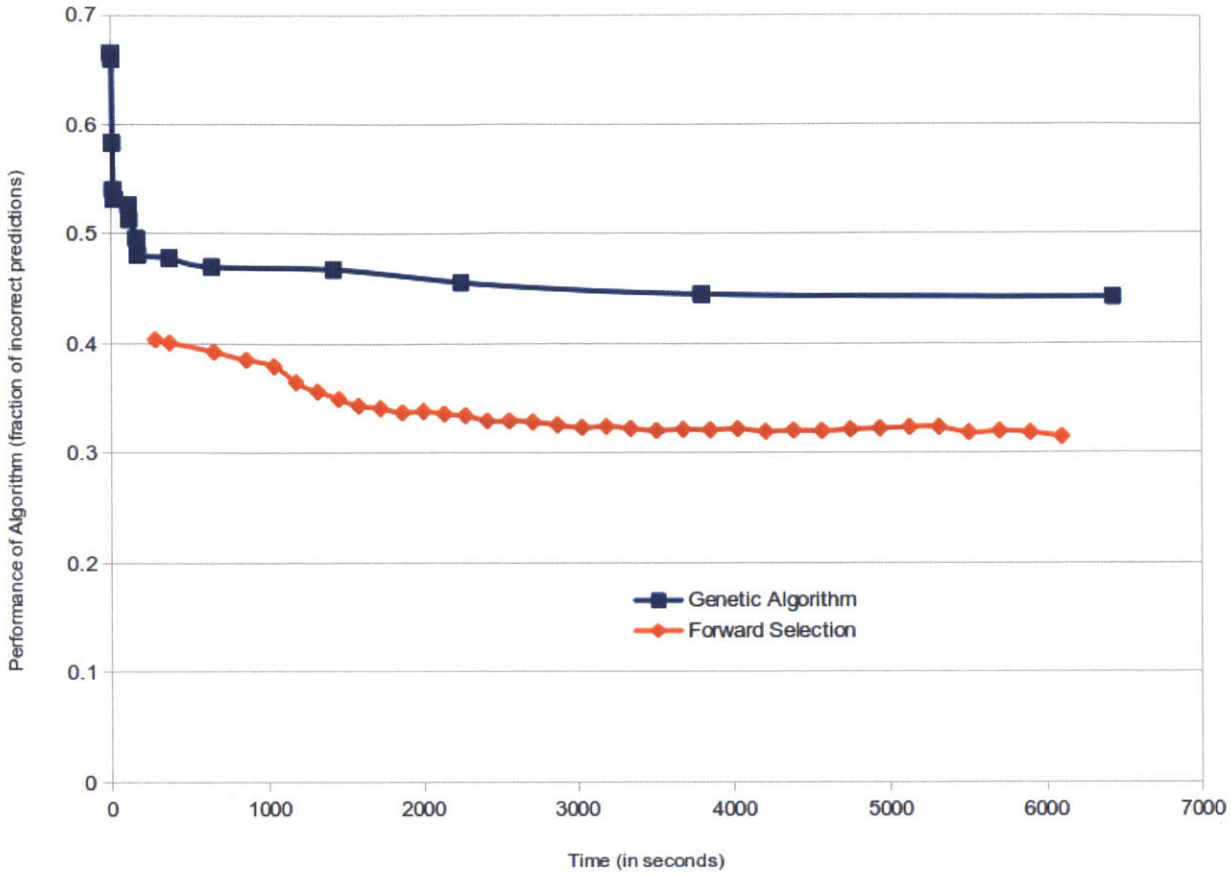


Figure 4-2: The performance of two feature algorithms versus time. The blue line represents the performance of the genetic algorithm when I only added the child if it was better than 90% of the existing population. The orange line represents the performance of the forward selection algorithm. The performance is measured in the fraction of games predicted incorrectly (using cross-validation), so the lower the performance, the better. The time is measured in seconds.

quire at least one hour, but testing over 400 pairs of parameter would require over 400 hours. Hence, I only performed parameter search for γ and C on $\{10^{-4}, 10^{-3}, 10^{-2}, \ldots, 10, 10^2, 10^3\}$, which required about 80 hours to completely finish (I performed a more extensive parameter search once the features were selected). The results of running both algorithms with parameter selection is shown in figure 4-3.

