DATA608 Final Report

Justin Hink

Sunday, April 16, 2017

Abstract

The project explores predicting player level outcomes in professional basketball, specifically the National Basketball Association (NBA). Two distincly different models classes are explored in isolation for this task: Regression to the Mean based systems and Neural Network based systems. Finally, these two broad classes of prediction algorithms are combined into a composite model.

In total, four models are constructed, each with numerous tweakable parameters. Peformance is evaluated on a per model, per parameter basis and a recommendation is made as to which model should yield best predictive performance going forward.

Win Shares Per 48 Minutes is used as the target metric for the modeling effort thought the project. The technical framework built should allow similar work to be done against any other rate or counting statistic.

Key Words and Concepts

True Talent vs. Luck Each player in the NBA has an underlying, ever-changing talent level. This talent level is quite often different than statisically measured results (ex. FG%, 3 Point FG%, Defensive Rebound Rate, Win Shares Per Plying time) will present. A player's measured results and true talent are governed by the following high-level and simplified equation:

$$Outcomes_{Team} = Talent_{Team} + Luck$$

Regression To The Mean Also commonly referred to as "Regression Toward the Mean" or simply "Mean Reversion", is a general concept that extreme observataion tend to fall closer to the population mean on subsequent measurements. The concept is a pillar of in sports analytcs and most first generation projection systems incorporate the idea into their algorithms.

Projection System An algorithm, or set of algorithms, that attempt to predict future outcomes. These systems usually involve predicting player level statistics at various time scales (ie - season length projections, career length projections, game based projections). Examples:

1) Projection System A predicts Giannis Antetokounmpo to average 26.4 points next season
2) Projection System B predicts Mike Trout (baseball player) to finish his career with 116 Wins Above Replacement, making him a sure-fire Hall of Famer.

Marcel Originally developed for baseball but wholly relevant for other sports, Marcel is very simple but effective system developed and published by renowned baseball analyst, Tom Tango (an alias, his real name is not available in the public domain). The system gets it's name from a pet monkey named Marcel and the fact that the algorithm is so simple that a monkey should be able to develop and understand it. The mathematical form of the algorithm is as follows [cite blog post]:

$$\hat{p}_i = \frac{t^* x_i + \left(\frac{t^* n_0}{t^* n_1}\right) t^* D_{ni} p_0}{t^* n_i + t^* n_0}$$

While Tango published the algorithm, he does not want to take credit for the results it produces. Rather he prefers the algorithm to be used as a baseline for projection systems to use as a measuring stick. [CITATION]

Win Shares A statistical concept originally developed by Bill James in the 1980s in his Baseball Abstracts, Win Shares attmpt to assign credit/blame to individual player for their team's performance. [CITATION]

Neural Network A class of computation models that utilize a large collection of simple neural units working together in order to glean intelligence and insight for input data. Uses are broad but can include classification algorithms, regression algorithms and generalized prediction engines.

Deep Learning A class of machine learning algorithms that use many layers of nonlinear processing units for model feature extraction and transformation. Output from one layer of the network is fed into the next as input. High level model features are derived from lower level features and create an overall hierarchical representation. [CITATION wiki]

Introduction

For this MS in Data Analytics Capstone project, I built and evaluated a number of projection algorithms for player level events in the National Basketball Association (NBA hereafter). The project establishes a baseline level of efficacy for a projection system using a technique similar to that found in the baseball analytics world known as “Marcel the Monkey Forecasting System” [CITATION proposal] and then attempt to move past that simple system’s performance with more advanced techniques.

For these more advanced techniques, Google’s popular and fast growing TensorFlow [CITATION proposal] machine learning library was used. Having not had the opportunity to use the library for a real project to date and this was the perfect opportunity for me to gain experience with it.

The metric Win Shares per 48 minutes has been chosen as the metric of study. It is one of modern basketball analytics best attempts to encapsulate player value in a singular number. Effectively projecting this value for individual players in future seasons is the overarching goal of the project.

