Charles Bolton, Tom Lancaster, Pat Rademacher, Chad Tolleson
CS 445/545: Machine Learning Group Project
6/12/2020

Introduction

Each year the National Basketball Association (NBA) holds a draft that is a life-changing event for many young men and the thirty professional team franchises in the league. Drafting a player is a multimillion-dollar investment for a team and can have a tremendous effect on the team's on-court success and the franchise's bottom line. Our group investigated and partially replicated a 2018 paper in the SMU Data Science Review titled "Predicting National Basketball Association Success: A Machine Learning Approach" by Southern Methodist University students Adarsh Kannan, Brian Kolovich, Brandon Lawrence, and Sohail Rafiqi[1] in which they discuss the use of machine learning techniques. They looked at the important attributes of a draft prospect and which attribute had the biggest influence on a player's career success.

In this study, the authors analyzed and compared three predictive models: support vector machines (SVM), logistic regression, and random forest. The study used both biometric and college stats datasets, and the authors defined success along the lines of "longevity," adopting the heuristic that a certain number of games played in the league (>174) was a sufficient metric of success. We adopted a similar approach including both college statistics and biometric information. One perhaps inconsequential difference in our study is that, in addition to the three models, we have also included a comparison of probabilistic classification with naïve Bayes. Also, we analyze several different versions of SVM.

Dataset

All of our data used in the project came from the data.world website including a player's final college season statistics, the number of NBA games played[2], and biometric information[3]. The number of NBA games played and college statistics date from professional basketball's inception in 1950 to 2018. Biometric information did not become standard until 2009 when the NBA Draft Combine began taking such measurements and since enough seasons have to accrue in order to deem a player's

[1] Kannan, Adarsh; Kolovich, Brian; Lawrence, Brandon; and Rafiqi, Sohail (2018) "Predicting National Basketball Association Success: A Machine Learning Approach," SMU Data Science Review: Vol. 1 : No. 3 , Article 7
[2] https://data.world/bgp12/nbancaacomparisons/workspace/file?filename=players.csv
[3] https://data.world/achou/nba-draft-combine-measurements

career successful or not, we limited the biometric data from 2009 to 2014/2015 which gives us a total data population of 283/326 players. We also considered using data such as the players' hometown and existing NBA performance statistics, but ultimately decided that data was unnecessary for our immediate purpose. The only relevant data missing to completely replicate the study is a player's rebounds and assists per game in college. Another potentially useful feature is a player's high school ranking. Below is a table describing our dataset in more detail:

Feature	Description	Feature Type
name	Player's Name	Categorical
pid	Player Unique ID	Categorical
college	Player's college	Categorical (0 = not from US college)
draft_yr	Year drafted	Ordinal
age_first_yr	Age at the start of their rookie season	Numeric (1-60 = drafted, 61 = undrafted)
draft_pick	Order of selection in player's respective draft	Numeric
hght_noshoes	Height w/o shoes, (inches)	Numeric
hght_wtshoes	Height w/ shoes, (inches)	Numeric
wingspan	Wingspan (inches)	Numeric
standing_reach	Standing Reach (inches)	Numeric
vert_max	Max vertical leap (inches)	Numeric
vert_maxreach	Max reach with vertical (inches)	Numeric
vert_nostep	Vertical w/ no steps (inches)	Numeric
weight	Player's weight at NBA draft combine	Numeric
body_fat	Player's body fat percentage at NBA draft combine	Numeric
hand_length	Player's hand length	Numeric
hand_width	Player's hand width	Numeric
clg_games	Total number of games played in college	Numeric

pts_ppg	Average points per game in college	Numeric
fg2_pct	Average 2 point field goal percentage from college career	Numeric
fg3_pct	Average 3 point field goal percentage from college career	Numeric
ft_pct	Average free throw percentage from college career	Numeric
guards	Binary variable indicating if player guard or not	Numeric
forwards	Binary variable indicating if player forward or not	Numeric
centers	Binary variable indicating if player center or not	Numeric
drafted	Binary variable indicating if player drafted or undrafted	Numeric
nba_gms_plyed	Total number of games played in NBA	Numeric
success	Dependent variable in the analysis for success	Numeric (1 = success, 0 = no success)

Preprocessing Data

To replicate the data used in our paper, we combined two different datasets into one Microsoft Excel spreadsheet. For the sake of time, we cleaned the majority of our dataset through the Microsoft program . This involved using the excel functions VLOOKUP and DATEDIF. The VLOOKUP function matched the names in each dataset to retrieve the college statistics found in one set to be combined with the biometric data in the other set. We encountered some difficulties with four of the players since they shared the same name as their father who also played in the NBA at an earlier juncture. We distinguished these duplicate players based on their start year in the NBA. The DATEDIF function helped us find the age of the player's first year in the league by subtracting their birthday from January 1 of their start year. By using these two functions, we were able to cut down on coding and preprocessing time.

Imputation

Following preprocessing, we still had a lot of missing information in the dataset, which was mainly located in five columns: hand length, hand width, vert_max, max_reach, no_step. We solved this problem by performing linear regression imputation. This was achieved by first producing a correlation matrix of all the features. Then, we selected the five features which were most highly correlated with the columns with missing values. For example, it was found that wingspan was the feature most highly correlated with hand length. Once identified, we created a linear regression model based on the two columns (the problem column with missing values removed and its strongest correlation column with no missing values). We then used this model to predict the missing values. Then we filled in the missing values programmatically with the predicted values. We chose regression because the missing data was over 3% of the total data in our set.

After our initial analysis was complete, we returned to this question and refilled the missing data using the mean of existing data, and then filling the remaining blanks with a 1 (which is only applicable when an entire column is empty). Interestingly, this did not have a meaningful impact on the performance of our algorithms. We speculate this is due to the fact that the affected features are not generally the most impactful when predicting successful NBA players. This may be due to either the reality that these features are relatively unimportant for this analysis, or due to the fact that the fields are heavily populated with imputed data. This might artificially decrease the spread of reported data, and thus reduce their predictive value.

Support Vector Machines

For the Support Vector Machines (SVM) we trained the models on the full dataset with an 80/20 split between training and testing sets. For our success threshold, we deemed a player a ‘success’ if the number of games played was at least 200. This is the threshold from the Kannan Kolovich study but prorated to represent the additional season in our data.

We experimented with several versions of the SVM algorithm. Four different kernels were tested, for some of the kernels several different hyperparameters were further tested.

The four kernels employed are:

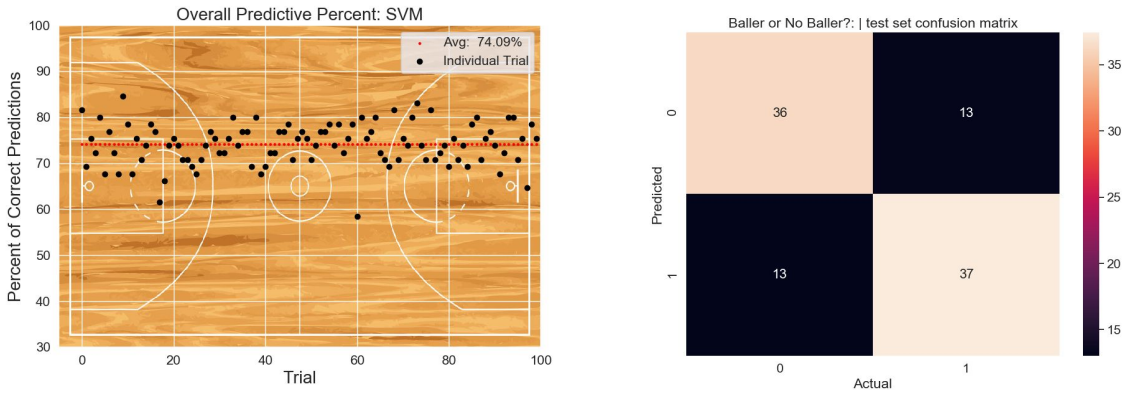
- linear

- sigmoid
- polynomial
- radial basis function

The hyperparameters considered are:

- *'C'* - Used by all four kernel functions. 'C' assigns a penalty to misclassified training examples. A larger value of 'C' produces lower bias and higher variance, while a lower value of 'C' produces higher bias and lower variance.
- *gamma* - Sigmoid, polynomial, and radial basis kernels have a hyperparameter of gamma. Gamma controls the weight of any one training example to the final results. Lower values of gamma mean that individual examples may have an outsized influence, while higher values of gamma mean that an example influence is moderated by its neighbors. In other words a smaller gamma value produces lower bias and higher variance, and larger gamma values produce higher bias and lower variance. The library we largely use to implement this experiment scikit-learn[4] has a feature that can automatically tune the gamma parameter during execution -- so we chose to take advantage of it.
- *degrees* - The polynomial kernel function has a hyperparameter for degrees.

