Machine Learning Engineer Nanodegree

Capstone Project

Matthew Ignal

February 5th, 2017 (Last Update: 2/16)

I. Definition

Project Overview

Every year, in mid-June, the lengthy process of scouting, interviewing, and getting to know young basketball players culminates in the NBA Draft, whereupon these athletes are selected by NBA franchises with the hopes they can develop into rotation players and potentially All-Stars. With each draft, there are players selected who will wildly out-perform their draft position while others will fail to live up to the lofty expectations that come with being selected with an early lottery pick. Nabbing an elite player can change the course of these billion-dollar franchises for 10 or 20 years, while failing to capitalize on a rare high-pick opportunity can lead to a decade of mediocrity.

With such high stakes, all 30 teams invest heavily in their scouting and departments to help inform their brain-trust to try and make the best decision at the time. They even work closely with psychologists to try and determine how players will develop! Nevertheless, the history of the NBA Draft, even in the modern era, suggests that this science is far from perfect. High-profile “busts” have been selected with the top pick (recently, Anthony Bennett), while the last pick of the draft has produced an multi-time All-Star (5-foot-9 Isaiah Thomas).

To assist with this process, various player projection algorithms have cropped up recently. Several exist publicly (catalogued at tothemean.com), although I am sure many more are only available internally to teams. Basketball is a passion of mine and I’m particularly interested in prospect projections, so I’d like to throw my hat into the ring while providing a few new wrinkles.

The working title of my algorithm, BEEM, stands for Bagging Error Estimate Using Meta-Features. I will attempt to explain the algorithm in more detail in the following sections.

Problem Statement

The problem to be solved is to create prospect projections so there can be a reasonable estimate of a player's future performance. I have incorporated ordinary box-score stats, player measurables, combine statistics, as well as play-by-play data to assist with the process of creating a regression model to predict future Win Shares.

My data includes standardized historical stats dating back to 2003, along with physical and athletic testing as well as play-by-play data where available. The sparsity of some of this data is a necessary obstacle to overcome given that nearly all prior projection models have already tried to make use of ordinary box-score stats.

It was possibly the biggest challenge of this project to handle the inconsistent availability of the data, but my solution was to create meta-features out the existing data to provide estimates of a player's skill level in a particular area regardless of the data. I also performed linear regressions to fill in things like athletic testing.

As far as working towards a solution, several algorithms are reasonable to put to use. Linear regressions are good where the relationship between features are mostly additive, but I suspect the relationship between the final features will be more complex than in the case of score creation, so decision tree methods will likely work best. While decision trees are prone to overfitting, their ensemble-method cousins in random forests and extremely randomized trees are resistant as they pick a random subset of features from which to determine a tree split (extreme randomization means that the split is chosen at random). Using a Bagging estimator may be promising as well, where the individual predictions are aggregated, which might improve the performance of the base model.

A nearest neighbors algorithm also might be able to handle the complex relationship between features. By finding the most similar player-seasons, nearest neighbors assumes that players who have similar features will have similar careers. This seems like a reasonable assumption. Because some of the features for a player will be identical year-to-year (height and agility, for example), using a grouped cross-validation technique will be necessary to reduce overfitting.

BEEM winds up generating a linear regression using a combination of box-score data and meta-features, and then adjusts the estimates based on error analysis by using these meta-features.

Solution Statement

The solution to prospect projections is to come up with a machine learning algorithm that puts all this data to use! This will produce estimates of future WS performance and provide rankings

for NBA drafts based on pre-NBA data for every year. If projections are accurate, then the model can be applied to the upcoming NBA draft.

Metrics

According to Basketball Reference, Win Shares are "a player statistic which attempts to divvy up credit for team success to the individuals on the team." The details by which Win Shares are calculated are beyond the scope of this project, but it is important to keep in mind that Win Shares are merely one of many ways to evaluate a basketball player and that there are other equally valid metrics. In a given season, an All-Star level talent can have a WS above 10.0, and it's possible for some players to be below 0.0.

