Machine Learning Engineer Nanodegree

Capstone Project Report

Kengo Arao

May 21, 2018

1 Definition

1.1 Project Overview

Statistics has always been an integral part of player evaluation in the game of basketball, ranging from simple field goal shooting percentage to rather comprehensive efficiency metrics such as offensive ratings introduced by Oliver (2004). Nevertheless, the abundance of statistical indicators is less utilised in systematic forecasting of inndividual player performance. This type of analysis has been qualitatively conducted by amateur fans and betters in fantasy sports websites on a daily basis. In the case of fantasy basketball, users select a lineup of 8 players from a given set of games under the salary cap of $50,000, and compete against each other with 'Fantasy Points' (FPTS), that are calculated based on statistics from actual games. The primary objective of this project is to predict performance of the NBA players given data obtained with responsible web scraping of Basketball-Reference.com and RotoGuru. I also implement an optimisation algorithm to create a lineup of 8 players with highest expected fantasy points, and examine how it compares to the predictions.

1.2 Problem Statement

This paper follows the definition of player performance according to DraftKings (2018) with 9 variables - points (PTS), 3 points made (3P), total rebounds (TRB), assists (AST), steals (STL), blocks (BLK), turnovers (TOV), double doubles (DD), and triple doubles (TD). Player i's performance y_i is defined as follows:

$$y_i = PTS_i + 0.5\ 3P_i + 1.25TRB_i + 1.25AST_i + 2STL_i + 2BLK_i - 0.5TOV + 1.5DD + 3TD$$

This project's task is to predict y_i by minimizing the loss function between the actual and predicted values given available data before the tip off. In addition to these 9 variables, we utilise 19 other variables from basic and advanced statistics from Basketball-Reference.com. These variables are aggregated over the past 10 games with weighted average, where statistics from recent games are considered to be more informative.

In addition, I engineer features that would influence player performance from games data, including home-court advantage, the the number of rest days (Barry, Canova, & Capiz, 2017), the number of active players of their position and the whole roster, players' positions, and if a player is a starter or not. Crucially, I include salary information from DraftKings determined by their algorithm and staff members considering qualitative information such as coaching decisions and match-up situations, which are less likely to be captured by the statistics above. In relation to this, 'Value' variable is constructed as a ratio between salary divided 1000 and FPTS. As a heuristic, a player with 'Value' higher than 5 is consider to be on 'fire', yielding more FPTS than his salary-adjusted expectation (Levitan, 2016). In order to address the curse of dimensionality, I reduce the number of features considering correlations betweens variables and feature importances from a boosting regressor.

I comparatively analyse the performances of regression, gradient boosting, and deep learning algorithms using mean absolute error (MAE) and root mean squared error (RMSE) with 5-fold cross validation.

1.3 Metrics

The primary objective is to produce prediction $\hat{y}_i$ for each player i , such that the RMSE is minimised where y_i is the actual performance and N is the number of data points.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

MAE will also be reported to give a sense of linear differences between prediction and actual performance that is an absolute value of the difference between y_i and $\hat{y}_i$.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}(|y_i - \hat{y}_i|)$$

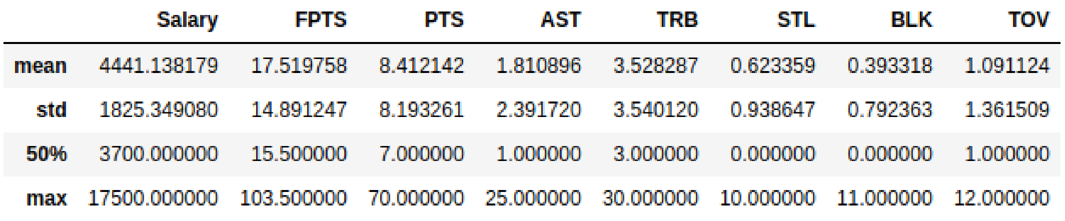
I chose RMSE to be the main loss function as it heavily penalises large deviations of $\hat{y}_i$from y_i due to the squared terms, and this is sensible as one player's underperformance could undermine the lineup significantly.

2 Analysis

2.1 Data Exploration

I obtained box scores from Basketball-Reference.com and salary and position information from RotoGuru for three regular seasons over the course of 2015-2018 period to ensure the number of data points. There were some missing or incomplete values encountered during the data scraping process. Columns for percentages often contained NaN values if the number of attempted shots in a given category is zero. For example, if a player did not attempt a single three-pointer, the 3-pointer percentage column returns NaN. These are replaced by 0s. On the other hand, the rows of players with 'Not with Team' or 'Did Not Dress' were entirely dropped from the dataset with an assumption that they were not expected to play prior to the game which likely indicates injury.

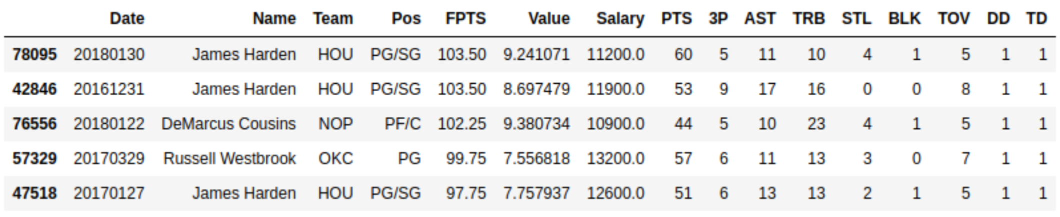
There are 30 teams in the NBA and each team plays 82 games per season with a maximum roster of 15 players. The merged dataset contains a total of 89,406 data points, and Figure 1 shows the maximum, mean, standard deviation and median (50th percentile) of the main variables. It is evident that these variables differ in magnitude and should be scaled in the range of $[0, 1]$. For variables except for salary, the median values do not deviate significantly from the mean value, implying that they are unlikely skewed by outliers.