The linear kernel implementation of SVM performed as follows:

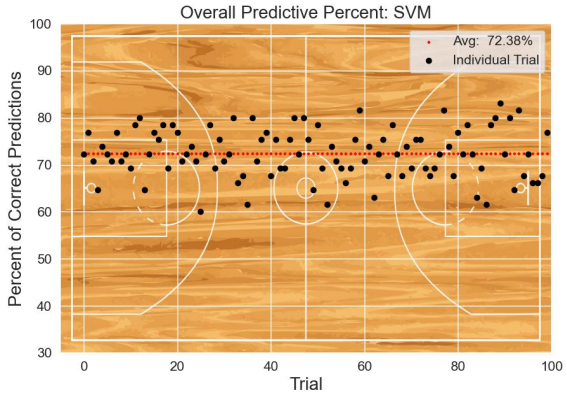


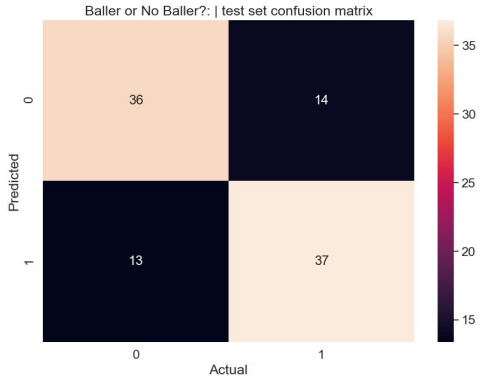
Above: Performance Measures for Linear Kernel SVM, C = 100

This model allows for us to inspect the features with the biggest impact on the results. Interestingly we generally landed on a 'C' value of 100 for all versions of the SVM, but when we reduce the value to 1 we see different results. The testing accuracy falls a couple of percentage points, and the most prominent feature switches from wingspan

[4] https://scikit-learn.org/

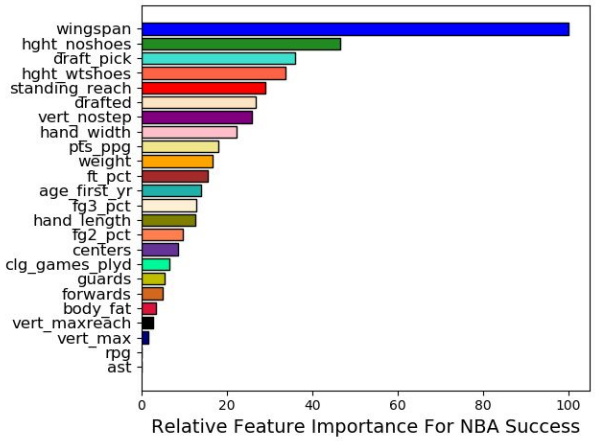
(for ‘C’=100) to draft_pick (for ‘C’=1). Draft pick, it’s worth noting, is the most important feature identified by the regression and random forest algorithms.

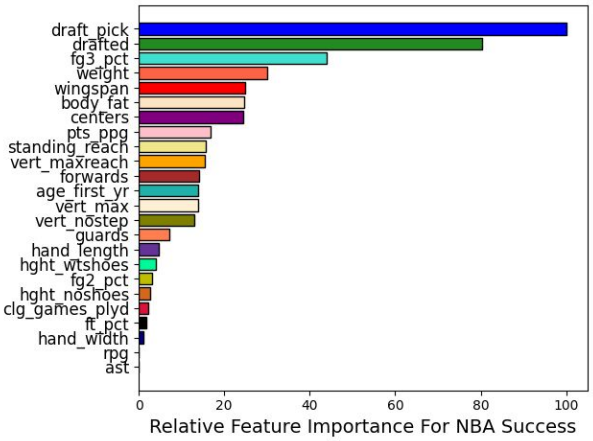




Above: Performance Measures for Linear Kernel SVM, C = 1

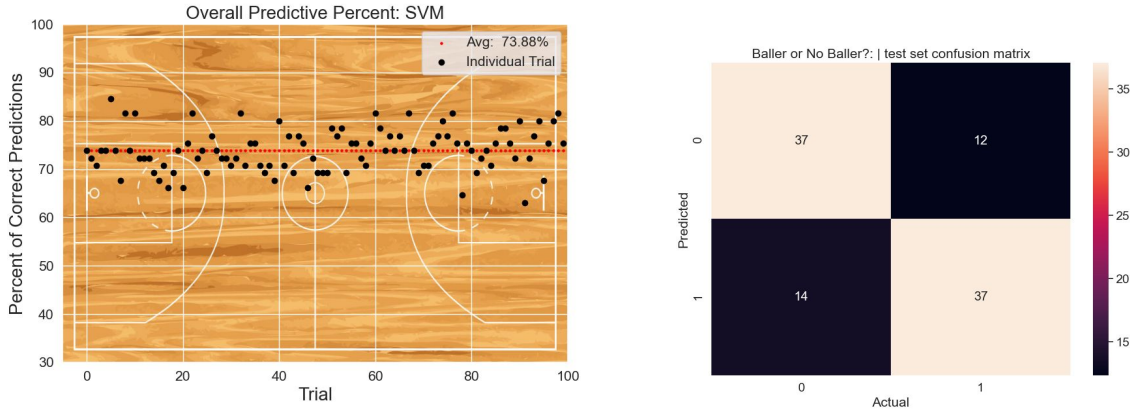
Here are the two charts of relative feature importance in the Linear SVM model, with different values of ‘C’:



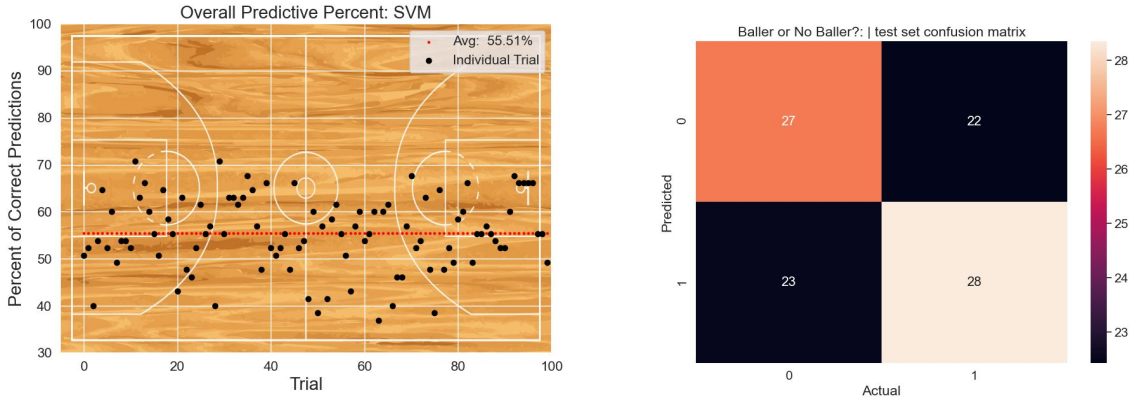


Above: example output of relative feature importance for Linear Kernel SVM, C = 100 & C = 1