Just as the time without tuning the parameters in the SVM, the forward feature selection algorithm performed much better than the genetic feature selection algorithm. Because of the parameter tuning, however, both algorithms performed better than they did without tuning, which is not very surprising. In fact, the genetic al-

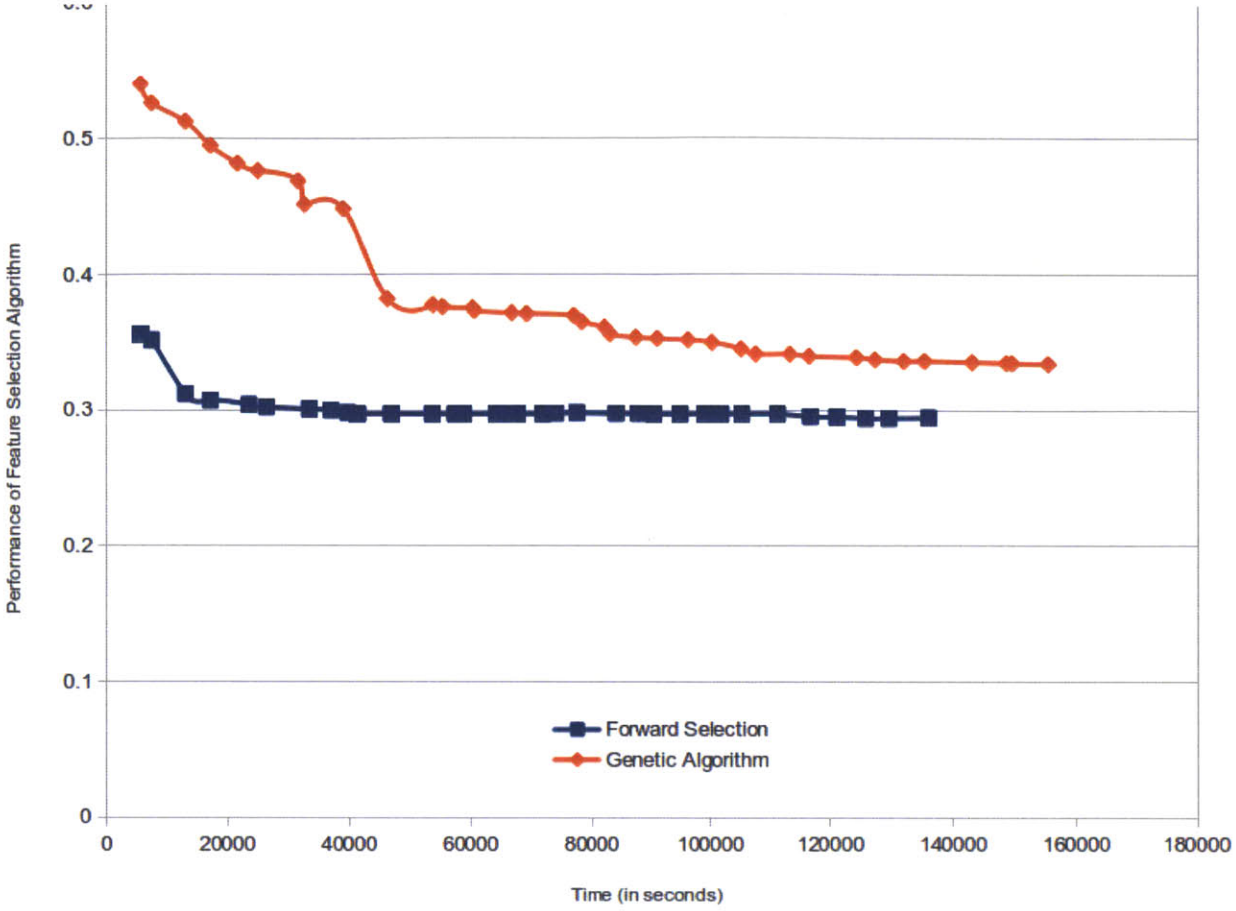


Figure 4-3: The performance of two feature algorithms versus time. The blue line represents the performance of the genetic algorithm when I only added the child if it was better than 90% of the existing population. The orange line represents the performance of the forward selection algorithm. The performance is measured in the fraction of games predicted incorrectly (using cross-validation), and the time is measured in seconds. The algorithms in this graph also tuned the parameters using grid search in the SVM.

gorithm performed almost as well with tuning as the forward selection algorithm did without tuning. However, both algorithms still required almost two days to achieve their optimal results.