Methodology

Data Source

All of the data in this project was gathered from BasketballReference.com [CITATION]. Player data was gathered for the 1995/1996 through 2015/2016 seasons.

Data Preparation

The data for this project was gathered using custom html scraping code. The code took advantage of the BeautifulSoup and Pandas libraries for this task.

Once downloaded, the data was transformed to a familiar tabular format which would be useful to build all of the following projection models. A sample of this transformed data follows.

player	age	ws_3	mp_3	ws_2	mp_2	ws_1	mp_1	ws_target	mp_target
A.C. Green	34.0	0.12	2687.0	0.1	2113.0	0.093	2492.0	0.095	2649.0
Aaron McKie	25.0	0.115	827.0	0.113	2259.0	0.103	1625.0	0.036	1813.0
Adam Keefe	27.0	0.154	1270.0	0.13	1708.0	0.118	915.0	0.153	2047.0
Adrian Caldwell	31.0	0.02	30.0	0.044	327.0	0.016	569.0	-0.022	3.0
Allan Houston	26.0	0.091	1996.0	0.112	3072.0	0.084	2681.0	0.102	2848.0
Alonzo Mourning*	27.0	0.153	2941.0	0.178	2671.0	0.174	2320.0	0.185	1939.0
Alton Lister	39.0	0.062	776.0	0.06	735.0	0.028	516.0	0.021	44.0
Andrew Lang	31.0	0.1	2340.0	0.047	2365.0	0.076	1194.0	0.047	692.0
Anfernee Hardaway	26.0	0.177	2901.0	0.229	3015.0	0.175	2221.0	0.085	625.0

The “ws_” variables represent win share per 48 minute values. The “mp_” variables represent total minutes played for a player in a given year. The numeric postfix represents how many years away the data was gathered from the target projection year. For example, ws_3 and mp_3 represent win share and minutes played values from 3 years prior to the projection year. The age variable represents the age of the player for the target projection year. In the above sample, this means that A.C. Green was 34 years old in the season that he posted 0.095 win shares per 48 minutes of playing time.

Variable	Type	Predictor/Target
ws_3	Numeric, Continuous	Predictor
ws_2	Numeric, Continuous	Predictor
ws_1	Numeric, Continuous	Predictor
mp_3	Numeric, Continuous	Predictor
mp_2	Numeric, Continuous	Predictor
mp_1	Numeric, Continuous	Predictor
mp_target	Numeric, Continuous	Target
ws_target	Numeric, Continuous	Target

Table 1: Variable Summary

For all of the algorithms in the following section, the goal is to predict target win shares accurately based on past performance, playing time and age.

To ensure that all testing was of algorithm performance was done out of sample, the data was split into training and test sets. The training data was used to build the algorithms and tune their various settings while the test data set was left for final testing and evaluation. In total there were 3272 records in the training data set and 1403 records in the test data set, all in the format of the above sample.

The code used to extract, transform and load this data can be found in the GitHub repository listed in Appendix A.

Description of Algorithms

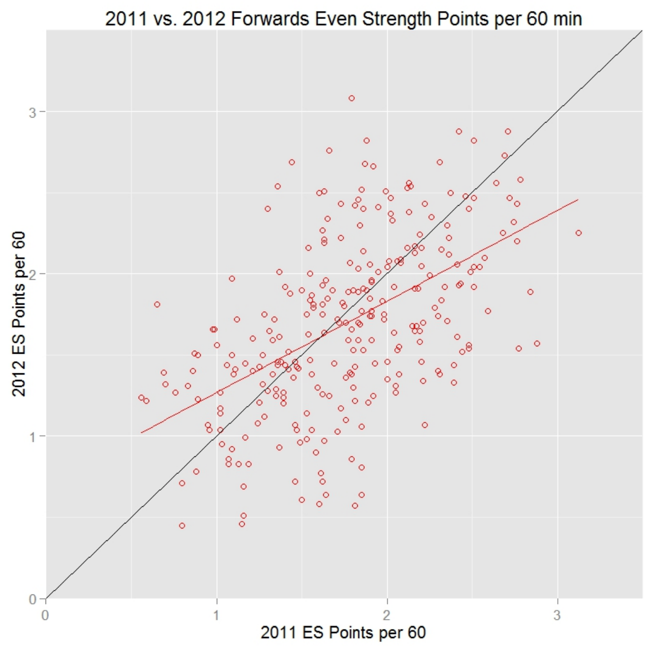
Algorithm 1: "Marcel"