The sigmoid kernel implementation of SVM performs even more dramatically different based on the ‘C’ value:

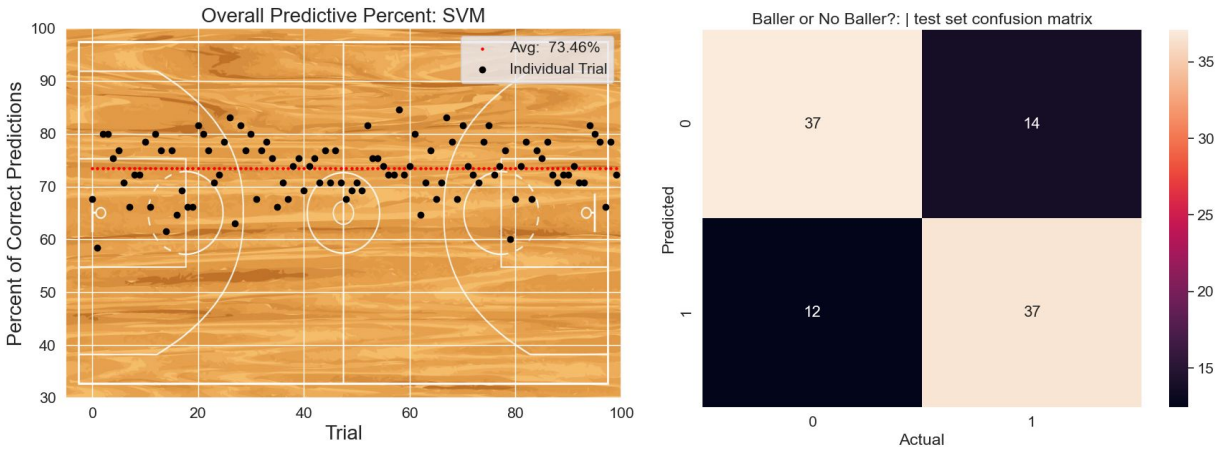


Above: Performance Measures for Sigmoid Kernel SVM, C = 100

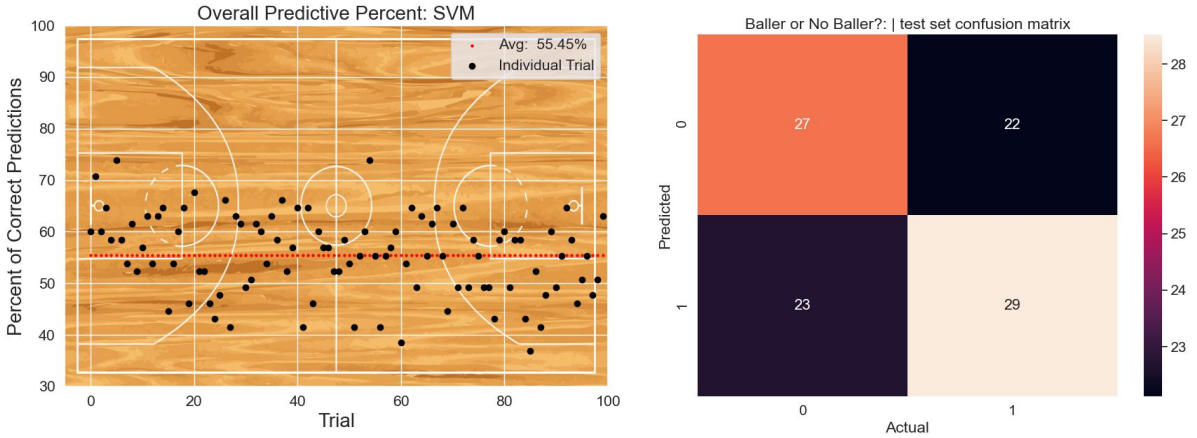


Above: Performance Measures for Sigmoid Kernel SVM, C = 1

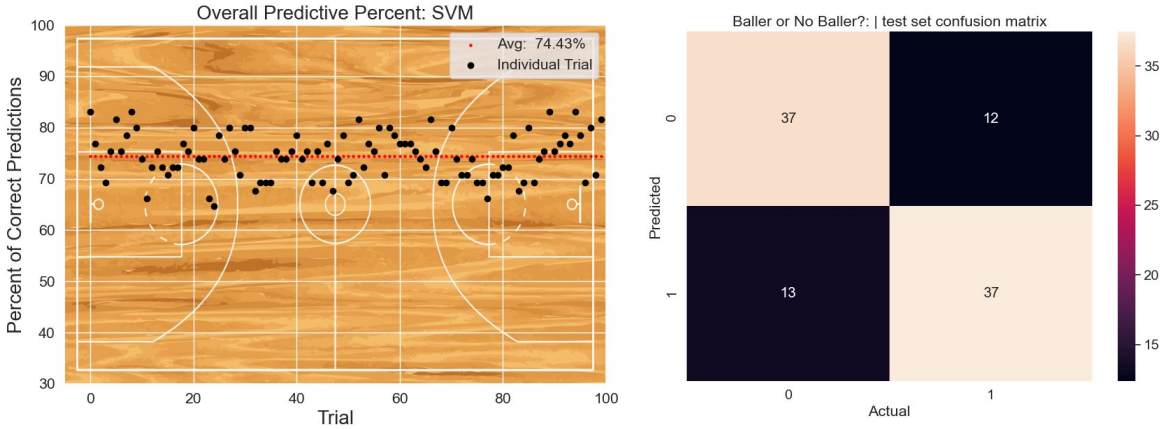
The polynomial kernel implementation of SVM performed similarly between different values of 'C'. The higher 'C' value produces better results. At the same time, changing the hyperparameter of degrees, which pertains solely to the polynomial kernel, has little bearing on the model's accuracy performance, and has a slight but noticeable impact on compute time. There seems to be a nice balance in performance and compute time at around 3 degrees and C=100.