Ideally, I would have liked to perform a much more exhaustive feature space search than the one I performed. However, this was not computationally feasible given my data-set, because both of the algorithms I used still required two days, and other feature selection algorithms would be very time-prohibitive, especially because the number of useful features (10 to 40) were a lot fewer than the number of actual

features (200 team level features and 500 player level features). In particular, if there are f useful features and F total features, where $f << F$, then forward selection will require $F + (F-1) + (F-2) + \ldots + (F-f+1) \approx fF$ calls to `train-and-test`. On the other hand, an algorithm such as backwards elimination would require $F + (F-1) + (F-2) + \ldots + f \approx F \cdot (F-f)$ calls to `train-and-test`. Because the cost of running a feature selection algorithm is directly proportional to the number of calls to `train-and-test`, the backwards selection algorithm would be incredibly cost prohibitive in this dataset.

4.2 Team Level Models

In my research, I tried many different models – Bayes Net, Naive Bayes, Linear and Logistic Regression, SVMs, *k-nn* – in predicting NBA outcomes. I processed my data using python, and performed most of the analysis in R. In particular, I used the `glm` function in R to perform Linear and Logistic Regression analysis, I used the `knn` function in R to perform analysis using the *k-nn* algorithm, and I used the SVM and Naive Bayes implementations in the `e1071` package in R. The most successful of these models were usually SVMs and logistic regression, so my analysis will mostly focus on results from those models. The results in this section are a result of using the forward feature selection algorithm described in section 4.1.1 with cross-validation and grid search on C and γ.

In table 4.1 below, I present the features which are most important if they were the only feature used in an SVM model with a radial basis function kernel.

Naturally, many of the features that performed the best when they were the only feature were high level features which encompass a lot of information about the team, such as point differential and winning percentage. There are two things that are interesting in table 4.1. First, the three most important features are high-level measures of strength of the home team; in fact, the first two measures add much more predictive power than any feature that measures the strength of the road team. Second, it turns out that clutch performance as measured in section 3.1.1 is actually

Feature	Correct Prediction Rate
Point differential (home team)	66.62%
Winning percentage (home team)	66.52%
Clutch Performance (home heam)	64.67%
Clutch Performance (away heam)	63.81%
Winning percentage (away team)	63.05%
Point differential (away team)	62.86%
3Pt Home Good Away Bad	61.67%
Old Clutch Performance (home team)	61.10%
3Pt Shooting Frequency (home-team)	60.94%

Table 4.1: This table shows the features and the prediction rate if that feature was the only one used in an SVM model with a RBF kernel.

one of the most important factors in predicting outcomes of games. Note that the the traditional measure of clutch performance is only slightly useful in predicting outcomes.

The best individual features from a logistic regression model are shown in table 4.2.

Feature	Correct Prediction Rate
Winning Percentage (home team)	66.54%
Point Differntial (home team)	66.16%
Clutch Performance (home team)	63.78%
Clutch Performance (away team)	63.68%
Point Differntial (away team)	63.49%
Winning Percent (away team)	63.26%
Old Clutch Performance (home team)	62.78%
3Pt Shooting Percentage (home -team)	62.11%

Table 4.2: This table shows the features and the prediction rate if that feature was the only one used in an Logistic Regression model.

Not surprisingly, the most important features using logistic regression are very similar to those using SVMs.

While running iterative (greedy) feature selection, I selected the feature which performs the best in predicting outcomes when combined with the existing features. The results of iterative feature selection on SVMs can be found in table 4.3.