Evaluation Metric #1: For evaluating projections of individual player-seasons as well as value over a period of years, I chose root mean-squared error. I chose this over the coefficient of determination because it is more punitive toward misses for high-ranking players (in either projection or actual), which is desirable given the stakes are higher near the top of draft.

Evaluation Metric #2: For comparing my model to the benchmark model, I chose Draft Rank Value. Jesse Fischer's Draft Rank Value metric contains the value $RMSE_{Ranking}$. This works as follows: 1) For an NBA Draft model with ranked players, display the name and rank next to the player's actual total Win Shares. 2) Take the Win Shares list and organize it top to bottom and put it alongside the actual Win Shares. This is the expected Win Shares for each draft slot. 3) Take the root mean squared error of the difference: abbreviated $RMSE_{Ranking}$.

There is one other value used in the formula. According to Fischer, "$RMSE_{Worst}$ is defined as the RMSE value for the worst possible ranking for a given year (optimal ranking reversed)."

Draft Rank Value (DRV) is then calculated by:

$$\textbf{DRV} = 100*(1-(RMSE_{Ranking}/RMSE_{Worst}))$$

Fischer suggests that this model works because it punishes for both overvaluing and undervaluing prospects, while more severely punishing mistakes at the top of the draft (where the stakes are higher).

II. Analysis

Data Exploration

The dataset for this project is a custom collection of stats obtained from the following sources via web-scraping: Basketball Reference and RealGM for historical stats, DraftExpress' Pre-Draft Measurement Database, as well as Hoop-Math.com and NBAMiner.com for play-by-play data.

Basketball Reference has the strongest NBA database, so I will use it to collect historical statistics for all NBA players, including WS. RealGM has the superior international and college database, collecting per-minute and advanced statistics since 2003. By making use of their entire database, these per-minute statistics can actually be converted to the more telling per-possession statistics. This database will also allow for standardization of international stats (see below) and provide the primary inputs on my final model. DraftExpress' database has physical and athletic testing on the majority of players dating from around 2003 as well.

NBAMiner.com has more in-depth play-by-play data since 1997 which can be used to generate scores for individual skills. Hoop-math has similar, albeit less thorough, collegiate play-by-play data dating from 2012.

The main dataset will be limited to players who were playing in non-High School leagues outside the NBA at some point since 2003, who later played in the NBA. There are 2525 player-seasons with about 76 features of which to make use. The RealGM draft database includes players who were either drafted by a team or played in the NBA.

Statistical Translations

Following collection of data, the stats for players around the world to be standardized since the model will include both college and international prospects. The different quality of leagues around the world means that it's more impressive to put up stats in one league than it is in another. As Layne Vashro details at Nylon Calculus, by comparing stats for players who played in multiple leagues within a certain span of time, we can measure the level of competition in all of the various leagues worldwide and standardize the statistics accordingly.

The statistical translations had the following process: 1) Determine which players had played in different leagues in either the same year or consecutive years. There is a working assumption that there are as many players moving forward between leagues as there are backward. 2) If there are at least 40 players between two leagues who had played at least 200 minutes, take the median ratio of the statistical feature between the two leagues. The 'original estimate' of a league's relative strength in each statistic is then the mean of all these median ratios. 4) For all these leagues, I take the means of their neighboring leagues to be the strength of their competition. The 'true estimate' is then simply the product of the original estimate and the 'competition score'. So, I standardize this estimate relative to the NBA Regular Season's estimate, before taking the median ratio of the updated statistical features in the 'true estimate' like in #2. 5) This has the effect of updating the 'competition score', so when this is again multiplied by the fixed 'original estimate', a new 'true estimate' is formed.

The whole point of this is to have the new competition score be continually updated by being multiplied by the fixed original estimate. It only takes a few iterations before the 'true estimates' stabilize with the updated information gained, but I used 100 iterations to be safe. The results pass a few 'basketball sanity checks': If the NBA Regular Season is 1.0, the only league which consistently rates as higher than 1.0 in the majority of categories is the NBA Playoffs. The Spanish ACB, known for being one of the highest-quality European leagues has one of the highest ensemble of scores, and a few pass-happy European leagues have slightly higher assist translations than the NBA Regular Season. Weaker leagues that have good original estimates because their neighbors are even worse wind up with relatively low final estimates.