Above: Performance Measures for Polynomial Kernel SVM, C = 100, degrees = 5

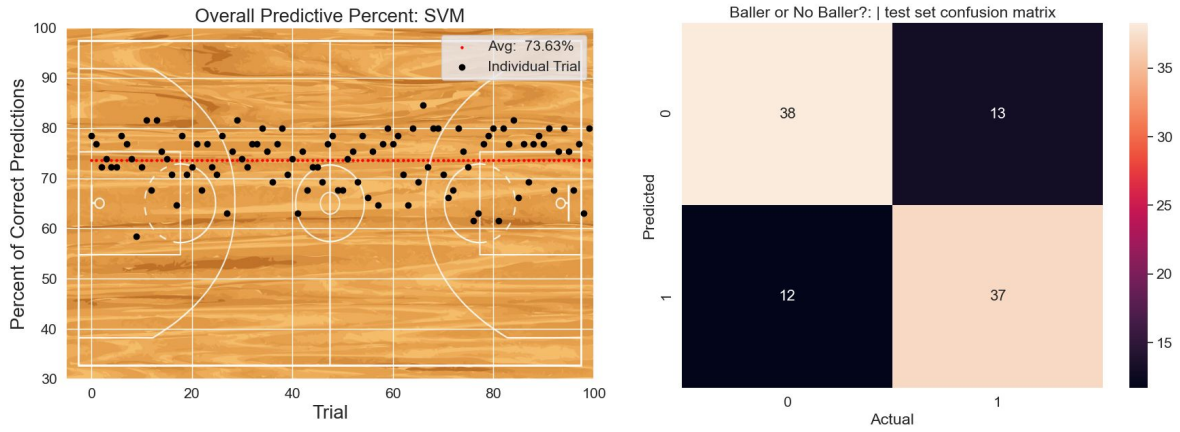


Above: Performance Measures for Polynomial Kernel SVM, C = 1, degrees = 3

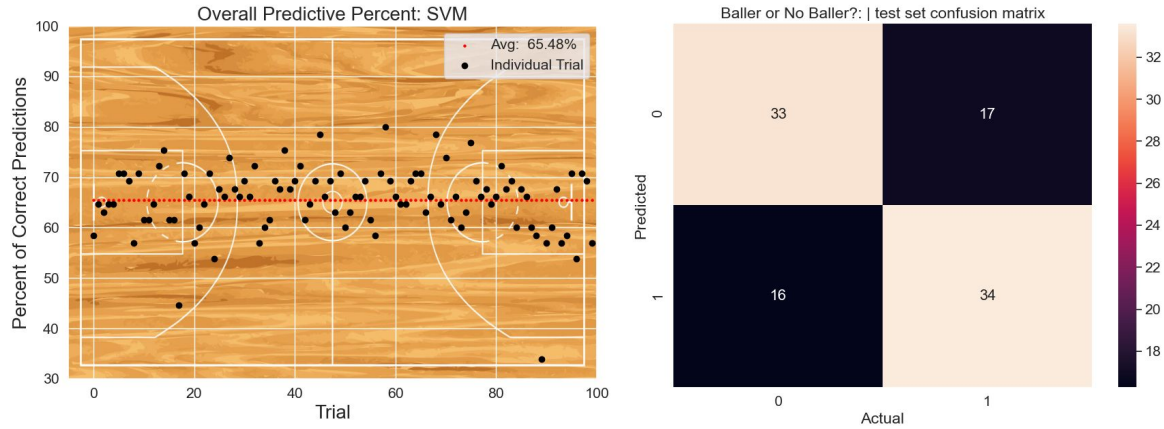


Above: Performance Measures for Polynomial Kernel SVM, C = 100, degrees = 3

The radial basis function (“rbf”) kernel implementation of SVM performs much like the others. Higher ‘C’ values produce better results.



Above: Performance Measures for RBF Kernel SVM, C = 100



Above: Performance Measures for RBF Kernel SVM, C = 1

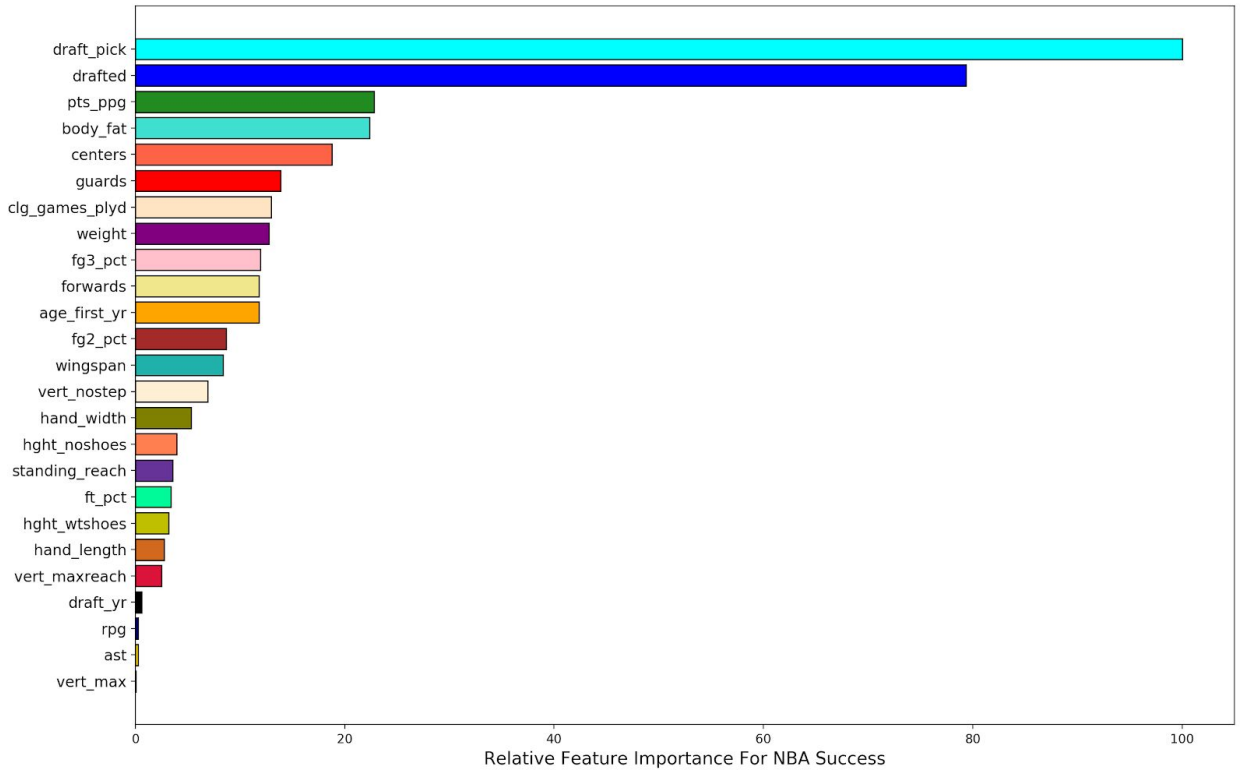
The most interesting takeaways from the SVM models is that a higher ‘C’ value consistently produces better results across kernels, the linear kernel (which is the most basic, relatively speaking) works just as well for this dataset as the others, and that the most important feature from the source data is different for low and high values of ‘C’. An additional project that would be interesting to further this experiment would be to parse out the reasons those two particular features (wingspan and drafted) are so important, and how they affect the different models as they do.

Logistic Regression

For the logistic regression model, we first trained the classifier on the full dataset with an 80/20 split on training and testing. For our success threshold, the first experiment

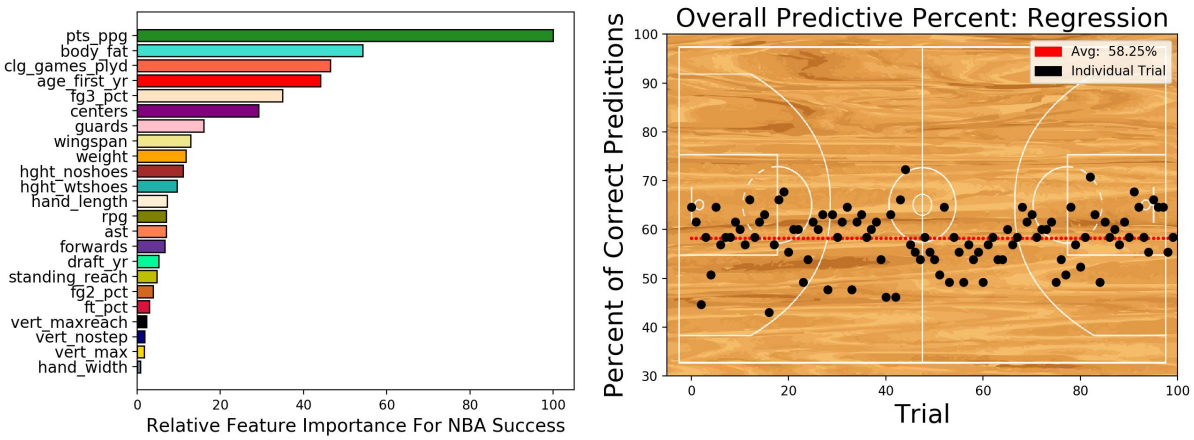
deemed a player a 'success' if the number of games played was at least 240 games. This number was the adjusted average number of games after adding to the average from the earlier study (because we included more seasons). Player 'success' was thus a binary classifier. We also experimented with lowering this threshold to 200, but this did not affect the model's predictive power, which on average correctly predicted that a player would have a successful career a little more than 70% of the time (see the plot below). When running the model, we also analyzed feature importance.

On the full-feature model, it was observed that the most important feature was the player's draft pick, followed by whether or not the player was drafted. The remaining features tended to shift around regularly, but other important features were *pts_ppg*, *centers*, and *fg3_pct*. This was pretty similar to the results obtained from the study. However, we suspected that including *draft_pick* in our data may be skewing the model, since a given player's draft pick number is enriched by the other data in the set.



