

Predicting Results for Professional Basketball Using NBA API Data

Jacob Perricone
Institute of Computational and
Mathematical Engineering
Stanford University
Stanford, CA
jacobp2@stanford.edu

Ian Shaw
Institute of Computational and
Mathematical Engineering
Stanford University
Stanford, CA
ieshaw@stanford.edu

Weronika Świąchowicz
Institute of Computational and
Mathematical Engineering
Stanford University
Stanford, CA
wswiecho@stanford.edu

Abstract—Modeling and forecasting the outcome of the NBA basketball game poses a challenging problem to both the scientific and general public communities. To produce accurate results, this paper exploits a vast amount of aggregated team and game data. The initial analysis focuses on the identification of feature sets most representative of the final outcome of the basketball game. Thereafter, we develop mathematical models that learn team strengths and predict the winner of the basketball match-up with reference to distinct time frames across the season. Ultimately, we demonstrate the value of the selected models by showing their predictions for the first quarter of the 2015-16 regular season NBA games.

I. INTRODUCTION

The prevalence of statistical research specializing in forecasting of team based sporting events saw a drastic increase in the past two decades. The advancement of the field is attributed to availability of plethora of high quality data. The current work focuses on predicting the final outcome of the National Basketball Association (NBA) game in regards to both the score differential, as well as the the winner of the matchup. Historically, the game was treated as a static even developing prediction using full historical data [1], the current research identifies the need of acknowledging the effect of time variation in the team performance on the final prediction of the competition [2]. That can be achieved in several ways; (i) introduction of game discretization and incorporation of additional data in the prediction, (ii) assignments of weights putting more importance on the most recent performances of the teams, (iii) incorporation of the player based statistics, or (iv) incorporation of Brownian motion based models to predict the final outcome of the game.

A. Our Direction

In this paper we integrate optimization of feature space and decision analysis in the form of outcome prediction and strategy selection. The input utilized in all algorithms is a varying parsing of historical NBA data [?]. We use these varying sets of data to select the best features sets and make predictions using models like support vector machines (SVM) with linear, sigmoid and Gaussian radial basis function (rbf), Excess Trees and Logistic Regression. We train the models to predict whether the home team will win the game.

B. Related Work

Current best in the field of NBA game prediction is an accuracy of 74% [3]. This is significantly better than accuracy of predictions casted by basketball experts, which fall slightly under 70% [4].

In a paper utilizing classical machine learning techniques [1], a student at University of Wisconsin at Madison topped out prediction at 66.81% utilizing a maximum likelihood method. In attempt with other techniques, his results were as follows: Naive Majority Vote Classifier (63.98%), Multilayer Perceptron Method (68.44%), and Linear Regression (67.89%). Here, the use of principle component analysis settled on the following features: win-loss percentage of home team games, point differential per game of home team, win-loss percentage of previous 8 games for visitor, and the win-loss percentage as visitor or home as the respective situation of the teams.

The use of more advanced, and computationally taxing techniques of neural networks showed notable improvement of the game's final prediction. These results were 74.33% for the full regular season [3]. Here, the first three highest results were attributed to a feature set of four statistics. That accomplished better results than models testing all six shooting statistics, or all 22 statistics noted in a box score. In all these models the common features were the field goal percentage and free-throw percentage for each team.

The practical knowledge, along with some statistics, has broken down team ability to win games based on shooting, turnovers, rebounding, and free throws (in descending importance) [5]. Existing machine learning analysis of basketball typically pays heed to this analysis by Dean Oliver, who collaborated with the legendary coach of the North Carolina Tarheels, Dean Smith. Further features, as shown in the former study, focus on team based winning percentages in hopes of these factors being aggregated on a team level by those statistics.

II. FEATURE SELECTION

All data used in this project was accessed directly from NBA database [6]. Thus, we had access to a variety of team aggregated, player based, as well as detailed play by play

statistics for the past 32 seasons (this project only uses the first two data types). We gathered a set of over fifty statistics for each team playing in a game, all of which are gathered from the stats nba api. A variety of different data-sets, capturing different elements of a team’s play, were used in our analysis. We use used a team’s past season average statistics as well as the past season’s quarter i statistics, with $i \in \{1, 2, 3\}$. Lastly we gathered statistics for each team over the first half of the season, and test our models on the remaining half. Keeping in mind all changes in the game rules, as well as the changing nature the game is played we reduced the time scope of the data to 5 most recent NBA seasons.

