ALEXANDER TECHNOLOGICAL INSTITUTE

OF THESSALONIKI

DEPARTMENT OF COMPUTER ENGINEERING

POST GRADUATE PROGRAM

WEB INTELLIGENCE

Applications of Machine learning in Daily Fantasy Sports

POST GRADUATE THESIS

of

KLEOMENIS CHATZIGEORGIOU

Supervisor: Konstantinos Diamantaras
Professor of ATEITH

Thessaloniki, October 2017

Υπόδειγμα φύλλου τίτλου (μπροστινή σελίδα) του αντιτύπου ΜΔΕ

Υπόδειγμα φύλλου τίτλου (μπροστινή σελίδα) του αντιτύπου ΜΔΕ

ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ - WEB INTELLIGENCE

Εφαρμογές Μηχανικής Μάθησης στα Φαντασιακά Αθλήματα

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΚΛΕΟΜΕΝΗ ΧΑΤΖΗΓΕΩΡΓΙΟΥ

Επιβλέπων : Κωνσταντίνος Διαμαντάρας
Καθητητής, Α.Τ.Ε.Ι.Θ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 1^{η} Οκτωβρίου 2017.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
....	....	....
Όνομα Επώνυμο	Όνομα Επώνυμο	'Ονομα Επώνυμο
Καθηγητής ΑΤΕΙ	Καθηγητής ΑΤΕΙ.	Καθηγητής ΑΤΕΙ.

Θεσσαλονίκη, Οκτώβριος 2017

(Υπογραφή)

....

ΟΝΟΜΑ ΕΠΩΝΥΜΟ

Τίτλος

Περίληψη

Τα φαντασιακά αθλήματα παρουσιάζουν μεγάλη αύξηση στην δημοτικότητά τους, κυρίως στις Ηνωμένες Πολιτείες Αμερικής και στον Καναδά. Επίσης με την εμφάνιση όλο και περισσότερων ιστότοπων που προσφέρουν ιστορικά δεδομένα για κάθε σπορ και αθλητή, είναι πιο εύκολο από ποτέ για κάποιον χρήστη να βρει στατιστικά για το άθλημα που τον ενδιαφέρει. Ταυτόχρονα μέθοδοι και αλγόριθμοι μηχανικής μάθησης είναι προσβάσιμοι στον μέσο χρήστη που χρησιμοποιεί έναν οικιακό προσωπικό υπολογιστή. Η προσβασιμότητα αυτή μαζί με την εξέλιξη της τεχνολογίας μπορούν να συνδυαστούν για την δημιουργία μοντέλων που προβλέπουν την επίδοση του εκάστοτε αθλητή και κατ' επέκταση να δώσουν στον χρήστη πλεονέκτημα σε κάποιο διαγωνισμό φαντασιακών αθλημάτων.

Σε αυτήν την διπλωματική εργασία αναπτύξαμε ένα μοντέλο μηχανικής μάθησης, το οποίο βασίζεται στις προηγούμενες επιδόσεις των αθλητών για να προβλέψει το φαντασιακό τους σκορ. Δοκιμάσαμε τρεις διαφορετικές μεθόδους μοντελοποίησης για παλινδρόμηση, Διανύσματα Υποστήριξης, Γραμμική Παλινδρόμηση και Μπεϊσιανή Παλινδρόμηση. Αναπτύξαμε επίσης με την χρήση δυναμικού προγραμματισμού έναν αλγόριθμο ο οποίος επιλέγει με βάση τις προβλέψεις μας τους καλύτερους δυνατούς παίκτες για κάποιον διαγωνισμό φαντασιακών αθλημάτων.

Για να εξετάσουμε την αποτελεσματικότητα των μοντέλων μας, τα δοκιμάσαμε σε τυχαίους διαγωνισμούς της περιόδου 2016-2017 του ιστότοπου www.draftkings.com. Επίσης συγκρίναμε τις προβλέψεις μας με υπηρεσίες που προσφέρονται από διάφορους ιστότοπους. Τα αποτελέσματα ήταν αρκετά ενθαρρυντικά. Στις περισσότερες περιπτώσεις οι προβλέψεις του μοντέλου μας θα κέρδιζαν τους εκάστοτε διαγωνισμούς και ήταν καλύτερες συγκριτικά με τις προβλέψεις διαφόρων ιστότοπων.

Λέξεις Κλειδιά: <<Μηχανική Μάθηση, Παλινδρόμηση, Φαντασιακά Αθλήματα>>

Abstract

Fantasy sport's popularity is growing rapidly especially in United States of America and Canada. Moreover with the emergence of sport analytics and sports data related websites, it's now easier than ever to find historical data for any given sport or athlete. At the same time machine learning algorithms and methods are accessible and usable by any user on an average computer. This accessibility on data and technology can be combined to create prediction models that can be used to estimate a player's performance and potentially give users a competitive advantage in daily fantasy sports contests.

In this thesis we developed a machine learning model that predicts fantasy performance of NBA players, based on their past performance. We tried three different regression approaches, Bayesian Regression, Linear Regression and Support Vector Regression. We also developed an algorithm that uses dynamic programming to generate a daily fantasy sports lineup based on our model's predictions.

We ended up using Bayesian Regression for our model due to better produced results compared to the other approaches. Next we generated lineups for random contests of NBA 2016-2017 regular season. We also compared our results with the predictions of websites that offer similar services. Through our tests we managed to produce several lineups that could potentially return profit in the long term and at the very least offer a tool that can be used by daily fantasy players to get an edge over their competition.

Keywords: <<Machine Learning, Daily Fantasy Sports, Regression>>

Table of Contents

1

Introduction

1.1 Data Analysis in Sports

The evolution of technology and the increased accessibility of the internet have lead data analysis to become an integral part of professional sports. Data analysis in the current state of the technology allows experts, to track every kind data imaginable and use it to analyze games and players. As a result there is a constant increase in Professional teams' prioritization of statistics and data when making decisions. Nowadays every major professional sports team has analytics department or analytical experts on their staff. Moreover sports fans show an immense interest in analytical content. For example www.basketball-reference.com, a website that collects historical basketball data, has almost four million unique visitors every month.

This rise in popularity of data analysis and statistics has made easier for researchers and gamblers to use historical data in order to predict sports outcomes. In addition the increase of computing power of the average computer allows users to develop and run their own prediction models and use them in order to gain a competitive edge.

1.2 Fantasy Sports

Fantasy sports is a type of online game where participants assemble an imaginary team of real athletes of a professional sport. These teams compete with each other based on the actual statistical performance of the athletes. Participants form leagues consisting from five to twenty people and lasts for twenty weeks. Scoring rules are set beforehand and can vary based on participants experience and competitiveness of the league. Fantasy sports are really popular in United States of America with around 56 million people playing them [1]. Almost 20% of adult men in U.S.A participate in a fantasy sports league and around 10% of adult women.

Even before the internet, there were a few sports enthusiasts, playing fantasy sports using a paper and a pencil, but the internet helped transform this hobby into a billion dollar industry. There are 2 ways that people play fantasy sports. The more traditional one doesn't involve any gambling (or includes a small prize pool for the winner) and includes a league of usually 8-10 competitors who hold a "draft" before every NBA/MLB/NFL season. During this draft the participants have a limited amount of virtual resources (usually a salary cap) available to them. Using these resources, each competitor selects a virtual team, compromised of real athletes. Fantasy competitors then face each other in heads-up games every week. The scoring is dictated by the statistical in game performance (i.e. rebounds, touchdowns, steals, points, etc.) of the athletes in their actual games. The challenge of fantasy sports is to maximize the value of your resources, by selecting players who provide good statistical performance relative to their price in the draft.

The daily format is a bit different from the traditional format. In daily fantasy sports people compete with each other in tournaments that last only 1 day. In websites like DraftKings and FanDuel, users draft a fantasy team in their sport of choice and pay an entry fee to submit that team into a pool of other teams. Those websites offer contests which hundreds of people can enter at the same time and potentially win from a multiple of their buy-in as a payout to thousands of dollars.

1.2.1 Industry Background

During the year 2003 the Travel Channel on American cable television started airing World Series of Poker inaugural season. World Series of Poker (WSOP) was a poker tournament with a buy-in of 10,000 $ and 10 million dollars prize pool. Moreover the tournament gave the chance to players to qualify for the main even by winning small online tournaments held on platforms like PokerStars. The winner of WSOP first season was Christopher "moneymaker" Bryan who qualified for the tournament by winning an online poker tournament with an entry fee of 39$. He was an accountant and an amateur poker player and won over 2.5 million dollars on the first season of WSOP. It was also the first live tournament that he participated. His win caused a huge growth in online poker's popularity around the world. Online poker doubled its numbers every year during the 2003-2006 period. That period is called "poker boom" [2].

That growth of the online poker industry came to an end at 13th of October 2006 when American government signed The Unlawful Internet Gambling Enforcement Act (UIGEA). This bill basically made the online poker websites illegal but at the same time contained explicit language that legalized daily fantasy sports. Since online gambling was pretty popular in the U.S.A and with online poker reaching its "golden age", there was a huge gap in the industry after the sign of UIGEA. That gap was filled successful with the introduction of Daily Fantasy Sports, since people were playing fantasy sports for years, just in a different format.

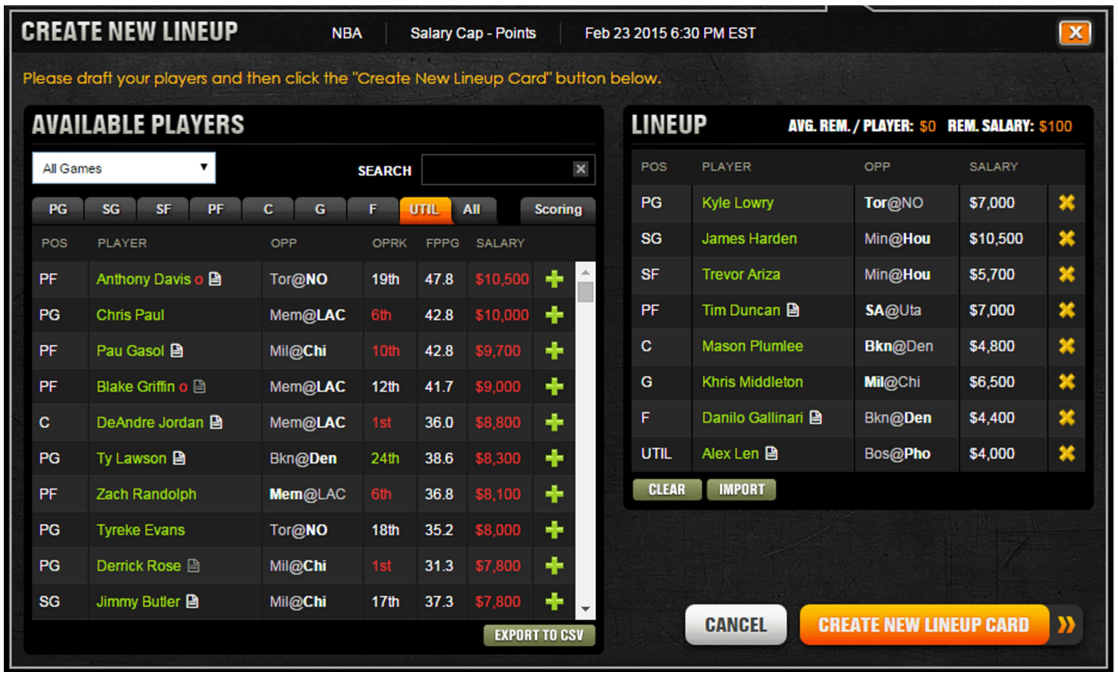
The first DFS website (Fantasy Sports Live) launched in 22 June 2007, followed by SnapDraft of NBC 1 year later and Fanduel in January of 2009. In 2013 Travis "Tspieldo" Spieth became the first DFS millionaire by winning 1 million dollars from a single contest. After that DFS kept growing fast and today every major sports league in the U.S.A has a DFS site as their official partner. DraftKings partnered with MLB, UFC and NHL and DraftDuel with NBA. In 2016 the revenue for DFS websites was around 3.6 billion dollars with DraftKing controlling more than 90% of the market. The market is projected to be at 4.8 billion dollars in 2020.

While daily fantasy sports have enjoyed explosive growth since 2013, the industry's origins can be traced back more than a decade. The concept for DFS was discussed throughout the

2000's, with the first gaming sites being founded in 2007. Since that time, sites have raised hundreds of millions of dollars in venture capital funding, while market leaders FanDuel and DraftKings have developed official business relationships with the NBA, MLB, NHL, UFC and dozens of professional sports franchises, as well as major corporations such as NBC and Comcast.

1.2.2 Scoring and Format

As described above, in DFS the users have a set amount of virtual currency available, which they use to draft real life players and enter a pool with other teams in order to compete for a prize. The contests last 1 day and the same players can be drafted by multiply users. Users can submit or modify their lineups until 5 minutes prior to the start of the first game of the day.

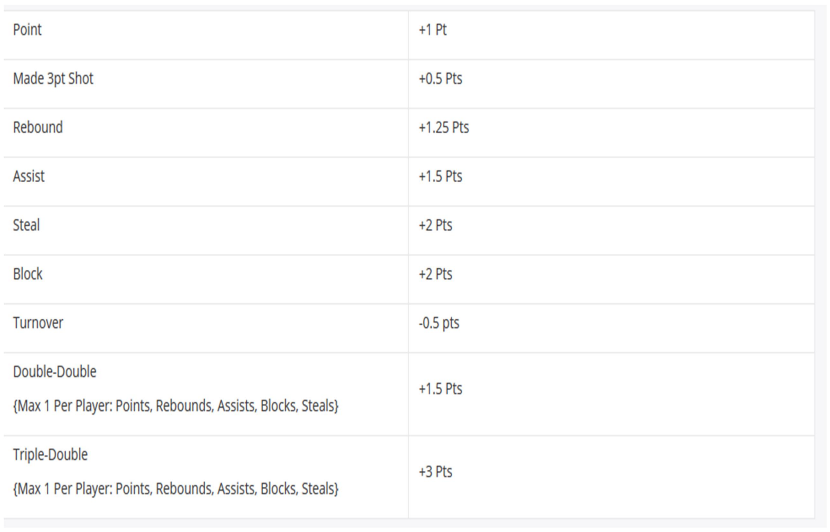


POS	PLAYER	OPP	OPRK	FPPG	SALARY
PF	Anthony Davis o	Tor@NO	19th	47.8	$10,500
PG	Chris Paul	Mem@LAC	6th	42.8	$10,000
PF	Pau Gasol	Mil@Chi	10th	42.8	$9,700
PF	Blake Griffin o	Mem@LAC	12th	41.7	$9,000
C	DeAndre Jordan	Mem@LAC	1st	36.0	$8,800
PG	Ty Lawson	Bkn@Den	24th	38.6	$8,300
PF	Zach Randolph	Mem@LAC	6th	36.8	$8,100
PG	Tyreke Evans	Tor@NO	18th	35.2	$8,000
PG	Derrick Rose	Mil@Chi	1st	31.3	$7,800
SG	Jimmy Butler	Mil@Chi	17th	37.3	$7,800

POS	PLAYER	OPP	SALARY
PG	Kyle Lowry	Tor@NO	$7,000
SG	James Harden	Min@Hou	$10,500
SF	Trevor Ariza	Min@Hou	$5,700
PF	Tim Duncan	SA@Uta	$7,000
C	Mason Plumlee	Bkn@Den	$4,800
G	Khris Middleton	Mil@Chi	$6,500
F	Danilo Gallinari	Bkn@Den	$4,400
UTIL	Alex Len	Bos@Pho	$4,000

Image 1. DraftKings drafting interface.

In the image 1 we can see the interface of DraftKings when a user drafts his lineup for an NBA contest. Except from the salary restriction there is also a restriction in the position of the drafted players. So for example in DraftKings, users have to Draft one point guard, one shooting guard, one small forward , one power forward, one center, one guard (point guard or shooting guard) and one forward (small or power forward). The last spot can be filled with a player who is eligible for any position.



Point	+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 {Max 1 Per Player: Points, Rebounds, Assists, Blocks, Steals}	+1.5 Pts
Triple-Double {Max 1 Per Player: Points, Rebounds, Assists, Blocks, Steals}	+3 Pts

Image 2. Drafkings NBA scoring.

Image 2 shows the scoring format on DraftKings website. For every point a player of a user's drafted team scores, user gains 1 fantasy point, for every assist 1.5 fantasy points, for every steal 2 fantasy points etc. After all games are finished, the fantasy points of the players that were drafted are summed up. This sum is the line up's total fantasy score.

The prize pool, entry fee and the number of participants vary for each contest. There are tournaments with entry fees as low as 0.5 dollars and there are tournaments that require thousands of dollars to enter. The final prize pool is determined by the entry fee. Usually the website keeps 15% of the user's entry fee and the rest goes to the prize pool. Tournaments can have from ten to thousands of participants. A big player pool translates into bigger prize for the winners, but at the same time smaller chance of placing on the top ranks.

1.2.3 Fantasy Contests

There are two different types of contests in daily fantasy sports: Cash games (also called 50/50 or double up) and Guaranteed Prize Pool (GPP).

- Cash games are tournaments where the user competes for a chance to double his entry money. The top 50% participants double their entry fee and the rest don't win anything. In a player pool consisted of 100 participants the prize will be the same for the participant who finished first and the one who finished at 50th place.
- Guaranteed prize pool contests have a guaranteed prize which doesn't dependent on participant's entry fee. The big differences with cash games are the scaling prizes and the much smaller percentage of places which get paid. In GPP contests the first place wins almost 15-20% of the prize pool and each place bellow wins a smaller amount. The percentage of player pool that wins is usually around 20%.

Each type of contest requires a different strategy in order to succeed and have different amount of luck and variance involved. Cash games tend to have less variance because of bigger amount of participants getting paid. On the other hand GPP games require some luck but at the same time a good placement results in high profit.

1.3 Project Goal

The goal of this thesis is to develop an algorithm that will provide a competitive advantage over the average DFS player and translate this advantage into profit. Let's assume that we submit 10 different line ups, generated by our models, in 10 different 50/50 contests with an entry fee of 5$ per contest. For every contest we win we get 9.5$ back. Winning 6 out of 10 contests will translate into 7$ or 11.4% increase of our original capital. This means that maintaining a 55% win rate will ensure profit in the long term.

To achieve that we used machine learning in order to predict players performance for each game they will play. Next we created a lineup optimizer, which generates the optimal lineup for a particular date based on our previous predictions.

We focused on NBA daily fantasy 50/50 or "cash" contests for www.draftking.com. We chose NBA because basketball is considered a more predictable sport compared to other popular sports like American football (NFL) and Baseball (MLB). The reason we chose to emphasize on cash contest is that compared to regular tournament (GPP) games, cash games have lower variable and a lot less luck involved in the final result. Although that doesn't mean that our predictions can't be used for GPP games but the results are better and more consistent on cash games. Moreover since every fantasy website has different scoring rules, we focused on one website but the model could be used for other fantasy websites with a few tweaks on the database.

To achieve our goal we collected statistical data from 2016-2017 season for each NBA player. Using that data we developed a regression model in python. The model was tested on 2016-2017 NBA season. To measure our success we set a target score of fantasy points that our lineup should achieve in order to win a contest. Then we chose random dates of 2016-2017 season and generated a lineup based on our model's predictions. Furthermore we compared our model predictions with web applications that offer daily fantasy sports predictions to their users.

1.3.1 Contribution

- We studied and researched various daily fantasy sports communities in order to find which kind of data would be useful in a prediction model.
- We tested different statistical categories and their impact on an NBA player's fantasy performance.
- We tested different regression models to find out which one can be used on daily fantasy score predictions.
- We developed a machine learning algorithm which predicts the daily fantasy score of an NBA player on a given date, based on the athlete's past performance.
- We developed a python algorithm that generates the optimal line up for a given date, based on our machine learning model's prediction.

- We evaluated our models and found out that we can maintain a positive winning rate on draftkins NBA cash contests, which can translate into profit in the long term.

2

Related Works

There are many DFS players that use one of the many DFS sites available that offer predictions for upcoming fantasy contests, such as https://www.fantasycruncher.com, https://www.fantasypros.com , https://dailyfantasynerd.com, https://www.rotoql.com/#oid=20119_1662. The most successful fantasy players develop their own prediction models which they use in order to gain advantage over the rest of the players. For example top fantasy player for 2015 Saahil Sud, has created his own prediction models and techniques[3].Moreover it is hypothesized that the DFS sites using something similar in order to define the price of the players in their DFS games. But due to the competitive nature of the game the code of these models isn't publicly available. As a result there wasn't any well documented work which we could study or base our model on.