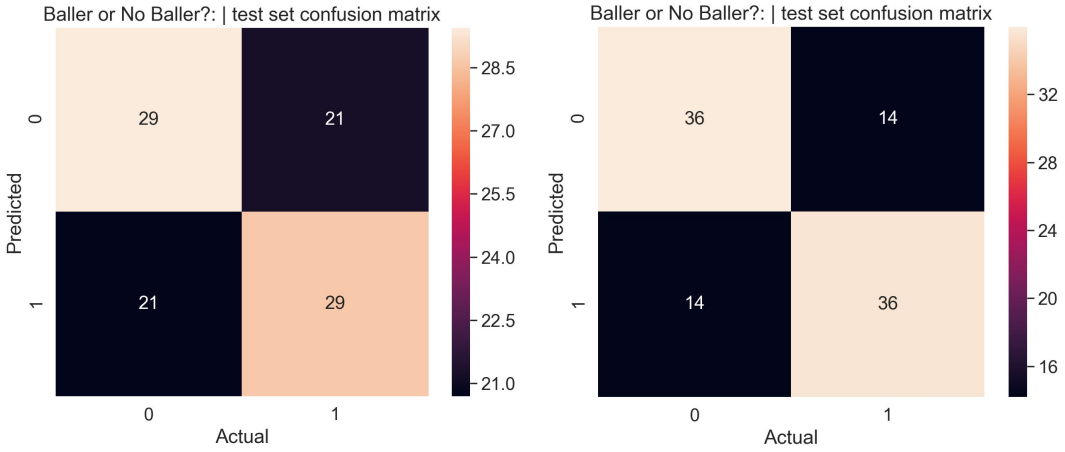
Above: example output of relative feature importance in the full-feature model

We therefore ran the model without the *draft_pick* or *drafted* features and saw the following:

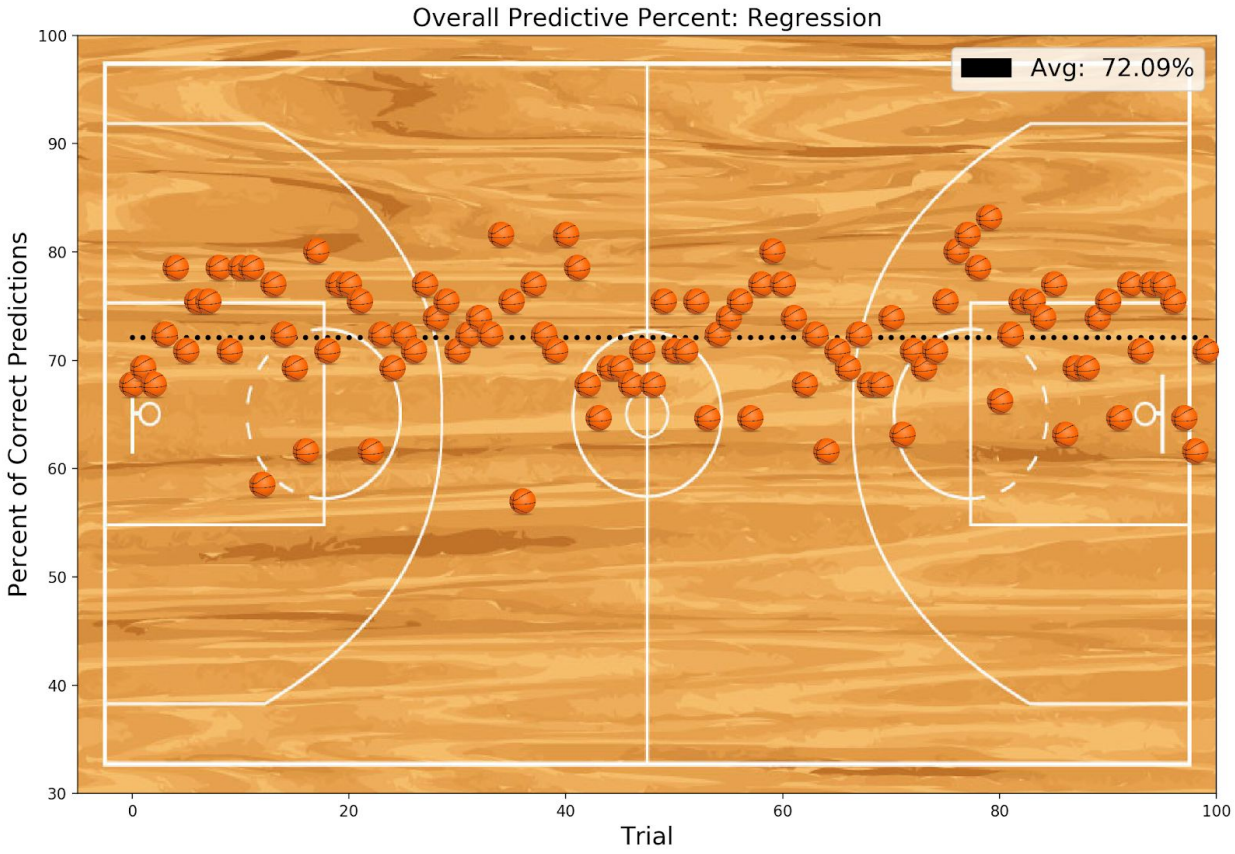


Above: running the LR model without draft_pick or drafted features

The clear influencer in this model was *pts_ppg*, but importantly, the predictive power was much worse, averaging worse than even the Bayesian model (below) and hardly better than random guessing, as shown in the confusion matrix.



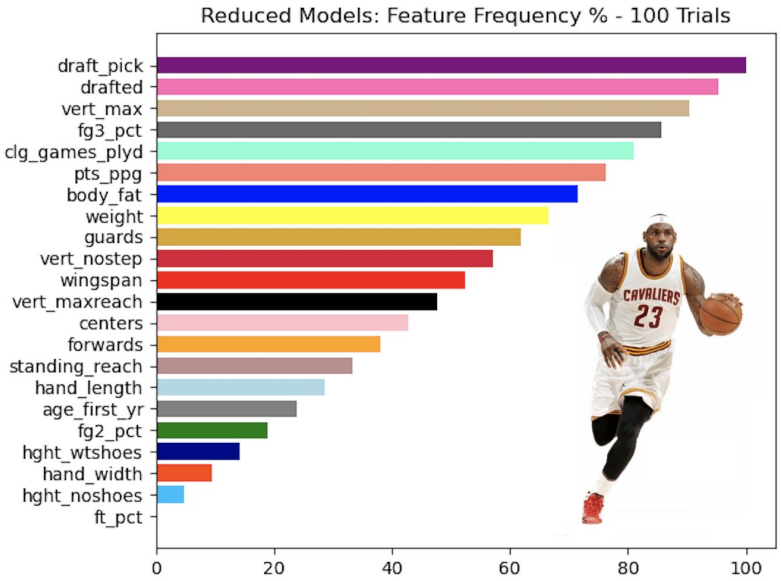
Above: confusion matrices for model without (left) and with (right) draft information



Above: average predictions of the logistic regression (full feature) model over 100 trials