	Salary	FPTS	PTS	AST	TRB	STL	BLK	TOV
mean	4441.138179	17.519758	8.412142	1.810896	3.528287	0.623359	0.393318	1.091124
std	1825.349080	14.891247	8.193261	2.391720	3.540120	0.938647	0.792363	1.361509
50%	3700.000000	15.500000	7.000000	1.000000	3.000000	0.000000	0.000000	1.000000
max	17500.000000	103.500000	70.000000	25.000000	30.000000	10.000000	11.000000	12.000000

Figure 1: Descriptive Statistics

Figure 2 shows the top 5 performances in terms of FPTS over the three seasons. While these are the opposite of a typical player performance, they give insight into what factors drive great performance. James Harden, DeMarcus Cousins and Russell Westbrook all saw heavy minutes in their 40-plus point games, recording triple doubles with double digits in points, total rebounds and assists. For example, when James Harden recorded a historic 60-point game against Orlando on the 30th January 2018, Houston Rockets were suffering from injuries of key players such as Chris Paul and Trevor Ariza (USAToday, 2018). This motivates the construction of a variable considering the availability of players known prior to the game, as it likely affects the usage of other players. I also expect metrics such as Usage% - which estimates the percentage of possessions a player uses - and minutes played (MP) to be of primary significance regarding predictive power.



	Date	Name	Team	Pos	FPTS	Value	Salary	PTS	3P	AST	TRB	STL	BLK	TOV	DD	TD
78095	20180130	James Harden	HOU	PG/SG	103.50	9.241071	11200.0	60	5	11	10	4	1	5	1	1
42846	20161231	James Harden	HOU	PG/SG	103.50	8.697479	11900.0	53	9	17	16	0	0	8	1	1
76556	20180122	DeMarcus Cousins	NOP	PF/C	102.25	9.380734	10900.0	44	5	10	23	4	1	5	1	1
57329	20170329	Russell Westbrook	OKC	PG	99.75	7.556818	13200.0	57	6	11	13	3	0	7	1	1
47518	20170127	James Harden	HOU	PG/SG	97.75	7.757937	12600.0	51	6	13	13	2	1	5	1	1

Figure 2: Top 5 Performances

2.2 Exploratory Visualisation

Another important aspect when considering player performance is consistency. In modern portfolio theory pioneered by Markowitz (1952), a portfolio of financial assets should maximise the expected return given a level of risk that is proxied by standard deviation. Applying this concept into fantasy basketball, I consider the expected return as the mean FPTS over the previous 10 games and the risk as the standard deviation. Note that this is for individual players and not a collection of players as per the original theory. Figure 3 plots this risk-return relationship for each player as of the 11st April of 2018, with risk on x-axis and expected return on y-axis (an interactive is available here). Players are reclassified into four positions; point guard (PG), shooting guard (SG), forward (F), and center (C), where power forward (PF) and PF/C are classified as center. This plot is relevant as it facilitates identifying players that produce consistent performance on a given level of risk or return. For instance, for a fixed level of risk with standard deviation of around 5 on the x-axis, we see that Khris Middleton outperforms Julius Randle with expected FPTS of 40.85 compared to 33.7. On the other hand, controlling for the expected return at around 50-55 FPTS, one could argue that James Harden is a more consistent player than Kevin Durant during this period with lower standard deviation of 10.70 in comparison to 14.14. This risk-return relationship motivates the introduction of standard deviation of FPTS as a feature later.

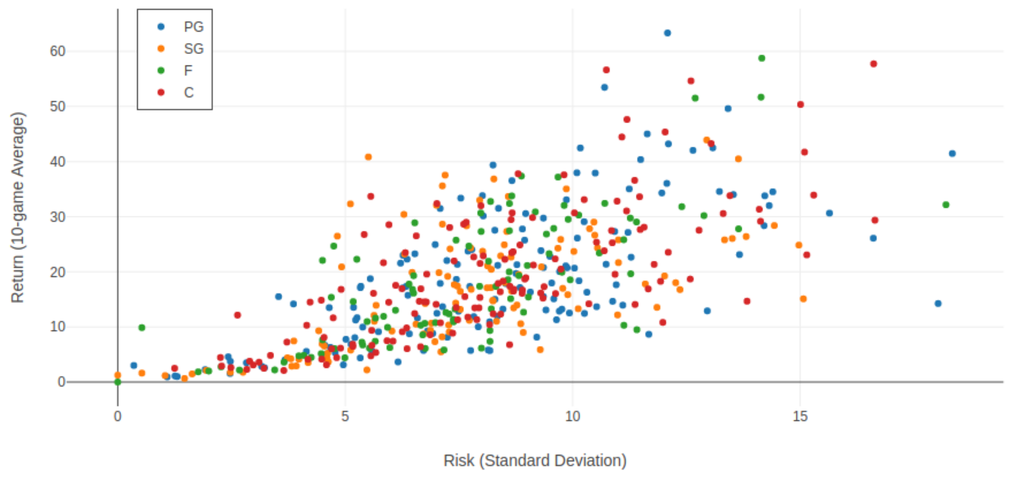


Figure 3: 10-game Risk-Expected Return Relationship

Figure 4 plots salary on the x-axis and actual FPTS on the y-axis with colours based on positions mentioned above based on games on April 11th, 2018, the last day of the 2017-18 regular season. An interactive version is available here. It is intuitive that there is a positive relationship, and this also encapsulates one of the key aspects of playing fantasy sports that is to identify 'Value' players who yield higher returns for their salary in terms of FPTS. Considering that regression line serves as the predictor for the actual FPTS, one could argue that Russell Westbrook exceeded the expectation given his salary of \$11,900 with 62.5 FPTS, and Lebron James performed very poorly based on this salary expectation with only 19.75 PTS for the price of \$11,400. This relationship is incorporated as 'Value' variable in the feature engineering section.