With these translations, I can simply multiply a player's original pre-NBA statistics by the translation factor to produce new estimates.

Feature Creation

We can add additional features to those in the original dataset. A single statistic over the course of a college season (~30 games) might not be as helpful as an ensemble of related statistics. For example, a player's shooting ability might not be captured solely by their 3-point percentage, which tends to be highly variable season-to-season. We should also take into account their 3-point attempts, long-2 %, and their free throw % as well to generate overall shooting scores. The weights of each statistic can be determined by training a regressor onto a target. (My early investigations suggest that a linear regression provides superior performance.)

What will the target of this regression be? Well, there is no single definitive shooting statistic, so I will use the robust NBA shooting data on NBAMiner.com to create overall shooting scores in the NBA. I can adjust the coefficients of the formula until they pass a quick sanity test. (In this case, are the great historical shooting seasons at the top of the list?) I can then train pre-NBA shooting statistics on a player's rookie-year outside shooting score to generate coefficients for each feature in a linear model.

Meta-Features

The dataset only stretches back to 2003. This might seem like a long time, but it means that players drafted since ~2010 or later probably haven't reached their peak. To solve this, I could either A) incorporate more (spotty) historical data, or B) impute data based off predictions for young players. Option A would limit the value of my features while option B creates some projected targets in lieu of real ones. I decided that B would preserve the overall method of the model and score creation.

One of the main benefits of creating these scores is that it allows for the use of fewer features, which reduces the effect of model bias. I can use the handful of scores along with physical profile and age to fit a cross-validated algorithm onto the data. So, a two-step process occurs:

projecting current NBA players' futures and then fitting a model based on the actual performance or projection.

Making Predictions

For the first part, I used the entire Basketball-Reference database. Since the feature to be created (Win Share predictions) was speculative, I wanted a simple model that was not subjected to the biases inherent in each era. So, I trained a simple 20-nearest-neighbors distance algorithm just using the features Age and Win Shares to predict the average of a player's top two seasons for players who had played past age 25. I then applied the algorithm to current players who were under the age of 25. If a player's predicted win shares were higher than their actual total, it replaced the actual top-2 season average.

Sample Dataset

Here is an incomplete look at a single player-season. While there are many more features, I tried to include one of each type. We see the athletic feature (speed) the physical feature (weight), the season, a sample statistic (PTS), the player's rookie year feature from which we can generate meta-features using regression techniques (OS_Score), age, and the target variable (WSPredict).

Name	Speed	Weight	Season	PTS	OS_Score	Age	WSPredict
C.J. McCollum	11.02	86	2010	20.6	3.4	18.3	6.83

Algorithms and Techniques

It occurs to me that there are several competing ideas when it comes to prospect projections:

A) Players who have a good statistics will tend to be successful.
B) Having a crucial weakness can be devastating while having a crucial strength can be a huge advantage.
C) Players who have similar features will have similar career trajectories.

Idea A intuitively makes sense and implies employing a regression model to predict future WS. Here we are more interested in overall fit, so we can use the (adjusted) coefficient of determination as a guide to develop a linear regression formula. However, this method is too simple on its own: the regression model(s) will likely overrate many "types" of players will underrating others.

In examining the patterns of the regression results, we might encounter scenarios where a player's skill (or lack thereof) in a particular area takes on a increased level of importance for their projection. Players are in many ways primarily defined by their strengths or their weaknesses. If a player projects as a historically great shooter or defender, that is tremendously important because these types of players are some of the most valuable players in the NBA. Similarly, a player's lack of ability to shoot or defend can keep them off the floor entirely.