Recursive Feature Elimination

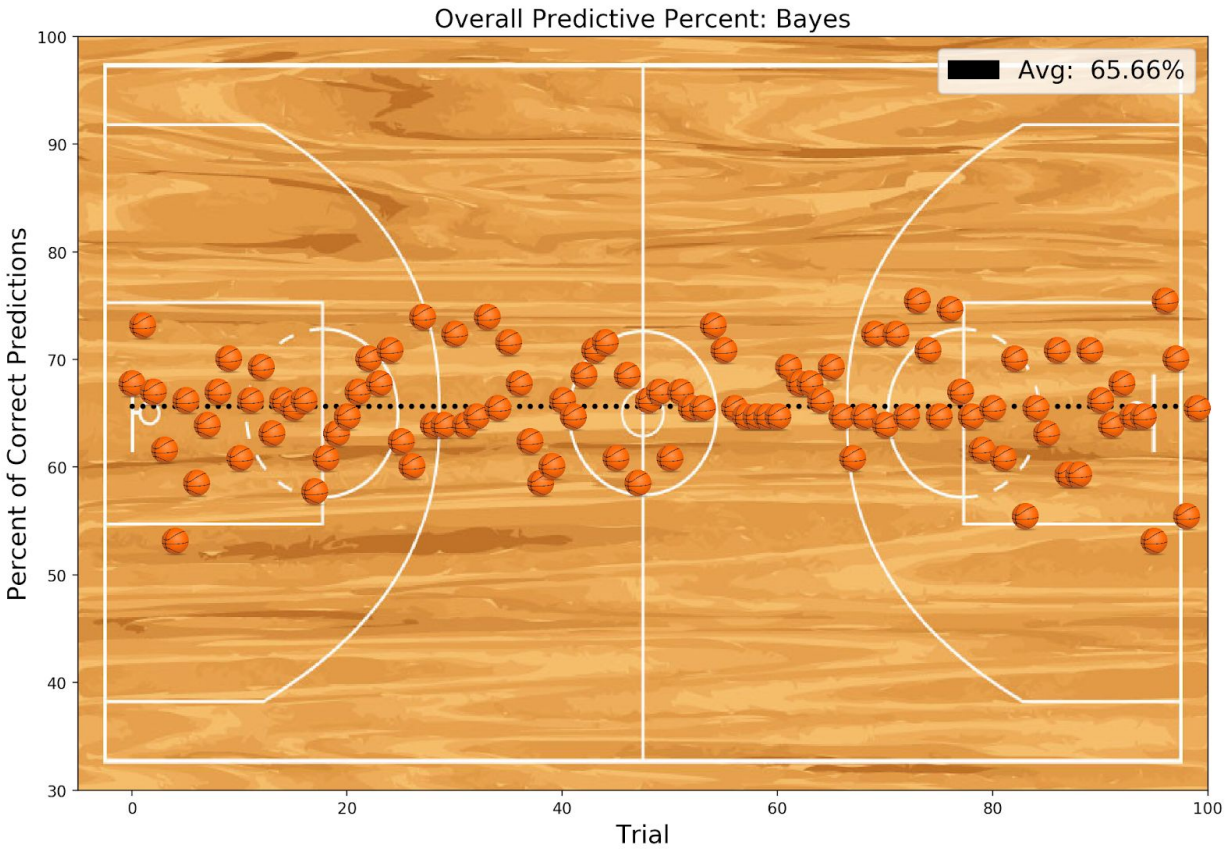
To learn more about our dataset, we also performed a recursive feature elimination (*rfe*) with the logistic regression model to see which features had the greatest influence on the prediction outcome. We measured this by running 100 trials of the *rfe* by slowly building out the feature model from 1 to *n*-1 and tallying each feature at each build. We then divided the tally counts by the highest possible tally count for 1 feature. Below is our results:



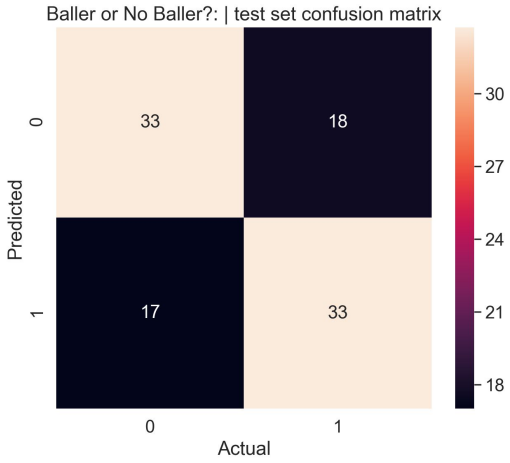
As the results indicate, we found the order in which a player was drafted or being drafted period had the greatest influence on a correct prediction. The rest of the features are listed in descending order by percentage.

Naive Bayes

As an extra experiment, we wanted to see how a probabilistic model would compare to a regression model. Knowing that a Bayesian approach works pretty well on spam email data with similar dimensionality to our dataset, we ran the data through a Bayesian classifier. The results were actually pretty poor, with the average prediction success around 65%, and what's more, the program took more than a minute to run. This should come as not a huge surprise, since many of the biometric features we measured in our dataset cannot sanely be interpreted to be independent of one another. For example, hand length/width and vertical measurements are directly correlated. Interestingly, spam email also exploits the naivete of feature independence despite its falsehood but performs much better. This could be due to the strong correlation of physical features mentioned above versus a perhaps weaker association between lexical and semantic features measured more appropriately by Bayes. As the model showed little success, we forgoed the opportunity to test it on a wider range of features, but we did notice this model also performs poorly (58%) without the *draft_pick* and *draft* features.



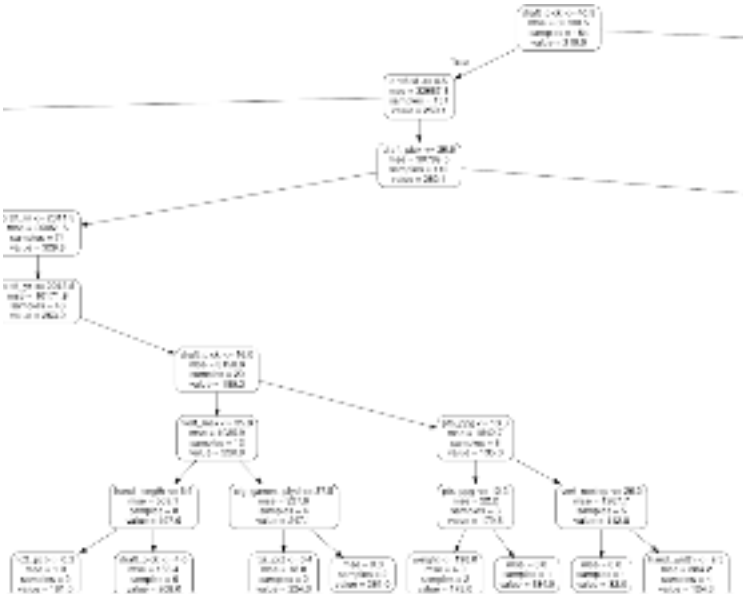
Above: The Bayes model accurately predicts success only 65% of the time on average. Below: confusion matrix for Bayes model