The first algorithm implemented in this project is an adaptation of the Marcel forecasting system that was developed for baseball. Marcel is a very basic system that nonetheless performs well in tests against more advanced systems. The first algorithm in this project can be thought of as the first logical pass at Marcel when applied to basketball. For each of the Marcel-style projections there are four general steps, which are explained below.

Weighted average of past years' outcomes The first step is to calculate a weighted average of each player's past outcomes, with more weight given to the most recent data. Four years of data are used, and the relative weights applied to each year are determined by fitting against historical data out of sample.

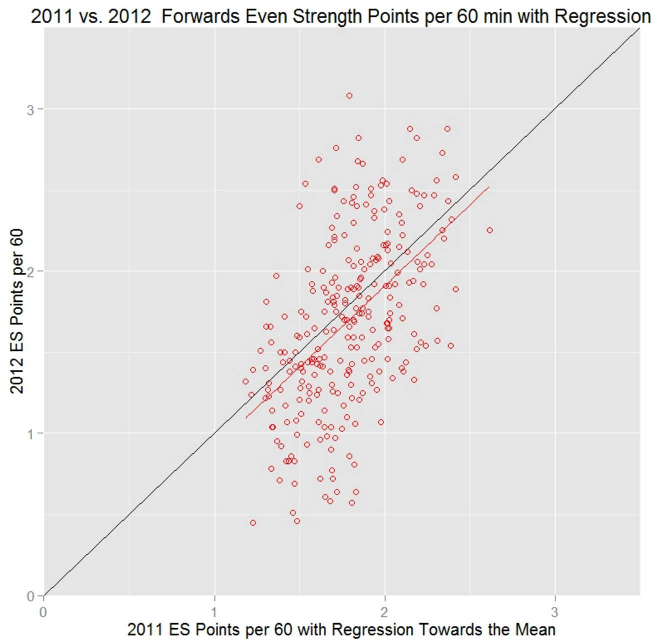
Regression towards the mean The second step is to apply regression towards the mean to account for the role of luck in past outcomes. Since a player's past outcomes are the sum of their talent and luck (a catch-all term that we're just using to mean all factors outside of the player's control), if a player's outcomes were above average we expect that he both had above average talent and above average luck, and vice-versa. Since only the talent part is expected to persist into the future (all players are expected to have average luck going forward), we should expect players whose past outcomes were above league average to on average have worse outcomes in the future (but still above league average), and we should expect players whose past outcomes were below league average to on average have better outcomes in the future (but still below league average).

The following graphs illustrate regression towards the mean in action. The graphs were generated using hockey data from a previous project of mine in this degree but the illustration of the core concept is wholly relevant to domains other than hockey. [CITATION] The first graph shows even strength points per 60 minutes for all forwards that played at least 500 even strength minutes in both the 2010-2011 and 2011-2012 seasons. The red line represents the best-fit for predicting a player's year 2 points per 60 given their year 1 points per 60. The black line is where year 2 points equals year 1 points.



Note that the red line has a shallower slope than the black line, and in particular that the red line falls below the black line when year 1 points per 60 is above average, and falls above the black line when year 1 points per 60 is below average. This shows that players who were above average in 2010-2011 tended to be less above average in 2011-2012, and those who were below average in 2010-2011 tended to be less below average in 2011-2012. This is exactly the expected pattern.