Using a combination of my meta-features and physical/athletic features, I tried several different algorithms using a 'GroupShuffleSplit' cross-validation. I will provide a brief overview for each regressor inclusion based on their explanation in the Sci-Kit Learn User Guide:

- Linear regressions are a simple way to construct an additive linear model using ordinary least squares regression. It is resistant to overfitting because of its simplicity, but there is not much room for hyper-parameter optimization of it, and if the feature-space has complex relationships, the method of generating coefficients for each feature is not sufficient.
- Decision trees operate by learning decision rules from the feature space and are easy to interpret. Tree methods are useful for looking at more complex relationships between variables. However, decision trees are prone to overfitting and as a result, sometimes fail to generalize well to new data. They are almost always outperformed in cross-validation testing by forests of random trees.
- As an ensemble estimator, random forests aggregate the predictions of decision tree estimators to improve the ability of the trees to generalize well to new data. Here, the best splits in the trees are from a random subset of the data to ensure greater robustness. Random Forests generally perform well out of the box and are resistant to overfitting.
- Extremely randomized trees takes the process one step further but instead of looking for the best split, the threshold splits are now chosen at random and the best of these then operates as the rule. This can reduce the variance in the model relative to random forests at the expense of an increase in bias.
- Bagging is another ensemble method where the individual predictions are aggregated, which might improve the performance of the base model and can further reduce the variance. Because they combine individual predictions, they are very resistant to overfitting and work well with complex models.
- Gradient boosted trees are yet another ensemble method that builds a model in a step-by-step fashion. According to the Sci-Kit Learn user guide, the advantages of GBRT are:
    - Natural handling of data of mixed type (= heterogeneous features)
    - Predictive power
    - Robustness to outliers in output space (via robust loss functions)
- A nearest neighbor algorithm may also prove promising. It operates under the assumption that samples with similar features will have similar outputs, and aggregates

the outputs of the K-nearest neighbors into a single prediction. Despite its simplicity, it is well-suited to handle a complex feature space because it is reliant on past performance.

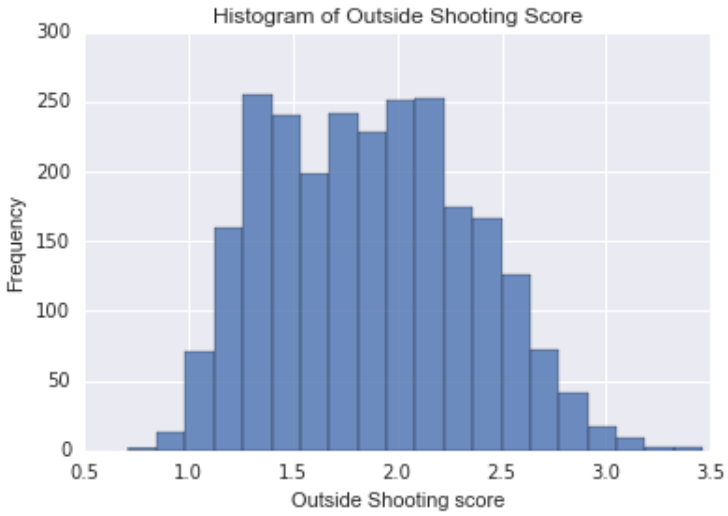
After testing, I employed a bagging regressor using these meta-features to estimate the error using a bagging estimator with an extremely randomized trees algorithm. Then I add the prediction to the linear regression to produce a final projection.

Benchmark

Fortunately, the website tothemean.com offers a valuable comparison of various draft models that can put the results of my model in context. The experimental accuracy of the various models over time are displayed on the website, and one can cycle through different models which best predict different “all-in-one” stats (Win Shares is used by default).

The benchmark model will be using the given Win Share values to compute the averaged Draft Rank Value (see the next section for details) for the ‘actual draft’ selections, which I calculated to be 45.1 (where 100 is perfect) for the years 2006-2015. 2006 is chosen as the starting point because it is the first year which implements the 19-year old rule for rookies, so players entering the league will not be coming out of high school, for which data quality is poor. If my model can outperform NBA General Managers or at least come close to the benchmark, I know it will have some value.

Exploratory Visualization



The visualization here is my meta-feature, ‘OS2’, or Outside Shooting Score. I chose this because the meta-features may be confusing. Whereas single-season statistics can be noisy, especially with regard to shooting, the meta-features are supposed to boil all the relevant features down to a single categorical score. We can see that the shape is fairly normal, albeit a little right-skewed. As we can see, the vast majority of players have outside shooting scores between 1 and 3, but there are a few outliers on the far right, corresponding to the historically good shooters that the regression system will favor.