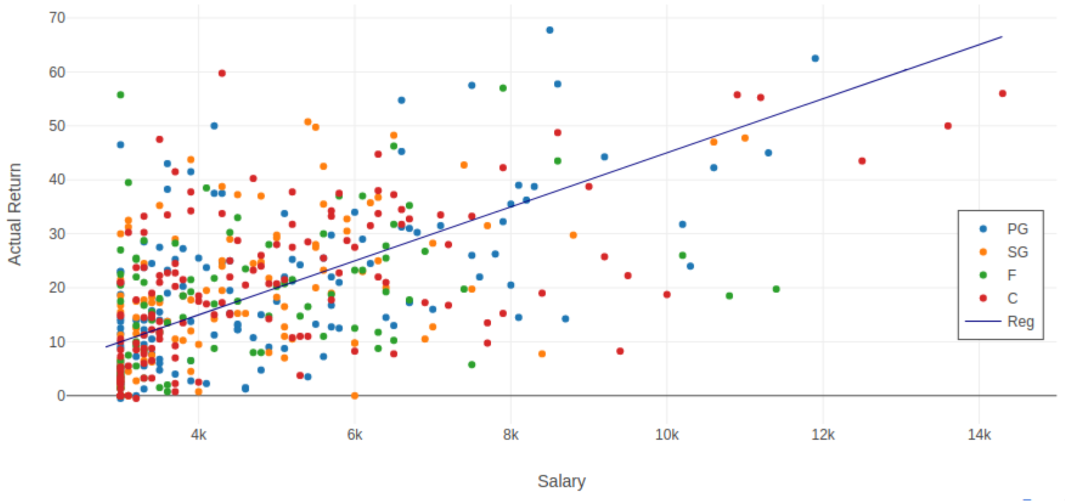


Figure 4: Salary-Return Relationship

2.3 Algorithms and Techniques

This paper comparatively examines the performances of three algorithms; linear regression; gradient boosting; and neural networks. 5-fold cross validation is used for validation with 20% of the dataset reserved for testing at each iteration. Linear regression serves as the starting model since it is computationally inexpensive and does not require parameter tuning. Since the dataset contains a number of dimensions, gradient boosting is expected to perform well as it sequentially ensembles a number of relatively weak and inaccurate rules to create a more accurate predictor (Shapiro, 2013). I chose XGBoost as it better handles the loss function of RMSE for regression (Chen, n.d.) and is proven to perform well in data science competitions. As there are quite a few hyperparameters to tune with most of values being continuous, I use Baysiean optimisation for obtaining the best parameters that models the distribution of expected improvement of the model based on the past trials, which is advantageous over the grid search approach (Pham, 2016). Finally, I also implement deep learning models with dropout layers and early stopping to avoid overfitting. Deep learning models are unlikely to match the performance of XGBoost because of our rather small dataset of less than 100,000 and the fact that we use well-defined statistical variables which might not exhibit hidden structures of importance.

2.4 Benchmark

The benchmark follows the prediction strategy provided by DraftKings, that is the simple average of past games in the season. In other words, to predict player performance of the N-th game, the average of N-1 games for each statistic is considered as the expected value. A linear combination of these statistics according to the rule above gives the expected FPTS. Calculating the average statistics for regular seasons 2015-16, 2016-17 and 2017-18, I obtain the baseline loss of 9.9434 (RMSE) and 7.4285 (MAE).

3 Methodology

3.1 Data Preprocessing

3.1.1 Name Standardisation

The two data sources I scraped the the box score and salary information from were inconsistent in terms of player names, such as; use of nicknames (Guillermo Hernangómez and Willy Hernangómez); spelling mistakes (D.J. Augustine and D.J. Augustin); use of name suffix (Larry Nance Jr. and Larry Nance); and use of dots in first name initials (J.J. Redick and JJ Redick). These problems are circumvented by sending a query to the search function of Basketball-Reference.com for names that show up only in one of the two datasets.

3.1.2 Variable Construction

In addition to the variables that directly go into the calculation of FPTS, the following variables are scraped from Basketball-Reference.com, where advanced statistics provide more realistic estimates of a player's performance in each statistical category in relation to other players. Full glossary is available here.

Basic	PTS	3P	AST	TRB	STL	BLK	TOV	DD	TD	
Additional	MP	FT	FTA	FGA	3PA	DRB%	ORB	FG%	3P%	FT%
Advanced	DRtg	ORtg	USG%	AST%	DRB%	ORB%	BLK%	TOV%	STL%	eFG%

Each variable is aggregated with weighted average over the past 10 previous games such that recent games are weighted more highly. This follows Barry, Canova, & Capiz (2017) in which games are averaged with linearly increasing weights for all the past games in the season. I extend this approach with three different weighting schema; square root; linear; and quadratic, where the coefficients for the n-th game with $n \in [1, 10]$ are given by $\sqrt{n}$, n, n^2, respectively. Weights are normalised such that they sum up to 1. This paper considers a fixed number of past games such that each statistic is comparable across seasons with reduced computational costs. The choice of the number 10 is based on the number of past games displayed in the lineup selection process on DraftKings. Figure 5 graphically shows how much weight is put in each game. Since the first 10 games are ignored for each season, the number of data points is now 75,239. The resulting statistics are standardised such that they are all in the range of $[0, 1]$.