Adding more features than those in 4.3 didn't generally seem to improve the

Feature	Correct Prediction Rate
PPG Differntial (home team)	66.62%
PPG Differntial (away team)	68.24%
Dunk Shot Freq Def Away Team	69.14%
3Pt Home Good Away Bad	69.58%
Jump Shot Efficiency (home team)	69.71%
Clutch Performance (home team)	70.34%
Off Reb Rate (home team)	70.62%
Home Bad Reb Away Good Reb	70.87%
Player 1 Min Home	71.09%
FT Freq Defense Home team	71.31%

Table 4.3: This table shows the features and the prediction rate if that feature used with the features above it in an SVM model with a RBF kernel.

predictive capability of the model. The first two features in table 4.3 are not much of a surprise because of the greedy nature of the forward selection algorithm. These two features (along with home court advantage) do the bulk of the work of taking the model close to its optimal predictive capabilities.

It is the next set of features which still improve the model, however, that are more interesting. The third most important feature is how often the away team allows opponents to get dunk shots. The fourth is whether the home team is good at taking and shooting 3pt shots and if the away team is bad at defending them. The fifth is the efficiency of taking jump shots for the home team. The sixth is the home team's clutch performance. The seventh is off rebounding rate for the home team. Eight is whether the home team is bad at rebounding and the away team is good. Ninth is the average number of minutes the player with the most minutes on the home team plays. Tenth is the frequency of free throws the home team allows in a possession.

What's interesting is that even though point differential measures how good an offense is, various frequencies/efficiencies of different shot types when combined with point differential are important in predicting outcomes. Furthermore, clutch performance, even when we already have point differentials, is also important in predicting outcomes. Finally, two of the team-to-team interaction features show up: if the home team is good at 3pt shots and the away team is bad at defending them, as well as if the home team is bad at rebounding while the away team is good at them. Note that

this last feature is present even though another feature, the offensive rebounding rate for the home team, also measures how good the home team is at offensive rebounding.

Using the set of feature in table 4.3, I performed a finer grid search of the parameter space than I did during feature selection. In particular, I performed grid search on C and γ in the range $\{2^{-20}, 2^{-19}, 2^{-18}, \ldots, 2^{8}, 2^{9}, 2^{10}\}$. Small C values (less than 2) decreased the performance of the model, while values of C above 2 had no noticeable improvement in the model. On the other hand, small γ values improved the performance. In particular, the model performed the best when γ was in the range $(2^{-14}, 2^{-10})$. These results are visually illustrated in figure 4-4.