Even though there are not prediction models available to the public, there is a plethora of online communities which discuss daily fantasy sports. We used those communities to find information about how each statistical category or any other parameter affects a player's fantasy performance. Some of those communities are:

- Reddit/DFSports: Reddit (www.reddit.com) is a social network platform designed to allow users to share web content and discuss. Content is organized by areas of interest called subreddits. There are over 1 million of subreddits. The subreddit DFSports (www.reddit.com/DFSports) is an online community dedicated to discussions around daily fantasy sports. It has 14 thousands of subscribers and even more readers. There are daily discussions about strategies for daily fantasy sports and links to useful resources.
- ***RotoGrinders:*** Rotogrinders (www.rotogrinders.com) is a website with various contest related to daily fantasy sports. Their contest includes strategy suggestions, player's analysis, player ranking, etc. They also provide a discussion forum where users can exchange tips or discuss about strategies in upcoming contests.
- RotoWorld: Similar to Rotogrinders, Rotoworld (www.rotoworld.com) is a website dedicated on daily fantasy sports. It has content that include fantasy guides, strategy suggestions, daily analysis of upcoming contests and almost real time news about injury status of players. They also provide a discussion forum.

3

Theoretical Background

3.1 Using machine Learning in DFS

The current state of the technology allows experts, to track every kind of data imaginable and use it to analyze games and players. As a result, there are new kinds of statistical categories created that can help measure and analyze players' performance. In NBA for example, advanced statistics like Player Efficiency Rating (PER), Win shares, Offensive/Defensive Efficiency, Points per Possession were introduced trying to provide a better platform for experts to analyze the game and also for coaches to measure their team's performance and try to improve it. But at the same time those stats can be used as a really helpful tool for predictions.

Machine learning offers a great tool to use for developing prediction algorithms. With the amount of data that is publicly available online, users can create datasets with up to date statistics for each player and use regression algorithms to make predictions.

The average user of a DFS website uses the website's predicted scores as their metric. The relationship between player salaries and their predicted scores is close to linear. So if a prediction model manages to beat those generic predictions that DFS websites provide, a participant using this model can find undervalued players and get an edge on the average user.

3.2 Machine Learning

According to Arthur Samuel, machine learning gives "computers the ability to learn without being explicitly programmed". It's basically the ability of machines (computers) to mimic human behavior and improve their performance. Using machine learning a system's performance can be increased through examples and past data, which results on less time and resources spent.

Machine learning is widely used to solve classification, clustering and regression problems using different kinds of algorithms. Those algorithms can be grouped in two categories based on their learning style:

- Semi-Supervised Learning: Traditional classifiers use only labeled data (feature / label pairs) to train their model. However labeled instances are often difficult, expensive or time consuming to obtain as they require the efforts of experienced human annotators. Meanwhile unlabeled data may be relatively easy to collect, but there has been few way to use them. Semi-supervised learning addresses this problem by using large amount of unlabeled data, together with labeled data, to build better classifiers [4].

- Supervised machine learning algorithms use externally supplied instances to produce general hypotheses, which then make predictions about future instances. In other words, the goal of supervised learning is to build a concise model of the distribution of class labels in terms of predictor features. The resulting classifier is then used to assign class labels to the testing instances where the values of the predictor features are known, but the value of the class label is unknown [4]

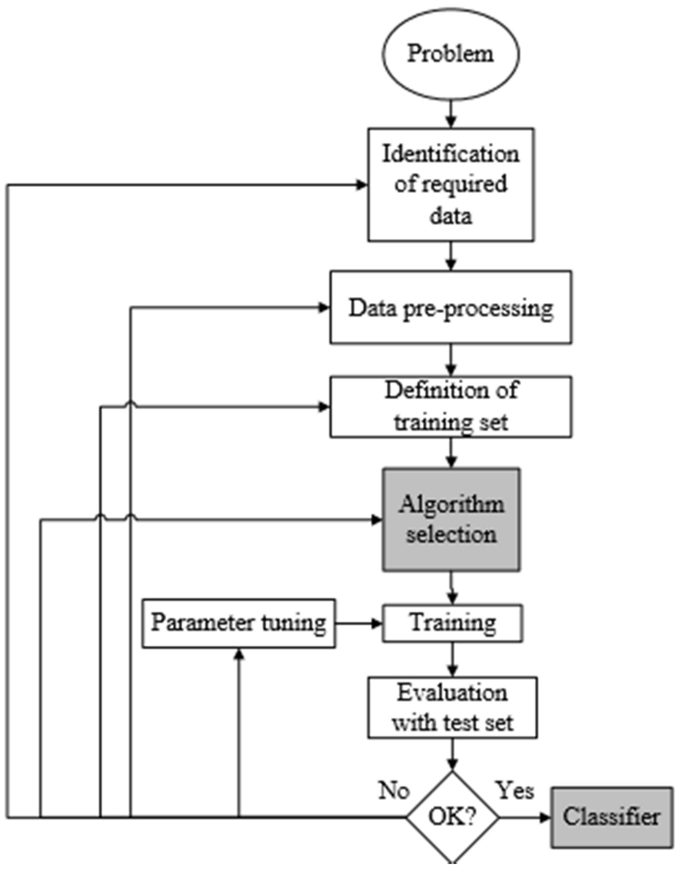


Image 4. The process of supervised learning [4].

- Unsupervised learning: In unsupervised learning the input data is not labeled and does not have a known result. The algorithm tries to classify the input data without prior knowledge or category information. Unsupervised learning algorithms have often been criticized for trying to solve a particular task in harder way than is necessary [5].

Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or infeasible; example applications include email filtering, detection of network intruders or malicious insiders working towards a data breach, optical character recognition learning to rank, and computer vision. Applications for machine learning include :

- Adaptive websites
- Affective computing
- Information retrieval
- Natural language understanding and processing
- Search engines
- Online advertising
- Speech recognition
- Economics
- Financial market analysis
- Translation

3.3 Regression

The problem we had to solve was a regression problem. In statistics, regression analysis includes any techniques for modeling and analyzing trends and relationships between a dependent variable and one or more independent variables. It's used to make predictions and forecast future results based on known data. There are different machine learning algorithms used to solve regression problems. Some of them are:

- Support Vector Regression (SVR)
- Ordinary Least Squares Regression (OLSR)
- Linear Regression
- Logistic Regression
- Stepwise Regression
- Multivariate Adaptive Regression Splines (MARS)
- Locally Estimated Scatterplot Smoothing (LOESS)

3.3.1 Linear Regression

Linear regression is a technique used to model a single response (outcome) variable based on one or more input variables. With linear regression we assume that there is a linear relationship between input and response variables[9].

Linear regression and regression models in general have two main objectives:

- Establish if there is a relationship between two variables. More specifically, establish if there is a statistically significant relationship between the two. For example we can establish if there is a statistical relationship between income and spending, wage and gender, student height and exam scores, etc.

- Forecast new observations. This means that we can use what we know about a relationship to forecast unobserved values. For example predict the sales over the next quarter based on the recent growth of sales and the growth rate.

In regression models variables can play two different roles. They can be a dependent variable or and independent variable. Dependent variable is the one that we want to forecast or explain and its value depends on some other variable. We denote the dependent variable as y. Independent variable is the variable that is used to explain or predict the other one. Its value is independent and we denote it as X

Mathematically, we can write a linear relationship as:

$$Y= \beta_0 + x\beta_1 + \varepsilon$$

Where:

- y = the dependent variable or our target
- x is the independent variable
- β_1 is x's slope or coefficient
- β_0 is the constant or intercept
- ε is the error term

3.2.2 Support Verctor Machines (SVM)

Support Vector Machines (SVM) are supervised learning models used in machine learning that analyze data for classification and regression analysis. The goal of SVM is to design a hyperplane that classifies all training vectors into two classes, while the hyperplane leaves maximum margin from both classes. Margin is the distance between the hyperplane and the closest element from each class.

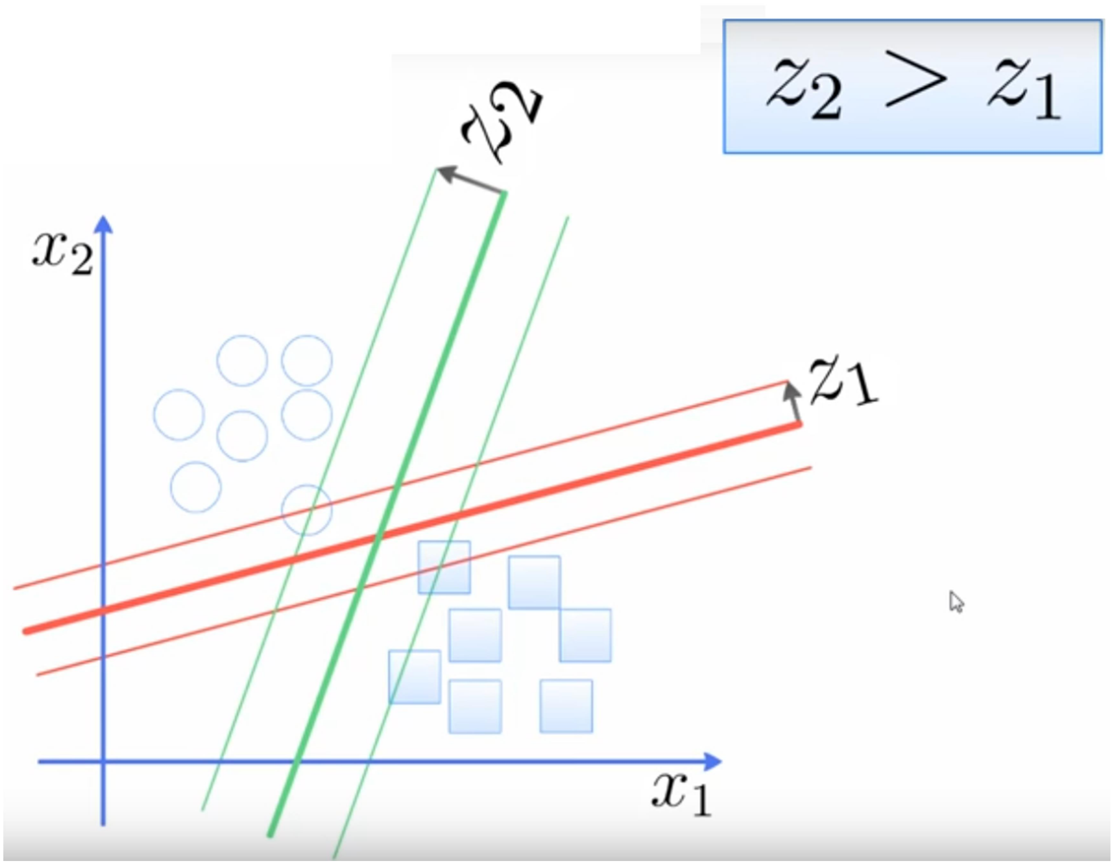


Image 5. SVM Example

In the image 5 we got two features x1 and x2, different elements to separate and also two different hyperplanes. We want to classify the elements into class square or class circle. Z1 and Z2 are the distance between the hyperplane and the closest element to it. This distance is called margin. Both hyperplanes classify the elements correctly but with SVM we want to find the optimal solution, the hyperplane with the highest margin. In this case we can see clearly that Z2 margin is bigger than Z1 which makes the green hyperplane the optimal solution.

We can describe SVM with a simple classification problem of two linear seperatable classes $\mathcal{C}_0, \mathcal{C}_1$. Assuming there is a separation function

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

for which we got a vector w and a polarization w_0, then:

$$g(\mathbf{x}) \begin{cases} < 0, & \alpha\nu\ \mathbf{x} \in \mathcal{C}_0 \\ > 0, & \alpha\nu\ \mathbf{x} \in \mathcal{C}_1 \end{cases}$$

The total of elements x for which

$$g(\mathbf{x}) = 0$$

define the separation surface. These are the elements we can't classify in class $\mathcal{C}_0$ or in class $\mathcal{C}_1$.

There isn't a single solution for this problem since we can separate those elements with infinite number of pairs w and w_0. In order to find the optimal solution we need to define an assessment criterion. In SVM we use margin γ.

The method of Support Vector Classification can be extended to solve regression problems. This method is called Support Vector Regression.

3.2.3 Random Forests

Random forests are an ensemble learning method of tree predictors where each tree depends on the values of a vector sampled independently and with the same distribution for all trees in the forest[8].

In order to achieve diversity among base decision trees, each tree is generated with the following method: If the number of records in the training set is N, then N records are sampled at random but with replacement, from the original data. This is a bootstrap sample. This sample is used as a training set for growing the tree. If there are M input variables, a number m << M is selected such that at each node, m variables are selected at random out of M and the best split on these m attributes is used to split the node. The value of m is held constant during forest growing. Each tree is grown to the largest extent possible. There is no pruning. Using these steps, random forests method ensures that multiple trees affect the forest. Once the forest is trained, random forests use all trees in order to classify a new instance. When a tree classifies a new instance, the result is recorded as a vote. After the new instance runs through all the trees in the forest, the votes are counted and the instance is classified to the class with the most votes[7].

In a tree of classifiers $\{ h (x,\Theta_k), k = 1 , \ldots\}$ the Θ_k are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x.

Random forests for regression are formed by growing trees depending on a random vector Θ such that the tree predictor $h(x, \Theta)$ takes on numerical values as opposed to class labels. The output values are numerical and we assume that the training set is independently drawn from the distribution of the random vector Y, X. The mean-squared generalization error for any numerical predictor h(x) is:

$$\mathbf{E_{X,Y}\,(Y-h(X))^2}$$

Random forests method can be an effective tool in prediction or classification due to the Law of Large Numbers theorem which indicates that the average of the results obtained from a large number of trials should be close to the expected value and will become closer as the volume of trials increases. Random forests does exactly what this theorem describes, by having multiply trees to vote for an instance and taking the result with the most votes as the correct one. Moreover forests give results competitive with boosting and adaptive bagging without changing the training set and they act to reduce bias.

3.3 Python

Python is a high level programming language. Python emphasize code readability and a syntax which allows programmers to express concepts in fewer lines of code than might be used in object oriented languages like Java or C++.

Python has a large standard library, commonly cited as Python's greatest strengths, providing tools suited to many tasks. Python's main feature is its versatility, as its interpreters are available for installation in many operating systems.

3.3.1 Scikit-learn

For this project we used scikit-learn. Scikit-learn is a Python module integrating a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems. This package focuses on bringing machine learning to non-specialists using a general-purpose high-level language. Emphasis is put on ease of use, performance, documentation, and API consistency. It has minimal dependencies and is distributed under the simplified BSD license, encouraging its use in both academic and commercial settings. Source code, binaries, and documentation can be downloaded from http://scikit-learn.sourceforge.net.

Scikit-learn harnesses this rich environment to provide state-of-the-art implementations of many well known machine learning algorithms, while maintaining an easy-to-use interface tightly integrated with the Python language. This answers the growing need for statistical data analysis by non-specialists in the software and web industries, as well as in fields outside of computer-science, such as biology or physics. Scikit-learn differs from other machine learning toolboxes in Python for various reasons: i) it is distributed under the BSD license ii) it incorporates compiled code for efficiency, unlike MDP (Zito et al., 2008) and pybrain (Schaul et al., 2010), iii) it depends only on numpy and scipy to facilitate easy distribution, unlike pymvpa (Hanke et al., 2009) that has optional dependencies such as R and shogun, and iv) it focuses on imperative programming, unlike pybrain which uses a data-flow framework. While the package is mostly written in Python, it incorporates the C++ libraries LibSVM (Chang and Lin, 2001) and LibLinear (Fan et al., 2008) that provide reference implementations of SVMs and generalized linear models with compatible licenses. Binary packages are available on a rich set of platforms including Windows and any POSIX platforms.

Furthermore, thanks to its liberal license, it has been widely distributed as part of major free software distributions such as Ubuntu, Debian, Mandriva, NetBSD and Macports and in commercial distributions such as the "Enthought Python Distribution"

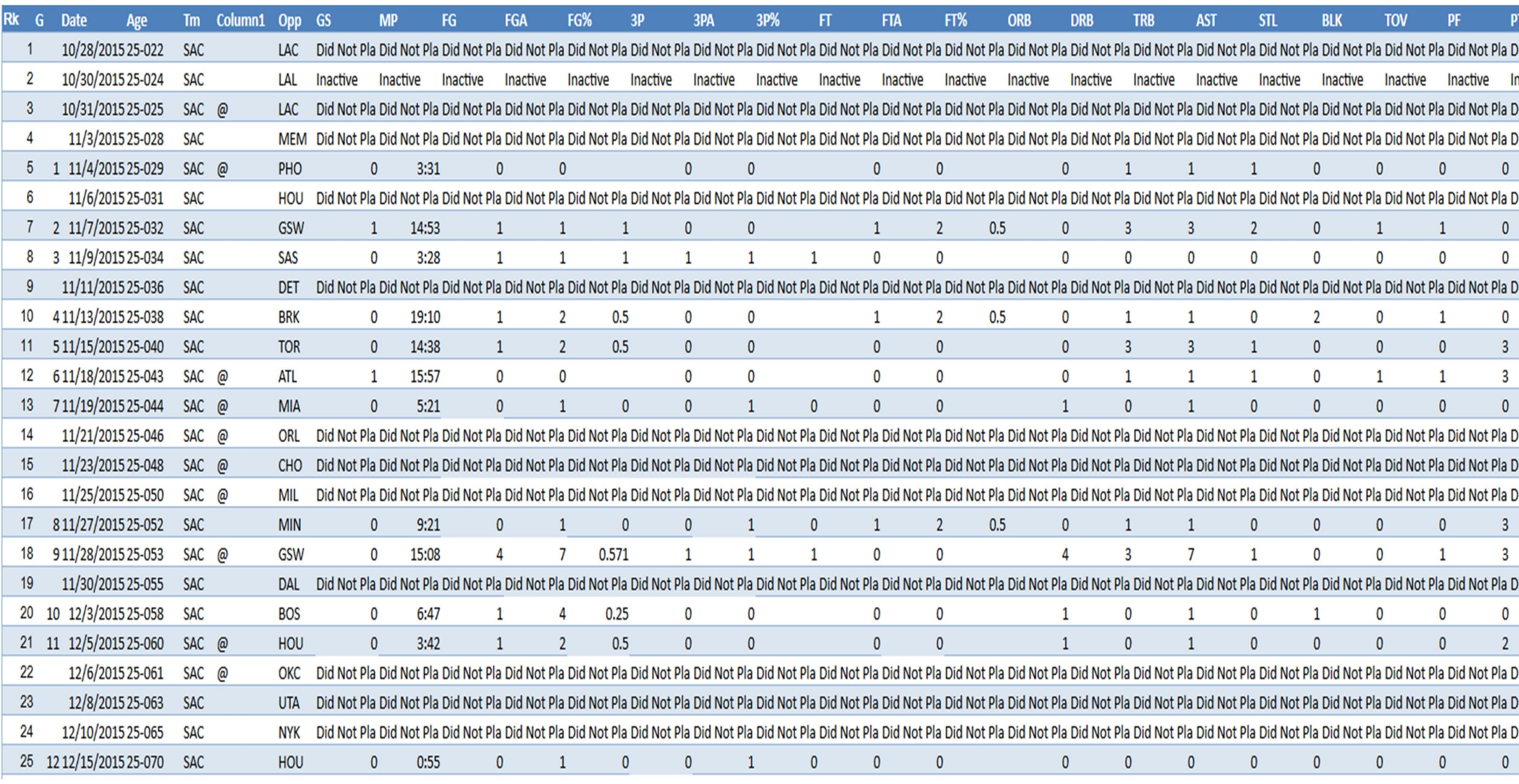
4

Development

4.1 Creating the dataset

In order to create a model that predicts the fantasy points for each player, first we had to create a dataset. The dataset had to be composed of the statistical values that constitute the fantasy score. Fantasy score in daily fantasy basketball is a sum of points, rebounds, steals, blocks, three points made and free throws. So the sum of the above real life statistics accumulated by a player is the value that we are trying to predict.

To create the dataset we parsed data from www.DFSgold.com. For each NBA player we collected the statistics we mentioned above for every game he played during the 2016-2017 NBA regular season. Next we created a CSV file for each player and put the data there as seen in the image 6 below.



