DS 5500 Project Report

Tian Sang
Zhaoxi Wang

10/30/2019

1. Summary

In this project, we are going to focus on the problem that how to pick the best lineup in online fantasy NBA game.

First, in case that you are not familiar with online fantasy sports, we will give a short introduction here. Online fantasy sports, take the NBA as an example, is that you pick 8 players from a pool of all current NBA players before each match day to form your lineup. Noticed that picking 8 players must follow some rules, like you have a salary cap of $50,000 and players must be in different positions & teams. After the match day, each player will get his performance score which is calculated using an equation. Now we compare the summed performance score for all lineups. The winner goes to the lineup with the highest score.

Our goal in this project is to use Machine Learning method to predict players' performance scores. And based on this, following the constraints, we optimize the lineup using the optimization method proposed by Hunter et al.[1].

How should we evaluate our project? Draftkings[2] is the biggest platform for fantasy game. We have collected some past game records, so that we can compare our lineup result to other's to tell if we win. Also, FantasyCrunch[3] is a website that sells lineup strategy. We can also compare our predicting results with theirs.

2. Methods

2.1 Collect data

There are 3 parts of data we need to collect:

2.1.1
2018-2019 season's players stats which is our historical player performance data to build prediction model; After some research we find the fantasy sports domain doesn't have too much open source data available. Thus we decided to write our own api to scrape data from NBA.com. Fortunately we were able to approach two main data sources: team table which contains team and player information, game_log table which contains player's stats for every

single game. After various data ETL we were able to get the 2018-2019 season's game_log, which give us all player's attribute in all games . In total we have around 2.5k rows, each row is a player's stats in one game.(we have around 600 players and 400 games in one season), and each column is an attribute for a player in one single game(points, assist, block, team, win/lose, health condition, salary, … etc)

We initially want test our algorithm in real time, namely this NBA season begins from October 2019. However, as we spend more time looking at our data and the structure of NBA league, we realize that we should at least wait until pre-season and a couple weeks of regular season's finish to get enough data for our model. (NBA's regular season starts in October 22, 2019 and ends in April 15, 2020, thus for now some team even hasn't started their first regular season match) Thus we will first backtest on last season's data and wait until Feb-2020 to test in real money.

2.1.2 The second part is lineup records from Draftkings.

This part refers to data for the "current day". Namely, a list of players and their updated attributes(position, salary, health condition, etc) in today's game, we would only predict and build optimal lineup from this set as they are the players actually playing. The dataset typically update at 1:pm in a game day, and the first game starts at 7:00 pm, so we have plenty of time to generate our optimal lineups. It's freely available from draftkings.com as a cleaned csv format and easy to access.

2.1.3 The third part is predictions made by FantasyCrunch.

One problem of our project is the difficulty to evaluate. We propose to use MSE or MAE as metrics for our regression problems, however we don't have a clear flag point to define if the model is "good". Thus we would compare our prediction versus a commercial website Fantasy Crunchers prediction. These websites offer same predictions as ours but typically cost $30~$50 monthly subscription fee. We assume their prediction is relatively good as it is a major player in fantasy sports' market.(7 billion market Cap)

2.2 Data cleaning & transformation

Surprisingly, the player stats dataset is pretty clean (no missing value). The thing we do for transformation includes calculating performance score, given by equation:

$$score = PTS + 0.5*FG3M + 1.25*REB + 1.5*AST + 2*STL + 2*BLK - 0.5*TOV + 1.5*DD + 3*TD$$

Variable score will be the target for predicting performance.

2.3 Feature engineering

Since the original player stats dataset is pretty simple, we apply feature engineering. The better the features that we prepare and choose, the better the results we will achieve.

Time series feature: we created many time series features, like “score for the previous game”, “PTS for the previous game”, “PTS for the second previous game”, “avg. PTS for the past five games”, and changing PTS to other stats like REB, FG_PCT etc. for previous one to ten games. The interesting thing is that we tried time series approach at first. But the result was not satisfactory. Instead, embedding time series feature into Machine Learning model has a better result.

Encode categorical feature: categorical features like team, opponent, players exist in the dataset. At first we tried to do the classic one-hot encoding for these features. It turned out the dimensionality of design matrix exploded and the performance was not ideal. Then we wanted to encode these categorical features into embedding vectors, like word2vec. However, we couldn’t find a proper way to do this. So our solution was to create new numerical features that can represent them. “Team winning rate” and “Oppo winning rate” are two features that indicates the winning rate for home team and oppo team before the match day. They are a general representation of the team feature, as it can measure a team’s competence. There are also features like “team avg score”, “team avg PTS” etc. to describe the feature team from different aspects.