Given a data set with over 50 features we faced a potential problem of over-fitting the training set. That forced us to explored feature selection methods which at their core reduce the size of the feature space. The idea behind feature selection is simple. We want to maximize the number of relevant independent variables and minimize the size of irrelevant features in our model. That is, we select the features that convey information about the output variable independently or conditionally on other relevant variables. In contrast, irrelevant features fail to convey any information about the target variable even when conditioned on relevant features and thus should be removed from the model altogether [7].

A. Extra Tree Classifier

In [8] Geurts et al. shows that the Extra Tree method reduces variance more strongly than other popular feature selection methods. Better performance of the tree-based technique is attributed to integration of ensemble averaging with randomly selected cut-points which replaced simple randomization employed by other methods. The basic idea of the method is a recursive partition of the learning sample into a decision tree each containing a classification prediction for the target variable. The method then integrates a prediction of several trees to cast the final output that minimizes the uncertainty in the final classification (for algorithm see, [8]).

In our analysis we test a set of 50 features each with 1650 data points per quarter. To maximize the computational power of the method we only select 10 trees starting with the 0^{th} seed. In addition we set the maximum number of features to 32, the unique number of team statistics. For the purpose of the feature selection, this method uses each vector as a variable and models if its effect on predicting the value of the target variable is relevant.

The method assigns an importance measure to each variable, i.e., the node, which is the total reduction of the class entropy due to the split, [7]

$$I(n) = |S.H_C(S)| - |S_t.H_C(S_t)| - |S_f.H_C(S_f) \quad (1)$$

where S defines the set of samples reaching each node n , $H_C(\cdot)$ denotes the Shannon entropy [9] of the class frequencies in a subset, S_t and S_f are the subset for which the test is true and false respectively. For each tree the importance score of the i^{th} feature is the sum of all values of the I function computed for all nodes n where the feature splits. The score

for the ensemble is just the average of measures of trees in the group.

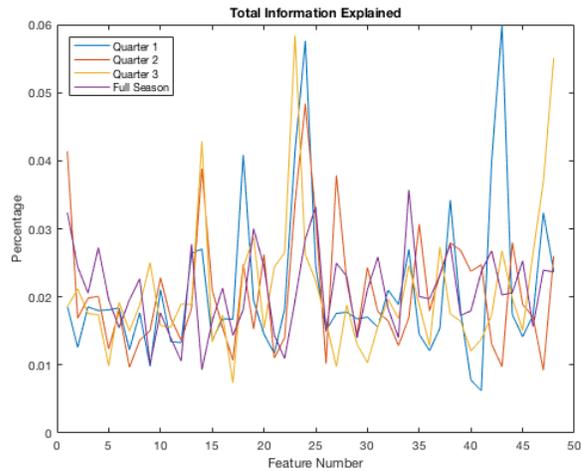


Fig. 1: Importance of features in the 2015-2016 season.

In Fig. 1 each score represents the percentage of the total information explained by the ensemble of trees with reference to a particular variable. It should be noted that the scores assigned to features containing full seasonal data are significantly different than score assigned to features containing only a single quarter data.

TABLE I: Number of Features Selected by the Extra Tree Classifier for each Type of Seasonal Data

| | |
|-------------|----|
| Quarter 1 | 16 |
| Quarter 2 | 18 |
| Quarter 3 | 12 |
| Full Season | 19 |

There were five common features selected for different data sets; losses at home, wins at home, win percentage at home, personal fouls per game when playing away, and number of steals per game when playing away.

III. CLASSIFICATION MODELS

All methods employed in this paper were implemented using `sklearn` library in `Python` assuming that we have a set of input variables X and a set of Target values Y . We tested a variety of different supervised binary classification methods. The following details all techniques used;

- 1) Extra Trees with 10 trees.¹
- 2) Gradient Boosting
- 3) K-nearest neighbors with 10 neighbors and 30 leafs.
- 4) Neural Networks
- 5) Logistic Regression
- 6) Support Vector Machine with the linear, rbf, and sigmoid kernels.