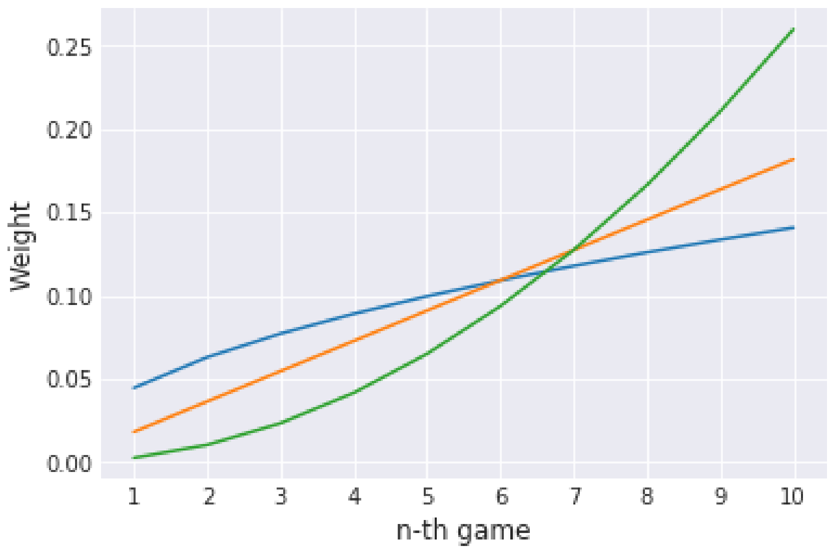


Figure 5: Weighting

The following table shows the result of linear regression of FPTS on all variables using the three different datasets. Implementation of linear regression is discussed in section 3.2, and the result is based on 5-fold cross validation. With the lowest RMSE of 9.5356, I proceed with data aggregated with quadratic weights for the rest of the paper.

	Square Root	Linear	Quadratic
MAE	7.2526	7.2124	7.1963
RMSE	9.5823	9.5437	9.5356

3.1.3 Feature Engineering

I include additional variables indicating home-court advantage (Home) and the number of rest days since the previous game (Rest) as in Barry, Canova, & Capiz (2017). Furthermore, I consider the number of available players on the roster based on the box score, both for a player's position (Rota_Pos) and for the whole team (Rota_All). The intuition is that if there are less players to substitute a player for, he will likely see higher usage in the game. The positions are based on the broader four categories specified above, which are also included as dummy variables.

To account for consistency, the standard deviation of FPTS over the past 10 games is included as FPTS_std, and salary information from DraftKings is also included as Salary. Lastly, a player on the starting unit is given a dummy variable of 1. Importantly, this assumes that whether a player will start the game or not is known prior to the game. Although this is reasonable as lineup changes are normally announced before the game by the team or sports analysts on social media such as Twitter, this requires further web mining to obtain this variable on a real-time basis.

3.1.4 Feature Selection

Reasonably, some features extracted from the box score are redundant because of correlations with at least one other variable. For instance, one can expect the field goal percentage (FG) to be highly related with effective shooting percentage (eFG%) as much less points are made from free throws that the latter takes into account. Additionally, some variables show multi-colinearity such as three-pointers made (3P), three-pointer attempts (3PA) and three point percentage (3P%). The base cases are omitted from the variables in these cases. Figure 6 is a correlation matrix of 40 features that are numerical. I reject the following 6 features based on the correlation threshold of 0.90; three-pointers attempted (3PA); defensive rebound (DRB); field goal attempted (FGA); field goal percentage (FG%); and offensive rating (ORtg).

Furthermore, I disregard variables with little predictive power. Figure 7 shows the feature importances of 34 variables based on gradient boosting regressor with default parameters. The following 5 features are omitted with coefficients of less than 0.01; dummy variables for positions (SG, F, C), three pointers made (3P), and triple doubles (TD). Ultimately, we use the remaining 29 variables as selected features for regression, gradient boosting and deep learning.