2.4 Machine Learning model

For the modeling part, given the dataset is in small volume (20,000+ records), we didn’t choose the trending deep neural nets as the model. Instead, we think tree based models might have good performance as in practice they often do. Some problems we have are there are numerous tree based models and tuning their hyper parameters takes so much time. As a result, we adopt H2O’s autoML as the solution.

H2O autoML can be used for automating the machine learning workflow, which includes automatic training and tuning of many models within a user-specified time-limit. It needs you to provide the target and design matrix, the number of models you want to try, and time-limit for running these models. It will output a table of results for different models (hyperparameter tuned). The reason we choose autoML is that it’s simple & convenient to implement and it’s result is often excellent.

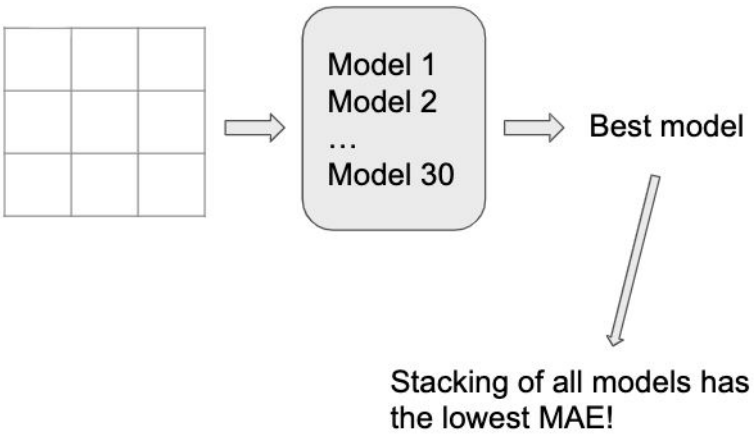

Figure 1: process of autoML modeling. Give it the target and design matrix, it will automatically train using 30 tuned models and produce a table of result.

2.5 Optimization

This chapter we discuss the optimization part of daily fantasy sports.

2.5.1 Distribution of Payoff

There are several payoff structure of daily fantasy sports, in our project we focus on the classic top-heavy game as it's most played and allow multiple entries. By saying payoff structure is top-heavy or "winner takes all", we mean most of the money will be allocated to the several players with top scores. For example, top 10 scored lineup will make 90% of the bounty while 10-50 ranked lineup make the remaining 10%. Imagine one must select a portfolio of entries for a contest where all or nearly all of the winnings go to the top performing entry,, one natural strategy is to select the entries in such a way that maximizes the probability that at least one of them has exceptional performance and wins the contest. In this case, optimizing expected lineup score is not enough, we would also want to maximize the variance inside our portfolio of lineups. Next, we will discuss applying integer programming to programmatically generate a portfolio of lineups that satisfies our requirement.

2.5.2 Formulation of integer programming

Winning daily fantasy sports is a matter of selecting the athletes that earn the most fantasy points. The usefulness of integer programming in decision making is clear, motivating us to apply the technique to create optimal lineups. Again, our goal is to maximize the probability that at least one(or a couple) of our lineups in our whole portfolio has exceptional performance and ranks at the top. We will use integer programming to optimize toward our goal while also satisfy all constraints defined by daily fantasy sports hosters.

There are two steps of optimizing, first step is to maximize expected return of one single lineup, and the second step is to maximize variance between each pair of lineups in our portfolio. We would achieve these two steps simultaneously by sequentially generate a list of lineups, such that all current and next lineup want maximize score, and also increase the pairwise variance by setting a max number of overlapped player.