Rk	G	Date	Age	Tm	Column1	Opp	GS	MP	FG	FGA	FG%	3P	3PA	3P%	FT	FTA	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	P
1		10/28/2015	25-022	SAC		LAC	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	D
2		10/30/2015	25-024	SAC		LAL	Inactive	Inactive	Inactive	Inactive	Inactive	Inactive	Inactive	Inactive	Inactive	Inactive	Inactive	Inactive	Inactive	Inactive	Inactive	Inactive	Inactive	Inactive	Inactive	In
3		10/31/2015	25-025	SAC	@	LAC	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	D
4		11/3/2015	25-028	SAC		MEM	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	D
5	1	11/4/2015	25-029	SAC	@	PHO	0	3:31	0	0		0	0		0	0		0	1	1	1	0	0	0	0	
6		11/6/2015	25-031	SAC		HOU	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	D
7	2	11/7/2015	25-032	SAC		GSW	1	14:53	1	1	1	0	0		1	2	0.5	0	3	3	2	0	1	1	0	
8	3	11/9/2015	25-034	SAC		SAS	0	3:28	1	1	1	1	1	1	0	0		0	0	0	0	0	0	0	0	
9		11/11/2015	25-036	SAC		DET	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	D
10	4	11/13/2015	25-038	SAC		BRK	0	19:10	1	2	0.5	0	0		1	2	0.5	0	1	1	0	2	0	1	0	
11	5	11/15/2015	25-040	SAC		TOR	0	14:38	1	2	0.5	0	0		0	0		0	3	3	1	0	0	0	3	
12	6	11/18/2015	25-043	SAC	@	ATL	1	15:57	0	0		0	0		0	0		0	1	1	1	0	1	1	3	
13	7	11/19/2015	25-044	SAC	@	MIA	0	5:21	0	1	0	0	1	0	0	0		1	0	1	0	0	0	0	0	
14		11/21/2015	25-046	SAC	@	ORL	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	D
15		11/23/2015	25-048	SAC	@	CHO	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	D
16		11/25/2015	25-050	SAC	@	MIL	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	D
17	8	11/27/2015	25-052	SAC		MIN	0	9:21	0	1	0	0	1	0	1	2	0.5	0	1	1	0	0	0	0	3	
18	9	11/28/2015	25-053	SAC	@	GSW	0	15:08	4	7	0.571	1	1	1	0	0		4	3	7	1	0	0	1	3	
19		11/30/2015	25-055	SAC		DAL	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	D
20	10	12/3/2015	25-058	SAC		BOS	0	6:47	1	4	0.25	0	0		0	0		1	0	1	0	1	0	0	0	
21	11	12/5/2015	25-060	SAC	@	HOU	0	3:42	1	2	0.5	0	0		0	0		1	0	1	0	0	0	0	2	
22		12/6/2015	25-061	SAC	@	OKC	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	D
23		12/8/2015	25-063	SAC		UTA	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	D
24		12/10/2015	25-065	SAC		NYK	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	Did Not Pla	D
25	12	12/15/2015	25-070	SAC		HOU	0	0:55	0	1	0	0	1	0	0	0		0	0	0	0	0	0	0	0	

Image 6. Quincy Acy's statistics for 2015-2016 season

Except of the 6 statistical categories that we used in daily fantasy basketball scoring, our first dataset contained some more data, like date, age of the player, opponent, minutes played, etc. Some of that data was used in the final excel files (like minutes played) and some like Age and 3PT% was not included. Since there is no documentation of similar work we couldn't predetermine which data would be useful for our model. Due to the lack of available information on algorithms and models that other fantasy players use, there was no way to know the impact of each statistical category on the final prediction. In order to determine which data would have positive impact in our predictions we used the trial and error method. We run our prediction model with different datasets and evaluated the success of the model after each run.

The statistical categories we ended up using were minutes played, points, field goals made, rebounds, assists, steals, turnovers, fouls and projected minutes. For every NBA player there was a CSV file with those statistical categories and the date of the game.

After trying different statistical categories we found out that minutes played was the statistical category that had the most impact in model's predictions. We run a lot of different tests with a variety of statistical categories, while not including minutes played. In every run the results were extremely inaccurate and the predictions seemed completely random. Only when we included minutes played as a feature, predictions started to be accurate.

4.2 Predicted minutes

As we mentioned above, minutes played was the statistical category that had the most impact in our predictions. But there isn't any way to be able to know how many minutes a player is going to play before the game starts. As a result we had to find a way to create a good estimation of the minutes a player is going to play on his next game.

The majority of NBA teams have some kind of fixed rotations during the regular season. There are players that are considered core which are usually 6-8 depending on the depth of the team and there are players called "bench warmers" who only play when the result of the game is already decided. Those core players are the players that usually score the most fantasy points. So those are the players whose minutes we want to predict.

Assuming that a player is available to play and he is not injured, or not excluded from the team's line up in order to get some rest, there are 4 factors that affect core players minutes during a game: injury, foul trouble, overtime and the "blowout" factor.

- "Blowout": Blowout is a term that is used to describe an easy one-sided victory. It occurs when one athletic team or individual performer outscores another by a large margin, in such fashion that allows the second team or individual little chance of a victory from a point early in a competition, game, contest or event. For example at December 26 2015 Cleveland cavaliers were playing against Portland Trailblazzers. LeBron James, Cleveland Cavalier's best player who was averaging 39.09 minutes per game in the last 5 games, was on the floor for only 25.19 minutes. He also scored only 23 fantasy points while he was averaging 46 in his last 5 games. The reason was that the Trailblazzers were winning by 20 points after the first period and they kept their lead for the rest of the game ending up winning by 29 points (105-76). Jared Cunningham, a bench warmer of Cavalier's was the one who benefited the most from this blowout situation. He did not play a single minute in the last two games and in the next game but in this game he was on the floor for 12 minutes. He also played less than 2 minutes in the next 4 of 6 Cavalier's games. But even though he played 12 minutes, he scored only 11 fantasy points. This example helps to understand that blowouts can cause a significant drop in a core player's fantasy value but at the same time the improvement on bench warmers is not that significant to make them worth including in a lineup.

- Injuries: Injuries are and will always be a part of a competitive athletic event. If an NBA player gets injured before the game starts, teams make public announcements and fantasy players can exclude this player from any lineup they will potentially submit. But an in game injury can occur, even early in a game and sadly there is no way to predict that.

- Foul Trouble: Foul trouble is a phrase used in basketball to describe a player's high foul total. In NBA players are allowed up to 6 fouls per game. After they exceed that they are forbidden to participate for the rest of the game. NBA coaches usually will put a player on the bench when he commits too many fouls early in the game (2 or more in first quarter, 3 or more in second quarter, etc). This obviously can cause a drop on player's minutes and consequently in the fantasy points that they will score. Even though there are some patterns that can be used to determine whether a player is on risk of getting in foul trouble, like which team they face, which position they play (Centers tend to get in foul trouble easier), we ignored this factor in our predictions.

- The last factor, overtime, is also completely unpredictable. Overtime or extra time is an additional period of play specified under the rules of a sport to bring a game to a decision and avoid declaring the match a tie or draw where the scores are the same.

So in order to predict the minutes an athlete would play we focused on blowout factor. To predict weather or not a game could go to "blowout" resulting in les minutes played for the core players, we used betting lines. The spread, or line, is a number assigned by the bookmakers which handicaps one team and favors another when two teams play each other and one is perceived as being more likely to win. The favorite "takes" points from the final score and the underdog "gives" points. For example if team A which is ranked 1 on the NBA standings table plays a home game versus team B which is ranked 30, the betting line could be something similar to +15 for team A. This means that if somebody bets on team A to win, team A needs to win by more than 15 points. Similarly, betting on team B to win would be successful if team B doesn't lose by more than 15 points.

We downloaded all betting lines from www.covers.com for 2015-2016 NBA regular season. Then we got the average minutes played of NBA players from last 3 games and adjusted them based on the betting lines. If the betting line for one game was equal or lower than +5 then minutes stayed the same. If the betting line was at 5.5 I multiplied player's minutes by 96%. For example if a player averaged 40 minutes per game in his last 3 games and the betting line for his next game was +5.5, his new projected minutes are 38.8. For every additional +1 in the betting line I lowered the multiplier by 1. It can be explained better with pseudo code:

```
If betting line < 5
        Projected player minutes for the next game = average minutes in the last 3 games
 Else If betting line > 5
        Projected player minutes for the next game = average minutes in the last 3 games
 Else if betting line = 5.5 or 6
        Projected player minutes for next game = average minutes in the last 3 games *96%
 Else if betting line = 6.6        or 6.5
        Projected player minutes for next game = average minutes in the last 3 games *95%
 Else if betting line = 7 or 7.5
        Projected player minutes for next game = average minutes in the last 3 games *94%
                                        ……
etc.
```

4.3 Developing a prediction model

After constructing the dataset we started creating the prediction model. The model would start predicting after the fifth game of the season. The reason behind that is the lack of relevant data at the first games of an NBA season. When an NBA season starts, teams might have an entirely different roster compared to last season, new coach, new playbooks, etc. Moreover the time period between the last NBA regular season game and the first game of the next season is around 6 months. In those 6 months players can develop new skills and improve or decline due to age. All of the above make it impossible to predict accurately enough a player's performance of the first games of the season based on last year's performance.

We used two different players, LeBron James and Aaron Gordon and we tried the following regression models on them in order to find the one that generates the best results:

- Linear Regression: Linear regression model performed decent on training and testing dataset. But when used on unseen data, it didn't produce optimal results.
- SVR with polynomial and RBF kernel: SVR's prediction was always the same number for training set and validation set. Both kernels generated a number and used it as prediction for any kind of data they were asked to predict.
- Random Forest Regression: Random forest regression was the method that produced the best results compared to the others, both in training and validation data. We ended up using the Random Forest method for our model.

- Bayesian Regression: Bayesian ridge regression was the method which produced the second best results. The results were acceptable but not as good as those produced by Random Forest Regression.

Starting our code, the first step was to create the dataset. We created a loop that reads every CSV file from a directory and read them 1 by 1.

```
path = "C:\\CSVlist\\"

filenames = glob.glob(path + "/*.csv")

for infile in glob.glob( os.path.join(path, '*csv') ):
```

Then using pandas, we read the CSV file and created a data frame with it. Our dataframe had ten columns: Date (date that the game was played), Rebounds, PTS (number of points that the athlete scored in that game), Assists, Steals, Blocks, DFS (the daily fantasy score), MP (the minutes the athlete played) and Salary (the fantasy cost of the athlete).

Next we defined the features and the target. The features consisted of Rebounds, Steals, Assists, Blocks, MP and PTS columns and the target was the DFS. So we excluded the first six games of the season and started predicting from the seventh. The code for defining our features includes a for loop that makes sure the model doesn't "cheat" by reading the statistics of the player on the date it's predicting. This loop concatenates the statistical values (PTS, FG, FGM, etc) from 6 games before the game that the model predicts and defines them as features. The code for this is the following:

```
forecast_col = 'DFS'

df333['label'] = df[forecast_col]

data = np.array(df333.drop(['Date','Position',’DFS’],1))

X = np.empty( [len(data)-6,6*len(data[0])] )

    for i in range(0, len(data)-6):

        if i > len(data) - 1:

          break


        result = np.array(data[i])

        for j in range(1,6):

          result = np.concatenate((result,data[i+j]))

        X[i] = result

y=labels[6:]
```

We also excluded 1 single game which was going to be used for validation and also for our test runs. User can set a date which won't be used for training or testing and later predict the outcome of this date based on model's training:

dfval=df.loc[:30917].tail(7)

dfval= df[df.index != 3917]

Given the value 30917 for dfval, the game for 9th of March 2017, will be excluded from training and testing dataset.

After splitting our data, we performed cross validation. We used the following parameters in order to find out which would return the best results:

- N estimators: 15 to 350
- Max features: 3 to 50
- Max depth: 10 to 60
- Minimum sample split: 2 to 40
- Bootstrap: True or False

The parameters we ended up using after performing cross validations are the following:

- N estimators: 50
- Minimum sample split: 2
- Max features: 7
- Max depth: 22

We used those parameters for every player during our tests. Ideally we would run cross validation again before we predict a player's fantasy score. But cross validation required nearly 10 hours to test every parameter, even on a computer with latest generation CPU, for just one player. Moreover sklearn doesn't provide the option to use GPU in order to increase processing time.

After completing the training of our model we tested it on validation data. We create a new dataframe which contained the game from the date we set before as dfval and the 6 dates before that particular date. We set our features and target the same way as before and we predict our new unseen data.

Having our predictions ready we wrote the results in a CSV file along with player's name, position and salary. This CSV file will be used later for our optimizer in order to generate a lineup.

dffinal=dfnew[(dfnew['Date']== 30917)]

dffinal.reset_index(drop=True, inplace=True)

dffinal2=dffinal.drop(['Date',"PTS","RB",'Assist','Blk','Steals','DFS','MP','label'],1)

dffinal2['filename'] = [df123]

dffinal3=pd.concat([dffinal2,ss],ignore_index=True,axis=1)

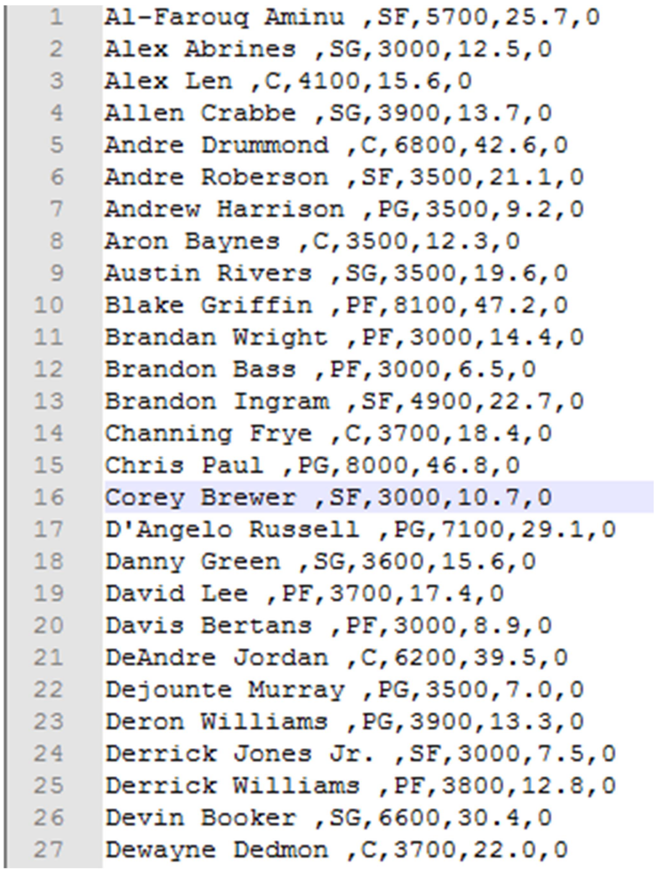
```
dffinal3.columns=['Position','Salary','Name','Pred']

dffinal3['Pred'] = dffinal3['Pred'].apply(lambda x: round(x,1))

dffinal3 = dffinal3[['Name','Position','Salary','Pred']]

 dffinal3.to_csv('dffinal',mode='a',index=False, header=None)
```

Image 7 shows the form of our final CSV file



```
Al-Farouq Aminu ,SF,5700,25.7,0
Alex Abrines ,SG,3000,12.5,0
Alex Len ,C,4100,15.6,0
Allen Crabbe ,SG,3900,13.7,0
Andre Drummond ,C,6800,42.6,0
Andre Roberson ,SF,3500,21.1,0
Andrew Harrison ,PG,3500,9.2,0
Aron Baynes ,C,3500,12.3,0
Austin Rivers ,SG,3500,19.6,0
Blake Griffin ,PF,8100,47.2,0
Brandan Wright ,PF,3000,14.4,0
Brandon Bass ,PF,3000,6.5,0
Brandon Ingram ,SF,4900,22.7,0
Channing Frye ,C,3700,18.4,0
Chris Paul ,PG,8000,46.8,0
Corey Brewer ,SF,3000,10.7,0
D'Angelo Russell ,PG,7100,29.1,0
Danny Green ,SG,3600,15.6,0
David Lee ,PF,3700,17.4,0
Davis Bertans ,PF,3000,8.9,0
DeAndre Jordan ,C,6200,39.5,0
Dejounte Murray ,PG,3500,7.0,0
Deron Williams ,PG,3900,13.3,0
Derrick Jones Jr. ,SF,3000,7.5,0
Derrick Williams ,PF,3800,12.8,0
Devin Booker ,SG,6600,30.4,0
Dewayne Dedmon ,C,3700,22.0,0
```

Image 7. Produced CSV file

5.3 Lineup Optimization

Predicting the fantasy score of the players through a machine learning model was the main focus of this thesis. But in order to test our predictions in actual daily fantasy contests, we had to choose 8 players which would constitute our lineup. For this purpose we developed an algorithm which based on our predictions, creates the best possible combination of 8 players for any given contest.

Any lineup that is submitted in a draftkings contest has to fulfill the following constrains:

- The lineup has to include at least 1 player who is eligible for PG (Point Guard) position.

- The lineup has to include at least 1 player who is eligible for SG (Shooting Guard) position.
- The lineup has to include at least 1 player who is eligible for SF (Small Forward) position.
- The lineup has to include at least 1 player who is eligible for PF (Power Forward) position.
- The lineup has to include at least 1 player who is eligible for C (Center) position.
- The lineup has to include at least 1 player who is eligible for F (Forward) position.
- The lineup has to include at least 1 player who is eligible for G (Guard) position.
- The total amount of salary for each lineup has to be below 80.000.000 $

So our algorithm had to maximize the value of our lineups (fantasy points) while taking into account these constrains.

The input data for our algorithm are all the NBA players available for selection in the current date. We place the athletes into a set called A. Then we divide the set to multiple sets based on the athlete's position. All NBA players who are eligible for Point Guard position are members of subset P, all shooting guard are members of subset S, all small forwards are members of subset M, all power forwards are members of subset P and all centers are members of the subset C. At the same time all shootings guards and point guards are members of the subset G and all small forwards and power forwards are members of the subset F. We can describe the input data better like this:

A = {a1,….,an}, which is the set of all NBA players available

$N \subset A$

$S \subset A$

$M \subset A$

$P \subset A$

$C \subset A$

$G \subset P,S$

$F \subset M, P$

Si = the salary of the athlete $\forall$ i $\in$ A

μi= the predicted fantasy points $\forall$ i $\in$ A

Every NBA player available has also a binary decision variable which represents whether or not the player is selected for the lineup. The variable xi is equal to 1 if the player is selected and equal to 0 if the player is not selected.

$$Xi = \begin{cases} 1 \; if \; the \; player \; is \; selected \; for \; the \; fantasy \; team \\ \qquad 0 \; otherwise \end{cases} \forall i \in A$$

Mathematically the above problem can be described like this:

Maximize $\sum_{i \in A} \mu_i x_i$

Subject *to:*

$$\sum_{i \in N} x_i = 1$$

$$\sum_{i \in S} x_i = 1$$

$$\sum_{i \in M} x_i = 1$$

$$\sum_{i \in P} x_i = 1$$

$$\sum_{i \in C} x_i = 1$$

$$\sum_{i \in G} x_i = 1$$

$$\sum_{i \in F} x_i = 1$$

$$\sum_{i \in N \cup S \cup M \cup P \cup C \cup G \cup F} x_i = 7$$

$$\sum_{i \in R} x_i S_i \leq 80.000$$

$X_i \in \{0,1\}^n$ for all $i \in A$

5

Evaluation

To evaluate the results of the prediction model we took into consideration the following:

The goal of this model isn't to achieve 100% accuracy on predictions. That would be nearly impossible. The goal is to create predictions which when used, can generate lineups that will return a profit in the long term. There are 3 important things that make a prediction model for DFS sports successful:

Spotting undervalued players: As we mentioned in chapter 1, in DFS users have a certain amount of resources to spend in order to create a lineup. Every athlete has a different value which depends on his average fantasy performance. Finding players who can score more than their salary value, it's crucial for a lineup's success. For example if player A costs 7,000$ and scores 35 fantasy points and player B costs 3,500$ and scores 30 fantasy points, player B is considered a value pick. Spotting those low cost players who can score more fantasy points than they are expected to its crucial for a line up's success.

Ability to spot extraordinary performance: It is really important for a lineup to include undervalued players who will score higher than expected. But at the same time including an expensive player who will score an extremely high fantasy score can be eough to guarantee a win.

We also mentioned before that our focus will be on cash games and not on GPP games.

5.1 Assessment parameters

To evaluate the model, we tested it on random dates of 2016-2017 NBA season and compared the predictions with the actual results. With a bit of research in online fantasy communities we can see that a score above 280 will almost always ensure a win (https://rotogrinders.com/threads/average-fanduel-nba-scores-required-to-cash-in-h2h-or-183106 , https://www.reddit.com/r/dfsports/comments/2o8i68/what_score_should_you_typically_aim_for_in). Of course that can vary from day to day. In certain contests, the score to actually win in cash games could be higher. But still a score on the range we mentioned is a good indicator of whether a lineup will be successful.