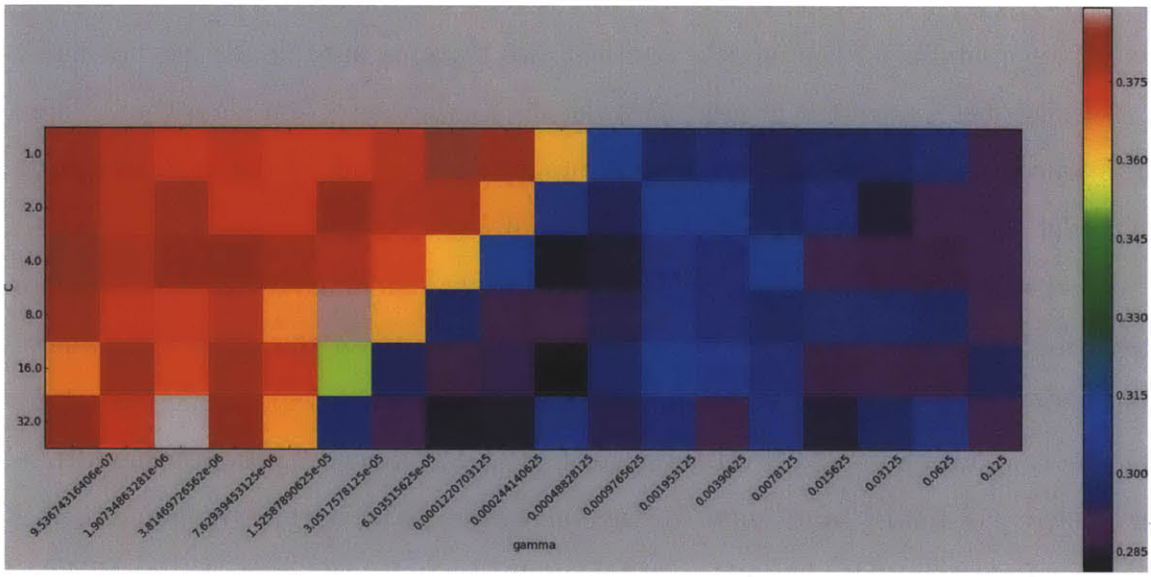


Figure 4-4: A plot of the performance of the SVM model with varying γ and C. In this table, γ is in $\{2^{-20}, 2^{-19}, 2^{-18}, \ldots, 2^{-2}\}$, while C is in $\{2^{0}, 2^{1}, 2^{2}, \ldots, 2^{5}\}$. Generally, as γ decreases in this range, performance of the model seems to improve up until a certain point, then remains constant, and finally worsens. C is mostly in its optimal range in this graph.

Using logistic regression resulted in a similar set of features to those in table 4.3: the first two features were again point differential for the home team and then for the away team. Clutch performance and some team-to-team features were still important, as were some features which measured efficiency and frequency of different shots. However, the predictive capability of the logistic regression models didn't quite reach above 71%.

Models other than logistic regression and SVMs didn't seem to work as well. Some reasons for why that may be are described in section 4.5. Furthermore, the genetic algorithms I implemented were not able to produce a better set of features than the iterative greedy algorithm.

4.3 Player Level Models

In my research, I also built models which used the player-level features described in chapter 3. In these models, I first assumed that I would know which 7 players on the team would get the most playing time. In general, this assumption seems mostly justified: about 87% of the time, the players can be predicted by just predicting the players from the previous game. When they can't, it is likely because of injuries, which you could realistically find out about through news sources. Similar to the team-level models, the most effective models in the player-level models were logistic regression and SVMs.

In table 4.4 below, I present the features which are most important if they were the only feature used in an SVM model with a radial basis function kernel.

Feature	Correct Prediction Rate
CAE P1 Floor (home team)	61.72%
CAE P2 Floor (home team)	61.52%
CAE P1 Floor (away team)	61.47%
Off Reb Per Opportunity P3 (home)	61.41%
CAE P1 Posession (home team)	61.35%
CAE P2 Floor (away team)	61.34%
P2P XPPS Increase P1 (home team)	61.27%
True Shooting % P1 (home team)	61.10%
Off Reb Per Opportunity P2 (away)	60.94%

Table 4.4: This table shows the player-level features and the prediction rate if that feature was the only one used in an SVM model with a RBF kernel.

Many of the single most important player-level features are context added metrics, which I had described in section 3.5. In particular, the context added metric which is most important is the context added efficiency per possession on the floor, a metric

which accounts for both the situational efficiency of players as well as the rewarding players who more frequently do better things for their team. Context added efficiency per possession used also shows up once, as does true shooting, a traditional measure of efficiency. Finally, how frequently a player rebounds, as well as how a player improves the shot selection of his team-mates (P2P XPPS Increase) also made the cut-off.

In table 4.5 below, I present the results of running the greedy feature selection algorithm.

Feature	Correct Prediction Rate
1. *CAE* P1 Floor (home team)	61.72%
2. *CAE* P1 Floor (away team)	62.15%
3. *CAE* P2 Floor (home team)	62.43%
4. Off Reb Per Opportunity P3 (home)	62.64%
5. *CAE* P2 Floor (away team)	62.96%
...	...
38. P2P *XPPS* Increase P2 (away team)	70.04%
39. Steal Per Opportunity P2 (away)	70.08%

Table 4.5: This table shows the player-level features and the prediction rate if that feature was used in conjunction with those above it in a SVM model with a RBF kernel.

As shown in table 4.5 above, there were 39 important features which were found using the greedy feature selection algorithm before the predictive capability of the model stopped improving. 8 of the selected features were the context added efficiencies per floor possession of different players, 6 were individual defense metrics for players, 5 of them were the offensive rebounding rate per opportunity of players, 4 of them were P2P *XPPS* Increase for different players, 4 of them were defensive rebounding rate per opportunity of players, 3 of them were assists per opportunity of players, 3 were Steal Per Opportunity, 3 were true shooting %, and 2 of them were minutes per game of players. The results when using logistic regression were quite similar as well. Furthermore, the features discovered in all of the models I attempted were similar to those found above.