¹Detailed explanation of the method in Section 2.1

7) Non-linear Support Vector Machine

where we adjust for classifications preferences of different methods by changed the winner classification from 0 to -1 (SVM).

A. *K*-Nearest Neighbors

The idea behind the *K*-nearest neighbors is significantly different the methodology of other methods used in this project. That is, it makes decisions about the classifications of a particular target value because on the "vote", i.e., the classification associated with the *k* number of nearest input values [10].

We selected this method of classification due to its ability to reduce the effect of noise (associated with the value of *k*) on the final prediction. The larger we choose *k* to be the lesser effect noise has on the results, but at the same time it is hard to classify points.

B. Neural Networks

The Neural Network classifier in `Python` uses the Multi-layer Perceptron (MLP) algorithm [11]. At this core the idea behind the MLP algorithm is simple. It creates a directed graph with neurons corresponding to vertices and edges called synapses (node that edge connects only parent and children layers in the graph). The graph begins with the vertices corresponding to input variables, which in return are either connected with the outputs or the first hidden layer. The hidden layer consists of weights referencing the importance of the corresponding neurons on the synapses. Before the output of the neuron is computed the values passed by the synapses are processed by the integration and later by the activation function. This procedure repeats for consecutive hidden layers up to the graph's leaf.

The method in general gives good results for predicting sporting evens [3]. In addition, it automatically learns dependencies within the data and with high chance it can predict similar patters in new data sets.

C. Gradient Boosting

The Gradient Boosting (GB) method is based on the same principle as other supervised learning methods. That is, given a set of input variables and corresponding target values, it wants to classify *y* by minimizing the expected value of the logistic loss function $L(yx^T\theta) = \log(1 + \exp(-yx^T\theta))$ (predefined in `Python`). The uniqueness of this method comes from the fact that GB assumes the classification of *y*. It then uses a series of decision trees (in our case 100), with each tree attempting to improve the prediction error of its predecessor. That error is corrected at the rate equal to 1. That means that our methods will quickly converge to the optimal θ that minimizes the lose function *L* (to access the GB Algorithm see, Algorithm 1 [12]).

In [13] the authors compare a variety of supervised learning methods in classifying genome based data. For the tested data the classification obtained by the GB was significantly better than that of SVM or Random Forest.

D. Logistic Regression

Logistic regression is a fairly simple model that classifies the target value according to the following hypothesis function

$$h_{\theta}(x) = \frac{1}{1 + \exp(-\theta^T x)} \quad (2)$$

where *x* denotes a vector of input values, θ is the set of corresponding coefficients and $h_{\theta}(x)$ is the prediction casted by the model. To develop the logistic regression model `Python` minimizes final sum of $\frac{1}{n} \sum_{i=1}^n f_i(x)$ using the Stochastic Average Gradient (SAG) method [14]. Here $f_i(x)$ is a regularizer function defined as, $f_i(x) = \frac{\lambda}{2} \|x\|^2 + l_i(x)$ where l_i is the misfit function. SGA classifies each target value y_i using Algorithm 1 in [14].

This method is simple to apply and gives good initial results. In addition, it can be easily combined with more accurate results by producing a relatively good starting estimate.

E. Support Vector Machines

Support Vector Machines is a supervised learning problem. Thus it classifies *y* based on the characteristics of the training set. In particular, SVM splits data relevant to two separate classes with a boundary curve. The curve is created to maximize the distance between the boundary and the data. The classification of the new *y* given *x*, is simply a matter of placing *x* on either side of the boundary representative of a particular classification.

In our paper, we tested SVM classifiers [15] with different kernels; linear, rbf, and sigmoid. That simply means that the boundary will have the shape resembling these kernels. The idea of the non-linear SVM [16] follows the same idea with one addition. We initially translate the feature space in \mathbb{R}^n to \mathbb{R}^{n+1} to make it linear. Then we use the above set of kernels to define the shape of the decision boundary.

The advantages of using SVM models are numerous. Starting with computational efficiency even when the number of samples is smaller than the number of features. Continuing with ability to work with high dimensional spaces, possibility to define a variety of different kernels, and finally memory efficiency due to working only with the support vectors.