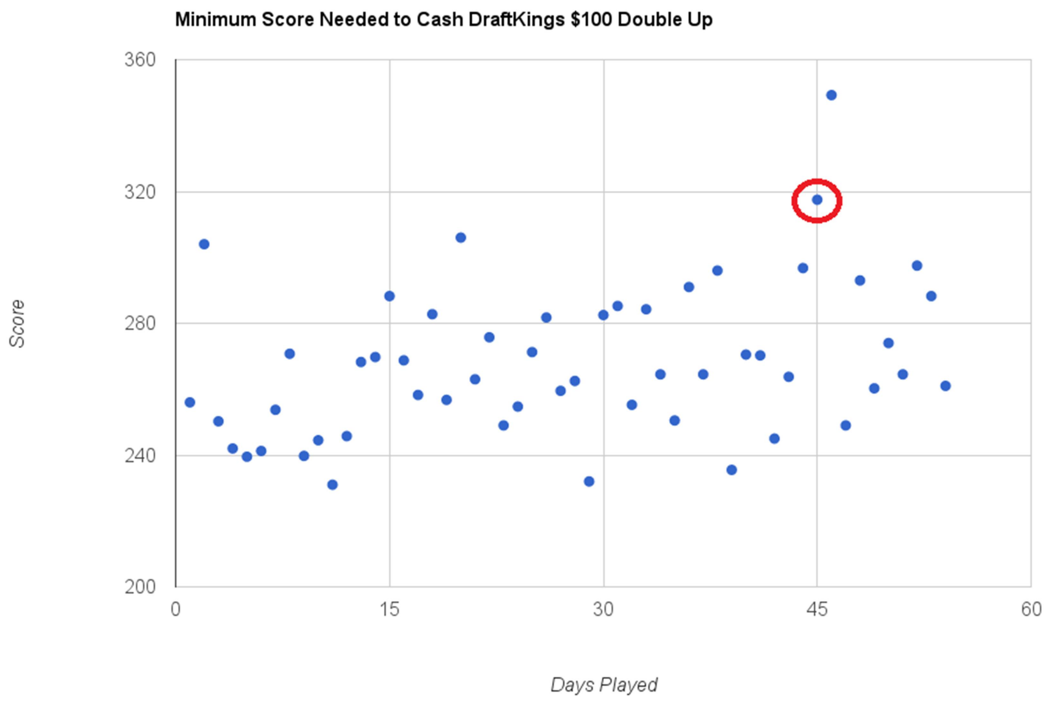


Image 8. Fantasy score required to win a Draft kings cash game in the span of 60 days

Image 6 illustrates the fantasy scored required to win a Draft kings cash game in the span of 60 days. Out of 60 days there were only 4 instances that a lineup of 300 fantasy score wouldn't translate into profit and a total of 16 instances where a 280 fantasy score wasn't enough. This means that if our model generated lineups that can hit our target score (280) we would win 44 out of 60 contests. Calculating the actual profits, $100 double up contests have an entry fee of 100$ and a prize of 185$ for the top 50% participants. In order to enter 60 different contests in the span of 60 days we would need 6,000 $. By winning 44 of them we would have 8,140$ by the end of the 60th day, which is 2,140$ profit on our original investment.

We also compared our results with projections and lineups from www.fantasy-cruncher.com. Fantasy cruncher is a website which provides projections for daily fantasy sports and also a line up optimization software. They charge 49.95 USD per month. Even though predictions are not fantasy cruncher's main feature, the comparison with our model can help us draw useful conclusions.

5.2 Tests

The first date we tested was 2nd of March 2017. There were 3 different games available, 6 teams and a total of 68 players. We run our prediction model on those players and gathered the results in a CSV file with the following format:

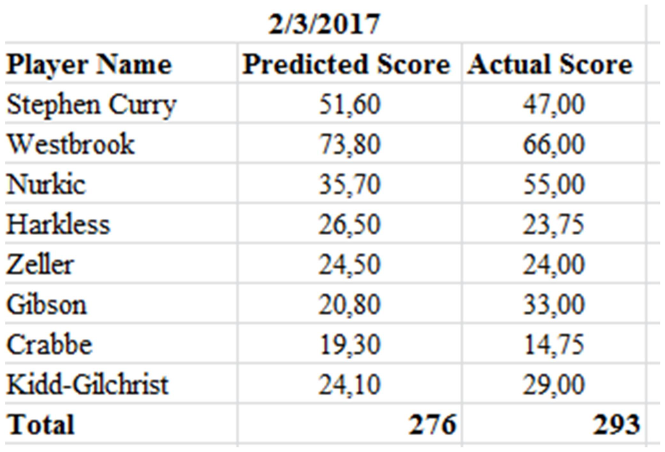
Al-Farouq Aminu ,SF,5700,25.7,0

Alex Abrines ,SG,3000,12.5,0

Alex Len ,C,4100,15.6,0

Each line of the CSV contained the name of the player, his position, his DraftKings salary and his projected fantasy points. The zero after the last comma indicates whether a player is going to be included in the lineup. If the value after that last comma is -1 the player will be excluded from the projected lineup and if its 1 the optimizer will be forced to include that player.

Next we run the CSV through our lineup optimizer, getting the following results:

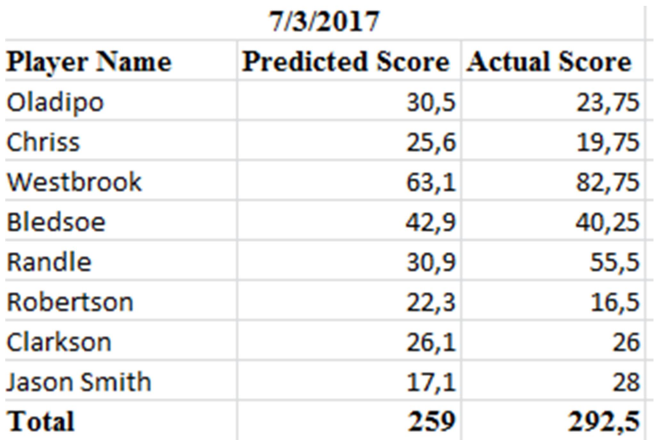


2/3/2017		
Player Name	**Predicted Score**	**Actual Score**
Stephen Curry	51,60	47,00
Westbrook	73,80	66,00
Nurkic	35,70	55,00
Harkless	26,50	23,75
Zeller	24,50	24,00
Gibson	20,80	33,00
Crabbe	19,30	14,75
Kidd-Gilchrist	24,10	29,00
Total	**276**	**293**

Image 9. Generated lineup for 2/3/2017

Our model predicted a fantasy score of 276 and our line up actually scored 293 fantasy points. With our first test we managed to reach our target of 280 fantasy points. Our model managed to predict the extraordinary fantasy performance of Russell Westbrook despite predicting that he will score 7.8 more fantasy points. On Cody Zeller our model had almost 100% accuracy, miscalculating his performance by only 0.5 fantasy points. On Nurkic's case, the 20 fantasy points difference between his actual score and our predicted fantasy score, didn't affect our final results. Due to his low salary (5600), a fantasy score of 35.7 points was enough for our optimizer to include him in the generated lineup.

The next date we used for testing was 7th of March 2017. There were again 3 games available, 6 teams and a total of 68 players.



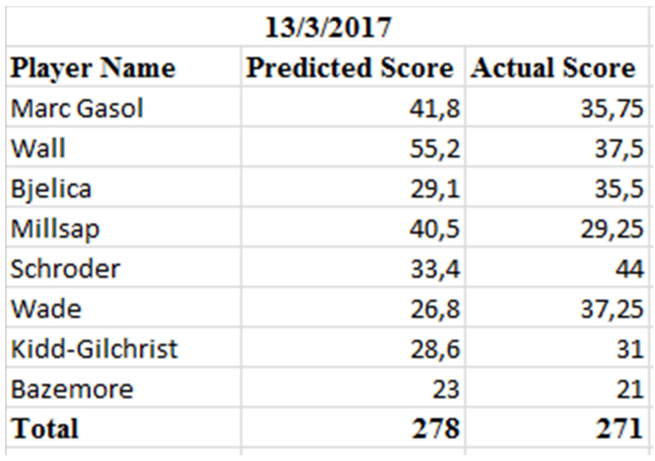
7/3/2017		
Player Name	**Predicted Score**	**Actual Score**
Oladipo	30,5	23,75
Chriss	25,6	19,75
Westbrook	63,1	82,75
Bledsoe	42,9	40,25
Randle	30,9	55,5
Robertson	22,3	16,5
Clarkson	26,1	26
Jason Smith	17,1	28
Total	**259**	**292,5**

Image 10. Generated lineup for 7/3/2017

Running our model and our optimizer we got the above lineup with a predicted score for 259 fantasy points and an actual score for 292.5 fantasy points. Same as our last test, we managed to reach our goal of 280 fantasy points despite the prediction accuracy being low.

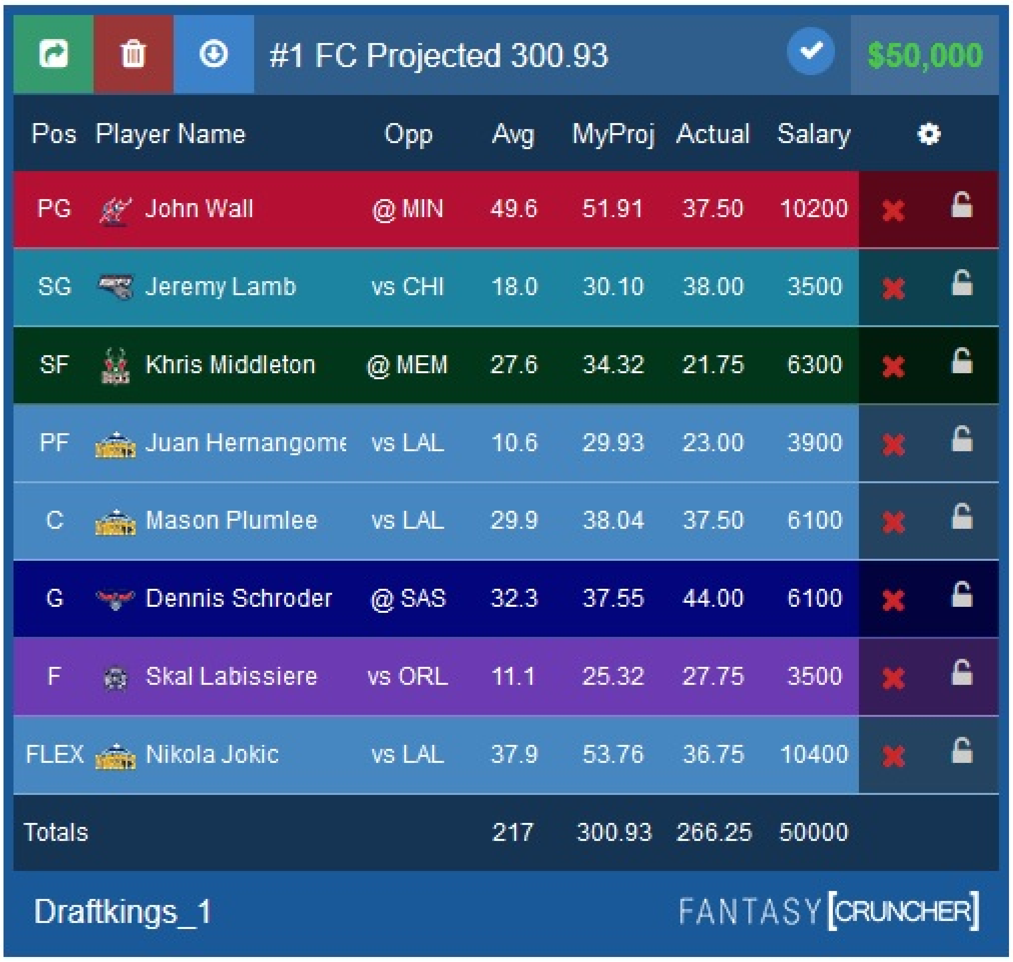
Our model had the most success with Clarkson's prediction, missing his actual score by only 0.1 fantasy points. It appears that our model undervalued Randle's and Russell Westbrook's fantasy performance but at the same time the predictions were high enough to indicate that they are worth including in a lineup.

Our next testing date was 13th of March 2017. This time there were 8 NBA games available, with 16 teams and 192 players.



13/3/2017		
Player Name	**Predicted Score**	**Actual Score**
Marc Gasol	41,8	35,75
Wall	55,2	37,5
Bjelica	29,1	35,5
Millsap	40,5	29,25
Schroder	33,4	44
Wade	26,8	37,25
Kidd-Gilchrist	28,6	31
Bazemore	23	21
Total	**278**	**271**

Image 11. Generated lineup for 13/3/2017


#1 FC Projected 300.93 $50,000

Pos	Player Name	Opp	Avg	MyProj	Actual	Salary
PG	John Wall	@ MIN	49.6	51.91	37.50	10200
SG	Jeremy Lamb	vs CHI	18.0	30.10	38.00	3500
SF	Khris Middleton	@ MEM	27.6	34.32	21.75	6300
PF	Juan Hernangome	vs LAL	10.6	29.93	23.00	3900
C	Mason Plumlee	vs LAL	29.9	38.04	37.50	6100
G	Dennis Schroder	@ SAS	32.3	37.55	44.00	6100
F	Skal Labissiere	vs ORL	11.1	25.32	27.75	3500
FLEX	Nikola Jokic	vs LAL	37.9	53.76	36.75	10400
Totals			217	300.93	266.25	50000

Draftkings_1 FANTASY [CRUNCHER]

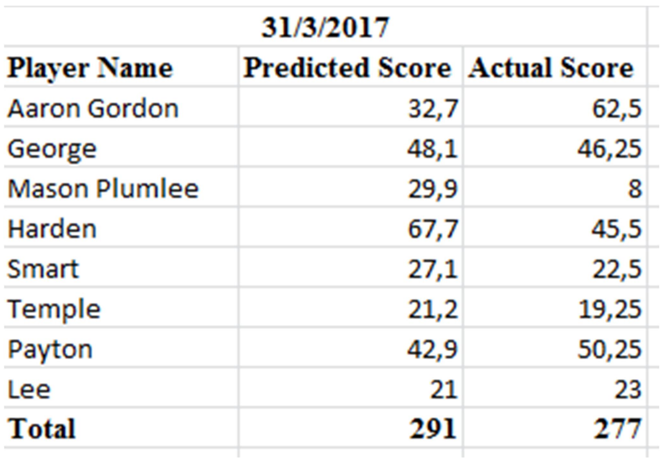
Image 12. Fantasy Cruncher's lineup for 13/3/2017

This time even though the total fantasy score of our lineup was only 7 fantasy points lower than the fantasy score our model predicted, we didn't manage to reach our goal of 280 fantasy points. Wall's difference between his predicted and actual score is the most noticeable. Looking at Wall's last 6 games, he averaged 54.13 fantasy points per game which can explain our model's prediction. He also averaged 48.8 fantasy points in his next 8 games.

Paul Millsap scored only 29.25 fantasy points, 11 less compared to our model's prediction, despite playing for 40 minutes. At the same time he scored over 37 fantasy points in his last 8 games, playing below 40 minutes in all of them.

Image 12 shows fantasy cruncher's lineup for that particular date. In comparison with our lineup, fantasy cruncher's lineup scored 5 points lower but the difference in their prediction score and actual score was a lot higher.

For our next date, 31th of March 2017, there were 8 NBA games, with 16 teams and 192 players available.

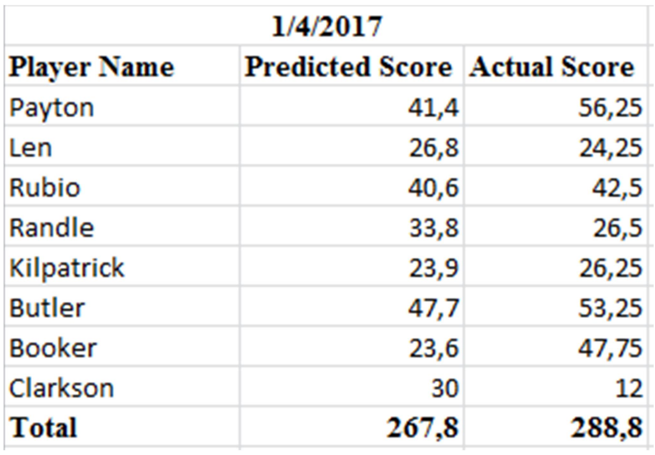


31/3/2017		
Player Name	**Predicted Score**	**Actual Score**
Aaron Gordon	32,7	62,5
George	48,1	46,25
Mason Plumlee	29,9	8
Harden	67,7	45,5
Smart	27,1	22,5
Temple	21,2	19,25
Payton	42,9	50,25
Lee	21	23
Total	**291**	**277**

Image 13. Generated lineup for 31/3/2017

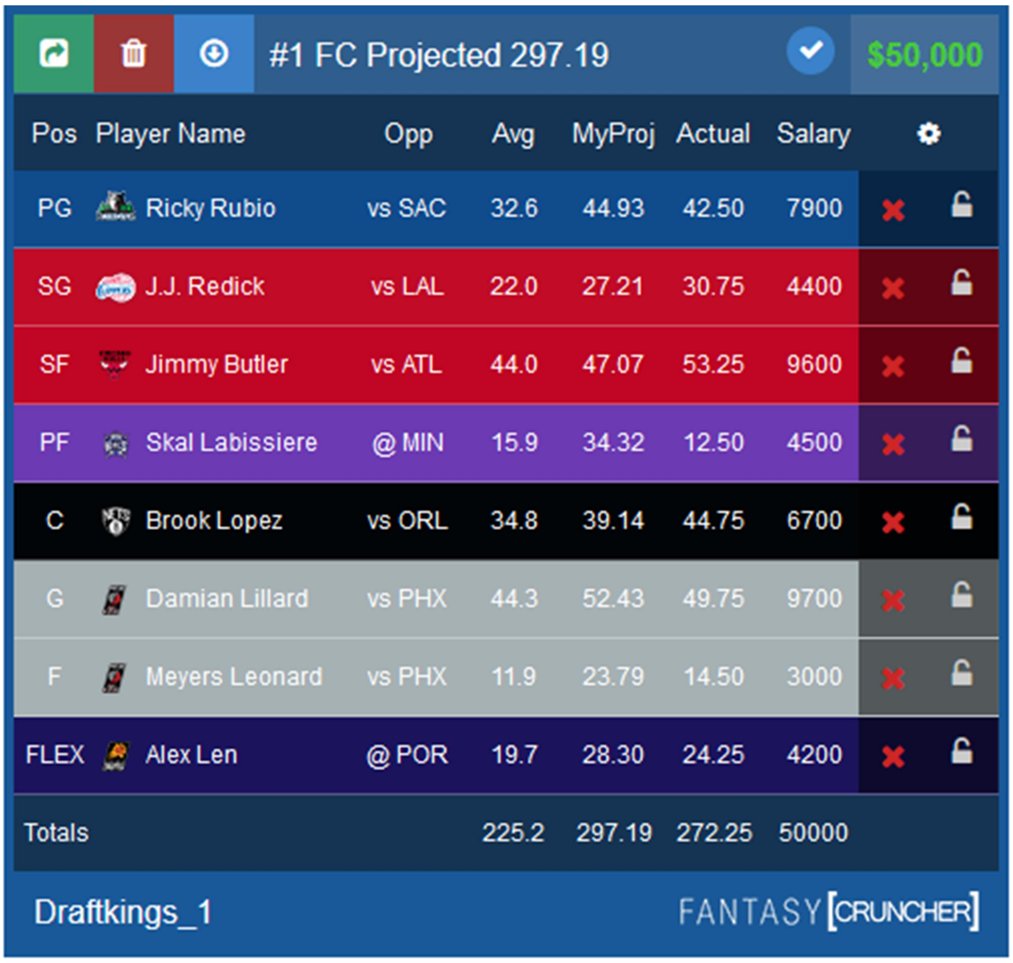
Our model's performance wasn't good enough to reach our target fantasy score. There was a significant difference between the predicted score and actual score (14 fantasy points). Moreover the inaccurate predictions of Mason Plumlee's and James Harden's performance resulted in inefficient use of our available salary. Even Aaron Gordon's 62.5 fantasy points for only 5,400$ wasn't enough for our lineup to reach our target score.

Our 5th test date was April 1 2017, with 5 NBA games, 10 teams and 120 players available.