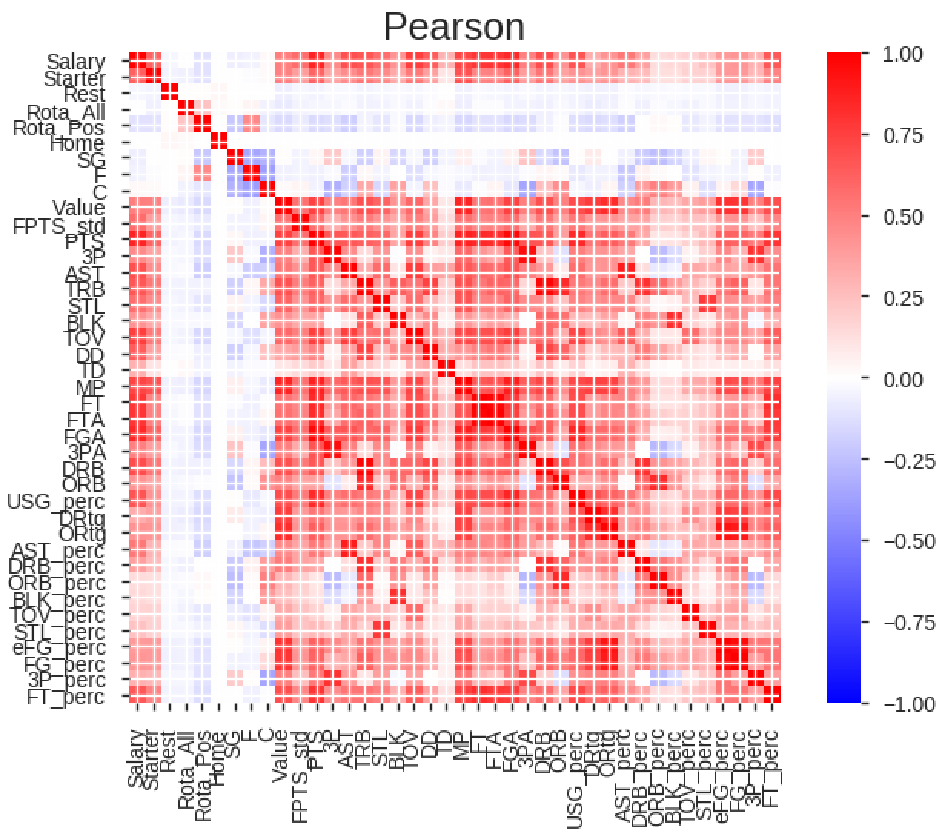


Figure 6: Correlation Matrix

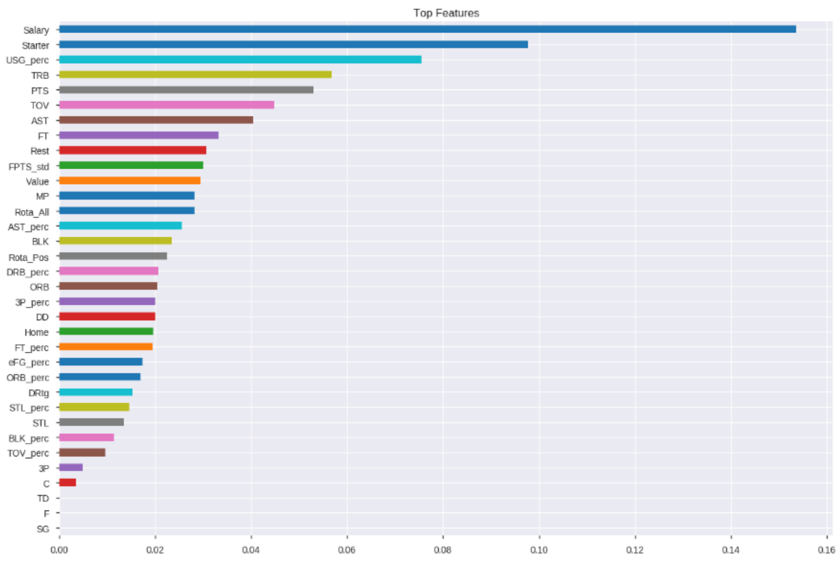


Figure 7: Top Features

3.2 Implementation

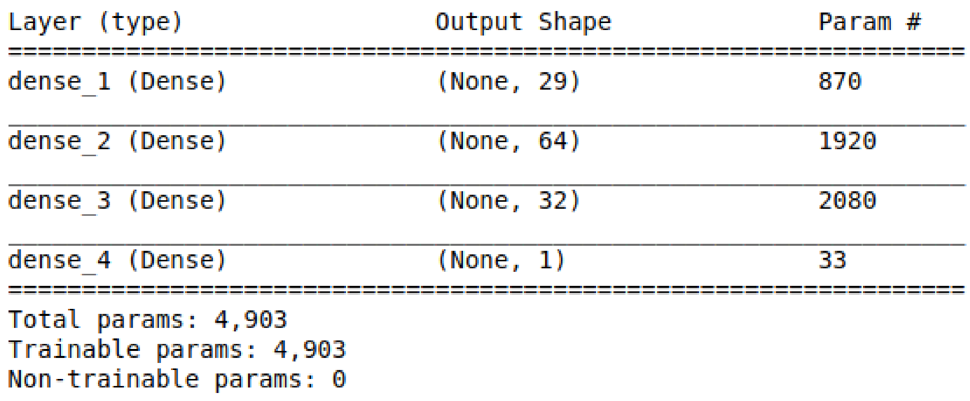
3.2.1 Environment

Throughout preprocessing, visualisation and model construction, I used `conda` distribution of Python 3.6.3 run on `jupyter notebook` in Ubuntu 16.04.4 LTS. In addition to modules included in Anaconda 4.3.34, I installed the following libraries; `beautifulsoup4` for static web scraping with `urllib`; `plotly` for interactive visualisations; `pandas-profiling` for data profiling and correlation matrices; `xgboost` for gradient boosting regressor; `sckit-optimize` for Bayesian optimisation of hyperparameters; and `keras` with `tensorflow` backend for deep learning.

3.2.2 Algorithms

I followed the `sklearn` specification of linear regression with default parameters. As variables were normalised into the range of $[0, 1]$ with `sklearn.preprocessing.MinMaxScaler` without centering, I left the parameters unchanged with `fit_intercept=True` and `normalize=False`. For gradient boosting, I used `scikit-learn` API for `xgboost` regressor with parameters tuned with Bayesian optimisation, the details of which described in the following section.

For deep learning, I used `scikit-learn` API for `keras` regressor with `tensorflow` backend. As `keras` does not provide RMSE as one of the default objective functions, mean squared error is used instead. Each layer uses ReLU for the activation function as it addresses the problem of vanishing gradient (Glorot, Bordes and Bengio, 2011). Adaptive Moment Estimation (Adam) is chosen as the optimiser which allows adaptive estimation of learning rates for each parameter (Ruder, 2016). The networks are trained for 30 epochs with the batch size of 32. Figure 8 shows a three-layer neural network used as the starting model.



```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 29)                870
_________________________________________________________________
dense_2 (Dense)              (None, 64)                1920
_________________________________________________________________
dense_3 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 33
=================================================================
Total params: 4,903
Trainable params: 4,903
Non-trainable params: 0
```

Figure 8: Initial Neural Network Architecture

3.3 Refinement

Refinement stems from parameter tuning of XGBoost and methods to prevent neural networks from overfitting. First, `XGBRegressor` without any parameter tuning using the 29 variables selected above yields an RMSE of 9.0316. I attempt to improve on this model with parameter tuning with Bayesian optimisation. The following table shows the parameters of `XGBRegressor` to optimise, in which 'Space' column shows the range of of each variable the the search was conducted in. Distributions of variables are set to be uniform except for learning rate with log-uniform distribution. Parameters are chosen randomly for the first 5 iterations, and are iteratively improved from the best performing set of parameters. In each iteration, 5-fold cross validation is used for evaluation of a given set of parameters. Figure 9 shows the convergence of the RMSE loss over 50 iterations. The best performing set of parameters are `max_depth:5, n_estimators:354, min_child_weight:0, subsample:0.7, colsample_bytree:1.0, colsample_bylevel:0.7, gamma:0.8, learning_rate:0.01526`, and tuning the regressor with these parameters leads to a better performance with 8.9581 RMSE.

Parameter	Space	Description
max_depth	[5, 50]	Maximum tree depth for base learners
n_estimators	[100, 500]	Number of boosted trees to fit
learning_rate	[0.001, 0.999]	Boosting learning rate
gamma	[0.0, 1.0]	Minimum loss reduction needed for further partition of leaf node
min_child_weight	[0, 5]	Minimum sum of instance weight(hessian) needed in a child
colsample_bytree	[0.7, 1.0]	Subsample ratio of columns when constructing each tree
colsample_bylevel	[0.7, 1.0]	Subsample ratio of columns for each split, in each level
subsample	[0.8, 1.0]	Subsample ratio of the training instance.