Now, let's see what happens when regression towards the mean is applied to the 2010-2011 statistics by adding 730 minutes of league average outcomes to each player's value:



Now the red line matches the black line (at least, the slope is the same - the reason it's slightly below the black line is because overall scoring for these players went down slightly from 2010-2011 to 2011-2012). Note that the year 1 statistics are now compressed into a smaller range than in the previous graph. That's what happens when regression towards the mean is applied - to account for the role of luck in past outcomes, the measured results get pulled back towards the mean, with the most extreme values being pulled back the most.

Aging adjustment The last step of the core projection engine is to apply an aging adjustment to account for the fact that young players tend to improve and old players tend to get worse. Advanced methods here are worthy of their own capstone project. Marcel, being the simplest possible system, elects to use a rudimentary heuristic here that ends up working quite well. The heuristic is simple, if a player is younger than 28 years of age, boost their age-agnostic projection by 0.4% times the number of years younger than 28. For example, if a player is 22 years old and their age-agnostic projection came out to be 0.150 WS/48 mins, their age adjusted projection would be calculated as:

$$Proj_{age-adj} = (0.15) * (1 \quad + \quad (28 - 22) * 0.004) \quad = \quad 0.1536$$

Conversely, if the player is 28 years of age or older, penalize their age-agnostic projection by 0.2% per year older than 28.

While the core Marcel algorithm is not new to this project, the implementation is. The core bit of python code used to drive this type of projection can be viewed below:

```
def weight_rttm_age(self, row):
    mp_sum = 6 * row['mp_3'] + 3 * row['mp_2'] + 1 * row['mp_3']

    w1 = self.weights[0]
    w2 = self.weights[1]
    w3 = self.weights[2]
    regk = 1000.0 # 1000 minutes

    # calculate a weighted average
```

```python
        temp = ((row['ws_1'] * row['mp_1'] * w1 +  row['ws_2'] * row['mp_2'] * w2
                  + row['ws_3'] * row['mp_3'] * w3)
                / (row['mp_1'] * w1 + row['mp_2'] * w2 + row['mp_3'] * w3))

        # apply regression towards the mean
        # akin to naive bayes with a weakly informative prior
        retval =  ((temp * (row['mp_1'] * w1 + row['mp_2'] * w2 + row['mp_3'] * w3)
                      + regk * self.AVG)
                    / (regk + (row['mp_1'] * w1 + row['mp_2'] * w2 + row['mp_3'] * w3)))

        # simple aging adjustment
        if row['age'] < 28:
            retval = retval * (1 + (28 - row['age']) * 0.004)
        elif row['age'] > 28:
            retval = retval * (1 + (28 - row['age']) * 0.002)

        return retval

    def project_players(self, test_data):
        test_data['proj_marcel'] = test_data.apply(self.weight_rttm_age, axis=1)
        return test_data
```

One important note here is that the defaults for the weighting of past year's performance are as follows (where N is the year the projection is to be built for):

Year	Weight
N-1	6.0
N-2	3.0
N-3	1.0

With a pure Marcel approach, these weights are essentially chosen from intuition alone. It makes sense that performance from one year ago should be weighted more heavily than performance from 2 years ago when attempting to determine current talent levels/future performance. The weights in the above table are chosen as such.

Algorithm 2: "Trained Marcel"

The second algorithm developed in this project is a small extension to the core Marcel framework. Instead of using semi-random/intuition-driven coefficients for the weighting portion of the algorithm, an initial first pass over the training data set is performed to derive these weights. The first pass is a simple ordinary least square (OLS) regression and outputs recommended coefficients for each of the previous year's past performance. These coefficients are then used in the subsequent core Marcel pass over the data instead of the weights in the table from the previous section.