1/4/2017		
Player Name	**Predicted Score**	**Actual Score**
Payton	41,4	56,25
Len	26,8	24,25
Rubio	40,6	42,5
Randle	33,8	26,5
Kilpatrick	23,9	26,25
Butler	47,7	53,25
Booker	23,6	47,75
Clarkson	30	12
Total	**267,8**	**288,8**

Image 14. Generated lineup for 1/4/2017



#1 FC Projected 297.19 $50,000

Pos	Player Name	Opp	Avg	MyProj	Actual	Salary
PG	Ricky Rubio	vs SAC	32.6	44.93	42.50	7900
SG	J.J. Redick	vs LAL	22.0	27.21	30.75	4400
SF	Jimmy Butler	vs ATL	44.0	47.07	53.25	9600
PF	Skal Labissiere	@ MIN	15.9	34.32	12.50	4500
C	Brook Lopez	vs ORL	34.8	39.14	44.75	6700
G	Damian Lillard	vs PHX	44.3	52.43	49.75	9700
F	Meyers Leonard	vs PHX	11.9	23.79	14.50	3000
FLEX	Alex Len	@ POR	19.7	28.30	24.25	4200
Totals			225.2	297.19	272.25	50000

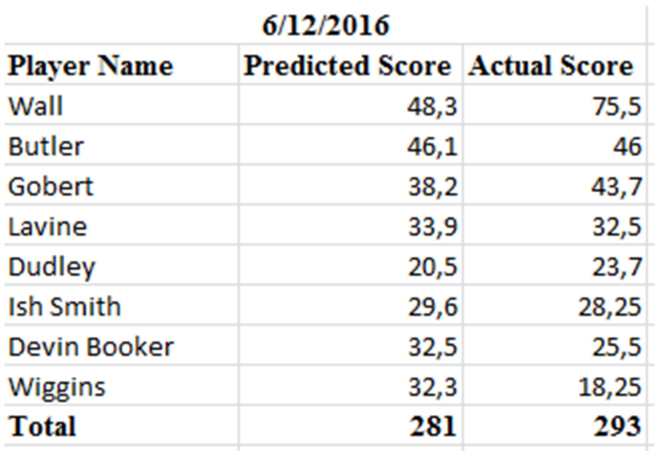
Draftkings_1

Image 15. Fantasy Cruncher's lineup for 1/4/2017

As we can see in the image above, our lineup reached our target of 280 fantasy points. Once again the total actual score was significantly higher than the predicted score, mainly because of the inaccurate prediction on Jordan Clarkson's and Elfrid Payton's performance. For 5 out of 8 players the difference between the actual score and the predicted score was below 5 fantasy points, which can be considered quite successful.

Comparing our prediction to fantasy cruncher's predictions, our lineup again had a better actual score while fantasy cruncher's lineup didn't even reach the target of 280 fantasy points.

Until now we tested our model in the late stages of NBA 2016-2017 regular season. This means that our model had more than 50 games to use for training and testing. For the next two tests we used dates from December of 2016, so our model had only around 20 games available for training and testing.

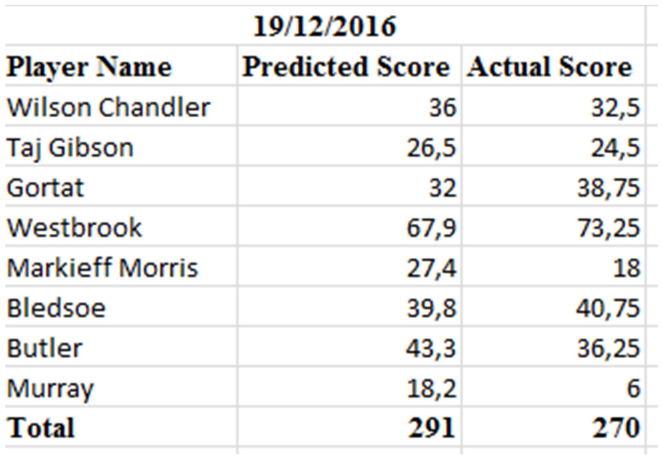


6/12/2016		
Player Name	**Predicted Score**	**Actual Score**
Wall	48,3	75,5
Butler	46,1	46
Gobert	38,2	43,7
Lavine	33,9	32,5
Dudley	20,5	23,7
Ish Smith	29,6	28,25
Devin Booker	32,5	25,5
Wiggins	32,3	18,25
Total	**281**	**293**

Image 17. Generated lineup for 6/12/2016

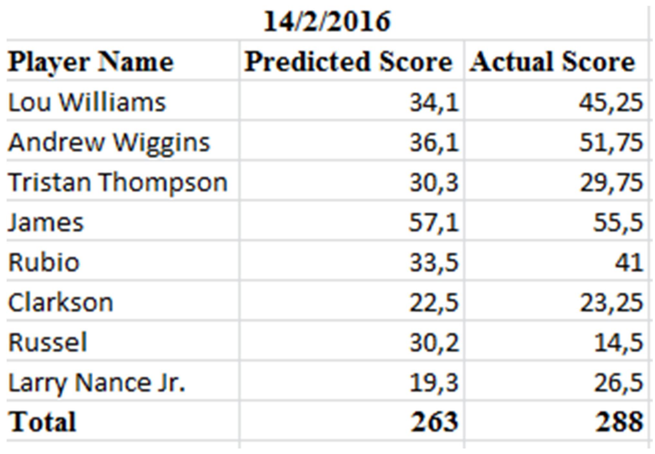
In this test, for 6/12/2016, we reached our target of 280 fantasy points and the difference between the total actual score and the predicted score of our lineup was 12 fantasy points. The most noticeable results are John Wall's and Andrew Wiggin's predictions. In the first case John Wall scored 75.5 fantasy points with a prediction of 48.3 fantasy points. Once again, the lower than actual score prediction on John Wall's fantasy points, wasn't enough to hurt our results. On the other hand the salary spent on Andrew Wiggins could be used more efficient on some other player considering that his salary cost was 6,000$. For the rest of the players the results could be considered adequately based on the fact that the difference between the predicted and the actual fantasy score was lower than 5 fantasy points in 5 of the total 8 players.

In the next 2 tests our lineup reached our target score in one case with 287.5 fantasy points and only scored 270 fantasy points in the other. The results can be seen in images 18 and 19.



19/12/2016		
Player Name	**Predicted Score**	**Actual Score**
Wilson Chandler	36	32,5
Taj Gibson	26,5	24,5
Gortat	32	38,75
Westbrook	67,9	73,25
Markieff Morris	27,4	18
Bledsoe	39,8	40,75
Butler	43,3	36,25
Murray	18,2	6
Total	**291**	**270**

Image 18. Generated lineup for 19/12/2016



14/2/2016		
Player Name	**Predicted Score**	**Actual Score**
Lou Williams	34,1	45,25
Andrew Wiggins	36,1	51,75
Tristan Thompson	30,3	29,75
James	57,1	55,5
Rubio	33,5	41
Clarkson	22,5	23,25
Russel	30,2	14,5
Larry Nance Jr.	19,3	26,5
Total	**263**	**288**

Image 19. Generated lineup for 14/2/2016

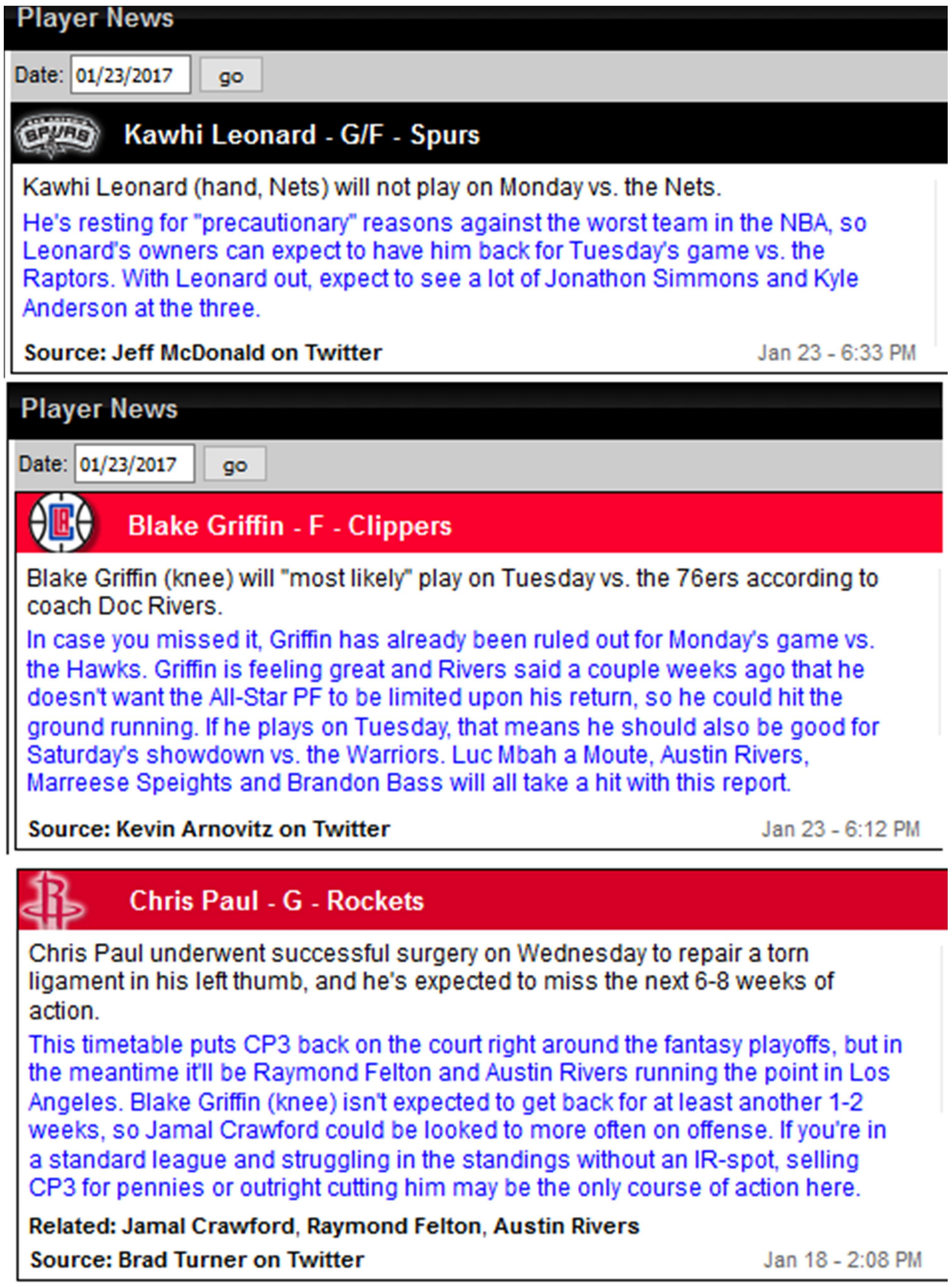
5.3 Testing with manual tweaks

In daily fantasy sports players have the option to modify their submitted lineups until 5 minutes before the first NBA game starts. This allows for last minutes tweaks based on news about player's availability, injury, etc.

When a starter NBA player is unavailable for a game, there is a significant amount of minutes that will be allocated to the rest of the team. This means that there will always be a player who will see a significant increase in his minutes and in most cases to his fantasy performance. When we generate a lineup using our model we can exclude an injured or unavailable player. But at the same time our model doesn't change the predicted score on the player that will take the injured player's place and minutes. To compensate that we can study some statistics like a team's statistics when X player is missing, to draw useful conclusions about possible increase or decrease on the fantasy performance of the rest of the team. Based on those statistics, we can extract information about specific players who will potentially benefit from X player's absence and "force" the optimizer to include that player in the generated lineup.

For the next tests we generated 2 lineups for each date. In the first lineup we let the optimizer choose the optimal lineup based on our model's predictions similar to the previous tests. For the second lineup we checked injured or rested players whose absence was reported before the contests started. Then we checked the statistics of the rest of the team while that player is not playing. If there was a player who benefited statistically from the injured player's absence, we forced our optimizer to include him in the lineup.

We tested our model at 23rd of January 2017. For that date Kawhi Leonard, Chris Paul and Blake Griffin weren't available due to injury or rest as reported by rotoworld.com.


Player News

Date: 01/23/2017 go

Kawhi Leonard - G/F - Spurs

Kawhi Leonard (hand, Nets) will not play on Monday vs. the Nets.

He's resting for "precautionary" reasons against the worst team in the NBA, so Leonard's owners can expect to have him back for Tuesday's game vs. the Raptors. With Leonard out, expect to see a lot of Jonathon Simmons and Kyle Anderson at the three.

Source: Jeff McDonald on Twitter Jan 23 - 6:33 PM

Player News

Date: 01/23/2017 go

Blake Griffin - F - Clippers

Blake Griffin (knee) will "most likely" play on Tuesday vs. the 76ers according to coach Doc Rivers.

In case you missed it, Griffin has already been ruled out for Monday's game vs. the Hawks. Griffin is feeling great and Rivers said a couple weeks ago that he doesn't want the All-Star PF to be limited upon his return, so he could hit the ground running. If he plays on Tuesday, that means he should also be good for Saturday's showdown vs. the Warriors. Luc Mbah a Moute, Austin Rivers, Marreese Speights and Brandon Bass will all take a hit with this report.

Source: Kevin Arnovitz on Twitter Jan 23 - 6:12 PM

Chris Paul - G - Rockets

Chris Paul underwent successful surgery on Wednesday to repair a torn ligament in his left thumb, and he's expected to miss the next 6-8 weeks of action.

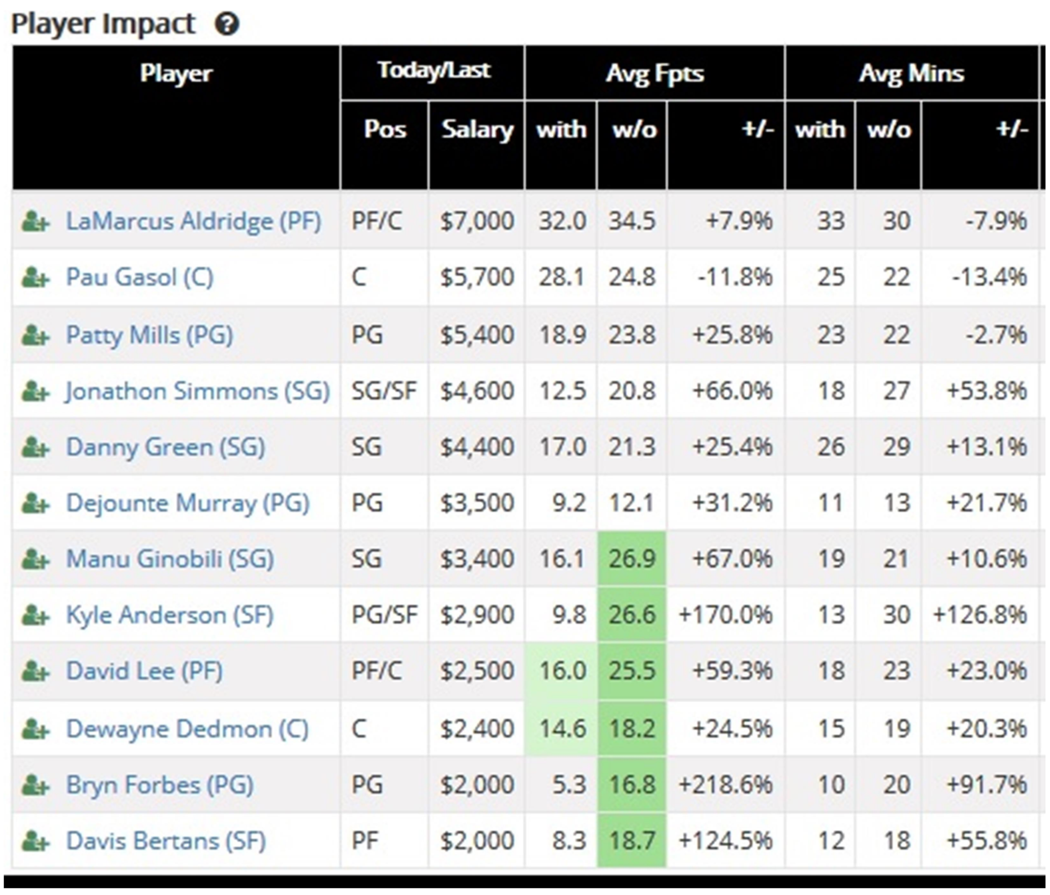
This timetable puts CP3 back on the court right around the fantasy playoffs, but in the meantime it'll be Raymond Felton and Austin Rivers running the point in Los Angeles. Blake Griffin (knee) isn't expected to get back for at least another 1-2 weeks, so Jamal Crawford could be looked to more often on offense. If you're in a standard league and struggling in the standings without an IR-spot, selling CP3 for pennies or outright cutting him may be the only course of action here.

Related: Jamal Crawford, Raymond Felton, Austin Rivers

Source: Brad Turner on Twitter Jan 18 - 2:08 PM

Image 20. www.rotoworld.com Injury Reports for 23 January 2017

In the image 21 we can see San Antonio Spur's statistics when Kawhi Leonard isn't playing.



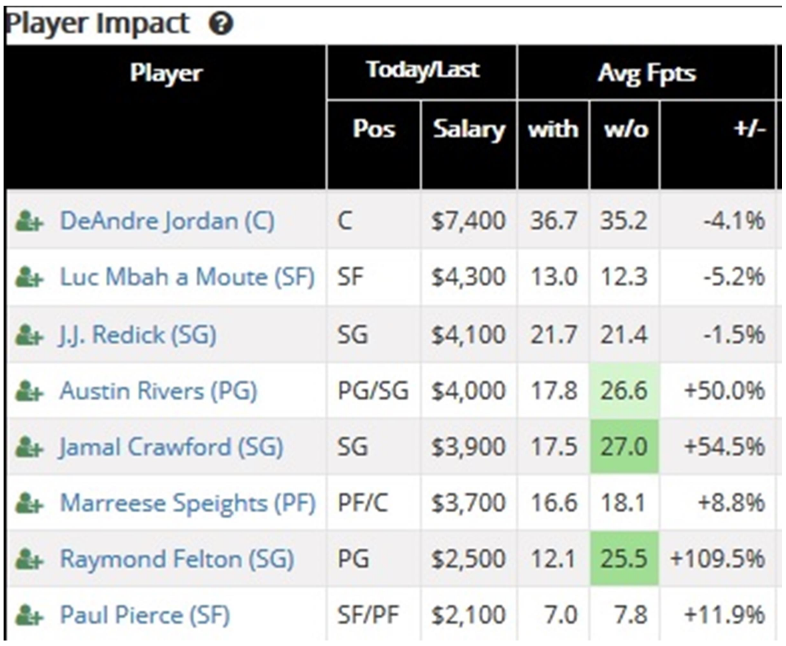
Player Impact

Player	Today/Last		Avg Fpts			Avg Mins		
	Pos	Salary	with	w/o	+/-	with	w/o	+/-
LaMarcus Aldridge (PF)	PF/C	$7,000	32.0	34.5	+7.9%	33	30	-7.9%
Pau Gasol (C)	C	$5,700	28.1	24.8	-11.8%	25	22	-13.4%
Patty Mills (PG)	PG	$5,400	18.9	23.8	+25.8%	23	22	-2.7%
Jonathon Simmons (SG)	SG/SF	$4,600	12.5	20.8	+66.0%	18	27	+53.8%
Danny Green (SG)	SG	$4,400	17.0	21.3	+25.4%	26	29	+13.1%
Dejounte Murray (PG)	PG	$3,500	9.2	12.1	+31.2%	11	13	+21.7%
Manu Ginobili (SG)	SG	$3,400	16.1	26.9	+67.0%	19	21	+10.6%
Kyle Anderson (SF)	PG/SF	$2,900	9.8	26.6	+170.0%	13	30	+126.8%
David Lee (PF)	PF/C	$2,500	16.0	25.5	+59.3%	18	23	+23.0%
Dewayne Dedmon (C)	C	$2,400	14.6	18.2	+24.5%	15	19	+20.3%
Bryn Forbes (PG)	PG	$2,000	5.3	16.8	+218.6%	10	20	+91.7%
Davis Bertans (SF)	PF	$2,000	8.3	18.7	+124.5%	12	18	+55.8%

Image21.San Antonio Spurs statistics when Kawhi Leonard isn't playing

We notice that Kyle Anderson has a significant increase in his fantasy points, plays the same position as Kawhi Leonard and at the same time his salary is quite low. That can potentially qualify him as an undervalued player by DraftKings. Kyle Anderson 2.900$ salary reflects his performance when Kawhi Leonard is available for the Sun Antonio Spurts.

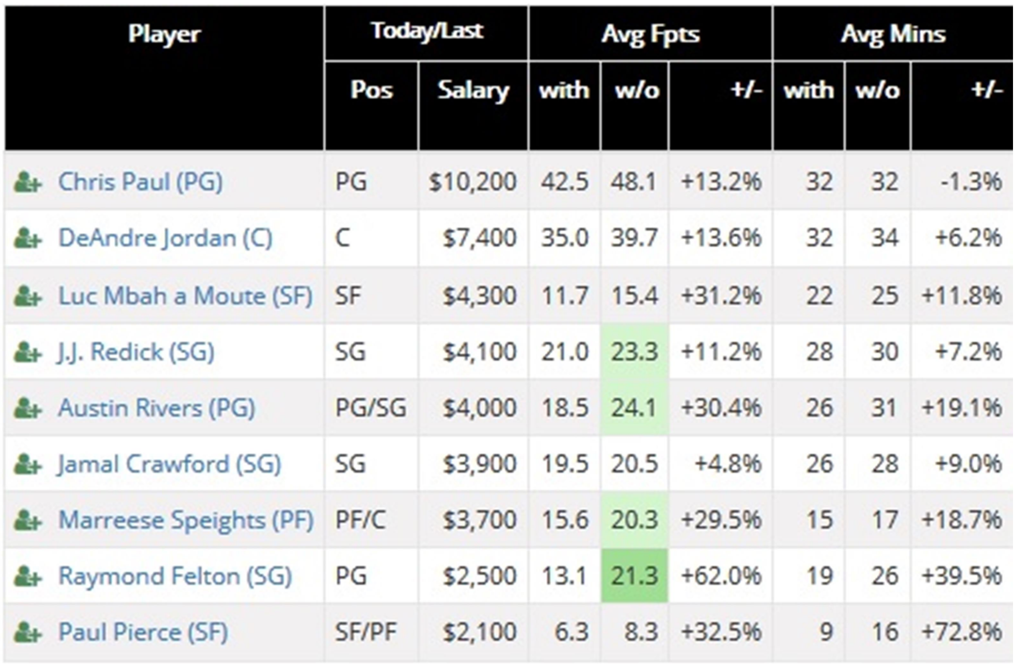
In the image 21 and we can see Los Angeles Clippers' statistics when Blake Griffin and Chris Paul are not not playing.