For deep learning, Figure 10 shows the learning process of the initial model over 30 epochs, with the number of epochs on the x-axis and model loss of mean squared error on the y-axis. 20% of the training data is reserved as validation data. It is clear that the model quickly starts to overfit, and the validation loss diverges from training loss. To mitigate this issue, a dropout layer is added which disregards 40% of data points randomly before they are fed forward to the final layer. I also use `EarlyStopping` method from `keras.callbacks`, which terminates the learning if validation loss does not decrease for more than 5 epochs. This marginally improved the generality of the model with RMSE decreasing from the initial model's 9.0678 to 9.0387.

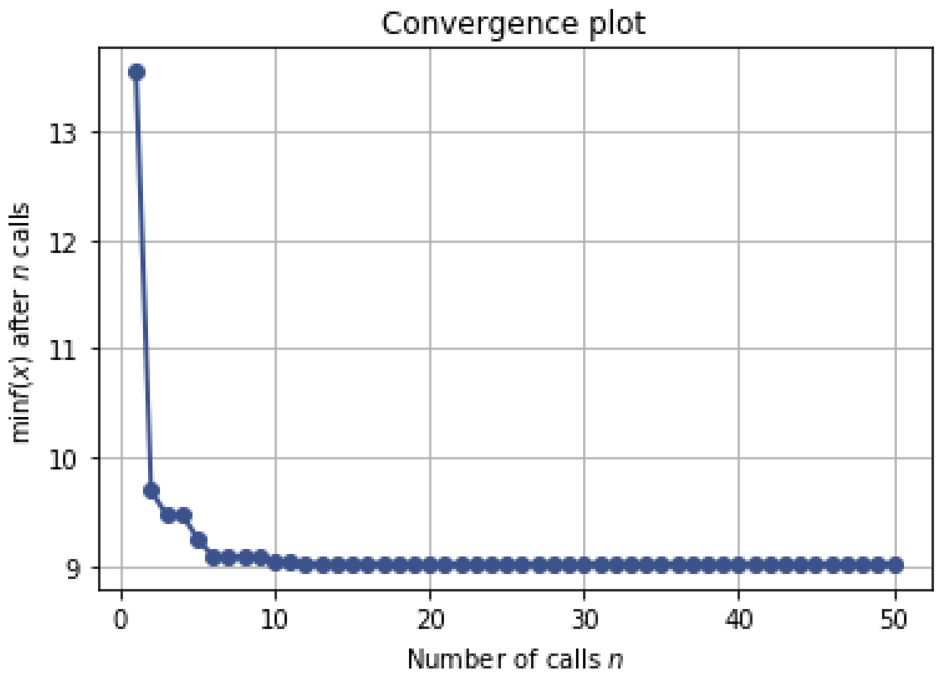


Figure 9: Convergence of RMSE loss with Baysesian Optimisation

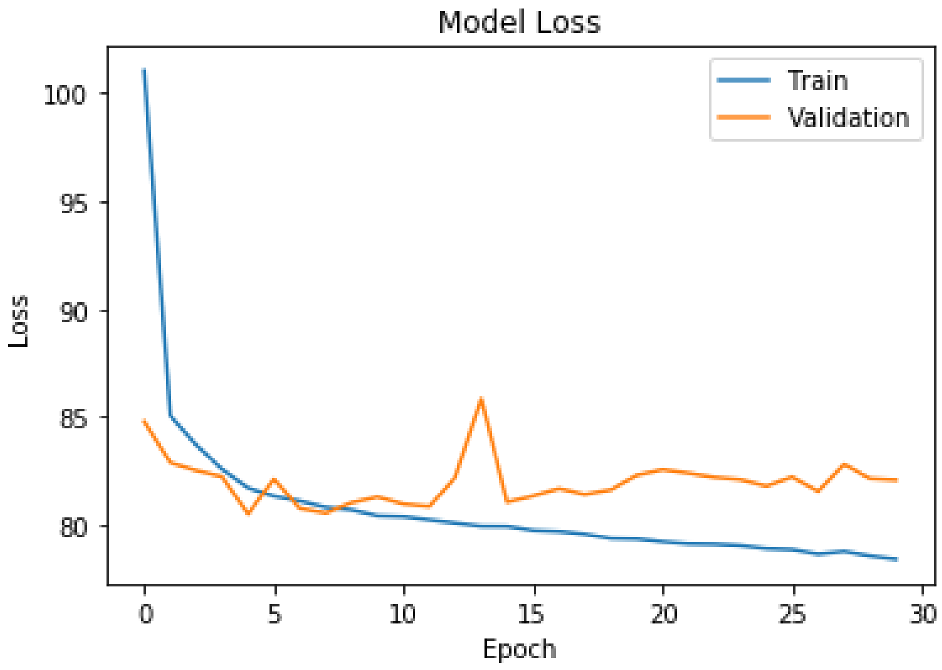


Figure 10: Learning Process of the Initial Model

4 Results

4.1 Model Evaluation and Validation

The following table compares the performances of different models, where models for the first and third rows linearly combine the average statistics using the coefficients defined by DraftKings. The last four rows with XGBoost and Neural Network use the selected 29 variables with quadratic weights. The results are generally in line with the expectation that gradient boosting algorithms would likely have the best performance, although deep learning performed better than expected with the use of a dropout layer and early stopping. With the lowest RMSE and MAE of 8.9581 and 6.8486, XGBoost regressor with hyperparameter tuning is selected as the final model.