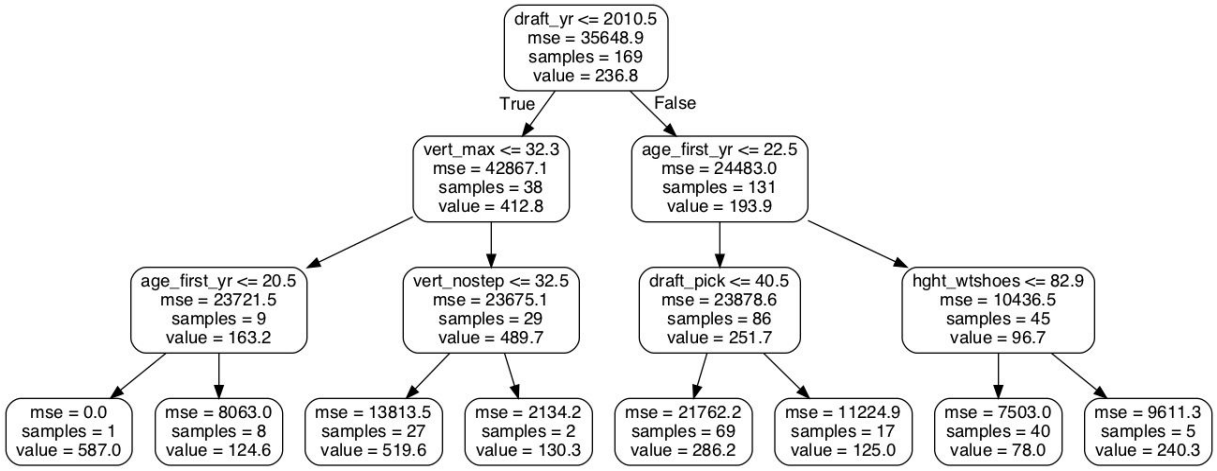


Random Forest

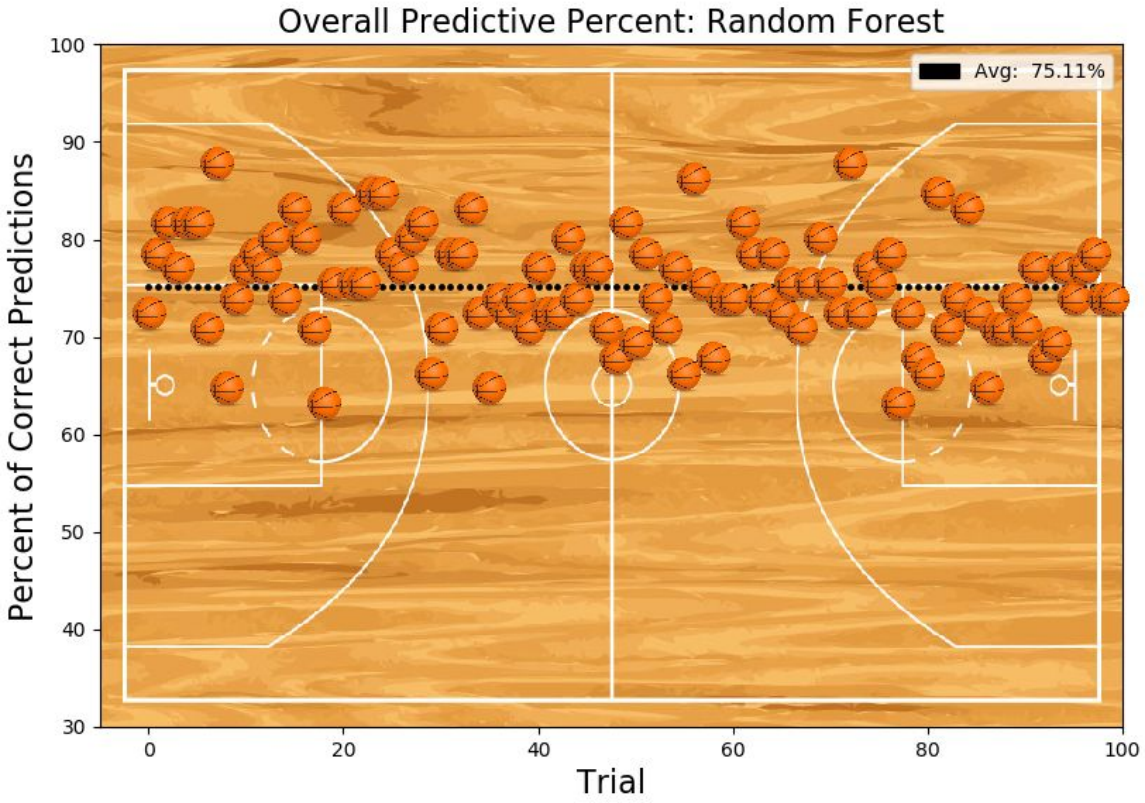
For the random forest model, we also did an 80/20 split with training and test data, respectively, over a total of 100 iterations, where each iteration shuffled the data. For each iteration, 100 trees were generated to estimate the results, and the results were then averaged among all trees for final predictions. When trying to run random forest in the same manner as the other models, by having the predictions be a 1 or 0 in regard to games played, the predictions came back as a float between 0 and 1, indicating the model ran continuously as opposed to discreetly. Therefore, we tuned the model to predict the number of games played. When predicting 240 games or more, we labeled it as a 1 for success and a 0 otherwise.

The results were similar to the other models, but here is where we had our highest overall accuracy of 75.11%. The most important features were about a neck-and-neck tie between draft year and draft pick, both accounting for approximately 20%, where draft pick was technically higher. Rebounds per game and assists had no impact whatsoever. These important factors were apparent when removed from the set, as accuracy plummeted to around 65% when removing either draft pick or draft year. Interestingly enough, removing a variable with no importance caused a somewhat significant change, discussed further below. When looking at the individual trees generated for determining predictions, the root was almost always based on a truth statement using draft pick or draft year. Below are two images: 1) a zoomed-in, partial look of a full tree; 2) a different tree with a maximum depth of 3 (this makes it easier to grasp all of the tree's nodes and leaves):