The python snippet that implements this simple modification looks as follows:

```python
    # Note, the following is in the constructor of the main projection class
    if not use_default_weights:
        model = ols("ws_target ~ ws_1 + ws_2 + ws_3 + 1", train_data).fit()
        self.weights[0] = model._results.params[1] * 1
```

```
        self.weights[1] = model._results.params[2] * 1
        self.weights[2] = model._results.params[3] * 1
```

Algorithm 3: "Wide and Deep Learning"

The third approach taken in this project is a significant departure in methodology from the methods described in the two previous sections. A neural network approach to projecting future player level outcomes was developed. The general structure of a neural network in the context of performing regression like behavior looks as follows: [CITATION]

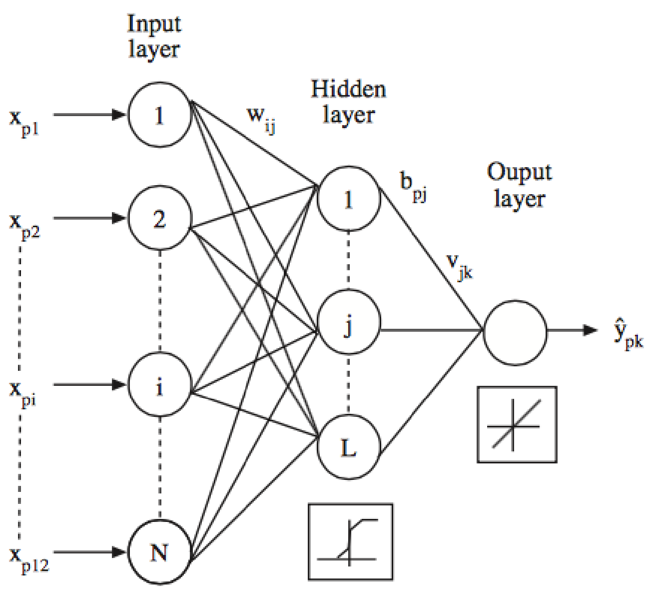


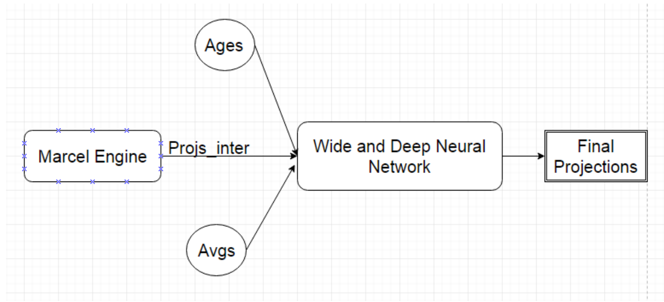
Figure 3A: Multilayerperception

High level model features are derived from lower level features and create an overall hierarchical model that can be used to predict target values. In the context of this project, the features fed in to the lower levels of the model are things like win shares and playing time (in minutes played) for past seasons as well as the player's age. The output is the prediction of win share performance for the target seasson.

In order to take on this complicated task, Google's TensorFlow library was used extensively. Tensorflow provides high level APIs that make training and using this class of model much simpler than it would be if the core training logic needed to be developed from scratch. The models built with Tensorflow provide a number of tunable knobs and dials. These include things such as model learning rate, number of hidden layers included in the model and the optimization algorithm used as part of both the deep and wide learning portions of the model's training. Throughout this portion of the project, a number of these tunable values were adjuted and their effect on model performance was examined.

Algorithm 4: "Marcel Net"

The final model that was developed for this project is a hybrid of the Marcel and Neural Network approaches described above. In this algorithm, a set of Marcel projections are generated. These intermediate projections along with player ages and league averages are then input to a wide and deep neural network for further training.



The entire code listing for generating all of the above models can be found in the GitHub repository listed in the appendices.

Data Source

Technology Used

The code for the project was developed entirely in Python. The well established libraries for data wrangling (Pandas, numpy, TensorFlow), building neural networks and analyzing results made this a logical choice for coding up this sort of prototype.

There was no need for persistent storage other than saving simulation results. Simple CSV files were used for this purpose.

All training runs of the models in this project were built on a machine with the following specifications. Note, GPU information is not included as TensorFlow's GPU extensions were not utilized for this project. Re-running the models on a box with these extensions enabled will be left for future work.