IV. RESULTS

A. Hyper-Parameter Selection

All hyper-parameters selected for these models were determined based on the analysis of 500 different constants (parameters interval $[0, 250]$ with step size 0.5). Final parameters are chosen based on the accuracy and computational efficiency of the program we run to obtain classification *Y* for a set of inputs *X*. For the *K*-Nearest-Neighbors we select *k* = 10, as larger *k* do not improve accuracy of our model by more than 0.1%, and increase the time computation by over 3 minutes for inputs with less than 600 samples. The Neural Networks models that gave the best results in reasonable time had 2 hidden layers each with 5 neurons and penalization on the l_2 norm $\alpha = 10^{-5}$. A larger number of hidden layers only increases the time necessary to obtain results and in some

cases decreases the accuracy of the classification. The Logistic Regression is specified with an inverse regularization constant of magnitude $\frac{10^{-4}}{\text{Input Sample Size}}$. For the SVM models we define the penalty of the error term to be 2.5. In our case that gave us the best accuracy of the classification.

B. Comparing Classifications

Our primary metrics of success were precision, recall, accuracy and the area under the receiver operating characteristic curve (ROC-AUC).

Precision deals with the inclination of a model for positive prediction, a ratio of true positives to all positive attributions.

$$P = \frac{t_p}{t_p + f_p}$$

Recall is a measure of the sensitivity of the model to a positive event, a ratio of true positive to all actual positive events.

$$R = \frac{t_p}{t_p + f_n}$$

Accuracy is a measure of how effective the model is at predicting outcomes, a ratio of correct predictions to all predictions.

$$A = \frac{t_p + t_n}{t_p + t_n + f_p + f_n}$$

The receiver operating characteristic curve plots a classifier in two dimensions, with the x-axis being the false positive rate and the y-axis being a true positive rate. Thus the closer the area under the curve is to the value of 1, the more true positives are output, creating a better classified.

$$ROC(AUC) = \int_1^0 TPR(x)FPR'(x)dx$$

TABLE II: 2015-16 Full Season Games with Cross Validation and Feature Selection

| Methods's Name | Accuracy | Precision | Recall | ROC-AUC |
|---------------------|----------|-----------|----------|----------|
| Neural Networks | 0.666014 | 0.679982 | 0.795871 | 0.713800 |
| Non-linear SVM | 0.663588 | 0.676749 | 0.817979 | 0.712372 |
| Logistic Regression | 0.677205 | 0.705065 | 0.779014 | 0.737191 |

TABLE III: 2015-16 Full Season with Cross Validation

| Methods's Name | Accuracy | Precision | Recall | ROC-AUC |
|---------------------|----------|-----------|----------|----------|
| Logistic Regression | 0.677205 | 0.705065 | 0.779014 | 0.737191 |
| Neural Net | 0.661123 | 0.685699 | 0.784817 | 0.716858 |
| Non-linear SVM | 0.663588 | 0.676749 | 0.817979 | 0.712372 |

TABLE IV: 2015-16 Quarter 1 with Cross Validation and Feature Selection

| Methods's Name | Accuracy | Precision | Recall | ROC-AUC |
|-------------------|----------|-----------|----------|----------|
| Gradient Boosting | 0.708940 | 0.717066 | 0.837081 | 0.772707 |
| Neural Net | 0.709000 | 0.728791 | 0.795719 | 0.776032 |
| Non-linear SVM | 0.704075 | 0.725215 | 0.803957 | 0.774346 |

TABLE V: 2015-16 Quarter 1 with Cross Validation

| Methods's Name | Accuracy | Precision | Recall | ROC-AUC |
|-------------------|----------|-----------|----------|----------|
| Gradient Boosting | 0.708940 | 0.717066 | 0.837081 | 0.772707 |
| Neural Net | 0.704862 | 0.732661 | 0.808162 | 0.769917 |
| Non-linear SVM | 0.704075 | 0.725215 | 0.803957 | 0.774346 |

TABLE VI: 2015-16 Quarter 2 with Cross Validation with Feature Selection

| Methods's Name | Accuracy | Precision | Recall | ROC-AUC |
|----------------|----------|-----------|----------|----------|
| Extra Trees | 0.702429 | 0.718956 | 0.814992 | 0.773042 |
| Non-linear SVM | 0.699197 | 0.721378 | 0.799810 | 0.771293 |
| Lin Kernel SVM | 0.710526 | 0.732782 | 0.803938 | 0.770081 |