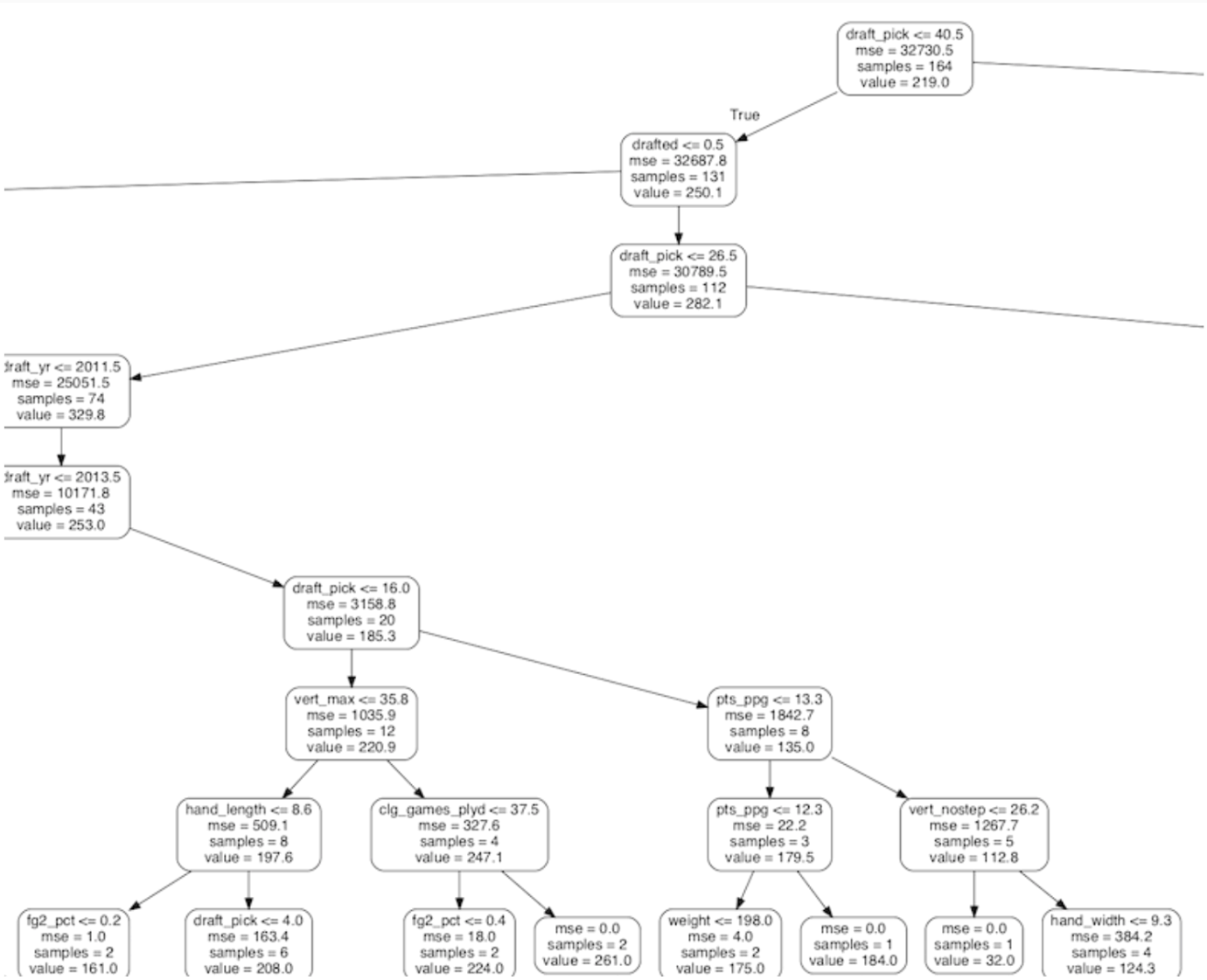


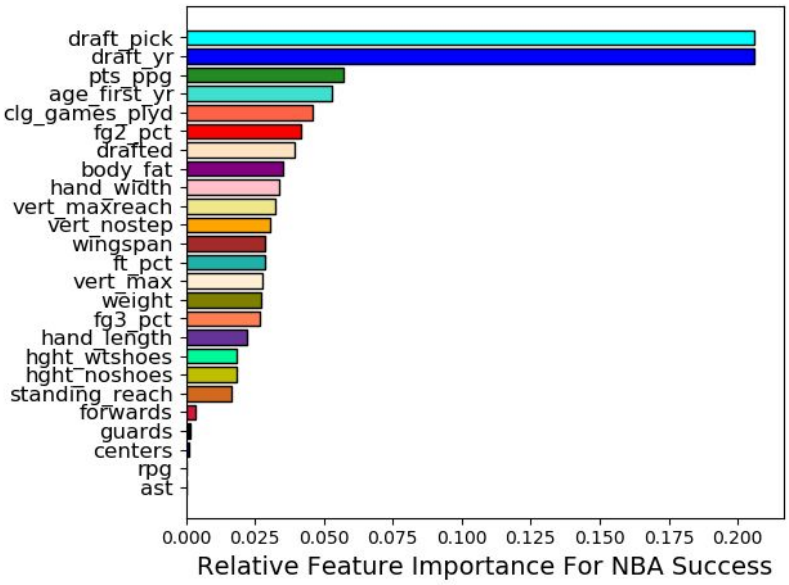


What was especially interesting about random forest was removing other variables from the data set to see what kind of effect it would have. Surprisingly, removing some variables resulted in a higher overall accuracy, though not by a huge margin. In the final model, there were only two variables that caused an increase in overall accuracy when removed individually, the most interesting being assists considering this had no degree of importance according to the original model with all variables. Nonetheless, this demonstrates how random forest differs from our other models, as having or adding more variables meant higher accuracy in their case. One can see the increase in accuracy resulted when dropping variables that had extremely. This is most likely due to how the data set was shuffled for each iteration, as we saw a multitude of other outputs when doing trials. Below are graphs and charts that demonstrate all the details of the model:

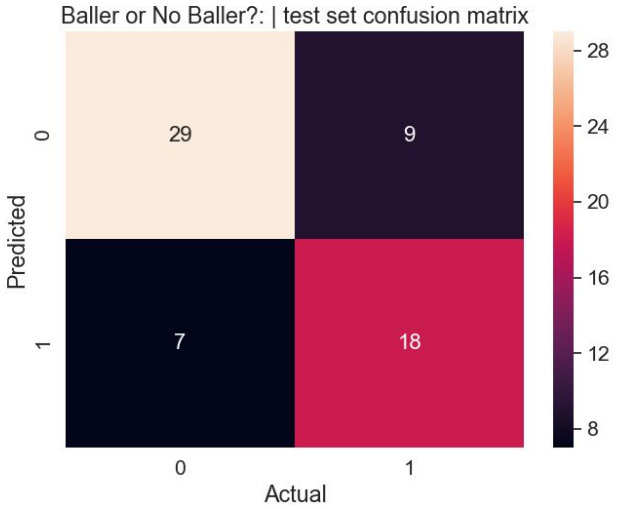


Above: Individual averages per 100 iterations when running random forest.


draft_pick <= 40.5
mse = 32730.5
samples = 164
value = 219.0
True
drafted <= 0.5
mse = 32687.8
samples = 131
value = 250.1
draft_pick <= 26.5
mse = 30789.5
samples = 112
value = 282.1
draft_yr <= 2011.5
mse = 25051.5
samples = 74
value = 329.8
draft_yr <= 2013.5
mse = 10171.8
samples = 43
value = 253.0
draft_pick <= 16.0
mse = 3158.8
samples = 20
value = 185.3
vert_max <= 35.8
mse = 1035.9
samples = 12
value = 220.9
pts_ppg <= 13.3
mse = 1842.7
samples = 8
value = 135.0
hand_length <= 8.6
mse = 509.1
samples = 8
value = 197.6
clg_games_plyd <= 37.5
mse = 327.6
samples = 4
value = 247.1
pts_ppg <= 12.3
mse = 22.2
samples = 3
value = 179.5
vert_nostep <= 26.2
mse = 1267.7
samples = 5
value = 112.8
fg2_pct <= 0.2
mse = 1.0
samples = 2
value = 161.0
draft_pick <= 4.0
mse = 163.4
samples = 6
value = 208.0
fg2_pct <= 0.4
mse = 18.0
samples = 2
value = 224.0
mse = 0.0
samples = 2
value = 261.0
weight <= 198.0
mse = 4.0
samples = 2
value = 175.0
mse = 0.0
samples = 1
value = 184.0
mse = 0.0
samples = 1
value = 32.0
hand_width <= 9.3
mse = 384.2
samples = 4
value = 124.3



Above: Most important features when running random forest -- draft pick and draft year are almost identical.



Above: Confusion Matrix for random forest

Average Accuracies When Removing Individual Variables - Random Forest

Variable Removed	Average Accuracy
draft_pick	64.92%
hght_wtshoes	69.23%
wingspan	73.23%
standing_reach	77.23%
vert_max	73.54%
vert_maxreach	73.23%
vert_nostep	71.38%
weight	71.69%
body_fat	72.62%
hand_length	69.85%
hand_width	72.0%
clg_games_plyd	73.23%
pts_ppg	73.85%
rpg	72.62%
ast	76.62%
fg2_pct	70.15%
fg3_pct	73.23%
ft_pct	69.85%
guards	73.23%
forwards	68.0%

Above: Eliminating variables 'standing_reach' and 'ast' individually result in higher average accuracies

Algorithm Results and Analysis

Below: feature, runtime and accuracy metrics based on 100 trials of the full feature set

	Logistic Regression	Bayes	Random Forest	SVM Linear	SVM Sigmoid	SVM Poly.	SVM Radial Basis
Accuracy	71.49%	67.69%	75.11%	74.09%	73.88%	74.43%	73.63%
Precision	72.12%	71.42%	76.57%	85.04%	85.1%	85.42%	85.24%
Recall	71.36%	55.55%	81.01%	85.19%	84.86%	85.27%	84.39%
Most Important Feature	draft_pick	n/a	draft_pick	wingspan	n/a	n/a	n/a
Run Time (100 trials)	.80 secs	1 min 9.09 secs	29.54 secs	3.26 secs	2.18 secs	2.26 secs	2.19 secs

Conclusion

Overall, when comparing our results to the original study, we witnessed similar model behavior with some important differences. Our results using SVM and logistic regression seemed to be significantly better, achieving more than 15% better average predictive power. The random forest models for both ours and the original study showed the best overall accuracy, so our results in this respect are more consistent with the original authors. All of our SVM models dominated in both precision and recall.

While we conducted our study in ways that were not identical to the ones from the original study, we can say with confidence that our models achieved similar and sometimes better results. It's evident that certain features do influence the final results, but as we saw different features selected in different models, it's inconclusive how reliable and important individual features are. Working with other data, perhaps related to health and/or lifestyle may have provided more insight, as some features like a player's draft year don't obviously have an impact on the longevity of their career. Additionally, there are certain improvements we could make to our study, such as taking the absolute value between a prediction from random forest and the test label, even if both were above or below the cutoff mark of 240 games. Part of the difference between our results and the original authors may also be attributed to the inclusion of more years' worth of NBA data.