III. Methodology

Data Preprocessing

Removing Outliers

I investigated my features for outliers by sorting the dataset by each feature. The vast majority of the data seemed legitimate, with the following exceptions: Francisco Garcia’s 2004 2-point percentage (a created feature calculated by 2PM/2PA) was over 100%. Pape Sow’s 2003 and Alex Stephenson’s 2012 seasons had impossible 3-point percentages as well. I was worried that the data was untrustworthy for these players, and being that they were neither top picks nor top performers, I simply had them removed from the dataset.

It appeared that some players that had an extremely high translated 3-point percentage of 83.6%. In fact, these scores were 100% prior to translation. This always occurred because these players had taken very few three-point attempts. I was worried about the effect that these unreasonable percentages would have on my data, so I treated percentage above 75% as if they had taken zero 3-pointers and converted them to NaN.

There were also impossible speed and agility scores for Rodrique Beaubois and Deandre Liggins. I converted these scores to the mean. There were also a few 0s entered into the athletic testing data, so I converted these to NaN with the intent of dealing with them later. All infinities in the dataset were also converted to NaN.

Feature Creation

I then created the following features: Wingspan-to-height ratio, weight-to-height ratio, steals added to blocks, the product of steals to blocks, the sum of rebounds, steals, and blocks, and finally the product of rebounds, steals, and blocks.

After this, I used the play-by-play data obtained from Hoop-math.com, removed the percentages and converted them to decimal to create the following per-possession features: 2-point jumpers, field goals at the rim, rim shots unassisted, 2 point jumpers unassisted, and 3 point jumpers

unassisted. These would be later used in my creation of Inside Scoring and Outside Scoring features.

Feature Transformation

It was important to remove skew from the features in the data to create normal distributions where possible. These should help to linearize the data and eventually improve the fit of my model.

For missing athletic data, I ran a multiple linear regression based on a combination of physical and statistical features and fill with the predicted results. The regression returns low, but still positive coefficients of determination, so the features do a slightly better job than a straight line.

From there, I created two more features: 'Vert_adj', which is the product of the no-step vertical and maximum vertical leap, and 'Move_adj', which is the product of speed and agility testing.

Score Creation

Finally, I create scores for outside scoring, inside scoring, defense, and playmaking using a combination of statistics and physical measurements in those realms using another multiple linear regression. As stated previously, I can then train pre-NBA statistics on a player's rookie-year scores generated from NBAMiner data to create coefficients for each feature in a linear model. Since college play-by-play data is only maintained since 2012, I will have to create two different linear regressions (one on the whole dataset and one with college players since 2012) to generate shooting scores, with the post-2012 regression taking precedence.

Implementation

With preprocessing completed, I moved on to implementation. I started off by using a random forest regressor to examine the feature importances. Rather than implement a feature selection method, I decided to use the feature importance scores as a loose guide for future regressions. Unsurprisingly, age and defensive scores were picked out as some of the most important features.

At first, I tried to predict future Win Shares ('WSPredict') by using a variety of machine learning methods. Using various tree methods like random forests or extremely randomized trees performed at roughly the same level as a linear regressor, producing a RMSE of approximately 2.44. The linear regression produced the most 'predictable' results, finding the best pre-NBA performers whereas the Tree methods had some odd choices for the most promising prospects.

However, the linear regression was not fully satisfying, so I decided to use the linear regression as an anchor from which I could generate a more accurate predictions. The linear regression is

prone to overrate some types of players while underrating others. The following features: 'Age', 'PTS', 'TS', 'TRB', 'STL', 'BLK', 'FTr', 'AST', 'ATr', 'FT%', and 'PF' produced reasonable results (and an RMSE of 2.44), so I took the error based on this linear prediction as the target for my next regression.

I then used the following features to estimate the linear error: 'Move_adj', 'Vert_adj', 'Height', 'WHr', 'WtHr', 'OS2', 'D', 'IS2', 'P', 'FTr', 'LinearPred'. To avoid overfitting, I didn't want to have too many features, and I used 2^x as guide here. As I have 2525 samples, I can afford to have 11 features. From here, I implemented a group shuffle and split cross-validation technique to try out the various machine learning algorithms I detailed above: linear regression, decision tree, random forest, extra trees, bagging, gradient boosted trees, and nearest neighbors, to see which performed the best.