4.4 Important Results

The work I present in my thesis is important for three main reasons: first, I present various new features derived from play-by-play data. Second, I show that many of these features are actually important in making predictions. Finally, I push the (public) state-of-the-art predictive capabilities when predicting outcomes of games in the NBA.

In chapter 3, I discussed the various features I derived in detail. In this section, I will discuss the second and third of the main contributions.

In section 4.2, I demonstrated that two of the team-level features I proposed, namely a new measure of clutch Performance and various team-to-team synergies are important in predicting outcomes of games. While these features only added marginal value in predicting outcomes when combined with more traditional features, the fact that they still added value suggests that they are at least somewhat important. In particular, demonstrating that clutch performance is somewhat important contradicts previous results that clutch performance can be explained by once factoring in other measures of team strength. Second, demonstrating that certain team-to-team synergies such as when a good 3pt shooting team is playing one that is poor at defending 3pt shooting suggests that team strength relationships aren't necessarily transitive. This insight not only has importance in making predictions, but could also be important to basketball teams when designing strategies to play other teams. In particular, when a "bad" team is playing a better team, the bad team could try to change its playing style to try to exploit these team-to-team synergies.

What's extremely encouraging is that even though the measures of clutch performance and team-to-team synergies that I propose are somewhat arbitrary (the thresholds I used for "good" 3pt shooting as well as when a time period is clutch was arbitrary), they are still useful in making predictions. It is very likely that even more features could be created that better measure these synergies, clutch performance, or other aspects of team-play that aren't yet known.

While clutch performance team-to-team synergies were important features that I proposed, some of the team level features I proposed did not have much predictive capability. For example, 3.2 which measures deviation from an ideal minute distribution did not have much predictive value in the models that I used. Furthermore, other features such as rest and distance travelled, both of which have predictive value in the NCAA, don't seem to have predictive value in the NBA. Perhaps distance travelled is not a good predictor of success because NBA teams have better and more comfortable modes of transportation, alleviating much of the physical burden that comes with travel, or perhaps we need better features to measure the impact of distance travelled. Furthermore, perhaps rest is not a good predictor of success because NBA teams plan playing time according to the schedule, so they play players fewer if they have an upcoming game soon. Hence, it could be entirely possible that teams play worse when they expect less rest than when they actually have less rest. Either way, the fact that certain features are not useful in predicting outcomes is not insignificant because they give us different insights or pave the way for better features.

In section 4.3, I demonstrated the usefulness of two sets of features that I proposed. The first was context added efficiency (CAE), a metric which incorporates the time left in the shot clock as well as general efficiency of the offensive and defensive team. The second set of features were features that measured how a player affects the shot selection of his team-mates, or his team-mates' $XPPS$.

CAE turned out to be the most important set of player-level features. This demonstrates that not only the outcome of a play is important, but also the context of that play is important, especially if we want to use the play to make future predictions. Furthermore, the fact that how a player alters his team-mates' $XPPS$ is useful in making predictions suggests that a certain set of players don't necessarily perform independently; rather, a player's performance can be highly dependent on his teammates'.

Finally, in both sections 4.3 and 4.2, I introduced models which outperform the

publicly available state-of-the-art predictors of NBA game outcomes. In particular, the team-level models increased predictive capability from 69.5% to 71.3%. While the team-level models do increase predictive performance, there does seem to be significant variance in the performance of the model from year-to-year. There is also much variance in the performance of the Vegas lines, as seen in table 4.6. While my model generally outperforms the Vegas predictions, it only outperforms the Vegas predictions in six out of the ten years. Furthermore, the performance of the model is not statistically significant: running a one-tailed paired t-test over the 10 seasons results in a p-value of 0.061, while running a two-tailed paired t-test results in a p-value of 0.123. Hence, it is possible that some of the positive results of my model occurred by chance. Running the model over more seasons of data could perhaps provide more conclusive results.