Model	RMSE	MAE	Description
Simple Average	9.9434	7.4285	Baseline result with simple average
Linear Regression	9.8578	7.5197	Regression with the main 9 variables with simple average
Weighted Average	9.7475	7.2059	Weighted average of past 10 games with quadratic weights
Linear Regression	9.5905	7.2577	Regression with the main 9 variables with weighted statistics
Linear Regression	9.2558	7.0478	Regression with selected 29 variables with weighted statistics
XGBoost	9.0316	6.9052	XGBoost regression with 29 variables without parameter tuning
XGBoost	8.9581	6.8486	XGBoost regression with 29 variables with parameter tuning
Neural Network	9.0678	6.9074	3-layer NN with 29 variables trained for 30 epochs
Neural Network	9.0387	6.8805	3-layer NN with 29 variables with dropout and early stopping

It is reasonable to conclude that the results shown in the table above generalise well to unseen data as they are based on 5-fold cross validation with fairly consistent RMSEs of 8.9910, 9.0522, 9.0148, 8.9351, 9.0831. To bolster further confidence in the robustness of the model, I examine how the model's performance is affected by slight changes in the input data. First, Gaussian noise is generated with mean 0 and variance of 1, which is then scaled into the range of $[0, 0.2]$ and added to continuous variables in the original input. Performing 5-fold cross validation with the independent variables imputed with noise, loss quantities see little changes as RMSE and MAE increased only marginally to 6.8964 and 9.0153. This robustness demonstrates that the the model captures fundamental patterns inherent in the dataset.

4.2 Justification

The final model exhibits a sizeable improvement of roughly 10% from the baseline model of 9.9434 to 8.9581 in terms of RMSE. This section performs t-test to demonstrate that the final model with XGBoost outperforms the baseline RMSE of 9.9434 with simple averaging. The null hypothesis is set up such that the performance of XGBoost predictor is equivalent to that of the baseline model of sample averaging. It is sensible to assume that two populations follow the normal distribution as the central limit theorem can be applied to quantities involving average values, which RMSE does by definition. Standard deviation σ and sample size n are calculated from the results of 5-fold cross validation, where the raw results are 8.9910, 9.0522, 9.0148, 8.9351, 9.0831. Let $\bar{L}$ be the loss of the XGBoost predictor and L_0 be the loss of the baseline model in terms of RMSE, then the t-statistic is given as

$$t = \frac{\bar{L} - L_0}{\frac{\sigma}{\sqrt{n}}} = \frac{9.9434 - 8.9581}{\frac{0.0405}{\sqrt{5}}} = 121.51 > 15.54$$

which is statistically significant at 0.1% level with 4 degrees of freedom and a critical value of 15.54 (van Belle et al., 2004). It is therefore concluded that the the use of weighted average and feature engineering as well as XGBoost regressor have successfully reduced the error in predicting NBA player performance.

5 Conclusion

5.1 Prediction Accuracy

This section visually discusses the performance of the final model in the context of selecting players when one selects a lineup of players on DraftKings. For this process, I implemented Genetic Algorithms (GA) inspired by Musser (n.d.) to select an optimal lineup of 8 players given a set of players and performance predictions. Although actual implementation of GA and parameter tuning is omitted as it is beyond the scope of this paper where the primary objective is to minimise prediction errors, the essential idea of GA is briefly summarised as follows.

First, a population of candidates are generated based on a set of properties (genes). Second, two of the 'fittest' individuals called parents are 'bred' by replacing one gene with the other, creating 2×N candidates called 'children' where N is the number of genes in a candidate. Third, an additional set of 'mutated' candidates are added to the children pool where a gene in a child is replaced by a random gene not present in the parents' genes. This process is repeated by initialising the mutated children pool as the starting pool of candidates.

Translating this concept into fantasy basketball terms, 'fitness' can be considered as the total predicted FPTS of a lineup and a 'gene' as a player for each of the 8 positions (Point Guard, Shooting Guard, Small Forward, Power Forward, Center, Guard, Forward, and Utility). Note that lineups that exceed the salary cap of $50,000 are given a fitness score of 0. After several iterations, the lineup with the highest expected FPTS is returned as the optimal lineup.

For instance, running this algorithm on 5 games played on the 10th of March, 2018 yields a lineup of the following 8 players with expected FPTS of 242.2643 shown in Figure 11 (interactive version is available here), with a total salary of 49,900. Bars in blue, orange and green respectively correspond to actual FPTS, the final model's predictions and the baseline predictions. Underneath the players' names are their positions. The final model predicted the actual FPTS much better than the baseline in for certain players such as Dillon Brooks (SF) and Dwight Powell (C). However, it often overestimated FPTS for players such as Tomas Satoransky (G) and Kobi Simmons (F). Overall, these particular predictions from games on this date appear quite accurate, with losses of 6.2836 (MAE) and 7.6538 (RMSE).

However, there are some cases where the actual FPTS deviates substantially from predictions. Figure 12 shows the optimal lineup based on 5 games played on the 26th March, 2018 (see an interactive version here). It is immediately noticeable that New York Knicks' shooting guard Trey Burke's actual FPTS far exceeds the final model's prediction. In fact, he recorded a career-high 42 points and 12 assists against Charlotte Hornets in this game, which Kalbrosky (2018) reported as 'electrifying performance'. This particular case alone constitutes almost one-third (4.9953) of the total RMSE for this lineup (15.7466). While this is a positive surprise from a fantasy owner's point of view, this type of deviation demonstrates inherent difficulty in predicting player performance and competing in fantasy basketball.