Spec	Value
CPU Cores	4 physical, 8 logical
CPU Clock	2.8 GHz
CPU Model	Intel Core i7-4900MQ
RAM	24 GB
OS (Host)	Windows 8.1
OS (VM)	Ubuntu 16.04.1 LTS
Hypervisor	VMWare Workstation

Results

TODO

Marcel

TODO

Eval Metric	Value
R	0.7242
RMSE	0.0370
MAE	0.0291

Trained Marcel

TODO

Eval Metric	Value
R	0.7236
RMSE	0.0375
MAE	0.0296

Pure Neural Network

TODO

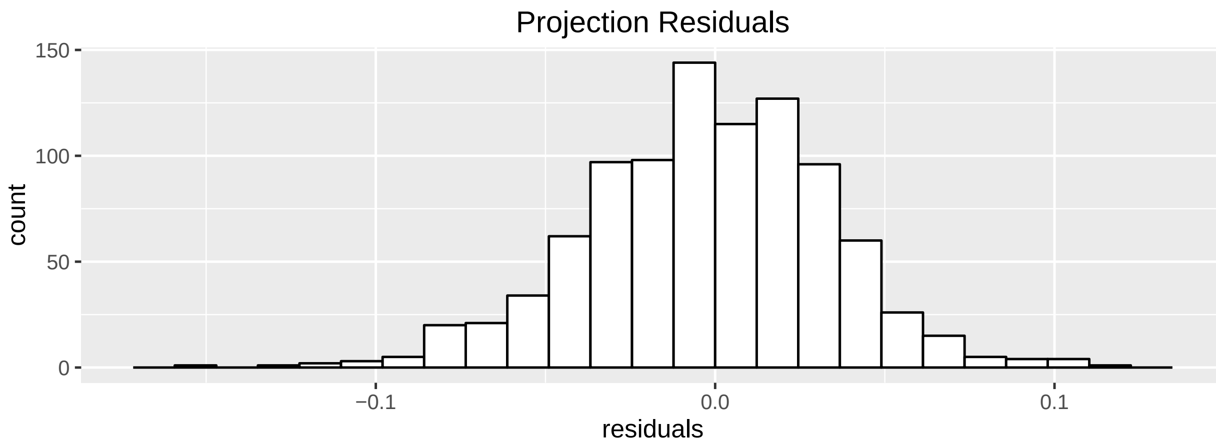
Run	Use.League.Avg	Optimizer	Hidden.Units	Learning.Rate	R	RMSE	MAE	TrainTime
1	No	Ftrl	1000:500:100	0.0100	0.6705	0.0401	0.0308	3602
2	No	RMSProp	1000:500:100	0.0100	0.6692	0.0389	0.0298	3711
3	No	RMSProp	1000:500:100	0.0010	0.6692	0.0389	0.0298	8832
4	No	RMSProp	1000:500:100:50	0.0010	0.6611	0.0461	0.0357	12432
5	Yes	RMSProp	1000:500:100	0.0100	0.6877	0.0398	0.0308	4123
6	Yes	ProxAda	1000:500:100	0.0100	0.6745	0.0389	0.0301	5522
7	Yes	ProxAda	1000:500:100	0.0010	0.6556	0.0487	0.0381	9973
8	Yes	Adam	1000:500:100	0.0100	0.6886	0.0381	0.0291	7824
9	Yes	Adam	125:64:12	0.0045	0.6906	0.0381	0.0289	233
10	Yes	Adam	32:16:03:1	0.0010	0.6906	0.0381	0.0288	41