TABLE VII: 2015-16 Quarter 2 with Cross Validation

| Methods's Name | Accuracy | Precision | Recall | ROC AUC |
|-------------------|----------|-----------|----------|----------|
| Extra trees | 0.702429 | 0.718956 | 0.814992 | 0.773042 |
| Gradient Boosting | 0.699197 | 0.712592 | 0.821899 | 0.771998 |
| Lin. Kernel SVM | 0.710526 | 0.732782 | 0.803938 | 0.770081 |

TABLE VIII: 2015-16 Quarter 3 with Cross Validation and Feature Selection

| Name | Accuracy | Precision | Recall | ROC AUC |
|-------------------|----------|-----------|----------|----------|
| Extra Trees | 0.708200 | 0.727137 | 0.809551 | 0.772429 |
| Gradient Boosting | 0.708167 | 0.714814 | 0.839935 | 0.770454 |
| Neural Networks | 0.702502 | 0.738492 | 0.808029 | 0.772752 |

TABLE IX: 2015-16 Quarter 3 with Cross Validation

| Methods's Name | Accuracy | Precision | Recall | ROC-AUC |
|-------------------|----------|-----------|----------|----------|
| Extra trees | 0.708200 | 0.727137 | 0.809551 | 0.772429 |
| Non-linear SVM | 0.702442 | 0.726247 | 0.797051 | 0.771752 |
| Gradient Boosting | 0.708167 | 0.714814 | 0.839935 | 0.770454 |

TABLE X: 2015-16 Half Season with Cross Validation and Feature Selection

| Methods's Name | Accuracy | Precision | Recall | ROC-AUC |
|-------------------|----------|-----------|----------|----------|
| Extra trees | 0.689251 | 0.698414 | 0.836637 | 0.758412 |
| Non-linear SVM | 0.692347 | 0.706659 | 0.822823 | 0.760219 |
| Gradient Boosting | 0.690786 | 0.701046 | 0.833934 | 0.757680 |

TABLE XI: 2015-16 Half Season with Cross Validation

| Methods's Name | Accuracy | Precision | Recall | ROC-AUC |
|-------------------|----------|-----------|----------|----------|
| Neural Networks | 0.695600 | 0.711003 | 0.806381 | 0.760346 |
| Non-linear SVM | 0.692347 | 0.706659 | 0.822823 | 0.760218 |
| Gradient Boosting | 0.690786 | 0.701046 | 0.833934 | 0.757680 |

1) *Classification Breakdown:* The results of our analysis can be seen in the tables of the prior sub-section. As you can see, the the ranking of accuracy for the model set is changes by the data scope. For example, logistic regression is the most accurate for the full games but Neural-Nets are much better at predicting game outcomes when trained on first quarter data. The full collection of confusion matrices, ROC curves, and learning curves can be found in our GitHub repository and provided upon request. Below, for reference, are the confusion matrix (Figure 2), ROC curve (Figure 3), and learning curve

(Figure 4) are provided for neural nets focused on first quarter data of the 2015-2016 season.

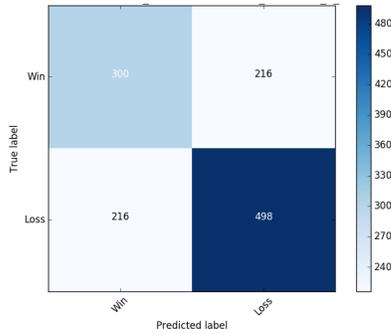


Fig. 2: Neural Networks Confusion Matrix. Quarter 1 of 2015-2016 Season

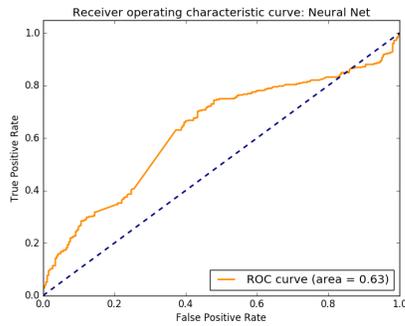


Fig. 3: Neural Networks ROC Curve. Quarter 1 of 2015-2016 Season.