Based on initial testing, I found that gradient boosting, bagging, nearest neighbors, and extremely randomized trees looked promising for predicting the linear error.

Refinement

Using a GridSearchCV to hyperparametize, I tried out the aforementioned regressors using a multitude of parameters under a GroupShuffleSplit as a cross-validation method. After recognizing that extra trees and nearest neighbors were performing ahead of other models in GridSearchCV, I tried a bagging estimator using both algorithms as a base estimator. Here, the Extra Trees won out, producing an RMSE of 2.5 as the base-estimator.

From there, I performed a cross-validation prediction using the algorithm. This is the estimated error. Finally, I can add the linear prediction to this estimated error to produce a second prediction. This bagging of the error estimate lowers my RMSE from 2.44 to 2.31. Finally, I added the weights 1.0, 0.4, and 0.15, which corresponded to play from the current year, the prior year, and two years prior, to come up with a single prediction for each player.

From here, I implemented the DRV process: 1) Lining up the actual pick WS with the sorted WS in each draft and taking the RMSE of the distance ($RMSE_{Ranking}$), 2) Lining up the reverse-sorted WS with the sorted WS ($RMSE_{Worst}$), and 3) calculating the DRV of each draft for the actual picks.

I then repeated the same process with the BEEM estimator.

IV. Results

Model Evaluation and Validation

The final model appears reasonable and in accordance with the solution expectation. The linear portion produces reasonable results and the bagging segment improves on it. Using extremely randomized trees with 50 estimators as the base estimator is appropriate for the problems.

The final model has been tested to evaluate whether the model generalizes well to unseen data. Using group cross-validation folds means that no two players are used in both training and testing. Indeed, the new data (the 2016 draft class) produces results that co-align with actual picks to roughly the same degree as prior years. The linear regression and bagging with extremely randomized trees as the base estimator are both resistant to overfitting.

A comparison of the RMSE of the different error-estimating tested in GridSearchCV algorithms after they have been combined with the original linear prediction is as follows:

	Extra Trees	Gradient Boosted Trees	Nearest Neighbors	Bagging with Extra Trees
RMSE	2.36	2.43	2.40	2.33 (now 2.31)

To test robustness, I perturbed the training data by running through the algorithm again, except this time I grouped my splits by draft class. Once again, I achieved a final RMSE of 2.33 (now 2.31). To further test robustness, I included the 2016 draft class in the training data before removing them from the final prediction and the results were nearly identical for the top-ranked prospects according to BEEM.

The results in the model are only as good as the data. I am primarily worried about the statistical translations and the overall lack of data. The athletic testing data may also be unreliable, as many of what most would consider to be the best athletes in the NBA did not have the best combine scores. The model, like every other statistical projection system, also does not include other important features that scouts study, like mentality, personality, or love for the game.

Justification

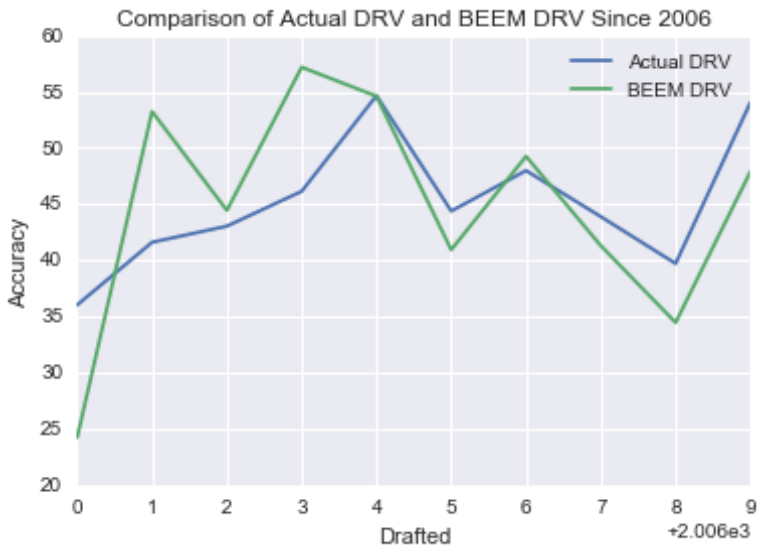
The draft rank value for the actual picks was 45.1 while BEEM produced 44.8 (where 100 represents perfect accuracy.) This difference favors the actual picks over BEEM, although given that the results are less than 0.4 points apart over the course of ten years, it is difficult to say with confidence that NBA GMs do a better job than the model.