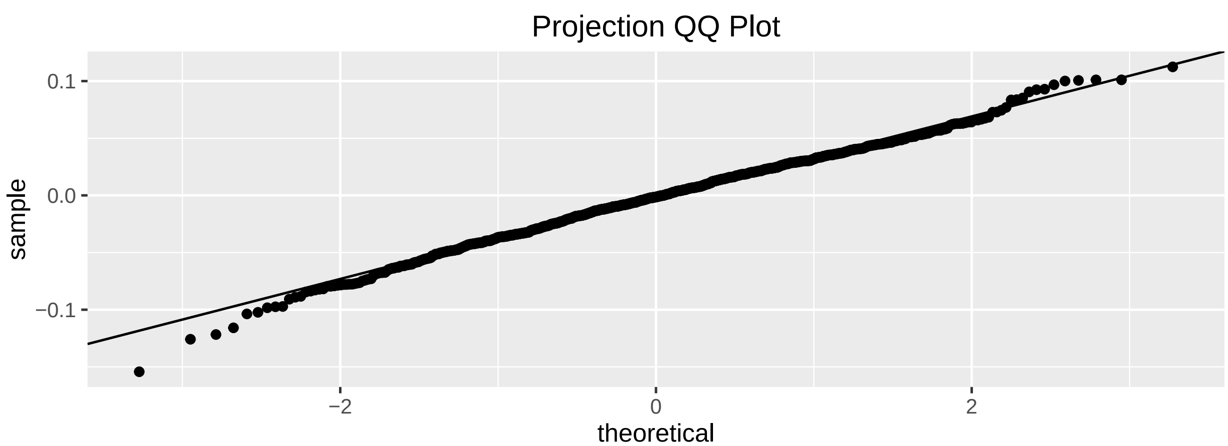
Marcel Net

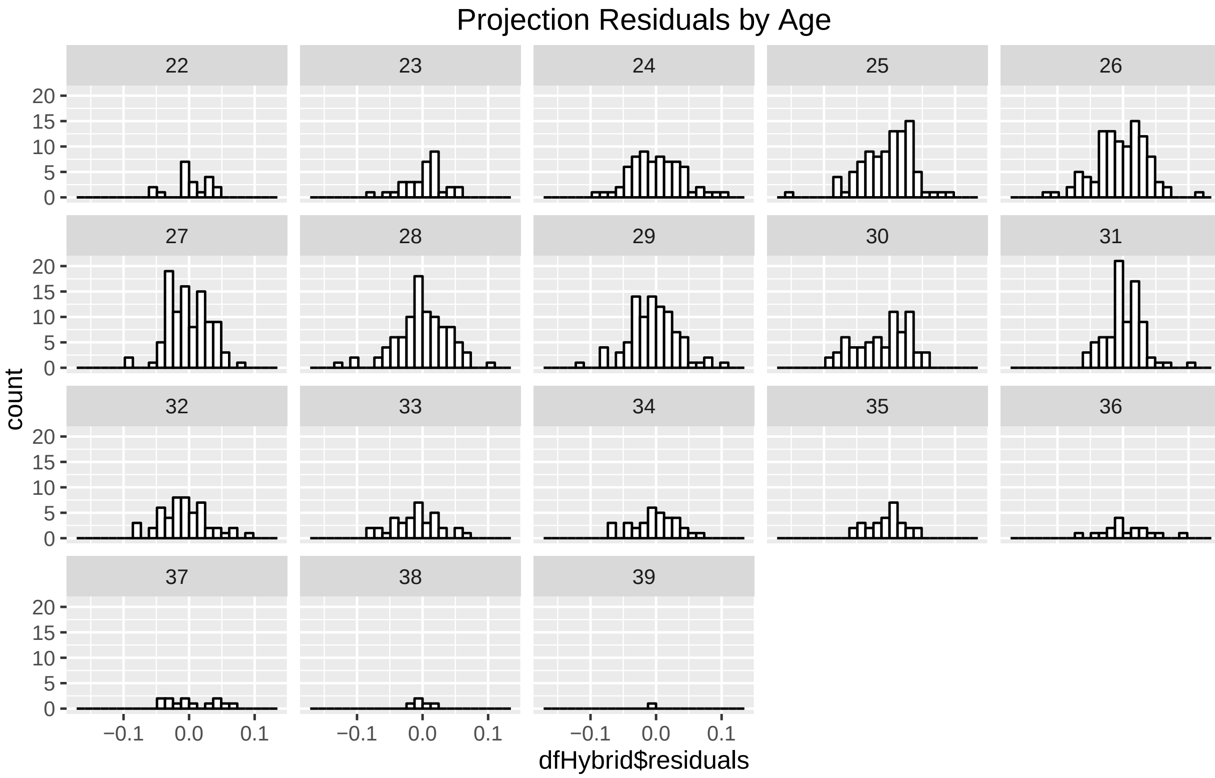
TODO

Run	Hidden.Units	Learning.Rate	R	RMSE	MAE
1	125:64:12	0.0045	0.7271	0.03610	0.0282
2	125:64:12	0.0010	0.7286	0.03595	0.0281
3	125:64:12	0.0005	0.7267	0.03598	0.0283
4	250:128:24	0.0010	0.7267	0.03610	0.0283

Run	Hidden.Units	Learning.Rate	R	RMSE	MAE
5	64:32:06:01	0.0010	0.7257	0.03630	0.0285
6	128:64:12:10:6:3:1	0.0010	0.7270	0.03600	0.2830







Future Work

The methods developed throughout this project should be general enough such that holistic measures of value other than Win Shares can be projected. For example, popular metrics such as Value Over Replacement Player (VORP) and Wins Above Replacement (WAR) should be projectable to a reasonable degree with both a pure Marcel approach and a Marcel/Deep Learning hybrid algorithm. Examining how effective the various methods are at projecting each metric would be an interesting project.

A deeper dive into creating a truly useful projection system for NBA players would involving projecting each individual component statistic that goes into the win share calculation. The first step down this path would be to project defensive (DWS) and offensive (OWS) win shares separately and then adding these together to create the overall Win Share projection. Logically, this can be extended further by first calculating individual components of DWS and OWS and then combining these projections into intermediate DWS and OWS which can then be added to give a final, more accurate (theoretically) overall Win Shares projection.

While this remains an untested hypothesis for basketball projections, this method of componentized projection has worked well for me in the past when building projection systems for other sports. For example, in hockey, projecting an individual's goal scoring rate (in goals per game or goals per minute) is easy enough to do directly. Using a Marcel based system, taking a weighted average of the target player's last 3 seasons goal rates and regressing that to the mean an appropriate amount will yield a decent amount of predictive power. However, large gains are made as soon as the goal rate projection is broken up into multiple sub projections. For goals, predicting a player's shot rate (the number of shots they