Player Impact

Player	Today/Last		Avg Fpts		
	Pos	Salary	with	w/o	+/-
DeAndre Jordan (C)	C	$7,400	36.7	35.2	-4.1%
Luc Mbah a Moute (SF)	SF	$4,300	13.0	12.3	-5.2%
J.J. Redick (SG)	SG	$4,100	21.7	21.4	-1.5%
Austin Rivers (PG)	PG/SG	$4,000	17.8	26.6	+50.0%
Jamal Crawford (SG)	SG	$3,900	17.5	27.0	+54.5%
Marreese Speights (PF)	PF/C	$3,700	16.6	18.1	+8.8%
Raymond Felton (SG)	PG	$2,500	12.1	25.5	+109.5%
Paul Pierce (SF)	SF/PF	$2,100	7.0	7.8	+11.9%

Image 22. Los Angeles Clippers statistics when Chris Paul isn't playing

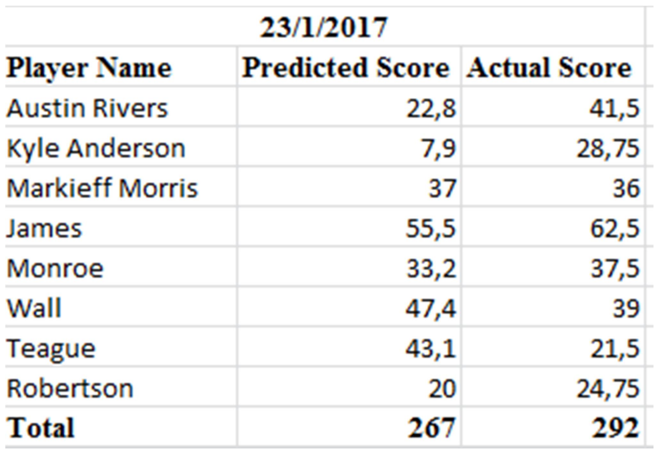


Player	Today/Last		Avg Fpts			Avg Mins		
	Pos	Salary	with	w/o	+/-	with	w/o	+/-
Chris Paul (PG)	PG	$10,200	42.5	48.1	+13.2%	32	32	-1.3%
DeAndre Jordan (C)	C	$7,400	35.0	39.7	+13.6%	32	34	+6.2%
Luc Mbah a Moute (SF)	SF	$4,300	11.7	15.4	+31.2%	22	25	+11.8%
J.J. Redick (SG)	SG	$4,100	21.0	23.3	+11.2%	28	30	+7.2%
Austin Rivers (PG)	PG/SG	$4,000	18.5	24.1	+30.4%	26	31	+19.1%
Jamal Crawford (SG)	SG	$3,900	19.5	20.5	+4.8%	26	28	+9.0%
Marreese Speights (PF)	PF/C	$3,700	15.6	20.3	+29.5%	15	17	+18.7%
Raymond Felton (SG)	PG	$2,500	13.1	21.3	+62.0%	19	26	+39.5%
Paul Pierce (SF)	SF/PF	$2,100	6.3	8.3	+32.5%	9	16	+72.8%

Image 23. Los Angeles Clippers statistics when Blake Griffin isn't playing

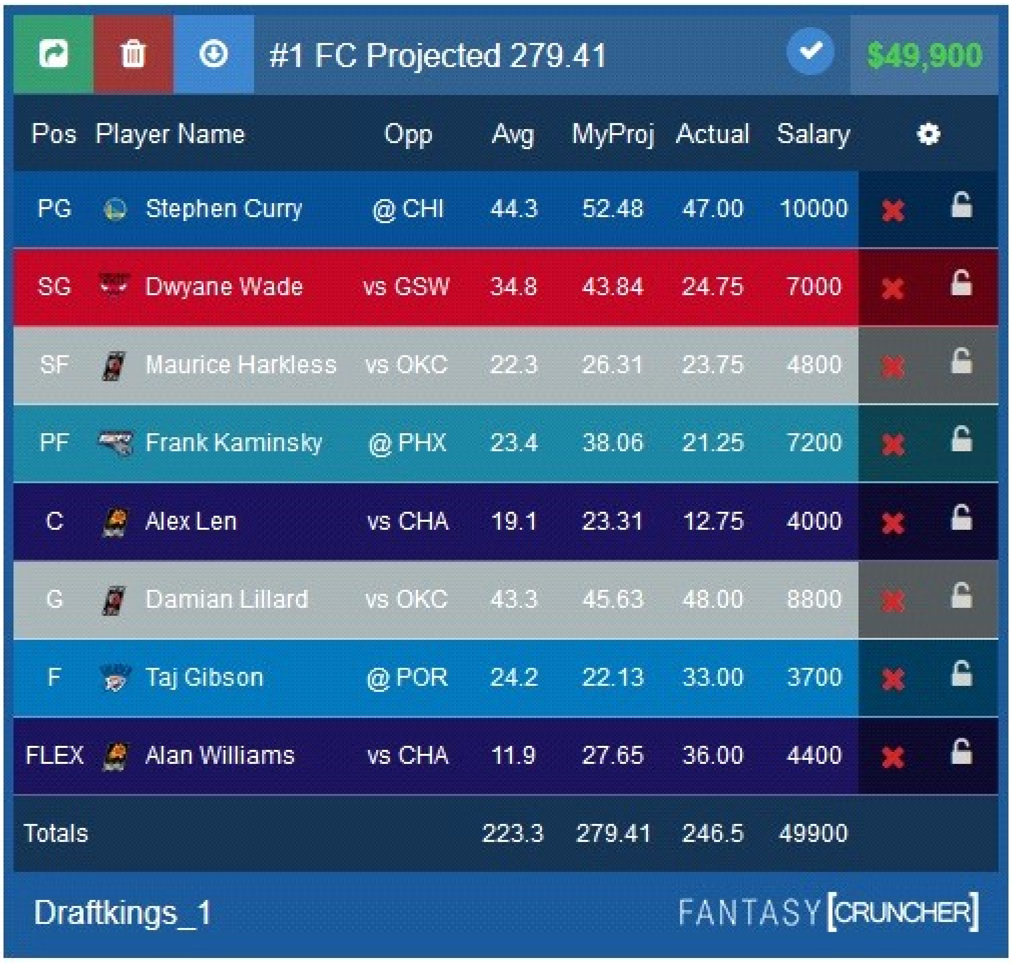
We notice that Austin Rivers and Jamal Crawford have the highest increase in fantasy points, play the same position as Chris Paul and at the same time their salary is quite low In relation to their average fantasy points.

While testing our model for 23 January 2017 we used the above information and we forced our optimizer to include a particular player.



23/1/2017		
Player Name	**Predicted Score**	**Actual Score**
Austin Rivers	22,8	41,5
Kyle Anderson	7,9	28,75
Markieff Morris	37	36
James	55,5	62,5
Monroe	33,2	37,5
Wall	47,4	39
Teague	43,1	21,5
Robertson	20	24,75
Total	**267**	**292**

Image 24. Generated lineup for 23/1/2017



#1 FC Projected 279.41 $49,900

Pos	Player Name	Opp	Avg	MyProj	Actual	Salary
PG	Stephen Curry	@ CHI	44.3	52.48	47.00	10000
SG	Dwyane Wade	vs GSW	34.8	43.84	24.75	7000
SF	Maurice Harkless	vs OKC	22.3	26.31	23.75	4800
PF	Frank Kaminsky	@ PHX	23.4	38.06	21.25	7200
C	Alex Len	vs CHA	19.1	23.31	12.75	4000
G	Damian Lillard	vs OKC	43.3	45.63	48.00	8800
F	Taj Gibson	@ POR	24.2	22.13	33.00	3700
FLEX	Alan Williams	vs CHA	11.9	27.65	36.00	4400
Totals			223.3	279.41	246.5	49900

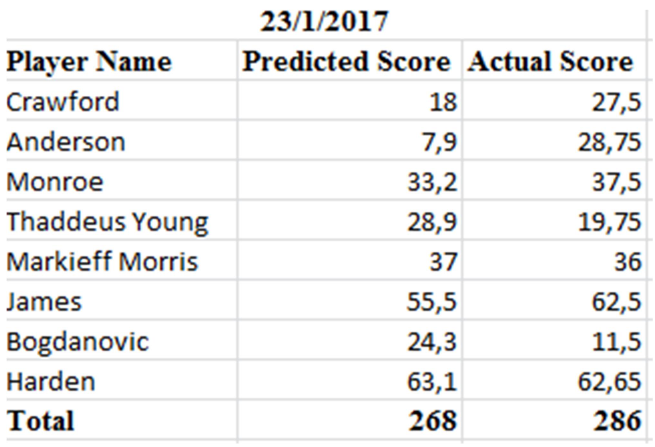
Draftkings_1 FANTASY [CRUNCHER]

Image 25. Fantasy Cruncher's lineup for 23/1/2017

For our first lineup we included Austin Rivers and Kyle Anderson. As we can see Kyle Anderson scored 28.75 fantasy points, which is 20 more than what our model predicted but it's almost exactly the fantasy score that he averages when Kawhi Leonard isn't playing. Moreover his salary cost was only 3,000$. Austin River's prediction was already a bit high compared to his usual fantasy numbers, due to Chris Paul's 2 weeks absence. But for this game Clippers would play without Blake Griffin. This resulted in Austin Rivers scoring 41.5 fantasy points.

Beside our tweaks our model performed great overall, with Jeff Teague's prediction being the only one with low accuracy. This lineup hit our target of 280 fantasy points and would most likely enter the prize pool in a cash game contest on DraftKings. Fantasy cruncher's optimizer produced a lineup that scored quite low fantasy points

We generated one more lineup for the same date but this time we included Jamal Crawford instead of Austing Rivers.



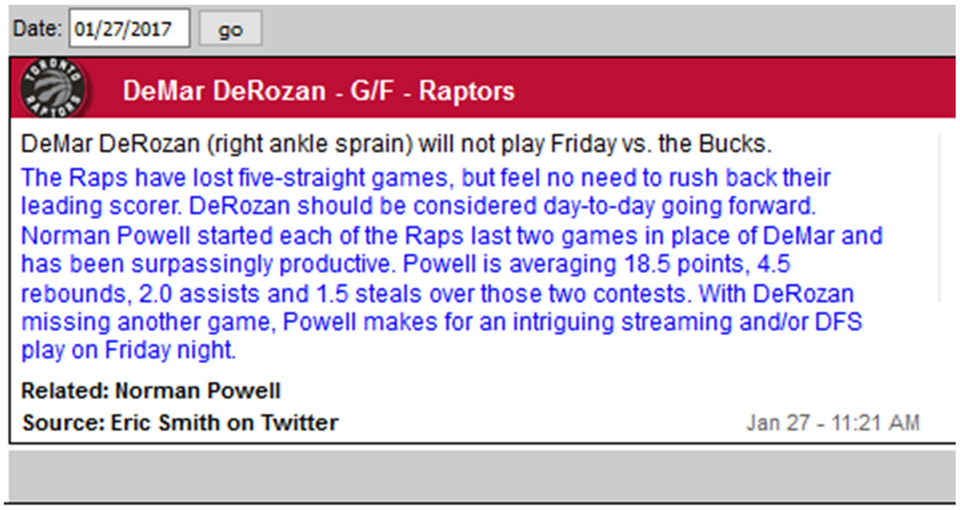
23/1/2017

Player Name	**Predicted Score**	**Actual Score**
Crawford	18	27,5
Anderson	7,9	28,75
Monroe	33,2	37,5
Thaddeus Young	28,9	19,75
Markieff Morris	37	36
James	55,5	62,5
Bogdanovic	24,3	11,5
Harden	63,1	62,65
Total	**268**	**286**

Image 26. Second generated lineup for 23/1/2017

Our second generated lineup reached our target score as well, with Jamal Crawford scoring exactly the fantasy points he averaged in games where Chris Paul was not playing.

For 27th of January 2017 we generated two line ups. One without any player preference and one with Normal Powell included. Image 27 shows Demar DeRozan's injury report prior to the game start and image shows 29 Normal Powell's fantasy points when DeRozan is not playing.



Date: 01/27/2017 go

DeMar DeRozan - G/F - Raptors

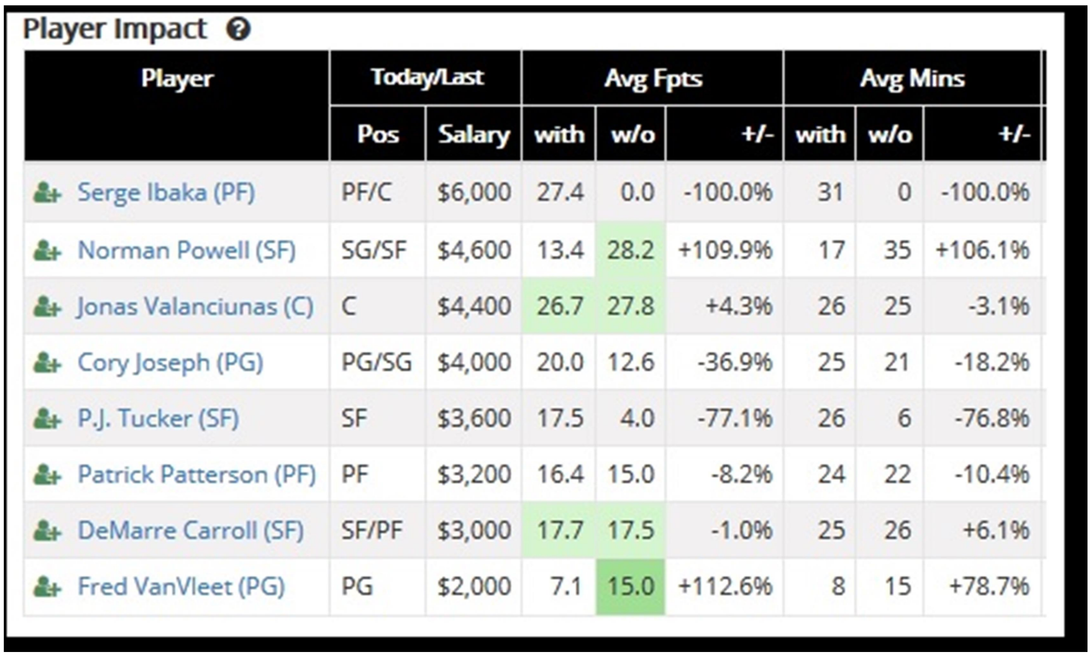
DeMar DeRozan (right ankle sprain) will not play Friday vs. the Bucks.

The Raps have lost five-straight games, but feel no need to rush back their leading scorer. DeRozan should be considered day-to-day going forward. Norman Powell started each of the Raps last two games in place of DeMar and has been surpassingly productive. Powell is averaging 18.5 points, 4.5 rebounds, 2.0 assists and 1.5 steals over those two contests. With DeRozan missing another game, Powell makes for an intriguing streaming and/or DFS play on Friday night.

Related: Norman Powell

Source: Eric Smith on Twitter Jan 27 - 11:21 AM

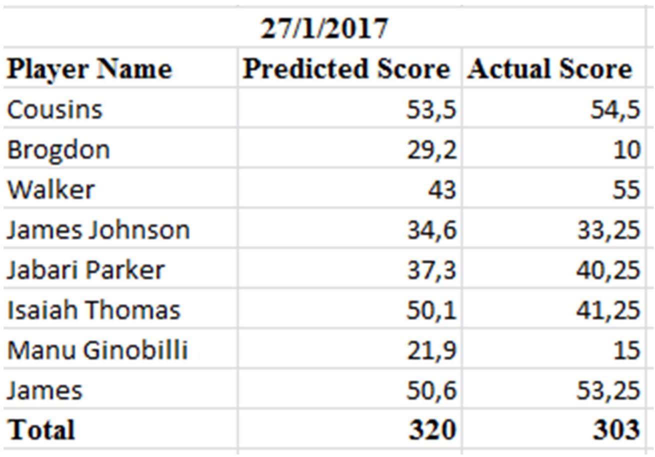
Image 27. DeMar Derozan's Injury report by www.rotoworld.com



Player Impact

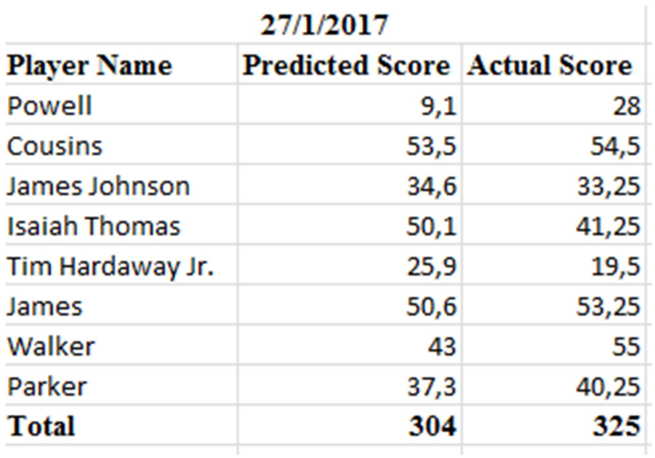
Player	Today/Last		Avg Fpts			Avg Mins		
	Pos	Salary	with	w/o	+/-	with	w/o	+/-
Serge Ibaka (PF)	PF/C	$6,000	27.4	0.0	-100.0%	31	0	-100.0%
Norman Powell (SF)	SG/SF	$4,600	13.4	28.2	+109.9%	17	35	+106.1%
Jonas Valanciunas (C)	C	$4,400	26.7	27.8	+4.3%	26	25	-3.1%
Cory Joseph (PG)	PG/SG	$4,000	20.0	12.6	-36.9%	25	21	-18.2%
P.J. Tucker (SF)	SF	$3,600	17.5	4.0	-77.1%	26	6	-76.8%
Patrick Patterson (PF)	PF	$3,200	16.4	15.0	-8.2%	24	22	-10.4%
DeMarre Carroll (SF)	SF/PF	$3,000	17.7	17.5	-1.0%	25	26	+6.1%
Fred VanVleet (PG)	PG	$2,000	7.1	15.0	+112.6%	8	15	+78.7%

Image 29. Toronto Raptors statistics when DeRozan isn't playing



	27/1/2017	
Player Name	**Predicted Score**	**Actual Score**
Cousins	53,5	54,5
Brogdon	29,2	10
Walker	43	55
James Johnson	34,6	33,25
Jabari Parker	37,3	40,25
Isaiah Thomas	50,1	41,25
Manu Ginobilli	21,9	15
James	50,6	53,25
Total	**320**	**303**

Image 30. First generated lineup for 27/1/2017

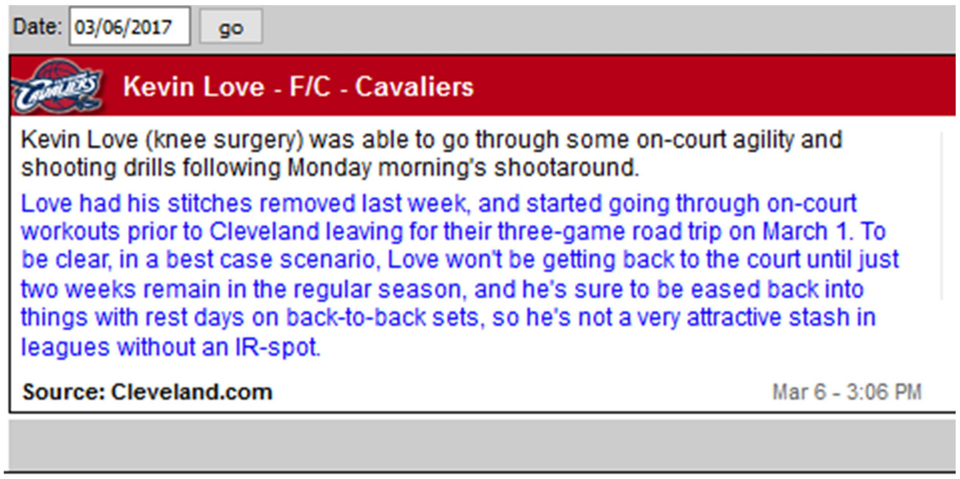


	27/1/2017	
Player Name	**Predicted Score**	**Actual Score**
Powell	9,1	28
Cousins	53,5	54,5
James Johnson	34,6	33,25
Isaiah Thomas	50,1	41,25
Tim Hardaway Jr.	25,9	19,5
James	50,6	53,25
Walker	43	55
Parker	37,3	40,25
Total	**304**	**325**

Image 31. Second generated lineup for 27/1/2017

In the first lineup we run the optimizer without any player preferences. In the second run we forced the optimizer to include Norman Powell in the lineup. Both lineups exceed our target score. In both cases the accuracy of our model was quite decent, without any major mistakes in individual predictions.