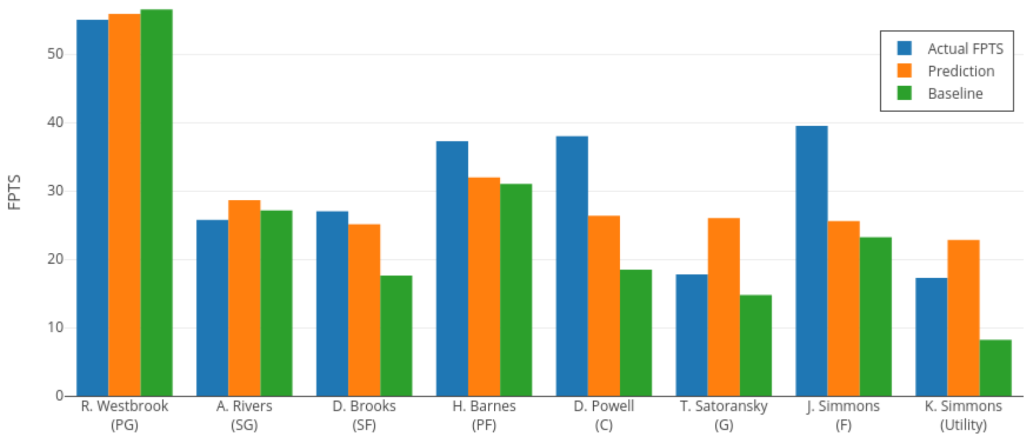


Figure 11: Optimal Lineup on the 10th March. 2018

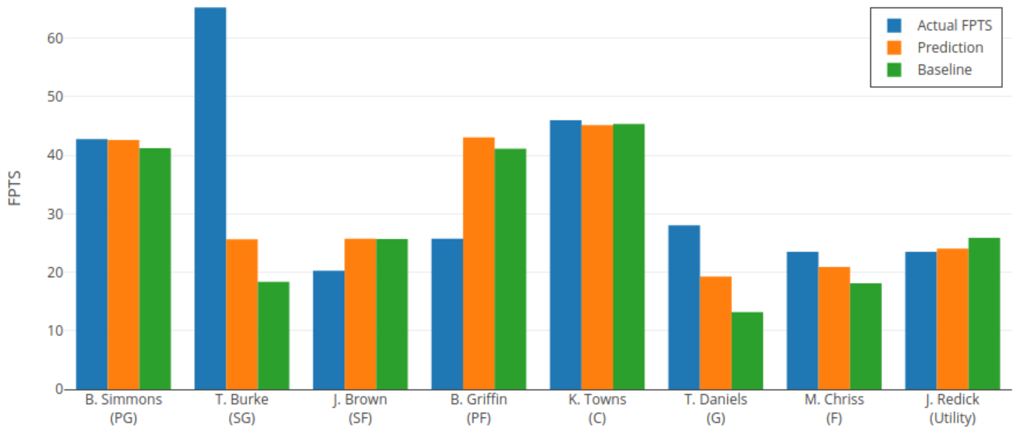


Figure 12: Optimal Lineup on the 26th March. 2018

5.2 Reflection

This project has attempted to predict NBA players' performance based on metrics provided by a leading fantasy sports website, DraftKings. The primary objective was to minimise the RMSE between prediction and true FPTS. Starting from the baseline prediction with the simple average of past statistics in the season, several feature engineering methods were implemented with different weighting scheme. Key variables in predicting player performances were salary information provided on DraftKings, whether a player is on the starting unit or not, usage percent that represents a player's presence on the court, and key statistics such as total rebounds and points. After feature selection and data normalisation, XGBoost with hyperparameter tuning showed the lowest RMSE of 8.9581 in comparison to 9.2558 with linear regression and 9.0387 with neural networks, a sizeable improvement from the baseline of 9.9434.

Over the whole data analysis process, feature importances and the correlation matrix shown in Figure 6 were particularly informative in understanding how statistical indicators capture different components of player performance. In terms of difficulties, data preprocessing, especially name standardisation, and feature engineering were the most consuming part of the project, given the absence of tangible results and uncertainty about the usefulness of the features.

5.3 Improvement

While the models implemented in this project generally followed expectations in terms of performance, the final improvement of roughly less than 10% in RMSE was somewhat underwhelming. Potential improvement could come from considering the opponents' defensive statistics and past match-ups such as defensive ratings of the team and players in a certain position. Another important aspect is coaching which determines the minutes players are given, and modeling player usage could give insight into how minutes vary under different game management especially for rookies and veterans. Additionally, DraftKings provides Daily Research articles recommending which players to pick. Incorporating this information using natural language processing would likely be beneficial both in terms of performance prediction and lineup optimisation. Finally, time series methods such as ARIMA or VAR models would also be potential alternatives to variable aggregatation methods employed in this project.

References

[1] Christopher Barry, Nicholas Canova, and Kevin Capiz. Beating Draftkings at Daily Fantasy Sports. 2017. [Online; Accessed 04-May-2018].

[2] Tianqi Chen. Introduction to Boosted Trees, n.d. [Online; Accessed 15-May-2018].

[3] DraftKings. NBA: Rules & Scoring, 2018. [Online; Accessed 11-May-2018].

[4] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. 2011. [Online; Accessed 11-May-2018].

[5] B. Kalbrosky. Knicks' Trey Burke has excelled with his midrange jumper and much more, 2018. [Online; Accessed 21-May-2018].

[6] Adam Levitan. 5 Secrets of Daily Fantasy Basketball, 2016. [Online; Accessed 13-May-2018].

[7] Harry Markowitz. Portfolio Selection. *The Journal of Finance*, 7(1), 1952. [Online; Accessed 17-May-2018].

[8] Ryan Musser. Show and Tell: I Published a Compounding Genetic Algorithm to Hack FanDuel NBA Lineup Optimization, n.d. [Online; Accessed 19-May-2018].

[9] D Oliver. *Basketball on Paper: Rules and Tools for Performance Analysis.* Potomac Books, 2004.

[10] V. Pham. Bayesian Optimization for Hyperparameter Tuning, 2016. [Online; Accessed 15-May-2018].

[11] S. Ruder. An overview of gradient descent optimization algorithms, 2016. [Online; Accessed 15-May-2018].

[12] R.E. Shapiro. Explaining Adaboost. 2013. [Online; Accessed 12-May-2018].

[13] USAToday. Harden has first 60-point triple-double in NBA history. 2018. [Online; Accessed 15-May-2018].

[14] G. van Belle, L. Fisher, P. Heagerty, and T. Lumley. *Biostatistics: A Methodology for the Health Sciences.* 2004. [Online; Accessed 20-May-2018].