take on goal per game or minute) and predicting their shooting percentage (rate of shots that become goals) separately and then combining together for an overall goal rate projection yields a substantial improvement. Breaking each projection down based on basic game context (Even Strength, Powerplay, Shorthanded) improves the method even further and does not meander into the realm of overfitting.

My professional work in baseball has shown similar results to the above hockey example. Projecting a measure like weighted on base average (wOBA) is simple enough and will provide a decent estimate of a player's overall batting skill. It is easily improved upon however, as soon as you start projecting component parts such as homerun per fly ball rate, ground ball rate, strikeout rate and walk rate separately.

Conclusion

This project set out to examine the efficacy of multiple algorithms for projecting future NBA player level performance. A total of 4 algorithms were examined with three of them heavily borrowing logic from the well established Marcel the Monkey projection system which originated in the baseball analytics community.

The most advanced system combined a pure Marcel approach (weighted time average plus regression to the mean) with a wide and deep neural network. This hybrid approach achieved superior performance compared to systems using Marcel or neural networks alone. The one significant downside of any of the algorithms leveraging a neural network was the training time and computing power needed when compared to a pure Marcel approach. The neural network based algorithms required orders of magnitude greater compute time to complete.

Ultimately, the choice of algorithm would depend on use case. If projection accuracy was the motivating decision variable, the hybrid system developed here provides that. But if the algorithm needs to be run and updated multiple times per minute, the amount of computing power required may become prohibitive (depending on budget of course). The Marcel system runs very quickly, has minimal code and library dependencies and is in the same realm accuracy wise as the more complicated systems. It may very well be the correct choice in some situations.

References

Appendix A: Code Repository

https://github.com/jhink7/nba_flow