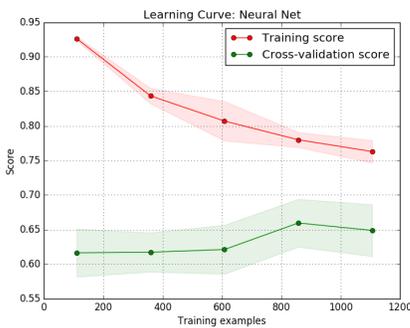


Fig. 4: Neural Networks Learning Curve. Quarter 1 of 2015-2016 Season.

C. Cross-Validation

For cross-validation we preceded with 10-fold validation, per the norm. This lead to the analysis of a little more than 8 different games per each subset.

V. DISCUSSION

Our model construction and analysis shows that optimal model selection depends on the intent of data training, even

if it is determined for just length of game. The model that works the best at predicting game outcome when trained on whole game data is different than the best model trained on first quarter data. This is possibly due to the unique character each quarter takes in with respect to game outcome, favoring the character of different classification techniques. Furthermore, training on a quarter as compared to a whole game can yield better results. This latter finding may be due to the fact that the data of other quarters is contradicting in aggregate, or that full game data actually leads to over-fitting. Overall, the highest accuracy of any model or data set was SVM with linear kernel on second quarter data with an accuracy of predicting whether or not the home team wins 71 % for the 2015-2016 regular season. We have determined based on the simple splitting nature of the model that favored the divergence of performance character of professional basketball. That was especially visible when analysing the second quarter results.

VI. CONCLUSION

Our efforts analyzed many different data parsings to train on across several classifiers, the best sifting out to be an SVM with a linear kernel using second quarter data with an accuracy of predicting game outcome at 71%. Unfortunately this fails to beat the publish neural network paper [3] who could predict at 74%. This gap can be explained by our lack of complexity in algorithm in comparison to the numerous algorithms tested, then strung together in the more successful paper.

VII. FUTURE WORK

Future work could include several variations of data parsing to pass into our successful algorithms. These could include exploring the value older data (the NBA API goes all the way back to the 1984-85 season) to develop team and player based clusters impacting the game, incorporating live data in the data of prior expectation, or focusing on a more granular look at player-based features. An exciting alternate approach could be to cluster game data to allow prediction of a game by determining its similarity to past games. Finally, a visually interesting development would be to develop a near-continuous time based prediction of a game, accumulating the data as a game goes on. This could show how each minute passes, the event effect the probability of game outcome.

REFERENCES

- [1] Renator Amorim Torres. Prediction of nba games based on machine learning methods. University of Wisconsin, Madison, 2013.
- [2] Chenjie Cao. Sports data mining technology used in basketball outcome prediction. Master's thesis, Dublin Institute of Technology, 2012.
- [3] Kenneth W. Bauer Bernard Loeffelholz, Earl Bednar. Predicting nba games using neural networks. *Journal of Quantitative Analysis in Sports*, 2009.
- [4] <http://www.espn.com>.
- [5] Four factors. website: <http://www.basketball-reference.com/about/factors.html>.
- [6] <http://stats.nba.com>.
- [7] *Exploiting tree-based variable importances to selectively identify relevant variables*, 2008.

- [8] Damien; Whenkel Louis Geurts, Pierre; Ernst. Extremely randomized trees. *MAchine Learning*, 2006.
- [9] C. F. Shannon and W. Weaver. *Mathematical Theory of Communication*. University of Illinois Press, 1949.
- [10] Kaushik Roy Prajesh P Anchalia. The k-nearest neighbor algorithm using mapreduce paradigm. *International Conference on Intelligent Systems, Modelling and Simulation*, 2014.
- [11] Frikha M. Chtourou M. Masmoudi, S. Efficient mlp constructive training algorithm using a neuron recruiting approach for isolated word recognition system. *International Journal of Speech Technology*, 2011.
- [12] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 2001.
- [13] *European workshop on QTL mapping and marker assisted selection (QTL-MAS)*, 2011. A comparison of random forests, boosting and support vector machines for genomic selection.
- [14] Francis Bach Mark Schmidt, Nicolas Le Roux. Minimizing finite sums with the stochastic average gradient. *HAL*, 2016.
- [15] Andrew Ng. Support vectorm machines. CS229 Course Lecture Notes.
- [16] Hwanjo Yu and Sungchul Kim. Svm tutorial: Classification, regression, and ranking.