We run our last test for the NBA games played at 9 March 2017. At that date Kevin Love, power forward of Cleveland Cavaliers was injured and would not participate at the upcoming game as reported by rotoworld.com.


Date: 03/06/2017 go

Kevin Love - F/C - Cavaliers

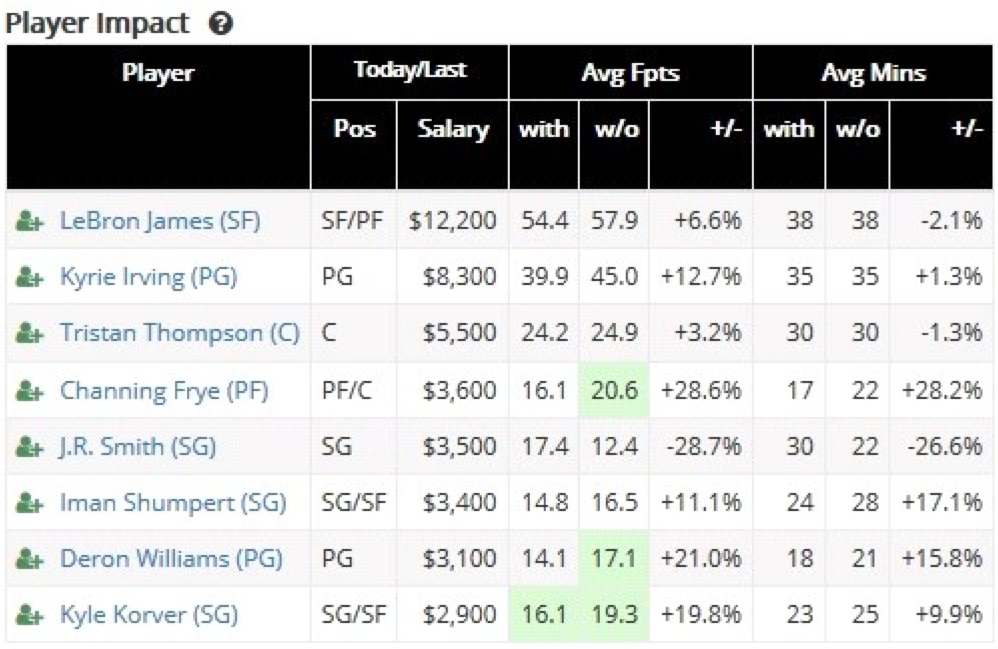
Kevin Love (knee surgery) was able to go through some on-court agility and shooting drills following Monday morning's shootaround.

Love had his stitches removed last week, and started going through on-court workouts prior to Cleveland leaving for their three-game road trip on March 1. To be clear, in a best case scenario, Love won't be getting back to the court until just two weeks remain in the regular season, and he's sure to be eased back into things with rest days on back-to-back sets, so he's not a very attractive stash in leagues without an IR-spot.

Source: Cleveland.com Mar 6 - 3:06 PM

Image 32. Kevin Love injury report

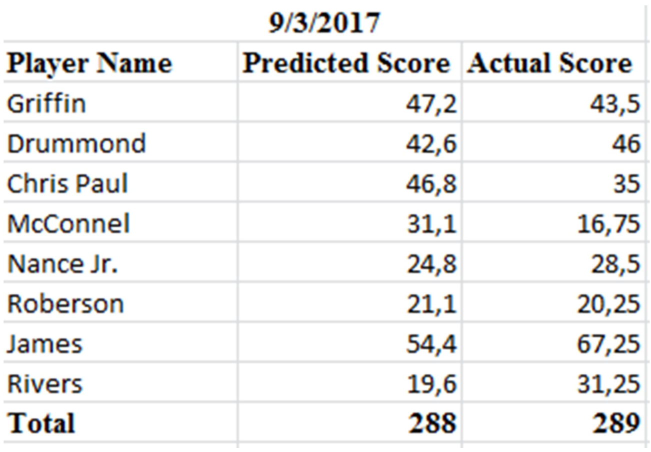
The player who benefits the most when Kevin Love is not playing is power forward Channing Frye, as seen in the image 33.


Player Impact

Player	Today/Last		Avg Fpts			Avg Mins		
	Pos	Salary	with	w/o	+/-	with	w/o	+/-
LeBron James (SF)	SF/PF	$12,200	54.4	57.9	+6.6%	38	38	-2.1%
Kyrie Irving (PG)	PG	$8,300	39.9	45.0	+12.7%	35	35	+1.3%
Tristan Thompson (C)	C	$5,500	24.2	24.9	+3.2%	30	30	-1.3%
Channing Frye (PF)	PF/C	$3,600	16.1	20.6	+28.6%	17	22	+28.2%
J.R. Smith (SG)	SG	$3,500	17.4	12.4	-28.7%	30	22	-26.6%
Iman Shumpert (SG)	SG/SF	$3,400	14.8	16.5	+11.1%	24	28	+17.1%
Deron Williams (PG)	PG	$3,100	14.1	17.1	+21.0%	18	21	+15.8%
Kyle Korver (SG)	SG/SF	$2,900	16.1	19.3	+19.8%	23	25	+9.9%

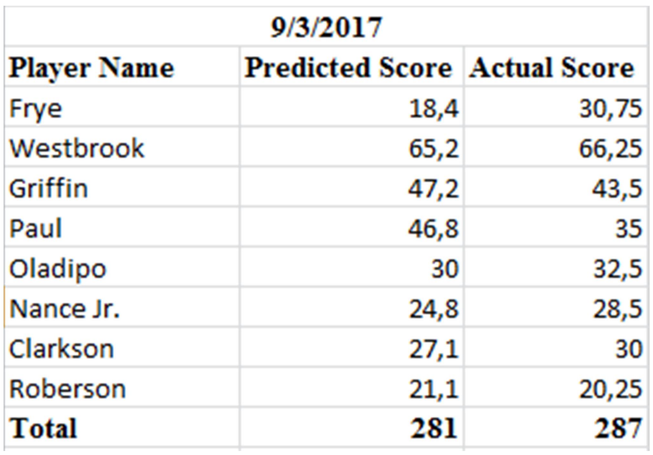
Image 33. Cleveland Cavaliers statistics when Kevin Love is not playing

Once again we generated two lineups, one without restrictions and one with Fry included. Images 35 and 36 show the results



9/3/2017		
Player Name	**Predicted Score**	**Actual Score**
Griffin	47,2	43,5
Drummond	42,6	46
Chris Paul	46,8	35
McConnel	31,1	16,75
Nance Jr.	24,8	28,5
Roberson	21,1	20,25
James	54,4	67,25
Rivers	19,6	31,25
Total	**288**	**289**

Image 35. First generated lineup for 9/3/2017



9/3/2017		
Player Name	**Predicted Score**	**Actual Score**
Frye	18,4	30,75
Westbrook	65,2	66,25
Griffin	47,2	43,5
Paul	46,8	35
Oladipo	30	32,5
Nance Jr.	24,8	28,5
Clarkson	27,1	30
Roberson	21,1	20,25
Total	**281**	**287**

Image 36. Second generated lineup for 9/3/2017

In this case we actually had better results without adding any player restrictions, even though Frye scored 10 more fantasy points above his average when Kevin Love isn't playing. At the same time the second lineup had better accuracy on individual predictions, with only Chris Paul's prediction being more than 5 fantasy points different than his actual score.

6.4 Evaluation of tests

We tested our model in 12 different dates of 2016-2017 NBA regular season. We set a target score of 280 fantasy points which according to our research, is the lowest score required to enter the prize pool in DraftKings' cash game contests. Moreover we combined our predictions with minor tweaks based on information that can be found online in order to further improve the results. In 9 out of 12 test cases our generated line up reached our target of 280 fantasy points.

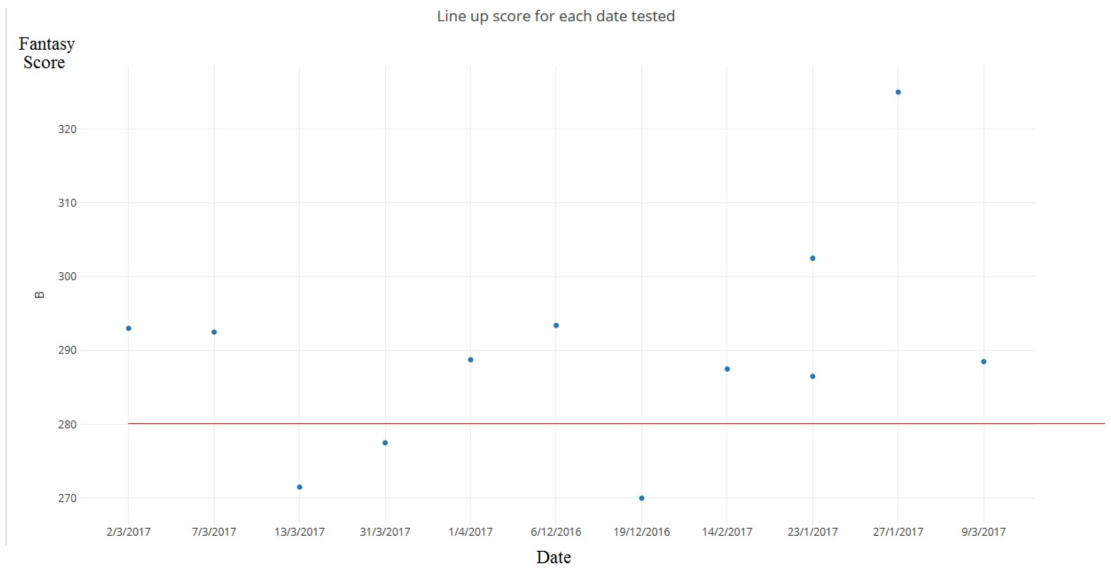


Image 37. Fantasy score for each date tested

In general our model seems to be able to correctly predict the performance of the average NBA player with quite decent accuracy. Our model struggled to produce the optimal results in cases that a player either underperformed or overperformed compared to his recent fantasy performance in an extreme way. In most cases that didn't hurt our results because of the correct predictions on the rest of the players.

6

Technical details

This section will present and analyze the technical details of the algorithms used in this thesis. We will describe how our web scrapper and lineup optimizer works. Technical details about our model were described in chapter 5 so won't be presented here.

6.1 Web Scrapper

In order to collect our data and construct the dataset we built a web scraper which reads player's data from www.dfsgold.com and writes them into a CSV file. The web scraper was built in python 3+. The libraries we used were:

- Pandas 0.20: Pandas is a python library used for data manipulation and analysis. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

- Beautiful Soup: Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with any parser and helps providing idiomatic ways of navigating, searching and modifying the parse tree. It basically presents the parsed data in a more human-friendly form.

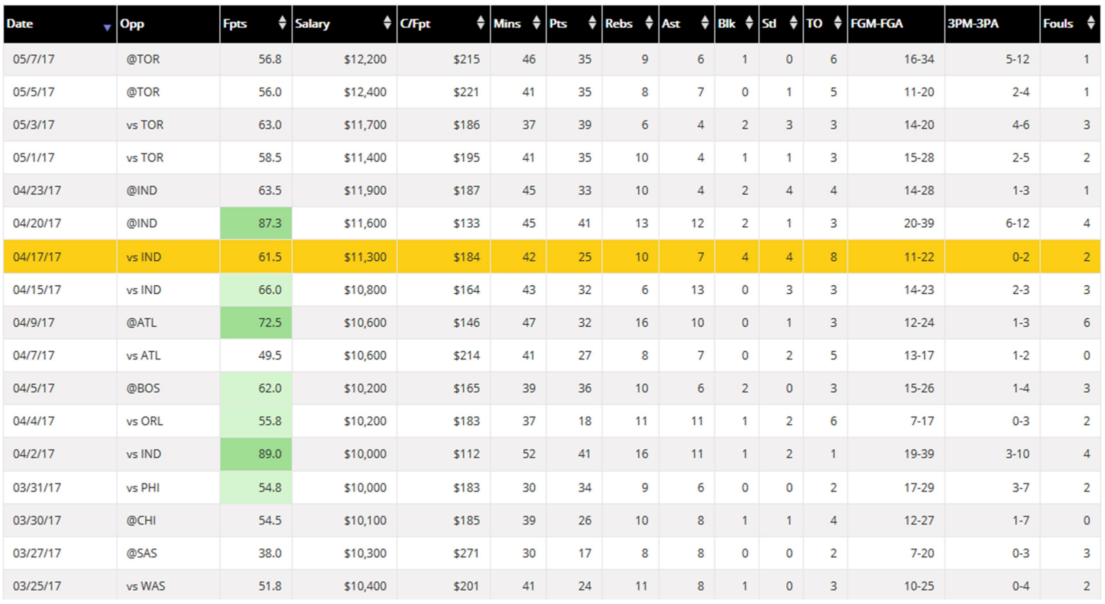
- Request: Requests allows users to send HTTP/1.1 requests, without the need for manual labor. There's no need to manually add query strings to URLs, or to form-encode the POST data.

The scrapper starts by reading a txt file which contains the URLs of the players that we want. It loops through every line of that txt file. The code is shown below:

```
with open("ur.txt","r") as fh:
   for lines in fh:
      lines = lines.rstrip()
```

Considering that the data we want to grab is already in a table format as illustrated at the image 38, we can just add that table contests in a dataframe using pandas library.

dfs = pd.read_html(lines)



Date	Opp	Fpts	Salary	C/Fpt	Mins	Pts	Rebs	Ast	Blk	Stl	TO	FGM-FGA	3PM-3PA	Fouls
05/7/17	@TOR	56.8	$12,200	$215	46	35	9	6	1	0	6	16-34	5-12	1
05/5/17	@TOR	56.0	$12,400	$221	41	35	8	7	0	1	5	11-20	2-4	1
05/3/17	vs TOR	63.0	$11,700	$186	37	39	6	4	2	3	3	14-20	4-6	3
05/1/17	vs TOR	58.5	$11,400	$195	41	35	10	4	1	1	3	15-28	2-5	2
04/23/17	@IND	63.5	$11,900	$187	45	33	10	4	2	4	4	14-28	1-3	1
04/20/17	@IND	87.3	$11,600	$133	45	41	13	12	2	1	3	20-39	6-12	4
04/17/17	vs IND	61.5	$11,300	$184	42	25	10	7	4	4	8	11-22	0-2	2
04/15/17	vs IND	66.0	$10,800	$164	43	32	6	13	0	3	3	14-23	2-3	3
04/9/17	@ATL	72.5	$10,600	$146	47	32	16	10	0	1	3	12-24	1-3	6
04/7/17	vs ATL	49.5	$10,600	$214	41	27	8	7	0	2	5	13-17	1-2	0
04/5/17	@BOS	62.0	$10,200	$165	39	36	10	6	2	0	3	15-26	1-4	3
04/4/17	vs ORL	55.8	$10,200	$183	37	18	11	11	1	2	6	7-17	0-3	2
04/2/17	vs IND	89.0	$10,000	$112	52	41	16	11	1	2	1	19-39	3-10	4
03/31/17	vs PHI	54.8	$10,000	$183	30	34	9	6	0	0	2	17-29	3-7	2
03/30/17	@CHI	54.5	$10,100	$185	39	26	10	8	1	1	4	12-27	1-7	0
03/27/17	@SAS	38.0	$10,300	$271	30	17	8	8	0	0	2	7-20	0-3	3
03/25/17	vs WAS	51.8	$10,400	$201	41	24	11	8	1	0	3	10-25	0-4	2

Image 38. Player statistics page from www.DFSgold.com

Next we remove the commas (,) in the salary columns because we will use commas as separator in the CSV file. Moreover we sort the dataframe by date in an ascending order.

pd.to_datetime(df.Date).order().index

df=df.ix[pd.to_datetime(df.Date).order().index]

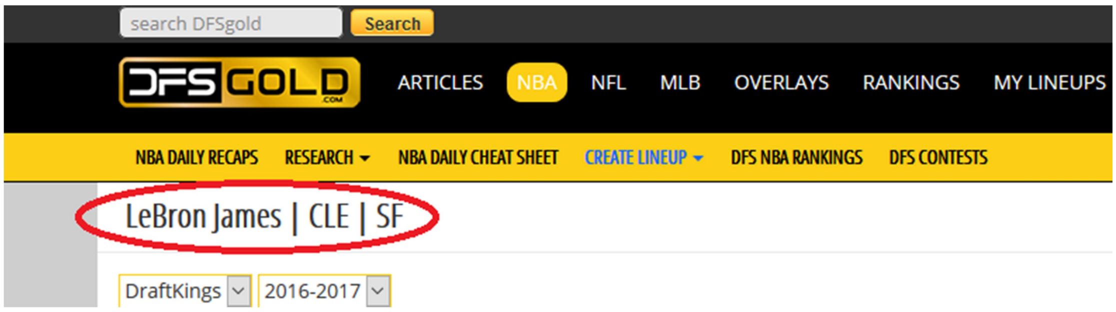
df['Salary'] = df['Salary'].str.replace(',','')

The next step is to remove any data we won't use. This step could potentially be skipped and clear the data while creating the dataset but we had already decided which statistical categories we would use.

data = df.drop(['Opp',"C/Fpt","FGM-FGA","3PM-3PA"],1)

data = data[["Date","Pts","Rebs","Ast","Stl","Blk","Fpts","Mins","Salary"]]

Even though our data is ready we still needed to get the player's name in order to use it later for naming our CSV file. Using request and beautiful soup libraries, we get the name of the player as shown in the image 39.



LeBron James | CLE | SF

Image 39. Name of the player

Moreover we removed the all the texy after the first | character. Finally we wrote our data to a CSV file and used player's name as file name.

```
r =requests.get(lines)

soup = BeautifulSoup(r.content)

for table in soup.find_all('h2', attrs={'class': 'page-title pull-left'}):

   print(1)

filename=table.text


filename=filename.split('|', 1)[0] + '|'

filename=filename.replace('|', '')

data.to_csv(filename+".csv", sep=';',index = False)
```

6.2 Lineup Optimizer

The focus on this thesis was to predict certain values (player's fantasy score) using machine learning models. But in order to test the result and make the predictions usable, a lineup needed to be created based on those predictions. This lineup consisting of 8 NBA players, have to fulfill a number of constraints in order to be eligible for submission in upcoming contests. This kind of problem is called Knapsack problem.

In Knapsack problem the goal is to maximize $\sum_{i=1}^{N} vixi$, subject to

$$\sum_{i=1}^{N} \mathbf{w_i x_I} \leq \mathbf{W}, \mathbf{x_i} \geq \mathbf{0}$$

and x_i integer (or x_i binary), for i =1 , 2, …., N; where v_i, w_i and W are known integers and w_i (i = 1,2,….N) and W are positive.