The small amount of separation between the DRV of the actual picks and my model suggests that my model can be useful, but should not be used in lieu of ordinary methods. It produces

mostly reasonable projections while having its share of big-time hits and misses. As I believe it can be a useful tool in statistical projections, I think the final solution is achieved.

Conclusion

Free-Form Visualization



This visualization shows the variance between BEEM and the actual draft selections year-to-year. BEEM performs better in four of the ten years, but worse in five.

Reflection

Overall, I was moderately pleased with the solution. The model is mostly successful at sorting the bad from the good and the good from the great, but does not represent a breakthrough in scouting. Projecting a player's success through statistics is a possible and worthy, but there are so many intangible variables (including personality and luck) that statistical models will never be able to achieve anything close to perfection. Rather statistical models in this field are worthwhile for questioning our own biases and highlighting characteristics that may have been overlooked.

The most challenging part of the project was trying to make use of the inconsistent data between multiple sources. There were lots of intermediate bouts of trial and error to arrive at a final gameplan to create an original dataset and algorithm. While the project was time-consuming, it was also absorbing and captured my full attention for long periods of time, and it was rewarding to see it finally come together.

Improvement

This model could likely achieve better scores by making a variety of alterations. One possible change: picking a new, or creating a custom, regression target. I imagine that the (presumably) high correlation between different ways of evaluating player-seasons would only cause a minor change in the mean compared to my choice (averaging the top two Seasons or predicting them if they are under 25), but it is also unlikely that my choice is the best method.

Incorporating more international data would be ideal. For example, for some international players, the league in which they played did not meet my criteria for conducting translations because too few players had played the requisite amount of minutes in multiple leagues. So altering the statistical translation method might allow for more international players to make it all the way to the final dataset.

Incorporating data prior to 2003 would necessitate major changes to the set of features. From there, a smaller percentage of the data would have athletic features. The same goes for play-by-play data, which was used to generate inside and outside scoring features. In addition, it would probably mean moving away from per-possession, and even per-minute data.

More conservatively, there could be manually imputation of missing data. In addition, the statistical meta-features might be improved by using more stable data than rookie year scores as the regression target. Athletic features might be more useful if they incorporated player size into their calculations, but I found it challenging to create a sensible size-adjusted athletic features. Strength-of-schedule within NCAA ranks is another feature that might be incorporated, but data there is limited.

The most sensible improvement might be incorporating ‘advanced’ stats into features. I was initially wary of doing statistical translations with them because RealGM does not record advanced stats for national tournaments, but there are certainly ways around this (either by leaving national stats out or imputing them). My best guess is that effective field goal percentage and usage rate would rate as one of the most important stats, and they can be used in additional feature creation. **Note: This change was implemented with True Shooting Percentage rating as more important than eFG%. Adding ‘TS’ improved mean DRV performance by about 1.75 points.**

Update 2/16: Looking at the out-of-sample results, the model seems to overrate Kay Felder and Tyler Ulis, who have great stats but are very small point guards. The next change I would like to implement will be adding a height for position metric. RealGM’s position database is inconsistent and messy, so I may opt to use Basketball Reference positions. Another option is reducing the amount of positions from 5 to 4, to account for the positional fluidity that characterizes the modern NBA.

(Although it’s not documented here, I tried implementing a stacking algorithm to make an estimation of the linear error, combining multiple algorithms based on the amount of information gained obtained from a correlation matrix, but was not able to achieve superior results to the bagging method. Still, it is possible that if any of the above changes were implemented, a stacking algorithm could improve predictions.)