Optimization Goal for a single Lineup:
maximize expected score

$$\sum_{p=1}^{N} f_p x_{pl}$$

Basic Constraints: we have a salary cap of 5000, and we want to select 8 players.

$$\sum_{j=1}^{p} c_j x_{ij} \leq 5000, \quad \text{(budget constraint)}$$
$$\sum_{k=1}^{p} x_{ij} = 8, \quad \text{(lineup size constraint)}$$

Position Constraints:
we want select 1 center player, 2 point guard player, 2 shooting guard, 2 power forward player, 2 small forward player and one 'freeman' that is allowed to play all positions.

$$1 \leq \sum_{j \in C} x_{ij} \leq 2$$
$$2 \leq \sum_{j \in (PG,SG)} x_{ij} \leq 3$$
$$2 \leq \sum_{j \in (PF,SF)} x_{ij} \leq 3$$

Team Constraints:
we want our players come from at least 3 teams

$$t_{il} \leq \sum_{k \in T_l} x_{ik}, \quad l = 1, \ldots, N_T$$
$$\sum_{l=1}^{N_T} t_{il} \geq 3$$
$$t_{il} \in \{0, 1\}, \quad l = 1, \ldots, N_T$$

Next to increase variance within our portfolio, we will set each current lineup k=l (the lineup being generated) is only allowed to have at most γ overlapped player with all previous lineups 1 to l-1. By doing this we are increasing the diversity of our portfolio, or in other words, we are increasing the variance such that at least one or a couple of lineups could be exceptionally good.

Lineup Constraints: (this is for each lineup, not for players in a single lineup)

$$\sum_{p=1}^{N} x_{pk}^{*} x_{pl} \leq \gamma, k = 1, \ldots, l-1$$

3. Results

We first used 30 models and set the time limit for each model to 150 seconds, and ran the autoML on 03/29/2019 subset. Use Mean Absolute Error (MAE) as the metric, you can see the best result came from the stacking of all other 29 models which reached MAE of 7.79.

model_id	mean_residual_deviance	rmse	mse	mae
StackedEnsemble_AllModels_AutoML_20191029_185211	97.465	9.87244	97.465	7.79825
GLM_grid_1_AutoML_20191029_185211_model_1	97.76	9.88736	97.76	7.80035
StackedEnsemble_BestOfFamily_AutoML_20191029_185211	97.8783	9.89335	97.8783	7.81343
XGBoost_grid_1_AutoML_20191029_185211_model_2	99.8868	9.99434	99.8868	7.88746
GBM_5_AutoML_20191029_185211	100.385	10.0192	100.385	7.9257

Then, we can compare our prediction with FantasyCrunch's prediction. We ran the model for 02/01/2019 - 03/31/2019 subset and calculated the average MAE:

```
2019-02-01
our mae: 7.574870246982369
fc mae: 7.340941176470588
----
2019-02-02
our mae: 8.097739493579375
fc mae: 7.746304347826088
----
2019-02-03
our mae: 7.129022610433484
fc mae: 8.01826923076923
----
2019-02-04
our mae: 8.463043336264581
fc mae: 8.417399999999999
----
2019-02-05
our mae: 7.67428420692324
fc mae: 8.11768656716418
----
2019-02-06
our mae: 7.543874364199293
fc mae: 7.492991452991453
----
2019-02-07
our mae: 7.253862397826567
fc mae: 7.494639175257732
```

Figure 2: result of MAE

FC's MAE	Our's MAE
7.80	7.67

It turned out that our result had better performance on MAE. And the next is the actual lineups we had.

	0	1	2	3	4	5
entry1	Russell_Westbrook	Karl_Anthony_Towns	Brandon_Ingram	Kyle_Kuzma	Dennis_Schroder	Reggie_Bullock
entry2	Joel_Embiid	Ben_Simmons	DeAndre_Jordan	Brandon_Ingram	Dennis_Smith_Jr.	Domantas_Sabor
entry3	Karl_Anthony_Towns	Damian_Lillard	Ben_Simmons	Patrick_Beverley	Alec_Burks	Al_Farouq_Aminu

Figure 3: actual lineups

Also, we had the lineup information from Draftkings. So we can test our optimized lineup in a real competition. We formed 49 lineups for the 02/01/2019 competition and it turns out that 4 of them actually won but none of them ranked in the top 20, which would give us a 16% loss. As commercial website don't keep their payoff data available we cannot do further backtest at this point. However, as new NBA season kicks off, we will follow that and keep testing and modifying our algorithms.

At this point, we are not feeling confident in our algorithm and decided not to invest in real money yet. Although we can imagine beating the market would be hard, our goal is to improve our approach until we can systematically make positive return in the long run.

4. Reference

[1] Hunter, David Scott, Juan Pablo Vielma, and Tauhid Zaman. "Picking winners using integer programming." *arXiv preprint arXiv:1604.01455* (2016).

[2] www.draftkings.com

[3] www.fantasycruncher.com