Year	SVM Performance	Vegas Performance
2003	73.45%	69.28%
2004	73.20%	68.52%
2005	72.45%	72.97%
2006	72.95%	69.63%
2007	69.95%	72.22%
2008	72.70%	67.04%
2009	69.45%	71.63%
2010	71.20%	67.10%
2011	70.20%	70.35%
2012	67.20%	66.65%

Table 4.6: The performance of the best SVM model and the performance of Vegas predictions from 2003 to 2012.

4.5 Future Work

While I advanced the public state-of-the-art predictive models in the NBA as well as created features which shine light into different aspects of basketball, there seems to be a lot of room to advance my work.

First, although some of the features I created were useful, they were still somewhat arbitrary. It is likely still possible to extract more meaningful features from play-by-

play data.

Second, I think that the learning techniques other than SVM and logistic regression largely failed because of two reasons: the multitude of features and poor feature selection. There were a multitude of features that were important, especially at the player level models. In a learning technique such as a Bayes' network where you attempt to learn meaningful connections between features, the space of these connections is an incredibly large space to search over, so finding the optimal set of connections would be computationally infeasible. Furthermore, learning this connections with a large feature space but only 10000 games often led to over-fitting when analyzing results using cross-validation. These problems could perhaps be alleviated by either coming up with more succinct features and limiting the feature spacc, or by creating better feature selection algorithms and limit the search space for connections.

Third, because greedy feature selection algorithms award features which incorporate the most information, I might have ignored features that I already have but which are more useful. This could also be addressed by using better feature selection algorithms; the genetic algorithms I attempted did not seem to perform better than the greedy ones.

Fourth, perhaps there is a fundamental limit to exactly how many games you could predict correctly, as some portion of the outcome is dictated by luck. Hence, focusing effort on other outcomes in the NBA, such as predicting the number of wins in a season, could lead to better insights about basketball.

Finally, as mentioned in section 1.2.3, a new set of data, visual tracking data, is starting to become more prevalently recorded in the NBA. This data is much richer than play-by-play data, and using this data correctly would inevitably lead to better insights.

Bibliography

[1] Basketball-reference. http://www.basketball-reference.com/, 2013.

[2] How to calculate wins produced. http://wagesofwins.com/how-to-calculate-wins-produced/, 2013.

[3] Sports betting odds, statistics and picks. http://www.covers.com, 2013.

[4] The warp rating system explained. http://www.sonicscentral.com/warp.html, 2013.

[5] Matthew Beckler and Hongfei Want. Nba oracle. http://www.mbeckler.org/coursework/2008-2009/10701_report.pdf.

[6] David J. Berri. *The Wages of Wins: Taking Measure of the Many Myths in Modern Sport.* Stanford University Press, September 2007.

[7] David J. Berri. Who is "most valuable"? measuring the player's production of wins in the national basketball association. *Manage. Decis. Econ*, 20:411–427, 1999.

[8] Paul Fearnhead and Benjamin Matthew Taylor. On estimating the ability of nba players. *Journal of Quantitative Analysis in Sports*, 47(3):1298, July 2011.

[9] Paul Kvam and Joel S. Sokol. A logistic regression/markov chain model for ncaa basketball. *Naval Research Logistics*, 53:788–803, 2006.

[10] Ian Levy. Hickory-high. http://www.hickory-high.com/?page_id=6195, 2013.

[11] Zach Lowe. *Lights, Cameras, Revolution.* http://www.grantland.com/story/_/id/9068903/the-toronto-raptors-sportvu-cameras-nba-analytical-revolution, 2013.

[12] Dragan Miljkovi and Ljubia Gaji. The use of data mining for basketball matches outcomes prediction. *International Symposium on Intelligent Systems and Informatics*, pages 309–312, 2010.

[13] Dean Oliver. *Basketball on Paper.* Potomac Books, 2004.

[14] Nate Silver. How we made our n.c.a.a. picks. http://fivethirtyeight.blogs.nytimes.com/2011/03/14/how-we-made-our-n-c-a-a-picks/, 2011.

[15] Wikipedia. Crossover (genetic algorithm). http://en.wikipedia.org/wiki/Crossover_%28genetic_algorithm%29, 2013.