In programming, this problem can be solved using a dynamic programming algorithm by computing the optimal value given a list of items with values and weights, and a maximum allowed weight. In our case we had to choose the 8 players with the best values while taking into account the following constraints:

- The total amount of salary can't be more than 50.000 $
- There has to be at least 1 player who's eligible for point guard position (PG)
- There has to be at least 1 player who's eligible for shooting guard position (SG)
- There has to be at least 1 player who's eligible for small forward position (SF)
- There has to be at least 1 player who's eligible for power forward position (PF)
- There has to be at least 1 player who's eligible for Center position (C)
- There has to be at least 1 player who's eligible for either point guard or shooting guard position (G)
- There has to be at least 1 player who's eligible for either small forward or power forward position (F)

Looking at the python code we got 2 classes and 4 functions. The player class defines the characteristics for each player. Each player has a name, 1 or more positions (pos), a salary (cost) and a projected value.

```
class player:
  cost = 0
  projected_value = 0
  name = ""
  pos = [0, 0, 0, 0, 0]

  def __init__(self, c, pro_val, name, possible_positions, force_include):
    self.cost = c
    self.projected_value = pro_val
    self.pos = possible_positions
    self.name = name
    self.include = force_include
```

Class entry checks the constraints for every player that is added to the list and calculates the new values

```
class entry:
  def __init__(self, past_entry, new_player, index):
    self.value = -1
    self.cost = -1
    self.cur_spots = [1, 1, 1, 1, 1, 1, 1, 1, 1]
    self.player_list = []
    self.past_index = -1

    if (past_entry is None):
      if (index == -1):
        return
      else:
        self.value = new_player.projected_value
        self.cost = new_player.cost
```

```
                self.past_index = index
                self.player_list.append(index)
                return


        self.value = past_entry.value + new_player.projected_value

        self.cost = past_entry.cost + new_player.cost
        self.past_index = index

        for i in range(9):
            self.cur_spots[i] = past_entry.cur_spots[i]


        for i in past_entry.player_list:
            self.player_list.append(i)

        self.player_list.append(index)
```

Function get_player_list reads the CSV file that contains our data. Next it appends players in groups of 8, creating every combination possible.

```
def get_player_list(possible_name):


    file_name = 'dffinal.csv'

    player_list = []
    with open(file_name) as csvfile:
        reader = csv.reader(csvfile, delimiter=',')
        reader.next()
        for row in reader:
            if (int(row[4]) == -1):
                continue

            name = row[0]
            pos_p = get_possible_positions(row[1])
            c = row[2]
            v = row[3]
            my_p = player(int(c) / 100, float(v), name, pos_p, int(row[4]))

            player_list.append(my_p)

    return player_list
```

Function find_best_player is the main function of the code. After reading the list of all possible combinations, it checks which combinations actually fit our criteria and then prints the one with the best value, which is the combination with the highest total fantasy score.

```
def dp_find_best(player_list, aflag):
    score_table = []

    force_list = []
    for p in range(len(player_list)):
        if (player_list[p].include == 1):
            force_list.append(p)

    for cost_max in range(501):
        col = []
        for play_num in range(9):
            best_index = -1
            best_score = -1
            break_afterwards = False
            for player in range(len(player_list)):

                if (play_num < len(force_list)):
                    player = force_list[play_num]
                    break_afterwards = False

                if (player_list[player].cost > cost_max):
                    if (break_afterwards):
                        break
                    continue

                potential_score = -1
                if (play_num != 0):
                    if (score_table[cost_max - player_list[player].cost][play_num - 1].value > 0):
                        potential_score = score_table[cost_max - player_list[player].cost][play_num -
1].value + \
                                          player_list[player].projected_value
                    else:
                        potential_score = -1
                else:
                    potential_score = player_list[player].projected_value

                if (potential_score > best_score):
                    if (play_num == 0):
                        best_score = potential_score
                        best_index = player
                        if (break_afterwards):
                            force_list.pop()
                            break

                        continue

                    if (add_player_possible(score_table[cost_max -
player_list[player].cost][play_num - 1],
                                           player_list[player], player)):
                        best_score = potential_score
                        best_index = player

                if (break_afterwards == True):
                    force_list.pop()
                    break

            if (cost_max > 0 and score_table[cost_max - 1][play_num].value > best_score):
                col.append(score_table[cost_max - 1][play_num])
            else:

                if (best_score == -1):
                    col.append(entry(None, None, -1))
                elif (play_num == 0):
                    col.append(entry(None, player_list[best_index], best_index))
                else:

                    col.append(entry(score_table[cost_max - player_list[best_index].cost][play_num
- 1],
                                    player_list[best_index], best_index))
```

7

Conclusion

In this chapter we will present the conclusion of this project and sum up the success rate of our model. We will also describe the ways that the model can be improved in the future.

7.1 Outline and conclusions

This thesis was an attempt to find possible applications of machine learning in daily fantasy sports. The growing popularity of daily fantasy sports along with them being a relative new and unexplored domain, made fantasy sports an interesting field to apply machine learning principles and techniques. Moreover the goal of fantasy sports, which is to select a high scoring line up, make them ideal for applications of prediction models.

Our goal was to create a model which would generate NBA lineups with potential to stay profitable when submitted in daily fantasy contests. We focused on NBA cash games because they are considered easier to predict in comparison with other sports like baseball and American football and GPP contests. Moreover due to the different scoring format on daily fantasy sites, our lineups would be suitable for draftkings only. But with some minor tweaks in the database it can be used for any daily fantasy website.

To evaluate our success we set a target score and also compared our results with paid services which offer daily fantasy sports predictions. We avoided testing our lineups in real time contests on draftkings due to the large amount of money that would be required to actually submit a big amount of lineups. But we provided evidence that reaching our target score on any given lineup would translate into profits in the long run.

Our model managed to reach the target score on the majority of dates that was tested, even on cases which the prediction on certain players was way off their actual score. At the same time it produced way better results than some websites that offer fantasy sports predictions. We believe that using our model combined with some basic human insight can result into profits for a daily fantasy sports player.

7.2 Future work

We developed our model while focusing on only one sport, one type of contest and one website. Adjusting the model for other websites would be quite simple, but a different sport like American football or even a different type of contest for the same sport would require a whole different approach. Basketball is considered quite easy to predict with the lowest amount of variance compared to other sports. We believe that developing a model for another sport would require a huge amount of time and resources and still the results would not be as good. The same applies for GPP contests.

As we mentioned, the minutes a player is going to play have a huge impact in his fantasy score. We tried to predict those minutes using player's average minutes in last games combined with betting lines, which is a method a lot of experienced fantasy players use. Even though the results were decent there is still room for improvement and more automation in this area. Reading and analyzing Twitter feeds, taking into consideration the time span between player's last games could improve the accuracy of predicted player minutes. Those examples could be even used for developing a machine learning model for minute predictions.

Except from the minutes played there are a lot more factors that impact a player's fantasy performance. There are measured statistics like the points allowed by a team for a specific position, team's allowed points per minutes, the pace that a team plays (which usually results on more points scored), etc. These statistics can be implemented in our model and improve it even more.

Daily fantasy sports it's a competitive online game which constantly grows in popularity. Right now it is considered to be on early stages and there are a lot of new players joining every year. But as players become more experienced and the use of prediction models becomes the norm, winning a contest and stay profitable in the long term becomes harder. Moreover having a big starting capital to invest can give the player an advantage and minimize the risk. At the end of the day, even though winning in daily fantasy sports or games like poker require a certain amount of skill, it's still a form of gambling which means luck will always be a big factor.

8

Annex

8.1 Web Scrapper

Requirements:

- Pytthon 3.2+
- Request python library
- BeatufiulSoup python library
- Pandas python library

Code:

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

with open("ur.txt", "r") as fh:
    for lines in fh:
        lines = lines.rstrip()
        print(repr(lines))

        dfs = pd.read_html(lines)

        for df in dfs:
            print(2)
        d3 = len(df.index)
        print (d3)
        if d3 > 2:

            pd.to_datetime(df.Date).order().index
            df = df.ix[pd.to_datetime(df.Date).order().index]
            df['Salary'] = df['Salary'].str.replace(',', '')

            data = df.drop(['Opp', "C/Fpt", "FGM-FGA", "3PM-3PA"], 1)
            data = data[["Date", "Pts", "Rebs", "Ast", "Stl", "Blk", "Fpts", "Mins",
"Salary"]]

            r = requests.get(lines)
            soup = BeautifulSoup(r.content)
            for table in soup.find_all('h2', attrs={'class': 'page-title pull-left'}):
```

```
            print(1)
          filename = table.text

          filename = filename.split('|', 1)[0] + '|'
          filename = filename.replace('|', '')
          data.to_csv(filename + ".csv", sep=';', index=False)
      else:
          pass
```

8.2 Prediction model

Requirements:

- Python 3.2+
- Pandas python library
- Numpy python library
- Sklearn python library
- Glob python library
- Os python library
- CSV files have to be placed in directory : C:\\CSVfiles

Code:

```
import pandas as pd
import numpy as np
from sklearn import cross_validation
from sklearn.ensemble import RandomForestRegressor
import glob
import os

# read files
path = "C:\\CSVfiles\\"
filenames = glob.glob(path + "/*.csv")
for infile in glob.glob(os.path.join(path, '*csv')):

    names = ['Position', 'Date', "PTS", "RB", 'Assist', 'Blk', 'Steals', 'DFS', 'MP',
"Salary"]
    df = pd.read_csv(infile, names=names, error_bad_lines=False)

    df123 = os.path.splitext(os.path.split(infile)[1])[0]
    df.index = df["Date"]
    df555 = df.loc[:3917].tail(7)
    df333 = df[df.index != 3917]
    df333 = df333.iloc[::-1]
    forecast_col = 'DFS'
    df333['label'] = df[forecast_col]
    labels = df333[forecast_col]
    data = np.array(df333.drop(['Date', 'Position'], 1))

    # define X and Y
    X = np.empty([len(data) - 6, 6 * len(data[0])])
```

```
for i in range(0, len(data) - 6):
    if i > len(data) - 1:
        break

    result = np.array(data[i])
    for j in range(1, 6):
        result = np.concatenate((result, data[i + j]))
    X[i] = result

y = labels[6:]
X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, y, test_size=0.01)

# Randomf Forest
clfRFR = RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=22,
                               max_features=7, max_leaf_nodes=None, min_impurity_decrease=0.0,
                               min_impurity_split=None, min_samples_leaf=1,
                               min_samples_split=2, min_weight_fraction_leaf=0.0,
                               n_estimators=50, n_jobs=1, oob_score=False, random_state=None,
                               verbose=0, warm_start=False)
clfRFR.fit(X, y)
nn = clfRFR.predict(X_train)
n2 = clfRFR.predict(X_test)
yR_true, yR_pred = y_test, clfRFR.predict(X_test)

# linear
clfLNR = LinearRegression()
clfLNR.fit(X, y)
nnL = clfLNR.predict(X_train)
nl2 = clfLNR.predict(X_test)
yLNR_true, yLNR_pred = y_test, clfLNR.predict(X_test)

names = ['Postion', 'Date', "PTS", "RB", 'Assist', 'Blk', 'Steals', 'DFS', 'MP',
"Salary"]

# datframe for validation
dfnew = df555

forecast_col2 = 'DFS'
dfnew['label'] = dfnew[forecast_col2]
df.dropna(inplace=True)
labels3 = np.array(dfnew['label'])

# define X and Y
data2 = np.array(dfnew.drop(['Date', 'Position'], 1))
Xnew = np.empty([len(data2) - 6, 6 * len(data2[0])])
for i in range(0, len(data2) - 6):
    if i > len(data2) - 1:
        break

    result2 = np.array(data2[i])
    for j in range(1, 6):
        result2 = np.concatenate((result2, data2[i + j]))
    Xnew[i] = result2
ye = labels3[6:]
df5 = dfnew[['Salary', 'Date', 'Position']]
```

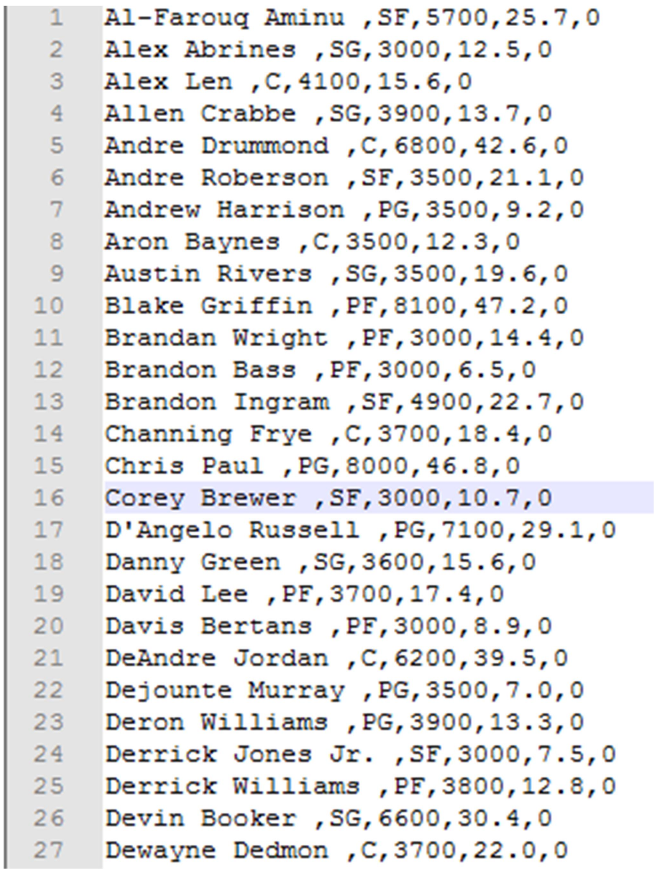
```
    ss = pd.DataFrame(clfRFR.predict(Xnew))
    yLNR2_true, yLNR2_pred = ye, clfRFR.predict(Xnew)

    # Set date and write to csv
    dfdoul = dfnew[(dfnew['Date'] == 3917)]
    dfdoul.reset_index(drop=True, inplace=True)
    dfdoul2 = dfdoul.drop(['Date', "PTS", "RB", 'Assist', 'Blk', 'Steals', 'DFS', 'MP',
'label'], 1)
    dfdoul2['filename'] = [df123]
    dfdoul3 = pd.concat([dfdoul2, ss], ignore_index=True, axis=1)
    dfdoul3.columns = ['Position', 'Salary', 'Name', 'Pred']
    dfdoul3['Pred'] = dfdoul3['Pred'].apply(lambda x: round(x, 1))
    dfdoul3 = dfdoul3[['Name', 'Position', 'Salary', 'Pred']]
    dfdoul3.to_csv('dffinal', mode='a', index=False, header=None)
```

8.3 Lineup Optimizer

Requirements:

- Python 2.7
- A CSV file named dffinal.csv with the format shown in the image 40



```
Al-Farouq Aminu ,SF,5700,25.7,0
Alex Abrines ,SG,3000,12.5,0
Alex Len ,C,4100,15.6,0
Allen Crabbe ,SG,3900,13.7,0
Andre Drummond ,C,6800,42.6,0
Andre Roberson ,SF,3500,21.1,0
Andrew Harrison ,PG,3500,9.2,0
Aron Baynes ,C,3500,12.3,0
Austin Rivers ,SG,3500,19.6,0
Blake Griffin ,PF,8100,47.2,0
Brandan Wright ,PF,3000,14.4,0
Brandon Bass ,PF,3000,6.5,0
Brandon Ingram ,SF,4900,22.7,0
Channing Frye ,C,3700,18.4,0
Chris Paul ,PG,8000,46.8,0
Corey Brewer ,SF,3000,10.7,0
D'Angelo Russell ,PG,7100,29.1,0
Danny Green ,SG,3600,15.6,0
David Lee ,PF,3700,17.4,0
Davis Bertans ,PF,3000,8.9,0
DeAndre Jordan ,C,6200,39.5,0
Dejounte Murray ,PG,3500,7.0,0
Deron Williams ,PG,3900,13.3,0
Derrick Jones Jr. ,SF,3000,7.5,0
Derrick Williams ,PF,3800,12.8,0
Devin Booker ,SG,6600,30.4,0
Dewayne Dedmon ,C,3700,22.0,0
```

Image 40. CSV file example for lineup optimizer

Code:

```
import csv
import sys
import getopt

#create a class that defines player's attributes
class player:
    cost = 0
    projected_value = 0
    name = ""
    pos = [0, 0, 0, 0, 0]

    def __init__(self, c, pro_val, name, possible_positions, force_include):
        self.cost = c
        self.projected_value = pro_val
        self.pos = possible_positions
        self.name = name
        self.include = force_include

    def print_self(self):
        print("%s, %f, %f" % (self.name, self.projected_value, self.cost))

#check the constraints for every player that is added
class entry:
    def __init__(self, past_entry, new_player, index):
        self.value = -1
        self.cost = -1
        self.cur_spots = [1, 1, 1, 1, 1, 1, 1, 1, 1]
        self.player_list = []
        self.past_index = -1

        if (past_entry is None):
            if (index == -1):
                return
            else:
                self.value = new_player.projected_value
                self.cost = new_player.cost
                self.past_index = index
                self.player_list.append(index)
                return


        self.value = past_entry.value + new_player.projected_value

        self.cost = past_entry.cost + new_player.cost
        self.past_index = index

        for i in range(9):
            self.cur_spots[i] = past_entry.cur_spots[i]


        for i in past_entry.player_list:
            self.player_list.append(i)

        self.player_list.append(index)


def add_player_possible(past_entry, new_player, player_index):
```

```python
    for current_player in past_entry.player_list:
        if (current_player == player_index):
            return False

    player_position = -1

    for i in range(5):
        if (new_player.pos[i]):
            player_position = i
            break

    if (player_position < 4):
        start_index = player_position * 2

        if (past_entry.cur_spots[start_index + 1]):
            return True
        else:
            return False

    else:
        if (past_entry.cur_spots[8]):
            return True
        else:
            return False


def dp_find_best(player_list, aflag):
    score_table = []

    force_list = []

    for p in range(len(player_list)):
        if (player_list[p].include == 1):
            force_list.append(p)

    for cost_max in range(501):
        col = []
        for play_num in range(9):
            best_index = -1
            best_score = -1
            break_afterwards = False
            for player in range(len(player_list)):

                if (play_num < len(force_list)):
                    player = force_list[play_num]
                    break_afterwards = False

                if (player_list[player].cost > cost_max):
                    if (break_afterwards):
                        break
                    continue

                potential_score = -1
                if (play_num != 0):
                    if (score_table[cost_max - player_list[player].cost][play_num - 1].value > 0):
```

```
                    potential_score = score_table[cost_max - player_list[player].cost][play_num -
1].value + \
                                         player_list[player].projected_value
                else:
                    potential_score = -1
            else:
                potential_score = player_list[player].projected_value

            if (potential_score > best_score):
                if (play_num == 0):
                    best_score = potential_score
                    best_index = player
                    if (break_afterwards):
                        force_list.pop()
                        break

                    continue

                if (add_player_possible(score_table[cost_max -
player_list[player].cost][play_num - 1],
                                          player_list[player], player)):
                    best_score = potential_score
                    best_index = player

            if (break_afterwards == True):
                force_list.pop()
                break

        if (cost_max > 0 and score_table[cost_max - 1][play_num].value > best_score):
            col.append(score_table[cost_max - 1][play_num])
        else:

            if (best_score == -1):
                col.append(entry(None, None, -1))
            elif (play_num == 0):
                col.append(entry(None, player_list[best_index], best_index))
            else:

                col.append(entry(score_table[cost_max - player_list[best_index].cost][play_num
- 1],
                                   player_list[best_index], best_index))


    score_table.append(col)


print("The highest projected score is: %f" % (score_table[500][7].value))

for p in score_table[500][7].player_list:
    player_list[p].print_self()

#define player positions
def get_possible_positions(pos):
    ps = [0, 0, 0, 0, 0]
    if (pos == 'PG'):
    ps[0] = 1
elif (pos == 'SG'):
    ps[1] = 1
elif (pos == 'SF'):
    ps[2] = 1
elif (pos == 'PF'):
    ps[3] = 1
elif (pos == 'C'):
    ps[4] = 1
else:
    print("POSTION NOT POSSIBLE!!!!")

return ps

#read the csv
def get_player_list(possible_name):


file_name = 'dffinal.csv'

player_list = []
with open(file_name) as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    reader.next()
    for row in reader:
        if (int(row[4]) == -1):
            continue

        name = row[0]
        pos_p = get_possible_positions(row[1])
        c = row[2]
        v = row[3]
        my_p = player(int(c) / 100, float(v), name, pos_p, int(row[4]))

        player_list.append(my_p)

return player_list


def main(argv):
aflag = True
fflag = False

try:
    opts, args = getopt.getopt(argv, "af:")
except getopt.GetoptError:
    print ("python optimal_lineup.py -a -f <optional_file_name>")
    sys.exit(2)

player_filename = ""


player_list = get_player_list(player_filename)

dp_find_best(player_list, aflag)
```

```
if __name__ == "__main__":
    main(sys.argv[1:])
```

9

References

[1] http://money.cnn.com/2015/10/06/news/companies/fantasy-sports-101/index.html

[2] https://fivethirtyeight.blogs.nytimes.com/2011/04/20/after-black-friday-american-poker-faces-cloudy-future/

[3] https://www.bloomberg.com/news/articles/2015-09-10/you-aren-t-good-enough-to-win-money-playing-daily-fantasy-football

[4] Xiaojin Zhu: Semi-Supervised Learning Literature Survey, University of Winsconsin, 2008

[5] S. B. Kotsiantis, Supervised Machine Learning: A Review of Classification Techniques, University of Peloponnesse, Greece, 2007.

[6] Khanum, M., Mahboob, T., Imtiaz, W., Abdul Ghafoor, H. and Sehar, R. (2015). A Survey on Unsupervised Machine Learning Algorithms for Automation, Classification and Maintenance. International Journal of Computer Applications, 119(13), pp.34-39., 41, 2, pp. 54-57, 1998.

[7] Vrushali Y Kulkarni, Dr Pradeep K Sinha, Random Forest Classifiers: A survey and Future Research Directions, International Journal of Advanced Computing, Vol:36, Issue:1, 2013

[8] Breiman, L. (2001). *Machine Learning*, 45(1), pp.5-32

[9] Lee, A. (2012). *Linear Regression Analysis*. Wiley.

[10] Hearst, M., Dumais, S., Osuna, E., Platt, J. and Scholkopf, B. (1998). Support vector machines, *IEEE Intelligent Systems and their Applications*